4rd Workshop on Current Trends in Cryptology (CTCrypt 2015)

June 3-5, 2015, Kazan, Russia. Pre-proceedings

Program and Table Of Contents

Program Co-chairs

Vladimir Sachkov	Academy of Cryptography, Russia
Alexey Kuzmin	TC 26, Russia

Program Committee

Farid Ablaev	Kazan (Volga region) Federal University, Russia
Mikhail Glukhov	Academy of Cryptography, Russia
Andrei Zubkov	Steklov Mathematical Institute of RAS, Russia
Igor Kachalin	TC 26, Russia
Anatoly Lunin	TC 26, Russia
Grigory Marshalko	TC 26, Russia
Dmitry Matyukhin	TC 26, Russia
Aleksandr Nechaev	Lomonosov Moscow State University, Russia
Andrei Pichkur	Educational and Methodical Association of Higher Educational Institutions of Russia on Education in Information Security, Russia
Bart Preneel	Catholic University of Leuven, Belgium
Eduard Primenko	Lomonosov Moscow State University, Russia
Markku-Juhani Olavi Saarinen	Norwegian University of Science and Technology, Norway
Yury Kharin	Research Institute for Applied Problems of Mathematics and Informatics, Belarus
Aleksandr Shoitov	Moscow State Institute of Radio-Engineering, Electronics and Automation (Technical University), Russia

External Reviewers

Evgeniy Alekseev
Vladimir Anashin
Vladimir Antipkin
Mikhail Cherepnev
Ivan Chizhov
Oleg Kamlovskiy
Oleg Kozlitin
Elena Kutyreva
Sergey Krendelev
Ivan Lavrikov
Oleg Lobastov
Oleg Logachev
Sergey Molotkov
Maxim Nikolaev
Vasiliy Nikolaev
Pascal Paillier
Alexey Pokrovskiy
Alexander Rybakov
Stanislav Smishlyaev
Robert Špalek
Andrey Trishin
Mikhail Tuzhilin
Nikolay Varnovsky
Artem Volgin
Mansur Ziatdinov

Parsimonious Models of High-order Markov Chains for Evaluation of Cryptographic Generators

Yuriy Kharin

Invited talk

Abstract

We propose parsimonious (small-parametric) models of high-order Markov chains that are determined by small number of parameters and used for modeling of output sequences in cryptographic generators and their blocks. The paper presents results on statistical identification (estimation of parameters and hypotheses testing by the observed output sequence) for the following parsimonious models: Jacobs – Lewis model, Raftery MTD model, Markov chain with partial connections, Markov chain of conditional order.

Keywords: cryptograhic generator, output sequence, high-order Markov chain, parsimonious model, statistical identification.

1 Introduction

Cryptographic generators are necessary elements for cryptographic systems of information protection [1]. Classification of cryptographic generators on their construction and implementation can be find in [9]. To evaluate cryptographic security of generators two approach are known: 1) algebraic approach based on construction of some algebraic model for generator [1]; 2) stochastic approach based on construction of some probabilistic model for generator [15, 16, 17]. Modern cryptographic generators have a complex structure, and algebraic approach often can be found not applicable. In these situations stochastic approach considered in this paper can be used. Following to this approach the probabilistic model for the output sequence of the considered generator (or its block) is constructed and used to predict future elements of the output sequence [7].

Under stochastic approach the general model of the observed output sequence (of the cryptographic generator or its block) is the discrete time series ($\equiv$ discrete valued random sequence) $x_t = x_t(\omega) : \mathbb{N} \times \Omega \to \mathcal{A}$ defined on the probability space $(\Omega, \mathcal{F}, \mathbf{P})$, where $t \in \mathbb{N} = \{1, 2, \ldots\}$ is discrete time, $\mathcal{A} = \{0, 1, \ldots, N-1\}$ is a finite set of N states, $2 \le N < +\infty$.

Following probabilistic models are known for evaluation of cryptographic security: $x_t = \xi_t$ is a scheme of independent Bernoulli trials, $\mathbf{P}\{\xi_t = 1\} = 1 - \mathbf{P}\{\xi_t = 0\} = p$, $\mathcal{A} = V = \{0, 1\}$, $p \in [0, 1]$; x_t is a "noised periodic sequence": $x_t = a_t \oplus \xi_t$, where a_t is some "deterministic" periodic sequence (e. g. a sequence generated by some LFSR or NLFSR); modification of the above-mentioned models using the Markovian model for ξ_t. A review of models is given in [16].

Modern cryptographic generators guarantee the uniform distribution of s-gram in the output sequence for sufficiently large values of s, and to construct a useful probabilistic model we need to exploit high-order s dependencies in x_t. An universal model for long-memory discrete time series is the high-order Markov chain [3, 5]. Unfortunately, the number of parameters for the s-order Markov chain $D = N^s(N-1)$ increases exponentially with respect to the order s, and identification of this model is a computationally hard problem; in addition, we need to have data sets of huge size $T > D$. This situation generates a topical problem of construction and statistical analysis of small-parametric (parsimonious) models for high-order Markov chains, i. e. the models determined by small number of parameters. Some of these models are: the Jacobs – Lewis model [6], the Mixture Transition Distribution model [14], the hidden Markov model [2], the variable length Markov chain [4], Markov chain of the order s with r partial connections [11], Markov chain of conditional order [10].

This paper is devoted to identification (estimation of parameters and hypotheses testing) of the parsimonious models by the observed output sequence. The paper has the following structure. After Introduction (Section 1) we present our results on identification of some known in the literature parsimonious models of high-order Markov chains: for the Jacobs – Lewis model JL(s) in Section 2 and for the Raftery MTD model in Section 3. Section 4 and Section 5 are devoted to new parsimonious models: Markov chain MC(s, r) of order s with r partial connections and Markov chain of conditional order respectively.

2 Identification of the Jacobs – Lewis model JL(s)

Let us remind [5] that the homogeneous Markov chain of order $s \in \mathbb{N}$ is determined by the generalized Markov property ($t > s$; $i_1, \ldots, i_t \in \mathcal{A}$):

$$\begin{gathered}\mathbf{P}\{x_t = i_t | x_{t-1} = i_{t-1}, \ldots, x_1 = i_1\} = \\ = \mathbf{P}\{x_t = i_t | x_{t-1} = i_{t-1}, \ldots, x_{t-s} = i_{t-s}\} = p_{(i_{t-s}, \ldots, i_{t-1}), i_t},\end{gathered} \tag{1}$$

where $P = \left(p_{i_1, \ldots, i_s, i_{s+1}}\right)$ is an $(s+1)$-dimensional matrix of one-step transition probabilities satisfying the normalization condition:

$$\sum_{i_{s+1} \in \mathcal{A}} p_{i_1, \ldots, i_s, i_{s+1}} \equiv 1,\ i_1, \ldots, i_s \in \mathcal{A}.$$

Jacobs – Lewis model JL(s) is defined [6] by the following stochastic difference equation of order $s \geq 2$ with random delay:

$$x_t = \mu_t x_{t-\eta_t} + (1 - \mu_t)\xi_t,\ t > s, \tag{2}$$

where $\{\xi_t, \eta_t, \mu_t\}$ are jointly independent random variables with probability distributions:

$$\begin{gathered}\mathbf{P}\{\mu_t = 1\} = 1 - \mathbf{P}\{\mu_t = 0\} = \rho;\ \mathbf{P}\{\xi_t = k\} = \pi_k,\ k \in \mathcal{A},\ \textstyle\sum_{k \in \mathcal{A}} \pi_k = 1; \\ \mathbf{P}\{\eta_t = j\} = \lambda_j,\ j \in \{1, 2, \ldots, s\},\ \textstyle\sum_{j=1}^{s} \lambda_j = 1,\ \lambda_s \neq 0;\end{gathered} \tag{3}$$

initial random values $x_1, \ldots, x_k$ are independent with probability distribution $\mathbf{P}\{x_1 = k\} = \ldots = \mathbf{P}\{x_s = k\} = \pi_k$, $k \in \mathcal{A}$. Number of parameters $\pi = (\pi_k)$, $\lambda = (\lambda_i)$, ρ depends linearly on s: $D_{\text{JL}} = N + s - 1$.

Probabilistic model (2), (3) of generation of randome sequence x_t is illustrated by Fig. 1. The generator consists of three elementary generators

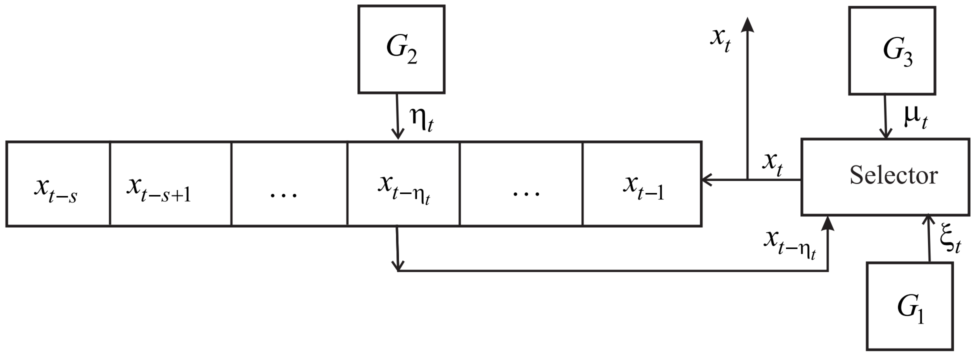

Fig. 1. Generation of x_t by JL(s) model

(G_1 for generation of ξ_t, G_2 for generation of η_t and G_3 for generation of binary sequence μ_t), a shift register and a selector that selects one of its input signals $x_{t-\eta_t}$, ξ_t depending on the value μ_t.

Theorem 1. *The discrete time series x_t determined by (2), (3) is a homogeneous Markov chain of the order s with the initial probability distribution $\pi_{i_1,\dots,i_s} = \pi_{i_1} \cdot \ldots \cdot \pi_{i_s}$ and the $(s+1)$-dimensional matrix of transition probabilities $P(\pi,\, \lambda,\, \rho) = \left(p_{i_1,\dots,i_{s+1}}\right)$:*

$$p_{i_1,\dots,i_s,i_{s+1}} = (1-\rho)\pi_{i_{s+1}} + \rho \sum\nolimits_{j=1}^{s} \lambda_j \mathbf{I}\{i_{s+1} = i_{s-j+1}\},\ i_1,\, \dots,\, i_{s+1} \in \mathcal{A}, \tag{4}$$

where $\mathbf{I}\{B\}$ means indicator of event B.

Proof. The generalized Markov property (1) follows from the definition (2), (3) of the JL(s) model: $x_t = f\left(x_{t-1},\, \dots,\, x_{t-s};\, \xi_t,\, \eta_t,\, \mu_t\right)$, where the function $f(\cdot)$ is determined by (2). By (3) and the total probability formula we have:

$$p_{i_1,\dots,i_s,i_{s+1}} = (1-\rho)\pi_{i_{s+1}} + \rho \sum\nolimits_{j=1}^{s} \mathbf{P}\{\eta_{s+1} = j\} \times$$
$$\times \mathbf{P}\{x_{s+1} = i_{s+1} | x_s = i_s,\, \dots,\, x_1 = i_1,\, \mu_{s+1} = 1,\, \eta_{s+1} = j\} =$$
$$= (1-\rho)\pi_{i_{s+1}} + \rho \sum\nolimits_{j=1}^{s} \lambda_j \mathbf{I}\left\{i_{s+1} = i_{s-j+1}\right\}.$$

Initial probability distribution follows from the model assumption. □

Corollary 1. *Maximum likelihood estimators (MLEs) $\hat{\pi}$, $\hat{\lambda}$, $\hat{\rho}$ for the parameters π, λ, ρ of the JL(s) model by the observations $X_T = (x_1,\, \dots,\, x_T)' \in$*

$\mathcal{A}^T$ *are determined by maximization of the following loglikelihood function* $l = l(\pi,\, \lambda,\, \rho)$*:*

$$l = \sum\nolimits_{t=1}^{s} \ln \pi_{x_t} + \sum\nolimits_{t=s+1}^{T} \ln \left((1-\rho)\pi_{x_t} + \rho \sum\nolimits_{j=1}^{s} \lambda_j \mathbf{I}\{x_t = x_{t-j}\} \right) \to \max_{\pi,\,\lambda,\,\rho} . \tag{5}$$

In [12] consistent estimators $\tilde{\pi},\ \tilde{\lambda},\ \tilde{\rho}$ are found and used as the initial approximation for the MLEs $\hat{\pi},\ \hat{\lambda},\ \hat{\rho}$ in the iterative solution of the maximization problem (5).

Define the hypotheses $H_0,\ H_1$ on the values of the parameters: $H_0 = \{\pi = \pi^0,\, \lambda = \lambda^0,\, \rho = \rho^0\}$, where $\pi^0,\ \lambda^0,\ \rho^0$ are some fixed hypothetical values for parameters (e. g., if $\rho^0 = 0$, $\pi_k^0 \equiv N^{-1}$ hypothesis H_0 means that x_t is uniformly distributed random sequence); $H_1 = \bar{H}_0$. Using the asymptotic normality property of the MLEs $\hat{\pi},\ \hat{\lambda},\ \hat{\rho}$ we get [12] the generalized probability ratio test of the asymptotic size $\varepsilon \in (0,\, 1)$:

$$d = d\left(X_T\right) = \mathbf{I}\{\Lambda_T \geq \Delta_\varepsilon\},\ \Lambda_T = 2\left(l\left(\hat{\pi},\, \hat{\lambda},\, \hat{\rho}\right) - l\left(\pi^0,\, \lambda^0,\, \rho^0\right) \right), \tag{6}$$

where Δ_ε is ε-quantile of the χ^2_{N+s-1}-distribution.

3 Identification of the Raftery MTD (Mixture Transition Distribution) model

Raftery MTD model [14] is defined by a special small-parametric (parsimonious) representation of the matrix P:

$$p_{i_1,\ldots,i_{s+1}} = \sum\nolimits_{j=1}^{s} \lambda_j q_{i_j,\, i_{s+1}}, \quad i_1,\, \ldots,\, i_{s+1} \in \mathcal{A}, \tag{7}$$

where $Q = (q_{ik})$ is a stochastic $(N \times N)$-matrix, $i,\, k \in \mathcal{A}$, and $\lambda = (\lambda_1,\, \ldots,\, \lambda_s)'$ is an s-column vector such that $\lambda_1 > 0,\ \lambda_2,\, \ldots,\, \lambda_s \geq 0$, $\lambda_1 + \ldots + \lambda_s = 1$. This model has $D_{\mathrm{MTD}} = N(N-1)/2 + s - 1$ parameters. The MTD model (7) can be generalized to obtain the MTDg model:

$$p_{i_1,\ldots,i_{s+1}} = \sum\nolimits_{j=1}^{s} \lambda_j q^{(j)}_{i_j,\, i_{s+1}}, \quad i_1,\, \ldots,\, i_{s+1} \in \mathcal{A}, \tag{8}$$

where $Q^{(j)} = \left(q_{ik}^{(j)}\right)$ is the jth stochastic matrix corresponding to the time lag $s-j$. The number of parameters in the MTDg model is $D_{\mathrm{MTDg}} = s(N(N-1)/2+1)-1$.

Let us introduce the notation: the distribution $\Pi^* = \left(\pi^*_{i_1,\ldots,i_s}\right)$, $i_1,\ldots,$ $i_{s+1}\in\mathcal{A}$ is an s-variate stationary probability distribution of the ergodic Markov chain; $\pi^*=\left(\pi^*_0,\ldots,\pi^*_{N-1}\right)'$ is a univariate stationary probability distribution; δ_{ij} is Kronecker symbol.

Theorem 2. *Under model (8), if for some $K\in\mathbb{N}$ every element of the matrix $\left(Q^{(1)}\right)^K$ is positive, the stationary probability distribution Π^* satisfies the equation:*

$$\pi^*_{i_1,\ldots,i_s} = \prod_{l=0}^{s-1}\left(\pi^*_{i_{s-l}} + \sum_{j=l+1}^{s}\lambda_j\left(q^{(j)}_{i_{j-l},i_{s-l}} - \sum_{r=0}^{N-1} q^{(j)}_{r,i_{s-l}}\pi^*_r\right)\right),$$
$$i_1,\ldots,i_{s+1}\in\mathcal{A}.$$

Corollary 2. *Under the model (7) the stationary bivariate marginal probability distribution of the random vector $(x_{t-m},x_t)'$ satisfies the relation $\pi^*_{ki}(m)=\pi^*_k\pi^*_i+\pi^*_k\lambda_{s-m+1}\left(q_{ki}-\pi^*_i\right)$, $1\le m\le s$, $k\in\mathcal{A}$.*

The proof of Theorem 2 and its corollary can be find in [12].

Let us construct estimators for the parameters of the MTD model by applying the property from Corollary 2. From the observed realization $X_1^T = (x_1,\ldots,x_T)'$ define the following statistics for $i,k\in\mathcal{A}$, $j=1,\ldots,s$:

$$\tilde\pi_i = (T-2s+1)^{-1}\sum_{t=s+1}^{T-s+1}\delta_{x_t,i};\quad \tilde\pi_{ki}(j) = (T-2s+1)^{-1}\sum_{t=s+j}^{T-s+j}\delta_{x_{t-j},k}\delta_{x_t,i};$$

$$\tilde q_{ki} = \begin{cases}\sum_{j=1}^{s}\tilde\pi_{ki}(j)/\tilde\pi_k-(s-1)\tilde\pi_i, & \tilde\pi_k>0,\\ 1/N, \text{ otherwise;}\end{cases}\tag{9}$$

$$z_{ki}(j) = \tilde\pi_{ki}(s-j)/\tilde\pi_k-\tilde\pi_i;\ d_{ki}=\tilde q_{ki}-\tilde\pi_i;\ \tilde\lambda =$$
$$=\arg\min_{\lambda}\sum_{i,k\in\mathcal{A}}\sum_{j=1}^{s}\left(z_{ki}(j)-\lambda_j d_{ki}\right)^2.$$

Theorem 3. *Under Corollary 2 conditions the statistics (9) are asymptotically unbiased and consistent estimators for Q and λ as $T\to\infty$.*

Proof. It is easy to show that the definitions of consistency and asymptotic unbiasedness are satisfied. □

The estimators $\tilde{Q}$, $\tilde{\lambda}$ defined by (9) give a good initial approximation for iterative maximization of the loglikelihood function, which yields the MLEs $\hat{Q}$, $\hat{\lambda}$: $l(Q, \lambda) = \sum_{t=s+1}^{T} \ln \sum_{j=1}^{s} \lambda_j q_{x_{t-s+j-1}, x_t} \to \max\limits_{Q, \lambda}$.

Generalized probability ratio test of the asymptotic ($T \to \infty$) size $\varepsilon \in (0, 1)$ for testing hypotheses (on the values of parameters Q, λ): $H_0 = \{Q = Q^0, \lambda = \lambda^0\}$, $H_1 = \bar{H}_0$ is constructed as test (6) in the previous section for JL(s) model.

4 Identification of the MC(s, r) model

Introduce the notation: $r \in \{1, \ldots, s\}$ is the parameter called the number of partial connections; $M_r^0 = \left(m_1^0, \ldots, m_r^0\right) \in M$ is an arbitrary integer r-vector with ordered components $1 = m_1^0 < m_2^0 < \cdots < m_r^0 \leq s$ which is called the connection template; M is the set of cardinality $K = |M| = C_{s-1}^{r-1}$ which is composed of all possible connection templates with r partial connections; and $Q^0 = \left(q^0_{j_1, \ldots, j_{s+1}}\right)$ is some $(r+1)$-dimensional stochastic matrix, $j_1, \ldots, j_{s+1} \in \mathcal{A}$.

A Markov chain of order s with r partial connections [9, 11], denoted as MC(s, r), is defined by specifying the one-step transition probabilities:

$$p_{i_1, \ldots, i_s, i_{s+1}} = q^0_{i_{m_1^0}, \ldots, i_{m_r^0}, i_{s+1}}, \; i_1, \ldots, i_{s+1} \in \mathcal{A}. \tag{10}$$

The relation (10) implies that the probability of the process entering a state i_{s+1} at time $t > s$ does not depend on every previous state of the process $i_1, \ldots, i_s$, but is affected only by the r chosen states $i_{m_1^0}, \ldots, i_{m_r^0}$. Thus, instead of $D = N^s(N-1)$ parameters, the model (10) is defined by $D_{\mathrm{MC(s,\,r)}} = N^r(N-1) + r - 1$ independent parameters that determine the matrices Q^0, M_r^0. The reduction in the number of parameters can be very significant: for instance, if $N = 2$, $s = 32$, $r = 3$, then we have $D \approx 4.1 \cdot 10^9$, and $D_{\mathrm{MC}(32,\,3)} = 10$.

Note that if $s = r$, $M_r^0 = (1, \ldots s)$, then $P = Q^0$, and MC(s, s) is a Markov chain of order s. A constructive example of MC(s, r) for modeling of output sequences generated by cryptographic devices and their blocks is a binary ($N = 2$) autoregression of order s with r nonzero coefficients, a special case of which is a linear recursive sequence defined in the ring $\mathbb{Z}_2$ and generated by a degree s polynomial with r nonzero coefficients [13].

Introduce the notation: $J_s = (j_1, \dots, j_s) = (J_{s-1}, j_s) \in \mathcal{A}^s$ is a multi-index of order s; the function S_t:

$$\mathcal{A}^T \times M \to \mathcal{A}^r,\ (X_T; M_r) \to (x_{t+m_1-1}, \dots, x_{t+m_r-1}) \in \mathcal{A}^r$$

is called a selector of order r with parameters $M_r \in M$ and $t \in \{1, \dots, T - s + 1\}$; $\Pi_{K_s} = \mathbf{P}\{X_s = K_s\}$ is the initial s-variate probability distribution of the Markov chain MC(s, r); $\Pi^*_{K_s}$ is the stationary distribution; $\nu_{r+1}(J_{r+1}; M_r) = \sum_{t=1}^{T-s} \mathbf{I}\{S_t(X_T; M_{r+1}) = J_{r+1}\}$ is the frequency of the $(r + 1)$-gram $J_{r+1} \in \mathcal{A}^{r+1}$ corresponding to the connection template $M_{r+1} = (M_r, s + 1)$ and satisfying the normalization condition

$$\sum\nolimits_{J_{r+1} \in \mathcal{A}^{r+1}} \nu_{r+1}(J_{r+1}; M_r) = T - s;$$

an index replaced by a dot denotes summation over all values of this index:

$$\nu_{r+1}(J_r \cdot; M_r) = \sum\nolimits_{j_{r+1} \in \mathcal{A}} \nu_{r+1}(J_{r+1}; M_r),\ \nu_{r+1}(\cdot j_{r+1}) = \sum\nolimits_{J_r \in \mathcal{A}^r} \nu_{r+1}(J_{r+1}; M_r).$$

Theorem 4. *The model MC(s, r) defined by (10) is ergodic if and only if there exists an integer $l \geq 0$ such that*

$$\min_{J_s, J'_s \in \mathcal{A}^s} \sum\nolimits_{K_l \in \mathcal{A}^l} \prod_{i=1}^{s+l} q^0_{S_i\left((J_s, K_l, J'_s); M^0_{r+1}\right)} > 0.$$

The proof is based on transforming MC(s, r) into a special s-vector Markov chain of order one.

Let us apply the plug-in principle to construct the information functional $\hat{I}_{r+1}(M_r)$ from the observed realization X_T. In other words let us construct a sample-based estimator for the Shannon information on the future symbol $x_{t+s} \in \mathcal{A}$ contained in the r-gram $S_t(X_T, M_r)$.

Theorem 5. *The maximum likelihood estimators $\hat{M}_r$, $\hat{Q} = \left(\hat{q}_{J_{r+1}}\right)$, where $J_{r+1} \in \mathcal{A}^{r+1}$, for the parameters M_r^0, Q^0 can be defined as*

$$\hat{M}_r = \arg\max\nolimits_{M_r \in M} \hat{I}_{r+1}(M_r),$$

$$\hat{q}_{J_{r+1}}(M_r) = \begin{cases} \nu_{r+1}(J_{r+1};\, \hat{M}_r)/\nu_{r+1}(J_r\cdot;\, \hat{M}_r), & \textit{if } \nu_{r+1}(J_r\cdot;\, \hat{M}_r) > 0, \\ 1/N, & \textit{if } \nu_{r+1}(J_r\cdot;\, \hat{M}_r) = 0. \end{cases} \tag{11}$$

Theorem 6. *If MC$(s,\, r)$ defined by (10) is stationary and the connection template $M_r^0 \in M$ satisfies the identifiability condition, then the maximum likehood estimators $\hat{M}_r$, $\hat{Q}$ defined by (11) are consistent for $T \to \infty$:*

$$\hat{M}_r \xrightarrow{\mathbf{P}} M_r^0, \quad \hat{Q} \xrightarrow{\mathbf{L}_2} Q^0,$$

and the following asymptotic expansion holds for the mean square error of $\hat{Q}$:

$$\Delta_T^2 = \mathbf{E}\left\{\|\hat{Q} - Q^0\|^2\right\} = \frac{1}{T-s} \sum\nolimits_{J_{r+1} \in \mathcal{A}^{r+1}} \frac{\left(1 - q_{J_{r+1}}^0\right) q_{J_{r+1}}^0}{\mu_{r+1}\left(J_r\cdot;\, M_r^0,\, M_r^0\right)} + o\left(\frac{1}{T}\right),$$

$$\mu_{r+1}\left(J_{r+1}; M_r, M_r^0\right) = \sum\nolimits_{K_{s+1} \in \mathcal{A}^{s+1}} \mathbf{I}\left\{S_1\left(K_{s+1}; M_{r+1}\right) = J_{r+1}\right\} \Pi_{K_s}^* p_{K_{s+1}}. \tag{12}$$

Theorems 5, 6 have been proved in [9].

The estimators (11) have been used to construct a statistical test for a null hypothesis $H_0 : Q^0 = Q_0$ against the alternative $H_1 = \bar{H}_0$, where $Q_0 = \left(q_{0J_{r+1}}\right)$ is some given stochastic matrix. The decision rule of a given asymptotic size $\varepsilon \in (0,\, 1)$ can be written as follows [9]:

$$d\left(X_T\right) = \mathbf{I}\{\rho > \Delta\},\ \rho = \sum\nolimits_{J_{r+1}:\, q_{0J_{r+1}} > 0} \nu_{r+1}(J_r\cdot; \hat{M}_r) \left(\hat{q}_{J_{r+1}} - q_{0J_{r+1}}\right)^2 / q_{0J_{r+1}} \tag{13}$$

where Δ is the $(1-\varepsilon)$-quantile of the χ^2 distribution with L degrees of freedom.

Note that to estimate orders $s \in [s_-, s_+]$, $r \in [r_-, r_+]$ $(1 \le s_- < s_+ < \infty$, $1 \le r_- < r_+ < s_+)$ we use a modification of the Bayesian Information Criterion (BIC) presented in [8].

Performance of the statistical estimators (12) and the test (13) was evaluated by Monte-Carlo simulation experiments, where the model parameters were fixed: $N = 2$, $s = 256$, $r = 6$; the chosen values of Q^0 and M_r^0 are omitted due to space limitation. For each simulated observation length T, 10^4 simulation rounds were performed.

Figure 2 illustrates the numerical results obtained for this MC(256, 6) model: the mean square error Δ_T^2 of the estimator $\hat{Q}$ is plotted against the observation length T; the curve has been computed theoretically from the leading term of the expansion (12), and the circles are the experimental values computed in the simulations.

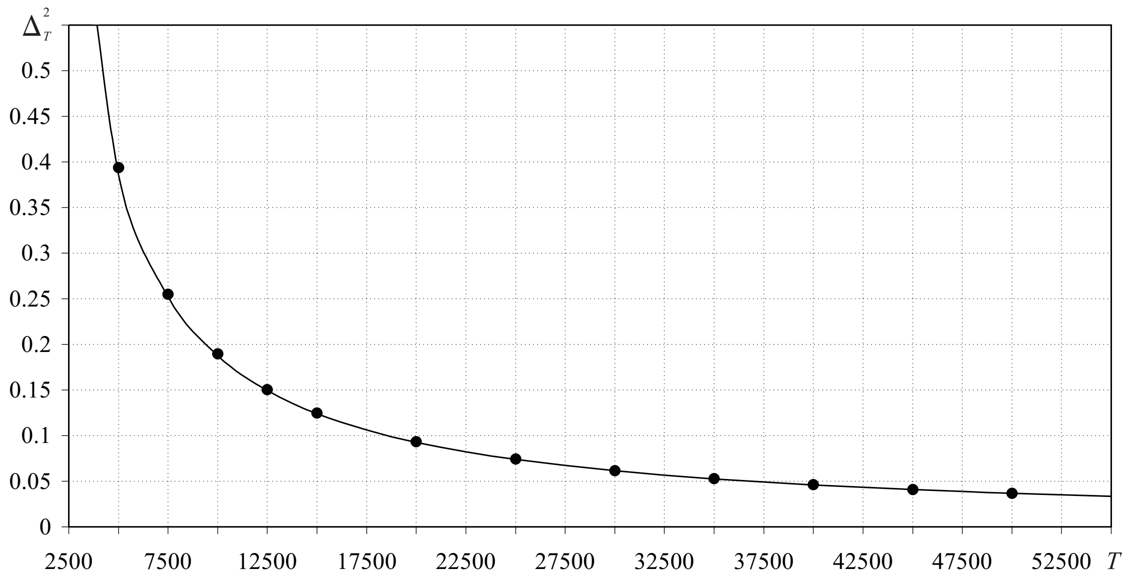

Fig. 2. Dependence of Δ_T^2 on T for $N = 2$, $s = 256, r = 6$

5 Markov chain of conditional order and its identification

Introduce the notation: $J_m^n = (j_m, j_{m+1}, \ldots, j_n) \in \mathcal{A}^{n-m+1}$, $n \geq m$, is the miltiindex (subsequence of indices from a sequence $j_1, j_2, \ldots$);

$$<J_n^m> = \sum_{k=n}^{m} N^{k-n} j_k \in \{0, 1, \ldots, N^{m-n+1} - 1\}$$

is the numeric representation of the multiindex $J_n^m \in \mathcal{A}^{m-n+1}$; $L \in \{1,\ 2, \ldots,\ s-1\}$, $K = N^L - 1$ are some positive integers; $Q^{(1)}, \ldots, Q^{(M)}$ are M $(1 \leq M \leq K+1)$ different square stochastic matrices of the order N: $Q^{(m)} = \left(q_{i,j}^{(m)}\right)$, $0 \leq q_{i,j}^{(m)} \leq 1$, $\sum_{j \in \mathcal{A}} q_{i,j}^{(m)} \equiv 1$, $i, j \in \mathcal{A}$, $1 \leq m \leq M$; $\pi_{J_1^s}^0 = \mathbf{P}\{X_1^s = J_1^s\}$ is the initial probability distribution.

The Markov chain $\{x_t \in \mathcal{A} : t \in \mathbb{N}\}$ is called the Markov chain of conditional order (MCCO(s)), if its one-step transition probabilities have the following parsimonious form [10]:

$$p_{J_1^{s+1}} = \sum_{k=0}^{K} \mathbf{I}\{< J_{s-L+1}^s >= k\}\, q_{j_{b_k}, j_{s+1}}^{(m_k)}, \tag{14}$$

where $1 \leq m_k \leq M$, $1 \leq b_k \leq s - L$, $0 \leq k \leq K$, $\min\limits_{0 \leq k \leq K} b_k = 1$; it is assumed that all elements of the set $\{1,\ 2,\ \ldots,\ M\}$ occur in the sequence $m_0, \ldots, m_K$. The sequence of elements J_{s-L+1}^s is called the base memory fragment (BMF) of the random sequence, L is the length of BMF; the value $s_k = s - b_k + 1$ is called the conditional order. Thus the conditional probability distribution of the state x_t at time t depends not on all s previous states, but it depends only on $L+1$ selected states $\left(j_{b_k},\ J_{s-L+1}^s\right)$. Note that if $L = s-1$, $s_0 = s_1 = \cdots = s_K = s$, we have the fully-connected Markov chain of the order s. If $M = K+1$, then each transition matrix corresponds to only one value of BMF, otherwise there exists a common matrix which corresponds to several values of BMF. The number of independent parameters $D_{\text{MCCO}(s)} = 2\left(N^L + 1\right) + MN(N-1)$.

As in previous section we construct the MLEs $\left\{\hat{Q}^{(i)}\right\}$, $\hat{L}$, $\{\hat{s}k\}$, $\{\hat{m}_k\}$ by maximization the loglikelihood:

$$l_T\left(X_T,\,\left\{Q^{(i)}\right\},\,L,\,\{s_k\},\,\{m_k\}\right)=\ln\pi^0_{X_1^s}+$$
$$+\sum\nolimits_{J_0^{L+1}\in\mathcal{A}^{L+2}}\sum\nolimits_{k=0}^{K}\mathbf{I}\left\{<J_1^L>=k\right\}\nu^s_{L+2,\,s_k-L-1}\left(J_0^{L+1}\right)\ln q^{(m_k)}_{j_0,\,j_{L+1}}\to\max,\tag{15}$$

where $\nu^s_{l,\,y}\left(J_1^l\right)=\sum_{t=1}^{n-s}\mathbf{I}\left\{x_{t+s-l-y+1}=j_1,\,X_{t+s-l+2}^{t+s}=J_2^l\right\},\,l\geq 2,\;0\leq y\leq s-l+1$, is frequency of the state $J_1^l\in A^l$ with the time gap of length y between the elements j_1 and J_2^l; $\nu_{s+1}\left(J_1^{s+1}\right)=\nu^s_{s+1,\,0}\left(J_1^{s+1}\right)$ is frequency of $(s+1)$-gram J_1^{s+1}.

From (15) we get the following expressions for the MLEs (for simplicity of expressions we give here results only for the case $M=K+1$) [10]:

$$\hat{q}^{(k+1)}_{j_0,\,j_{L+1}}=\begin{cases}\sum\nolimits_{J_1^L\in\mathcal{A}^L}\mathbf{I}\left\{<J_1^L>=k\right\}\dfrac{\nu^s_{L+2,\,g(s_k,\,L)}\left(J_0^{L+1}\right)}{\nu^s_{L+1,\,g(s_k,\,L)}\left(J_0^L\right)}, & \text{if } \nu^s_{L+1,\,g(s_k,\,L)}\left(J_0^L\right)>0,\\ 1/N, \quad \text{if } \nu^s_{L+1,\,g(s_k,\,L)}\left(J_0^L\right)=0;\end{cases}$$

$$\hat{s}_k=\arg\max_{L+1\leq y\leq s}\sum\nolimits_{J_1^L\in\mathcal{A}^L}\mathbf{I}\left\{<J_1^L>=k\right\}\sum\nolimits_{j_0,\,j_{L+1}\in\mathcal{A}}\nu^s_{L+2,\,g(y,\,L)}\left(J_0^{L+1}\right)\ln\left(\hat{q}^{(k+1)}_{j_0,\,j_{L+1}}\right).$$

To estimate the order s and the BMF length L we use Bayesian information criterion:

$$\left(\hat{s},\,\hat{L}\right)=\arg\min_{\substack{1\leq L'\leq L_+,\\ 2\leq s'\leq S_+}}\mathrm{BIC}\left(s',\,L'\right),\;\mathrm{BIC}(s',\,L')=-2\sum\nolimits_{J_0^{L'+1}\in\mathcal{A}^{L'+2}}\sum\nolimits_{k=0}^{K}$$

$$\mathbf{I}\left\{<J_1^{L'}>=k\right\}\times\times\nu^{s'}_{L'+2,\,\hat{g}(s_k,\,L')}\left(J_0^{L'+1}\right)\ln\hat{q}^{(k+1)}_{j_0,\,j_{L'+1}}+D_{\mathrm{MCCO(s)}}\ln(n-s'),$$

where $S_+\geq 2,\;1\leq L_+\leq S_+-1$, are maximal admissible values of s and L respectively.

Statistical decision rule for testing of hypotheses on the values of parameters for the MCCO(s) model is constructed [10] by MLEs $\left\{\hat{Q}^{(k)}\right\}$, $\{\hat{s}_k\}$ in the same way as in the previous section.

The author would like to thank Prof. A. M. Zubkov for fruitful discussions and comments.

References

[1] Alferov A., Zubov A., Kuzmin A., Cheremushkin A. Fundamentals of Cryptography. — Moscow: Gelios ARV, 2001. — 480 P. (in Russian).

[2] Berchtold A. High-order extensions of the double chain Markov model // Stochastic Models. — 2002. — V. 18. No 2. — P. 193-227.

[3] Billingsley P. Statistical methods in Markov chains // The Annals of Mathematical Statistics. — 1961. — V. 32. — P. 12-40.

[4] Buhlmann P., Wyner A. Variable length Markov chains // The Annals of Statistics. — 1999. — V. 27. No 2. — P. 480-513.

[5] Doob J. Stochastic Processes. — N.Y.: Wiley, 1953.

[6] Jacobs P. A., Lewis P. A. W. Discrete time series generated by mixtures I: correlational and runs properties // Journal of the Royal Statistical Society. Ser. B. — 1978. — V. 40. No 1. — P. 94-105.

[7] Kharin Yu. Robustness in Statistical Forecasting. — N.Y.: Springer, 2013. — 356 p.

[8] Kharin Yu. Statistical analysis of discrete time series based on the MC(s, r)-model // Austrian Journal of Statistics. — 2011. — V. 40. No 1&2. — P. 75-84.

[9] Kharin Yu., Agievich S., Vasiljev D., Matsveev G. Cryptology. — Minsk: BSU, 2013. — 512 P. (in Russian).

[10] Kharin Yu., Maltsew M. Markov Chain of Conditional Order: Properties and statistical analysis // Austrian Journal of Statistics. — 2014. — V. 43. No 3&4. — P. 205-217.

[11] Kharin Yu. S., Piatlitski A. I. A Markov chain of order s with r partial connections and statistical inference on its parameters // Discrete Mathematics and Applications. — 2007. — V. 17. No 3. — P. 295-317.

[12] Kharin Yu., Yarmola A. Statistical estimation of parameters for the MTD-model of discrete time series // Vesti of Nat. Acad. of Sci. of Belarus. — 2006. No 2. — P. 20-26. (in Russian).

[13] Maksimov Yu. I. On Markov chains in binary shift registors with random elements // Proceedings on Discrete Mathematics. — 1997. — V. 1. — P. 203-220 (in Russian).

[14] Raftery A. A model for high-order Markov chains // Journal of the Royal Statistical Society. — 1985. — Ser. B. V. 47. No 3. — P. 528-539.

[15] Sachkov V. N. Introduction to Combinatoric Methods of Discrete Mathematics. — Moscow: Nauka, 1982. — 489 P. (in Russian).

[16] Shojtov A. Probabilistic models of pseudorandom sequences in cryptography // In: Problems of IT-security. — Moscow: MSU, 2007. — P. 116-134 (in Russian).

[17] Zubkov A. M. Generators of pseudorandom numbers and their applications // In: Mathematics and IT-security. — Moscow: MSU, 2003. — P. 200-206 (in Russian).

A Graph of Minimal Distances of Bent Functions

Nikolay Kolomeec

Abstract

A notion of a graph of minimal distances of bent functions is introduced. It is an undirected graph $(V,\ E)$ where V is the set of all bent functions in $2k$ variables and $(f, g) \in E$ if the Hamming distance between f and g is equal to 2^k (it is the minimal possible distance between two bent functions). It is shown that its subgraph induced by all functions affine equivalent to Maiorana—McFarland bent functions is connected.

Keywords: Boolean functions, bent functions, the minimal distance

Introduction

In this work a graph of minimal distances of bent functions is considered. Bent functions are Boolean functions in even number of variables that have the maximal possible nonlinearity. They were proposed by O. Rothaus [7]. Bent functions have a lot of applications in algebra, combinatorics, coding theory, cryptography, see [5, 8, 9]. Here we prove that for any two bent functions f, g that are affine equivalent to Maiorana—McFarland bent functions there exist bent functions $f_0, \ldots, f_n$ (they are affine equivalent to Maiorana—McFarland bent functions too) such that $f = f_0$, $g = f_n$ and the Hamming distance between f_i and f_{i+1} is the minimal possible. This statement is also true for any bent functions f, g in a small number of variables (2, 4 and 6).

Let us give definitions. *A Boolean function* in n variables is a mapping $f : \mathbb{F}_2^n \to \mathbb{F}_2$. Denote by $\mathcal{F}_n$ the set of all Boolean functions in n variables. *The Hamming distance* between two Boolean functions f, g in n variables $\mathrm{dist}(f, g)$ is the number of $x \in \mathbb{F}_2^n$ such that $f(x) \neq g(x)$. Define by $\langle x, y\rangle = x_1y_1 \oplus x_2y_2 \oplus \ldots \oplus x_ny_n$ the inner product of two vectors $x, y \in \mathbb{F}_2^n$.

Restriction of a Boolean function $f \in \mathcal{F}_n$ on the set $S \subseteq \mathbb{F}_2^n$ is a mapping $f|_S : S \to \mathbb{F}_2$ where $f|_S(x) = f(x)$ for all $x \in S$. A Boolean function is called *affine* if its algebraic degree is not more than 1 and *quadratic* if its degree is equal to 2.

A shift $s \oplus D$ of the set $D \subseteq \mathbb{F}_2^n$, $s \in \mathbb{F}_2^n$, is the set $\{s \oplus x \ : \ x \in D\}$. A set $U \subseteq \mathbb{F}_2^n$ is *an affine subspace* of $\mathbb{F}_2^n$ if it is a shift of a linear subspace of $\mathbb{F}_2^n$. *Dimension* of an affine subspace is a dimension of a corresponding linear subspace.

A Boolean function $f \in \mathcal{F}_n$ is *affine on an affine subspace* L of $\mathbb{F}_2^n$ if $f|_L(x) = \langle a, x \rangle \oplus c$ for some $a \in \mathbb{F}_2^n$ and $c \in \mathbb{F}_2$.

Two Boolean functions $f, g \in \mathcal{F}_n$ are called *affinely equivalent* if there exist an invertible $n \times n$ binary matrix A and $b \in \mathbb{F}_2^n$ such that $g(x) = f(xA \oplus b)$ for any $x \in \mathbb{F}_2^n$. Note that $\mathrm{dist}(f, g) = \mathrm{dist}(f(xA \oplus b), g(xA \oplus b))$.

A Bent function is a Boolean function from $\mathcal{F}_{2k}$ that is at the maximal possible distance from the set of all affine functions in $2k$ variables.

The following functions form Maiorana—McFarland [6] class of bent functions $\mathcal{M}_{2k}$:

$$f(x, y) = \langle x, \pi(y) \rangle \oplus \varphi(y) \text{ where}$$

- $x, y \in \mathbb{F}_2^k$,
- π is a permutation on $\mathbb{F}_2^k$ and
- φ is an arbitrary Boolean function in k variables.

Denote by $\widetilde{\mathcal{M}}_{2k}$ the set of all bent functions affinely equivalent to functions from $\mathcal{M}_{2k}$ (this class is also called *completed* Maiorana—McFarland class). It is obvious that $f \oplus \ell \in \widetilde{\mathcal{M}}_{2k}$ for any $f \in \widetilde{\mathcal{M}}_{2k}$ and any affine function ℓ in $2k$ variables.

Denote by Ind_D, $D \subseteq \mathbb{F}_2^n$, a Boolean function in n variables that takes value 1 only on the set D. Denote by $\mathrm{supp}(f)$, $f \in \mathcal{F}_n$, the set $\{x \ : \ f(x) = 1, x \in \mathbb{F}_2^n\}$.

The minimal possible distance between two bent functions in $2k$ variables is equal to 2^k. There is the following construction of bent functions at the distance 2^k. Let f be a bent function in $2k$ variables and f be affine on a k-dimensional affine subspace L of $\mathbb{F}_2^{2k}$. Then $f \oplus Ind_L$ is also a bent function. This construction was proposed by C. Carlet [1]. Any bent function at the distance 2^k from f can be constructed in this way [4].

1 A graph of minimal distances of bent functions

An undirected graph $GB_{2k} = (V, E)$ is called *a graph of minimal distances of bent functions* if

- V is the set of all bent functions in $2k$ variables and
- $(f, g) \in E$ if and only if $\mathrm{dist}(f, g) = 2^k$.

Denote by GM_{2k} a subgraph of GB_{2k} induced by all vertices from $\widetilde{\mathcal{M}}_{2k}$. Summarize known facts in terms of GB_{2k} and GM_{2k}.

- The maximal degree of a vertex is equal to $2^k(2^1+1)(2^2+1)\ldots(2^k+1)$, any vertex of the maximal degree is a quadratic bent function. It is true for both GB_{2k} and GM_{2k}, see [3].
- Degree of a vertex of GM_{2k} is not less than $2^{2k+1} - 2^k$, see lemma 2.

Next, in section 2 a subgraph of GB_{2k} induced by all vertices from $\mathcal{M}_{2k}$ will be considered. In section 3 auxiliary results concerning quadratic bent functions will be obtained. A proof of the main result is based on properties of quadratic Boolean functions. In section 4 a connectivity of GM_{2k} will be proved. It is the main result of this work.

2 Maiorana—McFarland bent functions

First of all, consider bent functions that are at the minimal distance from a Maiorana—McFarland bent function.

Lemma 1 *Let $f \in \mathcal{M}_{2k}$, i.e. $f(x, y) = \langle x, \pi(y)\rangle \oplus \varphi(y)$. Let $g(x, y) = \langle x, \pi'(y)\rangle \oplus \varphi'(y)$ where for some distinct $a, b \in \mathbb{F}_2^k$ it holds that*

$$\begin{aligned} \pi'(y) &= \pi(y) \text{ for all } y \in \mathbb{F}_2^k,\ y \neq a, b, \\ \pi'(a) &= \pi(b) \text{ and } \pi'(b) = \pi(a); \\ \varphi'(y) &= \varphi(y) \text{ for all } y \in \mathbb{F}_2^k,\ y \neq a, b. \end{aligned}$$

Then $g \in \mathcal{M}_{2k}$ and dist$(f, g) = 2^k$.

Proof. Since π' is a permutation too, $g \in \mathcal{M}_{2k}$. Note that $f(x, y) = g(x, y)$ for all $x, y \in \mathbb{F}_2^k, y \neq a, b$. It means that it is sufficient to calculate distance when $y = a, b$. Let $y = a$. Then $f(x, a) = \langle x, \pi(a)\rangle \oplus \varphi(a)$ and $g(x, a) = \langle x, \pi(b)\rangle \oplus \varphi'(a)$. Therefore, $f(x, a) \neq g(x, a)$ if and only if

$$\langle x, \pi(a) \oplus \pi(b)\rangle \oplus \varphi(a) \oplus \varphi'(a) = 1.$$

Since $\pi(a) \neq \pi(b)$, there are exactly 2^{k-1} distinct $x \in \mathbb{F}_2^k$ on which $f(x, a) \neq g(x, a)$. The $y = b$ case is the same. Thus, $\mathrm{dist}(f, g) = 2^{k-1} + 2^{k-1} = 2^k$. □

So, since the set of all transpositions generates any permutation, a subgraph of GB_{2k} induced by vertices from $\mathcal{M}_{2k}$ is connected.

Lemma 2 *Let $f \in \widetilde{\mathcal{M}}_{2k}$. Then there are at least $2^{2k+1} - 2^k$ bent functions from $\widetilde{\mathcal{M}}_{2k}$ that are at the distance 2^k from f.*

Proof. Since an affine transform does not change distance between any two Boolean functions, without loss of generality we can suppose that $f \in \mathcal{M}_{2k}$. According to lemma 1 there are at least $4 \cdot 2^k(2^k - 1)/2 = 2^{2k+1} - 2^{k+1}$ bent functions from $\mathcal{M}_{2k}$ that are at the distance 2^k from f.

In addition, if $f(x, y) = \langle x, \pi(y)\rangle \oplus \varphi(y)$, we can add 2^k functions of the form $g(x, y) = \langle x, \pi(y)\rangle \oplus \varphi'(y)$ where $\mathrm{dist}(\varphi, \varphi') = 1$. □

It is not difficult to prove the following lemma that helps us to determine whether a bent function be affine equivalent to a Maiorana—McFarland bent function.

Lemma 3 (A. Canteaut et al. [2]) *Let f be a bent function in $2k$ variables. Then $f \in \widetilde{\mathcal{M}}_{2k}$ if and only if there exists a k-dimensional affine subspace L of $\mathbb{F}_2^{2k}$ such that f is affine on $a \oplus L$ for any $a \in \mathbb{F}_2^{2k}$.*

3 Affinity of a quadratic Boolean function on an affine subspace

In this section we give auxiliary results concerning affinity of a quadratic Boolean function on an affine subspace.

Proposition 1 *Let f be a quadratic Boolean function in n variables and f be affine on an affine subspace L of $\mathbb{F}_2^n$. Then function f is affine on $a \oplus L$ for any $a \in \mathbb{F}_2^n$.*

Proof. Note that $f(x \oplus a) = f(x) \oplus (f(x) \oplus f(x \oplus a))$. Since degree of derivative function $f(x) \oplus f(x \oplus a)$ is less than degree of f (i.e. it is not more than 1), $f(x \oplus a)$ is affine on L and, therefore, f is affine on $a \oplus L$. $\square$

Proposition 2 *Let f be a quadratic Boolean function in n variables and f is affine on a t-dimensional affine subspace L of $\mathbb{F}_2^n$, $t \leq n/2$.*

Then there exist distinct affine subspaces $a_1 \oplus L, \ldots, a_{2^{n-2t}} \oplus L$ such that for some $w \in \mathbb{F}_2^n$ and $c_1, \ldots, c_{2^{n-2t}} \in \mathbb{F}_2$ it holds that

$$f_{a_i \oplus L}(x) = \langle w, x\rangle \oplus c_i,\ \ i \in \{1, \ldots, 2^{n-2t}\}.$$

Proof. Denote by S_w the set of all shifts of L such that the function $f(x) \oplus \langle w, x\rangle$ is a constant on it.

Note that if $f|_{a\oplus L}(x) = \langle w, x\rangle \oplus c$, then for any $w' \in w \oplus L^{\perp}$ it holds that $f|_{a\oplus L}(x) = \langle w', x\rangle \oplus \langle w \oplus w', a\rangle \oplus c$. Thus, $S_w = S_{w\oplus u}$ for $u \in L^{\perp}$.

According to statement 1, f is affine on each of 2^{n-t} distinct shifts of L. Therefore, it is true that

$$\frac{1}{|L^{\perp}|}\sum_{w\in\mathbb{F}_2^n} |S_w| = \frac{1}{2^{n-t}}\sum_{w\in\mathbb{F}_2^n} |S_w| \geq 2^{n-t},$$

that is why $|S_w| \geq 2^{n-t}2^{n-t}/2^n = 2^{n-2t}$ for some $w \in \mathbb{F}_2^n$. $\square$

Proposition 3 *Let f be a Boolean function in n variables, L be an affine subspace of $\mathbb{F}_2^n$, $a \in \mathbb{F}_2^n$ and for some $w \in \mathbb{F}_2^n$ and $c_1, c_2 \in \mathbb{F}_2$ it holds that*

$$\begin{aligned} f|_L(x) &= \langle w, x\rangle \oplus c_1, \\ f|_{a\oplus L}(x) &= \langle w, x\rangle \oplus c_2. \end{aligned}$$

Then f is affine on affine subspace $L \cup (a \oplus L)$.

Proof. Suppose that $a \notin L$, in other case the statement is trivial. It is also clear that $L \cup (a \oplus L)$ is an affine subspace. Consider function

$f'(x) = f(x) \oplus \langle w, x\rangle \oplus c_1$: function f is affine on an affine subspace if and only if f' is affine on it. If $c_1 = c_2$, the statement is obvious. In other way we have $f'|_L = 0$ and $f'|_{a\oplus L} = 1$. Without loss of generality suppose that L is a linear subspace (we can consider affinity of function $f'(x \oplus s)$ on a linear subspace for any $s \in L$). Let $v \in L^\perp$. Then $f'|_L(x) = 0 = \langle v, x\rangle$. Moreover, if for any v it holds that $\langle v, a\rangle = 0$, it is true $a \in (L^\perp)^\perp = L$, but $a \notin L$. So, $\langle v', a\rangle = 1$ for some $v' \in L^\perp$. Therefore,

$$f'|_{a\oplus L}(x) = 1 + 0 = \langle v', a\rangle \oplus \langle v', a \oplus x\rangle = \langle v', x\rangle$$

because of $a \oplus x \in L$. Thus, $f'|_{L\cup(a\oplus L)}(x) = \langle v', x\rangle$. □

Lemma 4 *Let f be a quadratic Boolean function in $2k$ variables and U be a $(2k-1)$-dimensional affine subspace of $\mathbb{F}_2^{2k}$. Then there exists a k-dimensional affine subspace $L \subseteq U$ such that f is affine on L.*

Proof. Note that $\mathbb{F}_2^{2k} = U \cup (c \cup U)$ for some $c \in \mathbb{F}_2^{2k}$, since U is of dimension $2k-1$.

We will prove by induction that there exists a t-dimensional affine subspace L of $\mathbb{F}_2^{2k}$, $t \leq k$, such that $L \subseteq U$ and f is affine on L.

The base of the induction $t = 0$ is obvious, since such affine subspace contains only one element.

Suppose that the statement is true for t, $t < k$. Let's prove that it holds for $t+1$. By the assumption there exists an affine subspace L of dimension t, such that f is affine on L and $L \subseteq U$. By statement 2 for some $w \in \mathbb{F}_2^{2k}$ there exist distinct $a_1 \oplus L, \ldots, a_{2^{2k-2t}} \oplus L$ such that for some $w \in \mathbb{F}_2^{2k}$ it holds that $f|_{a_i\oplus L}(x) = \langle w, x\rangle \oplus c_i$, $c_i \in \mathbb{F}_2$. Since $t < k$, we have $2^{2k-2t} \geq 4$ distinct affine subspaces.

Therefore, always there exist distinct $a \oplus L$, $b \oplus L$, $a, b \in \mathbb{F}_2^{2k}$, among $a_1 \oplus L, \ldots, a_{2^{2k-2t}} \oplus L$ such that $a \oplus L, b \oplus L \subseteq U$ or $a \oplus L, b \oplus L \subseteq c \oplus U$. Next, by statement 3 function f is affine on $L' = (a \oplus L) \cup (b \oplus L)$ of dimension $t+1$. If $L' \subseteq U$, the lemma is proved. Otherwise $L' \subseteq c \oplus U$. By statement 1 function f is affine on $c \oplus L'$ and $c \oplus L' \subseteq U$. □

4 Connectivity of GM_{2k}

The following lemma is the main result for the connectivity proof of GM_{2k}.

Lemma 5 *Let f be a quadratic bent function in $2k$ variables. Then there is a path in GM_{2k} between f and $f(x_1, \dots, x_{2k-1}, x_{2k} \oplus x_1 \oplus c)$, $c \in \mathbb{F}_2$.*

Proof. Since f is quadratic, represent it as the following:

$$f(x_1, \dots, x_{2k}) = f'(x_1, \dots, x_{2k-1}) \oplus (w_1x_1 \oplus \dots \oplus w_{2k-1}x_{2k-1} \oplus d)x_{2k},$$

where $w_1, \dots, w_{2k-1}, d \in \mathbb{F}_2$. Then for $g(x_1, \dots, x_{2k}) = f(x_1, \dots, x_{2k-1}, x_{2k} \oplus x_1 \oplus c)$ it holds that

$$g(x) = f(x) \oplus (w_1x_1 \oplus \dots \oplus w_{2k-1}x_{2k-1} \oplus d)(x_1 \oplus c).$$

Consider $S = \mathrm{supp}(w_1x_1 \oplus \dots \oplus w_{2k-1}x_{2k-1} \oplus d)(x_1 \oplus c)$. Note that S is an affine subspace. Prove that there exists a k-dimensional affine subspace L of $\mathbb{F}_2^{2k}$ such that $L \subseteq S$ and f is affine on L.

Case $w_1 = \dots = w_{2k-1} = 0$ is impossible, because a bent function $f(x) \oplus dx_k$ must depend on each its variable. Therefore, there exists $w_t \neq 0$ for some $1 \leq t \leq 2k-1$.

If only w_1 is nonzero, then $S = \mathrm{supp}(x_1 \oplus c)(x_1 \oplus d)$. If $c \neq d$, functions f and g are the same. Otherwise S is a linear subspace of dimension $2k-1$ and by lemma 4 there exists required L.

If there exists other nonzero w_t, without loss of generality suppose that $t = 2k-1$. Consider $f|_{x_{2k-1} = w_1x_1 \oplus \dots \oplus w_{2k-2}x_{2k-2} \oplus d \oplus 1} \oplus x_{2k}$ which is equal to

$$f'(x_1, \dots, x_{2k-2}, w_1x_1 \oplus \dots \oplus w_{2k-2}x_{2k-2} \oplus d \oplus 1)$$

as a function in $2k-2$ variables. Let $U = \mathrm{supp}(x_1 \oplus c)$. Then by lemma 4 there exists a $(k-1)$-dimensional affine subspace L' of $\mathbb{F}_2^{2k-2}$ that $L' \subseteq U$ and $f'(x_1, \dots, x_{2k-2}, w_1x_1 \oplus \dots \oplus w_{2k-2}x_{2k-2} \oplus d \oplus 1)$ is affine on L'.

Therefore, f is affine on a k-dimensional affine subspace $L \subseteq \mathbb{F}_2^{2k}$,

$$L = \{(y, w_1y_1 \oplus \dots \oplus w_{2k-2}y_{2k-2} \oplus d \oplus 1, z) \ : \ y \in L', z \in \mathbb{F}_2\},$$

and at the same time $L \subseteq S$, because $(w_1x_1 \oplus \ldots \oplus w_{2k-1}x_{2k-1} \oplus d)(x_1 \oplus c)$ does not depend on x_{2k}, $w_1x_1 \oplus \ldots \oplus w_{2k-1}x_{2k-1} \oplus d = 1$ for any $x \in L$ and $x_1 \oplus c = 1$ due to choosing L'. Required L has been found.

To complete the proof, note that, thanks to choosing L, it holds that

$$S = (a_1 \oplus L) \cup \ldots \cup (a_m \oplus L) \text{ for some } a_1, \ldots, a_m \in \mathbb{F}_2^{2k},\ m = 2^{\dim S - \dim L}.$$

And since f is quadratic, it is affine on each $a_i \oplus L$ too. Next, let

$$f_0 = f \text{ and } f_i = f_{i-1} \oplus Ind_{a_i \oplus L},\ 1 \leq i \leq m.$$

Then $f_m = g$ and each f_i is a bent function because of $a_1 \oplus L, \ldots, a_m \oplus L$ are not intersected. Moreover, each f_i is affine on $a \oplus L$ for any $a \in \mathbb{F}_2^{2k}$. Thus, all $f_i \in \widetilde{\mathcal{M}}_{2k}$ by lemma 3. □

Theorem 1 *Graph GM_{2k} is connected for all $k \geq 1$.*

Proof. According to lemma 1, for any bent function $f_0 \in \mathcal{M}_{2k}$ there are $f_1, \ldots, f_n \in \mathcal{M}_{2k}$ for some n with $\mathrm{dist}(f_i, f_{i+1}) = 2^k$ and $f_n(x, y) = x_1y_1 \oplus x_2y_2 \oplus \ldots \oplus x_ky_k$.

Therefore, for any bent function $f_0 \in \widetilde{\mathcal{M}}_{2k}$ there are $f_1, \ldots, f_n \in \widetilde{\mathcal{M}}_{2k}$ for some n with $\mathrm{dist}(f_i, f_{i+1}) = 2^k$ and f_n is a quadratic bent function, i.e. there is a path in GM_{2k} between any $f_0 \in \widetilde{\mathcal{M}}_{2k}$ and some quadratic bent function.

Thus, it is enough to prove that there exists a path in GM_{2k} between any two quadratic bent functions.

According to Dickson's theorem, any two quadratic bent functions are affinely equivalent, i.e. for any two quadratic bent functions f, g in $2k$ variables there exists an invertible $2k \times 2k$ binary matrix A and $b \in \mathbb{F}_2^{2k}$ such that $g(x) = f(xA \oplus b)$ for any $x \in \mathbb{F}_2^{2k}$.

At the same time by lemma 5 there exists a path in GM_{2k} between any quadratic f and $f(x_1, \ldots, x_{2k-1}, x_{2k} \oplus x_1 \oplus c)$. On one hand, we can easily extend lemma 5 (using permutations on variable numbers) to transformations of the form

$$\begin{aligned} x'_l &= x_l \text{ for all } l \in \{1, \ldots, 2k\} \backslash \{i\}, \\ x'_i &= x_i \oplus x_j \oplus c \end{aligned}$$

for any $i, j \in \{1, \dots, 2k\}$, $i \neq j$ and $c \in \mathbb{F}_2$. On the other hand, the set of all these transformations generates any invertible affine transform $xA \oplus b$. The theorem is proved. □

Corollary 1 *Graphs GB_2, GB_4 and GB_6 are connected.*

It follows from all bent functions in 2, 4 and 6 variables are affinely equivalent to Maiorana—McFarland bent functions.

5 Conclusion

In general, GB_{2k} is not connected starting with $k = 7$ due to existing of isolated vertices, i.e. such bent functions for which there are no bent functions at the distance 2^k from them. Such bent functions are called *non-weakly normal*, they were constructed in [2]. Connectivity of GB_{2k} without isolated vertices is an open question.

6 Acknowledgments

The author was supported by the Russian Foundation for Basic Research (project no. 15-07-01328) and by Grant NSh-1939.2014.1 of President of Russia for Leading Scientific Schools.

References

[1] Carlet C. Two new classes of bent functions // EUROCRYPT'93. LNCS. — 1994. — V. 765. — P. 77–101.

[2] Canteaut A., Daum M., Dobbertin H., and Leander G. Finding non-normal bent functions // Discrete Appl. Math. — 2006. — V. 154, No. 2. — P. 202–218.

[3] Kolomeec N. A. An upper bound for the number of bent functions at the distance 2^k from an arbitrary bent function in $2k$ variables // Prikladnaya Diskretnaya Matematika. — 2014. — No 3. — P. 28–39.

[4] Kolomeec N. A, Pavlov A. V Properties of bent functions with minimal distance // Prikladnaya Diskretnaya Matematika. — 2009. — No 4. — P. 5–20.

[5] Logachev O. A., Sal'nikov A. A., Smyshlyaev S. V., Yashenko V. V. Boolean functions in coding theory and cryptology. MCCME (Moscow, Russia), 2012. 584 p.

[6] McFarland R. L. A family of difference sets in non-cyclic groups // J. Combin. Theory, Ser. A. — 1973. — V. 15. — P. 1–10.

[7] Rothaus O. On bent functions // J. Combin. Theory, Ser. A. — 1976. — V. 20, No 3. — P. 300–305.

[8] Tokareva N. N. Bent functions: results and application. A survey. // Prikladnaya Diskretnaya Matematika. — 2009. — No 1. — P. 15–37.

[9] Tokareva N. N. Bent Functions, Results and Applications to Cryptography. Acad. Press. Elsevier, 2015. 230 p. ISBN: 978–0128023181.

Solutions Set Stability of a System of Equations in a Case of their Random Distortions

Vladimir Mikhailov Artem Volgin

Abstract

We consider two systems of equations: a system where in the left part there are functions of a special type and a system where in the left part there are functions received from functions of the first system by their independent random distortion. Conditions are derived for the probabilistic laws of distortions of functions providing three versions of reciprocal behaviour of solution sets of these systems at the increase in number of the equations and number of unknown variables.

Keywords: system of equations, linear function.

1 Introduction

In the cryptoanalysis of symmetric ciphers a significant role is played by systems of the equations and methods of their solutions ([1], [3]). According to an encryption algorithm, known bits of the plaintext and ciphertext are tied by a system of the equations with unknown bits of a secret key. One of methods of the solution of systems of equations is based on searching of the "close" easily solved systems. In [4], [5] concepts of "close" functions are considered. Two systems are "close" if the values of corresponding functions in this systems differ in a small number of variables. Ranges of search of the "close" systems are given in this report.

Let $N, T \in \mathbb{N}$. Consider a system of equations

$$f_t(x) = b_t, \quad t = 1, ..., T, \tag{1}$$

where $f_t : \{0, ..., N-1\} \to \{0, 1\}$, $b_t \in \{0, 1\}$, and a similar system, in a left side of which functions are obtained from functions $f_1, ..., f_T$ by their independent distortion. In this article, for several types of random distortions conditions are obtained supporting the following variations of an asymptotic behaviour of a size of a common part of these systems solutions set at a concerted increase of an equations number and dimensionality:

– conditions, under which the probability of not intercrossing of these sets tends to one,

– conditions, under which the probability of these sets coincidence tends to one,

– intermediate conditions, under which the number of system's general solutions has a non-degenerate limiting distribution (this role is played by a binomial distribution).

By $\mathbf{B}(N)$, let us denote a set of all 2^N functions on the set $\{0, ..., N-1\}$ and by $\oplus$ the addition operator modulo 2. Let us note that in the case when functions $f_1, ..., f_T$ are changed with functions $f_1 \oplus b_1, ..., f_T \oplus b_T$, the system (1) reduces to a uniform system

$$f_t(x) = 0, \quad t = 1, ..., T, \tag{2}$$

for which the main calculations are given here.

2 Main theorem

For each function $f \in \mathbf{B}(N)$, let us juxtapose sets $A_0(f)$ and $A_1(f)$ of those argument values, when it takes values of zero and one respectively. For an integer $0 \le v \le |A_0(f)|$, let us assume that

$$B_0(f, v) = \{g \in \mathbf{B}(N) : |A_1(g) \cap A_0(f)| = v\}.$$

A distribution on functions set $B_0(f, v)$ will be called by us as a *uniform distribution with respect of zeros of the function* f, if for each subset $A \subset A_0(f)$ of power $|A| = v$, the following equation is true:

$$\mathbf{P}\{A_1(g) \cap A_0(f) = A\} = \frac{1}{C^v_{|A_0(f)|}}. \tag{3}$$

The condition (3) means that the set $A_1(g) \cap A_0(f)$ is distributed in such a way as if it was built by a selection with equal probability without return of v points of the set $A_0(f)$. In other respects, the distribution of the set $A_1(g)$ (i.e. of a random function from $B_0(f, v)$) is arbitrary.

From this feature, it follows, for example, that for any $x \in A_0(f)$, the following equation is true:

$$\mathbf{P}\{x \in A_0(g)\} = 1 - \frac{v}{|A_0(f)|}. \tag{4}$$

In a similar way on the set

$$B_1(f, u) = \{g \in \mathbf{B}(N) : |A_0(g) \cap A_1(f)| = u\}$$

a distribution is determined being *uniform with respect of units of the function* f. For such distribution under any $x \in A_1(f)$

$$\mathbf{P}\{x \in A_1(g)\} = 1 - \frac{u}{A_1(f)}.$$

Let $v_1, ..., v_T$ − be integers satisfying to relations $0 \leq v_t < |A_0(f_t)|$, $t = 1, ..., T$. At each $t = 1, ..., T$ on the set $B_0(f_t, v_t)$, let us define some probability distribution uniform with respect of zeros of the function f_t. In accordance with these distributions, let us select functions $\tilde{f}_1, ..., \tilde{f}_T$ randomly and independently.

Let us introduce denomination $\overline{v} = (v_1, ..., v_T)$ and denomination $\tilde{S}(\overline{v})$ for solutions set of a system of random equations

$$\tilde{f}_t(x) = 0, \quad t = 1, ..., T. \tag{5}$$

Let S − be a solutions set of the uniform system (2).

Remark 1. In statements of this paper at a proceeding to limit, changes of all parameters and functions are allowed in limits of natural restrictions and conditions specified in statements.

Theorem 1. *Let us assume that the equations left sides of the system* (5) *are obtained by independent random selection of the functions* $\tilde{f}_1, ..., \tilde{f}_T$ *from the sets* $B_0(f_1, v_1), ..., B_0(f_T, v_T)$ *respectively with aid of distributions uniform with respect of zeros of the functions* $f_1, ..., f_T$. *Let* $|S| \geq 1$, *and* $N, T \to \infty$. *Then:*

1) *if*

$$\sum_{t=1}^{T}\frac{v_t}{|A_0(f_t)|}-\ln|S|\to\infty, \tag{6}$$

then $\mathbf{P}\{\tilde{S}(\overline{v})\cap S=\varnothing\}\to 1$*;*

2) *if* $|S|=const$ *and*

$$\sum_{t=1}^{T}\frac{v_t}{|A_0(f_t)|}\to\kappa\in(0,\infty),\quad \max_{t=1,\dots,T}\frac{v_t}{|A_0(f_t)|}\to 0,\quad \sum_{t=1}^{T}\frac{1}{|A_0(f_t)|}\to 0, \tag{7}$$

Then the distribution of the random value $|\tilde{S}(\overline{v})\cap S|$ *converges to the binomial distribution with the parameters* $|S|$ *and* $e^{-\kappa}$*;*

3) *if*

$$|S|\sum_{t=1}^{T}\frac{v_t}{|A_0(f_t)|}\to 0, \tag{8}$$

then $\mathbf{P}\{\tilde{S}(\overline{v})\supseteq S\}\to 1.$

Remark 2. If $N,T\to\infty$ and condition $|S|=O(1)$ is true, then condition (6), the first part of (7) and condition (8) of the theorem 1 have the following representation:

$$\sum_{t=1}^{T}\frac{v_t}{|A_0(f_t)|}\to\infty,\quad \sum_{t=1}^{T}\frac{v_t}{|A_0(f_t)|}\to\kappa\in(0,\infty),\quad \sum_{t=1}^{T}\frac{v_t}{|A_0(f_t)|}.$$

Remark 3. The theorem 1 statement and all the calculations used for its proof are correct for any distributions on sets $B(f_t,v_t)$, uniform with respect of zeros of the function f_t, $t=1,...,T$. For different equations of the system, these distributions can differ. Among them, one of a special interest for us is the following version of a random distortion of the equations of the system (1). For each function f at integers $0\le u<|A_1(f)|$ and $0\le v<|A_0(f)|$, let us assume that

$$B(f,u,v)=\{g\in\mathbf{B}(N):|A_0(g)\cap A_1(f)|=u,|A_1(g)\cap A_0(f)|=v\}.$$

Note that $B(f,u,v)\subset B(f,v)$ and $|B(f,u,v)|=C^u_{|A_1(f)|}C^v_{|A_0(f)|}$. On the sets $B(f_1,u_1,v_1),...,B(f_T,u_T,v_T)$, let us define uniform probability distributions, according to which randomly and independently we would select functions $\tilde{f}_1, ..., \tilde{f}_T$. These functions can be interpreted as the random distortions of the functions $f_1, ..., f_T$. Let $\tilde{S}(\overline{u}, \overline{v})$ to be a solution set of the system (1) distorted in such a way. For this set, the theorem 1 takes the following form.

Corollary 1. *Let us assume that the equation's left sides of the system* (5) *are obtained by an independent equiprobable selection of the functions* $\tilde{f}_1, ..., \tilde{f}_T$ *from the sets* $B(f_1, u_1, v_1), ..., B(f_T, u_T, v_T)$ *respectively. Let* $|S| \geq 1$, $N, T \to \infty$. *Then:*

1) *if the condition* (6) *is true, then* $\mathbf{P}\{\tilde{S}(\overline{u}, \overline{v}) \cap S = \varnothing\} \to 1$*;*

2) *if* $|S| = const$ *and the conditions* (7) *are true, then the random value distribution* $|\tilde{S}(\overline{u}, \overline{v})|$ *converges to binomial distribution with parameters* $|S|$ $e^{-\kappa}$*;*

3) *if the condition* (8) *is true, then* $\mathbf{P}\{\tilde{S}(\overline{u}, \overline{v}) \supseteq S\} \to 1$.

Remark 4. A uniform distribution on the set $B(f, u, v)$ has the special feature that it is uniform with respect to both zeros and ones of the function f. This is why replacing of the functions f_t and $\tilde{f}_t$ in the system (1) and in the system

$$\tilde{f}_t(x) = b_t, \quad t = 1, ..., T, \tag{9}$$

with functions $f_t \oplus b_t$ $\tilde{f}_t \oplus b_t$, $t = 1, ..., T$, leads to zeros in the right sides of their systems just because of positions changing of distribution parameters of the distorted system. So we obtain the following statement.

Lets denote the system (1) solutions set as S_b and the system (9) solutions set as $\tilde{S}_b(\overline{u}, \overline{v})$. Let's also denote $b = (b_1, ..., b_T)$. Note designators u_t and v_t on $v_t^{(1)}$ and $v_t^{(0)}$ respectively.

Corollary 2. *Let us assume that the left sides of the system* (9) *obtained by the way of the independent equiprobable selection of the functions* $\tilde{f}_1, ..., \tilde{f}_T$ *from the sets* $B(f_1, v_1^{(1)}, v_1^{(0)}), ..., B(f_T, v_T^{(1)}, v_T^{(0)})$ *respectively. Then:*

1) *if*

$$\sum_{t=1}^{T} \frac{v_t^{(b_t)}}{|A_{b_t}(f_t)|} - \ln |S_b| \to \infty,$$

then $\mathbf{P}\{\tilde{S}_b(\overline{u}, \overline{v}) \cap S_b = \varnothing\} \to 1$*;*

2) *if* $|S_b| = const$ *and*

$$\sum_{t=1}^{T} \frac{v_t^{(b_t)}}{|A_{b_t}(f_t)|} \to \kappa \in (0,\infty), \quad \max_{t=1,\dots,T} \frac{v_t^{(b_t)}}{|A_{b_t}(f_t)|} \to 0, \quad \sum_{t=1}^{T} \frac{1}{|A_{b_t}(f_t)|} \to 0,$$

then the distribution of the random value $|\tilde{S}_b(\overline{u},\overline{v}) \cap S_b|$ *converges to the binomial distribution with the parameters* $|S_b|$ *and* $e^{-\kappa}$*;*

3) *if*

$$|S_b| \sum_{t=1}^{T} \frac{v_t^{(b_t)}}{|A_{b_t}(f_t)|} \to 0,$$

then $\mathbf{P}\{\tilde{S}_b(\overline{u},\overline{v}) \supseteq S_b\} \to 1.$

3 Distortion by replacing of functional values with a random value

Let us assume that on a numbers set $\{0,\dots,N-1\}$ probability distributions $\mathcal{P}_1,\dots,\mathcal{P}_T$ are defined and $\varsigma_1,\dots,\varsigma_T$ are independent random values with
distributions $\mathcal{P}_1,\dots,\mathcal{P}_T$. For each function $f \in \mathbf{B}(N)$, let's juxtapose a set $C(f,\varsigma)$, $\varsigma \in \{0,\dots,N-1\}$, of those functions $g \in \mathbf{B}(N)$ that differ from the function f uniformly at values ς of argument $x \in \{0,\dots,N-1\}$.

On the sets $C(f_1,\varsigma_1),\dots,C(f_T,\varsigma_T)$, let us define uniform probability distributions, according to which randomly and independently we shall select functions $\tilde{f}_1,\dots,\tilde{f}_T$. These functions can be interpreted as random perturbances of the functions $f_1,\dots,f_T$. Let's consider a system of random equations

$$\tilde{f}_t(x) = 0, \quad t = 1,\dots,T, \tag{10}$$

with functions $f_1,\dots,f_T$.. Let's denote system (10) solutions set as $\tilde{S}(\mathcal{P}_1,\dots,\mathcal{P}_T)$.

Theorem 2. *Let us assume that the left sides of the system* (10) *equations are obtained by independent and random selection of the functions* $\tilde{f}_1,\dots,\tilde{f}_T$ *from the set* $\mathbf{B}(N)$ *in accordance with the distributions* $\mathcal{P}_t$,

$t = 1, ..., T$. *Let us assume that* $N, T \to \infty$, $|S| = const \geq 1$, *while distributions* $\mathcal{P}_t$ *are changed in such a way that*

$$\mathbf{D}_{\mathcal{P}}\varsigma_t = O(\mathbf{E}_{\mathcal{P}_t}\varsigma_t) \text{ is uniform on } t = 1, ..., T. \tag{11}$$

Then:

1) *if*

$$\frac{1}{N}\sum_{t=1}^{T}\mathbf{E}_{\mathcal{P}_t}\varsigma_t \to \infty,$$

then $\mathbf{P}\{\tilde{S}(\mathcal{P}_1, ..., \mathcal{P}_T)\bigcap S = \varnothing\} \to 1$*;*

2) *if*

$$\frac{1}{N}\sum_{t=1}^{T}\mathbf{E}_{\mathcal{P}_t}\varsigma_t \to \kappa \in (0, \infty), \quad \max_{t=1,...,T}\frac{\mathbf{E}_{\mathcal{P}_t}\varsigma_t}{N} \to 0, \quad \sum_{t=1}^{T}\frac{1}{|A_0(f_t)|} \to 0,$$

then the distribution of the random value $|\tilde{S}(\mathcal{P}_1, ..., \mathcal{P}_T) \cap S|$ *converges to the binomial distribution with the parameters* $|S|$ *and* $e^{-\kappa}$*;*

3) *if*

$$\frac{1}{N}\sum_{t=1}^{T}\mathbf{E}_{\mathcal{P}_t}\varsigma_t \to 0,$$

then $\mathbf{P}\{\tilde{S}(\mathcal{P}_1, ..., \mathcal{P}_T) \supseteq S\} \to 1$.

Remark 5. The condition (11) of the theorem 2 can be weakened if instead of it we add the following in the condition of the point 1):

$$\max_{t=1,...,T}\frac{\mathbf{D}_{\mathcal{P}_t}\varsigma_t}{N} < m_1, \quad m_1 = \text{const} > 0$$

as well as the following in the condition of the point 2):

$$T\max_{t=1,...,T}\frac{\mathbf{D}_{\mathcal{P}_t}\varsigma_t}{N} < m_2, \quad m_2 = \text{const} > 0.$$

Let's formulate a number of derived relations from the theorem 2 for three different kinds of distortion of the functions $f_1, ..., f_T$:

– In the first case to each function $f_t \in \mathbf{B}(N)$ the set $C_1(f_t, d_t)$, $t = 1, ..., T$, is juxtaposed with those functions $g \in \mathbf{B}(N)$, that differ from f_t

exactly at d_t values of the argument $x \in \{0, ..., N-1\}$. It is supposed that on the sets $C_1(f_1, d_1), ..., C_1(f_T, d_T)$, uniform probability distributions are defined.

– In the second case to each function $f_t \in \mathbf{B}(N)$ the set $C_2(f_t, d_t)$, $t = 1, ..., T$, is juxtaposed with those functions $g \in \mathbf{B}(N)$, that differ from f_t not more than under d_t values of the argument $x \in \{0, ..., N-1\}$. It is supposed that on the sets $C_2(f_1, d_1), ..., C_2(f_T, d_T)$, uniform probability distributions are defined.

– In the third case to each function $f_t \in \mathbf{B}(N)$ the set $C_3(f_t, d_t)$, $t = 1, ..., T$, is juxtaposed with those functions $g \in \mathbf{B}(N)$, that differ from f_t at every value of the argument $x \in \{0, ..., N-1\}$ with a probability p_t, $t = 1, ..., T$, which depends on equations number. Along with it, the distortions are entered independently. It is supposed that on the sets $C_3(f_1, d_1), ..., C_3(f_T, d_T)$, uniform probability distributions are defined.

For the first two kinds of distortions, the derived relations from the theorem 2 are formulated in the same way. Let's denote the system (10) solutions sets as $\tilde{S}_1(d_1, ..., d_T)$ in the first case and as $\tilde{S}_2(d_1, ..., d_T)$ in the second case.

Corollary 3. *Let us assume that the left sides of the system* (10) *equations are obtained by a way of an independent random selection of functions $\tilde{f}_1, ..., \tilde{f}_T$ from the sets $C_1(f_1, d_1), ..., C_1(f_T, d_T)$ or from the sets $C_2(f_1, d_1), ..., C_2(f_T, d_T)$. Also let $N, T \to \infty$, $|S| = const \geq 1$. Then:*

1) *if*

$$\frac{1}{N} \sum_{t=1}^{T} d_t \to \infty,$$

then $\mathbf{P}\{\tilde{S}_1(d_1, ..., d_T) \bigcap S = \varnothing\} \to 1$ and $\mathbf{P}\{\tilde{S}_2(d_1, ..., d_T) \bigcap S = \varnothing\} \to 1$;

2) *if*

$$\frac{1}{N} \sum_{t=1}^{T} d_t \to \kappa \in (0, \infty), \quad \max_{t=1,...,T} \frac{d_t}{N} \to 0, \quad \sum_{t=1}^{T} \frac{1}{|A_0(f_t)|} \to 0,$$

then the distributions of the random values $|\tilde{S}_1(d_1, ..., d_T) \cap S|$ and $|\tilde{S}_2(d_1, ..., d_T) \cap S|$ converges to the binomial distribution with the parameters $|S|$ $e^{-\kappa}$;

3) *if*
$$\frac{1}{N}\sum_{t=1}^{T} d_t \to 0,$$
then $\mathbf{P}\{\tilde{S}_1(d_1,...,d_T) \supseteq S\} \to 1$ *and* $\mathbf{P}\{\tilde{S}_2(d_1,...,d_T) \supseteq S\} \to 1$.

Let's denote the system (10) solutions set as $\tilde{S}_3(p)$ under distortions of the third case.

Corollary 4. *Let us assume that the left sides of the system* (10) *equations are obtained by a way of an independent random selection of functions* $\tilde{f}_1,...,\tilde{f}_T$ *from the sets* $C_3(f_1,d_1),...,C_3(f_T,d_T)$. *Let* $N,T \to \infty$, $|S| = const \geq 1$. *Then:*

1) *if*
$$p(1)+...+p(T) \to \infty,$$
then $\mathbf{P}\{\tilde{S}_3(p) \bigcap S = \varnothing\} \to 1$*;*

2) *if*
$$p(1)+...+p(T) \to \kappa \in (0,\infty), \quad \max_{t=1,...,T} p(T) \to 0, \quad \sum_{t=1}^{T} \frac{1}{|A_0(f_t)|} \to 0,$$
then the distribution of the random value $|\tilde{S}_3(p) \cap S|$ *converges to the binomial distribution with the parameters* $|S|$ $e^{-\kappa}$*;*

3) *if*
$$p(1)+...+p(T) \to 0,$$
then $\mathbf{P}\{\tilde{S}_3(p) \supseteq S\} \to 1$.

4 Conclusion

Let's consider the following problem connected with the above problem. Let us assume that it is necessary to solve an equations system. If as a result of a "just small distortion" of the left sides of the equations, this system is transformed into an easily solvable system, the task reduces down to finding and solving of a nearest easily solvable system. These problems are quite similar to problems arising in case of linear analogs use ([2], [3]).

As those easily solvable systems, systems of linear Boolean equations and triangular systems can serve. The most convenient case for the purposes of illustration is one referred to Boolean non-linear equations; and as an easily solvable system, a system of linear equations acts.

We remind once more that in our case, only very good analogs are items of interest. That is why their search (given that such analog exists) is much easier than in the classical case. It is enough by chance or in any other way from a functions table to select $n+1$ different couples "argument value – function value." By these couples, a linear function is defined unequivocally. If a table of an original function and one of a being-sought linear analog differ just insignificantly, then almost for sure it will be the needed linear function. That is why a solution algorithm for a system with such Boolean functions is absolutely clear.

1. For each function from a left side of a system, its linear analog is found in the specified way.
2. The built system of the linear equations is solved.
3. Checking is needed that all these solutions comply with the original non-linear system.

Note that basically an original non-linear system can have other solutions.

The theorems given above and derived relations obtained from them, specify deviation scopes from analogs, in frame of which this procedure makes sense.

References

[1] Agibalov G.P. Methods of the decision of equations systems over a finite field. – Bulletin of Tomsk Statement University. Appendix, 2006. – No 17. P. 4-9. (In Russian).

[2] Balakin G.V. Introduction to the theory of the random equations systems. – Works of discrete mathematics. V. 1, 1997. P.1-18. (In Russian).

[3] Matsui M. Linear Cryptanalysis Method for DES cipher. – LNCS, 1993. V. 765. P. 386-397.

[4] Mihailov V.G. Assessment of accuracy of Poisson approximation for number of empty cells in equiprobable scheme of particles arrangement by complexes and its application area. – Works of Mathematical institute named after V.A.Steklov, RAS, 2013. P. 165-180. (In Russian).

[5] Volgin A.V. Asymptotic features of solutions set of distorted equations systems. – Abstracts of conference SIBECRYPT'14, 2014. P. 48. (In Russian).

5 Appendix

5.1 Proof of the theorem 1

Due to the function $\tilde{f}_t$ distribution uniformity with respect to zeros of the function f_t, as well as due to the equation (3), we have

$$\mathbf{P}\{\tilde{f}_t(x) = 0\} = 1 - \frac{v_t}{|A_0(f_t)|} \quad \text{for any } x \in S, \quad t = 1, ..., T. \tag{12}$$

Let's prove the statement 1). From the condition of distortions independency of separate system functions (1) and (4), it follows that

$$\mathbf{P}\{x \in S\} = \prod_{t=1}^{T} \mathbf{P}\{\tilde{f}_t(x) = 0\} = \prod_{t=1}^{T} \left(1 - \frac{v_t}{|A_0(f_t)|}\right) \text{for any } x \in S, \tag{13}$$

and it means that

$$\mathbf{E}|S \cap \tilde{S}(\overline{v})| = \sum_{x \in S} \mathbf{P}\{x \in \tilde{S}(\overline{v})\} = |S| \prod_{t=1}^{T} \left(1 - \frac{v_t}{|A_0(f_t)|}\right). \tag{14}$$

Now from (6) and (14), it follows that

$$\mathbf{E}|S \cap \tilde{S}(\overline{v})| \leq |S| \exp\left\{-\sum_{t=1}^{T} \frac{v_t}{|A_0(f_t)|}\right\} \to 0.$$

Consequently, $\mathbf{P}\{\tilde{S}(\overline{v}) \cap S = \varnothing\} \to 1$.

Let's prove the statement 3). From (12), the equations follow:

$$\mathbf{P}\{\tilde{f}_t(x) = 1\} = \frac{v_t}{|A_0(f)|} \quad \text{for any;} \, x \in S, \quad t = 1, ..., T, \tag{15}$$

and from (15) we obtain that

$$\mathbf{P}\{x \notin \tilde{S}(\overline{v})\} \leq \sum_{t=1}^{T} \mathbf{P}\{\tilde{f}_t(x) = 1\} = \sum_{t=1}^{T} \frac{v_t}{|A_0(f_t)|} \quad \text{for any } x \in S.$$

Hence,

$$\mathbf{E}|S \setminus \tilde{S}(\overline{v})| \leq \sum_{x \in S} \mathbf{P}\{x \notin \tilde{S}(\overline{v})\} \leq |S| \sum_{t=1}^{T} \frac{v_t}{|A_0(f_t)|}. \tag{16}$$

According to condition (8), the expression in the right side (17) tends to zero. So $\mathbf{P}\{\tilde{S}(\overline{v}) \supseteq S\} \to 1$.

For proving of the statement 2), we'll need the following result.

Lemma 1. *Let* $|S| = \text{const}$, $0 \le v_t < |A_0(f_t)|$, $P = \prod_{t=1}^{T}\left(1 - \frac{v_t}{|A_0(f_t)|}\right)$. *Then it will be found such absolute constant value* $C < \infty$ *that*

$$\max_r \left| \mathbf{P}\{|\tilde{S}(\overline{v}) \cap S| = r\} - C_{|S|}^r P^r (1-P)^{|S|-r} \right| \le C \sum_{t=1}^{T} \frac{1}{|A_0(f)| - v_t}.$$

Let's prove the statement 2). From the conditions (7), it follows that

$$\sum_{t=1}^{T} \frac{1}{|A_0(f_t)| - v_t} \to 0.$$

It means that according to lemma 1, the distribution of elements number of the set $\tilde{S}(\overline{v}) \cap S$ approximates to the binomial distribution $\text{Bi}(|S|, P)$. In its turn, from the condition (7) of the theorem, it follows that $P \to e^{-\kappa}$. This is why the distribution $\text{Bi}(|S|, P)$ approximates to the $\text{Bi}(|S|, e^{-\kappa})$. The same is true also for the elements distribution of the set $|\tilde{S}(\overline{v} \cap S)|$. The theorem 1 is proved.

5.2 Proof of the lemma 1

Let J_x to be an indicator of an event $x \in \tilde{S}(\overline{v})$, $1 \le m < |S|$ and

$$k_1, ..., k_m \in S = \bigcap_{t=1}^{T} A_0(f_t), \quad k_i \ne k_j \ (i \ne j).$$

Then with due consideration of the formula

$$\frac{(|A_0(f_t)| - m)_{v_t}}{(|A_0(f_t)|)_{v_t}} = \frac{(|A_0(f_t)| - v_t)_m}{(|A_0(f_t)|)_m},$$

we obtain

$$\mathbf{E} J_{k_1} ... J_{k_m} = \prod_{t=1}^{T} \frac{(|A_0(f_t)| - m)_{v_t}}{(|A_0(f_t)|)_{v_t}} = \prod_{t=1}^{T} \frac{(|A_0(f_t)| - v_t)_m}{(|A_0(f_t)|)_m}. \tag{17}$$

At $m = 1$ this formula coincides with (13).

As
$$\frac{(|A_0(f_t)| - v_t)_m}{(|A_0(f_t)|)_m} \leq \frac{(|A_0(f_t)| - v_t)^m}{(|A_0(f_t)|)^m},$$
then
$$\mathbf{E}J_{k_1}...J_{k_m} \leq \mathbf{E}J_{k_1}...\mathbf{E}J_{k_m} = (\mathbf{E}J_{k_1})^m = \left(1 - \frac{v_t}{|A_0(f_t)|}\right)^m. \qquad (18)$$
From the inequalities
$$(A)_m \geq A^m - C_m^2 A^{m-1}, \quad (A)_m \leq A^m,$$
true for natural numbers $A > m \geq 2$, we obtain that
$$\begin{aligned} &\frac{(|A_0(f_t)| - v_t)^m}{(|A_0(f_t)|)^m} - \frac{(|A_0(f_t)| - v_t)_m}{(|A_0(f_t)|)_m} \\ &= \frac{(|A_0(f_t)| - v_t)^m (|A_0(f_t)|)_m - (|A_0(f_t)| - v_t)_m (|A_0(f_t)|)^m}{(|A_0(f_t)|)^m (|A_0(f_t)|)_m} \\ &\leq \frac{(|A_0(f_t)| - v_t)^m - (|A_0(f_t)| - v_t)_m}{(|A_0(f_t)|)_m} \\ &\leq C_m^2 \frac{(|A_0(f_t)| - v_t)^{m-1}}{(|A_0(f_t)| - v_t)_m}. \end{aligned} \qquad (19)$$
From (17) and (19), we obtain
$$\mathbf{E}J_{k_1}...J_{k_m} \geq \prod_{t=1}^{T} \left(\left(1 - \frac{v_t}{|A_0(f_t)|}\right)^m - C_m^2 \frac{(|A_0(f_t)| - v_t)^{m-1}}{(|A_0(f_t)|)_m} \right). \qquad (20)$$
To the right side of (20), let's apply the inequality
$$(a_1 - x_1)...(a_T - x_T) \geq a_1...a_T \left(1 - \sum_{t=1}^{T} \frac{x_t}{a_t}\right)$$
at $0 \leq x_t < a_t$; and if to take
$$a_t = \left(1 - \frac{v_t}{|A_0(f_t)|}\right)^m, \quad x_t = C_m^2 \frac{(|A_0(f_t)| - v_t)^{m-1}}{(|A_0(f_t)|)_m},$$

we'll obtain the inequality

$$\mathbf{E}J_{k_1}...J_{k_m} \geq (1 - b_{k_1,...,k_m})\mathbf{E}J_{k_1}...\mathbf{E}J_{k_m}$$

$$= (1 - b_{k_1,...,k_m}) \left(1 - \frac{v_t}{|A_0(f_t)|}\right)^m,$$

where

$$b_{k_1,...,k_m} = C_m^2 \sum_{t=1}^{T} \frac{(|A_0(f_t)| - v_t)^{m-1}}{(|A_0(f_t)|)^m} \left(1 - \frac{v_t}{|A_0(f_t)|}\right)^{-m}$$

$$\leq C_m^2 \max_{1 \leq t \leq T} \frac{|A_0(f_t)|^m}{(|A_0(f_t)|)_m} \sum_{t=1}^{T} \frac{1}{|A_0(f_t)| - v_t}. \tag{21}$$

A probability of any outcome of a vector $(J_1, ..., J_r)$ is expressed with a finite linear combination of values $\mathbf{E}J_{k_1...J_{k_m}}$, $m = 1, ..., r$. This is why from (18) and (21), it follows that the maximum probability of differences of distributions outcomes of the vector $(J_1, ..., J_r)$ and the vector $(V_1, ..., V_r)$ from independent random values distributed with probabilities

$$\mathbf{P}\{V_k = 1\} = 1 - \mathbf{P}\{V_k = 0\} = \mathbf{E}J_k = P, \quad k = 1, ..., r,$$

tends to zero as $O\left(\sum_{t=1}^{T} \frac{1}{|A_0(f_t)| - v_t}\right)$. So Lemma 1 is proved.

Mathematical Problems of the First International Student's Olympiad in Cryptography NSUCRYPTO

Sergey Agievich   Anastasia Gorodilova   Nikolay Kolomeec
Svetla Nikova   Bart Preneel   Vincent Rijmen
George Shushuev   Natalia Tokareva   Valeriya Vitkup

Invited talk

Abstract

Mathematical problems of the first international student's Olympiad in cryptography NSUCRYPTO'2014 are considered. We discuss solutions of the problems related to cipher constructing such as studying of differential characteristics of S-boxes, S-box masking, determining of relations between cyclic rotation and additions modulo 2 and 2^n, constructing of special linear subspaces in $\mathbb{F}_2^n$; problems about the number of solutions of the equation $F(x) + F(x + a) = b$ over the finite field $\mathbb{F}_{2^n}$ and APN functions. Some unsolved problems in symmetric cryptography are discussed.

Keywords: cryptography, cipher, S-box, APN function, AES, Olympiad.

1 Introduction

The First Siberian Student's Olympiad in Cryptography with International participation — NSUCRYPTO'2014 was held on November 2014. There exist several school competitions in cryptography and information security, but this one is the first cryptographic Olympiad for students and professionals. The aim of the Olympiad was to involve students and young researchers in solving of curious and hard scientific problems of the modern cryptography. From the very beginning the concept was not to stop on the training olympic tasks but include unsolved research problems at intersection of mathematics and cryptography.

In this abstract we would like to discuss several mathematical problems of the Olympiad and their solutions. We consider problems related to cipher constructing such as studying of differential characteristics of S-boxes, S-box masking, determining of relations between cyclic rotation and additions modulo 2 and 2^n, constructing of special linear subspaces in $\mathbb{F}_2^n$. Problems about the number of solutions of the equation $F(x)+F(x+a) = b$ over the finite field $\mathbb{F}_{2^n}$ and APN functions are discussed. Some unsolved problems are proposed such as the problem about the special watermarking cipher.

Organizers of the Olympiad are Novosibirsk State University, Sobolev Institute of Mathematics (Novosibirsk), Tomsk State University, Belarusian State University and University of Leuven (KU Leuven, Belgium). Programm committee was formed by G. Agibalov, S. Agievich, N. Kolomeec, S. Nikova, I. Pankratova, B. Preneel, V. Rijmen and N. Tokareva. Local organizing committee from Novosibirsk consisted of A. Gorodilova, N. Kolomeec, G. Shushuev, V. Vitkup, D. Pokrasenko and S. Filiyzin. N. Tokareva was the general chair of the Olympiad.

More than 450 participants from 12 countries were registered on the website of the Olympiad, `www.nsucrypto.nsu.ru`. There were two independent Internet rounds. The First round (duration 4 hours 30 minutes) was individual and consisted of two sections: school and student's. Theoretical tasks in mathematics of cryptography were offered to participants. The second, team, round (duration 1 week) was devoted to hard research and programming problems of cryptography. Results of the Olympiad can be found on the official website. Here we concentrate our attention only on mathematical problems and solutions.

2 Problem "Linear subspaces"

Recall several definitions and notions. Each element $x \in \mathbb{F}_2^k$ is a binary vector of length k, i. e. $x = (x_1, \ldots, x_k)$, where $x_1, \ldots, x_k \in \mathbb{F}_2$. For two vectors x and y of length k their sum is $x \oplus y = (x_1 \oplus y_1, \ldots, x_k \oplus y_k)$, where $\oplus$ stands for XOR operation. Let $\mathbf{0}$ be the zero element of the vector space, i. e. vector with all-zero coordinates. A nonempty subset $L \subseteq \mathbb{F}_2^k$ is called a *linear subspace* if for any $x, y \in L$ it holds $x \oplus y \in L$. It is easy to see that zero vector belongs to every linear subspace. A linear subspace L of $\mathbb{F}_2^k$ has *dimension* n if it contains exactly 2^n elements.

Problem. For constructing a new secret sharing scheme Mary has to solve the following task on binary vectors. Let n be an integer number, $n \geqslant 2$. Let $\mathbb{F}_2^{2n}$ be a $2n$-dimensional vector space over $\mathbb{F}_2$, where $\mathbb{F}_2 = \{0, 1\}$ is a prime field of characteristic 2. Do there exist subsets $L_1, \ldots, L_{2^n+1}$ of $\mathbb{F}_2^{2n}$ such that the following conditions hold

- L_i is a linear subspace of dimension n for every $i \in \{1, \ldots, 2^n + 1\}$;
- $L_i \cap L_j = \{\mathbf{0}\}$ for all $i, j \in \{1, \ldots, 2^n + 1\}$, $i \neq j$;
- $L_1 \cup \ldots \cup L_{2^n+1} = \mathbb{F}_2^{2n}$?

If "yes", show how to construct these subspaces for an arbitrary integer n.

Solution. Consider $\mathbb{F}_2^{2n}$ as 2-dimensional vector space over $\mathbb{F}_{2^n}$, where $\mathbb{F}_{2^n}$ is the Galois field of order 2^n. Denote this vectorial space as V.

Define the following family of sets:

$$L_\alpha = \{(x, \alpha x) \mid x \in \mathbb{F}_{2^n}\}, \text{ where } \alpha \in \mathbb{F}_{2^n}; \quad L_{2^n+1} = \{(0, y) \mid y \in \mathbb{F}_{2^n}\}.$$

It is obvious that every such a set is a linear subspace in V and contains exactly 2^n elements. Let us show that an arbitrary element $(x, y) \in V$ is covered by the union of these subspaces. If $x = 0$ it is covered by L_{2^n+1}. Otherwise, $(x, y) = (x, (y/x)x)$ belongs to the subspace $L_{y/x}$. Note that every two subspaces have only one common element $\mathbf{0}$, since cardinality of V is exactly $2^{2n} = (2^n + 1)(2^n - 1) + 1$. Thus, the answer for the task is "yes" and the system is constructed.

3 Problem "Number of solutions"

Problem. Let $\mathbb{F}_{256}$ be the finite field of characteristic 2 with 256 elements. Consider the function $F : \mathbb{F}_{256} \to \mathbb{F}_{256}$ such that $F(x) = x^{254}$. Since $x^{255} = 1$ for all nonzero $x \in \mathbb{F}_{256}$, we have $F(x) = x^{-1}$ for all nonzero elements of $\mathbb{F}_{256}$. Further, we have $F(0) = 0$.

Alice is going to use the function F as an S-box (that maps 8 bits to 8 bits) in a new block cipher. But before she wants to find answers to the questions below. Please, help to Alice!

- How many solutions may the equation $F(x+a) = F(x) + b$ have for all different pairs of nonzero parameters a and b, where $a, b \in \mathbb{F}_{256}$?
- How many solutions does the equation $F(x+a) = F(x) + b$ have for the function $F(x) = x^{2^n-2}$ over the finite field $\mathbb{F}_{2^n}$ for an arbitrary n?

Solution. Consider a general case, i.e. $F : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$, $F(x) = x^{2^n-2}$ (and $F(x) = x$ for $n = 1$). We should determine how many solutions the equation $F(x+a) = F(x) + b$ may have for all different pairs of nonzero parameters a and b, where $a, b \in \mathbb{F}_{2^n}$. Since characteristic of $\mathbb{F}_{2^n}$ is 2, operations "$-$" and "$+$" coincide.

First of all, note that for $n = 1$ there exists only one pair $(a, b) = (1, 1)$, in this case there are 2 solutions of the equation. In what follows let $n > 1$.

Note that the function $F_a(x) = F(x) + F(x+a)$ has some symmetry: $F_a(x) = F_a(x+a)$, it means that 2 divides the number of solutions and at least for 2^{n-1} distinct $b \in \mathbb{F}_{2^n}$ it holds $F_a(x) \neq b$ for all $x \in \mathbb{F}_{2^n}$.

Therefore, for all $n > 1$ there exist $b \neq 0$ and $a \neq 0$, such that the equation has no solutions. That is why the number of solutions of the equation $F(x+a) = F(x) + b$ can be 0. Consider other possibilities.

Simplify the given equality: suppose that $x \neq 0$ and $x \neq a$. Note also that $y^{2^n-1} = 1$ for all $y \in \mathbb{F}_{2^n}^*$. Then

$$\begin{aligned}
(x+a)^{2^n-2} + x^{2^n-2} &= b, && \mid \cdot\, x(x+a)\\
x(x+a)(x+a)^{2^n-2} + x(x+a)x^{2^n-2} &= bx(x+a), &&\\
bx^2 + abx + a &= 0, && \mid \cdot\, a^{-2}b^{-1}\\
x^2/a^2 + x/a + (ab)^{-1} &= 0. &&
\end{aligned}$$

Note that the number of solutions of $x^2/a^2 + x/a + (ab)^{-1} = 0$ depends only on ab and $x = 0$, $x = a$ are not its solutions. Rewrite this equality as

$$z^2 + z + (ab)^{-1} = 0,$$

where $z = x/a$. Then we have two cases:

- $x = 0$ and $x = a$ are solutions of $F(x) + F(x + a) = b$. Then it should be $a = b^{-1}$, i. e. $ab = 1$. We already have 2 solutions. Next, solve the equality $z^2 + z + 1 = 0$. Note that the number of its solutions plus 1 equals to the number of solutions of $z^3 + 1 = 0$. Below we determine this last number. Let α be a primitive element of $\mathbb{F}_{2^n}$.
  - If n is even, then α^3 is not primitive as far as $2^n - 1 = 2^{2k} - 1 = 4^k - 1 = 0 \pmod 3$, i. e. we have more than one solution for $z^3 + 1 = 0$. It means that $z^2 + z + 1 = 0$ has solutions, moreover, exactly two, since z and $z + 1$ are both solutions. Therefore, the equation $F(x) + F(x + a) = b$ has exactly 4 solutions.
  - If n is odd, α^3 is a primitive element, since $2^n - 1 = 2 \cdot 2^{2k} - 1 = 2 \cdot 4^k - 1 = 1 \pmod 3$, i. e. $z^3 + 1$ has exactly one solution. It means that $z^2 + z + 1 = 0$ has no solutions. So, the equation $F(x) + F(x + a) = b$ has exactly 2 solutions.
- $x = 0$ and $x = a$ are not the solutions of $F(x) + F(x + a) = b$. Therefore, $ab \notin \mathbb{F}_2$. Note that equation $z^2 + z = z(z + 1) = (ab)^{-1}$ have 0 or 2 solutions. Since $(ab)^{-1}$ can be an arbitrary element from $\mathbb{F}_{2^n} \setminus \mathbb{F}_2$, at least for $2^{n-1} - 2$ distinct ab there are exactly two solutions: so, if $n > 2$, it can be 2 solutions in this case. If $n = 2$, then $z^2 + z \in \mathbb{F}_2$, i. e. for $ab \notin \mathbb{F}_2$ there is no solution.

The answer is the following. For $n = 1$ there are always 2 solutions; for $n = 2$ there can be 0 or 4 solutions; for odd n ($n > 1$) there can be 0 or 2 solutions; for even n ($n > 2$) there can be 0, 2 or 4 solutions.

4 Problem "A special parameter"

A special parameter that we consider here is called the *differential branch number* of a transformation, see for example book [2]. In differential cryptanalysis of block ciphers this parameter is used to measure the diffusion strength of a cipher. Some properties of it are discussed in the problem.

Problem. Let n, m be positive integer numbers. Let $a = (a_1, \ldots, a_m)$ be a vector, where a_i are elements of the finite field $\mathbb{F}_{2^n}$. Denote by $\mathrm{wt}(a)$ the number of nonzero coordinates a_i, $i = 1, \ldots, m$, and call this number the *weight* of a. We say that $a, b \in \mathbb{F}_{2^n}^m$ represent *states*. The sum of two states a, b is defined as $a + b = (a_1 + b_1, \ldots, a_n + b_n)$.

Thus, the *special parameter* P of a function $\varphi : \mathbb{F}_{2^n}^m \to \mathbb{F}_{2^n}^m$ is given by

$$P(\varphi) = \min_{a,\, b,\, \text{such that } a \neq b} \{\mathrm{wt}(a + b) + \mathrm{wt}(\varphi(a) + \varphi(b))\}.$$

- Rewrite (simplify) the definition of $P(\varphi)$ when the function φ is linear (recall that a function ℓ is linear if for any x, y it holds $\ell(x + y) = \ell(x) + \ell(y)$).
- Rewrite the definition of $P(\varphi)$ in terms of linear codes, when the linear transformation φ is given by a $m \times m$ matrix M over $\mathbb{F}_{2^n}$, i. e. $\varphi(x) = M \cdot x$.
- Let φ be an arbitrary function. Find a tight upper bound for $P(\varphi)$ as a function of m.
- Can you give an example of the function φ with the maximal possible value of P?

Solution. Suppose $\varphi : \mathbb{F}_{2^n}^m \to \mathbb{F}_{2^n}^m$ and $a, b \in \mathbb{F}_{2^n}^m$.

• Since φ is linear, i. e. $\varphi(x + y) = \varphi(x) + \varphi(y)$ for all $x, y \in \mathbb{F}_{2^n}^m$, and the condition $a \neq b$ is equivalent to $a + b \neq 0$, we can rewrite the definition of $P(\varphi)$ in the following way, where by c we denote $a + b$:

$$P(\varphi) = \min_{c \neq 0} \{\mathrm{wt}(c) + \mathrm{wt}(\varphi(c))\}.$$

• Let us consider vectors $(x, \varphi(x)) = (x, M \cdot x)$ of length $2m$, where $x \in \mathbb{F}_{2^n}^m$. Then the set $C = \{(x, M \cdot x) \mid x \in \mathbb{F}_{2^n}^m\}$ is a linear code. Since we have $\mathrm{wt}(a + b) + \mathrm{wt}(\varphi(a) + \varphi(b)) = \mathrm{wt}((a + b, \varphi(a) + \varphi(b))) = \mathrm{dist}((a, \varphi(a)), (b, \varphi(b)))$, where $\mathrm{dist}(x, y)$ means the Hamming distance between vectors x and y, the parameter $P(\varphi)$ is equal to the minimal distance between distinct codewords of the code C.

• Since $a \neq b$, the minimal value of $\mathrm{wt}(a + b)$ is equal to 1. Also the maximal possible value $\mathrm{wt}(\varphi(a) + \varphi(b))$ is m by the definition. Thus, the maximal possible value of $P(\varphi)$ is not more than $m + 1$.

• One can construct an example of such a transformation using a Maximal Distance Separable code with parameters $[2m, m, m+1]$, for example Reed — Solomon code.

5 Problem "S-box masking"

Problem. To provide the security of a block cipher to the side channel attacks, some ideas on masking of elements of the cipher are exploited. Here we discuss masking of S-boxes. Alice takes a bijective function S (S-box) that maps n bits to n bits. Bob claims that for every such a function S there exist two bijective S-boxes, say S' and S'', mapping n bits to n bits, such that it holds $S(x) = S'(x) \oplus S''(x)$ for all $x \in \mathbb{F}_2^n$. Hence, Alice is able to mask an arbitrary bijective S-box by "dividing it into parts" for realization. But Alice wants to see the proof of this fact. Please help to Bob in giving the arguments.

Solution. We would like to give a solution proposed by Qu L. et al. in the first variant of the paper [4]. Let us represent an arbitrary bijective function $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$ as $S'(x) \oplus S''(x)$, where $S', S'' : \mathbb{F}_2^n \to \mathbb{F}_2^n$ are bijective too. As for "Linear subspaces" problem, consider $\mathbb{F}_2^n$ as $\mathbb{F}_{2^n}$. Let $\alpha \in \mathbb{F}_{2^n}$ and $\alpha \neq 0, 1$. If $n > 1$ such an element α does exist. It is clear that $S(x) = \alpha S(x) + (\alpha + 1)S(x)$. Note that $\alpha S(x)$ and $(\alpha + 1)S(x)$ are bijective since each of them is a composition of two bijective mappings. So, for $n > 1$ the required representation exists. If $n = 1$ there are only two bijective function $S(x) = x$ and $S(x) = x \oplus 1$; their sum is a constant and hence there is no the required representation in this case.

6 Problem "Add-Rotate-Xor"

Problem. Let $\mathbb{F}_2^n$ be the vector space of dimension n over $\mathbb{F}_2 = \{0, 1\}$. A vector $x \in \mathbb{F}_2^n$ has the form $x = (x_1, x_2, \ldots, x_n)$, where $x_i \in \mathbb{F}_2$. This vector can be interpreted as the integer $x_1 \cdot 2^{n-1} + x_2 \cdot 2^{n-2} + \ldots + x_{n-1} \cdot 2 + x_n$.

Alice can implement by hardware the following functions from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ for all vectors $a, b \in \mathbb{F}_2^n$ and all integers r, $0 < r < n$:

1) $f_a(x) = x \boxplus a$ — addition of vectors x and a as integers modulo 2^n;

2) $g_r(x) = x \lll r$ — cyclic rotation of a vector x to the left by r positions for a fixed positive integer r;

3) $h_b(x) = x \oplus b$ — coordinate-wise sum of vectors x and b modulo 2.

- Bob asks Alice to construct two devices that compute the functions S_1 and S_2 from $\mathbb{F}_2^2$ to $\mathbb{F}_2^2$ given by their truth tables:

x	(00)	(01)	(10)	(11)
$S_1(x)$	(01)	(00)	(10)	(11)
$S_2(x)$	(01)	(11)	(00)	(01)

  Can Alice do it? If "yes", show how it can be done; if "no", give an explanation.

- Generalizing the problem above: can we construct any function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ using only a finite number of compositions of functions f_a, g_r and h_b? And what about any permutation over $\mathbb{F}_2^n$? Consider at least the cases $n = 2, 3, 4$.

- Is it possible to compute every function h_b using only f_a and g_r?

Solution. The complicated algebraic solution of this problem was given by T. Zieschang in 1997, see [5]. Here we introduce a simple solution proposed by the participants of the Olympiad.

- $S_1(x) = g_1(f_1(g_1(f_2(x))))$. It is obvious that all f_a, g_r and h_b are bijective, therefore, any its composition is bijective too. S_2 is not bijective, so, it can not be represented as a composition of them.

- Only permutations on $\mathbb{F}_{2^n}$ can be constructed in this way. It is well known that compositions of function f_1 (a cycle of length 2^n) and transpositions of adjacent elements in the cycle give us all permutations on $\mathbb{F}_{2^n}$. The following construction gives a certain transposition τ: $\tau(x) = g_1(f_{2^n-1}(g_{n-1}(f_2(x))))$. Indeed, $g_{n-1}(y) = 2^{n-1}y_n + \lfloor y/2 \rfloor$; and if $x < 2^n - 2$, then $f_2(x) = x + 2$, so,

$$g_{n-1}(f_2(x)) = g_{n-1}(x+2) = 2^{n-1}(x+2)_n + \lfloor (x+2)/2 \rfloor =$$
$$2^{n-1}x_n + \lfloor x/2 \rfloor + 1.$$

Next, f_{2^n-1} eliminates "+1" and g_1 cyclically rotates $(x_n, x_1, \ldots, x_{n-1})$ to the left by one position. Also $\tau(2^n-2) = 2^n-1$: $f_2(2^n-2) = 0$, $g_{n-1}(0) = 0$, $f_{2^n-1}(0) = 2^n-1$, $g_1(2^n-1) = 2^n-1$.

- Yes. The third item is obvious, since we can construct any permutation on $\mathbb{F}_{2^n}$ using the mentioned functions.

7 Problem "Watermarking cipher" (unsolved)

Here we present an unsolved problem that remains so till now. It was stated by Gennadiy Agibalov and Irina Pankratova.

The most deep analysis of this problem was proposed by R. Zhang and A. Luykx (winners of the second round of NSUCRYPTO in category "professional"), but nobody has introduced a concrete solution. May be you can do it?

Problem. Let X, Y and K be the sets of plaintexts, ciphertexts and keys respectively, where $X = Y = \{0,1\}^n$ and $K = \{0,1\}^m$ for some integer n and m. Recall that two functions $E : X \times K \to Y$ and $D : Y \times K \to X$ are called *an encryption algorithm* and *a decryption algorithm* respectively if for any $x \in X$, $k \in K$ it holds $D(E(x,k),k) = x$. Together E and D form *a cipher*. Let us call a cipher *watermarking* if for any key $k \in K$ and any subset $I \subseteq \{1,2,\ldots,n\}$ there exists a key k_I such that for any $x \in X$ it holds $D(E(x,k),k_I) = x'$, where x' is obtained from x by changing all bits with coordinates from I. So, the problem is to construct a watermarking cipher. Please think about easy usage of it by users.

A simple example of such a cipher. Let $m = n$ and encryption and decryption algorithms be the following: $E(x,k) = x \oplus k$ and $D(y,k) = y \oplus k$. For any set I and any key k we can easily get the key k_I that is obtained from k by changing all bits with coordinates from I. The main disadvantage of such a cipher that every key should be used only once. **How can we use a watermarking cipher?** Suppose you own some digital products (for example, videos), which you want to sell. Let x represent a binary code of a product. For each customer of x you choose the unique set I of coordinates and send to him the encrypted with the key k copy y and the correspondent key k_I. Then after receiving y and k_I the customer decrypts y and gets x'. The difference between the original x and x' is not significant; thus the customer does not know about it. If someone illegally spreads on the Internet bought by him product, you can easily understand who do it because you choose the unique set I for each customer!

8 Problem "APN permutation" (unsolved)

This is another unsolved problem of the Olympiad; it is the well known long standing problem of cryptography. Some ideas on it were proposed by the team of G. Beloshapko, A. Taranenko and E. Fomenko (winners of the second round of NSUCRYPTO in category "students"). One participant has proposed an online service for distributed search of APN permutations, see [6]. But till now this problem is unsolved.

Problem. Suppose we have a mapping F from $\mathbb{F}_2^n$ to itself (recall that $\mathbb{F}_2^n$ is the vector space of all binary vectors of length n). This mapping is called a *vectorial Boolean function in n variables.* Such functions are used, for example, as S-boxes in block ciphers and should have special cryptographic properties. In this problem we consider the following two properties and the problem of combining them.

- A function F in n variables is a *permutation* if for all distinct vectors $x, y \in \mathbb{F}_2^n$ it has distinct images, i. e. $F(x) \neq F(y)$.
- A function F in n variables is called *Almost Perfect Nonlinear* (APN) if for any nonzero vector $a \in \mathbb{F}_2^n$ and any vector $b \in \mathbb{F}_2^n$ an equation $F(x) \oplus F(x \oplus a) = b$ has at most 2 solutions. Here $\oplus$ is the coordinate-wise sum of vectors modulo 2.

Find an APN permutation in 8 variables or prove that it does not exist.

History of the problem. The question "Does there exist an APN permutation in even number of variables?" has been studied for more that 20 years. If the number of variables is odd, APN permutations exist as it was proved by K. Nyberg in 1994, see [3]. It is known that for 2 and 4 variables the answer is "No". But for 6 variables K. Browning, J. F. Dillon, M. McQuistan, and A. J. Wolfe have found such a function in 2009, see [1]. You can see it bellow:

$$G = \left(\begin{array}{llllllllllllllll} 0 & 54 & 48 & 13 & 15 & 18 & 53 & 35 & 25 & 63 & 45 & 52 & 3 & 20 & 41 & 33 \\ 59 & 36 & 2 & 34 & 10 & 8 & 57 & 37 & 60 & 19 & 42 & 14 & 50 & 26 & 58 & 24 \\ 39 & 27 & 21 & 17 & 16 & 29 & 1 & 62 & 47 & 40 & 51 & 56 & 7 & 43 & 44 & 38 \\ 31 & 11 & 4 & 28 & 61 & 46 & 5 & 49 & 9 & 6 & 23 & 32 & 30 & 12 & 55 & 22 \end{array} \right).$$

This function is presented as the list of its values, i. e. $G(0) = 0$, $G(4) = 15$, $G(16) = 59$ and so on. For brevity we use integers instead of binary vectors. A binary vector $x = (x_1, \dots, x_n)$ corresponds to an integer $k_x = x_1 \cdot 2^{n-1} + x_2 \cdot 2^{n-2} + \dots + x_{n-1} \cdot 2 + x_n$.

9 Problem "Super S-box" (unsolved)

The next unsolved problem is directly related to AES construction. J. Daemen and V. Rijmen, the designers of AES (Rijndael), have introduced the Super-Sbox representation of two rounds of AES in order to study differential properties, see [2]. The function G from the problem can be considered as a simplified Super-Sbox model of two rounds of AES. To study resistance of AES to differential cryptanalysis, we welcome you to start with differential characteristics of the function G.

Problem. Let $\mathbb{F}_{256}$ be the finite field of 256 elements and α be a primitive element (i. e. for any nonzero $x \in \mathbb{F}_{256}$ there exists $i \in \mathbb{N}$ such that $x = \alpha^i$). Let $\mathbb{F}_{256}^4$ be the vector space of dimension 4 over $\mathbb{F}_{256}$. An arbitrary function from $\mathbb{F}_{256}^4$ to $\mathbb{F}_{256}^4$ can be considered as the set of 4 coordinate functions from $\mathbb{F}_{256}^4$ to $\mathbb{F}_{256}$. Define the following auxiliary functions $F_4, M : \mathbb{F}_{256}^4 \to \mathbb{F}_{256}^4$:

$$F_4(x_1, x_2, x_3, x_4) = (x_1^{254}, x_2^{254}, x_3^{254}, x_4^{254});$$

$$M(x_1, x_2, x_3, x_4) = (x_1, x_2, x_3, x_4) \times \begin{bmatrix} \alpha+1 & 1 & 1 & \alpha \\ \alpha & \alpha+1 & 1 & 1 \\ 1 & \alpha & \alpha+1 & 1 \\ 1 & 1 & \alpha & \alpha+1 \end{bmatrix}.$$

Consider the function $G : \mathbb{F}_{256}^4 \to \mathbb{F}_{256}^4$ that is a combination of F_4 and M: $G(x_1, x_2, x_3, x_4) = F_4(M(F_4(x_1, x_2, x_3, x_4)))$. Find the number of solutions of the equation $G(x + a) = G(x) + b$, where parameters a and b run trough all nonzero vectors from $\mathbb{F}_{256}^4$.

Notes to solution. The problem is still unsolved. Only one team of G. Beloshapko, A. Taranenko and E. Fomenko from Novosibirsk State University has sent a solution with an analysis of the problem for smaller field. They considered $\mathbb{F}_{16}$ and found the exact number of pairs a, b for each number of solutions. It seems that any even number between 0 and 44 can be the number of solutions. They proposed a hypothesis that for $\mathbb{F}_{256}$ it is true the same result: it may be any even number of solutions bounded by some number.

10 Conclusion

We thank Novosibirsk State University for the financial support of the Olympiad and invite you to take part in the next NSUCRYPTO that starts on November 15, 2015. Your ideas on the mentioned unsolved problems are also very welcome and can be sent to `olymp@nsucrypto.ru`.

We are very grateful to Gennadiy Agibalov and Irina Pankratova for their valuable contribution to this paper. A. Gorodilova, N. Kolomeec, G. Shuhuev, N. Tokareva and V. Vitkup would like to thank RFBR (grant 15-31-20635).

References

[1] *Browning K. A., Dillon J. F., McQuistan M. T., Wolfe A. J.* An APN Permutation in Dimension Six. // Post-proceedings of the 9-th International Conference on Finite Fields and Their Applications Fq'09, Contemporary Math., AMS, v. 518, 2010. P. 33–42.

[2] *Daemen J., Rijmen V.* The Design of Rijndael: AES — The Advanced Encryption Standard. Springer, 2002. 238 pages.

[3] *Nyberg K.* Differentially uniform mappings for cryptography // Eurocrypt 1993, LNCS, 1994, V. 765, N 2, P. 55–64.

[4] *Qu L., Fu S., Dai Q., Li C.* When a Boolean Function can be Expressed as the Sum of two Bent Functions // Cryptology ePrint Archive. 2014/048.

[5] *Zieschang T.* Combinatorial Properties of Basic Encryption Operations. Eurocrypt, 1997.

[6] `http://writeupsd.blogspot.ru/2014/11/apn-permutation-finder.html.`

Some Remarks On Elliptic Curve Discrete Logarithm Problem

Alexey Nesterenko

Let $p > 3$ be a prime and $\mathcal{E}$ be an elliptic curve, defined over field $\mathbb{F}_p$ in short Weierstrass form

$$\mathcal{E} : \quad v^2 \equiv u^3 + Au + B \pmod{p},$$

where $4A^3 + 27B^2 \not\equiv 0 \pmod{p}$. Let we have a point $P \in \mathcal{E}$ with prime order q, which forms a subgroup $G = \langle P \rangle \in \mathcal{E}$.

For every point $Q \in G$ we can define an elliptic curve discrete logarithm problem (ECDLP). In this problem we need to find an integer $x \in \mathbb{F}_q^*$, such that

$$Q = [x]P = \underbrace{P + \cdots + P}_{x}. \tag{1}$$

For solving this problem, see [1], we can use Pollard's Rho and Lambda methods or parallel algorithm, introduced by van Oorschot and Wiener, see [2, 4]. The running time of these algorithms is $O(\sqrt{q})$ and doesn't depend on x.

In this paper we present a new algorithm, based on ideas of Pollard, van Oorschot and Wiener, which has a running time, depending on x.

The main result is follows. If we know an multiplicative order of x in $\mathbb{F}_q^*$, i.e. if we know r, such

$$r | q - 1 \quad \text{and} \quad x^r \equiv 1 \pmod{q}, \tag{2}$$

then we can find x with $O(\sqrt{r}\log q)$ running time.

1 Backgound

Let g be a primitive root modulo q and $r|q-1$. Define

$$a \equiv g^{\frac{q-1}{r}} \pmod{q},$$

where $a^k \not\equiv 1 \pmod{q}$ for any $k|r$, $k > 1$.

It's well know, that for every x with order r, there exists an integer n, $0 < n < r$, such that $x \equiv a^n \pmod{q}$. Hence, we can write

$$Q = [x]P = [a^n]P \tag{3}$$

and now, we need to find n.

Define $h = \lceil \sqrt{r} \rceil$ and write $n = n_1 h - n_0$, where $0 \le n_0 < h$, $0 < n_1 \le h$. Hence, we can write (3) as follows

$$[(a^h)^{n_1}]P = [a^{n_0}]Q,$$

and using "baby steps, giant steps", see [5], find values of n_0 and n_1.

Since the evaluation of $[a^\xi]P$ has running time $O(\log q)$ for every $\xi \in \mathbb{F}_q^*$ and values of n_0, n_1 is not greater than $\sqrt{r}$, we have running time of "baby steps, giant steps" algorithm is $O(\sqrt{r}\log q)$.

2 The algorithm

Since "baby steps, giant steps" algorithm uses a lot of memory, in practical evaluation we can use a modification of Pollard's Lambda method, see [4, 6]. This modification directly finds a solution of (1).

Let $s = \lceil \log_2 r \rceil$, and choose a random numbers $\xi_0, \dots, \xi_{s-1}$ such that $0 < \xi_i < r$, $i = 0, \dots, s-1$. Define a map $f : \mathcal{E} \to \mathcal{E}$. For any point $R = (x_R, y_R) \in \mathcal{E}$ let

$$f(R) = [\zeta_i]R, \quad \text{where} \quad i \equiv x_R \pmod{s}, \quad \zeta_i \equiv a^{\xi_i} \pmod{q}. \tag{4}$$

On the first step, we construct a sequence of points

$$R_{k+1} = f(R_k), \quad R_k \in \mathcal{E}, \quad R_0 = P, \quad k = 0, 1, \dots. \tag{5}$$

For every point we have

$$R_{k+1} = [\zeta_k]R_k = [\zeta_k\zeta_{k-1}]R_{k-1} = \cdots = [\mu_{k+1}]P,$$

where $\mu_{k+1} \equiv \prod_{j=0}^{k} \zeta_j \pmod q$ and ζ_j defined by (4).

Starting with k_0, we can keep all points $R_{k_0+1}, \ldots, R_{k_0+b}$ for some natural number b, with corresponding values $\mu_{k_0+1}, \ldots, \mu_{k_0+b}$. This set $\mathcal{S}$ of keeping points is a trap-set with cardinality $|\mathcal{S}| = b$. The exact values of k_0, b can be found in [6].

On the last step of algorithm, we shall construct a sequence of points $U_0, U_1, \ldots$, starting from a random point $U_0 = [a^\xi]Q$, where ξ is a random integer, $0 < \xi < r$. Our sequence consists of points

$$U_{k+1} = f(U_k), \quad U_k \in \mathcal{E}, \quad k = 0, 1, \ldots, \tag{6}$$

and every point U_k in this sequence can be represented as

$$U_{k+1} = [\zeta_k]U_k = [\zeta_k\zeta_{k-1}]U_{k-1} = \cdots = [\nu_{k+1}]Q,$$

where $\nu_{k+1} \equiv a^\xi \prod_{j=0}^{k} \zeta_j \pmod q$,

For every point U_k, $k = 1, 2, \ldots$, we can check a condition $U_k \in \mathcal{S}$, where $\mathcal{S}$ is a trap-set. If condition holds, then we found some index j such

$$[\mu_j]P = R_j = U_k = [\nu_k]Q, \tag{7}$$

hence, $\mu_j \equiv \nu_k x \pmod q$ and

$$x \equiv \mu_j \nu_k^{-1} \pmod q. \tag{8}$$

is a solution of (1). If our condition does'nt hold for all $U_1, \ldots, U_h$, we need to chose another value of ξ and repeat this step another one.

It's easy to see, that algorithm runs succesfully only if a set of points

$$[a]P, \quad [a^2]P, \quad \ldots, \quad [a^{r-1}]P, \quad [a^r]P = P$$

contains the point Q. This is equivalent to condition (2).

3 Parallel version of the algorithm

Our algorithm can be easily modificated for parallel computations. Consider a parallel paradigm of evaluation. Let we have a $2w$ threads with common memory and every thread can produce evaluation independently.

We fix an integer b and divide all set of threads on two groups, say $\mathcal{P}$ and $\mathcal{Q}$, with same cardinality w and common set of trap-points, $S_{\mathcal{P}}$ and $S_{\mathcal{Q}}$ correspondingly.

Every thread of first group $\mathcal{P}$ can produce evaluation as follows: choose a random point $R_0 = [\zeta_0]P$, where $\zeta_0 \equiv a^{\xi} \pmod{q}$ and ξ is a random integer, $0 < \xi < r$, and construct a sequence

$$R_k = f(R_{k-1}), \quad R_k = (x_{R,k}, y_{R,k}) \in \mathcal{E}, \quad k = 1, 2, \ldots,$$

where map f defined by (4). As before, for every point we have

$$R_k = [\mu_k]P, \quad \mu_k \equiv \prod_{j=0}^{k} \zeta_k \pmod{q}.$$

For every point R_k thread checks the condition $x_{R,k} \equiv 0 \pmod{b}$. If condition holds, thread searches the point R_k in $S_{\mathcal{Q}}$ set of trap-points. If he doesn't find this point in $S_{\mathcal{Q}}$, then he adds this point to $S_{\mathcal{P}}$ set.

In a similar way, every thread in a second group $\mathcal{Q}$ evaluates a sequence

$$U_k = [\nu_k]Q, \quad U_k = (x_{U,k}, y_{U,k}), \quad \nu_k \equiv \prod_{j=0}^{k} \eta_k \pmod{q},$$

wich started from a random point $U_0 = [a^{\xi}]Q$ with random integer $0 < \xi < r$.

For every point U_k thread checks the condition $x_{U,k} \equiv 0 \pmod{b}$. If condition holds, thread searches this point in $S_{\mathcal{P}}$ set of trap-points. If he doesn't find the point U_k in $S_{\mathcal{P}}$, then he adds this point to $S_{\mathcal{Q}}$ set.

In this manner, we can evaluate a w sequences of points, which started from point $P \in \mathcal{E}$ and w sequences, which started from point $Q \in \mathcal{E}$. All trap-points in $S_{\mathcal{P}}$ and $S_{\mathcal{Q}}$ sets satisfy the condition $x \equiv 0 \pmod{b}$, where

x is a x-coordinate of point. When the thread finds a trap-point, then immediately we have an equality (7) and can find a solution, using (8).

As an example, we can demonstrate a results of practical realization of our parallel algorithm. Consider the elliptic curve $\mathcal{E}$ defined by equation

$$v^2 \equiv u^3 - 2u + 83161154912977162385779023371676872267 \pmod{p},$$

where $p = 170144519623114011343322490539658030031$ is prime and holds $\lceil \log_2 p \rceil = 128$. This curve has on order $2q$, where q is prime and

$$\begin{aligned} q &= 85072259811557005664489355982895124223, \\ q-1 &= 2 \cdot 3 \cdot 139 \cdot 643 \cdot 12281 \cdot 51593 \cdot 53887 \cdot 4646248420547414411. \end{aligned}$$

We need to find an integer $x \in \mathbb{F}_q^*$, such $Q = [x]P$, where

$$P = (127079991335379663215392766670928701845, \\ 146478651337885753760004547813245055780),$$

$$Q = (135768511924514909185266977775889591902, \\ 104833826712192434308111206165615249692),$$

$|\langle P \rangle| = q$ and $Q \in \langle P \rangle$. Additionally, we know, that multiplicative order of x in $\mathbb{F}_q^*$ are equal to $r = 34143537841471 = 12281 \cdot 51593 \cdot 53887$, $\lceil \log_2 r \rceil = 46$.

Firstly, we find a generator $a \in \mathbb{F}_q^*$, which has order r

$$a = 69038627934287400758544579548029658020$$

and construct a 46 values $\zeta_1, \ldots, \zeta_{46} \in \mathbb{F}_q^*$, where $\zeta_i \equiv a^{\xi_i} \pmod{q}$ is a random integers for all $i = 1, \ldots, 46$ and $0 < \xi_i < r$. For decreasing complexity of evaluation, we choose a random integers ξ_i with bounded hamming weight of values ζ_i, say $w(\zeta_i) < 40$, where hamming weight defined by

$$w(\zeta_i) = \sum_{j=0}^{127} c_{i,j}, \quad \zeta = \sum_{j=0}^{127} c_{i,j} 2^j, \quad c_{i,j} \in \{0,1\}, \quad i = 1, \ldots, 46.$$

This simple trick gives us a 25% of speed up.

After this, our personal computer (AMD processor with 8 cores, 2.5Hz) calculated a 8 sequences for $b = 1024$. When we find a two points R_j, U_k satisfying an equalify (7)

$$R_j = [178574163682345398152859188282742 5715]P = [392246999408260179955993732826343 0847]Q = U_k,$$

for some values of indexes j and k, we immediately calculate

$$x = 111993268900250420930399627452392 77210.$$

The time of all calculations is 224 seconds, the cardinality of $S_{\mathcal{P}}$ was 2270 and cardinality of $S_{\mathcal{Q}}$ was 2356.

4 Conclusion

In this paper we present an algorithm, which is not useful for real cryptosystems, because we don't know in practice a multiplicative order of x. From the other side, we know the value of q. Hence, we can find all factors of $q - 1$ and for every factor r check, if x has an order r or if it hasn't.

If x is a primitive root modulo q, then we have a maximal complexity of our algorithm $O(\sqrt{q}\log_2 q)$, which is a bit more than complexity of Pollard and van Oorschot-Wiener methods.

References

[1] *Blake I., Seroussi G., Smart N.* Elliptic Curves In Cryptography. — London Mathematical Society Lecture Notes. — Vol. 265 — Cambridge University Press. — 1999.

[2] *van Oorschot P.C., Wiener M.J.* Parallel collision search with cryptanalytic applications // Journal of Cryptology. — Vol.12. — №. 1. — Jan. 1999. — pp.1-28.

[3] *Pollard J.M.* Theorems on Factorization And Primality Testing // Proceedings of the Cambridge Philosophical Society. — №. 3. — Vol. 76. — 1974. — pp. 521-528.

[4] *Pollard J.M.* Monte Carlo methods for index computation ($mod\ p$) // Mathematics Of Computation. — №. 143. — Vol. 32. — 1978. — pp. 918-924.

[5] *Shanks D.*, Class number, a theory of factorization and genera. — Proc. Symposium Pure Math. — Vol 20. — 1971.

[6] *Teske E.* Square-root algorithms for the discrete logarithm problem (a survey) // Public-key cryptography and computational number theory (Warsaw, 2000). — Berlin. — 2001. — pp. 283-301.

Approximate Common Divisor Problem and Continued Fractions

Kirill Zhukov

Abstract

In this paper we describe two algorithms for computing common divisors of two integers, one of them given inaccurate. We generalize known method based on the continued fraction technique. In some cases new algorithms overcome the strongest algorithm based on Coppersmith's method: a less accurate approximation is recurred for computing a divisor.

Keywords: approximate common divisors, continued fractions, Diophantine approximation

1 Introduction

Let us describe a partially approximate common divisor problem (PACDP). Consider two integers N_1 and N_2. Let A be a nontrivial common divisor of N_1 and $N_2 - \Delta$ for some small integer Δ. The goal is to find A provided that it is big.

The PACDP was introduced in 2001 by N. Howgrave-Graham [3] who used the continued fraction techniques and Coppersmith's method.

An algorithm of S. Sarkar and S. Maitra [4] for solving the PACDP is known to be the strongest. Based on Coppersmith's method this algorithm finds a common divisor in time which is polynomial of $n = \max\{[\ln N_1], [\ln N_2]\}$ provided that $l_A > l_{\frac{N_1}{A}} + l_\Delta - \frac{l_{B_1}^2}{l_{N_1}}$, where l_A, $l_{\frac{N_1}{A}}$, l_{N_1} and l_Δ are the binary lengths of A, $\frac{N_1}{A}$, N_1 and Δ respectively. In this paper we describe two algorithms for computing common divisor, their time complexity is polynomial of n and c provided that $l_A \geq l_{\frac{N_1}{A}} + l_\Delta + 2 - \log_2 c$.

2 Definitions and notations

Let us sketch the main definitions from [1] related to continued fractions. An expression of the form

$$a_0 + \cfrac{1}{a_1 + \frac{1}{a_2 + \ldots}}, \tag{1}$$

where a_0 is integer, and a_i is positive integer for $i = 1, 2, \ldots$, is called an infinite continued fraction.

As a shorthand for expression (1), a continued fraction is often expressed as $\langle a_0, a_1, a_2, \ldots \rangle$. For $i = 0, 1, \ldots$ we define the i-th convergent of continued fraction (1) to be a finite continued fraction

$$\frac{P_i}{Q_i} = \langle a_0, a_1, a_2, \ldots, a_i \rangle \tag{2}$$

For $i = 2, 3, \ldots$ there are recurrent formulas for numerator and denominators of convergents:

$$\begin{aligned} P_i &= P_{i-1} a_i + P_{i-2}, \\ Q_i &= Q_{i-1} a_i + Q_{i-2}, \end{aligned} \tag{3}$$

with start condition:

$$P_0 = a_0, \quad Q_0 = 1, \quad P_1 = a_0 a_1 + 1, \quad Q_1 = a_1. \tag{4}$$

One can easily show using the method of mathematic induction that for $i = 0, 1, 2, \ldots$ the following equality holds

$$P_{i+1} Q_i - P_i Q_{i+1} = (-1)^i. \tag{5}$$

Any real number α has a continued fraction representation $\alpha = \langle a_0, a_1, a_2, \ldots \rangle$ If $\alpha = \frac{P}{Q}$ is a rational than the continued fraction expansion of α has a finite number of elements: $\alpha = \langle a_0, a_1, a_2, \ldots, a_r \rangle$, $r \in \mathbb{N} \cup \{0\}$. In that case we can compute elements of the continued fraction with Euclidean algorithm for P and Q.

Convergents of α are good rational approximations. For any $i = 0, 1, 2, \ldots$ the following estimates hold

$$\frac{1}{Q_i(Q_i + Q_{i+1})} < \left|\alpha - \frac{P_i}{Q_i}\right| < \frac{1}{Q_i Q_{i+1}} \tag{6}$$

On the other hand Legendre's theorem states that if a rational is a good approximation to some real number, than that rational equals some convergent of this real number.

Theorem 1 ([1], Theorem 245). *Let α be a real number, a and b be nonzero coprime integers ($b > 0$), such that $\left|\alpha - \frac{a}{b}\right| < \frac{1}{2b^2}$. Then $\frac{a}{b}$ is a convergent of α.*

This result has the following generalization.

Theorem 2 ([2], Theorem 1). *Let α be a real number, a and b be nonzero coprime integers ($b > 0$), such that $\left|\alpha - \frac{a}{b}\right| < \frac{c}{b^2}$, where c is a real positive number. Then there exists a pair of adjacent convergents $\frac{P_k}{Q_k}$ and $\frac{P_{k+1}}{Q_{k+1}}$ of α, such that the equality*

$$\frac{a}{b} = \frac{uP_{k+1} \pm vP_k}{uQ_{k+1} \pm vQ_k}$$

holds for some nonnegative integers u and v, $uv < 2c$.

3 Continued fraction methods for the PACDP

Suppose that the integers N_1 and $N_2 - \Delta$ have a nontrivial common divisor A for some integer Δ. Then the integers N_1 and N_2 have representations $N_1 = AB_1$ $N_2 = AB_2 + \Delta$. Howgrave-Graham [3] showed how to find a divisor A using the continued fraction techniques under some bounds on B_1 and Δ. The method is based on a special case of the following lemma.

Lemma 1. *Let A, B_1, B_2 be natural numbers and Δ be an integer. Let $N_1 = AB_1$ and $N_2 = AB_2 + \Delta$. Suppose that the inequality*

$$A > \sqrt{\frac{N_1 |\Delta|}{c}}, \tag{7}$$

holds for some positive real number c. Then:

$$\left|\frac{N_2}{N_1} - \frac{B_2}{B_1}\right| < \frac{c}{B_1^2}. \tag{8}$$

Algorithm 1 (PACDP)

```
Require: $N_1$ and $N_2$ — natural numbers; $c$ — a method's real parameter
Ensure: $S \subset \mathbb{N} \times \mathbb{Z}$ — a set of all pairs $(A, \Delta)$ such that $N_1 = AB_1$, $N_2 = AB_2 + \Delta$ and
    $A > \sqrt{\frac{N_1|\Delta|}{c}}$
1: $S \leftarrow \varnothing$
2: Compute all the convergents $\frac{P_0}{Q_0}, \dots, \frac{P_r}{Q_r}$ of $\frac{N_2}{N_1}$ using Euclidean algorithm and formulas (3), (4)
3: for all $k = 0, 1, \dots, r-1$ do
4:   for all $j = 0, 1$  $v = 1, 2, \dots, [2c]$, $u = 0, 1, \dots, [\frac{2c}{v}]$ do
5:     if $uQ_{k+1} + (-1)^j vQ_k | N_1$ then
6:       $B_1 \leftarrow uQ_{k+1} + (-1)^j vQ_k$, $B_2 \leftarrow uP_{k+1} + (-1)^j vP_k$, $A \leftarrow \frac{N_1}{B_1}$, $\Delta \leftarrow N_2 - AB_2$
7:       $S \leftarrow S \cup \{(A, \Delta)\}$
8: print $S$
```

The algorithm is correct under Lemma 1 and Theorem 2.

Proposition 1. *Let S be the set from Algorithm 1. Suppose that $|S| = O(n^{\ln \ln n})$, where $n = \max\{[\ln N_1],\ [\ln N_2]\}$. Then the complexity of Algorithm 1 is $O(cn^2 \ln c \ln n \ln \ln n)$ binary operations.*

Consider a special case of Algorithm 1 with fixed $c = \frac{1}{2}$. Then the number of iterations of loop 4–7 equals 1. All the candidates B_1 and B_2 are of the form Q_k and P_k respectively. Hence all the pairs $\{(A, \Delta)\}$ are distinct and we could rewrite step 7 as $S \leftarrow S \sqcup \{(A, \Delta)\}$. In particular, it follows that $|S| = O(n)$ provided that $c = \frac{1}{2}$.

The method described above with $c = \frac{1}{2}$ was introduced in [3].

We shall show how to modify Algorithm 1 to reduce complexity $n = \max\{[\ln N_1], [\ln N_2]\}$ times under additional restriction.

While using Theorem 2 we should guess u and v for all the pairs of adjacent convergents. In the paper [2] A. Dujella introduce an idea, that allows to guess u and v only for a few pairs of convergents in a particular task. We generalize this idea with the following theorem.

Theorem 3. *Let α be a rational number, a and b be nonzero coprime integers ($b > 0$), such that*

$$0 < \left|\alpha - \frac{a}{b}\right| < d \le \frac{c}{b^2}, \tag{9}$$

where d and c are fixed positive real numbers. Let $\frac{P_0}{Q_0}, \dots, \frac{P_r}{Q_r} = \alpha$ be the convergents of α and $r \ge 3$.

Suppose that $0 \le k \le r-3$ is the largest integer satisfying $\left(\alpha - \frac{a}{b}\right)^k > 0$ and $\left|\alpha - \frac{P_k}{Q_k}\right| \ge d$. Then:

$$\frac{a}{b} = \frac{uP_{k+1} - vP_k}{uQ_{k+1} - vQ_k} \quad or \quad \frac{a}{b} = \frac{uP_{k+2} + vP_{k+1}}{uQ_{k+2} + vQ_{k+1}} \quad or \quad \frac{a}{b} = \frac{uP_{k+3} - vP_{k+2}}{uQ_{k+3} - vQ_{k+2}},$$

for some nonnegative integers u and v, $uv < 2c$.

It should be mentioned, that the interval from 0 to $r-3$ may not contain a number k from the condition of Theorem 3.

Lemma 2. *Let A, B_1, B_2 be natural numbers and Δ be an integer. Let $N_1 = AB_1$ and $N_2 = AB_2 + \Delta$. Suppose that inequalities*

$$A \ge D > \sqrt{\frac{N_1 |\Delta|}{c}}, \tag{10}$$

hold for some positive real numbers c and D. Then:

$$\left|\frac{N_2}{N_1} - \frac{B_2}{B_1}\right| < \frac{D^2 c}{N_1^2} \le \frac{c}{B_1^2}. \tag{11}$$

Lemma 1 is a special case of Lemma 2.

Suppose that we claim to find a divisor A above fixed known bound D in the PACDP with input N_1 and N_2. Then we could use the following algorithm.

Algorithm 2 (PACDP)

```
Require: $N_1$ and $N_2$ — natural numbers; $c, D$ — a method's real parameter
Ensure: $T \subset \mathbb{N} \times \mathbb{Z}$ — a set of pairs $(A, \Delta)$, such that $N_1 = AB_1$, $N_2 = AB_2 + \Delta$ and
    $A \geq D > \sqrt{\frac{N_1|\Delta|}{c}}$ and $A > 1$
 1: $T \leftarrow \varnothing$
 2: Compute all the convergents $\frac{P_0}{Q_0}, \ldots, \frac{P_r}{Q_r}$ of $\frac{N_2}{N_1}$ using Euclidean algorithm and formu-
    las (3), (4)
 3: $k_1 \leftarrow -1$, $k_2 \leftarrow -1$
 4: if $r \geq 3$ then
 5:   for all $k = 0, \ldots, r-3$ do
 6:     if $\left|\frac{N_2}{N_1} - \frac{P_{k+2}}{Q_{k+2}}\right| < \frac{D^2 c}{N_1^2} \leq \left|\frac{N_2}{N_1} - \frac{P_k}{Q_k}\right|$ then
 7:       if $k$ — even then
 8:         $k_1 \leftarrow k$
 9:       if $k$ — odd then
10:         $k_2 \leftarrow k$
11: for $k = k_1, k_2$ do
12:   if $k > -1$ then
13:     for $j = 0, 1, 2$  $v = 1, 2, \ldots, [2c]$, $u = 0, 1, \ldots, [\frac{2c}{v}]$ do
14:       if $uQ_{k+j+1} + (-1)^j vQ_{k+j} | N_1$ then
15:         $B_1 \leftarrow uQ_{k+j+1} + (-1)^j vQ_{k+j}$, $B_2 \leftarrow uP_{k+j+1} + (-1)^j vP_{k+j}$, $A \leftarrow \frac{N_1}{B_1}$,
            $\Delta \leftarrow N_2 - AB_2$
16:         $T \leftarrow T \cup \{(A, \Delta)\}$
17: print $T$
```

The algorithm is correct under Lemma 2 and Theorem 3. Note that there is no guarantee, that all the divisors $A > 1$ such that condition (10) holds are in the set T. The interval from 0 to $r-3$ may not contain an even or an odd number k such that $\left|\frac{N_2}{N_1} - \frac{P_{k+2}}{Q_{k+2}}\right| < \frac{D^2 c}{N_1^2} \leq \left|\frac{N_2}{N_1} - \frac{P_k}{Q_k}\right|$. In all our practical experiments this k was always founded in the specified interval.

Proposition 2. *Let T be the set from Algorithm 2. Suppose that $|T| = O(n^{\ln\ln n})$, where $n = \max\{[\log_2 N_1], [\log_2 N_2]\}$. If $n = O(c \ln c)$, then the complexity of Algorithm 2 is $O(cn \ln c \ln n \ln\ln n)$ binary operations.*

Using m parallel cores (without common RAM) we gain speed-up for Algorithms 1 and 2 by a factor of m. We should divide a set of all the pairs (u, v) to guess into m equinumerous subsets.

Let us write out the applicability condition for Algorithms 1 and 2 in terms of the binary representation lengths of the numbers A, B_1 and Δ.

Proposition 3. *Let $N = AB$ be a composite integer. Let l_A, l_B and l_Δ be the lengths of the binary representations of A, B and Δ respectively. Suppose that the condition*

$$l_A \geq l_B + l_\Delta + 2 - \log_2 c \tag{12}$$

holds for some real c. Then:

$$A \geq 2^{l_A - 1} > \sqrt{\frac{N\Delta}{c}}. \tag{13}$$

It follows from Proposition 3 that we can compute common divisor A for the integers $N_1 = AB_1$, $N_2 = AB_2 + \Delta$ using Algorithm 1, under condition (12). Suppose that l_A (the binary representation length of A) is known in addition. Then we could use Algorithm 2 with input parameters c and $D = 2^{l_A - 1}$.

4 Comparison of new algorithms with S. Sarkar's and S. Maitra's method

Let us compare our algorithms with the strongest algorithm for the PACDP. Algorithm of S. Sarkar and S. Maitra based on Coppersmith's method compute a common divisor provided that $\beta < 1 - 3\alpha + \alpha^2$, where $\alpha \approx \log_{N_1} B_1$, $\beta \approx \log_{N_1} \Delta - \log_{N_1} B_1$. Using inequality $N_1^\beta < N_1^{1-3\alpha+\alpha^2}$, we see that S. Sarkar's and S. Maitra's method may be applied for the PACDP under the condition:

$$l_A > l_{B_1} + l_\Delta - \frac{l_{B_1}^2}{l_{N_1}}, \tag{14}$$

where l_A, l_{B_1}, l_{N_1} and l_Δ are the lengths of the binary representations of A, B_1, N_1 and Δ respectively.

The comparison of inequalities (12) and (14) shows that the method based on the continued fraction techniques stronger if the following inequality holds:

$$\log_2 c - 3 > \frac{l_{B_1}^2}{l_{N_1}}. \tag{15}$$

Our algorithms require less accurate approximation under condition (15). For practical values of parameter c see section 5.

5 Experements

We implemented Algorithm 2 using MPIR [5] library for big numbers. We used the Microsoft Visual C++ compiler (64-bit). We ran our program on a single core of an Intel Core i7 processor (3.33 GHz).

In table 1 as before l_A denotes the length of the binary representation of A, l_Δ denotes the length of Δ, l_B denotes the lengths of B_1 and B_2, and c is method's parameter such that condition (12) holds.

Table 1: Program implementation of Algorithm 2

l_A	l_B	$\log_2 c$	l_Δ	Time, sec
2795	277	27	2543	1156
		30	2546	10250
3819	277	21	3561	17
		30	3570	12804

The values from the first and the third lines of table 1 are on the edge of applicability S. Sarkar's and S. Maitra's method. The values from the second and the forth do not satisfy condition of applicability S. Sarkar's and S. Maitra's method. In each case the computing of a common divisor with our implementation took the time showed in the last column of table 1.

References

[1] A. A. Buhshtab. Number Theory, Moscow: Prosvechenie, 1966.

[2] A. Dujella, *Continued fractions and RSA with small secret exponent*, Tatra Mt. Math. Publ. **29** (2004), 101–112.

[3] N. Howgrave-Graham. *Approximate integer common divisors.* In Proceedings of CaLC'01, Lecture Notes in Computer Science. **2146** (2001), 51 – 66.

[4] S. Sarkar, S. Maitra. *Approximate integer common divisor problem relates to implicit factorization.* IEEE Trans. Inform. Theory. **57** (2011), 4002 – 4013.

[5] The Multiple Precision Integers and Rationals Library. `http://www.mpir.org/`

A Proof of Lemma 1

Proof. Let us square the both sides of inequality (7), then divide both sides by A. We get $A > \frac{B_1|\Delta|}{c}$. It is clear now that:

$$\frac{1}{A} < \frac{c}{B_1\,|\Delta|}. \tag{16}$$

Let us write down ratio N_2 and N_1:

$$\frac{N_2}{N_1} = \frac{AB_2 + \Delta}{AB_1} = \frac{B_2}{B_1} + \frac{\Delta}{N_1}. \tag{17}$$

Hence the following equality holds:

$$\left|\frac{N_2}{N_1} - \frac{B_2}{B_1}\right| = \frac{|\Delta|}{AB_1}. \tag{18}$$

Notice, that combining (16) and (18) we obtain the state of lemma. □

B Proof of Proposition 1

Proof. The complexity of step 2 (Euclidean algorithm) is estimated in $O(n^2 \ln n \ln\ln n)$ binary operations.

The complexity of statement 5 and producing a pair (A, Δ) (step 6) are estimated in $O(n \ln n \ln\ln n)$ binary operations. We can insert new pair in the set S with $O(n \ln |S|) = O(n \ln n \ln\ln n)$ binary operations.

Total complexity of steps 5–7 estimated in $O(n \ln n \ln\ln n)$ binary operations.

Let us estimate the number of algorithm iterations for steps 5–7. The number of iterations of loop 3–7 equals the total number of convergents $r = O(n)$. The number of iterations of loop 4–7 is estimated in $O(c \ln c)$.

Therefore, the algorithm iterates per steps 5–7 $O(cn \ln c)$ times. Hence, the total complexity of the algorithm is $O(cn^2 \ln c \ln n \ln \ln n)$ binary operations. □

C Proof of Theorem 3

Proof. For any $i = 0, \ldots, r-1$ and $j = 0, 1$ a determinant of the system of linear equations

$$\begin{cases} a = xP_{i+1} + y(-1)^j P_i, \\ b = xQ_{i+1} + y(-1)^j Q_i; \end{cases} \tag{19}$$

equals $(-1)^{i+j}$ under (5). The pair $(aQ_i - bP_i)(-1)^i$, $(bP_{i+1} - aQ_{i+1})(-1)^{i+j}$ is a solution of system (19).

Further proof is divided into two parts depending on relative position of fractions $\frac{P_{k+2}}{Q_{k+2}}$ and $\frac{a}{b}$.

Suppose that $\left|\alpha - \frac{P_{k+2}}{Q_{k+2}}\right| < \left|\alpha - \frac{a}{b}\right|$. Consider the triple

$$\begin{aligned} u &= (aQ_k - bP_k)(-1)^k, \\ v &= (bP_{k+1} - aQ_{k+1})(-1)^k = (aQ_{k+1} - bP_{k+1})(-1)^{k+1}, \\ w &= (bP_{(k+1)+1} - aQ_{(k+1)+1})(-1)^{(k+1)+1}. \end{aligned}$$

Obviously the pairs (u, v) and (v, w) are solutions of system (19) for $i = k$, $j = 0$ and for $i = k+1$, $j = 1$ respectively. We claim that the coefficients u and v are positive.

Suppose that the difference $\alpha - \frac{a}{b}$ is negative; then k is even and the following inequalities hold: $\frac{P_{k+1}}{Q_{k+1}} < \frac{P_{k+2}}{Q_{k+2}} < \frac{a}{b} < \frac{P_k}{Q_k}$. Under $b > 0$ and $Q_k > 0$, right inequality in the chain above is equivalent to $aQ_k - bP_k < 0$. Hence u is positive and so are v and w.

Suppose the difference $\alpha - \frac{a}{b}$ is positive; then k is odd and the following inequalities hold: $\frac{P_k}{Q_k} < \frac{a}{b} < \frac{P_{k+2}}{Q_{k+2}} < \frac{P_{k+1}}{Q_{k+1}}$. In this case coefficients u, v and w are also positive.

The goal is to show that at least one of estimates $uv < 2c$ and $vw < 2c$ holds. Let t and s be some positive real numbers determined from the equalities $\left|\alpha - \frac{P_k}{Q_k}\right| = \frac{tc}{b^2}$ and $\left|\alpha - \frac{P_{k+1}}{Q_{k+1}}\right| = \frac{sc}{b^2}$. Notice, that the following estimates hold:

$$u = bQ_k \left|\frac{P_k}{Q_k} - \frac{a}{b}\right| < bQ_k \left|\alpha - \frac{P_k}{Q_k}\right| = \frac{tcQ_k}{b},$$

$$v = bQ_{k+1} \left|\frac{P_{k+1}}{Q_{k+1}} - \frac{a}{b}\right| = \left|\frac{P_{k+1}}{Q_{k+1}} - \alpha\right| + \left|\alpha - \frac{a}{b}\right| < \frac{(s+1)cQ_{k+1}}{b},$$

$$w = bQ_{k+2} \left|\frac{P_{k+2}}{Q_{k+2}} - \frac{a}{b}\right| < \left|\alpha - \frac{a}{b}\right| < \frac{cQ_{k+2}}{b}.$$

Let us get upper estimate for Q_kQ_{k+1} $Q_{k+1}Q_{k+2}$. Firstly note that

$$\frac{1}{Q_kQ_{k+1}} = \left|\frac{P_{k+1}}{Q_{k+1}} - \frac{P_k}{Q_k}\right| > \left|\alpha - \frac{P_k}{Q_k}\right| = \frac{tc}{b^2}.$$

On the analogy we get estimate

$$\frac{1}{Q_{k+1}Q_{k+2}} > \frac{sc}{b^2}.$$

Now we could see, that products of coefficients uv and vw are bounded above:

$$uv < \frac{Q_kQ_{k+1}t(t+1)c^2}{b^2} < \frac{t(s+1)c}{t} = (s+1)\,c,$$

$$vw < \frac{Q_{k+1}Q_{k+2}(t+1)c^2}{b^2} < \frac{(s+1)c}{s} = \left(1 + \frac{1}{s}\right)c.$$

Hence, if $s \leq 1$, then $uv < 2c$, and if $s > 1$, then $vw < 2c$.

Suppose now, that $\left|\alpha - \frac{P_{k+2}}{Q_{k+2}}\right| \geq \left|\alpha - \frac{a}{b}\right|$. Obviously the pair

$$u' = (aQ_{k+2} - bP_{k+2})(-1)^{k+2},$$
$$v' = (bP_{k+3} - aQ_{k+3})(-1)^{k+2},$$

is a solution of system (19) for $i = k+2$ $j = 0$. By analogy with reasoning above the coefficient u' is nonnegative and the coefficient v is positive. Let us obtain upper estimate for the coefficients u' and v':

$$u' = bQ_{k+2} \left|\frac{P_{k+2}}{Q_{k+2}} - \frac{a}{b}\right| < bQ_{k+2} \left|\frac{P_{k+2}}{Q_{k+2}} - \alpha\right| < bQ_{k+2}\frac{1}{Q_{k+2}Q_{k+3}} = \frac{b}{Q_{k+3}},$$

$$v' = bQ_{m+3}\left|\frac{a}{b} - \frac{P_{k+3}}{Q_{k+3}}\right| \le bQ_{m+3}\left|\frac{P_{k+2}}{Q_{k+2}} - \frac{P_{k+3}}{Q_{k+3}}\right| = \frac{b}{Q_{k+2}}.$$

Let us obtain upper estimate for the value $(Q_{k+2}Q_{k+3})^{-1}$:

$$\frac{1}{Q_{k+2}Q_{k+3}} = \left|\frac{P_{k+2}}{Q_{k+2}} - \frac{P_{k+3}}{Q_{k+3}}\right| = \left|\frac{P_{k+2}}{Q_{k+2}} - \alpha\right| + \left|\alpha - \frac{P_{k+3}}{Q_{k+3}}\right| <$$

$$< 2\left|\frac{P_{k+2}}{Q_{k+2}} - \alpha\right| < \frac{2c}{b^2}.$$

Now we can see, that the product of coefficients u' and v' is majorized

$$u'v' < \frac{b^2}{Q_{k+2}Q_{k+3}} < 2c.$$

□

D Proof of Lemma 2

Proof. Let us square the both sides of the right inequality from (10), then multiply both sides by c and divide by N_1^2. We obtain inequality

$$\frac{D^2c}{N_1^2} > \frac{|\Delta|}{N_1}. \tag{20}$$

Using the inequality $A \ge D$ we get

$$\frac{|\Delta|}{N_1} < \frac{D^2c}{N_1^2} \le \frac{A^2c}{N_1^2} = \frac{c}{B_1^2}. \tag{21}$$

Arguing as in the proof of Lemma 1, we see that

$$\left|\frac{N_2}{N_1} - \frac{B_2}{B_1}\right| = \frac{|\Delta|}{N_1}. \tag{22}$$

Notice, that combining (21) (22) we obtain the state of lemma. □

E Proof of proposition 2

Proof. The complexity of step 2 (Euclidean algorithm) is estimated in $O(n^2 \ln n \ln \ln n)$ binary operations.

The complexity of loop 4–10 (choosing the right convergent) is also estimated in $O(n^2 \ln n \ln \ln n)$ binary operations.

Arguing as in the proof of proposition 1 complexity of loop 11–16 (guessing u and v) is estimated in $O(c \ln c \ \ (n \ln |T| + n \ln n \ln \ln n)) = = O(cn \ln c \ln n \ln \ln n)$ binary operations.

Using $n = O(c \ln c)$, we obtain the total complexity of the algorithm is $O(cn \ln c \ln n \ln \ln n)$ binary operations. □

F Proof of proposition 3

Proof. The inequality $A \geq 2^{l_A - 1}$ is obvious. We shall prove that $2^{l_A-1} > > \sqrt{\frac{N\Delta}{c}}$.

Under the condition of proposition it follows that

$$2^{l_A} \geq \frac{4 2^{l_B} 2^{l_\Delta}}{c} > \frac{4B\Delta}{c}.$$

Hence

$$2^{l_A - 2} > \frac{N\Delta}{Ac}.$$

Using the inequality $2^{l_A} > A$ we conclude that

$$2^{2l_A - 2} > 2^{l_A - 2} A > \frac{N\Delta}{c}.$$

Extracting the square root we obtain inequality (12). □

On Operad-based Cryptography

Alina Gaynullina, Sergey Tronin

Abstract
We show how to use commutative operads in public-key cryptography.

Keywords: Commutative operad, public-key cryptography

1 Introduction

In the modern mathematical cryptography we see algorithms which use various algebraic structures. For example, groups are widely used [8]. Our goal is to show that there is a possibility of using commutative operads [12] in public-key cryptography. The definitions and the notations necessary for the further can also be found in [12].

2 Commutative operads

Definition 1. A Σ-*operad* is a family $R = \{\, R(n) \,|n = 1, 2, \ldots\}$ of sets such that at each $R(n)$ the permutation group Σ_n, $n = 1, 2, \ldots$ acts on the right and for arbitrary ordered sequences of nonnegative integers $m, n_1, \ldots, n_m$, there are defined some composition operations

$$R(m) \times R(n_1) \times \ldots \times R(n_m) \longrightarrow R(n_1 + \cdots + n_m),$$
$$(\omega, \omega_1, \ldots, \omega_m) \to \omega\omega_1 \ldots \omega_m.$$

The following properties hold:

1. (Associativity)

$$\omega(\omega_1\omega_{1,1} \ldots \omega_{1,k_1}) \ldots (\omega_m\omega_{m,1} \ldots \omega_{m,k_m}) =$$
$$= (\omega\omega_1 \ldots \omega_m)(\omega_{1,1} \ldots \omega_{1,k_1} \ldots \omega_{m,1} \ldots \omega_{m,k_m})$$

2. There is a distinguished element $\varepsilon \in R(1)$ (the identity of the operads), such that the identity $\omega(\varepsilon \ldots \varepsilon) = \omega$ and $\varepsilon\omega = \omega$ are valid for all $\omega \in R(m)$.

Also, the properties that bind the composition operation and the actions of the group Σ_n must hold. These properties can be found (in a slightly different form and in another context) in [12].

Consider two examples of operads.

Example 1.
Let X be an arbitrary set and let $Map(A, B)$ be the set of all mappings from A into B. Put $E_X = \{E_X(n) | n \geq 1\}$, $E_X(n) = Map(X^n, X)$ and define the composition as follows. Take $\omega_i : X^{n_i} \to X$, $1 \leq i \leq m$, and $\omega : X^m \to X$. Then $\omega\omega_1 \ldots \omega_m : X^{n_1+\cdots+n_m} \to X$. Let $\bar{x} \in X^{n_1+\cdots+n_m}$. By definition, $X^{n_1+\cdots+n_m} = X^{n_1} \times \cdots \times X^{n_m}$, then $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_m)$, where $\bar{x}_i \in X^{n_i}$. Then

$$\omega\omega_1 \ldots \omega_m(\bar{x}) = \omega(\omega_1(\bar{x}_1), \ldots, \omega_m(\bar{x}_m)).$$

The permutation group Σ_n acts as follows:

$$\omega\sigma(x_1, \ldots, x_m) = (x_{\sigma(1)}, \ldots, x_{\sigma(m)}).$$

Here $\omega \in E_X(m)$, $x_1, \ldots, x_m \in X$, and $\sigma \in \Sigma_m$.

Example 2.
Let G be a semigroup with the identity element 1.
Put $G = \{G(n) | n \geq 1\}$, where $G(n) = G^n$. An element $G(n)$ is a sequence (string) $\bar{x} = (x_1, \ldots, x_n)$ of elements $x_i \in G$. The action of an element $g \in G$ on a string $\bar{x}$ is defined as follows: $g\bar{x} = (gx_1, \ldots, gx_n)$. The composition in this operad is defined as follows:

$$G(m) \times G(n_1) \times \ldots \times G(n_m) \to G(n_1 + \cdots + n_m) \ ,$$
$$(\bar{x}, \bar{y}_1, \ldots, \bar{y}_m) \mapsto \bar{x}\, \bar{y}_1 \ldots \bar{y}_m,$$

where $\bar{x} = (x_1, \ldots, x_m) \in G(m)$, $\bar{y}_i = (y_{i,1}, \ldots, y_{i,n_i}) \in G(n_i)$ for $1 \leq i \leq m$, and $\bar{x}\, \bar{y}_1 \ldots \bar{y}_m = (x_1\bar{y}_1, \ldots, x_m\bar{y}_m)$.

The permutation group Σ_n acts on the set $G(n)$ as follows:

$$(x_1, \ldots, x_n)\sigma = (x_{\sigma(1)}, \ldots, x_{\sigma(n)}).$$

Definition 2. Suppose that R is a Σ-operad. By an *algebra over R* we mean a set A that is endowed with some mappings of the form:

$$R(n) \times A^n \longrightarrow A, \qquad (r, \bar{a}) \mapsto r\bar{a}.$$

Here $r \in R(n), \bar{a} = a_1 \ldots a_n$, $a_i \in A$. Moreover, the following conditions must be satisfied:

1. $(rr_1 \ldots r_m)\bar{a}_1 \ldots \bar{a}_m = r(r_1\bar{a}_1) \ldots (r_m\bar{a}_m)$.
2. $\varepsilon a = a$ for all $a \in A$. Here $\varepsilon \in R(1)$ is the identity of R.
3. The identity $(rf)a_1 \ldots a_m = ra_{f(1)} \ldots a_{f(m)}$ holds for every $r \in R(m)$, $a_1, \ldots, a_m \in A$, $f \in \Sigma_m$.

Definition 3. Assume that Z is a $\Sigma-$operad. We call the operad Z *commutative* if the identity

$$\lambda \overbrace{\omega \ldots \omega}^{n} = (\omega \overbrace{\lambda \ldots \lambda}^{m})\sigma_{n,m},$$

is valid for all $\lambda \in Z(n)$ and $\omega \in Z(m)$. Here $\sigma_{n,m} \in \Sigma_{nm}$ and

$$\sigma_{n,m}(i + (j-1)n) = (j + (i-1)m),$$

for $1 \leq i \leq n$, $1 \leq j \leq m$.

We denote by $\sum\limits_{i=1}^{n}{}^{(\lambda)} a_i$ the result of the composition of $\lambda \in Z(n)$ with $a_1, \ldots, a_n \in A$. Let Z be a commutative operad, then for all $\lambda \in Z(n)$ and $\omega \in Z(m)$ the identity:

$$\sum_{i=1}^{n}{}^{(\lambda)} \sum_{j=1}^{m}{}^{(\omega)} a_{i,j} = \sum_{j=1}^{m}{}^{(\omega)} \sum_{i=1}^{n}{}^{(\lambda)} a_{i,j}$$

is valid in every Z-algebra.

In these notations, a homomorphism between the algebras over commutative operads is a map h, such that $h(\sum\limits_{i=1}^{n}{}^{(\lambda)}\, a_i) = \sum\limits_{i=1}^{n}{}^{(\lambda)}\, h(a_i)$ for all $\lambda \in Z(n)$ and $a_1, \ldots, a_n \in A$.

The value of a commutative operad is showed by the following theorem proved in [13].

Theorem 1. *The center of a multicategory is a commutative operad. The center of a commutative operad R coincides with R.*

Consider several examples of commutative operads.

Example 3.
Consider an operad Z for which $Z(0) = \emptyset$, $Z(1)$ is a singleton, and if $n \geq 2$ then $Z(n) = \emptyset$. We may assume that Z is a $FSet$-operad. The definition of a commutative operad for Z is fulfilled trivially. The category of algebras over this operad is actually the category of all sets.

Example 4.
Generalizing Example 3 to some extent consider an operad with the unique nonempty component $Z(1)$, which is a commutative monoid. This operad is also commutative, and the category of algebras over it is rationally equivalent to the category of left $Z(1)$-sets, i.e. the sets on which the left action of the monoid $Z(1)$ is defined.

Example 5.
Let G be a commutative monoid with the operation written multiplicatively. Consider an operad from example 2. It could be easily verified that the so-constructed operad is commutative. If G is a commutative associative ring with unity then the variety of algebras over the operad is rationally equivalent to the category of left G-modules. In some cases, it will be convenient to denote the above operad like the monoid itself, i.e. by G.

Example 6.
Clearly, a suboperad of a commutative operad is also commutative. In many cases, some important examples are given by operads defined over a smaller verbal category than that over which the ambient commutative

operad is defined. However, there are interesting cases when no restriction of the verbal category appears. For example, the operad of simplices Δ, studied in [11], is a suboperad in the $FSet$-operad $\mathbf{R}$, where $\mathbf{R}$ is the field of reals, and the corresponding operad is constructed as in Example 3. We proved in [11] that Δ admits a $FSet$-operad structure.

3 Cryptographic Protocols

Let Z be a commutative operad, A be an algebra over Z. These data are public (not secret).

Protocol 1. Creation of a common secret key
Alice's secret is a $\omega \in Z(n)$. Bob's secret is a $\lambda \in Z(m)$. Public elements are $a_{i,j} \in A$, $1 \le i \le n$, $1 \le j \le m$.

1. Alice computes $\alpha_j = \sum\limits_{i=1}^{n}{}^{(\omega)} a_{i,j}$, $1 \le j \le m$.
2. Bob computes $\beta_i = \sum\limits_{j=1}^{m}{}^{(\lambda)} a_{i,j}$, $1 \le i \le n$.
3. Alice sends the elements α_j to Bob.
4. Bob sends the elements β_i to Alice.
5. Finally, Alice computes $\sum\limits_{i=1}^{n}{}^{(\omega)} \beta_i$, and Bob computes $\sum\limits_{j=1}^{m}{}^{(\lambda)} \alpha_j$.

By definition of a commutative operad, $\sum\limits_{i=1}^{n}{}^{(\omega)} \sum\limits_{j=1}^{m}{}^{(\lambda)} a_{i,j} = \sum\limits_{j=1}^{m}{}^{(\lambda)} \sum\limits_{i=1}^{n}{}^{(\omega)} a_{i,j}$. Thus Alice and Bob share a common secret key.

The security of the protocol is based on the complexity of the task of finding $\xi \in Z(k)$ using known $b_1, \ldots, b_k \in A$ and $\sum\limits_{i=1}^{k}{}^{(\xi)} b_i \in A$.

Protocol 2. Key exchange
Public data is a commutative operad Z, a number n, an element $\omega \in Z(n)$.

1. Alice picks a random element $\alpha \in Z(m)$ and sends to Bob the element $\alpha \overbrace{\omega \ldots \omega}^{m} = \alpha\omega^m$.

2. Bob picks a random element $\beta \in Z(k)$ and sends to Alice the element $\beta\omega^k$.

3. Alice computes $\alpha(\beta\omega^k)^m = (\alpha\beta^m)\omega^{mk}$.

4. Bob computes

$$\beta(\alpha\omega^m)^k = (\beta\alpha^k)\omega^{mk} = ((\alpha\beta^m)\sigma_{m,k})(\omega^{mk}) = (\alpha\beta^m)(\omega^{mk})\sigma'.$$

The security of protocol is based on the complexity of the task of finding α using known $\alpha\omega^m$ and ω.

Protocol 3. Encryption
A bit string m of length ℓ is being encrypted. The public data is a $g \in Z(n)$ and a hash function h that maps the elements of the operad Z to bit strings of length ℓ.

The secret key is $x \in Z(m)$. The public key is $y = xg \dots g \in Z(mn)$.

The encryption begins with a random selection of the session key $k \in Z(d)$. The first part of the ciphertext is $c_1 = kg \dots g \in Z(dn)$. The second part of the ciphertext is the bit string $c_2 = m \oplus h(ky \dots y)$.

The decryption: $m = c_2 \oplus h((xc_1 \dots c_1)\sigma)$. Here $\sigma = {\sigma_{d,m}}^* \alpha$, $\alpha = \overbrace{(n, \dots, n)}^{dm}$ (see [13], p. 52). The security of this protocol is based on the complexity of the task of finding an element of operad x according to the known g and $xg \dots g$.

Protocol 4. Authentication
Alice's secret is an element $x \in Z(n)$.

1. Bob picks an element $g \in Z(m)$ and sends it to Alice.

2. Alice computes $y = xg^n$ and sends y to Bob.

3. Bob picks an element $k \in Z(\ell)$ and sends it to Alice.

4. Finally, Alice computes the $z = xk^n g^{n\ell}$, $z \in R(n\ell m)$ and sends it to Bob.

Verification. Bob knows g, k, y and can compute:

$$k\underbrace{y\dots y}_{\ell} = k\underbrace{(xg^n)\dots(xg^n)}_{\ell} = (k\underbrace{x\dots x}_{\ell})g^{n\ell} =$$
$$= ((xk\dots k)\sigma_{\ell,n})g^{n\ell} = ((xk\dots k)g^{n\ell})\sigma'.$$

Then Bob compares this element with the received z.

The security of protocol is based on the complexity of the task of finding an element of operad $x \in Z(n)$ using known $g \in Z(m)$ and $y = xg^n$.

4 Implementation and cryptographic security

In this section, we describe and explore the cryptographic security of Protocol 1.

Let K be an associative commutative ring or semiring, Z be a commutative operad of Examples 2 and 4, where $G = K$, $Z(n) = K^n$.

Let k be a fixed positive integer. Consider an arbitrary suboperad R of operad Z, and determine the structure of R-algebra on $A = K^m$. We determine mappings:

$$R(n) \times A^n \longrightarrow A$$
$$\xi a_1 \dots a_n = \sum_{i=1}^{n}{}^{(\xi)} a_i = x_1^k a_1 + \dots + x_n^k a_n,$$

where $\xi = (x_1, \dots, x_n) \in R(n)$, $a_i = (a_{1,i}, \dots, a_{m,i})$, $1 \le i \le n$.

Lemma 1. *A is a R-algebra.*
Proof. A check of definition.

Lemma 2. *Let $b = (b_1, \dots, b_m) \in A$. The equality $\xi a_1 \dots a_n = b$ is equivalent to the system of equations in the ring or semiring K:*

$$\left\{\begin{array}{ccccccc} a_{1,1}x_1^k & + & \cdots & + & a_{1,n}x_n^k & = & b_1 \\ \cdots & & \cdots & & \cdots & & \cdots \\ a_{m,1}x_1^k & + & \cdots & + & a_{m,n}x_n^k & = & b_m \end{array}\right. \qquad (1)$$

Proof. By definition.

Theorem 2. *The cryptographic security of Protocol 1 depends on the complexity of solving large systems of equations of the type* (1) *over the* *ring or semiring K. Moreover, only solutions $(x_1, \dots, x_n) \in R(n)$ should be considered.*

Proof. The proof follows from Lemmas 1 and 2.

Next, consider some examples of rings and semirings.

Example 7.
Let K be a tropical semiring. Recall [7], [10] that a tropical semiring is a semiring which has support $\mathbf{N} \cup \{+\infty\}$ and operations $a \oplus b = \min\{a, b\}$ and $a \otimes b = a + b$.

The tropical semiring are already used in cryptography [6]. There are several works where algorithms for solving systems of linear equations over such K [2] were introduced and investigated [3], [5]. A.P. Davydov recently proved that the Grigoriev's algorithm is a non-polynomial time algorithm $(t = \Omega(n^{\frac{m}{6}} \log(\mathrm{poly}(n^{\frac{m}{6}}))))$ [2]. Thus, the case of the tropical semirings looks promising.

Example 8.
Let $K = \mathbf{Z}$ and $A = \mathbf{Z}^m$. Then (1) is a system of the Diophantine equations. The complexity of the solutions for the case of $k = 1$ (linear Diophantine equations) studied in [9]. It was shown that there exists the polynomial-time algorithm for solution of (1).

Lemma 3. ([9], p. 54) *A linear diophantine equation with rational coefficients can be solved in polynomial time.*

Example 9.
Let K be a finite field and $k = 1$. Yu.V. Nesterenko considers [4] only the case of the square sparse matrixes. G.V. Bard considers [1] the case of $GF(2)$. There is an polynomial-time algorithm in both cases ([1], Part 3 and [4], p. 96-98). However, this does not exhaust the entire range of cases that are interesting to us .

Obviously, these three examples do not exhaust the plurality of rings and semirings that should be explored. Our research will be continued in subsequent publications.

Some results of this paper were announced in [14].

5 Acknoledgements

This work was partially funded by the subsidy allocated to Kazan Federal University for the state assignment in the sphere of scientific activities (the registration number is 114 090 970 010).

References

[1] Bard G. Algebraic Cryptanalysis. — New York: Springer, 2009. — 356 p.

[2] Davydov A.P. Upper and lower bounds for Grigorievs algorithm for solving integral tropical linear systems // Journal of Mathematical Sciences. — 2013. — V. 192. No 3. — P. 295-302.

[3] Davydov A.P. New Algorithms for Solving Tropical Linear Systems — 17 pages. — http://arxiv.org/pdf/1309.5206v1.pdf, 20.09.2013.

[4] Nesterenko Yu.V. et al. Hard Computational Problems in Number Theory. — Moscow: Moscow University Press, 2012. — 312 p.

[5] Grigoriev D. Complexity of Solving Tropical Linear Systems // Computational Complexity. — 2013. — V. 22. No 1. — P. 71-88.

[6] Grigoriev D., Shpilrain V. Tropical Cryptography // Communications in Algebra. — 2014. — V. 42. No 6. — P. 2624-2632.

[7] Krob D. The Equality Problem for Rational Series With Multiplicities in the Tropical Semiring is Undecidable // International Journal of Algebra and Computation. — 1994. — V. 4. No 3. — P. 405-425.

[8] Myasnikov A. et al. Group-based Cryptography. (Advances courses in Math., CRM, Barselona). — Basel, Berlin, New York: Birkhauser Verlag, 2008. — 183 p.

[9] Schrijver A. Theory of Linear and Integer Programming. — Chichester: Wiley, 1998. — 484 p.

[10] Speyer D., Sturmfels B. Tropical Mathematics // Mathematics Magazine. — 2009. — V. 82. No 3. — P. 163-173.

[11] Tronin S.N. Operads in the category of convexors. I // Russian Mathematics (Iz. VUZ). — 2002. — V. 46. No 3. — P. 38-46.

[12] Tronin S.N. Operads and varieties of algebras defined by polylinear identities // Siberian Mathematical Journal. — 2006. — V. 47. No 3. — P. 555-573.

[13] Tronin S.N. Natural multitransformations of multifunctors // Russian Mathematics (Iz. VUZ). — 2011. — V. 55. No 11. — P. 49-60.

[14] Tronin S.N., Gaynullina A.R. Some applications of the Operad Theory in Public-Key Cryptography // Proc. Internat. Conf. “Algebra and Mathematical Logic: Theory and Applications” (Kazan, Russia, June 2–6, 2014). — Kazan: Kazan University Press, 2014. — P. 146-147.

Reconstruction of a Skew Non-reducible MP LRS over Galois Ring of Characteristic 2^d by its Highest Digit

Sergey Zaitsev

Abstract

Let $R = GR(q^d, 2^d)$ be a Galois ring, $S = GR(q^{nd}, 2^d)$ be a Gaolis extension of R and $\mathcal{L} = \mathcal{L}({}_RS)$ be the ring of all linear transformations of the module ${}_RS$.

This paper studies linear recurring sequences v over the module ${}_{\mathcal{L}}S$ of the maximal period (skew MP LRS), i.e. if $T(v) = (q^{mn} - 1)p^{d-1}$. The sequence of highest digits (in some digital set) of members MP LRS is considered in cryptography as pseudo-random sequence, wherein it is important, that appropriate generator should have no equivalent keys (initial states). In case of $n = 1$ (classic MP LRS) the absence of equivalent keys is proved for some class $\mathcal{K}$ digital sets. In this paper we prove this absence for arbitrary n and digital sets from $\mathcal{K}$ with some requirement for law of recursion. In particular, this requirement is fulfilled for the laws, which generate *non-reducible* sequences, i.e. sequences v, for which non-trivial relation $\psi(v_0) = x^l v_0, \psi \in \mathcal{L}$ is impossible, where v_0 is the lowest digital sequence of v. Also we present constructive algorithm for restoring the key by this sequence.

Keywords: *skew linear recurrence, skew polynomial of maximal period, Teichmuller's digital set.*

1 Introduction

Linear recurring sequences (LRS) over the finite fields and primary residue rings and its images are widely considered in researches of synthesis and analysis of pseudo-random generators. Large part of these works studies properties of filtering generators over fields and digital sequences over primary residue rings [5, 7, 8, 11, 12, 13]. Some important properties of LRS over fields and primary residue rings can be formulated over Galois ring, as generalization (e.g. [6, 9, 14, 15, 16, 17, 18, 19, 20, 21]). Particularly, authors of [1, 10, 13, 14, 22, 23, 24, 25, 26, 27, 28, 29, 30] considered pseudo-random sequences, which were produced by taking image of the highest digit of LRS over Galois ring (in some digital set).

In FSE of 1994, Preneel [4] set forth the following problem: *how to design fast and secure FSRs with the help of the word operations of modern pro-*

cessors and the techniques of parallelism. In stream ciphers such as SOBER [31, 32], SNOW [33, 34], and Turing [35], word-oriented primitive LRS over finite field were used, which were carefully chosen so that the Hamming weights of the generating primitive polynomials of the component sequences are large and have fast software implementation.

In respect of above, it is interesting to consider the highest digital sequence (as pseudo-random sequence), which is produced by word-oriented feedback shift register (FSR) over Galois ring. Here it is important, that an appropriate generator should have no equivalent keys. In this paper the absence of equivalent keys is proved for a large class of such FSRs and corresponding reconstruction algorithm is provided.

Let $R = GR(q^d, 2^d)$ be a Galois ring, $S = GR(q^{nd}, 2^d)$ be a Galois extension of the ring R of dimension n, $\bar{S} = S/2S = GF(q^n)$ be the residue field of the ring S, $\mathcal{L} = \mathcal{L}({}_RS)$ be the ring of all linear transformations of the module ${}_RS$ and $S^{\langle 1\rangle}$ be the set of all sequences over the ring S. Let us define the product of a sequence $v \in S^{\langle 1\rangle}$ by a polynomial $\Psi(x) = \psi_m x^m + ... + \psi_1 x + \psi_0 \in \mathcal{L}[x]$ by the equalities:

$$\Psi(x)v = w : w(i) = \sum_{j\geq 0} \psi_j(v(i+j)),\ i \geq 0. \tag{1}$$

A sequence $v \in S^{\langle 1\rangle}$ is called an *(R-)skew linear recurring sequence (LRS)* over the ring S of order m, if there exists a monic polynomial $\Psi(x) \in \mathcal{L}[x]$ of degree m, such that

$$\Psi(x)v = 0, \tag{2}$$

i.e. if v is LRS of order m over the module ${}_{\mathcal{L}}S$.

In this case the polynomial $\Psi(x)$ is called a *characteristic polynomial* of LRS v. The set of all skew LRS with characteristic polynomial $\Psi(x)$ is denoted by $L_S(\Psi)$. In a particular case, when $\Psi(x) \in S[x]$ the sequence v is called a *classic LRS*. (Here and further we'll identify S and the set of homotheties generating by the elements of S in $\mathcal{L}$.) If $v = \psi(u)$ for some $\psi \in \mathcal{L}$ and classic MP LRSu, then v is called a *linearized* one. A characteristic polynomial of the least degree is called a *minimal polynomial* and its degree is called the *rank (skew complexity)* of v. It's known [2], that the period $T(v)$ of the a skew LRS v of order m over S is not greater than $\tau 2^{d-1} = (q^{mn} - 1)2^{d-1}$. If $T(v) = \tau 2^{d-1}$, the sequence v is called a *skew LRS of maximal period (MP LRS)*.

Let us denote by $\bar{a}$ the image of an element $a \in S$ under the natural epimorphism $S \to \bar{S}$. A subset $K \subset S$ is called a *digital* one, if $0 \in K$ and for any $a \in S$ there exists a unique element $\varkappa(a) \in K$ such that $\bar{a} = \overline{\varkappa(a)}$. In this case [6] each element $a \in S$ has unique representation in the form

$$a = \varkappa_0(a) + 2\varkappa_1(a) + \ldots + 2^{d-1}\varkappa_{d-1}(a), \quad \varkappa_t(a) \in K, \ t \in \overline{0, d-1}. \quad (3)$$

We'll call the function $\varkappa_t : S \to K$ a *t-th digital* one (in the digital set K), and the representation (3) a *representation of an element a in the digital set K (K-representation of a).* It's evident, under above notations the set K is satisfied the condition $K = \varkappa_0(S)$.

An important example of digital set is 2-*adic* digital set (Teichmuller)

$$\Gamma(S) = \{c \in S : c^{q^n} = c\}.$$

We'll write the $\Gamma(S)$-representation of $a \in S$ in the form

$$a = \gamma_0(a) + 2\gamma_1(a) + \cdots + 2^{d-1}\gamma_{d-1}(a), \ \gamma_t(a) \in \Gamma(S), \ t \in \overline{0, d-1}, \quad (4)$$

and call (4) a 2-*adic digital representation of* a. We'll call the digital functions $\gamma_t : S \to \Gamma(S), t \in \overline{0, d-1}$, in the same manner, respectively.

When considering a K-representation of members of a skew LRS v

$$v(i) = w_0(i) + 2w_1(i) + \ldots + 2^{d-1}w_{n-1}(i),$$
$$w_t(i) = \varkappa_t(v(i)) \in K, \quad t \in \overline{0, d-1}, \quad (5)$$

we map the *digital sequences* $w_0, \ldots, w_{d-1}$ over K to the sequence v. We'll write the $\Gamma(S)$-representation of v in the following form

$$v(i) = v_0(i) + 2v_1(i) + \ldots + 2^{d-1}v_{n-1}(i),$$
$$v_t(i) = \gamma_t(v(i)) \in \Gamma(S), \quad t \in \overline{0, d-1}, \quad (6)$$

For any digital set K of the ring S there exists a unique polynomial $\varphi_K(x) \in S(x)$ with the properties

$$\varphi_K(\Gamma(S)) = K, \quad \deg\varphi_K(x) \le q^n - 1, \quad \varphi_K(x) \equiv x (\mathrm{mod}\ pS).$$

This polynomial is called an *interpolation polynomial* of the digital set K [6].

We consider binary decomposition of a number $t \in \mathbb{N}$

$$t = t_0 + 2t_1 + \ldots + 2^k t_k, \quad t_0, t_1, \ldots, t_k \in \overline{0, 1}.$$

We call the number

$$\mathrm{ind} lx^t = t_0 + \ldots + t_k,$$

a *nonlinearity index* of the nonzero monomial lx^t, herewith $\mathrm{ind} 0 \overset{\Delta}{=} -\infty$. *The nonlinearity index* $\mathrm{ind}\varphi(x)$ of an arbitrary polynomial $\varphi(x)$ is the maximum of indices of nonlinearity of its monomials, by definition. We denote by $\mathcal{K}$ the class of all digital sets for which $\mathrm{ind}\varphi_K(x) \leq 1$.

Let us define the following operations on the set K

$$a \oplus b = \varkappa_0(a + b), a \otimes b = \varkappa_0(a \cdot b), a, b \in K.$$

It's proved, that $(K, \oplus, \otimes)$ is a field, and $K \cong \bar{S}$. Also the set $P = \varkappa_0(R)$ is a subfield of K, and $P \cong \bar{R}$.

We'll call the linear transformation $\zeta \in \mathcal{L}({}_PK)$ a *(generalized) multiplier* of reversible sequence u over K if there exists $l \in \mathbb{N}_0$ such that

$$x^l u = \zeta(u).$$

The minimal such $l \in \mathbb{N}$ is called a *reduced period* of the sequence u. We denote by $\mathfrak{M}^*(u)$ the set of all multipliers of the sequence u and $\mathfrak{M}(u) \overset{\Delta}{=} \mathfrak{M}^*(u) \cup \{0\}$. In [3] it is proved, that if u is a P-skew MP LRS over the field K, then the set $\mathfrak{M}(u)$ is a field with operations of the ring $\mathcal{L}({}_PK)$, moreover

$$P < \hat{\mathfrak{M}}(u) < K,$$

where $\hat{\mathfrak{M}}(u) \cong \mathfrak{M}(u)$. The fact that $\hat{\mathfrak{M}}(u) = K$ and fact that the recurrence u is *linearized* one are equivalent [3]. In the other interesting case, when $\hat{\mathfrak{M}}(u) = P$, we'll call the recurrence u a *(maximal) non-reducible* one. Obviously, reduced period of such sequence is maximal. The recurrence v is called a *(maximal) non-reducible* LRS if $v_0 = \varkappa_0(v)$ is such one.

In [2] it's proved, that if v is MP LRS of order m over S, then there exists MP-polynomial $F(x) \in R[x]$ of degree mn, for which $F(x)v = 0$. We fix some root θ of the polynomial $F(x)$ in the Galois extension $Q = R[\theta]$.

We denote by $\overset{F}{\equiv}$ the relation of congruences of polynomial $F(x)$ on $R[x]$.

Lemma 1.1. *[6, 5] If $F(x) \in R[x]$ is MP-polynomial of degree mn, then for some polynomial $\Phi(x) \in R[x]$*

$$x^\tau \overset{F}{\equiv} e + p\Phi(x), \quad \deg \Phi(x) < mn, \tag{7}$$

where

$$\bar{\Phi}(x) \neq 0, \quad d = 2,$$
$$\bar{\Phi}(x) \notin \{0, \bar{e}\}, \quad d > 2.$$

The main result of this paper is

Theorem 1.2. *For $n \geq 1$ and digital set $K \in \mathcal{K}$ a skew MP LRS v over the ring S is uniquely reconstructible by its highest digit if*

$$\bar{\Phi}(\bar{\theta}) \notin \hat{\mathfrak{M}}(v_0). \tag{8}$$

In particular case if v is non-reducible $\mathbb{Z}_2$-skew MP LRS the relation (8) *is true.*

In case of $S = R = \mathbb{Z}_{p^d}$, $p \geq 2$ and the sequence v is classic MP LRS, this theorem and some generalizations are proved without the requirement (8) [8-18]. In [1] this theorem without the requirement (8) is proved for classic MP LRS ($n = 1$) over the arbitrary Galois ring of characteristic p^d, $p \geq 2$.

2 Reconstruction

Lemma 2.1 ([6]). *For MP-polynomial $F(x) \in R[x]$ of degree mn, $t \in \overline{0, d-1}$ and numbers $\tau_t = \tau p^t$ there exist polynomials $\Phi_1(x), ..., \Phi_d(x) \in R[x]$, for which the following relations are true*

$$x^{\tau_t} \overset{F}{\equiv} e + p^{t+1}\Phi_{t+1}(x), \text{ deg } \Phi_{t+1}(x) < mn, \ \bar{\Phi}_{t+1}(x) \neq 0. \tag{9}$$

wherein, if $d > 2$, then

$$\bar{\Phi}_2(x) \overset{F}{\equiv} \bar{\Phi}_1^2(x) + \bar{\Phi}_1(x), \quad \Phi_{t+1}(x) \overset{2^t}{\equiv} \Phi_t(x), \quad t \geq 2 \tag{10}$$

Let us introduce the notations

$$v^{(t)} = \Phi_t(x)v, \ w_s^{(t)} = \varkappa_s(v^{(t)}), \ t \in \overline{0, d-1}. \tag{11}$$

We'll call the sequence $v^{(t)}$ from (11) a *t-th derivative sequence* of the sequence v. Note that the polynomial Φ_t from (13) is defined only modulo p^{d-t} and the sequence $v^{(t)}$ defined only modulo p^{d-t} too. We'll consider, that the polynomial $\Phi_t(x) \in R[x]$ is selected and fixed.

According to (13), the polynomial $\Phi_t(x)$ and $F(x)$ are relatively prime, so $v^{(t)} \in L_S^*(F)$, and (10) implies

$$w_0^{(t)} = w_0^{(2)}, \quad t \geq 2,$$
$$w_1^{(t)} = w_1^{(3)}, t \geq 3; \tag{12}$$

Lemma 2.2. *For each $s, t \in \overline{0, d-1}$ the reconstruction of the set $w_0, ..., w_s$ is equivalent to the reconstruction of the set $v_0, ..., v_s$ and it's equivalent to the reconstruction of the set $w_0^{(t)}, ..., w_s^{(t)}$.*

2.1 Reconstruction of the sequence w_0

We define the product of an element $a \in K$ by an element $\bar{r} \in \bar{S}$ by the equality

$$\bar{r}a = \varkappa_0(ra).$$

Now K is a $\bar{S}$-algebra and we can multiply any sequence over K by polynomial from $\bar{S}[x]$ and one can consider that any periodic sequence over K is an LRS over the field $\bar{S}$.

The *section* of the digital sequence w_s from (5) is the result of multiplication of this sequence by some polynomial from $\bar{S}[x]$, which is more simple sequence than w_s [1].

Lemma 2.3. *For $s \geq 1$ the following statement takes place*

$$(x^{\tau_{s-1}} - \bar{e})w_s = w_0^{(s)}. \tag{13}$$

According to this result we can reconstruct the sequence $w_0^{(d-1)}$ and according to the lemma 2.2 this is equivalent to the reconstruction of the sequence w_0.

Now, for $d = 2$, the theorem 1.2 is proved and further we consider only $d \geq 3$.

2.2 Carry function and associated sections

The *carry function* in the digital set K [1] is the function $\Delta_K : K \times K \to K$ which is defined by the equality

$$\Delta_K(x, y) = \varkappa_1(x + y)$$

Lemma 2.4. *The following statements take place*

$$(x^{\tau_{s-2}} - \bar{e})w_s = w_1^{(s-1)} \oplus \Delta(w_{s-1}, w_0^{(s-1)}), \quad s \geq 2 \tag{14}$$

Lemma 2.5 ([1])**.** *If $K \in \mathcal{K}$, then*

$$\Delta_K(x, y) \overset{2}{\equiv} (xy)^h. \tag{15}$$

2.3 Reconstruction of the sequence w_1 in $|\mathfrak{M}(v_0)|$ variants

We remind that $d \geq 3$ and so we can construct the set

$$I_0^{(d-2)} = \{i \in \mathbb{N}_0 : w_0^{(d-2)}(i) = 0\}$$

by the available sequence $w_0^{(d-2)}$.

Let $\tilde{R} = R/4R = \mathbb{Z}_4$ and we denote by $\tilde{S}, \tilde{F}, \tilde{v}, \ldots$ the images $S, F, v, \ldots$ under the natural epimorphism $R \to \tilde{R}$.

Lemma 2.6. *There exist exactly $2^{[P:\mathfrak{M}(v_0)]} = |\mathfrak{M}(v_0)|$ different skew LRS $\tilde{y} \in L_{\tilde{S}}(\tilde{\Psi})$ such that*

$$\tilde{y}_0^{(d-2)} = \tilde{v}_0^{(d-2)},\ \tilde{y}_1^{(d-2)}(i) = \tilde{v}_1^{(d-2)}(i),\ i \in I_0^{(d-2)}.$$

The set L of all such LRS is the set of all sequences

$$\tilde{v}^{\langle\zeta\rangle} = (\varepsilon + 2\zeta)(\tilde{v}), \quad \zeta \in \mathfrak{M}(v_0). \tag{16}$$

Now, if we have the sequence w_{d-1} then we can find $|\mathfrak{M}(v_0)|$ candidates for the sequence $\tilde{v} = v(\text{mod } 4)$. These candidates are described by the formula (16), wherein the genuine sequence correspond $\zeta = 0 \in \mathfrak{M}(v_0)$.

It should be noted that each candidate $\tilde{v}^{\langle\zeta\rangle}$ for $\tilde{v}$ is point to unique candidate $\tilde{w}_1^{\langle\zeta\rangle}$ for w_1 and

$$w_1^{\langle\zeta\rangle} = w_1 \oplus \varphi_K(\zeta(w_0)) = w_1 \oplus \hat{\zeta}(w_0), \quad \hat{\zeta} = \varphi_K(\zeta).$$

Analogously, the following relations take place

$$w_1^{\langle\zeta\rangle(k)} = w_1^{(k)} \oplus \hat{\zeta}(w_0^{(k)}), \quad k = 1, 2, \ldots \tag{17}$$

2.4 Restoring 2-adic digital sequences $v_2, \dots, v_{d-3}$ by w_0, w_1, w_{d-1}

If $d = 3$, then $d - 1 = 2$ and the problem is solved, so here we consider $d \geq 4$. One can assume that the sequences $w_0, w_1, w_{d-1}, w_0^{(t)}, w_1^{(t)}, t \in \overline{0, d-1}$ are available. From (14) and (12) for $s \geq 3$ we can say that

$$\Delta_K(w_{s-1}, w_0^{(2)}) = (x^{\tau_{s-2}} - \bar{e})w_s \ominus w_1^{(s-1)},$$

and with the equality (15), squaring this relation, we obtain for $s \geq 3$

$$w_{s-1}w_0^{(2)} = \left((x^{\tau_{s-2}} - \bar{e})w_s \ominus w_1^{(s-1)}\right)^2. \tag{18}$$

Now, there exists the set of numbers $J = \{j_0 < j_1 < \dots < j_{m-1}\} \subset \{0, 1, \dots, \tau_{d-1}\}$ such that coordinates of the vector $w_0^{(2)}[J] = (w_0^{(2)}(j_0), \dots, w_0^{(2)}(j_{m-1}))$ are reversible in the field K and the sequence v can be uniquely restored from $v(j_0), \dots, v(j_{m-1})$.

To restore the vector $v[J]$ we use the decomposition (5)

$$v[J] = w_0[J] + 2w_1[J] + \dots + 2^{d-1}w_{d-1}[J].$$

Note, that we know sequences w_0, w_1, w_{d-1}, so we can construct the vectors $w_{d-2}[J], \dots, w_2[J]$ successively.

Let $I_*^{(2)} = \{i \in \mathbb{N}_0 : w_0^{(2)} \neq 0\}$. Using the operations in the field $(K, \oplus, \otimes)$ and the relation (18), for $s \in \{d-1, \dots, 2\}$ we can obtain the equalities

$$w_{s-1}(i) = w_0^{(2)}(i)^{-1}\left(w_s(i + \tau_{s-2}) \ominus w_s(i) \ominus w_1^{(s-1)}(i)\right)^2, \quad i \in I_*^{(2)}. \tag{19}$$

Note, that

$$J + l\tau_0 \subset I_*^{(2)}, \quad l \in \mathbb{N}_0.$$

It's evident, the equalities (19) let us restore $w_{d-2}[J], w_{d-2}[J + \tau_0], \dots, w_{d-2}[J + \tau_{d-4}]; \dots; w_3[J], w_3[J + \tau_1]; w_2[J]$ successively, by the sections $w_{d-1}[J]$, $w_{d-1}[J+\tau_0], \dots, w_{d-1}[J+\tau_{d-3}] = w_{d-1}[J+2^{d-4}\tau_1]$ of available sequence w_{d-1}.

Here, one can obtain the section $v[J]$ and LRS v. So, we can reconstruct the sequence v and its 2-adic digital sequences by its highest digital sequence w_{d-1} in $|\mathfrak{M}(v_0)|$ variants.

So for successful reconstruction v by w_0, w_1, w_{d-1} we need $\mathrm{O}(m2^d)$ elements of the sequence w_{d-1} and $\mathrm{O}(dm2^d)$ operations of the field K.

2.5 Rejection false candidates $w_1^{\langle\zeta\rangle}$ by the sequence w_{d-1}

Here we consider $d \geq 3$. Using the results of the section 2.4, if we have the sequences w_{d-1}, w_0, then we can restore the sequences $w_{d-2}^{\langle\zeta\rangle}, \dots, w_2^{\langle\zeta\rangle}$ successively by each candidate

$$w_1^{\langle\zeta\rangle} = w_1 \oplus \hat{\zeta}(w_0), \quad \hat{\zeta} \in \mathfrak{M}(w_0),$$

using last as genuine. So we obtain the sequence

$$v^{\langle\zeta\rangle} = w_0 + 2w_1^{\langle\zeta\rangle} + \cdots + 2^{d-2}w_{d-2}^{\langle\zeta\rangle} + 2^{d-1}w_{d-1}.$$

If considering candidate $w_1^{\langle\zeta\rangle}$ is genuine, that $v^{\langle\zeta\rangle} = v \in L_S(\Psi)$. To prove theorem 1.2 we have to prove the following statement.

Proposition 2.7. *If $v^{\langle\zeta\rangle} \in L_S(\Psi)$, then $\zeta = 0$ or there exists $\eta \in \mathfrak{M}^*(v_0)$, such that $\eta(v_0) = v_0^{(1)}$.*

Further we assume

$$w_0^{\langle\zeta\rangle} = w_0, \quad w_{d-1}^{\langle\zeta\rangle} = w_{d-1}.$$

It should be noted, that if $q = 2$, then $n = 1$, i.e. v is a classic MP LRS over $S = R = \mathbb{Z}_{2^d}$, and theorem 1.2 is proved in [10] without the requirement (8). So we consider only $q > 2$.

To prove this lemma we should prove the following three lemmas successively.

Lemma 2.8. *Under the conditions of theorem 2.7*

$$v_0^{(2)}\left(\zeta\left(v_0^{(1)}\right)^2 \oplus \zeta\left(v_0(i)\right)v_0^{(1)}\right) = 0. \tag{20}$$

Lemma 2.9. *Let $a, b, c, d \in L_{\Gamma(S)}(\gamma_0(\Psi))$, then the following equivalence takes place*

$$d(a^2 \oplus bc) = 0 \Leftrightarrow a^2 = bc. \tag{21}$$

From (20) and (21) we can obtain that

$$\zeta\left(v_0^{(1)}\right)^2 = \zeta(v_0)v_0^{(1)}. \tag{22}$$

Since the sequences $\zeta(v_0^{(1)}), \zeta(v_0), v_0^{(1)}$ are skew MP LRS over the field K of order m, the we can say that in the left part of (22) there are exactly $q^{m-1} - 1$ zeros on the period, and this mean in right too. This implies that in the sequences $\zeta(v_0)$ и $v_0^{(1)}$ zeros are in the same places.

Lemma 2.10. *Let $a, b \in L_{\Gamma(S)}(\gamma_0(\Psi))$ и $a(i) = 0 \Leftrightarrow b(i) = 0,\ \ i \in \overline{0, \tau_0 - 1}$, then there exists $\eta \in \mathfrak{M}^*(a)$ such that*

$$b = \eta(a).$$

So if $\zeta \neq 0$ that there exists $\eta \in \mathfrak{M}^*(v_0)$, for which

$$\eta(v_0) = v_0^{(1)}. \tag{23}$$

Now, the statement of the theorem 1.2 is followed from proposition 2.7 and [3].

2.6 Complexity

To summarize this section we denote, that for the successful reconstruction of a LRS v by its highest digit w_{d-1} we need to reconstruct all $|\mathfrak{M}(v_0)|$ candidates $v^{\langle\zeta\rangle}$ applying the algorithm from paragraph 2.4, and for each of these candidates we need to check, whether one is a MP LRS with characteristic polynomial $\Psi(x)$. So, before checking, we need $\mathrm{O}(m2^d)$ elements of the highest digit of v and $\mathrm{O}(md2^d) \cdot |\mathfrak{M}(v)| \leq \mathrm{O}(md2^{nd})$ operations of the field K. In the special case, if v_0 is non-reducible $\mathbb{Z}_{2^d}$-skew LRS, then $|\mathfrak{M}(v)| = 2$ and we need $\mathrm{O}(md2^d)$ operations. From experiments we can conclude, that for verification whether the sequence $v^{\langle\zeta\rangle}$ is correct, we need a section of w_{d-1}, which length is significantly less than the full period. This length (in practice) is estimated by $\mathrm{O}(m)$.

We need to point out, that above algorithm doesn't pretend to be the most effective, it is just a proving method of the theorem 1.2.

3 Conclusions

The class of word-oriented pseudorandom generators, for which we can prove the absence of equivalent keys, is described. In future research, it would be interesting to consider some generalizations of this class and to try to prove absence of equivalent keys for ones, for example to consider Galois rings of arbitrary characteristic ($p \geq 3$) or to try to revoke the additional contribution (8).

References

[1] A.S. Kuzmin, A.A. Nechaev. *Reconstruction of a linear recurrence of maximal period over a Gaolis ring from its highest coordinate sequence* // Diskr. Mat., 2011, 23:2, pp. 3–31.

[2] M.A. Goltvanitsa, S.N. Zaitsev, A.A. Nechaev. *Skew linear recurring sequences of maximal period over Galois rings* // Fundam. Prikl. Mat., 2012, 17:3, pp. 5–23.

[3] Zaitsev S. N. *Description of maximal skew linear recurrences in terms of multipliers* // Mat. vopr. kriptorg., 2014, 5:2, pp. 57–70.

[4] B. Preneel, *Introduction to the Proceedings of the Fast Software Encryption 1994 Workshop* (Ed. Bart Preneel), LNCS 1008 (1995) 1–5.

[5] Kurakin V. L., Kuzmin A. S., Mikhalev A. V., Nechaev A. A. *Linear recurring sequences over rings and modules.* J. Math. Sci. 76 (1995), є 6, 2793–2915.

[6] A.S. Kuzmin, A.A. Nechaev. *Linear recurring sequences over Galois rings* // Algebra Logika, (1995) 3, No2, c. 169–189.

[7] R. Lidl and H. Niederreiter, *Finite Fields*, Addison-Wesley, London (1983).

[8] G. Birkhoff and T. C. Bartee, *Modern Applied Algebra*. McGraw-Hill, New York (1970).

[9] A. A. Nechaev, *Kerdoc code in a cyclic form*, Diskr. Math.,1, No. 4, 123–139 (1989).

[10] A. S. Kuzmin and A. A. Nechaev *A construction of noise stable codes using linear recurrences over Galois rings*, Uspekhi Mat. Nauk,47, No. 5, 183–184 (1992).

[11] A. S. Kuzmin, *Distribution of elements on cycles of linear recurring sequences over residue rings*, Uspekhi Mat. Nauk,47, No. 6, 213–214 (1993).

[12] A. S. Kuzmin, *Lower bounds of ranks for coordinate sequences of linear recurring sequences over residue rings*, Uspekhi Mat. Nauk,48 (1993).

[13] A. S. Kuzmin, *On periods of binary digits of linear recurring sequences over prime finite fields*, Uspekhi Mat. Nauk,48 (1993).

[14] V. L. Kurakin, *The first coordinate sequence of a linear recurrence of maximal period over a Galois ring*, Diskr. Mat. 1994, 6:2, pp. 88–100.

[15] V. L. Kurakin, *Representations of linear recurring sequences of maximal period over a finite field*, Diskr. Mat. 1995, 7:2, pp. 34–39.

[16] V. L. Kurakin, *Representations of linear recurring sequences and regular prime numbers*, Uspekhi Mat. Nauk,47, No. 6, 215–216 (1992).

[17] A. A. Nechaev, *The cyclic types of linear substitutions over finite commutative rings*, Mat. Sb.,184, No. 3, 21–56 (1993).

[18] A. A. Nechaev, *Linear recurring sequences over quasi-Frobenius modules*, Uspekhi Mat. Nauk, 48 (1993).

[19] Eichenauer-Herrman, H. Grothe, and J. Lehn, *On the period length of pseudorandom vector-sequences generated by matrix generators*, Math. Comput.,2, No. 185, 145Ц148 (1989).

[20] B. Jansson, *Random Number Generators*, Almqvist and Wiksell, Stockholm (1966).

[21] A. A. Nechaev, *Linear recurring sequences over commutative rings*, Diskr. Math.,3, No. 4, 107–121 (1991).

[22] A.S. Kuzmin, G.B. Marshalko, A.A. Nechaev *Reconstruction of linear recurrent sequence over prime residue ring from its image* // Mat. vopr. kriptogr., (2010) 1, No2, 31–56.

[23] Min-Qiang Huang, Zong-Duo Dai, Projective maps of linear recurring sequences with maximal p l-adic periods. Fibonacci Quart. (1992) 30, No2, 139-143.

[24] Xuan-Yong Zhu, Wen-Feng Qi, Compression mappings on primitive sequences over Z_{2^n}. IEEE Trans. Inform. Theory (2004) 50, No10, 2442-2448.

[25] Xuan-Yong Zhu, Wen-Feng Qi, Further result of compressing maps on primitive sequences modulo odd prime powers. IEEE Trans. Inform. Theory (2007) 53, No8, 2985-2990.

[26] Tian Tian, Wen-Feng Qi, Injectivity of compressing map on primitive sequences over Z_{p^l}. IEEE Trans. Inform. Theory (2007) 53, No8, 2960-2966.

[27] Zong-Duo Dai, Binary sequences derived from ML-sequences over rings I: periods and minimalpolynomials. J. Cryptology (1992) 5, No4, 193-207.

[28] Weng-Feng Qi, Xuan-Yong Zhu, Compressing maps on primitive sequences over Z_{2^n} and its Galois extension. Finite Fields Appl. (2002) 8, No4, 570-588.

[29] Weng-Feng Qi, Compressing maps on primitive sequences over Z_{2^n} and analysis of their derivative sequences, PhD Thesis. Zhengzhou Inform. Eng. Univ., Zhengzhou, China, 1997.

[30] Weng-Feng Qi, Jun-Hui Yang, Jin-Jun Zhou, ML-sequences over rings Z_{2^n}. Lecture Notes Computer Sci. (1998) 1514, 315-325.

[31] P. Hawkes and G.G. Rose, *Primitive Specification and Supporting Documentation for SOBER-t16 Submission to NESSIE*, Proceedings of the first NESSIE Workshop (2000)

[32] P. Hawkes and G.G. Rose, *Primitive Specification and Supporting Documentation for SOBER-t32 Submission to NESSIE*, Proceedings of the first NESSIE Workshop, (2000)

[33] P. Ekdahl and T. Johansson, *SNOW – a new stream cipher*, Proceedings of the first NESSIE Workshop, (2000)

[34] P. Ekdahl and T. Johansson, *A New Version of the Stream Cipher SNOW*, Selected Areas in Cryptography, LNCS 2595 (2002) 47–61

[35] P. Hawkes and G.G. Rose. *Turing: A Fast Stream Cipher*, Fast Software Encryption, LNCS 2887 (2003) 24–26

The First Digit Sequence of Skew Linear Recurrence of Maximal Period over Galois Ring

Mikhail Goltvanitsa

Abstract

A rank of the first digit sequence of a skew linear recurrence of maximal period is determined under natural restrictions on the digit set.

1 Introduction. Main results

In what follows, $R = GR(q^d, p^d)$, where $q = p^r$, is a Galois ring, $S = GR(q^{nd}, p^d)$ is a Galois extension of dimension n of R [1]. It is known that the group $\mathrm{Aut}(S/R)$ of automorphisms of S over R is a cyclic group of order n [2]. Let σ be a generator of this group and $\check{S} = S^{\sigma}\langle\sigma\rangle$ be a skew group ring of the group $\langle\sigma\rangle$ over the ring S, i.e. the set of formal sums $\psi = \sum_{i=0}^{n-1} s_i\sigma^i$, $s_0, ..., s_{n-1} \in S$, with natural addition and multiplication, which is defined using distributive property by the identity $\forall s \in S: \quad \sigma s = \sigma(s)\sigma$.

Each element $\psi \in \check{S}$ defines an endomorphism of the module ${}_RS$ such that $\psi(s) = \sum_{i=0}^{n-1} s_i\sigma^i(s)$ for every $s \in S$. So we have the isomorphisms $\check{S} \cong \mathrm{End}({}_RS) \cong R_{n,n}$, where $R_{n,n}$ is a ring of $n \times n$ matrices over R. The equality $\psi \cdot s = \psi(s)$ defines on S a structure of the left $\check{S}$-module.

The set $S^{\langle 1\rangle}$ of all sequences over S is a left module over the ring of polynomials $\check{S}[x]$, where a product of the sequence $v \in S^{\langle 1\rangle}$ and the polynomial $A(x) = \sum_{i\geq 0} a_ix^i \in \check{S}[x]$ is defined by the equality

$$A(x)v = w \in S^{<1>}: \quad w(j) = \sum_{i\geq 0} a_i(v(i+j)),\ j \geq 0.$$

Following [3], we say that a sequence $v \in S^{\langle 1\rangle}$ is a *skew linear recurrent sequence* (*LRS*) of order $m > 0$ over the ring S if it is an LRS of order

m over the module ${}_{\check{S}}S$ [1, 4], i.e. $\Psi(x)v = 0$ for some monic polynomial $\Psi(x) = x^m - \psi_{m-1}x^{m-1} - ... - \psi_0 \in \check{S}[x]$ of degree m, called *characteristic polynomial of LRS v*. In other words, the sequence v satisfies the law of recursion

$$\forall i \in \mathbb{N}_0 : \quad v(i+m) = \psi_{m-1}(v(i+m-1)) + ... + \psi_0(v(i)).$$

Denote by $L_S(\Psi)$ the set of all LRS v over S with characteristic polynomial Ψ. If $\Psi(x) \in S[x]$, then we say that the sequences from $L_S(\Psi)$ are *classic LRS*.

Proposition 1. ([3]) *For any skew LRS v of order m over S there exists a monic polynomial $F(x) \in R[x]$ of degree mn with the property $v \in L_S(F)$. Moreover, $T(F) \leq \tau = (q^{nm} - 1)p^{d-1}$.*

If $T(v) = \tau$ we say that v is a skew LRS of *maximal period* (*MP LRS*). Studying of skew MP LRS over Galois rings was started in articles [3, 5], where, in particular, the results allowing to construct large classes of such sequences without brute force method were obtained.

A subset $\mathcal{B} \subset S$ is called *digit set (of the ring S)*, if $0 \in \mathcal{B}$ and for every $a \in S$ there exists a unique element $\varkappa^{\mathcal{B}}(a) \in \mathcal{B}$ such that $\overline{a} = \overline{\varkappa^{\mathcal{B}}(a)}$, where $\overline{a}$ is an image of a under canonical epimorphism $\mu : S \to S/pS$. So, every element $a \in S$ has a unique decomposition $a = \varkappa_0^{\mathcal{B}}(a) + \ldots + p^{d-1}\varkappa_{d-1}^{\mathcal{B}}(a)$, $\varkappa_j^{\mathcal{B}}(a) \in \mathcal{B}$, $j \in \overline{0, d-1}$, where the notation $\overline{0, d-1}$ stands for $\{0, 1, \ldots, d-1\}$. We call a function $\varkappa_t^{\mathcal{B}} : S \to \mathcal{B}$ *the t-th digit function (in the digit set $\mathcal{B}$)*. The set $\mathcal{B}$ has a structure of the field of q^n elements with respect to operations $a \oplus b = \varkappa_0^{\mathcal{B}}(a+b)$, $a \odot b = \varkappa_0^{\mathcal{B}}(ab), a, b \in \mathcal{B}$. An important example of a digit set is $\Gamma(S) = \{\pi \in S : \pi^{q^n} = \pi\}$, called p-adic digit set. We use the notation γ_t for the t-th digit function $\varkappa_t^{\Gamma(S)} : S \to \Gamma(S)$. For any sequence $w \in S^{\langle 1 \rangle}$ there exists a unique tuple $w_0, \ldots, w_{d-1}$ of its *digit sequences (in digit set $\mathcal{B}$)*, defining from the relations $w(i) = w_0(i) + \ldots + p^{d-1}w_{d-1}(i)$, where $w_j(i) \in \mathcal{B}, j \in \overline{0, d-1}$. Digit sequences of skew linear recurrences were studied in article [6]. For any sequence $w \in \mathcal{B}^{\langle 1 \rangle}$ we denote by $\mathrm{rank}_B w$ the rank (the degree of minimal polynomial) of the sequence w as LRS over the field $\mathcal{B}$ [7].

Let $\Gamma(S) = \{\pi_0 = 0, \ldots, \pi_{q^n-1}\}$. Further, we assume that if $\mathcal{B}$ is an arbitrary digit set of S, $\mathcal{B} = \{\mu_0, \ldots, \mu_{q^n-1}\}$, then $\mu_0 = 0$ and $\overline{\mu}_t = \overline{\pi}_t$ for $t = 1, \ldots, q^n - 1$. The following result is valid.

Lemma 2. ([9]) *There exists a unique polynomial* $\Lambda_{\mathcal{B}}(x) = \sum\limits_{j=1}^{q^n-1} \lambda_j x^j$ *over* S *such that* $\Lambda(\pi_t) = \mu_t$, $t \in \overline{0, q^n - 1}$. $\Lambda_{\mathcal{B}}(x) = x + p\Lambda^*_{\mathcal{B}}(x)$, *where* $\Lambda^*_{\mathcal{B}}(x) = \sum_{j=1}^{q^n-1} \lambda^*_j x^j \in S[x]$ *is a polynomial uniquely defined modulo* p^{d-1}.

We say $\Lambda_{\mathcal{B}}(x)$ is an *interpolation polynomial* of the digit set $\mathcal{B}$ and put $L_{\mathcal{B}} = \{l \in \overline{2, q^n - 1} : \lambda_l \not\equiv 0(\mathrm{mod} p^2)\}$. Under certain restrictions on this polynomial one can exactly determine a rank of the first digit sequence of the skew linear recurrence of maximal period.

We define a *p-ary weight* of number $k = \sum_{s \geq 0} p^s \nu_s(k)$, where $\nu_s(k) \in \overline{0, p-1}$ as an arithmetic sum $w_p(k) = \sum_{s \geq 0} \nu_s(k)$.

Let v be a skew MP LRS of order m over S and $K = GR(q^{nmd}, p^d)$ be an extension of dimension m of the ring S. Denote by $\tilde{\sigma}$ the generator of the group $\mathrm{Aut}(K/R)$, taking each element $\eta \in K$ to $\tilde{\sigma}(\eta) = \gamma_0(\eta)^q + p\gamma_1(\eta)^q + \ldots + p^{d-1}\gamma_{d-1}(\eta)^q$ [2]. It is known [3] that the i-th term $v(i)$ of the sequence v has a representation

$$v(i) = \mathrm{Tr}^K_S(\sum_{j=0}^{n-1} \varepsilon_j \tilde{\sigma}^j(\vartheta)^i), \tag{1.1}$$

for some $\varepsilon_0, \ldots, \varepsilon_{n-1}, \vartheta \in K$, $\mathrm{ord}\vartheta = (q^{mn} - 1)p^{d-1}$, where Tr^K_S is a trace–function from the ring K onto the ring S [10], defined by the rule $\mathrm{Tr}^K_S(\eta) = \sum_{l=0}^{m-1} \tilde{\sigma}^{ln}(\eta)$, for all $\eta \in K$.

Define

$W_0 = \{j \in \overline{0, n-1} : \gamma_0(\varepsilon_j) \neq 0\}, W_1 = \{j \in \overline{0, n-1} \setminus W_0 : \gamma_1(\varepsilon_j) \neq 0\}$.

In this article we prove the following the results, announced in0 [6].

Theorem 3. *If* $\Lambda(x) = x + p\Lambda^*(x)$ *is an interpolation polynomial of the digit set* $\mathcal{B}$ *such that* $\mathrm{deg}\Lambda(x) \leq q - 1$, *then for a skew MP LRS* v *represented by* (1.1) *such that* $|W_0| = n_0, |W_1| = n_1$ *the following is true:*

$\mathrm{rank}_{\mathcal{B}} \varkappa^{\mathcal{B}}_1(v) = \sum\limits_{l \in L_{\mathcal{B}}} \prod\limits_{s=0}^{r-1} \binom{mn_0 + \nu_s(l) - 1}{\nu_s(l)} + \binom{mn_0 + p - 1}{p} + m(n_0 + n_1),$

in particular,
$\operatorname{rank}_{\Gamma(S)}\gamma_1(v) = m(n_0+n_1) + \binom{mn_0+p-1}{p}$.

For comparison we reproduce here the following known result.

Theorem 4. ([9]) *Let polynomial $\Lambda_{\mathcal{B}}(x)$ be an interpolation polynomial of the digit set $\mathcal{B}$. Then the rank of the first digit sequence of the classical MP LRS $u \in S^{\langle 1\rangle}$ of order m is equal to*
$\operatorname{rank}_{\mathcal{B}}\varkappa_1^{\mathcal{B}}(u) = \sum\limits_{l\in L_{\mathcal{B}}}\prod\limits_{s=0}^{nr-1}\binom{m+\nu_s(l)-1}{\nu_s(l)} + \binom{m+p-1}{p} + m.$

So taking into account Theorem 3 and Theorem 4, we obtain

Corollary 5. *Under conditions of Theorem 3 if $n_0 = n$, then the rank of the first digit sequence of skew MP LRS v of order m over S is equal to the rank of the first digit sequence of classical MP LRS u of order mn over S.*

The author is grateful to Professor A. A. Nechaev for helpful discussions and valuable remarks.

2 Proof of Theorem 3

It is known [2] that for every $a \in S$: $\gamma_0(a) \equiv a^{q^n} (\bmod p^2)$. Hence,

$$p\varkappa_1^{\mathcal{B}}(v) \equiv v - \Lambda(v^{q^n}) = v - v^{q^n} - p\lambda_1^* v - p\sum_{l\in L}\lambda_l^* v^l \ (\bmod p^2). \qquad (2.1)$$

Using (1.1), we have:
$v(i) = \operatorname{Tr}_S^K(z(i)) \equiv \sum_{k=0}^{m-1}(z_0(i)^{q^{nk}} + pz_1(i)^{q^{nk}})$ $(\bmod p^2)$, where $z(i) = \sum_{j=0}^{n-1}\varepsilon_j\sigma^j(\vartheta)^i$, $z_k(i) = \gamma_k(z(i)), k = 1, 2$. Hence,

$$v(i)^{q^n} \equiv \left(\sum_{k=0}^{m-1} z_0(i)^{q^{nk}}\right)^{q^n} = \sum_{k=0}^{m-1} z_0(i)^{q^{n(k+1)}} +$$
$$+ \sum_{\substack{0\le j_0,\dots,j_{m-1}<q^n \\ j_0+\dots+j_{m-1}=q^n}} \frac{q^n!}{j_0!\dots j_{m-1}!} z_0(i)^{j_0+q^n j_1+\dots+q^{n(m-1)}j_{m-1}} (\bmod\ p^2). \qquad (2.2)$$

Since $z_0(i) \in \Gamma(K)$, we obtain $\sum_{k=0}^{m-1} z_0(i)^{q^{n(k+1)}} = \sum_{k=0}^{m-1} z_0(i)^{q^{nk}}$.

All multinomial coefficients in the second sum of (2.2) are divisible by p, so we can write (2.1) in the following form
$p\varkappa_1^B(v)(i) \equiv pu(i) - pw(i) - p\tilde{w}(i) (\text{mod } p^2)$, where

$$u(i) = \sum_{k=0}^{m-1} z_1(i)^{q^{nk}} - \lambda_1^* v_0(i),\ \tilde{w}(i) = \sum_{l\in L} \lambda_l^* v_0(i)^l,$$
$$w(i) = \sum_{\substack{0\le j_0,\dots,j_{m-1}<q^n \\ j_0+\dots+j_{m-1}=q^n}} \frac{q^n!/p}{j_0!\dots j_{m-1}!} z_0(i)^{j_0+q^n j_1+\dots+q^{n(m-1)}j_{m-1}},$$

where $v_0 = \gamma_0(v)$. Further proof is carried out in the form of chain of lemmas.

Lemma 6. $\text{rank}_{\Gamma(K)}\gamma_0(u) = m\binom{n_0+p-1}{n_0} + m(n_0+n_1)$.

Lemma 7. $\text{rank}_{\Gamma(K)}\gamma_0(w) = \sum\limits_{\substack{0\le l_0,\dots,l_{m-1}<p \\ l_0+\dots l_{m-1}=p}} \prod_{s=0}^{m-1} \binom{l_s+n_0-1}{l_s}$.

Lemma 8. $\text{rank}_{\Gamma(K)}\gamma_0(\tilde{w}) = \sum\limits_{l\in L}\prod\limits_{s=0}^{r} \binom{mn_0+\nu_s(l)-1}{\nu_s(l)}$.

Lemma 9. *Minimal polynomials of the sequences $\gamma_0(u), \gamma_0(w), \gamma_0(\tilde{w})$ are coprime.*

Lemma 10. $\text{rank}_{\Gamma(K)}\gamma_0(u) + \text{rank}_{\Gamma(K)}\gamma_0(w) = m(n_0+n_1) + \binom{mn_0+p-1}{p}$.

Proof of Lemma 6.

□ In [11, 3] it is proved that $\vartheta_0 \ne 0, \vartheta_1 \ne 0$, so we have

$$z(i) \equiv \sum_{j=0}^{n-1} (\varepsilon_{j0} + p\varepsilon_{j1})(\vartheta_0^{q^j} + p\vartheta_1^{q^j})^i \equiv$$
$$\equiv \sum_{j=0}^{n-1} \varepsilon_{j0}\vartheta_0^{q^j i} + p(\sum_{j=0}^{n-1} \varepsilon_{j1}\vartheta_0^{q^j i} + i\sum_{j=0}^{n-1} \varepsilon_{j0}(\tfrac{\vartheta_1}{\vartheta_0})^{q^j}\vartheta_0^{q^j i}) \ (\text{mod } p^2),$$

where $\varepsilon_{jk} = \gamma_k(\varepsilon_j)$. Hence,

$$z_1(i) = \gamma_1(\sum_{j=0}^{n-1} \varepsilon_{j0}\vartheta_0^{q^j i}) \oplus \gamma_0(\sum_{j=0}^{n-1} \varepsilon_{j1}\vartheta_0^{q^j i} + i\sum_{j=0}^{n-1} \varepsilon_{j0}(\frac{\vartheta_1}{\vartheta_0})^{q^j}\vartheta_0^{q^j i}). \qquad (2.3)$$

Without loss of generality we can assume $W_0 = \{0, 1, \dots, n_0 - 1\}$.

Theorem 11. ([12]) *If $a_1, \ldots, a_k \in \Gamma(K)$, $\hat{r} = q^{mn}/p$, then*

$$\gamma_1(a_1 + \ldots + a_k) \equiv \sum_{\substack{j_0+\ldots+j_k=p \\ o \le j_i < p}} \frac{1}{j_0! \ldots j_k!} a_1^{\hat{r}j_1} \ldots a_k^{\hat{r}j_k} \pmod p \tag{2.4}$$

Using Theorem 11, we have

$$\begin{aligned} u(i) &= \sum_{k=0}^{m-1} z_1(i)^{q^{nk}} - \lambda_1^* v_0(i) \equiv \\ &\equiv \sum_{k=0}^{m-1} \sum_{\substack{s_0+\ldots+s_{n_0-1}=p \\ o \le s_k < p}} \frac{1}{s_0! \ldots s_{n_0}!} \vartheta_0^{\hat{r}q^{nk}(s_0+\ldots+s_{n_0-1}q^{n_0-1})i} \prod_{j \in W_0} \varepsilon_{j0}^{s_j \hat{r} q^{nk}} + \\ &+ \sum_{k=0}^{m-1} \sum_{j=0}^{n-1} \varepsilon_{j1}^{q^{nk}} \vartheta_0^{iq^{nk+j}} + i \sum_{k=0}^{m-1} \sum_{j \in W_0} \varepsilon_{j0}^{q^{nk}} (\tfrac{\vartheta_1}{\vartheta_0})^{q^{nk+j}} \vartheta_0^{iq^{nk+j}} - \\ &- \lambda_1^* \sum_{k=0}^{m-1} \sum_{j \in W_0} \varepsilon_{j0}^{q^{nk}} \vartheta_0^{iq^{nk+j}} (\bmod p). \end{aligned} \tag{2.5}$$

Let $\sum_k$ be the k-th double sum from (2.5), $k \in \overline{1,4}$, and let $y(i) = \sum_2 + \sum_3 + \sum_4$, $\tilde{y} = \sum_1$. It can be easily seen that the minimal polynomials of the sequences $\gamma_0(y)$ and $\gamma_0(\tilde{y})$ are coprime and, therefore, $\mathrm{rank}_{\Gamma(K)}\gamma_0(y + \tilde{y}) = \mathrm{rank}_{\Gamma(K)}\gamma_0(y) + \mathrm{rank}_{\Gamma(K)}\gamma_0(\tilde{y})$.

Taking into account (2.5) we obtain the decomposition of y into the sum of binomial sequences of the first and second order [7], hence, $\mathrm{rang}_{\Gamma(\mathrm{S})}\gamma_0(y) = 2mn_0 + mn_1$.

Proposition 12. [13] *For every $a, b \in \mathbb{N}_0$, the equation $x_1 + \ldots + +x_a = b$ has $\binom{a+b-1}{b}$ solutions in nonnegative integers.*

Using Proposition 12, we obtain that $\mathrm{rank}_{\Gamma(K)}(u)$ is equal to the value $m(\binom{n_0+p-1}{p} - n_0) + 2mn_0 + mn_1 = m\binom{n_0+p-1}{p} + m(n_0 + n_1)$. □

Proof of Lemma 7.

□

Lemma 13. ([14]) *For every $k \in \mathbb{N}_0$ and every prime p the largest integer i with the property $p^i | k!$ is equal to $\frac{k - w_p(k)}{p-1}$.*

Therefore, multinomial coefficients in the second sum of (2.2) are divisible by p for every tuple $(j_0, \dots, j_{m-1}) \in \overline{0, q^n - 1}^m$ such that $j_0 + \dots j_{m-1} = q^n$ and not divisible by p^2 exactly if

$$j_s = \frac{q^n}{p} l_s,\ l_s \in \overline{0, p-1}, s \in \overline{0, m-1}. \tag{2.6}$$

Let $c = c(l_0, \dots, l_{m-1}) = \dfrac{q^n}{p}(l_0 + q^n l_1 \dots + q^{n(m-1)} l_{m-1})$, then

$$z(i)_0^c \equiv \sum_{\substack{0 \le k_0, \dots, k_{n_0-1} \le c \\ k_0 + \dots + k_{n_0-1} = c}} \frac{c!}{k_0! \dots k_{n_0-1}!} \cdot \vartheta_0^{(k_0 + \dots k_{n_0-1} q^{n_0-1}) i} \prod_{j=0}^{n_0-1} (\varepsilon_{j0}^{k_j}) (\mathrm{mod} p). \tag{2.7}$$

Theorem 14. (*Lukas′* [*15, Theorem* 4.71]) *If p is prime number, $N = \sum_{i=0}^{I} N_i p^i$, $M = \sum_{i=0}^{I} M_i p^i \in \mathbb{N}_0$, then* $\binom{N}{M} \equiv \prod_{i=0}^{I} \binom{N_i}{M_i}$ (mod p)

By Theorem 14 we conclude that the multinomial coefficient in sum (2.7) is not divisible by p exactly if

$$\begin{array}{l} k_j = \frac{q^n}{p}(l_{0j} + l_{1j} q^n + \dots + l_{m-1,j} q^{n(m-1)}),\ j \in W_0, \\ \sum_{j \in W_0} l_{sj} = l_s,\ l_{sj} \in \overline{0, p-1},\ s \in \overline{0, m-1},\ l_{sj} \in \mathbb{N}_0. \end{array} \tag{2.8}$$

Taking into account (2.8) it is easy to see that elements

$$\vartheta_0^{k_0 + k_1 q + \dots + k_{n_0-1} q^{n-1}} = \vartheta_0^{\frac{q^n}{p} \sum_{j \in W_0} \sum_{s=0}^{m-1} l_{sj} q^{ns+j}} \tag{2.9}$$

are distinct, because for every $a, b \in \Gamma(K)$ such that $a \ne b$ inequality $a^{\frac{q^n}{p}} \ne b^{\frac{q^n}{p}}$ is fulfilled and all values $\sum_{j \in W_0} \sum_{s=0}^{m-1} l_{sj} q^{ns+j}$ are distinct and less then $\mathrm{ord}\vartheta_0 = q^{mn} - 1$.

So, using (2.6) and (2.7) we get the decomposition of the sequence w into binomial sequences with distinct roots. Finally, taking into account (2.8) and Proposition 12 we obtain the desired result □

Proof of Lemma 8.

□ For every $l \in L$, using relation $v(i)^l \equiv \left(\sum\limits_{k=0}^{m-1} z_0(i)^{q^{nk}}\right)^l (\mathrm{mod} p)$, we get

$$v(i)^l \equiv \sum_{\substack{0 \le l'_0,\ldots,l'_{m-1} \le l \\ l'_0+\ldots+l'_{m-1}=l}} \sum_{\substack{0 \le k_0,\ldots,k_{n_0-1} \le c \\ k_0+\ldots+k_{n_0-1}=c}} \frac{l!}{l'_0!\ldots l'_{m-1}!} \frac{c!}{k_0!\ldots k_{n_0-1}!} \prod_{j\in W_0} \varepsilon_{j0}^{k_j} \cdot \vartheta_0^{(k_0+k_1q+\ldots+k_{n_0-1}q^{n_0-1})i} (\mathrm{mod} p), \tag{2.10}$$

where $c = l'_0 + l'_1 q^n + \ldots + l'_{m-1} q^{n(m-1)}$. If for nonnegative integers $a_1, \ldots, a_k, b$ such that $a_1 + \ldots + a_k = b$ the following condition $\forall t \in \mathbb{N}_0 \;\; \nu_t(b) = \nu_t(a_1) + \ldots + \nu_t(a_k)$ is fulfilled, then we will write $b = a_1 \dot{+} \ldots \dot{+} a_k$. Taking into account Theorem 14, we get that the product of multinomial coefficients in (2.10) is not divisible by p exactly if

$$l = l'_0 \dot{+} l'_1 \dot{+} \ldots \dot{+} l'_{m-1}; c = k_0 \dot{+} k_1 \dot{+} \ldots \dot{+} k_{n_0-1}. \tag{2.11}$$

Since $c = l'_0 + l'_1 q^n + \ldots + l'_{m-1} q^{n(m-1)}$, combining the second equality in (2.11) and Theorem 14, we get

$$\forall j \in W_0 \; k_j = \sum_{s=0}^{m-1} l'_{sj} q^{ns}, \quad \sum_{j \in W_0} l'_{sj} = l'_s, \tag{2.12}$$

for some $l'_{sj} \in \mathbb{N}_0$. Let us prove that

$$l = l'_{00} \dot{+} l'_{01} \ldots \dot{+} l'_{0,n_0-1} \dot{+} \ldots \dot{+} l'_{m-1,0} \dot{+} \ldots \dot{+} l'_{m-1,n_0-1}. \tag{2.13}$$

Taking into account inequality $l \le q-1$, it suffices to prove that $\nu_b(l) = \sum_{j\in W_0} \sum_{s=0}^{m-1} \nu_b(l'_{sj})$, for all $b \in \overline{0, r-1}$. It is easily to seen that for all $a, b \in \mathbb{N}_0,\; a < m, b < rn$ the following condition holds $\nu_{arn+b}(c) = \nu_b(l'_a)$. On the other hand, using (2.11), (2.12), we have
$\nu_{arn+b}(c) = \sum_{j=0}^{n_0-1} \nu_{arn+b}(k_j) = \sum_{j=0}^{n_0-1} \nu_b(l'_{aj})$. So, using the first equality in (2.11), we obtain $\nu_b(l) = \sum_{j\in W_0} \sum_{s=0}^{m-1} \nu_b(l'_{sj})$, for all $a \in \overline{0, m-1}, b \in \overline{0, rn-1}$. Hence (2.13) is proved.

It is easy to see that from conditions (2.13), (2.12) condition (2.11) follows. So conditions (2.13) and (2.11) are equivalent. Hence, the number of non-zero terms in (2.10) is equal to the number of decompositions of l in the form (2.13). Using Proposition 12, we obtain that this number is equal to $\prod_{s=0}^{r-1}\binom{mn_0+\nu_s(l)-1}{\nu_s(l)}$. Note that (2.10) is decomposition of v into the sum of binomial sequences with roots of the form

$$\vartheta_0^{\sum_{s=0}^{m-1}\sum_{j\in W_0} l'_{sj}q^{ns+j}}. \tag{2.14}$$

Since $l'_{sj} \le l \le q-1$ and $\mathrm{ord}\vartheta_0 = q^{mn}-1$, we obtain that all elements (2.14) are distinct. So, the following equality is fulfilled

$$\mathrm{rank}_{\Gamma(S)}\gamma_0(v^l) = \prod_{s=0}^{r-1}\binom{mn_0+\nu_s(l)-1}{\nu_s(l)}.$$

Finally, note that for distinct $l_1, l_2 \in L$ minimal polynomials of the sequences $\gamma_0(v^{l_1})$ и $\gamma_0(v^{l_2})$ are coprime. $\square$

Proof of Lemma 9.

$\square$ Let us prove that minimal polynomials of the sequences $\gamma_0(u)$ and $\gamma_0(w)$ are coprime. It suffices to show that the set of roots of binomial sequences in decomposition (2.5) does not intersect with the set of elements (2.9). Assume the contrary, that for some $k \in \overline{0, m-1},\ t \in \overline{0, n-1}$ one of the equalities is fulfilled

$$\vartheta_0^{\hat{r}q^{nk}(b_0+b_1q+\dots b_{n_0-1}q^{n_0-1})} = \vartheta_0^{\frac{q^n}{p}\sum_{j\in W_0}\sum_{s=0}^{m-1} l_{sj}q^{ns+j}}, \tag{2.15}$$

$$\vartheta_0^{q^{nk+t}} = \vartheta_0^{\frac{q^n}{p}\sum_{j\in W_0}\sum_{s=0}^{m-1} l_{sj}q^{ns+j}}, \tag{2.16}$$

where $\hat{r} = \frac{q^{mn}}{p}$, l_{sj} satisfy condition (2.8), and $\sum\limits_{j\in W_0} b_j = p, 0 \le b_j < p$.

Let us show that equality (2.15) is not fulfilled (For equality (2.16) the proof is analogous). If $k = 0$, then condition (2.15) is equivalent to condition

$$\vartheta_0^{q^{n(m-1)}(b_0+b_1q+\dots b_{n_0-1}q^{n_0-1})} = \vartheta_0^{\sum_{j=0}^{n_0-1}\sum_{s=0}^{m-1} l_{sj}q^{ns+j}}. \tag{2.17}$$

Using (2.8), we have

$$\sum_{j\in W_0} l_{sj} = l_s,\ l_{sj} \in \overline{0, p-1},\ s \in \overline{0, m-1}, \sum_{s=0}^{m-1} l_s = p,\ 0 \le l_s < p,$$

therefore, there exist $0 \leq s_1 \leq m-1, 0 \leq s_2 \leq m-1, s_1 \neq s_2$ such that $l_{s_1} \neq 0, l_{s_2} \neq 0$. Hence, there exist $j_1, j_2 \in W_0$ such that $l_{s_1 j_1} \neq 0,\ l_{s_2 j_1} \neq 0$. So, equation (2.17) is not fulfilled, since the exponents on ϑ_0 in the left-hand side and right-hand side in (2.17) are different and less then ordϑ_0.

If $0 < k \leq m-1$, then condition (2.15) is equivalent to condition

$$\vartheta_0^{q^{n(k-1)}(b_0+b_1q+\ldots b_{n_0-1}q^{n_0-1})} = \vartheta_0^{\sum_{j=0}^{n_0-1}\sum_{s=0}^{m-1} l_{sj}q^{ns+j}}. \tag{2.18}$$

In this case the proof is analogous to the proof in the case $k = 0$.

Let us show that minimal polynomials of the sequences $\gamma_0(w)$ and $\gamma_0(\tilde{w})$ are coprime. It suffices to show that the set of elements of the form (2.9) does not intersect with the set of elements of the form (2.14). Assume the contrary that for some tuple l_{sj}, where $s \in \overline{0, m-1}, j \in W_0$ such that $\sum_{j\in W}\sum_{s=0}^{m-1} l_{sj} = p,\ 0 \leq l_{sj} < p$ and l of the form (2.13) the equality is fulfilled

$$\vartheta_0^{\sum_{j\in W_0}\sum_{s=0}^{m-1} l'_{sj}q^{ns+j}} = \vartheta_0^{\frac{q^n}{p}\sum_{j\in W_0}\sum_{s=0}^{m-1} l_{sj}q^{ns+j}}. \tag{2.19}$$

Denote

$$a \equiv p\sum_{j\in W_0}\sum_{s=0}^{m-1} l'_{sj}q^{ns+j} (\mathrm{mod}\ (q^{mn}-1)),$$
$$b \equiv q^n\sum_{j\in W_0}\sum_{s=0}^{m-1} l_{sj}q^{ns+j} (\mathrm{mod}\ (q^{mn}-1)),\ 0 \leq a, b < q^{mn}-1.$$

Equality (2.19) is equivalent to the condition $a = b$. Now we consider two cases $q = p$ and $q > p$. If $q = p$, then
$w_p(a) = l \leq q-1 = p-1 < p, w_p(b) = p$. So, $a \neq b$ and condition (2.19) is not fulfilled. This is a contradiction. Let us consider the case $q > p$, i.e $r = \log_p q > 1$. In this case there are two possible subcases. If the following condition is fulfilled

$$\exists s_0 \in \overline{0, m-1}, j_0 \in W_0 :\ p^{r-1} \not| l'_{s_0 j_0} \neq 0, \tag{2.20}$$

then there exists $t > 0$ such that

$$r \not| t,\ \nu_t(a) \neq 0. \tag{2.21}$$

It is easy to seen, that condition (2.21) does not fulfilled for b. This is a contradiction. If condition (2.20) is not valid, i.e

$$\forall s \in \overline{0, m-1}, j \in W_0 : \; p^{r-1} | l'_{s_0 j_0}, \tag{2.22}$$

then $l'_{sj} = p^{r-1} h_{sj}$. Taking into account (2.13), we obtain that $w_p(a) = \sum_{j\in W_0} \sum_{s=0}^{m-1} h_{sj} < p$. From the other hand $w_p(b) = p$. Hence, $a \neq b$. This is a contradiction. So, minimal polynomials of the sequences $\gamma_0(w)$ and $\gamma_0(\tilde{w})$ are coprime. The proof of the fact that minimal polynomials of the sequences $\gamma_0(u)$ and $\gamma_0(\tilde{w})$ are coprime is analogous. $\square$

Proof of Lemma 10

Combining Lemma 6 with Lemma 7, we get

$$\mathrm{rank}_{\Gamma(K)}\gamma_0(u) + \mathrm{rank}_{\Gamma(K)}\gamma_0(v) = m\binom{n_0+p-1}{n_0} + m(n_0+n_1)+$$
$$+ \sum_{\substack{0\le l_0,\dots,l_{m-1}<p \\ l_0+\dots l_{m-1}=p}} \prod_{s=0}^{m-1} \binom{l_s+n_0-1}{l_s} = \sum_{\substack{0\le l_0,\dots,l_{m-1}\le p \\ l_0+\dots l_{m-1}=p}} \prod_{s=0}^{m-1} \binom{l_s+n_0-1}{l_s}+$$
$$+ m(n_0+n_1).$$

Using Proposition 12, we have

$$\sum_{\substack{0\le l_0,\dots,l_{m-1}\le p \\ l_0+\dots l_{m-1}=p}} \prod_{s=0}^{m-1} \binom{l_s+n_0-1}{l_s} = \binom{mn_0+p-1}{p}.$$

So, Lemma 10 is proved. This completes the proof of Theorem 3.

References

[1] Kurakin V. L., Kuzmin A. S., Mikhalev A. V., Nechaev A. A. Linear recurring sequences over rings and modules. // J. of Math. Sciences — 1995 — V.76. — No 6. — P. 2793-2915. (In English). Itogi Nauki i Tekhniki, Seriya Sovremennaya Matematika i Ee Prilozheniya. Tematicheskie Obzory. — 1994 — V.10, Algebra-2. (In Russian).

[2] Nechaev A.A. Kerdock Code in a Cyclic Form. // Discrete Math. and Appl — 1991 — V. 1. — No 4. — P. 365-384. (In English). Diskr. Mat. — 1989 — V. 1. — No 4. — P. 123-139. (In Russian)

[3] Goltvanitsa M.A., Nechaev A.A.,Zaitsev S.N. Skew linear recurring sequences of maximal period over Galois rings // Journ. Math. Sci — 2012 — V. 187 — No 2. — P. 115-128. (In English). Fundamentalnaya i Prikladnaya Matematika — 2011/12 — V. 17 — No 3. — P. 5-23 (In Russian).

[4] Kurakin V. L., Mikhalev A. V., Nechaev A. A., and Tsypyschev V. N. Linear and polylinear recurring sequences over abelian groups and modules. Journal of Mathematical Sciences — 2000 — V. 102 — No 6. — P. 4598-4626.

[5] Goltvanitsa M.A., Nechaev A. A., Zaitsev S. N. Skew LRS of maximal period over Galois rings // Mat Vop Kriptogr — 2013 — V.4 — No 2. — P. 59-72.

[6] Goltvanitsa M.A. Digit sequences of skew linear recurrences of maximal period over Galois rings. // Mat Vop Kriptogr — 2015 — V.6 — No 2. — P. 189-197.

[7] Glukhov M. M., Elizarov V. P., Nechaev A. A. Algebra. book II. — Moscow 2003. Gelios ARV. (In Russian)

[8] Kuzmin A.S., Nechaev A. A. Linear recurring sequences over Galois rings. // Russian Mathematical Surveys — 1993 — V. 48. — No 1. — P. 171-172. (In English). Uspehi Mat Nauk — 1993 — V. 48. — No 1. — P. 167-168. (In Russian)

[9] Kurakin V.L. The first coordinate sequence of a linear recurrence of maximum period over a Galois ring // Diskrete Math. and Appl. — 1994— V. 4 — No 2. — P. 129-141. (In English). Diskr. Mat. — 1994— V. 6 — No 2. — P. 88-100. (In Russian).

[10] Nechaev A.A. Finite Rings with Applications. Handbook of Algebra // Edited by M. Hazewinkel Elsevier B.V. — 2008 — V. 5 — P. 213-320.

[11] *Nechaev A.A.* Cycle types of linear substitutions over finite commutative rings. // Russian Academy of Sciences. Sbornik. Mathematics. — 1994

— V. 78 — No 2. — P. 283-311. (In English). Mat. Sb. — 1993 — V. 184 — No 3. — P. 21-56. (In Russian).

[12] Kurakin V.L. The first digit carry function in the Galois ring // Discrete Math. Appl. — 2012 — V. 22 — No 3. — P. 241-259. (In English). Diskr. Mat. — 2012 — V. 24 — No 2. — P. 21-36. (In Russian).

[13] Sachkov V.N. Introduction to combinatorial mathods in discrete mathematics. // Nauka. — 1982 — Moscow. (In Russian).

[14] Lidl R. and Niederreiter H. Finite Fields, in: Encyclopedia of Mathematics and its Applications — 1983 — V. 20 Cambridge University Press. (In English). Moscow. — 1988 — Mir (In Russian)

[15] Berlecamp E.. Algebraic Coding Theory. // US. — McGraw-Hill Inc. — 1968. (In English). Moscow. — 1971. — Mir. (In Russian).

Second Coordinate Sequence of the MP-LRS over Non-trivial Galois Ring of the Odd Characteristic

Vadim Tsypyshev

Abstract

We describe divisors of the minimal polynomial of the second p-adic coordinate sequence of the maximal period linear recurrent sequence MP-LRS over non-trivial Galois ring of odd characteristic in dependence of the initial vector of this LRS.

Also we describe polynomials divisible by that minimal polynomial in dependence of the initial vector of this LRS.

As a corollary we get non-trivial upper and lower estimations for the rank of the second coordinate sequence of such MP-LRS

We say that the Galois ring is *non-trivial*, if it differs from Galois field and from quotient ring too.

These results were worked out in 2004 with participation of V.L. Kurakin as a supervisor. The author is very grateful to V.L.Kurakin for his participation in this work.

1 Introduction

Let $R = GR(q^n, p^n)$ be a Galois ring [11, 12], $q = p^r$, p is a prime, u is a linear recurrent sequence of the full period over R with characteristic Galois polynomial $F(x)$ of degree m [7].

Let $S = GR(Q^n, p^n)$, $Q = q^m$, be a Galois extension of R, splitting ring of the polynomial $F(x)$, θ is a root of $F(x)$ in the ring S. Then [6] there exists a unique constant $\xi \in S$ with the property:

$$u(i) = \mathrm{Tr}_R^S(\xi\theta^i),\ i \in \mathbb{N}_0, \tag{1}$$

where $\mathrm{Tr}_R^S(x) = \sum\limits_{\sigma\in\mathrm{Aut}(S/R)} x^\sigma$ is a *trace* function from the ring S into ring R.

It is known that an arbitrary element $s \in S$ may be uniquely represented in the form

$$s = \sum_{i=0}^{n-1} \gamma_i(s)p^i, \ \gamma_i(s) \in \Gamma(S), \ i = \overline{0, n-1}, \tag{2}$$

where $\Gamma(S) = \{x \in S \mid x^Q = x\}$ is a *p-adic coordinate set of the ring S (Teichmueller's representatives system).*

The set $\Gamma(S)$ with operations $\oplus : x \oplus y = (x+y)^{Q^n}$ and $\otimes : \ x \otimes y = xy$ is a Galois field $GF(Q)$.

The field $\Gamma(S)$ contains as a sub field the set $\Gamma(R) = \{x \in R \mid x^q = x\}$ which is a p-adic coordinate set of the ring R.

Operations on elements of the $\Gamma(R)$ are defined in the same way. Because of that the set $\Gamma(R)$ is a field $GF(q)$.

It is known that [11, 12] the group $\mathrm{Aut}(S/R)$ is a cyclic and is generated by the *Frobenius automorphism* ρ which acts upon an element $s \in S$ of the form (2) according to the rule

$$\rho(s) = \sum_{i=0}^{n-1} \gamma_i(s)^q p^i. \tag{3}$$

Representation analogous to the (2) takes place for elements of the ring R.

The sequence $u(i)$, $i \in \mathbb{N}_0$, uniquely determines n *p-adic coordinate sequences* $u_l(i) = \gamma_l(u(i))$, $l = \overline{0, n-1}$, $i \in \mathbb{N}_0$, over the field $(\Gamma(R), \oplus, \cdot)$.

If we have a task to generate a pseudo random sequences relying on linear recurrence over Galois ring we may choose one or several elder coordinate sequences of this linear recurrence. In this way it is interesting to have estimations for the ranks of those coordinate sequences $\gamma_l(u)$, $l = \overline{0, n-1}$. In [10] lower and upper estimations for the ranks of coordinate sequences of linear recurrences of maximal period over primarily residue rings were obtained. Besides that, there were obtained minimal polynomials of coordinate sequences for some types of such linear recurrences.

Also in [10] minimal polynomials of sequences u_l, $l = \overline{0, 1}$ over nontrivial Galois ring were obtained.

In [3] the minimal polynomial and the rank of the first coordinate sequence of linear recurrence u over non-trivial Galois ring determined in arbitrary coordinate set were obtained.

Further in the article [9] were obtained exact values of ranks for second coordinate sequence of faithful linear recurrent sequence over binary residue ring with minimal Galois polynomial of degree not less then 5 in dependence on the initial vector of this LRS.

Below we will provide polynomials over Galois field $\Gamma(R)$ which respectively divides and are divisible by minimal polynomial of the second coordinate sequence of the linear recurrence u in p-adic coordinate set under condition of $p \geq 5$. These results provide a way to obtain upper and lower estimations for the rank of this linear recurrence. Previously these results in less faithful form were published in [15].

Furthermore, relying on these results in the next article we will obtain rank estimations for elders coordinate sequences of MP LRS over nontrivial Galois ring with specially selected numbers.

2 Main results

Let $M, w \in \mathbb{N}$. We will denote by $\mathcal{I}(M, w)$ the set of strings $\vec{\jmath} = (j_1, \ldots, j_M)$, $0 \leq j_l \leq p-1$, $l = \overline{1, M}$, with property : $\sum_{l=1}^{M} j_l = w$.

By $\left\{ {M \atop w} \right\}$ we will denote the cardinality of the set $\mathcal{I}(M, w)$. Let us note that $\left\{ {M \atop w} \right\}$ is equal to the quantity of allocations of w identical balls into M different boxes under condition that into each box there disposes not greater then $p-1$ balls. It is known that [8, C.215]:

$$\left\{ {M \atop w} \right\} = \sum_{s=0}^{\min\{w,(M-w)/p\}} (-1)^s \binom{w}{s} \binom{M+w-ps-1}{M-1}, \tag{4}$$

if $0 \leq w \leq M(p-1)$ and

$$\left\{ {M \atop w} \right\} = 0 \tag{5}$$

in other case.

Let m is a degree of any polynomial under investigation. Further we will denote by $N = N(m)$ the value $\left\{ {m \atop p} \right\}$. Besides that, we will concern that the strings $\vec{\jmath}^{(1)}, \ldots, \vec{\jmath}^{(N)}$ in the set $\mathcal{I}(m, p)$ are allocated in the lexicographical order.

Let $R = GR(q^n, p^n)$ be a Galois ring, $q = p^r$, $p \geq 5$, $r \geq 2$, $F(x)$ is a Galois polynomial of degree m over the ring R, the sequence $u \in L_R(F)$ is a non-zero by modulus pR and is represented by the trace-function

$$u(i) = \mathrm{Tr}_R^S(\xi\theta^i),$$

where $S = GR(q^{mn}, p^n)$ is a splitting ring of the polynomial $F(x)$, θ is a root of $F(x)$ in the S, the constant $\xi \in S$ is unequally determined. Let, further, $Q = q^m = p^t$, $t = rm$, $\theta_s = \gamma_s(\theta)$, $\xi_s = \gamma_s(\xi)$, $s = \overline{0, n-1}$.

The element ∇ is introduced in this way: according to Wilson theorem we have $(p-1)! \underset{p}{\equiv} -1$. Hence when $p \geq 3$ number $(p-1)!$ as an element of the ring $\mathbb{Z}_{p^n}$ has a p-adic representation of the form

$$(p-1)! \underset{p^2}{\equiv} -1 - p\nabla, \nabla \in \Gamma(\mathbb{Z}_{p^n}).$$

Let's denote:

$$G(x) = \prod_{\vec{j} \in \Xi} \left(x \ominus \theta_0^{\sum_{l=0}^{m-1} j_l p^{rm+rl-1}} \right),$$

$$\Xi = \left\{ \vec{j} \in \mathcal{I}(m,p) : \nabla \cdot \gamma_0 \left(\frac{1}{\prod_{l=0}^{m-1} j_l!} \right) \neq \ominus \gamma_1 \left(\frac{1}{\prod_{l=0}^{m-1} j_l!} \right) \right\},$$

$$W(x) = \prod_{s=0}^{p} \prod_{(\vec{\lambda}, \vec{\zeta}) \in \Omega_s} \left(x \ominus \theta_0^{\sum_{l=0}^{m-1} p^{t+rl-2}(\lambda_l + p\zeta_l)} \right),$$

where

$$\Omega_s = \Bigg\{ (\vec{\lambda}, \vec{\zeta}) \ : \ \vec{\lambda} \in \mathcal{I}(m, ps), \vec{\zeta} \in \mathcal{I}(m, p-s),$$

$$\sum_{\substack{\vec{\mu} \in \mathcal{I}(N,p): \\ \sum_{i=1}^{N} \mu_i \cdot j_l^{(i)} = \lambda_l + p\zeta_l, l = \overline{0,m-1}}} \!\!\!\!\!\! \bigoplus \gamma_0 \left(\frac{1}{\prod_{i=1}^{N} \mu_i! (\prod_{l=0}^{m-1} j_l^{(i)}!)^{\mu_i}} \right) \neq 0 \Bigg\}, s = \overline{0,p},$$

$$H_s(x) = \prod_{\substack{\vec{\lambda} \in \mathcal{I}(m, ps), \\ \vec{\zeta} \in \mathcal{I}(m, p-s)}} \left(x \ominus \theta_0^{\sum_{l=0}^{m-1} p^{rm+rl-2}(\lambda_l + p\zeta_l)} \right), s = \overline{0, p-1},$$

$$H(x) = H_1(x), Z(x) = H_0(x),$$

$$D_1(x) = \text{ GCD } (G(x), W(x)),\ D_2(x) = \text{GCD } (W(x), H(x)).$$

Pay attention to the fact that

$$G(x) \mid Z(x).$$

Besides that, let's denote by

$$\tilde{F}(x) = \gamma_0(F(x))$$

the polynomial obtained from $F(x)$ by picking out zeroth p-adic digits of every coefficient of polynomial. It is obviously that $\tilde{F}(x) \in \bar{F}(x)$, where $\bar{F}(x)$ is a residue class of the $F(x)$ in the ring $R[x]/pR[x]$.

Theorem 2.1 *Let $R = GR(q^n, p^n)$ be a Galois ring, $q = p^r$, $p \geq 5$, $r \geq 2$, $F(x)$ is a Galois polynomial of degree m over the ring R, the sequence $u \in L_R(F)$ is a non-zero by modulus pR and is represented by the trace-function*

$$u(i) = \text{Tr}_R^S(\xi\theta^i),$$

where $S = GR(q^{mn}, p^n)$ is a splitting ring of the polynomial $F(x)$, θ is a root of $F(x)$ in the S, the constant $\xi \in S$ is unequally determined. Let, further, $Q = q^m = p^t$, $t = rm$, $\theta_s = \gamma_s(\theta)$, $\xi_s = \gamma_s(\xi)$, $s = \overline{0, n-1}$.

If $F(x)$ is a MP-polynomial, in other words [14], θ_0 is a primitive element of the field $\Gamma(S) = GF(Q)$, and $\theta_1 \neq 0$, then for a minimal polynomial of the second coordinate sequence u_2 of the LRS u in the p-adic coordinate set these dependencies hold:

$$\tilde{F}(x)^{p+1} \cdot \frac{G(x)}{D_1(X)} \cdot \frac{W(x)}{D_1(x) \cdot D_2(x)} \cdot H(x)^p \,\Bigg|\, m_2(x),$$

$$m_2(x) \,\Bigg|\, \tilde{F}(x)^{p+1} \cdot Z(x)^\epsilon \cdot H(x)^p \cdot \text{ LCM } \left(\frac{W(x)}{D_2(x)}, \prod_{s=2}^{p-1} H_s(x)^{\beta_s}\right).$$

Under the same conditions these inequalities hold:

$$\epsilon \le p, \quad \beta_s \le p-1, s = \overline{2, p-1}.$$

Besides that in described cases there are known equal values of the parameter ϵ:

(a) If $\xi_1 \neq 0$, then under additional conditions

$$\forall \vec{\zeta} \in \mathcal{I}(m,p) \quad \sum\nolimits^{\oplus}_{\kappa=\overline{0,m-1}\ :\ \zeta_\kappa>0} (\xi_0^{-1}\xi_1)^{p^{t+r\kappa-1}} \neq 0$$

and

$$\forall \vec{\zeta} \in \mathcal{I}(m,p) \quad \sum\nolimits^{\oplus}_{l=\overline{0,m-1}\ :\ \zeta_l>0} \gamma_0 \left(\frac{\zeta_l}{\prod_{\kappa=0}^{m-1}\zeta_\kappa!}\right) (\theta_0^{-1}\theta_1)^{\sum_{\kappa=0}^{m-1}\zeta_\kappa p^{t+r\kappa-1} - p^{t+rl-1}} \neq 0$$

the equality holds: $\epsilon = p$, and $Z(x)^\epsilon \mid m_2(x)$.

(b) If $\xi_1 = 0$, then $\epsilon = 2$.

Under the same condition

$$Z(x)^\epsilon \mid m_2(x).$$

These results were obtained in 2004. Since that time neither equal value of the parameter $\deg G(x)$ nor its upper bound are obtained.

It is clear that this parameter is not greater then

$$\left\{ \begin{matrix} m \\ p \end{matrix} \right\},$$

and also depends on p.

Something the same may be declared about parameters $\deg W(x)$ and $\deg D_i(x), i = \overline{1,2}$.

The same reasoning as in proof of the Theorem 2.1 is valid in the case of $R = \mathbb{Z}_{p^n}$.

However under condition $r = 1$ all previous symbolization becomes invalid and because of that all previous rank estimations become invalid too.

Theorem 2.2 *Under conditions of the Theorem 2.1 these inequalities hold:*

$$\begin{aligned} m(p+1)+p\left\{{m \atop p}\right\}\left\{{m \atop p-1}\right\} \leq \operatorname{rank} u_2 \leq \; & m(p+1)+p\left\{{m \atop p}\right\}\left\{{m \atop p-1}\right\}+ \\ & +(p-1)\sum_{s=2}^{p-1}\left\{{m \atop ps}\right\}\left\{{m \atop p-s}\right\}+ \\ & +p\left\{{m \atop p}\right\}+\left\{{m \atop p^2}\right\}. \end{aligned}$$

References

[1] Elwyn R. Berlekamp Algebraic Coding Theory (English) // Zbl 0988.94521 New York, NY: McGraw-Hill Book. xiv, 466 p. (1968).

[2] Kurakin, V.L. Representations over $\mathbb{Z}_{p^n}$ of a linear recurring sequence of maximal period over $GF(p)$. (English; Russian original) Discrete Math. Appl. 3, No.3, 275-296 (1993); translation from Diskretn. Mat. 4, No.4, 96-116 (1992). Zbl 0811.11077

[3] Kurakin, V.L. The first coordinate sequence of a linear recurrence of maximal period over a Galois ring. (English; Russian original) Discrete Math. Appl. 4, No.2, 129-141 (1994); translation from Diskretn. Mat. 6, No.2, 88-100 (1994). Zbl 0824.11072

[4] Kurakin, V.L. The first digit carry function in the Galois ring. (English; Russian original) // Discrete Math. Appl. 22, No. 3, 241-259 (2012); translation from Diskretn. Mat. 24, No. 2, 21-36 (2012). Zbl 1281.11020

[5] Lidl, Rudolf; Niederreiter, Harald Finite fields. // 2nd edition— 1996 (English) — Encyclopedia of Mathematics and Its Applications— 20— Cambridge: Cambridge University Press (ISBN 978-0-521-06567-2/pbk)— 755 p. — Zbl 1139.11053

[6] Nechaev, A.A. Kerdock code in a cyclic form. (English; Russian original) Discrete Math. Appl. 1, No.4, 365-384 (1991); translation from Diskretn. Mat. 1, No.4, 123-139 (1989). Zbl 0734.94023

[7] Nechaev, A.A. Linear recurrence sequences over commutative rings. (English; Russian original) Discrete Math. Appl. 2, No.6, 659-683 (1992); translation from Diskretn. Mat. 3, No.4, 105-127 (1991).Zbl 0787.13007

[8] Sachkov, V.N. Introduction to combinatorial methods of discrete mathematics // Moscow, *Nauka*, 1982, 384P.

[9] Helleseth T., Martinsen M. Binary sequences of period $2^m - 1$ with large linear complexity // Informatrion and Computation, **151**, 73–91, (1999)

[10] Kuzmin A.S., Nechaev A.A. Linear recurrent sequences over Galois rings // II Int.Conf.Dedic.Mem. A.L.Shirshov—Barnaul—Aug.20-25 1991 (Contemporary Math.—v.184—1995—p.237-254)

[11] McDonald C. Finite rings with identity // New York: Marcel Dekker—1974—495p.

[12] Radghavendran R. A class of finite rings // Compositio Math.—1970— v.22—N1—p.49-57

[13] V. N. Tsypyshev Matrix linear congruent generator over a Galois ring of odd characteristic Proceedings of the 5th Int. Conf. "Algebra and number theory: modern problems and applications", Tula State Pedagogic Univ., Tula, 2003, p 233-237 (In Russian) MathSciNet: 2035586

[14] Tsypyschev, V.N. Full periodicity of Galois polynomials over nontrivial Galois rings of odd characteristic. (English. Russian original) Zbl 1195.11160 // J. Math. Sci., New York 131, No. 6, 6120-6132 (2005); translation from Sovrem. Mat. Prilozh. 2004, No. 14, 108-120 (2004)

[15] Tsypyschev, V.N. Rank estimations of the second coordinate sequance of MP-LRS over nontrivial Galois ring of odd characteristic (in Russian) // II Int. Sci. Conference on Problems of Security and Counter-Terrorism Activity — Moscow, MSU, October 25-26, 2006

— Proceedings published by Moscow Independent Center for Mathematical Education—2007—pp287–289

[16] N.Zierler, W.Mills Products of linear recurring sequences // J. of Algebra—27(1973)—pp.147-157

On Binary Digit-position Sequences over Galois Rings, Admitting Twofold Reduction of a Period

Sergey Kuzmin

Abstract

A class of binary digit-position sequences, obtained from a linear recurring sequence of maximal period (LRS MP) over Galois rings of odd characteristics, admitting an effect of twofold reduction of a period, has been found. A condition was found, when sequences of some fixed LRS MP over Galois rings with such property, are exhausted only by that class.

Keywords: linear recurring sequences of maximal period, binary digit-position sequences, Galois rings, Galois fields, period of sequence.

1 Introduction

A special interest in recent years can be observed in studying p-adic digit-position sequences over residue ring modulo p^n, where p is a prime number. This is due to the fact that these sequences possesses high linear complexity, hence they can be used in random-number generators. A list of papers on this topic can be seen in [1].

A lot of attention is paid to reconstruction of LRS over prime residue rings from its images, especially when LRS MP over residue ring is mapped into the highest order p-adic digit position sequence [2].

Less attention was paid to r-ary digit position sequences over prime fields and residue rings where $r \neq p$. Such digit-position sequences were studied by Kuzmin A.S. in paper [3]. The author has found all binary digit position-sequences over finite prime fields, that admit the effect of reduction of period.

This paper, to some extent, generalizes paper [3] in conditions $n \geq 1$ for Galois rings, and for not simple finite fields.

Let $R = GR(p^{nm}, p^n)$, be a Galois ring with the characteristic polynomial $F(x)$, $\deg F(x) = m$ (see, for example, [3]), notably $R = \mathbb{Z}_{p^n}[x]/(F(x))$, $u = (u(i))_{i=0}^{\infty}$ is LRS MP over this ring, with characteristic polynomial $f(x)$. We consider that f(x) is basic monic irreducible over the ring R and $\deg f(x) = l$. Let $S = GR(p^{nml}, p^n)$ be a Galois extension of the ring R, and let $Aut(S/R)$ be a group of all automorphisms of the ring S, leaving the elements of R unchanged. Let us determine a trace function $Tr_R^S : S \to R$ by the following equation $Tr_R^S(x) = \sum_{\tau \in Aut(S \backslash R)} \tau(x)$. Let α be a root of a polynomial $f(x)$, in the ring S. The existence of a single element $b \in S$ such, that $u(i) = Tr_R^S(b\alpha^i)$, $i \geq 0$, follows from [4, theorem 8].

Every element $c = [\sum_{j=0}^{m-1} c_j x^j]_{/F(x)} \in GR(p^{nm}, p^n)$ can be uniquely represented by its vector of coefficients $(c_0, c_1, ..., c_{m-1})$, where $c_i \in \mathbb{Z}_{p^n}$, $i \in \overline{0, m-1}$, by-turn coefficient c_i is uniquely represented as follows $c_i = \sum_{s=0}^{d} c_{i,s} 2^s$, $c_{i,s} \in \{0, 1\}$, $d = [\log_2 p^n] + 1$. Hence, every member $u(i)$ of some LRS MP u over Galois ring can be represented in the form

$$u(i) = (\sum_{s=0}^{d} u_{0,s}(i) 2^s, \sum_{s=0}^{d} u_{1,s}(i) 2^s, ..., \sum_{s=0}^{d} u_{m-1,s}(i) 2^s), \tag{1}$$

where $d = [\log_2 p^n] + 1$.

A sequence $u_{\bullet,s}$ with elements $u(i)_{\bullet,s} = (u_{0,s}(i), u_{1,s}(i), ..., u_{m-1,s}(i))$ $i \geq 0$, will be the s-th digit-position sequence of LRS MP u over Galois ring. It generalizes digit-position sequences from [3].

In this paper we study periods of binary digit-position sequences, obtained from the LRS MP over Galois rings. It is known, that the period $T(u)$ of the LRS MP u over ring S equals to $p^{n-1}(p^{ml} - 1)$ [1, p.178].

2 Main result

In the proof of the main results of this article, the the following proposition is used.

Proposition 1. [2, p. 43]. *Let $S = GR(p^{nml}, p^n)$ be an extension of degree l of a ring R, $u(i) = Tr_R^S(b\alpha^i)$, $i \geq 0$ be an LRS MP over R. Then the following equation holds*

$$u(i) + u(i + \frac{T(u)}{2}) = Tr_R^S(b\alpha^i) + Tr_R^S(b\alpha^{i+\frac{T(u)}{2}}) = 0.$$

Let us state and prove some accessory statements, before proving our main result. Let u_r be an r-th digit-position sequence of LRS MP u over $\mathbb{Z}_{p^n}$.

Statement 1. *For each prime $p \geq 3$, in primary rings like $\mathbb{Z}_{p^{2k}}$, $k \in \mathbb{N}$, for the period of a binary digit-position sequence u_1, obtained from LRS MP u according to the rule* (1) *for $m = 1$, it holds that $T(u_1)|\frac{T(u)}{2}$.*

Statement 2. *For each prime $p \geq 3$ such that $p - 1 \equiv 0(\bmod 4)$, in primary rings like $\mathbb{Z}_{p^{2k+1}}$, $k \in \mathbb{N}_0$, for the period of a binary digit-position sequence u_1, obtained from LRS MP u according to the rule* (1) *for $m = 1$, it holds that $T(u_1)|\frac{T(u)}{2}$.*

Statement 3. *For each prime $p \geq 3$ such that*

$$p = a(s)2^{s+1} + 2^s - 1,$$

$a(s) \geq 0$, $s \geq 2$, in primary rings like $\mathbb{Z}_{p^{2k+1}}, k \in \mathbb{N}_0$ for the period of a binary digit-position sequence u_s, obtained from LRS MP u according to the rule (1) *for $m = 1$, it holds that $T(u_s)|\frac{T(u)}{2}$.*

Remark 1. *Note that if $p = 2^s - 1$ we consider $p = a(s)2^{s+1} + 0 * 2^s + 2^s - 1$, where $a(s) = 0$.*

Remark 2. *Expression $u_r(i) = u_r(i + \frac{T(u)}{2}) \oplus 1$, $r < s$, is a singular analog of expressions from paper* [2, p. 43].

Statement 4. *Let binary digit-position sequences u_r, $r \in \overline{0, d}$ are formed by the rule* (1)*for $m = 1$ from LRS MP u. Let there exists at least one of each elements from $\mathbb{Z}_{p^n} \backslash \{0\}$ on the period of u. If r is not equal to the indexes of digit-position sequences mentioned in statements* 1-3, *then $\frac{T(u)}{2}$ will not be divided by $T(u_r)$.*

Now we can prove the following theorem.

Theorem 1. *Let $R = GR(p^{nm}, p^n)$, $m \in \mathbb{N}$, be a Galois ring, $p = a(s)2^{s+1} + 2^s - 1$, $a(s) \geq 0$, let u be an LRS MP over this ring with characteristic polynomial $f(x)$,* $\deg f(x) = l$ *and*

$$z = \begin{cases} 1, & \textit{for } n = 2k \textit{ or } n = 2k-1 \textit{ and } p \equiv 1(\mathrm{mod} 4), \\ s, & \textit{for } n = 2k-1 \textit{ and } p \equiv 3(\mathrm{mod} 4). \end{cases},$$

for some fixed $k \in \mathbb{N}$. Then for the period $T(u_{\bullet,z})$ of a binary digit-position sequence $u_{\bullet,z}$, obtained from LRS MP u according to the rule (1)*, the following expression holds*

$$T(u_{\bullet,z}) | \frac{T(u)}{2}. \qquad (2)$$

In case when at least one component of m-dimensional vector of coefficients $(c_0, c_1, ..., c_{m-1})$, where $c_i \in \mathbb{Z}_{p^n}$, representing elements $c \in R$ in the period of sequence u, possesses all the values from $\mathbb{Z}_{p^n} \backslash \{0\}$, then the sequence $u_{\bullet,z}$ is unique with property (2)*.*

In particular case, considering results of paper [5], we obtain the folowing corollary.

Corollary 1. *Let u be a LRS MP over Galois ring $R = GR(p^{nm}, p^n)$ with the generator polynomial $f(x)$, $\deg f(x) = l$. If there exists at least one invertible element among $u(0), u(1), ..., u(l-1)$ and the inequality $l \geq \frac{2(nm+n-1)}{m}$ holds, then there exists a unique binary digit-position sequence, admitting effect of reduction of period in two times, obtained from LRS MP u.*

Applying the theory of finite fields [6], we obtain the corollary.

Corollary 2. *Let $F = GF(p^m)$ be an extension of the finite prime field $GF(p)$ of degree m, $p = a(s)2^{s+1} + 2^s - 1$, $a(s) \geq 0$, u be a LRS MP over this field. Then for the period $T(u_{\bullet,s})$ of a binary digit-position sequence $u_{\bullet,s}$, obtained from the sequence u according to rule* (1)*, when $n = 1$, the expression $T(u_{\bullet,s}) | \frac{T(u)}{2}$, holds and sequence $u_{\bullet,s}$ is a unique with that property.*

Remark 3. *Notice that if p is a Mersenne number and $n = 1$, the expression $T(u_{\bullet,s}) | \frac{T(u)}{2}$ doesn't hold.*

3 Conclusion

A class of binary digit-position sequences, obtained from the LRS MP over Galois rings of odd characteristics, admitting an effect of reduction of period in 2 times, has been found. A condition was found, when sequences of some fixed LRS MP over Galois fields with such property, are exhausted only by that class. Our result generalizes some results of paper [3] for $n \geq 1$, and non-prime finite fields.

References

[1] Satchkov V.N., Gorchinskiy Y.N.,Zubkov A.M.,Yablonskiy S.V. Trudy po diskretnoy matematike. vol.1 Moscow:1997-VIII, 280.p

[2] Kuzmin A.S., Marshalko G.B., Nechaev A.A. Reconstruction of linear recurrent sequence over prime residue ring from its image. Mathematical Aspects of Cryptography, 2010, vol. 1, no. 2, pp. 31-56.

[3] Kuzmin A.S. About periods of digit positions in r-ary notation of elements of linear recurring sequences over finite prime fields. Bezopasnost Inforpatsionnykh Tekhnology, no. 4 1995, pp. 71-75

[4] Nechaev A.A. Kerdock code in a cyclic form, Discrete Math. Appl., vol. 1 (1991) no. 4. , pp. 123-139

[5] A.A.Kamlovskiy O.V.,Kuzmin A.S. Bounds for the number of occurences of elements in a linear recurring sequence over Galois ring, Fudamentalnaya i prikladnaya matematika, vol. 6 (2000), no. 4, pp. 1083-1094

[6] Gluhov M.M., Elizarov V.P., Nechaev A.A. Algebra. Moscow: Gelios ARV, 2003. - vol. 2

On the Concept of Quantum Hashing

Farid Ablayev Marat Ablayev

Invited talk

Abstract

We present the notion of quantum hashing which a natural generalization of classical hashing. We present the concept of a quantum hash generator and offer a design, which allows one to build a large number of different quantum hash functions.

The construction is based on composition of a classical ϵ-universal hash family and a given family of functions – quantum hash generators.

Keywords: hashing, quantum hashing, quantum hash function.

1 Introduction

Quantum cryptography describes the use of quantum mechanical effects (in particular quantum communication and quantum computation) (a) to perform cryptographic tasks or (b) to break cryptographic systems or to perform cryptographic tasks.

Quantum key distribution is the well-known example for the first direction of quantum cryptography. The answer of the cryptography community for the second direction is "Post-quantum cryptography", which refers to research on problems (usually public-key cryptosystems) that are no more efficiently breakable using potential quantum computers. Currently post-quantum cryptography includes several approaches, in particular, hash-based signature schemes such as Lamport signatures and Merkle signature schemes.

Quantum versions of signature schemes were considered by several authors. Gottesman and Chuang proposed a quantum digital system [7], based on quantum mechanics. Their results are based on quantum a fingerprinting technique [5] and add "quantum direction" for post-quantum cryptography. Gavinsky and Ito [6] generalized quantum a fingerprinting technique of [5] and viewed quantum fingerprints as cryptographic primitives.

In [1, 2] we explicitly defined a notion of quantum hashing as a generalization of classical hashing and presented examples of quantum hash functions. It is easy to see that quantum fingerprinting and its generalizations we mentioned above, are quantum hash functions. Informally speaking, we defined a quantum hash function ψ to be a function that maps words of length k to a quantum s-qubit states ($\psi : \Sigma^k \to (\mathcal{H}^2)^{\otimes s}$) and has the following properties:

- Function ψ must be designed to have maximum output difference between adjacent inputs. In quantum case this means that for different words w, w' states $\psi(w)$, $\psi(w')$ must be "almost orthogonal" (δ-orthogonal).

- Function ψ must be physically one-way. In quantum case this means that $k \gg s$.

Our contribution.

In this paper, we define the concept of a quantum hash generator (Definition 3). Informally speaking, we call a family G of discrete functions a quantum hash generator, if G allow to construct a quantum function ψ_G which is quantum hash function. We offer a design, which allows one to build quantum hash functions based on specific families G (Theorem 4).

Theorem 4 give possibilities to build a large amount of different quantum hash functions.

Organization of the paper. The results of the paper mainly based on the arXiv paper [3]. We include here only the original proofs and omit proofs that can be find in [3].

2 Definitions and Notations

We begin by recalling some definitions of classical hash families from [11]. Given a domain $\mathbb{X}$, $|\mathbb{X}| = K$, and a range $\mathbb{Y}$, $|\mathbb{Y}| = M$, (typically with

$K \geq M$), a hash function f is a map

$$f : \mathbb{X} \to \mathbb{Y},$$

that hash *long* inputs to *short* outputs.

We let q to be a prime power and $\mathbb{F}_q$ be a finite field of order q. Let Σ^k be a set of words of length k over a finite alphabet Σ. In the paper we let $\mathbb{X} = \Sigma^k$, or $\mathbb{X} = \mathbb{F}_q$, or $\mathbb{X} = (\mathbb{F}_q)^k$, and $\mathbb{Y} = \mathbb{F}_q$. A hash family is a set $F = \{f_1, \ldots, f_N\}$ of hash functions $f_i : \mathbb{X} \to \mathbb{Y}$.

We recall known definition.

Definition 1 (ϵ universal hash family.) *A family F is called an ϵ-universal (ϵ-$U(N; K, M)$) hash family if for any two distinct elements $w, w' \in \mathbb{X}$, when function the f is chosen uniformly at random from F, then the probability $Pr_{f \in F}[f(w) = f(w')]$ that elements w, w' collide under f is at most ϵ.*

The parameter ϵ is often referred to as the collision probability of the hash family F.

Classical-quantum function. The notion of a quantum function was considered in [9]. In this paper we use the following variant of a quantum function. First recall that mathematically a qubit $|\psi\rangle$ is described as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers, satisfying $|\alpha|^2 + |\beta|^2 = 1$. So, a qubit may be presented as a unit vector in the two-dimensional Hilbert complex space $\mathcal{H}^2$. Let $s \geq 1$. Let $(\mathcal{H}^2)^{\otimes s}$ be the 2^s-dimensional Hilbert space, describing the states of s qubits, i.e. $(\mathcal{H}^2)^{\otimes s}$ is made up of s copies of a single qubit space $\mathcal{H}^2$

$$(\mathcal{H}^2)^{\otimes s} = \mathcal{H}^2 \otimes \ldots \otimes \mathcal{H}^2 = \mathcal{H}^{2^s}.$$

For $K = |\mathbb{X}|$ and integer $s \geq 1$ we define a $(K; s)$ classical-quantum function to be a map of the elements $w \in \mathbb{X}$ to quantum states $|\psi(w)\rangle \in (\mathcal{H}^2)^{\otimes s}$

$$\psi : \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}. \tag{1}$$

We will also use the notation $\psi : w \mapsto |\psi(w)\rangle$ for ψ.

3 Quantum hashing

What we need to define for quantum hashing and what is implicitly assumed in various papers (see for example [1] for more information) is a collision resistance property. However, there is still no such notion as *quantum collision*. The reason why we need to define it is the observation that in quantum hashing there might be no collisions in the classical sense: as we will see later a quantum hash function we define is one-to-one mapping of classical words to quantum states. But the procedure of comparing those (different !) quantum states implies measurement, which can lead to collision-type errors.

So, a *quantum collision* is a situation when a procedure that tests the equality of quantum hashes and outputs "the hashes are the same", while hashes are different. This procedure can be a well-known SWAP-test (see for example [1] for more information and citations) or REVERSE-test we consider below. Anyway, it deals with the notion of distinguishability of quantum states. Since non-orthogonal quantum states cannot be perfectly distinguished, we require them to be "nearly orthogonal".

- For $\delta \in (0, 1/2)$ we call a function

  $$\psi : \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}$$

  δ-resistant, if for any pair w, w' of different elements,

  $$|\langle \psi(w) \,|\, \psi(w') \rangle| \leq \delta.$$

Theorem 1 *Let $\psi : \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}$ be a δ-resistant function. Then*

$$s \geq \log\log |\mathbb{X}| - \log\log\left(1 + \sqrt{2/(1-\delta)}\right) - 1.$$

Proof. See [3] for the proof. □

The notion of δ-resistance naturally leads to the following notion of quantum hash function.

Definition 2 (Quantum hash function) *Let K, s be positive integers and $K = |\mathbb{X}|$. We call a map*

$$\psi : \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}$$

an δ-resistant $(K; s)$ quantum hash function if ψ is a δ-resistant function.

We use the notation δ-R $(K; s)$ as an abbreviation for δ-resistant $(K; s)$ quantum hash functions.

Quantum hashing property. One of the extra justification of the quantum hash function definition is as follows. Let $\mathbb{X} = \Sigma^k$. Let $\psi : w \mapsto |\psi(w)\rangle$ be δ-resistant.

Let us consider the procedure that we call a *REVERSE-test* [2] and which proposed to check if a quantum state $|\psi(w)\rangle$ is a hash of a classical string v.

Essentially the test applies the procedure that inverts the creation of a quantum hash, i.e. it "uncomputes" the hash to the initial state (usually the all-zero state).

Formally, let the procedure of quantum hashing the string w be given by unitary transformation $U(w)$, applied to initial state $|0\rangle$, i.e. $|\psi(w)\rangle = U(w)|0\rangle$. Then the REVERSE-test, given v and $|\psi(w)\rangle$, applies $U^{-1}(v)$ to the state $|\psi(w)\rangle$ and measures the resulting state. It outputs $v = w$ iff the measurement outcome is $|0\rangle$. Denote by $Pr_{reverse}[v = w]$ the probability that the *REVERSE-test* having quantum state $|\psi(w)\rangle$ and a word v outputs the result that $v = w$.

The following property presented implicitly in [4].

Property 1 *Let ψ be δ-resistant. Then for any two different words $w, v \in \Sigma^k$ it is true that*

$$Pr_{reverse}[v = w] < \delta^2.$$

Proof. $v \neq w$. By δ-resistance property $\langle 0\,|\,U^{-1}(v)\psi(w)\rangle < \delta$, which bounds the probability $Pr_{reverse}[v = w] = |\langle 0\,|\,U^{-1}(v)\psi(w)\rangle|^2$ by δ^2. □

Following Definition 1 one can call δ^2 a collision probability of the δ-reversible quantum hash function ψ.

3.1 Quantum fingerprinting function is a quantum hash function.

One of the first explicit quantum hash functions was defined in [5]. Originally the authors invented a construction called "quantum fingerprinting" for testing the equality of two words for a quantum communication model. The cryptography aspects of quantum fingerprinting are presented in [6]. The quantum fingerprinting technique is based on binary error-correcting codes. Later this construction was adopted for cryptographic purposes. Here we present the quantum fingerprinting construction from the quantum hashing point of view.

An (n, k, d) *error-correcting code* is a map $C : \Sigma^k \to \Sigma^n$ such that, for any two distinct words $w, w' \in \Sigma^k$, the Hamming distance $d(C(w), C(w'))$ between code words $C(w)$ and $C(w')$ is at least d. The code is binary if $\Sigma = \{0, 1\}$.

The construction of a quantum hash function based on quantum fingerprinting is as follows. Let k be a positive integer and $n > k$. Let $E : \{0,1\}^k \to \{0,1\}^n$ be an (n, k, d) binary error-correcting code. Let $s = \log n$. Define the classical-quantum function $\psi : \{0,1\}^k \to (\mathcal{H}^2)^{\otimes s}$, determined by a word w as

$$\psi(w) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} (-1)^{E_i(w)} |i\rangle. \tag{2}$$

The next property is the reformulation of the property of "quantum fingerprinting" [5] in terms of quantum hashing.

Property 2 *For an (n, k, d) binary* error-correcting code *E, for $s = \log n$, for $\delta = (1 - d/n)$ the function ψ (2) is an δ-R $(2^k; s)$ quantum hash function.*

Proof. For two different words w, w', for $i \in \{1, \ldots, n\}$, $a_i(w, w') = E_i(w) + E_i(w')$ (mod 2) we have

$$|\langle \psi(w) \,|\, \psi(w') \rangle| = \frac{1}{n} \sum_{i=1}^{n} (-1)^{a_i(w,w')} = \frac{n - d(E(w), E(w'))}{n} \leq 1 - d/n = \delta.$$

□

Observe, that the above construction of a quantum hash function needs $s = \log n$ qubits for the $\delta = 1 - d/n$. This number of qubits is good enough in the sense of the lower bound of Theorem 1 which gives the following lower bound for s when $n = ck$.

$$s \geq \log(n/c) - \log\log\left(1 + \sqrt{2n/d}\right) - 1.$$

4 Quantum Hash Generator

In this section we define notion of quantum hash function which allow to construct quantum hash functions.

Definition 3 (Quantum hash generator) *Let $K = |\mathbb{X}|$ and let $G = \{g_1, \ldots, g_D\}$ be a family of functions $g_j : \mathbb{X} \to \mathbb{F}_q$. Let $\ell \geq 1$ be an integer. For $g \in G$ let ψ_g be a classical-quantum function $\psi_g : \mathbb{X} \to (\mathcal{H}^2)^{\otimes \ell}$ determined by the rule*

$$\psi_g : w \mapsto |\psi_g(w)\rangle = \sum_{i=1}^{2^\ell} \alpha_i(g(w))|i\rangle, \tag{3}$$

where the amplitudes $\alpha_i(g(w))$, $i \in \{1, \ldots, 2^\ell\}$, of the state $|\psi_g(w)\rangle$ are determined by $g(w)$.

Let $d = \log D$, $s = d + \ell$. We define a classical-quantum function $\psi_G : \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}$ by the rule

$$\psi_G : w \mapsto |\psi_G(w)\rangle = \frac{1}{\sqrt{D}} \sum_{j=1}^{D} |j\rangle \big|\psi_{g_j}(w)\big\rangle. \tag{4}$$

We say that the family G generates the δ-R $(K; s)$ quantum hash function ψ_G. We call G a δ-R $(K; s)$ quantum hash generator, if ψ_G is a δ-R $(K; s)$ quantum hash function. We call G a quantum hash generator if ψ_G generates a δ-R $(K; s)$ quantum hash function for some δ, K, and s.

Below we present two examples of quantum hash generators – the binary and non binary constructions.

4.1 Binary Quantum Hash Generator.

First we present the construction of quantum fingerprinting function from the section 3.1 in terms of Definition 3.

Theorem 2 *For an (n, k, d) binary error-correcting code $E : \{0,1\}^k \to \{0,1\}^n$ define a family of functions $F_E = \{E_1, \dots, E_n\}$, where $E_i : \{0,1\}^k \to \mathbb{F}_2$ is defined by the rule: $E_j(w)$ is the j-th bit of the code word $E(w)$. The family F_E is an δ-R $(2^k; s)$ quantum hash generator for $\delta = 1 - d/n$ and $s = \log n$.*

Proof. For $E_j \in F_E$ we define $\psi_{E_j} : \{0,1\}^k \to \mathcal{H}^2$ as $\left|\psi_{E_j}(w)\right\rangle = (-1)^{E_j(w)}|1\rangle$ and let

$$|\psi_{F_E}(w)\rangle = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} |j\rangle \left|\psi_{E_j}(w)\right\rangle = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} (-1)^{E_j(w)} |j\rangle|1\rangle.$$

We can omit the $|1\rangle$ from the last equation and finally we get

$$|\psi_{F_E}(w)\rangle = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} (-1)^{E_j(w)} |j\rangle.$$

The last together with the equality (2) and Property 2 proves the statement. $\square$

4.2 Non-binary Quantum Hash Generator.

We start by recalling some definitions, notations, and facts from [10]. For a field $\mathbb{F}_q$, the discrete Fourier transform of a set $B \subseteq \mathbb{F}_q$ is the function

$$f_B(a) = \sum_{b \in B} e^{\frac{2\pi i}{q} ab}$$

defined for every $a \in \mathbb{F}_q$. let $\lambda(B) = \max_{a \neq 0} |f_B(a)|/|B|$.

For $\delta > 0$ we define $B \subseteq \mathbb{F}_q$ to be δ-good if $\lambda(B) \leq \delta$. By $B_{\delta,q}$ we denote δ-good subset of $\mathbb{F}_q$.

For a field $\mathbb{F}_q$, let $B \subseteq \mathbb{F}_q$. For every $b \in B$ and $w \in \mathbb{F}_q$, define a function $h_b : \mathbb{F}_q \to \mathbb{F}_q$ and a family H_B by the rule

$$h_b(w) = bw \pmod q, \qquad H_B = \{h_b : b \in B\}.$$

We denote a family H_B of functions by $H_{\delta,q}$ and call $H_{\delta,q}$ δ-good if B is δ-good ($B = B_{\delta,q}$).

The following facts presented in [10]

- Let $\delta = \delta(q)$ be any function tending to zero as q grows to infinity. Then there exits δ-good set $B_{\delta,q}$ with $|B_{\delta,q}| = (\log q/\delta(q))^{O(1)}$.
- Several optimal (in the sense of the above lower bound) explicit constructions of δ-good sets $B_{\delta,q}$ presented by different authors which for

$$\delta(q) = \frac{1}{(\log q)^{O(1)}} \quad \text{achive} \quad |B_{\delta,q}| = (\log q)^{O(1)}.$$

Theorem 3 *Let $\delta > 0$ and q be a prime power. Let $H_{\delta,q}$ be δ-good. Then for $s = \log |H_{\delta,q}|$ a family $H_{\delta,q}$ is an δ-R $(q; s)$ quantum hash generator.*

Proof. in the appendix □

5 Quantum Hashing via Classical ϵ-Universal Hashing Constructions

In this section we present a construction of a quantum hash generator based on the composition of an ϵ-universal hash family with a given quantum hash generator. We begin with the definitions and notation that we use in the rest of the paper.

Let $K = |\mathbb{X}|$, $M = |\mathbb{Y}|$. Let $F = \{f_1, \ldots, f_N\}$ be a family of functions, where

$$f_i : \mathbb{X} \to \mathbb{Y}.$$

Let q be a prime power and $\mathbb{F}_q$ be a field. Let $H = \{h_1, \ldots, h_T\}$ be a family of functions, where

$$h_j : \mathbb{Y} \to \mathbb{F}_q.$$

For $f \in F$ and $h \in H$, define composition $g = f \circ h$,

$$g : \mathbb{X} \to \mathbb{F}_q,$$

by the rule

$$g(w) = (f \circ h)(w) = h(f(w)).$$

Define composition $G = F \circ H$ of two families F and H as follows.

$$G = \{g_{ij} = f_i \circ h_j : i \in I, j \in J\},$$

where $I = \{1, \dots, N\}$, $J = \{1, \dots, T\}$.

Theorem 4 *Let $F = \{f_1, \dots, f_N\}$ be an ϵ-U $(N; K, M)$ hash family. Let Let $H = \{h_1, \dots h_T\}$ be a δ-R $(M; \ell)$ quantum hash generator.*

Then the composition $G = F \circ H$ is an Δ-R $(K; s)$ quantum hash generator, where

$$s = \log N + \ell \tag{5}$$

and

$$\Delta \le \epsilon + \delta. \tag{6}$$

Proof. See the paper [3] for the proof. □

In [3] and in the Appendix B presented different explicit constructions of quantum hash functions based on the above Theorem.

References

[1] F. Ablayev, A. Vasiliev : Quantum Hashing, 2013, arXiv:1310.4922 [quant-ph] (2013)

[2] F. Ablayev, A. Vasiliev : Cryptographic quantum hashing, Laser Physics Letters Vol. 11 issue 2, p. 025202, 2014

[3] F. Ablayev, M. Ablayev : Quantum Hashing via Classical ϵ-universal Hashing Constructions, 2014 arXiv:1404.1503 [quant-ph] (2014)

[4] F. Ablayev, A. Vasiliev : Computing Boolean Functions via Quantum Hashing, Computing with New Resources, LNCS, Springer, Essays Dedicated to Jozef Gruska on the Occasion of His 80th Birthday, 149-160, 2014.

[5] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf : Quantum fingerprinting. Phys. Rev. Lett. 87, 167902 (2001)

[6] D. Gavinsky, T. Ito: Quantum fingerprints that keep secrets. Quantum Information & Computation Volume 13 Issue 7–8, 583–606 (2013)

[7] D. Gottesman, I. Chuang: Quantum digital signatures, Technical report, available at http://arxiv.org/abs/quant-ph/0105032, 2001.

[8] R. Freivalds: Probabilistic Machines Can Use Less Running Time. Proceedings of the IFIP Congress 77, Toronto, Canada, 1977. North-Holland, 1977: 839–842, (1977)

[9] A. Montanaro and T. Osborne: Quantum Boolean functions. Chicago Journal of Theoretical Computer Science, 1, 2010. arXiv:0810.2435.

[10] A. Razborov, E. Szemeredi, and A. Wigderson: Constructing small sets that are uniform in arithmetic progressions. Combinatorics, Probability & Computing, 2: 513–518, 1993.

[11] D.R. Stinson. On the connections between universal ϵ-hashing, combinatorial designs and error-correcting codes. Congressus Numerantium 114, 7–27, (1996)

Appendix A. Proof of Theorem 3.

Let $B_{\delta,q} = \{b_1, \ldots, b_T\}$ determines δ-good family $H_{\delta,q}$. We let $H = H_{\delta,q}$ in the proof. For function $h_b \in H$ we define $\psi_{h_b} : \mathbb{F}_q \to \mathcal{H}^2$ as

$$|\psi_{h_b}(a)\rangle = e^{\frac{2\pi i}{q} h_b(a)}|1\rangle = e^{\frac{2\pi i}{q} ab}|1\rangle$$

and define function $\psi_H : \mathbb{F}_q \to (\mathcal{H}^2)^{\otimes(s+1)}$ by the rule

$$|\psi_H(a)\rangle = \frac{1}{\sqrt{T}} \sum_{j=1}^{T} |j\rangle \left|\psi_{h_{b_j}}(a)\right\rangle = \frac{1}{\sqrt{T}} \sum_{j=1}^{T} e^{\frac{2\pi i}{q} ab_j} |j\rangle|1\rangle.$$

We consider the following projection $\psi_H : \mathbb{F}_q \to (\mathcal{H}^2)^{\otimes s}$ of function ψ_H (speaking informally we omit the $|1\rangle$ from the above definition of ψ_H)

$$|\psi_H(a)\rangle = \frac{1}{\sqrt{T}} \sum_{j=1}^{T} e^{\frac{2\pi i}{q} ab_j} |j\rangle.$$

The quantum state $|\psi_H(a)\rangle$ composed from s qubits. To show that ψ_H is δ-R $(q; s)$ quantum hash function we prove the δ-resistance of ψ_H. Consider a pair a, a' of different elements from $\mathbb{F}_q$ and their inner product $\langle \psi_H(a) \,|\, \psi_H(a')\rangle$. Recall that the inner product of two complex vectors $|\alpha\rangle = (\alpha_1, \ldots, \alpha_T)$ and $|\beta\rangle = (\beta_1, \ldots, \beta_T)$ is the sum $\langle \alpha \,|\, \beta\rangle = \sum_j \alpha_j \bar{\beta}_j$ where $\bar{\beta}_j$ is the complex conjugate of β_j. Using the fact that the conjugate of $e^{i\phi}$ is $e^{-i\phi}$, and the fact that $B_{\delta,q}$ is δ-good we have that

$$\langle \psi_H(a) \,|\, \psi_H(a')\rangle = \frac{1}{|B|} \sum_{b \in B} e^{\frac{2\pi i}{q}(a-a')b} \leq \lambda(B_{\delta,q}) \leq \delta.$$

$\square$

Notice that the construction of quantum hash function ψ, generated by quantum hash generator $H_{\delta,q}$, presented in [2] terms of real valid amplitudes.

Appendix B.

6 Explicit Constructions of Quantum Hash Functions Based on Classical Universal Hashing

The following statement is a corollary of Theorem 4 and a basis for explicit constructions of quantum hash functions in this section. Let q be a prime power and $\mathbb{F}_q$ be a field.

Theorem 5 *Let $F = \{f_1, \dots, f_N\}$ be an ϵ-U $(N; K, q)$ hash family, where $f_i : \mathbb{X} \to \mathbb{F}_q$. Let $H_{\delta,q}$ be an optimal δ-R $(q, O(\log\log q)$ quantum hash generator. Then family $G = F \circ H_{\delta,q}$ is a Δ-R $(K; s)$ quantum hash generator, where*

$$s \leq \log N + O(\log\log q) \quad \text{and} \quad \Delta \leq \epsilon + \delta.$$

Proof. family $H_{\delta,q} = \{h_1, \dots, h_T\}$, where $h_i : \mathbb{F}_q \to \mathbb{F}_q$, $T = \lceil (2/\delta^2)\ln(2q) \rceil$, $\ell = 1$, and $s = \log T + 1 \leq \log n + \log\log q + 2\log 1/\delta + 3$ is δ-R $(q; s)$ quantum hash generator. □

6.1 Quantum hashing based on Freivalds' fingerprinting

For a fixed positive constant k let $\mathbb{X} = \{0,1\}^k$. Let $c > 1$ be a positive integer and let $M = ck\ln k$. Let $\mathbb{Y} = \{0, 1, \dots, M-1\}$.

For the i-th prime $p_i \in \mathbb{Y}$ define a function (fingerprint) $f_i : \mathbb{X} \to \mathbb{Y}$ by the rule $f_i(w) = w \pmod{p_i}$. Here we treat a word $w = w_0 w_1 \dots w_{k-1}$ also as an integer $w = w_0 + w_1 2 + \dots + w_{k-1}2^{k-1}$. Consider the set

$$F_M = \{f_1, \dots, f_{\pi(M)}\}$$

of fingerprints. Here $\pi(M)$ denotes the number of primes less than or equal to M. Note that then $\pi(M) \sim M/\ln M$ as $M \to \infty$. Moreover,

$$\frac{M}{\ln M} \leq \pi(M) \leq 1.26\frac{M}{\ln M} \qquad \text{for} \quad M \geq 17.$$

The following fact is based on a construction, "Freivalds' fingerprinting method", due to Freivalds [8].

Property 3 *The set F_M of fingerprints is a $(1/c)$-U $(\pi(M); 2^k, M)$ hash family.*

Proof (sketch). For any pair w, w' of distinct words from $\{0,1\}^k$ the number $N(w,w') = |\{f_i \in F_M : f_i(w) = f_i(w')\}|$ is bounded from above by k. Thus, if we pick a prime p_i (uniformly at random) from $\mathbb{Y}$ then

$$Pr[f_i(w) = f_i(w')] \leq \frac{k}{\pi(M)} \leq \frac{k \ln M}{M}.$$

Picking $M = ck \ln k$ for a constant c gives $Pr[f_i(w) = f_i(w')] \leq \frac{1}{c} + o(1)$. $\square$

Theorem 5 and Property 3 provide the following statement.

Theorem 6 *Let $c > 1$ be a positive integer and let $M = ck \ln k$. Let $q \in \{M, \ldots, 2M\}$ be a prime. Let $\delta = 1/(\log q)^{O(1)}$ and let $H_{\delta,q}$ be an optimal δ-R $(q, O(\log\log q)$ quantum hash generator. Then family $G = F_M \circ H_{\delta,q}$ is a Δ-R $(2^k; s)$ quantum hash generator, where*

$$s \leq \log ck + O(\log\log k) \quad \textit{and} \quad \Delta \leq \frac{1}{c} + \delta.$$

Proof. From Theorem 5 From the choice of c above we have that $M = ck \ln k$. Thus

$$s = \log|F_M| + O(\log\log q) \leq \log \pi(M) + \log\log M \leq \log ck + O(\log\log k).$$

$\square$

Construction of ψ_G. For a word $w \in \{0,1\}^k$ we define ψ_G by the rule

$$|\psi_G(w)\rangle = \frac{1}{\sqrt{|F_M|T}} \sum_{l=1,j=1}^{|F_M|,T} e^{\frac{2\pi i}{q} h_j(f_l(w))} |lj\rangle.$$

Remark 1 *Note that from Theorem 1 we have*

$$s \geq \log k + \log\log(ck \ln k) - \log\log\left(1 + \sqrt{2/(1-\delta)}\right) - 1.$$

This lower bound shows that the quantum hash function ψ_G is good enough in the sense of the number of qubits used for the construction.

6.2 Quantum hashing and error-correcting codes

Let q be a prime power and let $\mathbb{F}_q$ be a field. An $(n,k,d,)$ error-correcting code is called *linear*, if $\Sigma = \mathbb{F}_q$, and $\mathcal{C} = \{C(w) : w \in \mathbb{F}_q^k\}$ is a subspace of $(\mathbb{F}_q)^n$. We will denote such linear code by an $[n,k,d,]_q$ code.

Theorem 7 *Let $\mathcal{C}$ be an $[n,k,d]_q$ code. Then for arbitrary $\delta \in (0,1)$ there exists a Δ-R $(q^k;s)$ quantum hash generator G, where $\Delta = (1-d/n)+\delta$ and $s \leq \log n + \log\log q + 2\log 1/\delta + 4$.*

Proof. The following fact was observed in [**?**, 11]. Having an $[n,k,d]_q$ code $\mathcal{C}$, we can explicitly construct a $(1-d/n)$-U $(n;q^k;q)$ hash family $F_{\mathcal{C}}$.

By Theorem 5 a composition $G = F_{\mathcal{C}} \circ H_{\delta,q}$ is an Δ-R $(q^k;s)$ quantum hash generator, where $\Delta = (1-d/n)+\delta$ and $s \leq \log n + \log\log q + 2\log 1/\delta + 4$. □

6.2.1 Quantum hash function via Reed-Solomon code

As an example we present construction of quantum hash function, using Reed-Solomon codes.

Let q be a prime power, let $k \leq n \leq q$, let $\mathbb{F}_q$ be a finite field. A *Reed-Solomon* code (for short RS-code) is a linear code

$$C_{RS} : (\mathbb{F}_q)^k \to (\mathbb{F}_q)^n$$

having parameters $[n,k,n-(k-1)]_q$.

$$C_{RS}(w) = (P_w(a_1)\dots P_w(a_n)).$$

Using Reed-Solomon codes, we obtain the following construction of quantum hash generator.

Theorem 8 *Let q be a prime power and let $1 \leq k \leq n \leq q$. Then for arbitrary $\delta \in (0,1)$ there is a Δ-R $(q^k;s)$ quantum hash generator G_{RS}, where $\Delta \leq \frac{k-1}{n} + \delta$ and $s \leq \log(q\log q) + 2\log 1/\delta + 4$.*

Proof. Reed-Solomon code C_{RS} is $[n,k,n-(k-1)]_q$ code, where $k \leq n \leq q$. Then according to Theorem 7 there is a family G_{RS}, which is an Δ-R $(q^k;s)$

quantum hash generator with stated parameters. $\square$

In particular, if we select $n \in [ck, c'k]$ for constants $c < c'$, then $\Delta \leq 1/c + \delta$ for $\delta \in (0,1)$ and in according to Theorem 1 we get that

$$\log(q \log q) - \log\log\left(1 + \sqrt{2/(1-\Delta)}\right) - \log c'/2 \leq s \leq$$

$$\leq \log(q\log q) + 2\log 1/\Delta + 4.$$

Thus, Reed Solomon codes provides good enough parameters for resistance value Δ and for a number s of qubits we need to construct quantum hash function ψ_{RS}.

Explicit constructions of G_{RS} and $\psi_{G_{RS}}$. Define $(k-1)/q$-U $(q; \mathbb{F}_q^k; q)$ hash family $F_{RS} = \{f_a : a \in A\}$ based on C_{RS} as follows. For $a \in A$ define $f_a : (\mathbb{F}_q)^k \to \mathbb{F}_q$ by the rule

$$f_a(w_0 \dots w_{k-1}) = \sum_{i=0}^{k-1} w_i a^i.$$

Let $H_{\delta,q} = \{h_1, \dots, h_T\}$, where $h_j : \mathbb{F}_q \to \mathbb{F}_q$ and $T = \lceil (2/\delta^2)\ln 2q \rceil$. For $s = \log n + \log T + 1$ composition $G_{RS} = F_{RS} \circ H_{\delta,q}$, defines function

$$\psi_{G_{RS}} : (\mathbb{F}_q)^k \to (\mathcal{H}^2)^{\otimes s}$$

for a word $w \in (\mathbb{F}_q)^k$ by the rule.

$$\psi_{G_{RS}}(w) =$$

$$= \frac{1}{\sqrt{n}} \sum_{i=1}^{n} |i\rangle \otimes \left(\frac{1}{\sqrt{T}} \sum_{j=1}^{T} |j\rangle \left(\cos\frac{2\pi h_j(f_{a_i}(w))}{q}|0\rangle + \sin\frac{2\pi h_j(f_{a_i}(w))}{q}|1\rangle \right) \right).$$

Quantum Attacks against Iterated Block Ciphers

Marc Kaplan

Abstract

We study the amplification of security against quantum attacks provided by iteration of block ciphers. In the classical case, the Meet-in-the-middle attack reduces the time required to break double iterations to only twice the time it takes to attack a single block cipher. Here, we prove that for quantum adversaries, two iterated ideal block ciphers are more much difficult to attack than a single one. We quantize the Meet-in-the-middle attack and use tools from quantum complexity theory to prove that it is optimal. We then quantize a technique against 4-encryption called the dissection attack. Contrary to the classical case, this quantum attack has a better time complexity than a quantum Meet-in-the-middle attack. It also shows that the resistance against quantum attacks decreases when the number of iteration grows.

1 Introduction

Quantum information processing has deeply changed the landscape of classical cryptography. In particular, cryptosystems based on integer factoring and discrete logarithm are known to be completely insecure against quantum computers. This opened the field of *post-quantum cryptography*, which tries to restore the security of classical cryptosystems against quantum attacks. Most research is devoted to public key cryptography and the common belief is that symmetric cryptography is not affected by quantum computing because security can be amplified by increasing key sizes.

This belief is based on the fact that symmetric cryptosystems are usually subject to generic attacks whose quantumization allow only polynomial speedups. However, attacking realistic, complex cryptosystems may require more effort than just applying basic quantum algorithms and understanding precisely the security against quantum attacks may require careful analysis [Ber10]. Our work aims to show that the tools developed to understand quantum speedups can find fruitful applications in cryptographic settings. Specifically, we use quantum walk algorithms [Amb07] to design quantum attacks and the generalized adversary method [HLŠ07] to prove security results.

We focus here on one of the most fundamental situation in symmetric cryptography: block cipher encryption: a plaintext is decomposed into constant-size blocks, and each one of them is encrypted using a permutation specified by a secret key. Block cipher encryption is widely used in practice. It is also an important block for building other cryptographic primitives.

We work at an abstract level in the ideal-cipher model, in which the block cipher is a collection of N random permutations of $[M]$, where $[N]$ is the space of keys, and $[M]$ the space of blocks. The set of permutations is public, and anyone can efficiently compute $F_i(X)$ and $F_i^{-1}(X)$ where F_i is the permutation specified by the secret key i and X is a block. We consider an attacker that knows a few pairs of plaintext with corresponding ciphertexts, all encrypted with the same key. Its goal is to recover the secret key that was used for encryption.

Although increasing the key length is a neat theoretical answer, it may not be possible to implement starting from a specific block cipher with fixed parameters. The question of security amplification was raised when brute-force attacks against the DES block cipher became realistic [DH77, MH81]. A simple attempt to increase the key size is to compose permutations with independent keys. For double encryption, the size is doubled, but there is a clever attack against this construction. Suppose that an attacker knows a pair of plaintext-ciphertext (P, C). These satisfy $C = F_{k_2}(F_{k_1}(P))$, where (k_1, k_2) are the keys used for encryption. Since inverse permutations can be computed, an attacker can construct tables $F_k(P)$ and $F_{k'}^{-1}(C)$ for every possible keys k, k'. Finding a collision $F_{k_1}(P) = F_{k_2}^{-1}(C)$ reveals the keys used for encryption.

This attack, known as the Meet-in-the-middle attack, shows that it only takes twice more time to attack double iterations than it takes to attack a single one. A naive cryptographer would expect here a quadratic improvement. Of course, this is optimal up to a factor two. The Meet-in-the-middle attack shows that even a simple idea like security amplification by iteration should be carefully studied. It also has practical consequences, and led to the standardization of triple-DES rather than the insecure double-DES.

We address the question of how resistant composition is against quantum adversaries. An obvious quantum attack against the double encryption is to use a collision finding algorithm, for example Ambainis' algorithm for the element distinctness problem Ambainis [Amb07]. This algorithm is optimal with respect to the number of queries to the input [AS04]. In our case, it extracts the keys with $N^{2/3}$ quantum queries to the permutations. However, the key extraction problem for double encryption has more structure than the element distinctness problem and there is no clear indication that this approach is optimal in this case. The problem has a lot more possible inputs and more queries are allowed. For this reason, there is no obvious way of proving the optimality of Ambainis' algorithm for key extraction by reduction from Element Distinctness.

In Section 2, we prove using the generalized adversary method that $N^{2/3}$ queries are required to extract the keys in the case of 2-encryption (**Theorem 3**). Starting from an adversary matrix for Element Distinctness, we build an adversary matrix for the new problem (**Lemma 1**). The consequence is that for quantum computers, contrary to the classical ones, double encryption is harder to break than a single ideal cipher.

A surprising corollary is that classical and quantum time-space products are very different (**Corollary 1**). In the classical case, the Meet-in-the-middle attack is time-efficient for an attacker that is willing to pay with more space, but the global time-space product is similar to the one achieved by an exhaustive search. Using quantum algorithms, the time-space product of the optimal algorithm of Ambainis is worse than an exhaustive search. While this may not be a surprise from the point of view of quantum complexity theory (see e.g. the conclusion of [BDH$^+$05]), this suggests that the time-space product, a common way of evaluating classical attacks [DDKS12], may not be the correct figure of merit to evaluate quantum attacks.

The results obtained for two iterations suggest that composition could be a good tool to amplify the resistance against quantum attacks because it prevents the quadratic speedups allowed by the quantization of an exhaustive search. We investigate this question further in Section 3 by looking at the case of 4-encryption. We give a quantization of the dissection attack for 4-encryption recently introduced by Dinur, Dunkelman, Keller and Shamir [DDKS12]. In order to quantify time and space complexity of the attack, we use the framework of quantum walks. In the classical world, this attack is not better than the Meet-in-the-middle attack in terms of time, but can be used to decrease the time-space product. Surprisingly, we show that the quantum attack can also decrease the time complexity, compared to a quantum Meet-in-the-middle attack (**Theorem 4**). Moreover, this shows that the resistance against quantum attacks decreases when the number of iterations goes from two to four.

While these tools appear to be very helpful to study two encryptions and four encryptions, we are not in position to make a statement for general multiple encryptions. Using the generalized adversary method and quantum walks has been very fruitful for studying Merkle puzzles in a quantum world [BHK$^+$11], in which these techniques were used to devise attacks and prove their optimality. We present here another important cryptographic scenario in which these tools can be applied to derive new results. A similarity between the two scenarios is that polynomial speedups are very insightful. Many specifically quantum techniques are available to study such speedups and their optimality in black-box settings. Even if the main question about successive encryption remains open, we hope that our work demonstrates that quantum techniques can be very interesting for PQ cryptography, and that it will motivate further interactions between quantum computer scientists and classical cryptographers.

2 Optimality of the quantum Meet-in-the-middle

In what follows, M and N are two integers of comparable size and $[N]$ is the set of integers from 1 to N. We denote $\mathcal{S}_{[N]}$ the set of permutations of $[N]$. The space of keys is $[N]$ and the space of blocks $[M]$.

We consider problems with inputs given as oracles (or black-boxes). Usually, we consider these inputs as functions f, and a classical query to the input returns $f(x)$ for some x in the domain of f. In some cases, we may also consider an input f as a string where f_i denotes the result of the classical query i to f. This notation is convenient in particular when considering adversary matrices whose entries are indexed by inputs to the problem. In the quantum setting, the only difference is that attackers can make quantum queries to the input. A brief exposition of the underlying model with the main theorems that we use to derive our results can be found in Appendix A.

The Element Distinctness problem has been extensively studied in quantum query complexity. In particular, Ambainis' quantum walk based algorithm [Amb07] is known to be optimal for this problem [AS04].

Definition 1. *The Element Distinctness (ED) problem takes input $F : [N] \to [M]$ with the promise that there exists a pair $i, j \in [N]$ such that $F(i) = F(j)$. The problem is to output the pair (i, j).*

In this paper, we use the slightly more structured problem known as *Claw Finding* [BHT98].

Definition 2. *Given two one-to-one functions $G : [N/2] \to [M]$ and $H : [N/2] \to [M]$, a claw is a pair $x, y \in [N]$ such that $G(x) = H(y)$. The Claw Finding (CF) problem is, on input G, H to return a claw (x, y) given that there is exactly one.*

It is easy to prove that CF and ED are equivalent up to constant factors. Given an input F for ED, an input for CF can be obtained by randomly cutting F into two functions. The probability that the two colliding elements are split is 1/2 and running the algorithm for CF a few times on different random cuts is sufficient to find the collision with high probability. Therefore, upper and lower bounds for ED apply similarly to CF, up to constant factors.

Theorem 1. *For $M \geq N$, the quantum query complexity and time complexity of ED and CF are $\Theta(N^{2/3})$. The most time-efficient algorithm for these problems uses memory $O(N^{2/3})$.*

The goal of this section is to study the problem of extracting keys from the double iteration of an ideal cipher. In this context, we assume that the quantum cryptanalyst has implemented the publicly known block cipher on a quantum computer. He then receives the classical data consisting in couples of plaintext (P) and ciphertext (C), all encrypted with the same key. Finally, he uses this data to extract the secret key with the help of the quantum computer. We assume that there is only one key that maps P to C. Equivalently, we can assume that the attacker knows a few pairs (P_i, C_i), all encrypted with the same keys. This ensures that the key mapping P_i to C_i for all i is unique with very high probability. This can always be simulated by giving only one pair to the attacker and increasing the size of the blocks. This introduces constant factors in the complexity analysis, and enforces the permutations to have a product structure, but do not induce any fundamental change to the security proof given here. Notice that applying this trick implies that M and N are then not comparable anymore, which is an important remark for Section 3.

Definition 3. *The 2-Key Extraction ($KE_2^{P,C}$) problem with $P, C \in [M]$ takes input $\mathcal{F}$ where $\mathcal{F} = \{F_1, \dots, F_N\}$ is a collection of permutations $F_i \in \mathcal{S}_{[M]}$ with the promise that there exists a unique couple (k_1, k_2) such that $F_{k_2}(F_{k_1}(P)) = C$. The goal of the problem is to output the pair (k_1, k_2).*

It is easy to prove that the complexity of the problem is independent of the pair (P, C). An algorithm for a given pair (P, C) can be easily adapted to solve the problem for another pair (P', C'). Let σ be the permutation that transposes P' and P and C' and C. It suffices to conjugate every permutation of an input $\mathcal{F} = \{F_1, \dots, F_N\}$ with σ before running the algorithm for $KE_2^{P,C}$. When it is clear from the context and unnecessary for proofs, we drop the exponent and write only KE_2.

The next theorem shows an upper bound on the complexity of KE_2. The idea is simply to reduce KE_2 to CF.

Theorem 2. *There exists a quantum algorithm that solves KE_2 in time $O(N^{2/3})$ using memory $O(N^{2/3})$.*

Proof. Assume that $\mathcal{F} = \{F_1, \dots, F_N\}$ is an input of $KE_2^{P,C}$. We construct the following input to CF: a pair of functions $G_1, G_2 : [N] \to [M]$, defined by $G_1(x) = F_x(P)$ and $G_2(y) = F_y^{-1}(C)$. Suppose that the algorithm for

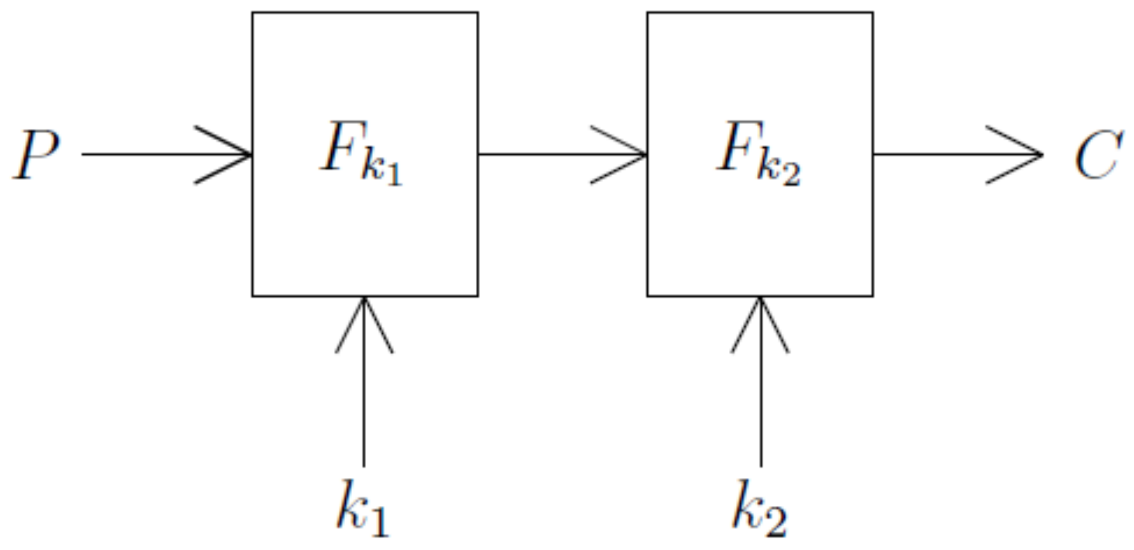

Figure 1: The 2-encryption problem consists in recovering the keys k_1 and k_2, given a plaintext P and a corresponding ciphertext C

CF returns a pair (x^*, y^*). This implies that it is the unique pair such that $F_{x^*}(P) = F_{y^*}^{-1}(C)$, leading to $F_{y^*}(F_{x^*}(P)) = C$. It is therefore the pair of keys used to encrypt P. By Theorem 1, this gives an algorithm for KE_2 in $O(N^{2/3})$ time. □

We also prove the following lower bound on the problem.

Theorem 3. *A quantum algorithm that solves KE_2 needs $\Omega(N^{2/3})$ quantum queries to the input $\mathcal{F} = \{F_1, \ldots, F_N\}$, including queries to inverse permutations, except with vanishing probabilities.*

A lower bound on query complexity translates into a lower bound on time complexity. Therefore, combining the upper and lower bounds on time complexity with the upper bound on memory gives the following corollary.

Corollary 1. *The most time-efficient attack on KE_2 has time-space product $O(N^{4/3})$.*

Notice that a simple Grover search leads to time $O(N)$ with logarithmic space, thus having a time-space product of N, better than the best known algorithm for CF. The proof of Theorem 3 has to deal with two important differences between CF and KE_2:

- It is possible to query a permutation F_i on any input $X \in [M]$.
- It is possible to query inverse permutations F_i^{-1}.

These differences imply that there is no obvious query-preserving reduction from CF to KE_2. One important issue is that, since F_i is a permutation, querying $F_i(X)$ with $X \neq P$ gives information on $F_i(P)$. Intuitively, we believe this information is so small that it may not be useful to solve the problem. However, this becomes very problematic in the quantum setting, where a single query can be made in superposition of all the inputs, preventing strategies that consist in building the reduction on the fly, when the queries to the input are done. To overcome this issue, we use a specific tool from quantum query complexity known as the generalized adversary method.

We prove a slightly stronger lower bound result by considering the decision version of KE_2. In this case, the problem is to determine if there exists keys k_1, k_2 such that $C = F_{k_2}(F_{k_1}(P))$. We denote this version $d{-}KE_2$. Of course, an algorithm for the search version can easily be transformed into an algorithm for the decision version. Consequently, a lower bound on the decision version is also a lower bound on the search problem. We compare this algorithm with the decision version of claw finding, denoted $d{-}CF$. The bounds given in Theorem 1 apply equivalently to the decision version of these problems.

The proof of Theorem 3 is in two steps. In Lemma 1, we prove a lower bound in the *worst-case quantum query complexity* using the generalized adversary method. In the second step, which is the proof of Theorem 3, we use a self-reducibility argument to prove that the probability of solving the problem on a random input with less queries than in the worst case is vanishing. Both proofs are given in full details in Appendix B

Lemma 1. *A quantum algorithm that solves $d\mathrm{-}KE_2$ needs $\Omega(N^{2/3})$ quantum queries to the input $\mathcal{F} = \{F_1, \ldots, F_N\}$, including queries to inverse permutations.*

In order to measure the gains achieved by quantum algorithms, we use a quantity similar to the one used by Dinur, Dunkelman, Keller and Shamir [DDKS12]. We consider $\log C/\log Q$, where C is a classical complexity measure and Q is its quantum counterpart. Intuitively, this corresponds to the factor by which the effective key size is decreased when the attack is quantized. For example the gain in time for Grover search over classical exhaustive search is 2. The gain for the time-space product is similar because both Grover and exhaustive search require only logarithmic space.

The gain for the Meet-in-the-middle attack is very different. The gain in time is 3/2, using the algorithm presented above. Since this algorithm is optimal, it is the largest possible gain. The gain for the time-space product of this algorithm is also 3/2. Interestingly, there exists other algorithms for ED leading to different gains. For example, the algorithm based on amplitude amplification (AA) of [BDH$^+$05] leads to a gain in time of 4/3 (over the classical Meet-in-the-middle attack), and a gain in time-space product of 8/5, better than the most time-efficient attack. The most time-efficient algorithm is not the one leading to the most important gain in time-space, and an attacker that is willing to pay with more time can save on the time-space product.

3 Quantum attack against 4-encryption

We now apply tools from quantum complexity theory to study the case of four iterated encryptions. We assume again that the attacker knows sufficiently many pairs (P_i, C_i) of plaintext-ciphertext in order to ensure that, with very high probability, there is only one quadruple of keys (k_1, k_2, k_3, k_4) that satisfies the relations $C_i = E_{k_4}(E_{k_3}(E_{k_2}(E_{k_1}(P_i))))$ for all i.

The classical Meet-in-the-middle attack can be applied in this situation by considering the pairs (k_1, k_2) and (k_2, k_3) as single keys. This requires

		Time	Time-space
Exhaustive search	Classical	N^2	N^2
	Quantum	N	N
	Gain	2	2
Collision finding based on MITM	Classical	N	N^2
	Quantum	$N^{2/3}$	$N^{4/3}$
	Gain	1.5	1.5
Collision finding based on AA	Classical	N	N^2
	Quantum	$N^{3/4}$	$N^{5/4}$
	Gain	$\simeq 1.3$	$\simeq 1.6$

Table 1: Summary of attacks against 2-encryption and gains of quantization

time and memory $O(N^2)$, and thus time-space product $O(N^4)$. This was the best known algorithm until the recent dissection attack [DDKS12], which still requires time $O(N^2)$ but memory only $O(N)$. This significantly improves the time-space product to $O(N^3)$.

The basic idea of the dissection attack is to make an exhaustive search of the intermediate value after two encryptions. The candidate values are then checked using a Meet-in-the-middle procedure. A notable difference with the previous case is that the complexity of the problem is now a function of both M and N. We assumed in the beginning that M and N were of comparable sizes. In order to keep this assumption, we cannot assume anymore that the attacker has a single pair of data. Instead, we assume that it has enough pairs, all encrypted with the same data, to be sure that, with large probability, there is only one quadruple of keys consistant with all the data. It can be shown that four pairs of plaintext with corresponding ciphertext.

Definition 4. *The 4-Key Extraction ($KE_4^{P,C}$) problem with $P = (P_1, P_2, P_3, P_4) \in [M]^4$ and $C = (C_1, C_2, C_3, C_4) \in [M]^4$ takes input $\mathcal{F}$ where $\mathcal{F} = \{F_1, \ldots, F_N\}$ is a collection of permutations of $[M]$. The goal of the problem is to output (k_1, k_2, k_3, k_4) such that $F_{k_4}(F_{k_3}(F_{k_2}(F_{k_1}(P_i)))) = C_i$ for all i.*

The attack uses the quantum Meet-in-the-middle algorithm presented in the previous section as a subroutine. The basic idea is to compose this algorithm with a Grover search of the value in the middle. Quantum query complexity has the remarkable property, derived from the generalized adversary method, that for two compatible functions f and g, $Q(f(g(x^1), \ldots, g(x^n))) = O(Q(f)Q(g))$ [LMR+11]. This leads to a $O(M^{1/2}N^{2/3})$ upper bound on the number of queries needed to solve the problem. However, the composition theorem that proves this upper bound holds for *quantum query* complexity, whereas we are interested here in bounding the *time* and *space* used by a quantum algorithm. For this purpose, we give an explicit composed algorithm. This algorithm is a quantized version of the $Dissect_2(4,1)$ algorithm of Dinur, Dunkleman, Keller and Shamir [DDKS12]. In the classical setting, this algorithm achieves the best known time-space product for 4-encryption.

Theorem 4. *There exists a quantum algorithm that solves KE_4 in time $O(M^{1/2}N^{2/3})$ and using memory $O(N^{2/3})$. The time-space product for this attack is $O(M^{1/2}N^{4/3})$.*

The full proof of Theorem 4 is given in Appendix C. If $M = N$, the attack given by the Theorem has a time complexity $O(N^{7/6})$ and time-space product $O(N^{11/6})$. The most important difference is that, while the classical Dissection attack could not improve the time complexity but only the time-space product, its quantum equivalent is able to decrease the time as well.

In table 2, we compare three different attacks against 4-encryption: (i) exhaustive search over the whole key-space, (ii) classical and quantum Meet-in-the-middle attack where both the first and the second pair of keys are treated as single keys, and (iii) the dissection attack and its quantized version. The quantization of exhaustive search gives a very good gain, but its time-performance is poor. Applying the Meet-in-the-middle attack from the previous section gives both poor gains and poor performances. The quantization of the dissection attack gives better gains than what we observed for 2-encryption, but we have no indications that this algorithm is optimal.

		Time	Time-space
Exhaustive search	Classical	N^4	N^4
	Quantum	N^2	N^2
	Gain	2	2
MITM Attack	Classical	N^2	N^4
	Quantum	$N^{4/3}$	$N^{8/3}$
	Gain	1.5	1.5
Dissection attack	Classical	N^2	N^3
	Quantum	$N^{7/6}$	$N^{11/6}$
	Gain	$\simeq 1.7$	$\simeq 1.63$

Table 2: Summary of attacks against 4-encryption and gains of quantization

4 Conclusion

We have studied security amplification by iterative block ciphers. In the case of double iteration, the quantum Meet-in-the-middle attack is optimal but requires much more time to break two iterated cipher than to break a single ideal one. This indicates that double iteration resists better to quantum attacks than to classical ones. Moreover, the most time-efficient attack has a worse time-space tradeoff than an exhaustive search.

We have then studied the case of 4-encryption, for which we have given a quantized version of the dissection attack. Although we don't know if this attack is optimal, it leads to better performances and gains than a Meet-in-the-middle attack both in terms of time complexity and time-space product. This is in stark contrast with the classical case in which the dissection attack improves only the time-space product but not the time complexity. This indicates that increasing the number of iterations may in fact decrease the resistance against quantum attacks.

Although the general question of security amplification by iteration remains opened, our work shows that the tools from quantum computing, such as quantum walks and the generalized adversary method, are well suited to tackle cryptographic questions.

Our work can be understood as a proof of principle that quantum complexity and algorithms theory has developed powerful tools to tackle important questions in cryptography and post-quantum cryptography. Maybe a better use of these tools can lead to improving and extending the results presented here. Or maybe these tools need to be sharpened in order to apply to specific cryptographic situations. In both cases, we hope that our work will motivate further interactions between classical cryptographers and quantum computer scientists. Such interactions seem crucial to establish a serious approach to post-quantum cryptography. In particular, iterating block ciphers is not a very good procedure to amplify security against classical adversary. We hope that the techniques presented here will be applied to more efficient cryptographic procedures in the future.

References

[Amb07] A. Ambainis. Quantum walk algorithm for element distinctness. *SIAM Journal on Computing*, 37:210–239, 2007.

[AS04] S. Aaronson and Y. Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM*, 51(4):595 – 605, 2004.

[BDH$^+$05] H. Buhrman, C. Dürr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, and R. de Wolf. Quantum algorithms for element distinctness. *SIAM Journal on Computing*, 34, 2005.

[Ber10] D. Bernstein. Grover vs. McEliece. In *Proc. of the Third international conference on Post-Quantum Cryptography (PQCrypto'10)*, pages 73–80, 2010.

[BHK$^+$11] G. Brassard, P. Høyer, K. Kallach, M. Kaplan, S. Laplante, and L. Salvail. Merkle puzzles in a quantum world. In *CRYPTO '11*, 2011.

[BHT98] G. Brassard, P. Høyer, and A. Tapp. Quantum cryptanalysis of hash and claw-free functions. In *3rd Latin American Theoretical Informatics Symposium (LATIN'98)*, volume 1380, pages 163–169, 1998.

[DDKS12] I. Dinur, O. Dunkelman, N. Keller, and A. Shamir. Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. In *Advances in Cryptology – CRYPTO*, 2012.

[DH77] W. Diffie and M.E. Hellman. Exhaustive cryptanalysis of the NBS data encryption standard. *Computer*, 10(6):74–84, 1977.

[HLŠ07] P. Høyer, T. Lee, and R. Špalek. Negative weights make adversaries stronger. In *Proc. of 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 526–535, 2007.

[LMR$^+$11] T. Lee, R. Mittal, B. Reichardt, R. Špalek, and M. Szegedy. Quantum query complexity of state conversion. In *Proc. 52nd IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 344–353, 2011.

[MH81] R. Merkle and M. Hellman. On the security of multiple encryption. *Communication of the ACM*, 1981.

[MNRS11] F. Magniez, A. Nayak, J. Roland, and M. Santha. Search via quantum walk. In *SIAM Journal on Computing*, volume 40(1), pages 142–164, 2011.

[San08] M. Santha. Quantum walk based search algorithms. In LNCS, editor, *Proceedings of 5th Theory and Applications of Models of Computation (TAMC08)*, volume 4978, pages 31–46, 2008.

A Quantum query complexity

The model

In the quantum query complexity model, the goal is to compute some function $F : S \to T$ on input x. For simplicity, we assume in this short presentation that $S \subseteq \{0,1\}^n$. The input is given as a black-box and accessed through *queries*. A classical query ℓ to the input x returns x_ℓ.

In the quantum setting, the algorithm is executed on an architecture with three quantum registers: an input register I, a query register Q and a working register W. The state of the quantum computer when the algorithm starts is $|x\rangle_I|0,0\rangle_Q|0\rangle_W$. There are several equivalent formalism to model quantum queries. Without loss of generality, we consider a quantum query as an operator

$$O : |x\rangle_I|i,y\rangle_Q \longmapsto |x\rangle_I|i, y + x_i \mod 2\rangle_Q.$$

Considering non-boolean inputs is equivalent, up to logarithmic factors.

The quantum algorithm supposed to compute f alternates between quantum query operations O_i and work operations that are unitaries U_i acting on the query register and the working register. After t queries to the input, the state of the quantum computer is $U_tO_tU_{t-1}\dots O_2U_1O_1|x\rangle_I|0,0\rangle_Q|0\rangle_W$.

We assume that the working register contains $\log|T|$ qubits to encode the value $f(x)$. The last step of the algorithm is then to measure these qubits and output the value that is obtained. Finally, an immediate corollary of the model is that the time complexity of f is at least equal to the query complexity of the algorithm. The generalized adversary method can be used to prove a lower bound on query complexity, which in turn bounds the time complexity.

Quantum walks

We first review the paradigm of quantum walks. We use this tool to design an attack against 4-encryption. More precisely, we use it to combine an exhaustive search with a collision finding algorithm. The framework of quantum walks allows to easily analyze the resources used by the algorithm.

A search algorithm aims to find a marked element in a finite set U. Classically, it can be seen as a walk on a graph whose nodes are indexed by subsets of elements of U. Each step of the algorithm consists in walking from one node to another. The algorithm can also maintain a data structure that is updated at each step. After each move, the algorithm checks if the node contains a marked element. The algorithm terminates when a marked element is found.

Magniez, Nayak, Roland and Santha have designed a generic theorem in order to quantize search algorithms expressed as random walks. The cost of the resulting quantum algorithm can be written as a function of S, U and C. These are the cost of setting up the quantum register in a state that corresponds to the stationary distribution, moving unitarily from one node to an adjacent node, and checking if a node is marked, respectively.

Theorem 5. *[MNRS11] Let P be an ergodic and reversible Markov chain with eigenvalue gap δ, and let $\varepsilon > 0$ be a lower bound on the probability that an element chosen from the stationary distribution of P is marked, whenever the set of marked element is non-empty. Then, there exists a quantum algorithm which finds, with high probability, a marked element if there is any at cost of order $\mathsf{S} + \frac{1}{\sqrt{\varepsilon}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C})$.*

Grover's algorithm can be seen as a trivial application of Theorem 5 (see also [San08]). The underlying graph is the complete graph whose nodes are indexed by elements of U, with no data structure. In our case however, the checking procedure is very different because it implies multiple queries to the input, and we use Theorem 5 to design our attack.

The generalized adversary method

We use the generalized adversary method to prove a lower bound on the problem KE_2. The intuition of the method is to consider pairs of inputs leading to different outputs. Each pair is given a weight[1] quantifying how difficult it is to distinguish them. The key point of the method is then to measure the progress made by a single quantum query to distinguish pairs of inputs. This intuition can be formalized in several different ways. We use the spectral version, an elegant algebraic formalization of the previous intuition [HLŠ07].

Definition 5. *Fix a function $F : S \to T$, with $S \subseteq \{0,1\}^n$. A symmetric matrix $\Gamma : S \times S \to \mathbb{R}$ is an adversary matrix for F provided $\Gamma[x,y] = 0$*

[1]The original method uses probability distribution, the generalized method allows for *negative weights* as well.

whenever $F(x) = F(y)$. Let $\Delta_\ell[x, y] = 1$ if $x_\ell \neq y_\ell$ and 0 otherwise. The adversary bound of F using Γ is

$$\mathsf{ADV}^{\pm}(F; \Gamma) = \min_{\ell} \frac{\|\Gamma\|}{\|\Gamma \bullet \Delta_\ell\|},$$

where $\bullet$ denotes entrywise (or Hadamard) product, and $\|A\|$ denotes the spectral norm of A. The adversary bound $\mathsf{ADV}^{\pm}(F)$ is the maximum, over all adversary matrices Γ for F, of $\mathsf{ADV}^{\pm}(F; \Gamma)$.

Høyer, Lee and Špalek introduced the generalized adversary method and proved that the adversary value is a lower bound on quantum query complexity [HLŠ07]. In our proof, we need the fact that the adversary bound characterizes quantum query complexity, up to a constant factor. Our proof starts by considering an adversary matrix for the problem CF such that the adversary bound is equal to the quantum query complexity. The fact that the generalized adversary method is tight was proven by Lee, Mittal, Reichardt, Špalek and Szegedy.

Theorem 6. *[LMR+11] Fix a function $F : S \to T$. The bounded-error quantum query complexity of F is characterized by the general adversary bound:*

$$Q(F) = \Theta(\mathsf{ADV}^{\pm}(F)).$$

B Proof of Theorem 3

We consider an optimal adversary matrix for $d{-}CF$ and use it to build an explicit adversary matrix for $d{-}KE_2$. Details on the adversary method can be found in the Appendix. In this proof, we need the fact that the generalized adversary method gives optimal lower bounds on quantum query complexity. Therefore, we can choose an adversary matrix for CF such that the adversary value given by this matrix is exactly the quantum query complexity of the function. Let Γ_{CF} be this adversary matrix for $d{-}CF$. The rows and columns of Γ_{CF} are indexed by inputs to the problem, that is, pairs of function G_1, G_2.

We now construct an adversary matrix Γ_{KE_2} for the problem $d{-}KE_2^{P,C}$. The rows and columns of Γ_{KE_2} are indexed by collections of permutations $\mathcal{F} = \{F_1, \dots F_N\}$. Given a row u, (resp. a column v), denote $u^{(P,C)}$ the row (resp. $v^{(P,C)}$ the column) of Γ_{CF} corresponding to functions G_1, G_2 defined by $G_1(k) = F_k(P)$ and $G_2(k) = F_k^{-1}(C)$. The input $u^{(P,C)}$ for the problem $d{-}CF$ is called the projection of the input u for the problem $d{-}KE_2$. This operation is represented on figure 2. We simply define the entries of Γ_{KE_2} by $\Gamma_{KE_2}[u,v] = \Gamma_{CF}[u^{(P,C)}, v^{(P,C)}]$. Our goal is now to apply Theorem 6. In order to apply it, we need to compute the values of $\| \Gamma_{KE_2} \|$ and $\max_q \| \Gamma_{KE_2} \bullet \Delta_q \|$, where "$\bullet$" denotes the entry-wise product of matrices[2].

This definition implies that Γ_{KE_2} has a simple tensor product structure. Notice that for two pairs (u,v) and (u',v') of inputs of $d{-}KE_2$ projecting onto the same pair of inputs $(\tilde{u}, \tilde{v})$ of CF, we get by definition $\Gamma_{KE_2}[u,v] = \Gamma_{KE_2}[u',v'] = \Gamma_{CF}[\tilde{u},\tilde{v}]$. Moreover, the number of inputs to $d{-}KE_2$ projecting onto the same of $d{-}CF$ is a constant. Denoting this constant D and $\mathbb{J}_{D\times D}$ the all-one matrix of dimension D, we get $\Gamma_{KE_2} = \Gamma_{CF} \otimes \mathbb{J}_{D\times D}$. This immediately gives the relation

$$\| \Gamma_{KE_2} \| = D \| \Gamma_{CF} \| .$$

The next step is to bound $\max_i \| \Gamma_{KE_2} \bullet \Delta_i \|$ where $\Delta_i[u,v] = 0$ if $u_i = v_i$, and 1 otherwise. A query to an input to $d{-}KE_2$ is a triplet (x,k,b) where $x \in [M], k \in [N]$ and $b \in \{-1, 1\}$ indicates if the query is to a permutation or to its inverse. The query thus returns $F_k^b(x)$.

We show that it is sufficient to consider a special set of queries. The set of queries $\mathcal{I}$ consists of queries $q = (x,k,b)$ where $x = P$ if $b = 1$ and $x = C$ if $b = -1$. We prove that if the value $\| \Gamma_{KE_2} \bullet \Delta_q \|$ is not maximized by a query from $\mathcal{I}$, it is possible to find another matrix with both the same norm and the same tensor product structure such that $\| \Gamma' \bullet \Delta_q \|$ is maximized by a query from $\mathcal{I}$. Formally, we prove the following claim.

Claim: There exists a permutation of rows and columns of Γ_{KE_2} leading to a matrix Γ' such that $\Gamma' = \Gamma'_{CF} \otimes \mathbb{J}_{D\times D}$, where Γ'_{CF} is obtained by permuting

[2]This product is usually denoted $\circ$, we use here a different notation to avoid confusion with the composition of permutations that is also used in this proof

the rows and columns of Γ_{CF} and $\max_q \parallel \Gamma_{KE_2} \bullet \Delta_q \parallel = \max_{q\in\mathcal{I}} \parallel \Gamma' \bullet \Delta_q \parallel$.

We first explain how to finish the proof assuming this claim. A query $q \in \mathcal{I}$ projects onto a query $\tilde{q}$ to inputs of $d{-}CF$. Formally, for a query $q \in \mathcal{I}$, there exists a query $\tilde{q}$ to inputs of $d{-}CF$ such that for any input u of $d{-}KE_2$, $u_q = u_{\tilde{q}}^{(P,C)}$. If $q = (x, k, b) \in \mathcal{I}$, the query $\tilde{q} = (k, b)$ on $u^{(P,C)}$ returns $G_1(k)$ if $b = 1$ and $G_2(k)$ if $b = -1$. By definition of $u^{(P,C)}$, we have $G_1(k) = F_k(P)$ and $G_2(k) = F_k^{-1}(C)$ and thus, $u_q = u_{\tilde{q}}^{(P,C)}$. This implies that for $q \in \mathcal{I}$, $(\Gamma' \bullet \Delta_q)[u, v] = (\Gamma'_{CF} \bullet \Delta_{\tilde{q}})[u^{(P,C)}, v^{(P,C)}]$. The tensor product structure ensures that $\Gamma' \bullet \Delta_q = (\Gamma'_{CF} \otimes \mathbb{J}_{D\times D}) \bullet \Delta_q = (\Gamma'_{CF} \bullet \Delta_{\tilde{q}}) \otimes \mathbb{J}_{D\times D}$, and thus

$$\max_q \parallel \Gamma_{KE_2} \bullet \Delta_i \parallel = \max_{q\in\mathcal{I}} \parallel \Gamma' \bullet \Delta_q \parallel = D \max_{\tilde{q}} \parallel \Gamma'_{CF} \bullet \Delta_{\tilde{q}} \parallel .$$

The last equality is true because since $\mathcal{I}$ and queries to inputs of $d{-}CF$ have the same cardinality, they are in one-to-one correspondance. Maximizing over queries to $\mathcal{I}$ is therefore equivalent to maximizing over queries to inputs to $d{-}CF$. Finally, this shows that the quantum query complexity of KE_2 is at least

$$\begin{aligned}\min_q \frac{\parallel \Gamma_{KE_2} \parallel}{\parallel \Gamma_{KE_2} \bullet \Delta_q \parallel} &= \min_q \frac{\parallel \Gamma' \parallel}{\parallel \Gamma' \bullet \Delta_q \parallel} = \min_{\tilde{q}} \frac{D \parallel \Gamma'_{CF} \parallel}{D \parallel \Gamma'_{CF} \bullet \Delta_{\tilde{q}} \parallel} \\ &= \min_{\tilde{q}} \frac{\parallel \Gamma_{CF} \parallel}{\parallel \Gamma_{CF} \bullet \Delta_{\tilde{q}} \parallel} = \Omega(N^{2/3}).\end{aligned}$$

It only remains to prove the above claim. Suppose that $\parallel \Gamma_{KE_2} \bullet \Delta_q \parallel$ is maximized by a query $q^* = (x^*, k^*, b^*)$. The intuition of the claim is that if $q^* \notin \mathcal{I}$, it can still be projected onto a query to inputs of $d{-}CF$. By mapping these to inputs of the original problems, we get the new matrix Γ'.

Assume that $b^* = 1$ and $x^* \neq P$ (the proof is similar with $b^* = -1$). Let σ denote the transposition $(x^*\ P)$. For an input u $= \{F_i\}_{i\in[N]}$, denote $u^\sigma = \{F_i \circ \sigma\}_{i\in[N]}$ and define Γ' as $\Gamma'[u, v] = \Gamma_{KE_2}[u^\sigma, v^\sigma]$. The operation $u \mapsto u^\sigma$ corresponds to a permutations of rows and columns of Γ_{KE_2}. Denote Π_σ

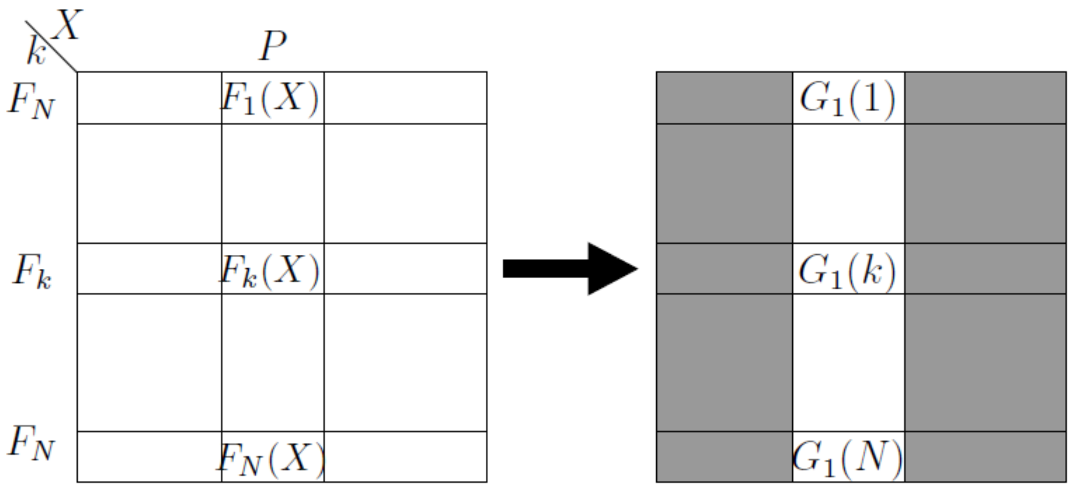

Figure 2: An input to KE_2 can be represented as a table whose rows and columns are indexed by k and X, respectively. Each line of the table is a permutation F_i. The projecting onto an input to CF is a restriction to one column. To build a complete input to CF, one also has to restrict the table of permutations F_k^{-1}.

this permutation, so that $\Gamma' = \Pi_\sigma^\dagger \Gamma_{KE_2} \Pi_\sigma$. Similarly, we have $\Pi_\sigma^\dagger \Delta_{q^*} \Pi_\sigma = \Delta_{q^{**}}$, where $q^{**} = (P, k^*, b^*)$.

Finally, we show that $\Gamma' = \Gamma'_{CF} \otimes \mathbb{J}_{D\times D}$ for some matrix Γ'_{CF}. The sets $\{u^{(P,C)}\}_u$ and $\{u^{(x^*,C)}\}_u$ are equal and therefore, there exists a bijection τ sending $u^{(P,C)}$ to $u^{(x,C)}$. This bijection satisfies $(u^\sigma)^{(P,C)} = \tau(u^{(P,C)})$. The matrix Γ'_{CF} is simply defined as $\Gamma'_{CF}[\tilde{u}, \tilde{v}] = \Gamma_{CF}[\tau(\tilde{u}), \tau(\tilde{v})]$. This gives

$$\begin{aligned} \Gamma'[u, v] &= \Gamma_{KE_2}[u^\sigma, v^\sigma] = \Gamma_{CF}[(u^\sigma)^{(P,C)}, (v^\sigma)^{(P,C)}], \\ &= \Gamma_{CF}[\tau(u^{(P,C)}), \tau(u^{(P,C)})] = \Gamma'_{CF}[u^{(P,C)}, v^{(P,C)}]. \end{aligned}$$

This immediately leads to $\Gamma' = \Gamma'_{CF} \otimes \mathbb{J}_{D\times D}$, and finishes the proof of the claim.

Proof of Theorem 3. We now show that the lower bound proved in Lemma 1 holds on random inputs, except with vanishing probability. The proof is in three steps. In the first step, we prove that the lower bound holds for the average number of queries made by the attack. In the second step, we show that the lower bound also holds when considering the average error.

Finally, we show that the lower bound hold for random inputs, except with vanishing probability.

Step 1: Let $\mathcal{A}$ be an attack such that, given an input $\mathcal{F} = \{F_1, \ldots, F_N\}$ to $KE_2^{P,C}$, returns (k_1, k_2) such that $F_{k_2}(F_{k_1}(P)) = C$ after q queries on average over the inputs and is successful with probability at least $1 - \varepsilon$ for any input. Consider then the following attack: run $\mathcal{A}$ for q/ε queries. If $\mathcal{A}$ stopped, then output the same value. Otherwise output a random value. By Markov's inequality, this new attach is successful with probability at least $1 - 2\varepsilon$ and therefore, by Lemma 1, $q/\varepsilon \geq \Omega(N^{2/3})$.

Step 2: Let $\mathcal{A}'$ be an attack such that, given an input $\mathcal{F} = \{F_1, \ldots, F_t\}$ to $KE_2^{P,C}$, returns (k_1, k_2) such that $F_{k_2}(F_{k_1}(P)) = C$ after q queries on average and is successful with average probability $1 - \varepsilon$, both over the inputs. Consider now the following attack: Choose N random permutations $\{\sigma_1, \ldots \sigma_N\}$, and run the $\mathcal{A}'$ on the input $\mathcal{F}' = \{\sigma_1 \circ F_1, \ldots, \sigma_N \circ F_N\}$. This is equivalent to running $\mathcal{A}'$ on a random input, so that the error made by the attack is now $1 - \varepsilon$ for any input. From Step 1, we get again $q = \Omega(N^{2/3})$.

Step 3: Let $\mathcal{A}''$ be an attack that solves $KE_2^{P,C}$ with error ε on average. Denote Q the random variable indicating the number of queries made by $\mathcal{A}''$ and fix $q = o(N^{2/3})$. Denote $\delta = \Pr[Q \leq q]$. We want to prove that δ is vanishing. Fix two constants k and k' and consider the following attack. Repeat k/δ times the following steps:

1. Choose N random permutation $\sigma_1, \ldots, \sigma_N$. Run $\mathcal{A}''$ on a random input as explained in Step 2, and stop it after $k'q$ queries.
2. If $\mathcal{A}''$ stopped, then output the same value and stop.

If after repeating these two steps k/δ times, no output was produced, output random values.

The number of queries of this attack is at most $kk'q/\delta$. Choosing k and k' large enough, the probability that at least one iteration of the loop is successful can be made arbitrarily close to one, so that the error probability of this attack is arbitrarily close to ε. We have constructed an attack that makes $O(q/\delta)$ queries on average and is successful with average probability $1 - \varepsilon$. From Step 2, we get that $q/\delta = \Omega(N^{2/3})$, which implies $\delta = o(1)$.

□

C Proof of Theorem 4

Assume that the attacker knows the pairs $(P_i, C_i)_{i=1..4}$. This is sufficient to ensure that only a single quadruple of keys is consistent with the data. We devise a search algorithm that consists in a composition of a Grover search with a quantum Meet-in-the-middle attack presented in Section 2. We use the framework of quantum walks in order to compose these two procedures. The details of the theorems that we use are given in the Appendix.

The walk is designed on the complete graph, whose vertices are indexed by elements from $[M]$ who are candidates for the value X after two encryptions. Once the correct value for X is found, it suffices to make $O(N^{2/3})$ queries to the input in order to find the keys, using some of the data (P_i, C_i). This correct value is unique, and the goal of the quantum walk is to find it.

Theorem 5 given in the Appendix states that the cost to find the correct value X is upper bounded by $\mathsf{S} + \frac{1}{\sqrt{\varepsilon}}(\frac{1}{\sqrt{\delta}}\mathsf{U} + \mathsf{C})$, where

- S is the cost for setting up a quantum register in a state that corresponds to the stationary of the classical random walk on the graph distribution
- U is the cost of moving from one node to an adjacent one,
- C is the cost of checking if the value X is the correct one,
- ε is the probability of finding the correct value X, and
- δ is the eigenvalue gap for the complete graph.

In our case, the setup is to build a uniform superposition of all states $|X\rangle$, which can be made with no query to the input, and constant time. The update is the same procedure. The checking consists in running a quantum Meet-in-the-middle attack several times in order to check that the value X satisfies $E_{k_2}(E_{k_1}(P_i)) = X$ and $E_{k_4}(E_{k_3}(X)) = C_i$ for all i

for some value k_1, k_2, k_3, k_4. The probability of finding the correct value is $\varepsilon = 1/M$. Finally, the eigenvalue gap for the complete graph is $\delta = 1 - \frac{1}{M-1}$. Overall, this leads to an attack that makes $O(M^{1/2}N^{2/3})$ queries, and takes the same amount of time. Moreover, the checking procedure is the only procedure using non-constant memory. The attack uses in total $O(N^{2/3})$ memory, leading to a time-space product bounded by $O(M^{1/2}N^{4/3})$.

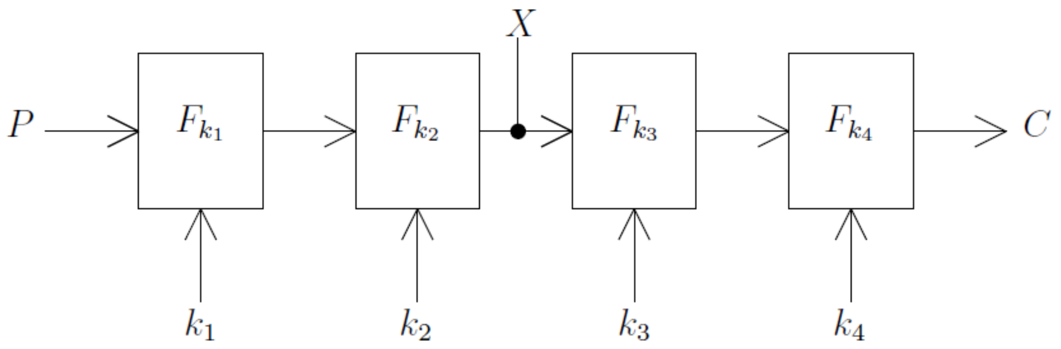

Figure 3: The attack on 4-encryption is an exhaustive search over the central value X combined with a $MITM$ algorithm to check the consistency with the data P, C

Quantum Random Number Generator

Konstantin Kravtsov Sergey Kulik Igor Radchenko

Abstract

A quantum random number generator (QRNG) based on a photoeffect as a truly-random process was proposed and realized as a stand-alone module. It uses a deterministic post-processing algorithm to overcome physical imperfections of real components. Our minimalist design of QRNG prevents possible loopholes and makes it suitable for commercial production. The proposed QRNG provides a binary output stream of 1.2 Mbit/s that successfully passes NIST statistical tests.

Keywords: random number generation, quantum random number generator, randomness extractor.

A random number generator (RNG) is a critical component of all modern key distribution systems (ether classical or quantum), with the exception of protocols where randomization is an implicit property of the key distribution scheme by itself (Ekert-like protocols [1]). Any RNG, which can be modeled with the classical physics laws only, suffers from its deterministic description and, therefore, theoretical possibility of output stream prediction, while a quantum random number generator (QRNG) is supplied by a physical process of quantum nature, which is natively random. At the moment, a broad number of QRNG approaches has been proposed and implemented [2, 3, 4, 5], but still, the nomenclature of commercially available devices remains negligible. The necessity of reliable and replicable QRNG construction have motivated the creation of the present QRNG.

The base of our QRNG is a photoeffect process in an avalanche photo diode stimulated by a non-coherent electromagnetic field. A light emitting diode (LED) and a thin depletion layer metal-resistor-semiconductor avalanche photodiode (APD) are mounted opposite of each other on the same axis (fig. 1). The electrical current flowing through the LED and, thus, the emitted light are so weak that the APD works in Geiger mode.

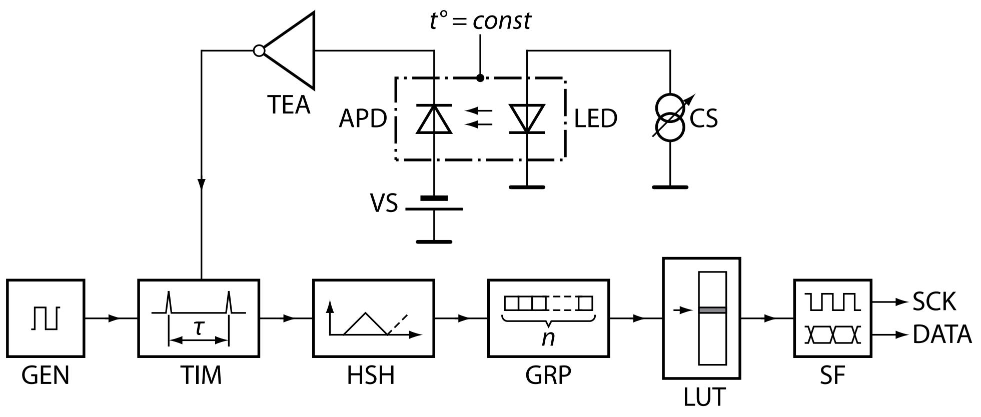

Figure 1: Block diagram of QRNG. LED – light emitting diode, APD – avalanche photo diode, CS and VS – current and voltage sources, TEA – transimpedance amplifier, GEN – clock generator, TIM – time interval measurement, HSH – hash function – randomness extractor, GRP – group letters in words, LUT – look-up table, SF – binary stream forming.

Time intervals between the clicks of the detector measured with a discrete time scale by the TIM constitute a raw sequence used for random number generation. The temperature of the LED and APD is held constant to stabilize their parameters. The LED current is controlled by a feedback loop to maintain a constant average APD count rate of 1.2 Mcps regardless of the spread and drift of actual setup parameters.

The time bin for interval measurements is $T = 20$ ns while the typical line width of the LED is 40 nm with a center wavelength of 630 nm, which gives a coherence time of $\tau_{coh} \sim 10^{-15}$ s $\ll T$. It moves us towards multimode detection regime, where the detector cannot feel the field statistics, so the action of light upon the APD is the same as for a classical field. Under these assumptions, a photoionization process is essentially quantum and therefore truly-random. The Mandel's formula [8] in this case predicts a Poisson statistics of the APD clicks and the exponential decay for the time-between-clicks distribution. However, due to the limitations of real components, the experimental distribution appears to be distorted. The deviations observed can be fully explained by the effects of a dark-time and an after-pulsing in a real APD [7]. The mismatch leads to the presence of non-zero correlations between the adjacent time intervals in the sequence, which are effectively suppressed by a modulo-like hash function

— randomness extractor:

$$H(\tau) = \begin{cases} \emptyset, & \tau < s, \\ (\tau - s) \bmod m, & \tau \geq s, (\tau - s) \bmod (2m) < m, \\ m - 1 - (\tau - s) \bmod m, & \tau \geq s, (\tau - s) \bmod (2m) \geq m, \end{cases} \quad (1)$$

If a time interval τ is shorter than s, it is rejected to weaken the influence of an after-pulsing effect; otherwise the interval is fed through a non-monotonic hash function. The value of $T \cdot s$ should be greater than APD dead-time and after-pulsing period; m should be much less than mean click period $\langle \tau \rangle$. The particular form of the extractor was chosen for reasons of practical implementation simplicity and satisfies 3 aims:

- to map semi-infinite set of time intervals to a finite number of letters in the output alphabet,
- to extract randomness from the initial sequence, and
- to make the distribution of letters more uniform.

The extractor (1) doesn't equalize probabilities of letters m perfectly, but still increases entropy to made the sequential algorithm more efficient.

We consider letter sequences to be non-correlated and stationary. To equalize outcome probabilities we use the Elias algorithm for $m \geq 2$ letters in $n \geq 2$ positions [6]. The letter sequence is cut into blocks or words of m elements each by the GRP. The words are then divided into classes, where the same letters appear in different orders, so all words in each class obviously have the same probability. By n umbering words in each class we obtain equiprobable output values. Our knowledge of the word number and the total amount of words in a class it belongs to, allows us to convert the sequence of words into the output binary stream.

To avoid significant real-time computations, our prototype implements the Elias algorithm via a pre-calculated look-up table (LUT) held in a 2 MiByte FLASH ROM. The parameters $s = 8$, $m = 4$ and $n = 10$ have been chosen to yield the average value of 1.0 output bit per APD click.

The output of the QRNG was tested by NIST statistical test suite [9] with the parameter $\alpha = 0.01$, upon 1000 streams of 10^6 bits. The results are

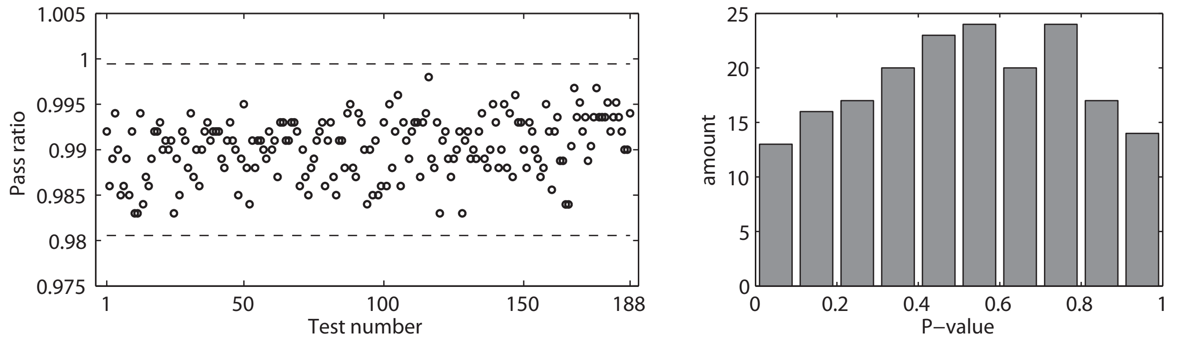

Figure 2: NIST statistical test suite results. On the left: portion of sequences passing a test, confidence interval is marked by dash lines. On the right: distribution of $P-values$.

presented in fig. 2. All samples of the sequence passing the tests are inside the confidence interval, while the $P - value$ distribution has parameters $\chi^2 = 7.7$; $Pvalue_T = 0.68 > 0.0001$, which suggests that the generated sequence is indistinguishable from a truly random one by the particular tests. The broad scope of the NIST suite and clear implemented QRNG operation principles guarantee that the generated random data are of high quality and may be used in critical applications.

References

[1] Ekert A.K. Quantum cryptography based on Bells theorem. — Phys. Rev. Lett. vol. 67, p. 661 (1991).

[2] Jennewein T. et al. A fast and compact quantum random number generator. — Rev. Sci. Instrum., vol. 71, pp. 1675-1680 (2000)

[3] Wayne M.A. et al. Photon arrival time quantum random number generation. — J. Mod. Opt., vol. 56, pp. 516-522 (2009)

[4] Wahl M. et al. An ultrafast quantum random number generator with provably bounded output bias based on photon arrival time measurements. — Appl. Phys. Lett., vol. 98, p. 171105 (2011)

[5] Wayne M.A., Kwiat P.G. Low-bias high-speed quantum random number generator via shaped optical pulses. — Opt. Express, vol. 18, pp. 9351-9357 (2010)

[6] A. Juels et at. How to turn loaded dice into fair coins. — IEEETIT: IEEE transactions on information theory, vol. 46, pp. 911-921 (2000)

[7] Cova S. et al. Avalanche photodiodes and quenching circuits for single-photon detection. — Appl. Opt., vol. 35, pp. 1956-1976 (1996)

[8] Mandel L. Fluctuations of photon beams: The distribution of photo-electrons. — Proc. Phys. Soc., vol. 74, pp. 233-242 (1959)

[9] NIST statistical test suite — http://csrc.nist.gov/groups/ST/toolkit/rng/index.html.

Minimizing Collisions for Quantum Hashing

Alexander Vasiliev Mansur Ziatdinov

Abstract

We present explicit algorithms for computation of quantum hashing parameters that minimize the probability of encountering quantum collisions.

Keywords: quantum computation, quantum hashing, hashing collisions, genetic algorithm, simulated annealing.

1 Introduction

Hashing is a well-known technique, widely used in computer science. Following the ideas and properties of the cryptographic hashing [1] we have proposed its quantum analogue in [2]. Just like in classical case it can find applications in different communication scenarios including single-bit quantum digital signature protocol from [3] and quantum communication protocols (e.g. in one-way quantum communication model and simultaneous message passing model [4]).

The key property of both classical and quantum hashing is the collision resistance. In [2] we have analyzed the set of numeric parameters for quantum hashing that determine its collision resistance. In this paper we investigate the construction of that set in more detail. Although there was a general method of obtaining good hashing parameters, it makes sense for comparatively large inputs. That is why we construct different algorithms to complement the general one. In particular, we give two heuristic algorithm for this problem: a genetic approach and annealing simulation.

2 Preliminaries

In this section we recall a quantum hash function from [2].

Let $q = 2^n$ and $B = \{b_1, b_2, \ldots, b_d\} \subset \mathbb{Z}_q$. We define a quantum hash function $\psi_{q,B} : \{0,1\}^n \to (\mathcal{H}^2)^{\otimes(\log d+1)}$ as follows. For an input $x \in \{0,1\}^n$ we let

$$|\psi_{q,B}(x)\rangle = \frac{1}{\sqrt{d}} \sum_{i=1}^{d} |i\rangle \left(\cos \frac{2\pi b_i x}{q} |0\rangle + \sin \frac{2\pi b_i x}{q} |1\rangle \right) \quad . \tag{1}$$

It follows from this definition that the quantum hash $|\psi_{q,B}(x)\rangle$ of an n-bit string x consists of $\log d + 1$ qubits. We have shown that d can be of order $O(n)$ without loosing the quality of hashing [2].

In [2] we have discussed the notion of *quantum collision*. The reason why we have defined it is the observation that in quantum hashing there might be no collisions in the classical sense: since quantum hashes are quantum states they can store arbitrary amount of data and can be different for unequal messages. But the procedure of comparing those quantum states implies measurement, which can lead to collision-type errors.

So, a *quantum collision* is a situation when a procedure that tests an equality of quantum hashes outputs true, while hashes are different. This procedure can be a well-known SWAP-test (see for example [2] for more information and citations) or something that is adapted for specific hash function. Anyway, it deals with the notion of distinguishability of quantum states. And since non-orthogonal quantum states cannot be perfectly distinguished, we require them to be "nearly orthogonal".

The set $B = \{b_1, b_2, \ldots, b_d\}$ of hashing parameters not only determines the size of the hash but also gives the function $\psi_{q,B}$ an ability to withstand collisions, i.e. to distinguish different hashes with bounded error probability. We have called this property *δ-resistance*.

Formally, for $\delta \in (0,1)$ we call a function $\psi : \mathbb{X} \to (\mathcal{H}^2)^{\otimes s}$ δ-resistant if for any pair w, w' of different inputs

$$|\langle \psi(w) \,|\, \psi(w') \rangle| \leq \delta \quad . \tag{2}$$

The value of δ for the hash function $\psi_{q,B}$ entirely depends on q (which is fixed here by the size of the input) and the set B, i.e. $\delta = \delta(q, B)$. In

[2] we have shown a construction for the set of polylogarithmic size (in q) based on [5]. We have also proved the following result.

Theorem 1. *For arbitrary $\delta \in (0,1)$ there exists a set $B = \{b_1, b_2, \dots, b_d\}$ of size $d = \lceil (2/\delta^2)\ln(2q) \rceil$ such that the quantum hash function $\psi_{q,B}$ is δ-resistant.*

In other words, for arbitrary $\delta \in (0,1)$ it is possible construct a δ-resistant quantum hash function $\psi_{q,B}$ that would produce an $\log d + 1 = O(\log\log q) = O(\log n)$-qubit hash out of n-bit input.

3 Optimization problem

It can be easily seen that for the function $\psi_{q,B}(x)$ we have

$$|\langle \psi_{q,B}(w) \,|\, \psi_{q,B}(w') \rangle| = \left| \frac{1}{d} \sum_{i=1}^{d} \cos \frac{2\pi b_i (w - w')}{q} \right|,$$

and we want it to be less than some δ for any value of $(w - w')$ except for 0. Thus, the optimization problem that arouses here is the following.

For a fixed q minimize the target function

$$\delta(q, B) = \max_{x \neq 0} \left| \frac{1}{d} \sum_{i=1}^{d} \cos \frac{2\pi b_i x}{q} \right|$$

over all $B = \{b_1, \dots, b_d\} \subset \mathbb{Z}_q$.

The best possible solution exists for $B = \mathbb{Z}_q$, since $\delta(q, \mathbb{Z}_q) = 0$. However, this would mean that the size of the hash is $\log q + 1 = n + 1$, i.e. even larger than the input, and hashing looses one of its important properties. So, we require that $d \ll q$, and we actually solve the above optimization problem several times for increasing d until it gives us the set B with desired value of $\delta(q, B)$.

4 Genetic Algorithm

The idea of genetic algorithms is described e.g. in [6]. Research in this area has started in 1954 and became widely spread in 1970s-1980s.

When applied to our optimization problem

- a phenotype is the set B sorted in ascending order;
- a fitness function is given by $\delta(q, B)$;
- a mutation is an increment or decrement of a random element of B;
- a crossover is performed by splitting sets in two parts and exchanging them.

To start the algorithm, we randomly generate a family of sets (a population). Then the population is evolved as following.

First of all, the population undergoes sudden mutations: we randomly pick several individuals and randomly "mutate" them, i.e. change the random element of a set by one.

Then for all individuals the value of the fitness function is evaluated. The half of all individuals with the best results give the next generation: we pick random pairs of phenotypes, their genotypes are split in two parts and exchanged in such a way that the values of the first parts are less or equal to the values of the second parts.

Finally, we remove the individuals with the worst fitness until the population has the initial size.

The evolution process repeats the given number of iterations or until a good enough solution is found. Thus, we need some value of δ as an input parameter.

5 Simulated Annealing

We also developed an simulated annealing algorithm to compute the set B. This algorithm is a heuristical search algorithm and it is described in [7]. We used `concurrent-sa` library for Haskell language for general procedure of simulated annealing.

Simulated Annealing is inspired by a physical process of melting some substance and then lowering the temperature slowly. This process allows the substance to get to optimal state (i.e. state with the lowest energy).

So we generate a population of random sets and allow them to evolve into the other (neighbour) states according to the current temperature. This temperature slowly decreases. After sufficient time population will have sets with low δ. To change a set we randomly change one element of the set.

We run simulated annealing for fixed time (1 sec) with population of 1000 random sets.

6 Acknowledgements

The work is performed according to the Russian Government Program of Competitive Growth of Kazan Federal University. Work was in part supported by the Russian Foundation for Basic Research (under the grant 14-07-00878).

References

[1] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal Roy and Willi Meier, editors, *Fast Software Encryption*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer Berlin Heidelberg, 2004.

[2] F M Ablayev and A V Vasiliev. Cryptographic quantum hashing. *Laser Physics Letters*, 11(2):025202, 2014.

[3] Daniel Gottesman and Isaac Chuang. Quantum digital signatures. Technical Report arXiv:quant-ph/0105032, Cornell University Library, Nov 2001.

[4] Alexander Vasiliev. Quantum communications based on quantum hashing. (arXiv:1312.1661), 2013.

[5] Alexander A. Razborov, Endre Szemeredi, and Avi Wigderson. Constructing small sets that are uniform in arithmetic progressions. *Combinatorics, Probability & Computing*, 2:513–518, 1993.

[6] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)*. Springer-Verlag, London, UK, 1996.

[7] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983.

Quantum Hashing Based on Symmetric Groups

Mansur Ziatdinov

Abstract

The notion of quantum hashing formalized by F. Ablayev and A. Vasiliev in 2013. F. Ablayev and M. Ablayev in 2014 introduced the notion of quantum hash generator which is convenient technical tool for constructing quantum hash functions. M. Ziatdinov in 2014 presented group approach for constructing quantum hash functions. All these mentioned above results present constructions of quantum hash functions based on abelian groups.

This paper continue the research on quantum hashing. Our approach allows to construct quantum hash function for any (finite) group. Also our approach allows to construct quantum hash functions based on classical hash function from NC^1.

Keywords: quantum hashing, quantum hashing on groups, symmetric groups

1 Introduction

H. Buhrman et al. [6] introduced the notion of quantum fingerprinting. Quantum fingerprinting is based on binary error correcting codes. Later F. Ablayev and A. Vasiliev in [3] proposed another (non binary) version of quantum fingerprinting. F. Ablayev and A. Vasiliev [4] defined notion of quantum hash-function and showed that quantum fingerprinting was a specific case of quantum hashing.

In [1] construction of Buhrman et al.[6] and Ablayev-Vasiliev's construction [4] were generalized. It was shown that both approaches could be viewed as composition of so called "quantum generator" and (classical) universal hash function.

In [7] we proposed a group approach to fingerprinting. We showed that instead of abelian group $\mathbb{Z}_m$ with $m > 0$ [4] we could use arbitrary abelian group. These constructions use specific so called "good" set of automorphisms. However, examples of such “good” sets (and, hence the quantum hash functions) were found only for abelian groups.

In this paper we propose “good” set of automorphisms for symmetric groups, and construct quantum hash function based on any finite group. This approach allows us to construct quantum hash functions based on classical functions from NC^1. We also discuss the procedure of finding “good” set of automorphisms.

2 Previous work

We start with recalling basic definitions.

We will consider functions $h : \{0,1\}^n \to G$, where G is a group.

Let us choose a set of automorphisms $\mathcal{K}$ from group of all automorphisms $\mathrm{Aut}(G)$:

$$k_i \in \mathcal{K} \subseteq \mathrm{Aut}(G), \qquad 1 \le i \le T, |\mathcal{K}| = T \tag{1}$$

We will use notation $k\{g\}$ for image of g under automorphism k.

Let us also choose a homomorphism f from group G to a group of all unitary transformations of m qubits.

Let us recall definitions and theorems from [4] and [7]

Quantum hash function is defined as follows.

Definition 1. $|\Psi(w)\rangle$ *is a quantum hash function if it maps n–bit message w from $\{0,1\}^n$ to m qubits and resulting vectors are nearly orthogonal: $\forall w, w' \in \{0,1\}^n (|\langle\Psi(w)|\Psi(w')\rangle| < \epsilon)$ for some $\epsilon \in (0,1)$.*

We call set K_{good} of elements of chosen $\mathcal{K}$ “good” set if for each non-unit group element g and some starting state $|\psi_0\rangle$:

$$\forall g \in G, g \neq e : \frac{1}{|K_{\text{good}}|^2} \left| \sum_{k \in K_{\text{good}}} \langle\psi_0|f(k\{g\})|\psi_0\rangle \right|^2 < \epsilon \tag{2}$$

In [7] it was proved that

Theorem 1. *If (3) holds, then "good" set exists and can be constructed by choosing d times element from $\mathcal{K}$ at random, and $d = \frac{2}{\epsilon}\ln|G|$*

$$\forall g \in G, g \neq e : \frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}}\langle\psi_0|f(k\{g\})|\psi_0\rangle = 0, \tag{3}$$

so, if (3) holds, there exists quantum hash function for arbitrary small ϵ (however, "good" set size d and therefore qubit count m will grow)

We will say "quantum hash function works for group G" or simply "quantum hash function for group G" if it has the form

$$|\Psi_{h,G,K,f,m,|\Psi_0\rangle}(x)\rangle = \frac{1}{\sqrt{t}}\sum_{j=0}^{t-1}\Big(|j\rangle \otimes f\big(k_j\{h(x)\}\big)|\psi_0\rangle\Big), \tag{4}$$

where h is a classic hash function mapping X^n to group G, $K = \{k_0, \ldots, k_{t-1}\}$ is "good" set of automorphisms and f is homomorphism from G to space $[(\mathcal{H}^2)^{\otimes m} \to (\mathcal{H}^2)^{\otimes m}]$.

It was also proven that

Theorem 2. *If for group G "good" set of automorphisms exists, then a quantum hash function for group G exists.*

3 Quantum hash function working on symmetric group

Theorem 3. *There exists a quantum hash function $|\Psi_{h,S_n,K,f,\log n}\rangle$ working on symmetric group.*

Specifically, f is a standard symmetric group representation in a space of n dimensions and K is a set of all automorphisms acting by conjugation to cyclic shift.

Proof. Theorems 1 and 2 state that if there exists a homomorphism f, a set $\mathcal{K}$ of automorphisms of G such that

$$\frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}}\langle\psi_0|f(k\{g\})|\psi_0\rangle = 0, \tag{5}$$

then $\Psi_{h,G,K,f,m}$ is a quantum hash function.

In our case, f is a standard symmetric group representation in a space of n dimensions with group S_n acting by coordinates permutation.

Let $\mathcal{K}$ be the set of all (inner) automorphisms that has the form:

$$\mathcal{K} = \{g_\sigma : \sigma \text{ is a cyclic shift}\}, \quad g_\sigma(\tau) = \sigma\tau\sigma^{-1}\} \tag{6}$$

Let $|\psi_0\rangle$ be some vector $c_1|1\rangle + c_2|2\rangle + \ldots + c_n|n\rangle$, such that:

$$\sum_{i=1}^{n} c_i = 0 \tag{7}$$

Image of $|\psi_0\rangle$ under $f(g_\tau\{\sigma\})$ for any σ and $\tau \in \mathcal{K}$ is

$$f(g_\tau\{\sigma\}) = c_{\sigma(1+k)-k}|1\rangle + \ldots + c_{\sigma(n+k)-k}|n\rangle, \tag{8}$$

where τ is a cyclic shift to k and addition and subtraction in indices are modulo n.

So, if we sum this for all automorphisms $\tau \in \mathcal{K}$ we get:

$$\sum_{g_\tau \in \mathcal{K}} \langle\psi_0|f(g_\tau\{\sigma\})|\psi_0\rangle = \sum_{k=0}^{n}\sum_{i=0}^{n} c_i c_{\sigma(i+k)-k} = \sum_{i=0}^{n} c_i \sum_{k=0}^{n} c_{\sigma(i+k)-k}. \tag{9}$$

We substituted $f(g_\tau\{\sigma\})|\psi_0\rangle$ with its value from 8.

We can observe that $\sigma(i+k) - k$ runs over all integers from 1 to n. So we can rewrite as follows:

$$\sum_{g_\tau \in \mathcal{K}} \langle\psi_0|f(g_\tau\{\sigma\})|\psi_0\rangle = \sum_{i=0}^{n} c_i \sum_{j=0}^{n} c_j = 0. \tag{10}$$

We use equation (9) and definition (7) of ψ_0.

Equation (10) is equivalent to (5), so theorems 1 and 2 can be applied, and a quantum hash function for S_n exists. □

Please note that this proof does not apply to A_5 representation from paper [2] and we cannot use their representation and approach of this article to define quantum hash functions based on NC^1 functions. In the section 4 we use another representation.

In [7] it was shown that if we find a set $\mathcal{K}$ satisfying equation (2), we can construct a "good" set with probability of $\frac{1}{|G|}$ by repeatedly ($d = \frac{2}{\epsilon}\ln|G|$ times) randomly choosing elements from $\mathcal{K}$.

4 Applications

We can use the defined quantum hash function working on symmetric group to construct other quantum hash functions. One way of such construction is defined in [7]: we construct a hash function working on (direct) product of groups. We present another way.

Lemma 1. *Let G be a finite group, group $G' \lhd G$ be its subgroup, and $|\Psi_{h,G,K,f,m}\rangle$ be a quantum hash function working on it.*

Then we can define a quantum hash function working on G'.

Proof. We can define h' to be a restriction of h on G'.

Then $|\Psi_{h',G',K,f,m}\rangle$ is a quantum hash function.

Let us consider square of scalar product of quantum hash function values on different inputs.

$$|\langle\Psi_{h',G',K,f,m}(x)|\Psi_{h',G',K,f,m}(x)\rangle|^2 = \left|\langle\Psi_{h,G,K,f,m(x)}|\Psi_{h,G,K,f,m}(x)\rangle\right|^2 < \epsilon$$

We use that $G' \lhd G$ and that h' is a restriction of h on G'. □

Of course, such way is inefficient for small finite subgroups of S_n, but it works for non-abelian groups.

We can use our approach to construct quantum hash functions based on classical hash functions in NC^1.

Let h be a hash function that can be computed by NC^1 circuit. We can now use theorem 3 to obtain a quantum hash function based on it as follows.

We can convert circuit to width–5 polynomial–size branching program and represent it as permutation branching program [5]. Then we compute quantum hash function based on h as follows. For each input symbol we simultaneously apply required permutation in all subspaces (under different automorphisms as described in theorem 3).

References

[1] F. Ablayev, M. Ablayev. Quantum Hashing via Classical ϵ-universal Hashing Constructions. arXiv:1404.1503v2 [quant-ph] 2014

[2] F. Ablayev, C. Moore, and C. Pollett. Quantum and stochastic branching programs of bounded width, *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP), 2002.* arXiv:quant-ph/0201139.

[3] F. Ablayev, A. Vasiliev. Algorithms for quantum branching programs based on fingerprinting. *Electronic Proceedings in Theoretical Computer Science* 9: 1–11, 2009.

[4] F. Ablayev, A. Vasiliev. Cryptographic quantum hashing. *Laser Physics Letters* 11.2, 2014.

[5] D. A. M. Barrington. Bounded-width polynomial-size branching programs can recognize exactly those languages in NC_1, *Journal of Computer and System Sciences* 38:150-164, 1989.

[6] H. Buhrman, R. Cleve, J. Watrous, R. de Wolf. Quantum fingerprinting. *Physical Review Letters*, 87(16), 2001.

[7] M. Ziatdinov. Quantum hashing. Group approach. http://shelly.kpfu.ru/e-ksu/docs/F1221792420/hash_8_ljm_en_draft.pdf

On Implementation Method Of Large Size Linear Transformation

Nikolay Borisenko Van Long Nguyen

Abstract

The article describes the implementation methods of a large size linear transformation over a non-prime finite field $GF((2^n)^m)$, using a modified linear feedback shift registers (MLFSR). It discusses key features and techniques used in software implementation of the transformation on various platforms.

Keywords: LFSR, Linear mapping, Composite field, Implementation

1 Preliminaries

A linear transformation is one of the most important elements of the cipher that provides its diffusion properties. It is known that a linear transformation is the slowest part of the cipher. Therefore, the effective implementation problem of the linear transformation is of great importance.

Let V_s be a vector space of dimension s. We denote a linear transformation by $L : V_s \mapsto V_s$. This transformation is defined by Galois (Fig. 1 – a) or Fibonacci (Fig.1 – b) linear feedback shift register (LFSR) over a non-prime finite field $GF((2^n)^m)$ [2], where $s = mxn$, that is based on a primitive polynomial $f(x) = x^n \oplus \sum\limits_{i=0}^{n} a_i x^i$ and an irreducible polynomial $h(y) = y^m \oplus \sum\limits_{i=0}^{m-1} h_i y^i$, where $a_i \in GF(2)$, $h_i \in GF(2^n)$ and $h_0 = 1$.

Let $X = (q_{m-1}, q_{m-2}, ..., q_2, q_1, q_0)$ and $Y = (q'_{m-1}, q'_{m-2}, .., q'_2, q'_1, q'_0)$ be initial LFSR state vector and output state vector after the first step of the register, correspondingly, where $q_i, q'_i \in GF(2^n), i = 0, 1, ..., m-1$ are LFSR cell values.

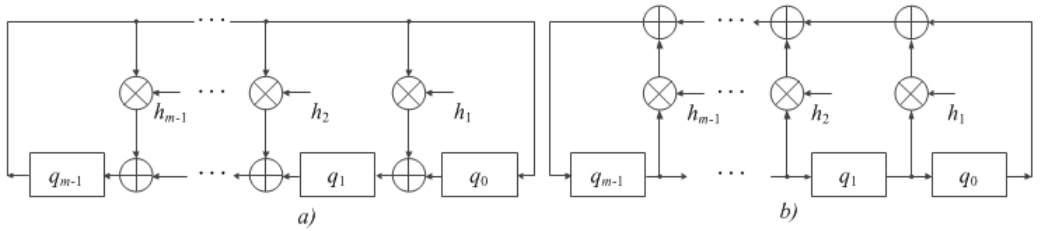

Figure 1. LFSR: a) - Galois and b) - Fibonacci

In the case of Galois LFSR each $q_i^{,}$ is defined by $q_i^{,} = h_i \cdot q_{m-1} \oplus q_{i-1}$ for all $i = m-1, ..., 2, 1$, and $q_0^{,} = h_0 \cdot q_{m-1}$. And in the case of Fibonacci LFSR each $q_i^{,}$ is defined by $q_i^{,} = q_{i+1}$ for all $i = 0, 1, ..., m-2$, and $q_{m-1}^{,} = \sum_{0}^{m-1} h_i \cdot q_i$. The operations "$\cdot$" (or "$\otimes$") and "$\oplus$" (or "$\sum$") are multiplication and addition in a finite field $GF(2^n)$, correspondingly.

Linear transformation of the initial data block is obtained after m steps of the LFSR. The transformation result is a new state of the register after m^{th} step. Inverse linear transformation L^{-1} is obtained by m steps of the LFSR that moves in the inverse direction.

Let $p_0, p_1, ..., p_d$, where $p_0 < p_1 < ... < p_d$, be all divisors of m. We denote $k = p_i, R = \dfrac{m}{k}$ and $W = nk$. Here the value W depends on the platform (8, 16, 32 or 64-bit), on which the given linear transformation is implemented.

In this article we will continue to investigate the effectiveness of a linear transformation implementation on arbitrary platforms by a method of simultaneous applications of internal and external polynomials of a non-prime finite field. This method was presented by us in [3]. It proved its effectiveness not only in software, but in hardware implementations as well. In addition, it allows a parallel implementation on multi-core processors.

2 Modified LFSR scheme

Modified Galois and Fibonacci LFSRs (MLFSRs) are presented in figures 2 and 3, correspondingly.

The main difference of MLFSR from LFSR is in the calculating method of

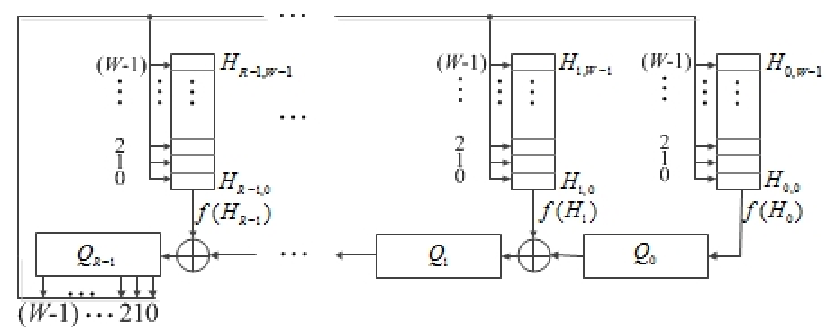

Figure 2. Galois MLFSR

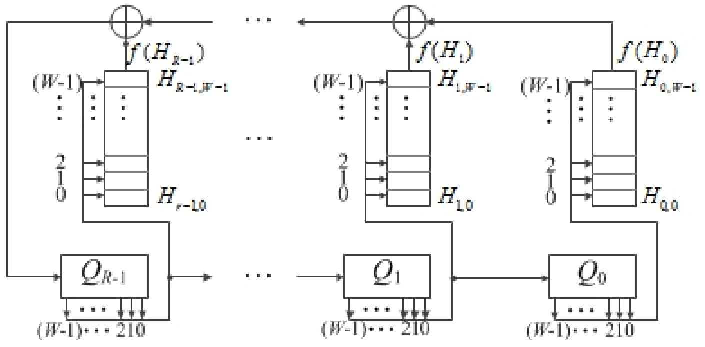

Figure 3. Fibonacci MLFSR

the feedback function values. In case of Galois MLFSR the feedback function values are determined via tabular data depending on the bits of the highest register cell, and in case of Fibonacci MLFSR by bits of all cells.

Let vector $X' = (Q_{R-1}, ..., Q_1, Q_0)$, where $Q_r = q_{kr+1} \parallel \cdots \parallel q_{kr-k+1}$, $0 \leqslant r \leqslant R-1$ is the initial MLFSR state vector, and vector $Y' = (Q'_{R-1}, ..., Q'_1, Q'_0)$ is the output state vector after the first step of MLFSR. The description of one MLFSRs step is presented in table 1.

Table 3. One step of MLFSRs

Galois	**Fibonacci**
$Q'_r = f(H_r) \oplus Q_{r-1}$, $\forall r = R-1, ..., 1$ and $Q'_0 = f(H_0)$, where $f(H_r) = = \sum_{j=0}^{W-1} z_{R-1,j} \cdot H_{r,j}$, and $z_{R-1,j}$ $\in GF(2)$ are bits of Q_{R-1}, $j = 0, 1, ..., W-1$, $r = R-1, ..., 1, 0$	$Q'_r = Q_{r+1}$, $\forall r = 0, ..., R-2$ and $Q'_{R-1} = \sum_{r=0}^{R-1} f(H_r)$, where $f(H_r) = = \sum_{j=0}^{W-1} z_{r,j} \cdot H_{r,j}$, and $z_{r,j} \in GF(2)$ are bits of Q_r, $j = 0, 1, ..., W-1$, $r = R-1, ..., 1, 0$

Table 2. Complexity of R MLFSR steps

Parameter	**MLFSR**	
	Galois	**Fibonacci**
Check number *"true-false"*	$m \cdot n$	$\frac{m^2 \cdot n}{k}$
XOR number	$\frac{m(mn-1)}{k}$	$\frac{m(mn-1)}{k}$
Required memory (bit)	$m \cdot n^2 \cdot k$	$m \cdot n^2 \cdot k$

To generate the correct output vector of each MLFSR (Galois or Fibonacci) it is necessary to define R tables $H_r, r = (R-1), ..., 0$. The new obtained state vector of MLFSR on the R^{th} step is the result of the linear transformation L. The complexity of R MLFSR steps is presented in Table 2. The table values evaluation is based on the superposition principle of linear mappings, according to which it is necessary to observe the bit influence of the corresponding LFSR (Galois or Fibonacci) cells on their output state after k steps. Finally, to compute the table values of feedback functions $f(H_r), r = (R-1), ..., 0$ we construct simple algorithms presented in Figures 4 and 5.

The principle of computing the required tables for inverse linear transformation L^{-1} is similar. But it is necessary to use the LFSRs (Galois or Fibonacci) scheme for inverse linear transformation L^{-1} to determine their state (block 7 in Fig. 4 or block 9 in Fig. 5). Finally, the obtained MLFSR moves in the inverse direction.

3 Experimental results

In this section the polynomials $f(x)$ and $h(y)$ are used as the initial polynomials of linear transformation $L : V_{128} \mapsto V_{128}$ over $GF((2^8)^{16})$:

$$f(x) = x^8 \oplus x^7 \oplus x^6 \oplus x \oplus 1. \tag{1}$$

$$h(y) = y^{16} \oplus 148y^{15} \oplus 32y^{14} \oplus 133y^{13} \oplus 16y^{12} \oplus 194y^{11} \oplus 192y^{10} \oplus y^9 \oplus \\ \oplus 251y^8 \oplus y^7 \oplus 192y^6 \oplus 194y^5 \oplus 16y^4 \oplus 133y^3 \oplus 32y^2 \oplus 148y \oplus 1. \quad (2)$$

These polynomials are applied to construct the linear transformation in Kuznyechik algorithm based on Fibonacci LFSR [4]. But in this section we will construct the linear transformation that is based on both types of LFSR (Galois and Fibonacci). Consequently, we obtained two linear transformations, each having a branch number equal to 17 (this is maximum value for the 128-bit data block using 16 8-bit S-boxes) and fixed points number, which is equal to 1 [6].

By applying our approaches we constructed MLFSR schemes (Fig. 6 and Fig. 7) allowing implementation of $L : V_{128} \mapsto V_{128}$ mapping on 8-bit processors. In this case we choose $k = p_0 = 1$ that is the first divisor of $m = 16$, $R = \dfrac{m}{k}$ and $W = mn = 8$.

It is not difficult to see that in implementation schemes on 8-bit processors, table values of both types of MLFSRs (Galois and Fibonacci) are identical. Their values are determined by the algorithms in Fig. 4, 5 and shown in table 3.

Table 3. 16 tables of the 8-bit Galois and Fibonacci MLFSR

No	H_{15}	H_{14}	H_{13}	H_{12}	H_{11}	H_{10}	H_9	H_8	H_7	H_6	H_5	H_4	H_3	H_2	H_1	H_0
7	E5	6D	B2	D7	6E	AD	80	DE	80	AD	6E	D7	B2	6D	E5	80
6	93	D7	59	8A	37	B7	40	6F	40	B7	37	8A	59	D7	93	40
5	A8	8A	CD	45	FA	BA	20	D6	20	BA	FA	45	CD	8A	A8	20
4	54	45	87	C3	7D	5D	10	6B	10	5D	7D	C3	87	45	54	10
3	2A	C3	A2	80	DF	CF	08	D4	08	CF	DF	80	A2	C3	2A	08
2	15	80	51	40	8E	86	04	6A	04	86	8E	40	51	80	15	04
1	EB	40	C9	20	47	43	02	35	02	43	47	20	C9	40	EB	02
0	94	20	85	10	C2	C0	01	FB	01	C0	C2	10	85	20	94	01

Due to the symmetry of the output polynomial $h(y)$ and the equality $h_0 = h_7 = h_9 = 1$, one can use only 7 tables (H_1, H_2, H_3, H_4, H_5, H_6, H_8) instead of 16. Therefore, the amount of required memory is reduced and is equal to $7 \times 8 = 56$ bytes. In this case we obtaine simplified MLFSR schemes (Fig. 8 and Fig. 9).

Similarly, in cases when $k = p_i = 2, 4$ and 8, with the algorithm shown in Fig.4 and 5 to determine all tables $H_r, r = 0, 1, ..., R-1$, we constructed MLFSR schemes which allow implementation of the given linear transformation on 16, 32 and 64-bit processors, correspondingly. An examples of such implementation on 32-bit processors for Galois and Fibonacci MLFSR are shown in Fig. 10.

Depending on the specification of Galois MLFSR we developed a method called *"Ready Linear Combinations"* method (further, the RLC method). This implementation approach is effective only for Galois MLFSR. The advantage of Galois MLFSR over Fibonacci MLFSR is that on each step, each check operation on *"true-false"* for all bits of the highest byte of Galois MLFSR state may be used for all tables of feedback function. This allows combining 16 tables $H_r, r = 0, 1, ..., R-1$ after calculating all possible linear combinations for that highest byte. The obtained implementation scheme is shown in Fig. 11.

In this method, for each byte of memory cells 1 and 2, all variants of linear combinations are pre-computed using 8-bit Galois MLFSR (Fig. 6) and stored in the corresponding memory cells. And the memory cells are structured so that the address 0x01 is the first, and the address 0xFF includes 255^{th} row of the both matrices. The given linear transformation requires 16 shift operations of 16 byte matrix, in average 32 calls to the memory and the same number of 64-bit words additions modulo two. It requires 4096 bytes of memory. Performance of this method is several times higher than that of the previous ones, but it requires memory of 4096 bytes, against 144 bytes.

We implemented the described methods in software. The results are presented in Tables 4 and 5, where "G" is a Galois MLFSR and "F" is a Fibonacci MLFSR. Then we applied the implementations to block cipher Kuznyechik [5]. At the same time its linear transformation was implemented by both types of LFSR: Galois and Fibonacci. The block cipher was running in ECB mode of operation. The obtained results are presented in table 6. We also presented the known implementation results for Kuznetik [5] there. The source codes are written in the C++ programming language. The codes were compiled in Visual Studio 2010 environment for x64/Intel platform. The measurements are performed on a single core of Intel Core i5-2500 @

3.2GHz in Windows 8 OS. The measured speed is given both in Megabytes per second and clock cycles per byte.

Table 4. Parameters of described methods

Platform size (bit)	Cycles number of MLFSR	Required memory (byte)		Check number "true-false"		XOR number		Time of 10^6 operations (sec.)	
		G	F	G	F	G	F	G	F
8	16	56	56	128	896	1024	1024	1.03	2.35
16	8	256	256	128	1024	1016	1016	0.60	3.50
32	4	512	512	128	512	508	508	0.27	1.34
64	2	1024	1024	128	256	254	254	0.20	1.11

Table 5. Parameters of RLC method

Platform size (bit)	Cycles number of	Required memory (byte)	XOR number	Time of 10^6 operations (sec.)
64	16	4096	32	0.058

Table 6. Encryption performance of block cipher Kuznyechik

Implementation	Memory (bytes)	Speed		Platform
		MB/s	cpb	
64-bit Galois MLFSR	1024	10.5	305	i5-2500 @ 3.2GHz, Win 8
64-bit Fibonacci MLFSR	1024	8.25	388	i5-2500 @ 3.2GHz, Win 8
RLC method	4096	77	42	i5-2500 @ 3.2GHz, Win 8
Borodin [5]	65536	125	26	i7-2600 @ 3.4GHz, Win7

4 Conslusion

The described implementation methods using MLFSR are bit-oriented and allow reducing the amount of required operations and memory size to store pre-computed tables. The theoretical and experimental results show that the linear transformation, built on the Galois LFSR has advantages over the Fibonacci LFSR.

The RLC method has encryption speed slower than the method described in [5], because it does not allow combining both nonlinear (S-boxes) and linear transformations of block cipher Kuznyechik simultaneously. But the required memory size is equal to 4096 bytes, against 65536 bytes.

References

[1] *Christof P.* Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields, Ph.D Thesis //, Dissertation, Institute for Experimental Mathematics, Universität Essen, Germany: 1994. — 120 p.

[2] *Зензин О.С., Иванов М.А.* Стандарт криптографической защиты AES. Конечные поля, Под ред. М. А. Иванова — М.: КУДИЦ—ОБРАЗ, 2002. — 175 с. (*Zenzin O.S., Ivanov M.A.* AES cryptographic standard. Finite fields. (in Russian))

[3] *Borisenko N., Nguyen V. L., Bulygins A.* Developing Algorithm for Software and Hardware Implementation of Large Size Linear Mapping // CTCrypt 2013, Ekaterinburg, Russia. — 2013. — p. 192–204.

[4] *Shishkin V. et al.* Low-Weight and Hi-End Draft Russian Encryption Standard. // CTCrypt2014, Moscow, Russia. — 2014 — p. 183–188.

[5] *Borodin M. et al.* High-Speed Software Implementation of the Prospective 128-bit Block Cipher and Streebog Hash-Function. // CTCrypt 2014, Moscow, Russia. — 2014 — p. 189–197.

[6] *Muhammad R. Z.* Analysis of Linear Relationship in Block Ciphers // Bachelor of science (Computer). Universiti Teknologi Malaysia: 2003. — 256 p.

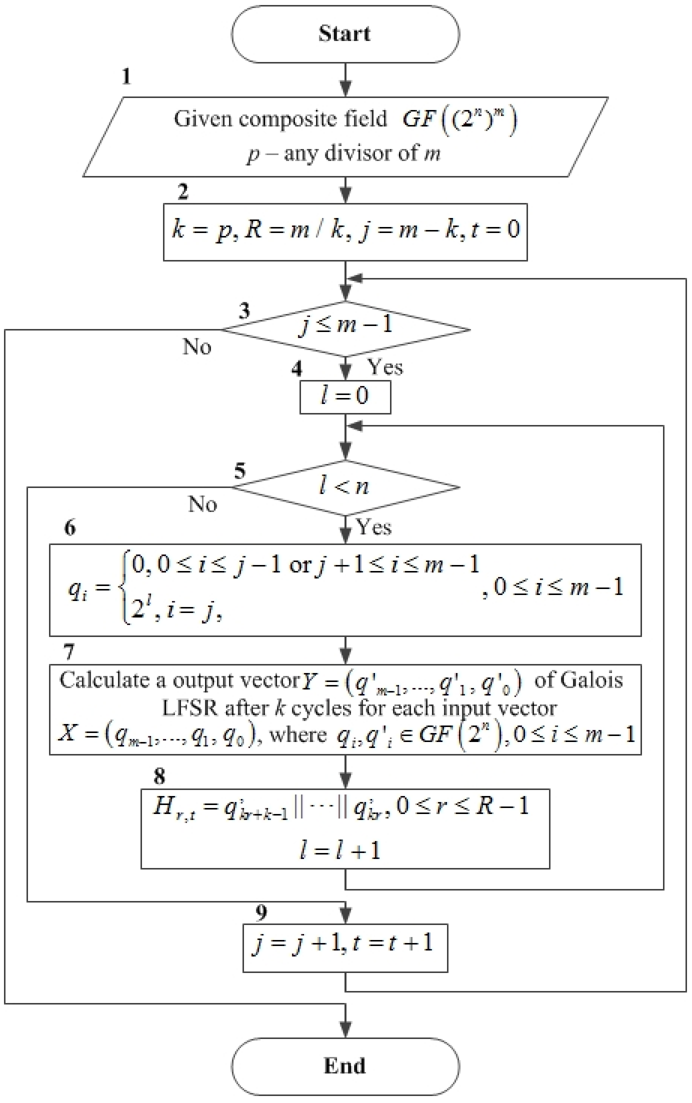
Start
1
Given composite field $GF((2^n)^m)$
p – any divisor of m
2
$k = p, R = m / k, j = m - k, t = 0$
3
$j \le m - 1$
No
Yes
4
$l = 0$
5
$l < n$
No
Yes
6
$q_i = \begin{cases} 0, 0 \le i \le j-1 \text{ or } j+1 \le i \le m-1 \\ 2^l, i = j, \end{cases}, 0 \le i \le m-1$
7
Calculate a output vector $Y = (q'_{m-1}, \ldots, q'_1, q'_0)$ of Galois LFSR after k cycles for each input vector $X = (q_{m-1}, \ldots, q_1, q_0)$, where $q_i, q'_i \in GF(2^n), 0 \le i \le m-1$
8
$H_{r,t} = q'_{kr+k-1} || \cdots || q'_{kr}, 0 \le r \le R-1$
$l = l + 1$
9
$j = j + 1, t = t + 1$
End

Figure 4. Algorithm for computing all R tables of Galois MLFSR

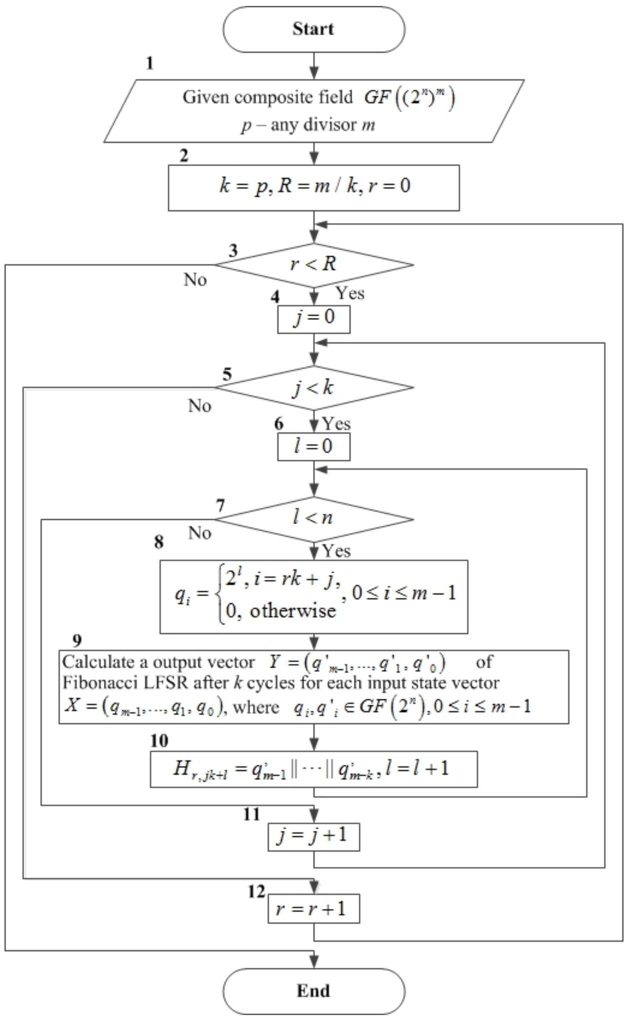
Start
1
Given composite field $GF((2^n)^m)$
p – any divisor m
2
$k = p, R = m / k, r = 0$
3
$r < R$
No
Yes
4
$j = 0$
5
$j < k$
No
6
Yes
$l = 0$
7
$l < n$
No
8
Yes
$q_i = \begin{cases} 2^l, i = rk + j, \\ 0, \text{otherwise} \end{cases}, 0 \le i \le m-1$
9
Calculate a output vector $Y = (q'_{m-1}, ..., q'_1, q'_0)$ of Fibonacci LFSR after k cycles for each input state vector $X = (q_{m-1}, ..., q_1, q_0)$, where $q_i, q'_i \in GF(2^n), 0 \le i \le m-1$
10
$H_{r, jk+l} = q'_{m-1} \| \cdots \| q'_{m-k}, l = l + 1$
11
$j = j + 1$
12
$r = r + 1$
End

Figure 5. Algorithm for computing all R tables of Fibonacci MLFSR

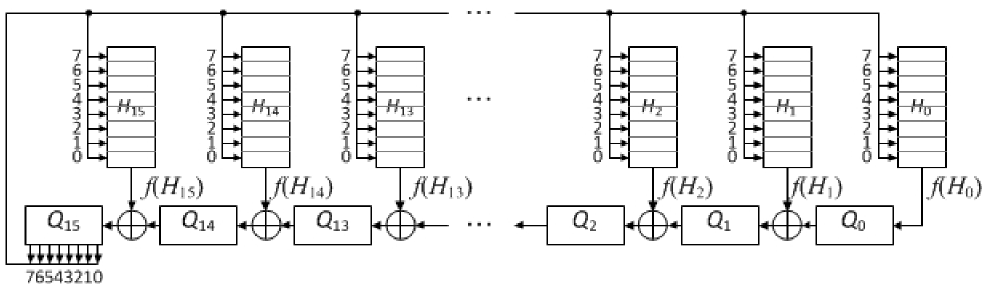

Figure 6. 8-bit Galois MLFSR

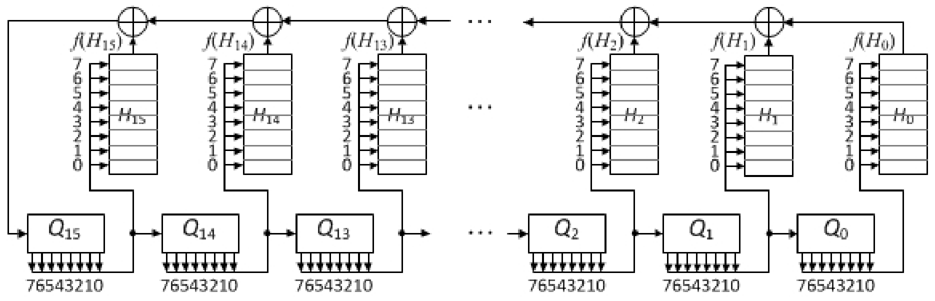

Figure 7. 8-bit Fibonacci MLFSR

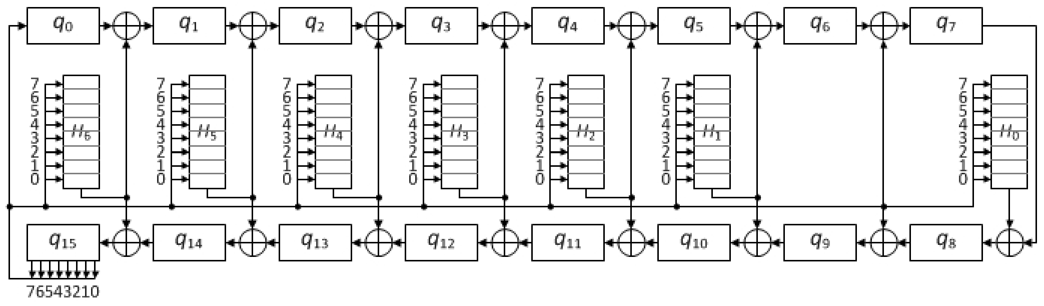

Figure 8. Simplified 8-bit Galois MLFSR

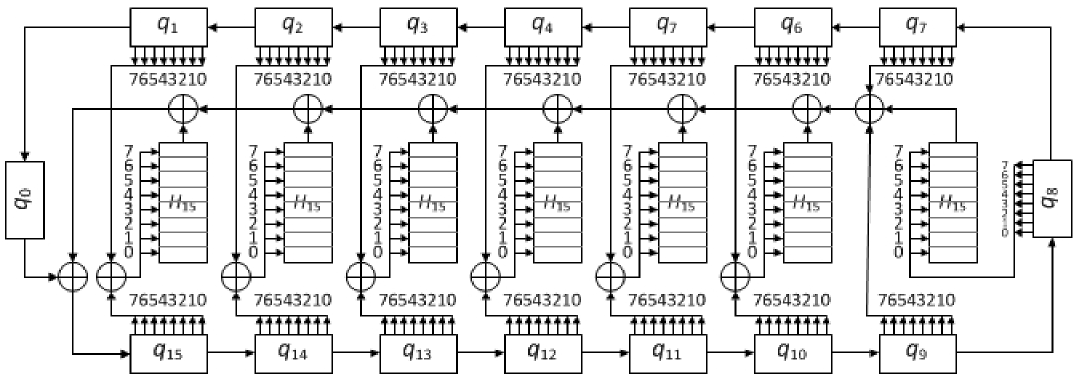

Figure 9. Simplified 8-bit Fibonacci MLFSR

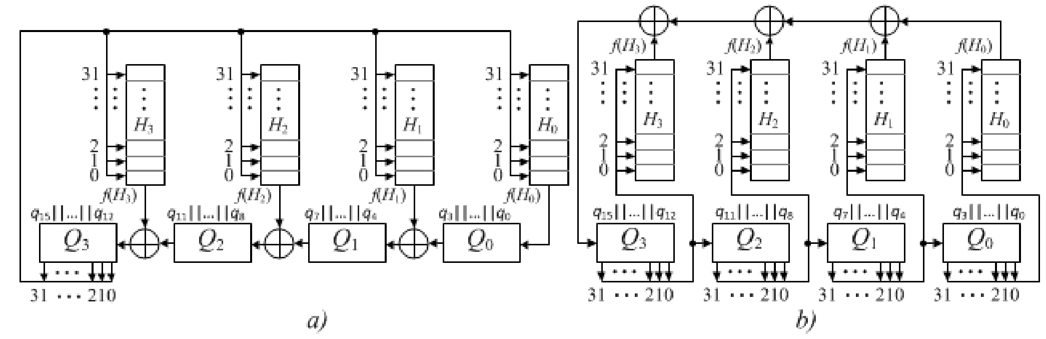

Figure 10. 32-bit MLFSR: a) - Galois and b) - Fibonacci

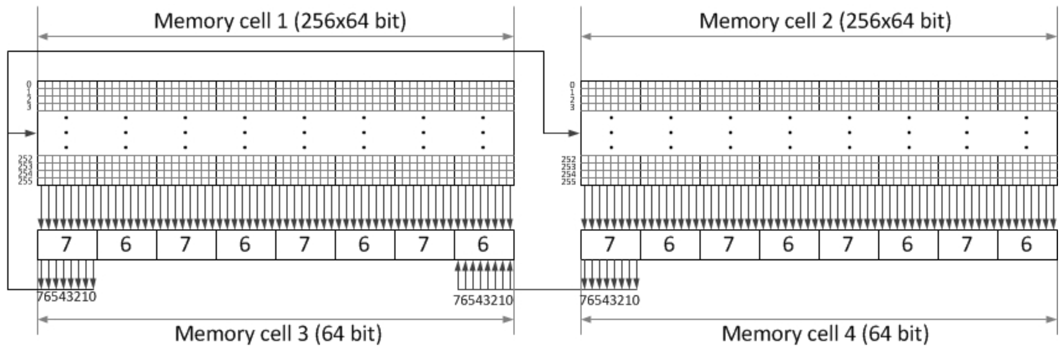

Figure 11. RLC method

Unifying Development and Implementation of Secure Network Protocols in C++

Pavel Lebedev

Abstract

In this paper we specify two main transition stages in the process of implementing a secure communication protocol that are prime sources of vulnerabilities. We describe requirements on the framework solution that would minimize their impact and streamline the development process. A C++ library that satisfies these conditions is described, which the author developed. We provide benchmarking results for this library that shows that it is possible to significantly reduce this attack surface while using domain-specific extensions to C++ to provide a natural way for cryptographers to describe secure communication protocols and immediately obtain optimized and scalable implementations.

Keywords: C++, coroutines, secure communication protocols

1 Introduction

Secure communication protocols are undeniably pervasive in modern networking. The current trend is to follow "secure by default" model and use cryptographic protocols on every remote connection made. TLS [1] is the most widespread protocol to negotiate and establish a secure communication channel which makes it a common attack target. These attacks focus on the protocol itself or its combinations with higher level protocols (see f.e. BREACH [2]) or particular implementations (ex.: Heartbleed [3]). TLS has been used in various applications, but no single protocol can cover for all technical and security requirements.

This shows that the research focused on new security protocols and improvements to the old ones is an urgent need. Such research has to go from a formal mathematic model to an optimized implementation. In this process we have to note two important transition phases:

1. *From mathematic model to implementation in a programming language.* The main problem in this transition is correctness of implementation. Modern cryptography requires not only high programming skills, but also fluency in various mathematical areas that are not common knowledge even for professional programmers. We propose to solve this problem by bringing the constructs of the programming language as close as possible to the mathematic description used by cryptographers.

2. *From a proof-of-concept to an optimized and production-ready implementation.* For the implementation to be suitable for production use, it must have adequate performance. First of all, this requires usage of optimized cryptographic primitives: ciphers, message authentication codes, arbitrary precision arithmetic, etc. There are various libraries available to fulfill this purpose, see f.e. `libgcrypt` [4] and `crypto++` [5]. An ability to switch the underlying implementation without changing the description of the protocol would give our solution additional flexibility.

   Furthermore, to exploit modern highly parallel and heterogeneous systems, implementations should be multi-threaded and use asynchronous programming interfaces to perform all input and output with network and storage devices. Efficient use of these APIs is non-trivial and usually requires modifications to the natural program flow, including the very definition of the protocol being implemented, that impact readability and introduce hard to spot bugs.

The attacks we've mentioned are but a few that were the results of vulnerabilities introduced during these transition phases, which we consider the main problems in the whole development and implementation scenario.

To remove these problematic stages there is a need for a toolkit that allows to describe a cryptographic protocol in the notation most close to the original mathematic one. The same description should also be usable without modifications as a kernel for a production system that supports modern high-performance programming approaches.

In this paper we describe a solution that satisfies these criteria based on a C++ library that includes a set of domain-specific extensions.

2 Protocol structure

Cryptographic protocols usually include one or both of the following phases:

- *Handshake.* This phase that consists of a short message exchange that are used to determine capabilities of the participants, algorithm parameters and whether they are acceptable to both parties. The set of cryptographic algorithms to use can be chosen here too if this is specified by the protocol.
- *Transport mode.* In this phase the negotiated transformations are applied to the data transferred to provide secure communication as specified by the protocol.

Various protocols make use of other ones as sub-protocols, therefore some of them may consist of only a single phase. For example, Diffie-Hellman key exchange only has a handshake phase — its result is a shared secret value and the protocol itself does not define how it is to be used. The solution we have developed allows for easy protocol composition to make it easy to build libraries of primitives and frameworks.

In our implementation the handshake phase is written as a sequence of actions that are either calculations or exchanges of data with a peer. This serial description is used regardless of chosen parallel computing model.

Transport mode characteristics are specified by declaring structure of the packet used to incapsulate data transferred over secure connection. It is used by our implementation to construct and parse transport mode packets containing user data from a byte stream.

3 Implementation details

To implement the required solution we have chosen to use the C++ programming language. It allows to achieve maximum implementation efficiency and allows direct access to many well-established libraries of mathematic and cryptographic libraries. Its meta-programming capabilities allow creation of domain-specific extensions that can be used to bridge the syntax gap between programming languages and mathematics. These facilities have been significantly extended in the last C++14 standard version and we're making full use of them in our solution.

We have developed the following facilities to allow secure communication protocol phases to be described in terms native to cryptographers:

1. *Cryptographic primitive type system.* The data types useful in cryptography are: boolean, byte arrays, integers of fixed and arbitrary precision, cryptographic algorithm keys and certificates, block ciphers, message authentication code and digital signature algorithms, elliptic curve and their points. High level primitives do not allow access to their representation to support changes to the primitives used without the need to change the protocol itself. We provide function and operator overloads for most common operations on these objects that follow mathematic instead of programming language syntax and semantics where possible. This also means we favor free over member functions to allow writing `size(x)` instead of `x.size()`.

2. *Tuples.* To preserve the mathematical notation of function as a mapping we cannot use reference parameters as output. We instead provide extended tuple facilities that allow treating multiple values as a single one, which can be used to return "multiple values" from a function. This facility is also used for (de)serialization as the tuple construction function (named _ for brevity) can be used to concatenate or split raw byte sequences or serial representations of objects. During this conversion it is possible to specify various options: explicit type sizes, byte order, etc. to conform to other implementations and their data formats.

3. *Communication functions.* In the description of handshake phase one can use functions to send, receive, or simultaneously send and receive data. Two versions of these functions are provided: for symmetric

protocols with the same algorithm for both peers (`send`, `receive`, `send_and_receive`) and asymmetric, which are also described jointly from the point of view of both peers, as is common in mathematical descriptions (`left->right`, `right->left`, `exchange`).

All these constructs use expression templates that avoid unwanted data copies and allow additional optimizations while preserving domain-specific syntax.

Asio [6] is used as base input/output library. It uses Proactor pattern that allows it to support both single- and multi-threaded environments without using separate threads for individual data streams as this approach doesn't scale in practice. The library is sufficiently well-known and tested and is planned to become a part of a future C++ language standard.

To solve the main difficulty in writing asynchronous code — splitting serial algorithm in multiple asynchronous callbacks — we're using coroutines [7] implementation from the boost.coroutine [8] library. It allows us to preserve a serial description of the algorithm without losing its asynchronous nature and performance, which is imperative to our stated goals. We've extended this facility to allow storage of coroutine-local data that is not dependent on the stack, which allows us to simplify syntax of many constructs by reducing the number of context-related parameters that can now be inferred and don't obscure the algorithm with implementation details.

An example description of the handshake phase for the protocol described in [9] is provided below:

```
void run(elliptic_curve e,
         point p,
         integer q,
         identifier ia,ib)
{
    bytes na,nb,s;
    integer ka,kb;
    point ra,rb,qa,qb;
    certificate ca,cb;
    bytes request,answer;
    bytes eta,etb;
    initial_vector iva,ivb;

    // Shared key generation, phase 1
    _(ka,ra) = kex(ec,p,q);

    na = random();
    send(na,ra,ca,sign(xa,_(na,ra,ia,ib)));

    // Phase 1 verification
    _(na,ra,ca,s) = receive();
    // Certificate validation
    if(not verify(ca))
        abort();
    // Point validation
    if(not verify(ra))
        abort();
    // Signature validation
    if(not verify(ca,s,_(na,ra,ia,ib)))
        abort();

    // Shared key generation, phase 2
    _(kb,rb) = kex(ec,p,q);

    // Phase 2 verification
    nb = random();
    send(nb,rb,cb,sign(xb,_(na,ra,ia,ib,nb,rb)));

    _(nb,rb,cb,s) = receive();
    if(!verify(cb))
        abort();
    if(!verify(rb))
        abort();
    if(!verify(cb,s,_(na,ra,ia,ib,nb,rb)))
```

```
        abort();

    // Mutual authentication

    qa = exp(ec,rb,ka);
    k = kdf(qa,na,nb);
    eta = encrypt(k,iva,request);
    iva = random();
    send(iva,eta,sign(xa,_(na,ra,ia,ib,nb,rb,iva,request)));

    qb = exp(ec,ra,kb);
    k = kdf(qb,na,nb);
    _(iva,eta,s) = receive();
    request = decrypt(k,iva,eta);
    if(not verify(ca,s,_(na,ra,ia,ib,nb,rb,iva,request)))
        abort();

    ivb = random();
    send(ivb,etb,sign(xb,_(na,ra,ia,ib,nb,rb,ivb,answer)));
    _(ivb,etb,s) = recieve();
    answer = decrypt(k,ivb,etb);
    if(not verify(cb,s,_(na,ra,ia,ib,nb,rb,ivb,answer)))
        abort();
}
```

The following example shows a declarative syntax for the common transport mode packet structure:

```
struct test_packet : packet
{
    // User data length, 64-bit unsigned integer, big endian
    chunk<uint64_be> s   = size(content);
    // User data padded with 1 and 0s and encrypted.
    chunk<bytes>     b   = encrypted(padded_01(content));
    // Authentication code for the previous packet fields.
    chunk<bytes>     mac = mac(_(s,b));
};
```

Other capabilities of the solution we've developed not shown in the examples are special sequence delimiters in transport mode packets instead of explicit length specifications and using multiple explicitly specified cryptographic primitives of the same type in a single algorithm. The system can be easily extended to support additional functionality.

We have shown that the developed solution solves the problem of representing mathematic protocol descriptions in a way that allows efficient implementations to be based on it which eliminates two transition phases we have stated are prime sources of vulnerabilities.

4 Benchmarking

Benchmarking of the system was performed in order to determine loss of performance incurred due to implementation details of the domain-specific language extensions.

The testing setup used is an Intel Core i7 920 based PC with 4 cores and 8 logical threads running two programs that implement client and server sides of the communication channel respectively running over the loopback interface using TCP. Both programs use a pool of 4 operating system threads to schedule asynchronous callbacks. Client application uses 64 simultaneous connections to the server, each running as a separate coroutine, that sends and receives 128 blocks of 64 KB before tearing down the connection and restarting. Server application spawns a separate coroutine for each client connection and uses it to echo incoming data back. perf utility was used to determine hot spots in the whole system.

In the first test case no security protocol was used to test behavior when I/O is the bottleneck. Total transfer speed in the system reached 15 Gbit/s without any special settings for a loopback interface. 96% percent of time was spent in kernel mode networking stack of the Linux operating system, which was running on the test machine. About a third of this time was used for the kernel-user space data copying and back, the rest was consumed by other parts of the OS network stack. Tests programs themselves did not contain a single data copy, so kernel is the only place where it happened. This is unavoidable while using the standard Linux networking API, although alternative solutions exist, for example [10]. Even in this scenario coroutine stack switches took no more than 0.5% of the total run time.

The second test case modeled a real secure communication scenario implemented using cryptographic primitives from libgcrypt library [4]. We used GOST 28147-89 block cipher in CFB mode and HMAC with 512-bit GOST 34.11-2012 hash function for the transport mode. Handshake consisted of mutual authentication by signing pseudorandom cookies with GOST 34.10-2012 digital signatures. In this case 97% of time was spent in cryptographic library routines and 2% in the OS kernel. The coroutine overhead was less than 0.01% of total time. Total transfer speed in the system approaches 250 MBit/s and reflects only the performance of cryptographic primitive implementation used with no measurable overhead due to the solution we've developed.

This proves that it is possible to achieve maximum performance limited only by external factors while using a natural representation for the secure network communication protocol while removing development stages that lead to deficiencies of implementation used as attack vectors. Author of this paper plans to improve upon this work by allowing GPUs to be used transparently in high performance secure communication systems.

References

[1] The Transport Layer Security (TLS) Protocol Version 1.3 (draft-ietf-tls-tls13-latest)

[2] *Angelo Prado, Neal Harris and Yoel Gluck*, SSL, gone in 30 seconds: A BREACH beyond CRIME

[3] CVE-2014-0160

[4] Libgcrypt, a general purpose cryptographic library

[5] Crypto++, a free C++ class library of cryptographic schemes

[6] asio C++ Library, a cross-platform C++ library for network and low-level I/O programming

[7] *M.E. Conway* Design of a Separable Transition-Diagram Compiler — Communications of the ACM (New York, NY, USA: Association for Computing Machinery) 6 (7): pp. 396–408

[8] boost.coroutine

[9] *A. Yu. Nesterenko* On an approach to the construction of secure connections — Mathematical Aspects of Cryptography, 2013, vol. 4, no. 2, pp. 101-111

[10] PF_RING ZC (Zero Copy) — Multi-10 Gbit RX/TX Packet Processing from Hosts and Virtual Machines

A Known Plaintext Attack on a Fully Homomorphic Cryptosystem Based on Factorization

Alina Trepacheva

Abstract

This paper presents a known plaintext attack on one recently proposed fully homomorphic cryptosystem based on factorization problem. We demonstrate that the cryptosystem is insecure even in the presence of only one pair of a plaintext and a corresponding ciphertext. The complexity of a proposed attack depends polynomially on a degree of polynomials representing ciphertexts and logarithmically on a plaintexts space size. The results of computer experiments are given.

Keywords: fully homomorphic encryption, known plaintext attack, factorization problem.

1 Introduction

Homomorphic cryptography is of great interest today. Homomorphic cryptosystem (HC) allows to compute some function $f(x_1, ..., x_t)$ over encrypted data set $\{c_1 = E(m_1), ..., c_t = E(m_t)\}$ without knowledge of a secret key sk. The owner of sk may extract the result of computation over plaintexts $f(m_1, ..., m_t)$ via decrypting ciphertext $c = f(c_1, ..., c_t)$. This property makes HCs an important tool for protecting private data in clouds [7]. Thin clients may rely on powerful cloud servers to conduct their computing tasks and don't worry about compromising the security of their data.

The most interesting problem in homomorphic cryptography is to design a *fully homomorphic cryptosystem* (FHC) [3] permitting to compute arbitrary f homomorphically with low time overhead. Until 2008 cryptographers were not aware whether secure FHC could be constructed. All suggested HCs were either not fully homomorphic or insecure [8]. Only in 2009 IBM researcher Craig Gentry developed the first FHC that is provably secure against known plaintext cryptanalysis [3]. The cryptosystem is based on ideal lattices framework. And breaking it is equivalent to finding the shortest vector in a lattice L. But although its overhead of homomorphic computation for any f depends polynomially on the cryptosystem's parameters, it is totally inefficient and inapplicable for practice. In subsequent works this FHC was improved in different ways to obtain better performance. The most efficient FHCs in Gentry style are described in [4]. But in spite of considerable efforts to obtain practical FHC following blueprint from [3], existing Gentry-like FHCs are still totally impractical. Due to [4] encryption via latest versions of FHCs converts 4 MB of plaintexts into 73 TB of ciphertexts. And this is not acceptable for real world applications.

Such state-of-the-art gave birth to many attempts to build some alternative FHC not on the basis of Gentry's blueprint [3]. And the bulk of efforts in this line is concentrated on designing FHCs based on factorization problem. This produced the simplest candidates for FHC with promising performance, for example [15, 6, 9, 14]. All this FHCs were not proven to be provably secure. Since the strong mathematical reductions to factorization problem were not given, their usage in practice is questionable.

In this paper we analyse one such factorization-based FHC [15] proposed in 2013 by Russian research group from Novosibirsk. The construction [15] is simple and efficient. And also the authors of [15] conjecture that their FHC is resistant to ciphertext only attack (COA) and known plaintext attack (KPA). Nevertheless, its security properties haven't been analysed thoroughly anywhere. In this work we fill the gap and show that FHC [15] is totally unsecure against KPA. Our attack is very similar to KPA [13] on Domingo-Ferrer HC [2, 5] that has common properties with FHC [15]. Compared with [13] we provide an accurate estimation of probability to find a key via this KPA. The main contribution of this work is that in order to recover a secret key for FHC [15] it's enough to intercept only one pair of a plaintext and a corresponding ciphertext. The running time of KPA depends polynomially on cryptosystem parameters.

Finally we'd like to note that cryptanalysis of FHC [15] is not only of theoretical interest, since it was mentioned in [10] that this scheme was used for implementation of a secure private data storage in cloud.

2 Preliminaries

2.1 Notations

A ring of integers modulo n is denoted by $\mathbb{Z}_n$, univariate polynomial ring over $\mathbb{Z}_n$ – by $\mathbb{Z}_n[x]$, the subset of $\mathbb{Z}_n[x]$ containing polynomials of degree d – by $\mathbb{P}_{n,d}$, the subset of $\mathbb{P}_{n,d}$ containing only monic polynomials – by $\mathbb{P}_{n,d,mon}$, the subset of $\mathbb{P}_{n,d,mon}$ containing only irreducible polynomials – by $\mathbb{I}_{n,d}$. If $a \in \mathbb{Z}$ then its residue modulo n is $[a]_n \in \mathbb{Z}_n$. Notation $[f(x)]_n$ means that all coefficients f_i are reduced modulo n. $x \xleftarrow{\$} R$ denotes a random element sampled according to uniform distribution over ring R, $x_i \xleftarrow{\$} R, i = \overline{1, m}$ – random elements sampled uniformly and independently, $x \xleftarrow{\mathbb{D}} R$ – element sampled according $\mathbb{D}$. $f(x) \xleftarrow{\$} \mathbb{Z}_n[x]$ means that $f_i \xleftarrow{\$} \mathbb{Z}_n, i = \overline{0, deg(f)}$.

An adversary trying to break cryptosystem is denoted by $\mathcal{A}$. For symmetric cryptosystem ε: P – plaintexts space, C – ciphertexts space, K – secret keys space, sk – secret key, $\mathbb{PD}$ – probabilistic distribution over P, $\mathcal{PP}$ – the set of cryptosystem's public parameters.

2.2 Resultant of polynomials

Let's recall the notion of resultant. It's necessary for our attack on [15]. Consider $f_i(x) = \sum_{j=0}^{d_i} f_{i,j} \cdot x^j \in \mathbb{Z}_n[x], i = 1, 2$. One may compose a Sylvester matrix $\mathcal{S} \in \mathbb{Z}_n^{(d_1+d_2)\times(d_1+d_2)}$ for f_1, f_2:

$$\mathcal{S} = \begin{pmatrix} f_{1,0} & \cdots & f_{1,d_1} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & f_{1,0} & \cdots & f_{1,d_1} \\ f_{2,0} & \cdots & f_{2,d_2} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & f_{2,0} & \cdots & f_{2,d_2} \end{pmatrix} \tag{1}$$

The resultant of f_1, f_2 is $\mathcal{R} = Res(f_1, f_2) = [det(\mathcal{S})]_n \in \mathbb{Z}_n$. $\mathcal{R} = 0$ iff f_1, f_2 have a common root or factor modulo n (for details see [12]). Also for further discussion we need the next proposition.

Proposition 1. *Consider $n = p \cdot q \in \mathbb{Z}_+$, $p, q \in \mathbb{Z}_+$ – primes, $p \neq q$ and $f_1(x), f_2(x) \in \mathbb{Z}_n[x]$. If f_1, f_2 have a common root or a factor modulo p (or q) then $[\mathcal{R}]_p = 0$ (or $[\mathcal{R}]_q = 0$). And also $\mathcal{R} = 0$ iff $[\mathcal{R}]_p = 0$ and $[\mathcal{R}]_q = 0$ hold.*

We omit the proof because it could be is easily derived from the Chinese reminder theorem and properties of congruences.

2.3 Overview of FHC proposed in [15]

Let's briefly recall the description of a cryptosystem from [15]. The scheme is defined by the choice of $P = \mathbb{Z}_p$, $K = \mathbb{I}_{p,d}$, $C = \mathbb{P}_{n,2\cdot d}$, where $d \in \mathbb{Z}_+$, $p, q \in \mathbb{Z}_+$ – prime numbers with bit length λ, $p < q$, $p \neq q$, $n = p \cdot q$, i.e n – RSA modulus. The factorization of n is a secret. The authors also introduce special public parameter – evaluation key Ek, which is necessary for computing over ciphertexts. Ek is a polynomial $w(x) \in \mathbb{P}_{n,2\cdot d+1}$.

Algorithm 1: KeyGen(λ, d).

```
Input: $\lambda, d \in \mathbb{Z}_+$
Output: sk and $\mathcal{PP}$
1 Generate $p, q, n$;
2 $u(x) \xleftarrow{\$} \mathbb{I}_{p,d}$
3 $sk := u(x)$;
4 $s(x) \xleftarrow{\$} \mathbb{I}_{p,d+1}$;
5 $r(x) \xleftarrow{\$} \mathbb{P}_{n,2\cdot d}$;
6 $w(x) := [s(x) \cdot u(x) + p \cdot r(x)]_n$;
7 $Ek := w(x)$;
8 $\mathcal{PP} := \{n, Ek\}$;
9 return sk,$\mathcal{PP}$;
```

This cryptosystem is FHC, because any polynomial function may be computed homomorphically. Indeed, consider $m_1, m_2 \in \mathbb{Z}_p$ and $c_1(x)$,

```
Algorithm 2: Encrypt($m, \mathbf{sk}$).
   Input: plaintext $m \in \mathbb{Z}_p$, $sk$
   Output: $c(x) \in \mathcal{C}$
1  $s(x) \xleftarrow{\$} \mathbb{I}_{p,d}$;
2  $r(x) \xleftarrow{\$} \mathbb{P}_{n,2\cdot d-1}$;
3  $c(x) := [s(x) \cdot u(x) + p \cdot r(x) + m]_n$;
4  return $c(x)$;
```

```
Algorithm 3: Decrypt($c(x)$).
   Input: $c(x) \in \mathcal{C}$, $sk$
   Output: plaintext $m \in \mathbb{Z}_p$
1  $m \leftarrow [c(x)]_p \bmod sk$;
2  return $m$;
```

$c_2(x) \in C$ – encryptions made for the same sk, d. In [15] the following proposition is proved.

Proposition 2. *For sk, d $c_+(x) = [c_1(x) + c_2(x)]_n$, $c_*(x) = [(c_1(x) \cdot c_2(x)) \bmod w(x)]_n$ are correct encryptions of plaintexts $[m_1 + m_2]_p$ and $[m_1 \cdot m_2]_p$ correspondingly.*

In practice $1024 \le \log(n) \le 4096$ and then ciphertext size is $S \le 8192 \cdot d$ bits. This implies $d < 125$ to obtain $S \le 10^6$. Such setting seems to be reasonable and in all latest FHCs [4] $S \le 10^6$. Larger S will make homomorphic computations too much expensive.

Ending with the description of this FHC we'll correct a little mistake made in [15]. The authors of [15] state that without factorization of n it's not possible to decrypt $c(x)$ using only $sk = u(x)$. But this is false. It is easy to verify that a division of $c(x)$ by $u(x)$ modulo n step by step eliminates terms $\sum s_i \cdot u_{j-i}$ and final residue $g(x)$ has coefficients $g_i = p \cdot a_i \in \mathbb{Z}_n$, $a_i \in \{0, ..., q-1\}$ except of free coefficient, which is equal to $[p \cdot b + m]_n$, $b \in \{0, ..., q-1\}$. So one may find p by computing $GCD(g_i, n)$ and then extract m.

3 A known plaintext attack on the cryptosystem from [15]

3.1 KPA based on resultant computation

The attack on described FHC exploits the following property of ciphertexts.

Proposition 3. *Let $c_1(x), c_2(x) \in \mathbb{Z}_n[x]$ – be encryptions of zero plaintext produced by algorithm 2 with sk. $c_1(x), c_2(x)$ has a common factor modulo p.*

Proof. $c_i(x) := [s_i(x) \cdot u(x) + p \cdot r_i(x)]_n, i = 1, 2 \Rightarrow [c_i(x)]_p := [[s_i(x) \cdot u(x)]_p, i = 1, 2$. So $u(x)$ is a common factor modulo p. □

Now let $\mathcal{A}$ intercepted two pairs – $(m_i \in \mathbb{Z}_p, c_i(x) \in C), i = 1, 2$ produced with $sk = u(x)$. According to proposition 3 $f_i(x) = [c_i(x) - m_i]_n \in \mathbb{Z}_n[x]$ have common factor $u(x)$ modulo p. Hence for $\mathcal{R} = res(f_1, f_2) \in \mathbb{Z}_n$ equality $\mathcal{R}_p = 0$ holds (see proposition 1). So if $\mathcal{R} \neq 0$, than secret modulus p may be recovered from the following equation:

$$p := GCD(n, \mathcal{R}). \tag{2}$$

Remark 1. *When computing (2) one works with $\mathcal{R}$ like with integer $\in \{0, ..., n-1\}$.*

So we only need to estimate the probability that for randomly intercepted pairs $\mathcal{R} \neq 0$ holds. Statement 1 yields $\mathcal{R} \neq 0$ iff $[\mathcal{R}]_q \neq 0$. So we have to estimate $Pr\{[\mathcal{R}]_q \neq 0\}$. This could be done with the following theorem from [1].

Theorem 4. *([1]) Let $(d_1, ..., d_m)$ be an ordered m-tuple of nonnegative integers (not all zeroes) and for $1 \leq i \leq m$ let $g_i(x) \xleftarrow{\$} \mathbb{Z}_q[x], i = \overline{1, m}$, where q is a prime. Then the probability that $g_i(x), i = \overline{1, m}$ are relatively prime is $1 - 1/q^m$.*

Remark 2. *Since in $\mathbb{Z}_q \setminus \{0\}$ all elements are invertible, in theorem 4 arbitrary $g_i(x)$ may be replaced by monic $g_i(x)$ (see the proof in [1] for details).*

Clearly, if $f_{q,i}(x) = [f_i(x)]_q, i = 1, 2$ may be considered as uniformly random in $\mathbb{Z}_q[x]$ then theorem 4 implies $[\mathcal{R}]_q = 1 - 1/q$ (≈ 1 for large q). Hence it's left to show that $f_{q,i}(x) \overset{\$}{\leftarrow} \mathbb{Z}_q[x]$.

Proposition 5. *Let $q \in \mathbb{Z}_+$, $a_1 \overset{\$}{\leftarrow} \mathbb{Z}_q$ and $a_2 \overset{\mathbb{D}}{\leftarrow} \mathbb{Z}_q$, where $\mathbb{D}$ is some distribution over $\mathbb{Z}_q$, a_1, a_2 – independent. Then random variable $a_+ = [a_1 + a_2]_q$ is s.t. $a_+ \overset{\$}{\leftarrow} \mathbb{Z}_q$.*

Proposition 6. *Let $q \in \mathbb{Z}_+$ is prime, $p \in \mathbb{Z}_+$, $p \neq 0$, $p < q$, $a \overset{\$}{\leftarrow} \mathbb{Z}_q$. Then for $a_p = [p \cdot a]_q$ there is $a_p \overset{\$}{\leftarrow} \mathbb{Z}_q$.*

Theorem 7. *Let $f(x) = [c(x) - m]_n \in \mathbb{Z}_n[x]$, where $c(x)$ encrypts m (by algorithm 2). Then $f_q(x) = [f(x)]_q \overset{\$}{\leftarrow} \mathbb{P}_{q,2\cdot d,mon}$ holds.*

Proof of lemmas 5, 6 and theorem 7 are in Appendix.

By theorem 7 and according to algorithm 2 there exist $f_{q,i}(x) \overset{\$}{\leftarrow} \mathbb{P}_{q,2\cdot d,mon}[x]$, $i = 1, 2$. So putting all things together we obtain theorem 8 compromising the KPA-security of FHC from [15].

Theorem 8. *If $\mathcal{A}$ intercepted $(m_i \in \mathbb{Z}_p, c_i(x) \in C), i = 1, 2$ produced with sk, then computation of $GCD(n, \mathcal{R})$ recovers p with probability $1 - 1/q$, where $\mathcal{R} = res(f_1, f_2) \in \{0, ..., n-1\}$, $f_i(x) = [c_i(x) - m_i]_n, i = 1, 2$.*

After recovering p $\mathcal{A}$ computes $f_{p,i}(x) = [f_i(x)]_p = [s_i(x) \cdot u(x)]_p, i = 1, 2$. To determine $u(x)$ one may calculate

$$GCD(f_{p,1}(x), f_{p,2}(x)) = GCD(s_1(x), s_2(x)) \cdot u(x) \in \mathbb{Z}_p[x].$$

Due to algorithm 2 $s_1(x), s_2(x) \overset{\$}{\leftarrow} \mathbb{I}_{p,d}$, so $s_1(x), s_2(x)$ are relatively prime with probability $Pr = 1 - 1/\nu_p$, where ν_p – the number of irreducible polynomials in $\mathbb{Z}_p[x]$ and $Pr \approx 1$ for large p. Thus the probability to find $u(x)$ is $1 - 1/\nu_p$.

Finally let's estimate the probability of obtaining $\{p, u(x)\}$ using described method. According to algorithm 2 events $GCD(s_1(x), s_2(x)) = const$ and $GCD(f_{q,1}(x), f_{q,2}(x)) = const$ are independent. So our attack is successful with probability $(1 - 1/\nu_p) \cdot (1 - 1/q)$ (≈ 1).

Table 1: Running times of attack for different p, q, d

d	Time for $\log(n) = 2^{10}, \log(p) = 2^9$	Time for $\log(n) = 2^{11}, \log(p) = 2^{10}$
8	132 ms	443 ms
16	521 ms	1.73 s
32	2.24 s	7.2 s
64	10.7 s	29 s
128	55 s	2.2 min
256	5.2 min	12.3 min
512	23 min	52 min

Table 2: Practical estimation of $\mathfrak{Pr}$ for different bit length of n, $8 \leq d \leq 512$

$\log(n)$	$\mathfrak{Pr}$
2^9	0.9998
2^{10}	1
2^{11}	1

If $\mathcal{A}$ has > 2 pairs he may apply this strategy several times. This increases the probability of success. But in fact two pairs are enough since $\log(p), \log(q) > 500$. Moreover it may be sufficient to intercept only one pair $(m, c(x))$ in the presence of $Ek = w(x)$ made for the same sk. $w(x)$ is encryption of zero and one may compute $\mathcal{R} = res([c(x) - m])_n, w)$ to find p.

The complexity of described KPA is $\approx O(d^3 \cdot \log^2(n))$.

3.2 Experimental data

FHC from [15] and presented KPA were implemented using Qt 1.3.1 and NTL library. For testing of the implementation the middle-range workstation was used: AMD Phenom(tm) II P960 Quad-Core Processor 1.80 GHz, 4 GB RAM. In tables 1,2 we present KPA running times for different p, q, d and practical estimation of probability $\mathfrak{Pr}$ to recover p and $sk = u(x)$. Please note that for estimation of $\mathfrak{Pr}$ two pairs $(m_i \in \mathbb{Z}_p, c_i(x) \in C)$ were generated randomly $\approx 10^5$ times.

3.3 Similarity and differences between FHC [15] and Domingo-Ferrer's HC

KPA on FHC [15] is similar to KPA on well-known Domingo-Ferrer HC [5]. This is due to the fact that Domingo-Ferrer HC [5] works in such a way that its secret key $sk \in \mathbb{Z}_p$ is a root of ciphertext $c(x) \in \mathbb{Z}_n[x]$ modulo p, where $n = p \cdot q$ – RSA modulus. In [15] instead of a common root ciphertexts has a common irreducible factor $u(x)$ modulo secret p.

Let's outline the difference. The success of resultant-based attack on Domingo-Ferrer HC [5] heavily depends on $\mathbb{PD}$, because in [11] it is shown that polynomials $[f(x)]_n = [c(x) - m]_n$ for [5] inherit $\mathbb{PD}$ and KPA works provably good only for uniform $\mathbb{PD}$. But if $\mathbb{PD}$ is s.t. $Pr\{m = 0\}$ is significant then probability of successful key recovering is small. But the situation is different for FHC [15]. In [15] polynomials $[f(x)]_n$ are independent of $\mathbb{PD}$ (see theorem 7) and so KPA works good for any $\mathbb{PD}$.

4 Conclusion

We have presented a KPA on factorization based FHC [15]. The complexity of the attack is polynomial in cryptosystem's parameters and the probability of successful key recovering is ≈ 1. To implement the attack it's enough to intercept two pairs of plaintext and corresponding ciphertext. Moreover in the presence of Ek interception of only one pair is sufficient. And this makes FHC from [15] completely KPA-insecure.

The existence of provably KPA-secure FHC based on factorization is still an open problem. Whether it is possible to construct it in principle is an important direction of further research.

Also in future we are planning to investigate the resistance of FHC from [15] to COA.

5 Acknoledgements

The work is supported by grant RFBR 15-07-00597-a.

References

[1] A. T. Benjamin and C. D. Bennett. The probability of relatively prime polynomials. *Mathematics Magazine*, pages 196–202, 2007.

[2] J. Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism*. In *Information Security*, pages 471–483. Springer, 2002.

[3] C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[4] A. Guellier. *Can Homomorphic Cryptography ensure Privacy?* PhD thesis, Inria; IRISA; Supélec Rennes, équipe Cidre; Université de Rennes 1, 2014.

[5] J. D. i Ferrer. A new privacy homomorphism and applications. *Information Processing Letters*, 60(5):277–282, 1996.

[6] A. Kipnis and E. Hibshoosh. Efficient methods for practical fully homomorphic symmetric-key encrypton, randomization and verification. *IACR Cryptology ePrint Archive*, 2012:637, 2012.

[7] T. Mather, S. Kumaraswamy, and S. Latif. *Cloud security and privacy: an enterprise perspective on risks and compliance.* " O'Reilly Media, Inc.", 2009.

[8] D. K. Rappe. *Homomorphic cryptosystems and their applications.* PhD thesis, Universität Dortmund, 2005.

[9] A. Rostovtsev, A. Bogdanov, and M. Mikhaylov. Secure evaluation of polynomial using privacy ring homomorphisms. *IACR Cryptology ePrint Archive*, 2011:24, 2011.

[10] K. Shatilov, V. Boiko, S. Krendelev, D. Anisutina, and A. Sumaneev. Solution for secure private data storage in a cloud. In *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, pages 885–889. IEEE, 2014.

[11] A. Trepacheva. Improved known plaintexts attack on domingo-ferrer homomorphic cryptosystem. *Proceedings of ISP RAS*, 26(5), 2014.

[12] J. Von Zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge university press, 2013.

[13] D. Wagner. Cryptanalysis of an algebraic privacy homomorphism. In *Information Security*, pages 234–239. Springer, 2003.

[14] L. Xiao, O. Bastani, and I.-L. Yen. An efficient homomorphic encryption protocol for multi-user systems. *IACR Cryptology ePrint Archive*, 2012:193, 2012.

[15] A. Zhirov, O. Zhirova, and S. F. Krendelev. Practical fully homomorphic encryption over polynomial quotient rings. In *Internet Security (WorldCIS), 2013 World Congress on*, pages 70–75. IEEE, 2013.

A Proofs

Proof of Proposition 5. For $\forall a \in \mathbb{Z}_q$ there is $a = [j + j_a]_q$ for $\forall j \in \mathbb{Z}_q$ and $j_a = [q - j + a]_q \in \mathbb{Z}_q$. For $j_1 \neq j_2$ $j_{a,1} \neq j_{a,2}$ holds. This implies $Pr\{a_+ = a\} = \sum_{j=0}^{q-1} Pr_{\mathbb{D}}\{j\} \cdot 1/q$, where $Pr_{\mathbb{D}}\{j\}$ – probability of j due to $\mathbb{D}$. Obviously $\sum_{j=0}^{q-1} Pr_{\mathbb{D}}\{j\} = 1$ and then $Pr\{a_+ = a\} = 1/q$. $\square$

Proof of Proposition 6. Since q is prime, any $j \in \mathbb{Z}_q$ may be represented as $j = [p\cdot(p^{-1}\cdot j)]_q$, where $[p\cdot p^{-1}]_q = 1$ and for $j_1 \neq j_2$ there is $[p^{-1}\cdot j_1]_q \neq [p^{-1}\cdot j_2]_q$. Then for fixed p we have $Pr\{a_p = j\} = Pr\{a = [p^{-1} \cdot j]_q\} = 1/q$. $\square$

Proof of Theorem 7. There is $f(x) := [s(x) \cdot u(x) + p \cdot r(x)]_n$, where $s(x) \xleftarrow{\$} \mathbb{I}_{p,d}$, $r(x) \xleftarrow{\$} \mathbb{P}_{n,2\cdot d-1}$. So $f_q(x) = [[s(x)\cdot u(x)]_q + [p\cdot r(x)]_q]_q$ holds. Let's look at $r_{p,q}(x) = [p \cdot r(x)]_q \in \mathbb{P}_{q,2\cdot d-1}$. Coefficients of $r(x)$ are s.t. $r_i \xleftarrow{\$} \mathbb{Z}_n, i = \overline{0, 2 \cdot d - 1}$ according to algorithm 2. This implies $[r_i]_q \xleftarrow{\$} \mathbb{Z}_q, i = \overline{0, 2 \cdot d - 1}$ and hence according to Proposition 6 $[p \cdot r_i]_q \xleftarrow{\$} \mathbb{Z}_q, i = \overline{0, 2 \cdot d - 1}$ holds. So using Proposition 5 one obtains $f_q(x) = [f(x)]_q \xleftarrow{\$} \mathbb{P}_{q,2\cdot d,mon}$. $\square$

Side Channel Cryptanalysis of Streebog

Gautham Sekar

Abstract

Streebog is the cryptographic hash function standard of the Russian Federation. It comprises two hash functions corresponding to two digest sizes, 256 bits and 512 bits. This paper presents a side channel attack that uses processor flag information to speed up message recovery by a factor of 2. Success is nearly guaranteed if the flag is set; the probability is 0.668 otherwise.

Keywords: Cryptographic hash function, Streebog, side channel cryptanalysis, carry flag, message recovery, HMAC.

1 Introduction

A hash function F takes an arbitrarily long bit string m as input and outputs a fixed length bit string H (called *hash value* or *digest*). A cryptographic hash function is meant to satisfy certain security properties, the most important of which are the following.

- **(First) preimage resistance**: given H, it is computationally infeasible to find an m' such that $F(m') = H$.
- **Second preimage resistance**: given an m and $F(m)$, it is computationally infeasible to find an $m' \neq m$ such that $F(m') = F(m)$.
- **Collision resistance**: it is computationally infeasible to find an m and an $m' \neq m$ such that $F(m) = F(m')$.

The general model for cryptographic hash functions involves what is called a compression function. The function transforms a bit string of a fixed length into a shorter string of a fixed length. The arbitrarily long message is partitioned into blocks after a process called padding (described later in the context of Streebog). The blocks are then sequentially processed, with the compression function acting on every block until all the blocks are processed. The final output is the hash value. The general model is described in good detail in [9, Sect. 2.4.1].

Streebog is a set of two hash functions and a Russian cryptographic standard (GOST R 34.10–2012) [5]. It was developed by the Center for Information Protection and Special Communications of the Federal Security Service of the Russian Federation, with participation of the Open Joint-Stock Company “Information Technologies and Communication Systems” (JSC “InfoTeCS”) [5], following a demand for “a hash function to meet modern requirements for cryptographic strength” [5]. In 2012, Streebog replaced GOST R 34.11–94 as the national standard.

The hash functions comprising Streebog have 256 bits and 512 bits as their digest lengths. We shall call the hash functions “Streebog-256” and “Streebog-512”, respectively. The compression function, common to both the versions, operates on 512-bit blocks in the Miyaguchi-Preneel mode, has 13 rounds, is based on a substitution-permutation network and uses a linear transformation.

Streebog has been analysed in [1, 2, 3, 6, 10]: in [1, 10], the rebound attack is used to find (semi-free-start) collisions for reduced versions of the Streebog compression function; [2] presents integral distinguishers on up to 7 rounds of the compression function; [3] reports preimages for 6-round Streebog; and [6] describes second preimage attacks on the full Streebog-512. The drawback of the attacks in [6] is that they work well only with long messages. For instance, if the length of the message is at least 2^{188} bits, then 2^{342} compression function evaluations are required. The time complexity can be brought down to as low as $O(2^{266})$ provided that the message is at least 2^{268} bits in length. For shorter messages, of bit-length $\gamma < 2^{188}$ (but greater than 512 bits), the number of compression function evaluations is estimated at $(\log_2 \gamma - 9) \cdot 2^{522 - \log_2 \gamma}$. We present in this paper the first side channel attack on the full Streebog. We also discuss the implications of our attack on the security of Streebog-based keyed-hash message authentication code (HMAC).

Processors have registers that store information on operations performed by their ALUs. For example, in the Intel IA-32 architecture, the *status flags* of the EFLAGS register indicate the result of arithmetic instructions such as ADD and DIV (divide) [7]. One of these flags, known as the *carry flag*, is a single bit that indicates an overflow in unsigned integer arithmetic. For instance, when two unsigned integers are added, the carry flag is set (to 1) if a carry is generated by the addition at the most significant bit position (we shall call this an *end carry*) and the flag is cleared (i.e., 0) otherwise. This may be exploited by an attacker as in [8] where Kelsey *et al.* use carry flag information to attack the block cipher RC5. In our side channel attack too we use the state of the carry flag. Our attack recovers a message block in about 2^{511} time with 99.9% success rate (number of successful recoveries per 100 messages uniformly distributed at random) if the carry flag is set and 66.8% otherwise. The only other attack known on the full Streebog is due to Guo *et al.* [6].

The paper is organised as follows. Section 2 describes Streebog and Sect. 3 details our meesage recovery attack. We propose countermeasures to our attack in Sect. 4 and conclude in Sect. 5.

2 Description of Streebog

Table 1 lists the notation and conventions followed in the rest of this paper.

Streebog is a simple design that uses only a few elementary arithmetic operators such as XOR and modular addition, and simple functions such as substitution, permutation and linear transformation. The hash function accepts any message M of length less than 2^{512} bits and returns a digest of length 256 bits or 512 bits. The round function or compression function has 13 iterations, the first twelve of which involve a substitution-permutation layer. If $512 \nmid |M|$, then padding prefixes M with a bit string $pad := \{0\}^{511-(|M| \bmod 512)} \| 1$. The padded message is then partitioned into $(k+1)$ 512-bit blocks $M_k, M_{k-1}, \ldots, M_0$; i.e., $pad \| M = M_k \| M_{k-1} \| \cdots \| M_0$. The compression function g that processes the message block M_i takes as additional inputs the chaining value H_i (of size 512 bits) and a length

Table 1: Notation and conventions

Symbol/notation	**Meaning**
$\|W\|$	length of W in bits
$\Gamma_i(W)$	ith 64-bit word of W; $i = 0$ denotes the least significant word
$W_{(i)}$	ith bit of W; $i = 0$ denotes the least significant bit
$\|$	concatenation
$\oplus$	exclusive OR
fg, where f and g are functions	$f \circ g$ (composition of f and g)
LSB	least significant bit
MSB	most significant bit

counter N_i, and outputs H_{i+1}. Algorithm 1 describes the working of Streebog. The IV in the algorithm is the initial value H_0 (Streebog-256 and Streebog-512 use different 512-bit IVs).

Algorithm 1 The Streebog algorithm

```
Require: The message $M$, $|M| < 2^{512}$
Ensure: A 256-bit or a 512-bit digest
 1: $M \to pad\|M \to M_k\|M_{k-1}\|\cdots\|M_0$;
 2: $H_0 = IV$;
 3: $N_0 = 0$;
 4: for $i = 0$ to $(k-1)$ do
 5:     $H_{i+1} = g(H_i, M_i, N_i)$;
 6:     $N_{i+1} = N_i + 512 \bmod 2^{512}$;
 7:     $\Sigma \leftarrow \Sigma + M_i \bmod 2^{512}$;
 8: $H_{k+1} = g(H_k, M_k, N_k)$;
 9: $N_{k+1} = N_k + \alpha \bmod 2^{512}$, where $\alpha = 512 - |pad|$;
10: $\Sigma \leftarrow \Sigma + M_k \bmod 2^{512}$;
11: $H_{k+2} = g(H_{k+1}, N_{k+1}, 0)$;
12: $H = g(H_{k+2}, \Sigma, 0)$;
13: Output $H$ if Streebog-512, else output $H \gg 256$;
```

The substitution-permutation layer includes the following components.

- Substitution function S: The input, a 512-bit string, is first partitioned into bytes. Every byte is then substituted by a byte from a set π', which is a permutation of $\{0, 1, \ldots, 255\}$, and concatenated.

- Permutation function P: Partitions its 512-bit input into bytes, permutes the bytes (i.e., shuffles their positions) and concatenates them.
- Linear transformation L: This is also a 512-bit-to-512-bit mapping. If the input is W, then $L(W) = \ell(\Gamma_7(W))\|\ell(\Gamma_6(W))\| \cdots \|\ell(\Gamma_0(W))$, where ℓ is a 64-bit-to-64-bit linear transformation that outputs the right multiplication of its input with a constant matrix $\mathbf{A}$ over $GF(2)$.
- The function $X[\cdot]$: If K and W are 512-bit strings, then $X[K](W) = K \oplus W$.

The compression function g is now given by:

$$g(H_i, M_i, N_i) = E(LPS(H_i \oplus N_i), M_i)) \oplus H_i \oplus M_i \,, \tag{1}$$

where

$$E(LPS(H_i \oplus N_i), M_i)) = X[K_{13}]LPSX[K_{12}]LPSX[K_{11}] \dots LPSX[K_1](M_i) \,, \tag{2}$$

and

$$\begin{aligned} K_0 &= LPS(H_i \oplus N_i) \,, & (3)\\ K_{j+1} &= LPS(K_j \oplus C_j) \,, \text{ for } j = 0, 1, \dots, 12, \text{ and constants } C_j. & (4) \end{aligned}$$

3 The Message Recovery Attack

The functions S and P do not involve modular addition or multiplication. The function X is a simple XOR operation. The linear transformation ℓ works as follows. Denoting its 64-bit input by $\beta := \beta_{(63)}\|\beta_{(62)}\| \cdots \|\beta_{(0)}$, we have:

$$\ell(\beta) = \bigoplus_{i=0}^{63} \beta_{(63-i)} \odot A[i] \,,$$

where the product $\odot$ is defined as follows:

$$\beta_{(63-i)} \odot A[i] = \begin{cases} \{0\}^{64} & \beta_{(63-i)} = 0 \,; \\ A[i] & \beta_{(63-i)} = 1 \,. \end{cases}$$

Hence, from (1)–(4), it immediately follows that Streebog compression does not involve any operation, such as addition modulo 2^{512}, that can alter the state of the carry flag. This means that only steps 6, 7, 9 and 10 of Algorithm 1 can potentially affect the carry flag.[1]

Now, the maximum length of M is $2^{512} - 1$. Given a message of this length, the number of blocks will be $\lceil (2^{512}-1)/512 \rceil = 2^{503}$.[2] If $k+1 < 2^{503}$ (to simply calculations, this can be considered a sure event as it happens with a probability that is very close to 1 if $|M|$ is uniformly distributed at random over $\{0, 1, \ldots, 2^{512} - 1\}$), then $N_k = 512k$, $512k < N_{k+1} \leq 512(k+1)$, and the carry flag will be unaffected by steps 6 and 9. This leaves us with steps 7 and 10. Now,

$$\Sigma = \left(\sum_{i=0}^{k-1} M_i \right) \bmod 2^{512} + M_k \bmod 2^{512} . \tag{5}$$

$$= T_{k-1} + M_k \bmod 2^{512}, \text{ say.} \tag{6}$$

Let $C := [C_{(511)} C_{(510)} \cdots C_{(0)}]$ denote the vector of carries generated in (6) such that $C_{(0)}$ is the carry at the LSB position. When $k \geq 1$ (this can also be considered a sure event), we have the following attack.

Scenario 1: Suppose that the carry flag is set at the end of Algorithm 1. If $|pad| \geq 2 \Rightarrow M_{k(511)} = 0$, or $|pad| = 0$ and $M_{k(511)} = 0$, then $T_{k-1(511)} = C_{(511)} = 1$. If the attacker knows $M_0, M_1, \ldots, M_{k-2}$, and all but the MSB of M_{k-1}, then she can recover $M_{k-1(511)}$ from $T_{k-1(511)} = 1$ performing $k - 1 < 2^{503} - 2$ additions (recall (5) and (6)).

If $|pad| = 0$ and $M_{k(511)} = 1$, or $|pad| = 1 \Rightarrow M_{k(511)} = 1$, then there are three possibilities: *(i)* $T_{k-1(511)} = C_{(511)} = 1$, *(ii)* $T_{k-1(511)} = 0$ and $C_{(511)} = 1$, *(iii)* $T_{k-1(511)} = 1$ and $C_{(511)} = 0$. Assuming these cases to be equally likely,[3] the attacker can assume with 2/3 probability that $T_{k-1(511)} = 1$,

[1]The `for`-loop of Algorithm 1 is implemented differently in [5]. To obtain M_0, the least significant 512-bit word of the padded message is extracted. The leftover message replaces the padded message and its 512 LSBs are extracted as M_1. This process is repeated until all the message blocks have been extracted. The carry flag is evidently unaffected by the process.

[2]Therefore, even if we go with the `for`-loop implementation (Algorithm 1), it will have no bearing on the carry flag.

[3]Since the distribution of $|M_k|$ is uniform, given the padding scheme employed, the distribu-

and recover $M_{k-1(511)}$.

Table 2 lists the above cases and their probabilities assuming that *(i)* $|M_k|$ is uniformly distributed at random over $\{0, 1, \ldots, 511\}$, and *(ii)* every message block other than M_k is uniformly distributed at random over $\{0, 1, \ldots, 2^{512} - 1\}$. The attack methodology is as follows. The attacker, knowing $M_0, M_1, \ldots, M_{k-2}$ and M_k, makes a guess for the 511 LSBs of M_{k-1}, obtains a value for the MSB of M_{k-1} (assuming that $T_{k-1(511)} = 1$), hashes $M_k \| M_{k-1} \| \cdots \| M_0$, and compares the digest with the given hash value. If the values do not agree, the guess is incorrect and the attacker makes another guess. The process is repeated until the hash values agree. The sum σ of $M_0, M_1, \ldots, M_{k-2}$ modulo 2^{512} can be precomputed (cost is $k-2$); $\sigma + M_{k-1} \bmod 2^{512}$ can be performed at each guess and, in doing so, can be avoided while computing the digest (i.e., $\sigma + M_{k-1} \bmod 2^{512}$ can be stored and reused). To minimise memory usage, the storage element can be rewritten at the next guess. The probability of success is the probability that $T_{k-1(511)} = 1$ holds true. From Table 2, this probability is simply $510/512 + 1/768 + 1/1024 + 1/1536 \approx 0.999$. The attack requires 2^{511} hash function evaluations plus a precomputation cost of $k - 2 < 2^{503} - 3$. Memory requirements are negligible.

Table 2: Computing $Pr(T_{k-1(511)} = 1)$ when the carry flag is 1; the probability q is given the condition on $|pad|$ and r is given the conditions on $|pad|$ and $M_{k(511)}$

$\|pad\|$	Pr. (p)	$M_{k(511)}$	Cond. pr. (q)	$T_{k-1(511)}$	Cond. pr. (r)	Overall pr. (pqr)
≥ 2	510/512	0	1	1	1	510/512
1	1/512	1	1	1	2/3	1/768
0	1/512	0	1/2	1	1	1/1024
0	1/512	1	1/2	1	2/3	1/1536

Scenario 2: Suppose that the carry flag is 0 at the end of Algorithm 1. If $|pad| \geq 2 \Rightarrow M_{k(511)} = 0$, or $|pad| = 0$ and $M_{k(511)} = 0$, then at least

tion of M_k is not uniform. This makes it tedious to compute the distribution of the carry vector C. Hence the assumption.

one of $T_{k-1(511)}$ and $C_{(511)}$ is 0. Knowing $M_0, M_1, \ldots, M_{k-2}$, and all but the MSB of M_{k-1}, the attacker can recover $M_{k-1(511)}$ assuming that $T_{k-1(511)} = 0$. The assumption is valid in two out of the three possible cases: *(i)* $T_{k-1(511)} = C_{(511)} = 0$, *(ii)* $T_{k-1(511)} = 0$ and $C_{(511)} = 1$, *(iii)* $T_{k-1(511)} = 1$ and $C_{(511)} = 0$. Assuming that these cases are equally likely, $Pr(T_{k-1(511)} = 0) = 2/3$.

When $|pad| = 0$ and $M_{k(511)} = 1$ or when $|pad| = 1 \Rightarrow M_{k(511)} = 1$, then $T_{k-1(511)} = C_{(511)} = 0$.

Table 3 lists the above cases and their probabilities under the assumption that *(i)* $|M_k|$ is uniformly distributed at random over $\{0, 1, \ldots, 511\}$, and *(ii)* every message block other than M_k is uniformly distributed at random over $\{0, 1, \ldots, 2^{512} - 1\}$. The attack methodology is identical to that described under Scenario 1, except that the attacker here assumes that $T_{k-1(511)} = 0$. The probability of success is the probability that $T_{k-1(511)} = 0$ holds true. From Table 3, this probability is $170/256 + 1/512 + 1/1536 + 1/1024 \approx 0.668$. The time complexity and memory requirements are the same as that in Scenario 1.

Note: The probability that $T_{k-1} = 0$ given that the carry flag is 0 and $M_{k(511)} = 0$ is at least 1/2 since Pr(case *(i)* or case *(ii)*) $= Pr(T_{k-1}) = 1/2$ (given the assumption that the message blocks other than M_k are uniformly distributed). Even if the conditional probability is 1/2, the success probability will be $255/512 + 1/512 + 1/2048 + 1/1024 > 1/2$ (see Table 3). The success probability calculated from Table 2 changes negligibly when 2/3 is replaced by 1/2.

Table 3: Computing $Pr(T_{k-1(511)} = 0)$ when the carry flag is 0; the probability q is given the condition on $|pad|$ and r is given the conditions on $|pad|$ and $M_{k(511)}$

$\|pad\|$	Pr. (p)	$M_{k(511)}$	Cond. pr. (q)	$T_{k-1(511)}$	Cond. pr. (r)	Overall pr. (pqr)
≥ 2	510/512	0	1	0	2/3	170/256
1	1/512	1	1	0	1	1/512
0	1/512	0	1/2	0	2/3	1/1536
0	1/512	1	1/2	0	1	1/1024

In summary, by simply guessing $T_{k-1(511)}$ to be equal to the carry flag, the attacker is able to recover M_{k-1} with 2^{511} hash function evaluations and $k-2$ precomputations. The number of precomputations can be negligible in comparison to 2^{511} and even the maximum number of precomputations $(2^{503}-4)$ is considerably smaller than 2^{511}. Moreover, each precomputation is only an addition of two 512-bit integers. Consequently, the precomputation cost can be ignored. The success probability is 0.668 if the carry flag is 0 and 0.999 otherwise. Arriving at a single value for the probability is involved given the difficulty in determining the distribution of the carry vector C. It is easy to see that the attack works for any $i \in \{0, 1, \ldots, k-2\}$ in place of $k-1$. In the ideal case, either 2^{512} hash function evaluations are required or the success probability is $1/2$ for 2^{511} evaluations.[4] Since the compression functions of Streebog-256 and Streebog-512 are identical, our attack applies to both the hash functions.

3.1 Implications of Our Attack

Our attack may be particularly relevant to HMACs. Proposed by Bellare *et al.* [4] as a message integrity checking mechanism, a HMAC employs a hash function h in conjunction with a secret key K and generates a MAC value as follows:

$$HMAC(K, m) = h((K_0 \oplus opad) \| h((K_0 \oplus ipad) \| m))\,,$$

where m is the message, *opad* and *ipad* are public constants, and K_0 is the secret key or a function of K. The lengths of K_0, *opad* and *ipad* equal the length of a message block. Given the HMAC value and $h((K_0 \oplus ipad)\|m)$, in certain cases, our attack appears to speed up the recovery of K_0 by a factor of 2. This is being further investigated.

4 Countermeasures

A simple way to preclude our preimage attacks is to introduce a low-cost arithmetic operation, after step 12 of Algorithm 1, that permanently sets

[4]This does not apply to M_k unless $|pad| = 0$. Knowing $|pad|$ and $M_0, M_1, \ldots, M_{k-1}$, the attacker can recover M_k in $2^{512-|pad|}$ time. Our attack is not intended to recover M_k.

or clears the carry flag. However, the approach fails if the attack model assumes that the attacker can determine the status of the carry flag after step 12.[5]

A faster and safer countermeasure is to implement the checksum using XOR; i.e., replace the addition modulo 2^{512} in steps 7 and 10 of Algorithm 1 with XOR.

5 Conclusions

In this paper, we have presented the first known side channel attack on Streebog. The attack speeds up message recovery by a factor of 2 with a probability that lies in $[0.668, 0.999]$. The attack is conjectured to be applicable to Streebog-based HMAC.

Our attack recovers the MSB of a message block. It is possible to recover bits of lower significance but calculating the success probabilities is involved and beyond the scope of this paper. We leave it as a problem for future work. Use of other processor flags such as the parity flag is also worth investigating.

We have also proposed countermeasures to our attacks. These evidently also preclude the above proposed extensions to the attacks.

References

[1] R. AlTawy, A. Kircanski, A. M. Youssef, "Rebound attacks on Stribog", *ICISC 2013* (H.-S. Lee, D.-G. Han, eds.), vol. 8565 of *LNCS*, pp. 175–188, Springer, 2014.

[2] R. AlTawy, A. M. Youssef, "Integral Distinguishers for Reduced-round Stribog", Information Processing Letters, vol. 114(8), pp. 426–431, 2014.

[5] A similar assumption is made in [8].

[3] R. AlTawy, A. M. Youssef, “Preimage attacks on Reduced-round Stribog”, AFRICACRYPT 2014, Lecture Notes in Computer Science, vol. 8469, pp. 109–125, 2014.

[4] M. Bellare, R. Canetti, H. Krawczyk, “Keying Hash Functions for Message Authentication”, *CRYPTO 1996* (N. Koblitz, ed.), vol. 1109 of *LNCS*, pp. 1–15, Springer, 1996.

[5] *Federal Agency On Technical Regulation And Metrology*, “NATIONAL STANDARD OF THE RUSSIAN FEDERATION GOST R 34.11–2012” (English Version), 01st January, 2013.

[6] J. Guo, J. Jean, G. Leurent, T. Peyrin, L. Wang, “The Usage of Counter Revisited: Second-Preimage Attack on New Russian Standardized Hash Function”, *SAC 2014* (A. Joux, A. Youssef, eds.), vol. 8781 of *LNCS*, pp. 195–211, Springer, 2014.

[7] Intel, “IA-32 Intel Architecture Software Developer’s Manual”, vol. 1 (Basic Architecture), 426 pages, 2003. Available via: http://flint.cs.yale.edu/cs422/doc/24547012.pdf.

[8] J. Kelsey, B. Schneier, D. Wagner, C. Hall, “Side Channel Cryptanalysis of Product Ciphers”, Journal of Computer Security, vol. 8, pp. 141–158, 2000.

[9] B. Preneel, “Analysis and Design of Cryptographic Hash Functions”, PhD thesis, Katholieke Universiteit Leuven, 1993.

[10] Z. Wang, H. Yu, X. Wang, “Cryptanalysis of GOST R hash function”, Information Processing Letters, vol. 114(12), pp. 655–662, 2014.

On Differential Properties of a Symmetric Cryptoalgorithm Based on Pseudo-dynamic Substitutions

Alexey Kozhevnikov Sergey Polikarpov Konstantin Rumyantsev

Abstract

In this work we propose a structure of a block cryptoalgorithm "Collapser" on the basis of pseudo-dynamic substitutions which application allows to combine advantages of fixed substitutions (high speed of work and efficiency of use of computing resources) and dynamic substitutions (neutralization of statistical methods of cryptanalysis). We provide primary assessment of differential properties of reduced version of "Collapser" in comparison with differential properties of random substitutions of similar dimension. The obtained results are well consistent with the principles laid down in the structure of "Collapser" and show that after 3 rounds "Collapser" looks as random substitution. However, the offered structure of symmetric cryptoalgorithm demands additional research of cryptographic properties.

Keywords: block cryptoalgorithm; pseudo-dynamic substitution.

1 Introduction

Importance of a task of search of new approaches to synthesis of the symmetric cryptoalgorithms resistant to all range of cryptanalytic attacks is confirmed by the series of competitions on cryptographic algorithms held in recent years and high activity of scientific researches in this area. Methods of linear and differential cryptanalysis, and also their derivatives are widely applied to an assessment of firmness of modern block cryptoalgorithms [1]. It should be noted that the basic element bringing complexity (nonlinearity) in cryptographic transformation are random substitutions (replacement blocks). The specified methods of cryptanalysis are directed on search and exploiting of weaknesses of substitutions and are most effective at fixed substitutions as demand for their work a considerable volume of statistics [2, 3].

In the existing block cryptoalgorithms the main approach for neutralization of the specified attacks is an application of a significant amount of rounds of encryption (usually more than 8), more effective operations of hashing and selection of substitutions with the maximum characteristics: maximum nonlinearity, minimum autocorrelated characteristics and others [4, 5, 6, 7, 8].

Naturally, the methods of synthesis of fixed substitutions are very important, as they must simultaneously satisfy a number of cryptographic properties [1, 9, 10, 11, 12]. A typical case is when synthesized substitution is strong performance for one characteristic, but on another - poor performance [7, 9]. The problem is compounded by the fact that the fixed substitution that has an ideal linear and differential characteristics, doesn't exist [13].

Another obvious approach for the neutralization of linear and differential cryptanalysis is an application of dynamic substitutions which can change in the encryption process [14, 15, 16, 17, 18]. However, such approach doesn't guarantee automatic decrease in efficiency of linear and differential cryptanalysis. The effect can be reached only when dynamic substitutions achieve the statistical characteristics which are coming closer to ideal (in the conditions of equiprobable dynamic change of tables of substitution). Besides, the majority of researches on application of dynamic substitutions consider only option of key-dependency of substitutions (Key-Dependent S-box) — substitution modification depending on a value of a cryptographic key [14, 15, 16, 17, 18]. A number of the cryptographic algorithms using this option is known [19, 20, 21], but it doesn't give them considerable advantages in comparison to analogues [5, 22].

In this work a structure of block symmetric cryptoalgorithm on the basis of pseudo-dynamic substitutions is considered (*PD-sbox* [23, 24]). Pseudo-dynamic substitutions allow to combine advantages of fixed substitutions (high speed of work and efficiency of use of computing resources) and dynamic substitutions (neutralization of statistical methods of cryptanalysis).

The goal of the our research — an assessment of differential properties of reduced versions of the proposed structure of block cryptoalgorithm "Collapser" in order to confirm the efficiency of the structure and application of pseudo-dynamic substitutions of *PD-sbox*.

2 Pseudo-dynamic substitutions

The structure of a pseudo-dynamic substitution (fig. 1) is based on the fixed substitutions *sbox* [23]. An argument of each fixed substitution will be parameterized by the individual value of a state S^i, where i – a number of the fixed substitution (from 0 to K-1). The current value of a state $S = \left\{S^0, S^1, S^2, \ldots, S^{K-1}\right\}$ sets one substitution from all set of possible substitutions of *PD-sbox*. We will call the table of substitution obtained by a concrete value of S the equivalent (generated) substitution for *PD-sbox*. Respectively, the number of various equivalent substitutions for *PD-sbox* is defined by quantity of possible states of S.

It is supposed that values of a state S are not necessarily fixed and can dynamically change during the encryption process, and probabilistic properties corresponds to the uniform distribution.

The general view of the expression, describing the structure of pseudo-dynamic substitution of *PD-sbox*, is the following:

$$Y = \bigoplus_{i=0}^{K-1} sbox_i(X \oplus S^i). \tag{1}$$

where *sbox* – fixed substitutions; K – number of the fixed substitutions; X – input bits; Y – output bits; S – bits of a state of a pseudo-dynamic substitution; $\oplus$ – addition modulo 2.

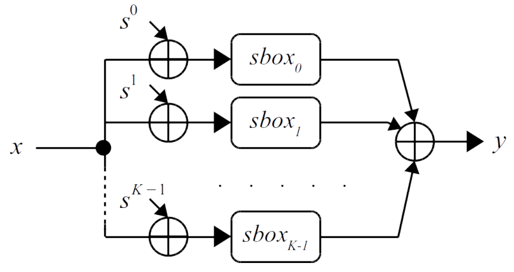

Figure 1: The structure of pseudo-dynamic substitution.

For a pseudo-dynamic substitution *PD-sbox* we performed a primary analysis of differential [24] and linear properties. In case of a dynamic equiprobable change of values of a state of S both differential and linear characteristics are close to ideal (when averaging characteristics on all generated substitutions). That allows to neutralize the existing methods of linear and differential cryptanalysis. However, the relevant structure of cryptoalgorithm is necessary for complete description of *PD-sbox* properties. A structure called "Collapser" (a black hole) is proposed in this work for an assessment of differential properties.

3 Notations and parameters of a structure of cryptoalgorithm "Collapser"

PD-sbox – pseudo-dynamic substitution;
L_{block} – length of the block of text, bit;
L_{key} – length of a secret key, bit;
nr – number of rounds of encryption;
M – number of input bits of pseudo-dynamic substitution of *PD-s*box;
$N_{words} = {}^{L_{block}}/_{M}$ – number of input words in the block;
$K \geq N_{words}$ – number of the fixed substitutions as a part of *PD-s*box;
$N_{rows} = K \geq N_{words}$ – number of lines of a matrix of state values;
$m = \{m_0, m_1, \ldots, m_{Nwords-1}\}$ – vector of the initial message;
$a = \{a_0, a_1, \ldots, a_{Nwords-1}\}$ – vector of values at the end of the 1st round;
$b = \{b_0, b_1, \ldots, b_{Nwords-1}\}$– vector of values at the end of the 2nd round;
$c = \{c_0, c_1, \ldots, c_{Nwords-1}\}$ – ciphertext vector;
IV – initialization vector;
i – number of a column (corresponds to number of the word);
j – line number;
$s_{i,j}^{nr}$ – state value;
$k_{i,j}^{nr}$ – values of an expanded key;
x_i^{nr} – input value of a pseudo-dynamic substitution of *PD-s*box;
y_i^{nr} – output value of a pseudo-dynamic substitution of *PD-s*box;

4 A short description of a structure of block cryptoalgorithm of "Collapser"

Figure 2 shows the easily scalable structure of a block cryptoalgorithm under the general name "Collapser". It's represented by a sequence of the linked pseudo-dynamic substitutions of *PD-sbox*.

The structure is compactly presented in the form of an source code (for example, in language C). An example of a pseudo-code of the program for the encryption function (when $nr = 3$) is the following:

```
encrypt_block_deadloop(m_in[], c_out[], IV[], key[][]) {

   uint a[Nwords], b[Nwords], c[Nwords], s[Nrows];

   for (j = 0; j < Nrows; j++) { s[j] = IV[j]; }

   round_func(m_in, a, s, key, RAPID_MIX); // PROCESS ROUND 1
   round_func(   a, b, s, key, RAPID_MIX); // PROCESS ROUND 2
   round_func(   b, c, s, key, SHADOW);    // PROCESS ROUND 3

   for (j = 0; j < Nrows; j++) {c_out[j] = c[j]; }
}
```

The minimum number of rounds of encryption is $nr \geq 3$ and is predetermined by the structure of cryptoalgorithm.

An example of a pseudo-code of the program for round function is the following:

```
round_func(in[], out[], s[], key[][], TYPE) {

  for(i = 0; i < Nwords; i++) {
   y = 0;
     for (j = 0; j < Nrows; j++)
        s[j] ^= key[i][j];
        s[j] = sbox[j][ s[j] ];
        y ^= s[j];
     }
   out[i] = in[i] ^ y;
     for (j = 0; j < Nrows; j++) {
        if(TYPE == RAPID_MIX) s[j] ^= out[i];
        if(TYPE == SHADOW)    s[j] ^= in[i];
     }
  }
```

16 }

It is supposed that values of the key array are obtained by means of expansion of an initial key, which is not considered within this work.

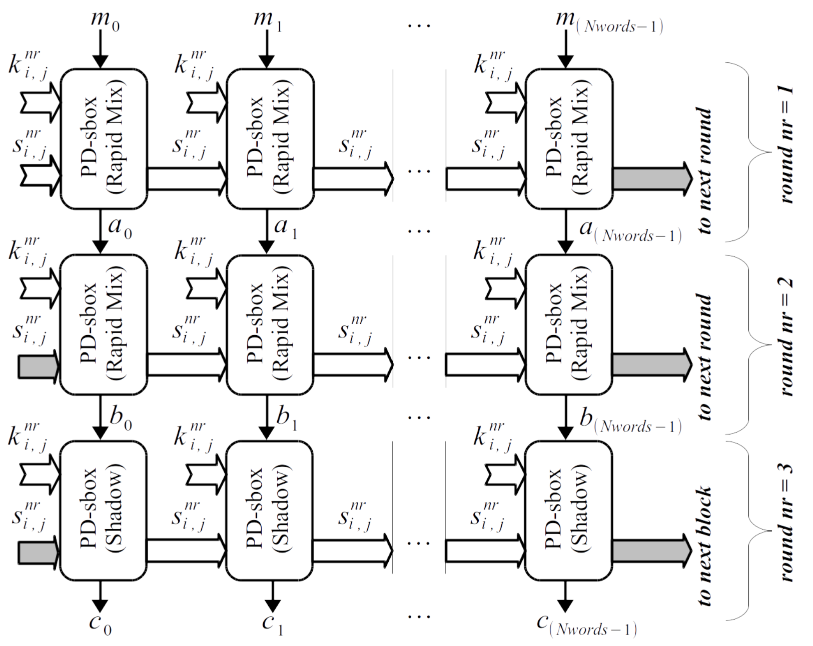

Figure 2: The structure of "Collapser".

Generally, *PD-sbox* is represented by 4 operations – expansions (Expand), mixing with an expanded key (Mixkey), the block of nonlinear substitutions (Nonlinear substitution), narrowings (Shrink). Coupling of pseudo-dynamic substitutions of *PD-sbox* is carried out through the state values $s_{i,j}^{nr}$,that allow to gain entropy and nonlinearity of a state consistently $s_{i,j}^{nr}$.

There are two options of connection *PD-sbox* - with intensive mixing state values $s_{i,j}^{nr}$ (RAPID MIX, Fig. 3) and with masking state values $s_{i,j}^{nr}$ (SHADOW, Fig. 4). "SHADOW" is implemented in the last round of encryption for "whitening".

As for not bijectivity of block transformation this structure is applicable only for the modes of encryption which aren't demanding reversibility of transformation. For example, modes of a keystream generation, calculation of hash values or message authentication codes. It should be noted that the proposed structure of cryptoalgorithm can be transformed to bijective function by breaking the connection between rounds $s_{i,j}^{nr}$ and alternation of

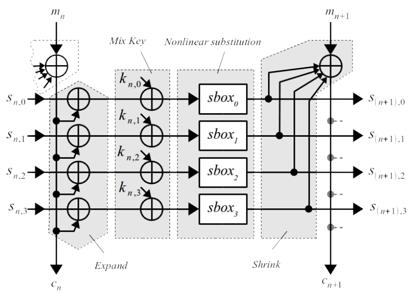

Figure 3: A pseudo-dynamic substitution, "RAPID MIX" type.

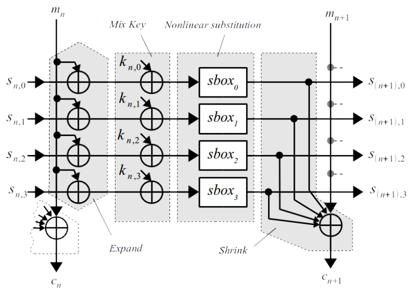

Figure 4: A pseudo-dynamic substitution, "SHADOW" type.

the direction of inclusion of *PD-sbox* (from left to right and vice versa), but thus the total nonlinearity of the transformation will decrease.

5 Basic principles laid down in "Collapser"

In this work justification of the proposed structure of cryptoalgorithm isn't considered. Separate works will be devoted to it. Briefly we will give the basic principles laid down in "Collapser":

1. Transition from the size of the input block ($Lblock$) to the size of the state ($Lblock \cdot Nrows$) (expansion).

2. Nonlinear transformation is applied to the state, which size is $Lblock \cdot Nrows$. This allows to maximize complexity (nonlinearity) of cryptographic transformation.

3. Obtaining output values by mapping the results of transformations from state of size ($Lblock \cdot Nrows$) to size of output block ($Lblock$) (shrinking).

4. Application of dynamically changeable substitutions based on *PD-sbox*

for neutralization of differential and linear cryptanalysis.

5. The special structure of the cryptographic algorithm, which allows efficient use of the pseudo-dynamic substitution *PD-sbox*:

5.1. The minimum number of types of operations – only additions modulo 2 (XOR) and substitutions are used.

5.2. Accumulation and transfer of entropy of a state $s_{i,j}^{nr}$ between successively connected *PD-sbox*'es within the round of encryption for maximizing dynamics of change of substitutions.

5.3. Accumulation of nonlinearity of transformations through values of a state $s_{i,j}^{nr}$ between successively connected *PD-sbox*'es within the round of encryption.

5.4. Intensive (rapid) mixing state values $s_{i,j}^{nr}$ between successively connected *PD-sbox*'es within the round of encryption.

5.5. Decrease in efficiency of cryptanalytic attacks due to restriction the manipulation with input values and the further accumulation of statistics.

5.6. Masking state values $s_{i,j}^{nr}$ due to introducing uncertainty in the transition from bigger dimension to smaller ($2^{M \cdot Nrows} \to 2^{M}$).

5.7. Additional transfer of values of a state $s_{i,j}^{nr}$ between rounds for maximizing uncertainty and nonlinearity of transformations.

6. Simplicity of scalability of a structure of cryptoalgorithm under various lengths of blocks and keys.

6 Assessment of full differentials for reduced versions of cryptoalgorithms on the basis of structure of "Collapser"

In order to determine the resistance against differential cryptanalysis we use the approach proposed in [5]and based on determination of full differentials for reduced versions of cryptoalgorithms and correlation of the received results with distribution of full differentials for random substitutions of similar dimension. As indicators we will use the maximum value of differential probability (DP_{max}^{f}) and average maximum value of differential probability (ADP_{max}^{f}) for key dependent function f. For random substitution of degree 2^{Lblock} in [5] a theoretical assessment for ADP_{max}^{f} in the form

$ADP^f_{max} \leq (L_{block} + 4)$ was proposed.

For experiment the 4 fixed bijective substitutions were created for $M = 4$ bits (tab. 1) in a random way.

Table 1: Fixed substitution components *PD-sbox*.

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$sbox_0(X)$	5	B	E	F	6	A	0	8	C	D	1	4	2	9	7	3
$sbox_1(X)$	7	8	6	0	B	E	A	C	5	4	1	9	D	F	3	2
$sbox_2(X)$	E	5	4	F	8	0	6	D	A	7	3	B	2	C	1	9
$sbox_3(X)$	4	5	1	B	E	3	6	2	7	0	A	C	9	D	F	8

Depending on the configuration in structure, *PD-sbox* includes from 2 to 4 fixed substitutions ($N_{rows} = 2 \dots 4$). Taking into account restriction $N_{words} = N_{rows}$, rhe length of the block $L_{block} = N_{words} \cdot M$ was 8, 12 and 16 bits respectively. For each configuration differential properties were defined for 100 keys. Results are shown in the columns "Collapser" of tables 2-4.

In addition, for the analysis of dynamics of change of differential characteristics, as output values we took values of a state $s^{nr}_{i,j}$ on a certain step, to the corresponding number of substitution of *PD-sbox* in structure of cryptoalgorithm (see fig. 2). So, $step = 0$ corresponds to a condition of the initial *PD-sbox* located in the top left corner of structure in figure 2. The number of a step was defined as $step = (nr - 1) \cdot N_{words} + i$. The received results are given in the columns of the same title.

For comparison, the results obtained in the column "Random subst" shows the values for 100 randomly generated bijective substitutions of the corresponding dimensions.

Table 2: Values of maximum of tables of differences for a case $N_{words} = N_{rows} = 2$ and M=4 ($L_{block} = 8$ bit)

	step=2	step=3	step=4	step=5	Collapser	Random subst.
	nr=2	nr=2	nr=3	nr=3		
ADP^f_{max}	256	50,82	12,56	**11,28**	**11,2**	**11,44**
DP^f_{max}	256	64	18	**14**	**14**	**16**

Table 3: Values of maximum of tables of differences for a case $N_{words} = N_{rows} = 3$ and M=4 ($L_{block} = 12$ bit)

	step=5	step=6	step=7	step=8	Collapser	Random subst.
	nr=2	nr=3	nr=3	nr=3		
ADP^f_{max}	141,04	124,68	**15,34**	**15,34**	**15,52**	**15,5**
DP^f_{max}	226	170	**18**	**18**	**20**	**20**

Table 4: Values of maximum of tables of differences for a case $N_{words} = N_{rows} = 4$ and M=4 ($L_{block} = 16$ bit)

	step=7	step=8	step=9	Collapser	Random subst.	babyAES
	nr=2	nr=3	nr=3			
ADP^f_{max}	81,78	**19,06**	**19,24**	**18,98**	**19**	**19,326**
DP^f_{max}	192	**22**	**24**	**22**	**22**	**20**

7 Conclusions

The results (tables 2-4) agree with the principles laid down in structure of "Collapser":

1. The first round ($nr = 1$) of encryption serves for the primary set of uncertainty and nonlinearity of values of states $s^{nr}_{i,j}$. Because the substitutions of *PD-sbox* are switching consistently, the greatest uncertainty is achieved on the last step of a round.

2. On the second round ($nr = 2$) encryption due to input of the state value $s^{nr}_{i,j}$ from the first round, there are interrelations between vector elements $b = \{b_0, b_1, \ldots, b_{Nwords-1}\}$, that leads to the intensive growth of nonlinearity of transformation. However, second round is not enough for achieving of necessary nonlinearity of transformation. For example, there is a short way (in only one substitution of *PD-sbox*) between values b_0 and $m_{(Nwords-1)}$ through value of a state $s^2_{0,j} = s^1_{Nwords,j}$.

3. In the third round ($nr = 3$) of encryption sufficient complexity of the interrelations between all the elements of the vector of the ciphertext $c = \{c_0, c_1, \ldots, c_{Nwords-1}\}$ is achieved . The minimum distance between values c_0 and $m_{(Nwords-1)}$ through values forms N_{words} many steps or substitutions of *PD-sbox*. High nonlinearity is confirmed by that on the junction of the

2nd and 3rd rounds (at the beginning of the 3rd round), distribution of differentials for values $s_{i,j}^{nr}$ of states comes closer to distribution of random substitutions. Regularity remains for various lengths of the block L_{block}.

4. Despite existence of considerable deviations in distribution of differentials for values $s_{i,j}^{nr}$ of a state of the first steps of the 3rd round (step=3 for tables 2 and step=6 for table 3) the final differentials distribution for the values of the ciphertext (column "Collapser") is indistinguishable from random distribution substitutions. That confirms property of masking of values of a state $s_{i,j}^{nr}$ due to introduction of uncertainty upon transition from bigger dimension to smaller ($2^{M \cdot Nrows} \to 2^M$) when forming output values.

The presented results need to be considered as an initial assessment of differential properties of the proposed structure of cryptoalgorithm of "Collapser" demanding additional research.

8 Acknoledgements

The work is supported by Ministry of Education and Science of the Russian Federation. Theme № 213.01-11 / 2014-9.

References

[1] Preneel B., Biryukov A., Canniere C. De et al. Final report of European project number IST-1999-12324, named New European Schemes for Signatures, Integrity, and Encryption. — Berlin Heidelberg NewYork London Paris Tokyo Hong Kong Barcelona Budapest : Springer-Verlag, 2004.

[2] Matsui Mitsuru. Linear Cryptoanalysis Method for DES Cipher // Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings. — 1993. — P. 386–397. — URL: http://dx.doi.org/10.1007/3-540-48285-7_33.

[3] Biham Eli, Shamir Adi. Differential Cryptanalysis of DES-like Cryptosystems // J. Cryptology. — 1991. — Vol. 4, no. 1. — P. 3–72. — URL: http://dx.doi.org/10.1007/BF00630563.

[4] Braeken An, Nikov Ventzislav, Nikova Svetla, Preneel Bart. On Boolean Functions with Generalized Cryptographic Properties // Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings. — 2004. — P. 120–135. — URL: http://dx.doi.org/10.1007/978-3-540-30556-9_11.

[5] Dolgov V.I., Kuznetsov A.A., Isaev S.A. Differentsialnyie svoystva blochnyih simmetrichnyih shifrov // Elektronnoe modelirovanie. — 2011. — T. 33, № 6. — C. 81–99.

[6] Gorbenko I.D., Dolgov V.I., Lisitskaya I.V., Oleynikov R.V. Novaya ideologiya otsenki stoykosti blochnyih simmetrichnyih shifrov k atakam differentsialnogo i lineynogo kriptoanaliza // Prikladnaya radioelektronika. — 2010. — T. 9, № 3. — C. 312–320.

[7] Kazymyrov O., Oliynykov R. Application of vectorial Boolean functions for substitutions generation used in symmetric cryptographic transformation // In Systems of information processing. — 2012. — Vol. 6, no. 104. — P. 97–102.

[8] Logachev O.A., Salnikov A.A., Yaschenko V.V. Bulevyi funktsii v teorii kodirovaniya i kriptologii. — M. : Moskovskiy tsentr nepreryivnogo matematicheskogo obrazovaniya, 2004. — C. 470.

[9] Ivanov Georgi, Nikolov Nikolay, Nikova Svetla. Reversed Genetic Algorithms for Generation of Bijective S-boxes with Good Cryptographic Properties // IACR Cryptology ePrint Archive. — 2014. — Vol. 2014. — P. 801. — URL: http://eprint.iacr.org/2014/801.

[10] Beelen Peter, Leander Gregor. A new construction of highly nonlinear S-boxes // Cryptography and Communications. — 2012. — Vol. 4, no. 1. — P. 65–77. — URL: http://dx.doi.org/10.1007/s12095-011-0052-4.

[11] Fu Shaojing, Matsuura Kanta, Li Chao, Qu Longjiang. Construction of highly nonlinear resilient S-boxes with given degree // Des. Codes Cryptography. — 2012. — Vol. 64, no. 3. — P. 241–253. — URL: http://dx.doi.org/10.1007/s10623-011-9568-z.

[12] Kazymyrov Oleksandr, Kazymyrova Valentyna, Oliynykov Roman. A Method For Generation Of High-Nonlinear S-Boxes Based On Gradient Descent // IACR Cryptology ePrint Archive. — 2013. — Vol. 2013. — P. 578. — URL: http://eprint.iacr.org/2013/578.

[13] Tokareva N.N. Generalizations of bent functions. A survey. // Diskretn. Anal. Issled. Oper. — 2010. — Vol. 17, no. 1. — P. 34–64.

[14] Zaibi Ghada, Peyrard Fabrice, Kachouri Abdennaceur et al. A new design of dynamic S-Box based on two chaotic maps // The 8th ACS/IEEE International Conference on Computer Systems and Applications, AICCSA 2010, Hammamet, Tunisia, May 16-19, 2010. — 2010. — P. 1–6. — URL: http://dx.doi.org/10.1109/AICCSA.2010.5586946.

[15] Ahmad Musheer, Khan Parvez Mahmood, Ansari Mohd. Zeeshan. A Simple and Efficient Key-Dependent S-Box Design Using Fisher-Yates Shuffle Technique // Recent Trends in Computer Networks and Distributed Systems Security - Second International Conference, SNDS 2014, Trivandrum, India, March 13-14, 2014, Proceedings. — 2014. — P. 540–550. — URL: http://dx.doi.org/10.1007/978-3-642-54525-2_48.

[16] Pradeep L. N., Bhattacharjya Aniruddha. Random Key and Key Dependent S-box Generation for AES Cipher to Overcome Known Attacks // Security in Computing and Communications - International Symposium, SSCC 2013, Mysore, India, August 22-24, 2013. Proceedings. — 2013. — P. 63–69. — URL: http://dx.doi.org/10.1007/978-3-642-40576-1_7.

[17] Hosseinkhani Razi, Haj H., Javadi Seyyed et al. Using Cipher Key to Generate Dynamic S-Box in AES Cipher System. — 2012.

[18] Mahmoud Eman Mohammed, Abd Ahmed, Hafez El et al. Dynamic AES-128 with Key-Dependent S-box. — 2013. — URL: `www.ijera.com`.

[19] Schneier Bruce. Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish) // Fast Software Encryption, Cambridge Security Workshop. — London, UK, UK : Springer-Verlag, 1994. — P. 191–204. — URL: `http://dl.acm.org/citation.cfm?id=647930.740558`.

[20] Kuznetsov A.A., Sergienko R.V., Nausko A.A. Simmetrichnyiy kriptograficheskiy algoritm ADE (Algorithm of Dynamic Encryption) // Prikladnaya radioelektronika. — 2007. — T. 6, № 2. — C. 241–249.

[21] Bogdanov A., Knudsen L. R., Leander G. et al. PRESENT: An Ultra-Lightweight Block Cipher // Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems. — CHES '07. — Berlin, Heidelberg : Springer-Verlag, 2007. — P. 450–466. — URL: `http://dx.doi.org/10.1007/978-3-540-74735-2_31`.

[22] Borghoff Julia, Knudsen Lars R., Leander Gregor, Thomsen Søren S. Cryptanalysis of PRESENT-Like Ciphers with Secret S-Boxes. // FSE / Ed. by Antoine Joux. — Vol. 6733 of Lecture Notes in Computer Science. — Springer, 2011. — P. 270–289.

[23] Polikarpov S.V., Rumyantsev K.E., Kozhevnikov A.A. Psevdodinamicheskie tablitsyi podstanovki: osnova sovremennyih simmetrichnyih kriptoalgoritmov // Nauchnoe obozrenie. — 2014. — № 12. — C. 162–166. — URL: `http://www.sced.ru/ru/files/7_12_1_2014/7_12_1_2014.pdf`.

[24] Polikarpov S.V., Rumyantsev K.E., Kozhevnikov A.A. Psevdodinamicheskie tablitsyi podstanovki: issledovanie differentsialnyih harakteristik // Fiziko-matematicheskie metodyi i informatsionnyie tehnologii v estestvoznanii, tehnike i gumanitarnyih naukah: sbornik materialov mezhdunarodnogo nauchnogo e-simpoziuma. Russia, Moscow, december 27-28, 2014 g. — Kirov : MTsNIP, 2015. — C. 77–89.

On a Secure Connection Establishment Protocol for a Functional Key Carrier

Evgeny Alekseev Vasily Nikolaev Igor Oshkin
Vladimir Popov Anton Prokhorov Stanislav Smyshlyaev
Lolita Sonina

Abstract

In the current paper we consider cryptographic properties of a secure connection protocol, designed for interacting with functional key carrier (or FKC). It is based on key establishment protocol using low-entropy pre-shared secret (password). The considered protocol is based on well-known SPAKE1 protocol modified for interacting with FKC and using Russian cryptographic algorithms. In the current paper we describe some features of protocol construction and provide security evaluation results in a certain adversary model.

1 Introduction

The safety of using digital signature mechanisms depends entirely on keeping private key secret. There exist different methods that can be used for this purpose. For a trivial example, a private key can be stored on a user computer protected by a local password, but this option has two disadvantages. First, the user can only sign documents on that particular computer. Second, the security of the private key depends entirely on the security of the user computer.

One of the safest ways to store private keys is putting them on a smart card. Smart cards lack disadvantages mentioned above. They allow user to use his private key on other computers. Typically, the user must activate his smart card by entering a personal identification number or a PIN-code. The number of attempts to enter the PIN-code is limited, which makes smart cards resistant to brute force attacks.

There are two widely distributed types of smart cards. Cards of the first type, which can be called passive, just store a private key. After entering the correct PIN-code such cards transmit the private key to a target application which uses the key to compute a digital signature. Passive smart cards have one big disadvantage: after transferring the private key it's security depends entirely on security properties of the user computer. Also, such cards are vulnerable to a passive adversary who can sniff the traffic between the computer and the smart card and obtain user PIN-code and key from sniffed data.

The second type is a class of active smart cards. They operate as follows: the hash calculated from the document is sent to the smart card, whose CPU signs the hash using the stored private key of the user and then returns the signed hash. A passive adversary in this case can't obtain user key from sniffed data but he can still obtain user PIN-code. So, if an adversary steals a smart card, he can sign arbitrary data as legal user without knowing the key. Also, passive smart cards are vulnerable to an active adversary having an ability to send arbitrary messages into the channel.

In the current paper we describe a smart card connection protocol that is resistant against an active adversary ([9]). It's central element is modified SPAKE1 protocol [1]. The important feature of this type of protocols is an additional requirement of an active opponent inability to obtain information that allows him password exhaustive search later without additional interaction with participants. In the current paper we present clarifications concerning protocol structural features, then we describe an adversary model and give the result (and a sketch of the proof) of the security evaluation in this model.

2 Notation

By V_n we will denote the set of n-component vectors from $GF(2)$. By $[n]$ we will denote the set $\{1, 2, \ldots, n\}$.

We assume further that all elliptic curve point operations are performed in the prime order q subgroup E of some elliptic curve point group. We will denote the base point as P and the order of the full group as m. We will use τ for denoting the complexity of scalar point multiplication. By 0_E we will denote the neutral element of the group E, $E^* = E \setminus 0_E$.

F denotes the PBKDF2 function, defined in [8]. H_{256} is used for the National Standard of Russian Federation GOST R 34.11-2012 hash function (also known as "Streebog", [7]) with output of 256 bits. E_K, $Imit_K$ and D_K are used respectively for encryption, MAC and decryption algorithms as described in GOST 28147-89 national standard ([5]) used with secret key K.

3 Protocol description

The protocol is supposed to be run by two parties, which we will denote as *protocol subjects*. Every subject stores some secret parameter. The subject who stores password as his secret parameter will be called *the client*, the other who stores point Q_{PW} as a secret will be called *the server*. If A is a protocol subject, then $A_{ID} \in ID$ is his identifier (which is in fact a fixed-length bit string). The client and the server (A and B in protocol scheme) store two provably pseudorandom points Q and P respectively (for detailed comments see 4.3). The client remembers his password $PW \in V_8^k$. The server stores the following parameters: $r \in [2^{64} - 1]$, $salt \in V_l$ and the point $Q_{PW} = F(PW, salt, 2000) \cdot r \cdot Q$ (the last parameter is kept secret). T_A, T_B — some fixed constants. The protocol scheme is given in the table 1. Comments and clarifications on some protocol features are provided in Section 4.

The idea which the subprotocol EKE_{KA} is based on is taken from [1]. The additional part of protocol EKE (or EKE_{KC}) is made due to the necessity of the checking that EKE_{KA} is successfully finished. Generated common secret can be used for further secure communication as encryption and authentication key (MAC by GOST 28147-89).

4 Structural features of the protocol

In this section we discuss structural features of the EKE protocol and their impact on it's properties.

A	**B**	
$A_{ID} \longrightarrow$		EKE_{KA}
	$\longleftarrow (r, salt)$	
$Q' = r \cdot Q$		
$Q^A_{PW} = F(PW, salt, 2000) \cdot Q'$		
$\alpha \in_R [q-1]$		
$u_1 = \alpha \cdot P - Q^A_{PW} \longrightarrow$		
	Quit if $u_1 \notin E$	
	$Q_B = u_1 + Q_{PW}$	
	$\beta \in_R [q-1], R \in_R E^*, UKM \in_R [2^{128}-1]$	
	if $\frac{m}{q} Q_B = 0_E$, then $Q_B = R$	
	$K_B = H_{256}((UKM \cdot \frac{m}{q} \cdot \beta \mod q) Q_B)$	
	$\longleftarrow u_2 = \beta \cdot P + Q_{PW}, UKM$	
Quit if $UKM \notin [2^{128}-1]$ or $u_2 \notin E$		
$Q_A = u_2 - Q^A_{PW}, R \in_R E^*$		
if $\frac{m}{q} Q_A = 0_E$, then $Q_A = R$		
$K_A = H_{256}((UKM \cdot \frac{m}{q} \cdot \alpha \mod q) Q_A)$		
$C_A = E_{K_A}(T_A) \longrightarrow$ $I_A = Imit_{K_A}(T_A) \longrightarrow$		EKE_{KC}
	Verification: $D' = D_{K_B}(C_A) \overset{?}{=} T_A$	
	Verification: $Imit_{K_B}(D') \overset{?}{=} I_A$	
	$SID \in_R V_8^4$	
	$\longleftarrow C_B = E_{K_B}(T_B \| SID)$ $\longleftarrow I_B = Imit_{K_B}(T_B \| SID)$	
Verification: $D' = D_{K_A}(C_B) \overset{?}{=} T_B$		
Verification: $Imit_{K_A}(D') \overset{?}{=} I_B$		

Table 1: Protocol *EKE* description

4.1 Counters of unsuccessful authentication attempts

Informally speaking, the *EKE* protocol is considered to be secure if the best way to recover the session key (or some information about it) is password guessing via interaction with network subject (further for clarity this method will be called ”online” password guessing). The alternative to ”online” guessing which is unacceptable for secure protocol is reasonably practical password exhaustive search that does not require such an interaction (”offline” guessing).

During the "online" guessing every unsuccessful attempt results in the break of connection caused by checks within the protocol EKE_{KC}. An absence of limitations on the total number of unsuccessful attempts leads to a sufficient increase of "online" guessing probability. That is the reason why the EKE protocol must provide the restriction on the total number of possible unsuccessful connection attempts. This counter is decremented as the first server request is received. If the connection is successfully established, then this counter is set to it's highest value (set by default — usually it's not greater than 10). In addition to the counter of consequential unsuccessful authentication attempts it is necessary to limit the overall number of such events. It provides the ability to prevent "online" password guessing between successful connections.

4.2 Processing of neutral point case

This subsection explains the reasons why the cases of key agreement when $\frac{m}{q}Q_B = 0_E$ (or $\frac{m}{q}Q_A = 0_E$) are processed in a special way. In this particular case "special processing" means random point R generation independent of Q_B value and it's usage as Q_B if $\frac{m}{q}Q_B = 0_E$.

The protocol where the specified case is not treated specially may be vulnerable to "online" password guessing if usage of unsuccessful authentication attempts counters is not strict. Let us explain this statement.

Let the counter of unsuccessful authentication attempts decrease in the beginning of the EKE_{KC} protocol and the case of $\frac{m}{q}Q_B = 0_E$ be not treated specially. Then the adversary imitating the A-side sends the B-side points $u_1 = X - Q_{PW'}$ with $\frac{m}{q}X = 0_E$ for different PW' and measures the time of the B-side response. After the B-side response is received the adversary breaks the connection. By this we mean that the B-side doesn't decrease the counter of unsuccessful connection attempts and the time measured gives an adversary a criterion for identification of the correct password — for the correct password this time will be substantially smaller because the multiple point computation appears to be degenerate.

If the subjects process the neutral point according to the protocol description but they generate R only if it is needed, then the adversary can also distinguish the correct password from the incorrect through the supplementary actions of R generation for the case of $PW = PW'$.

The time difference in conditional test of $\frac{m}{q}Q_B = 0_E$ for the cases $Q_B = 0_E$ and $Q_B \neq 0_E$ is insignificant because according to [6] value of $\frac{m}{q}$ is extremely small.

Any of the methods described above become inapplicable if the subjects decrease the counters of unsuccessful authentication attempts at the very beginning.

Additionally we note that if an adversary imitates the B-side, then he can keep UKM value under control. However he can't make it equal to 0 to ensure degeneracy of multiple point computation because the A-side detects this possibility.

4.3 Pseudorandomness of points P and Q

The points P and Q must be chosen in such a way that the multiplicity of each of them relating to any other is unknown and the complexity of multiplicity computation must be comparable to complexity of a solution to the discrete logarithm problem in the group of points of the used elliptic curve. The fulfillment of this condition can be gained by choosing the specified points according to the conception of provable pseudorandomness [4]. Thus the choice of a point of an elliptic curve in Weierstrass form ($y^2 = x^3+ax+b$) can be carried out in the following manner: generate random string s and assume $x = H(s)$ for as long as the value of the expression $x^3 + ax + b$ stays quadratic nonresidue; set $y = \sqrt{x} \mod p$. For example, hash function GOST R 34.11-2012 can be used here as H.

If the multiplicity of Q relating to P is known, then the adversary imitating server can get the criterion for incorrect password rejection (therefore he can guess password ”offline”). For that end it is sufficient to interact with a client only once. We outline the possible sequence of the adversary's actions. For clarity, in the description of key agreement procedure we assume that parameters UKM and $\frac{m}{q}$ are equal to 1.

If the multiplicity of Q relating to P is known, then the multiplicity of Q'

relating to P is also known: let $Q' = \gamma P$, let $z = F(PW, salt, 2000)$. Then in response to a message $u_1 = \alpha P - z\gamma P$ an adversary sends a message $u_2 = \beta P + z'\gamma P$, where β and z' are values arbitrarily chosen by the adversary. The next client message is a known fixed string encrypted on the key $K = H_{256}\left(\alpha\left(\beta + z'\gamma - z\gamma\right)P\right)$, which is unknown for the adversary. Correctness of decryption of this message is the criterion for incorrect session key rejection and the relation

$$K = H_{256}\left(\left(\beta + z'\gamma - z\gamma\right) \cdot \left(u_1 + z\gamma P\right)\right),$$

in the right part of which only z value depending on password is unknown for the adversary, gives the opportunity to guess the password.

If the multiplicity of Q' relating to P is unknown, then the adversary has to determine α, because the key will be calculated using a formula $K = H_{256}\left(\alpha\beta P + \alpha r' Q' - \alpha r Q'\right)$.

5 Considered models

In this section we will introduce a brief description of the considered models. A formal detailed description can be found, for example, in [1] and [3].

When considering a model we suppose there exists a network, which consists of users interacting with each other using *EKE* protocol only. Every subject can interact in only one protocol session in any moment. The subjects that interact in the session are modelled using oracles.

5.1 Protocol $\boldsymbol{EKE_{KA}}$ adversary model

In this section an active adversary model (that is, we consider an adversary who controls all the communication channels between all network users and has, besides, an ability to obtain some successfully established connection keys) is being considered.

Let us consider an adversary who has no ability to make queries to the oracles modelling network users, but still has an ability to make queries to five special oracles. Four of these oracles describe his network interacting abilities, while the fifth one models an ideal necessary primitive — a hash function.

We won't go deep into details for O_{exec}, O_{send} and O_{reveal} oracles, but we'll mention that they are needed to model passive channel eavesdropping, active channel intruding and established session key revealing respectively (the last is possible if, for example, keys are used in unappropriate way). We would like to describe the O_{test} oracle in a more detailed way, because we formulate the threat using it. The adversary queries O_{test} only once and gets from it either session key of desired user or a random bit string (it depends on a random bit, which is set by O_{test}). It's prohibited to ask this oracle for the session key, which was got from O_{reveal}. The threat is if the adversary could guess what has been returned by O_{test}: the session key or a random bit string.

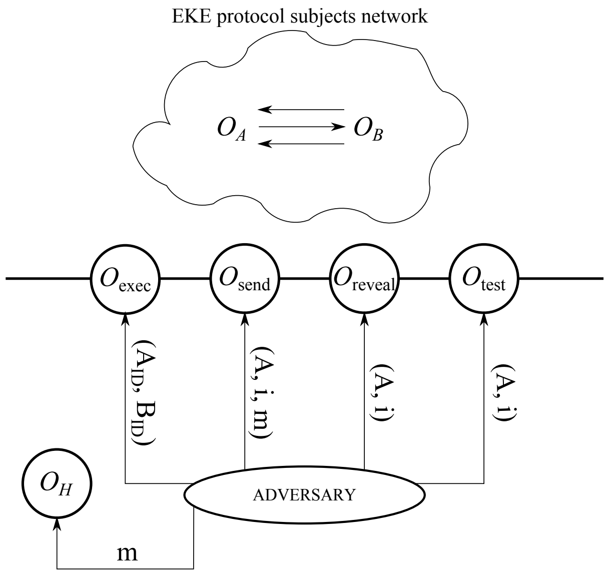

Figure 1: Protocol EKE_{KA} adversary model

Procedure 5.1. *We consider the procedure, in which an adversary $\mathcal{A}$ has access to $\mathcal{O}_{exec}$, $\mathcal{O}_{send}$, $\mathcal{O}_{reveal}$, $\mathcal{O}_H$ and $\mathcal{O}_{test}$ oracles and can make a query to $\mathcal{O}_{test}$ only once. As a result, an adversary $\mathcal{A}$ returns either 0 or 1. Let $SUCC$ denote an event, when the value returned by the adversary matches bit b, which was used by $\mathcal{O}_{test}$ oracle when working.*

By the advantage of the adversary $\mathcal{A}$, who is trying to solve the distinguishing problem described above, we will denote

$$\mathbf{Adv}(\mathcal{A}) = |2 \cdot \Pr\{SUCC\} - 1|$$

. For the adversary, who solves some computational problem, by $\mathbf{Adv}(\mathcal{A})$ we will denote the probability of success. It's always clear from the context, which specific problem is considered.

For positive integers t, q_H, q_{send}, q_{exec} and q_{reveal} by

$$\mathcal{A}_{Task}(t, q_H, q_{send}, q_{exec}, q_{reveal})$$

we will denote the set of adversaries, operating in less than t steps, making less than q_H, q_{send}, q_{exec} and q_{reveal} queries to corresponding oracles and solving $Task$ problem. By $\mathbf{Adv}_{Task}(t, q_H, q_{send}, q_{exec}, q_{reveal})$ we will denote

$$\mathbf{Adv}_{Task}(t, q_H, q_{send}, q_{exec}, q_{reveal}) = \max_{\mathcal{A} \in \mathcal{A}_{Task}(t, q_H, q_{send}, q_{exec}, q_{reveal})} \mathbf{Adv}(\mathcal{A}).$$

When we consider a set of adversaries, limited not by all the parameters (or some limitations have no sense for considered problem), the redundant parameters are omitted. For example, $\mathbf{Adv}_{Task}(t)$ is a set of adversaries, who operate faster than t steps.

5.2 *EKE* protocol adversary model

EKE protocol adversary model is mostly the same as model for EKE_{KA} protocol, so we are going to describe only the difference.

5.2.1 Key revealing adversary model

For *EKE* protocol a key revealing adversary model is being considered. Besides the abilities which the adversary, who was considered for EKE_{KA} analysis, has, here the adversary can make queries (K) to $\mathcal{O}_{A,cipher}$ and $\mathcal{O}_{B,cipher}$ oracles, which return strings $F_{T_A}(K)$ and $F_{T_B}(K)$ respectively. These strings are values of two random functions F_{T_A} and F_{T_B}. Usage of these oracles helps to model encryption of the strings T_A and T_B with key K. Instead of O_{test} oracle O^K_{test} oracle is being considered, which receives string S as an input and session identifier and returns 1 if this string is the indicated session key and 0 otherwise. In this case the main of the protocol security characteristics is the probability $\mathbf{Adv}(\mathcal{A})$ that the adversary $\mathcal{A}$ can get the session key.

5.3 Certain practical adversary models

This section is devoted to show the fact that the model described above includes all common practically considered adversary models. *EKE* protocol is designed for being used mostly for connection between key carrier (smart card) and a client (computer). In this case the client stores the password, while the card stores Q_{PW} point. Further in this section we will say *"client"* and *"card"* to denote protocol participants.

Example 5.2. *Adversary is a client. Attack: the adversary acts as a client before card. It can send the card any messages and receive it's answers. Threat: the adversary is successfully authenticated by the card.*

This example is fully covered by the oracles described in 5.1. In this case the adversary addresses $\mathcal{O}_{send}$ oracle in order to send the card some messages, impersonating the client. The successful authentication of the adversary by the card implies establishing common secret key between them. That means the adversary has solved a problem, which is at least as hard as distinguishing session key from a random bit string.

Example 5.3. *An adversary is a man-in-the-middle. The adversary can intercept any messages the participants exchange. It can also change and hold these messages, as well as send messages on his own choice to every protocol participant impersonating any other participant. Threat: the adversary obtains some information about the session key.*

We would like to notice that all the adversary's opportunities of arbitrary manipulating transmitted messages are covered by his ability to address $\mathcal{O}_{send}$ oracle, which was described in 5.1. Thus, «man-in-the-middle» adversary is at most as powerful as an adversary having access to $\mathcal{O}_{send}$. The implementation of the desired threat is then at least as hard as distinguishing the session key from a random bit string.

Finally, we would like to mention another example.

Example 5.4. *Adversary is a card. Suppose, the adversary is a subverted card, which is used to obtain client's password (or, at least, Q_{PW} point).*

In this case, the card can interact with the client as a normal card. This behaviour is fully covered by using the $\mathcal{O}_{send}$ oracle. If the adversary card obtains client's password, then it obtains all the future session keys, established between client and the adversary card. This implies that the adversary card can distinguish the session key from a random bit string.

It's worth noticing that $\mathcal{O}_{reveal}$ oracle should be also considered due to a future possible incorrect usage of the session key (for example, as a key for a weak cryptographic algorithm).

6 The *EKE* protocol security

The EKE protocol security is estimated in two stages. At first the EKE_{KA} protocol security in relation to the threat of key distinguishing is evaluated. Then through the usage of the obtained estimation the EKE protocol security is evaluated in relation to the threat of key revealing.

Further all problems are formulated for subgroup E of elliptic curve point group with $|E| = q$ where q is prime.

6.1 The evaluation of the EKE_{KA} protocol security

Further q_{exec}, q_{send}, q_H denote the number of queries to the oracles $\mathcal{O}_{exec}$, $\mathcal{O}_{send}$, $\mathcal{O}_H$ respectively that are made by the adversary $\mathcal{A}_{EKE_{KA}}$. $\mathcal{D}$ is the "vocabulary" that contains the password.

The treatment of a simple case when the adversary actually tries to guess the password of one of the connections leads to a lower bound of the EKE_{KA} protocol security.

Theorem 6.1. *For $q_{send} \geqslant 2$ and some t, sufficient for performing of one interaction by one client according to protocol, the following inequality is valid:*

$$\mathbf{Adv}_{EKE_{KA}}(t, q_{send}) \geqslant \frac{1}{|\mathcal{D}|} - \frac{1}{q}.$$

The EKE_{KA} protocol security is based on the complexity of solving computational Diffie-Hellman problem (CDH): given a nonzero point $P \in E$ and points $X = xP$ and $Y = yP$ $(x, y \in_R \mathbb{Z}_q)$, find $Z = xyP$. Further $\mathbf{Adv}_{\text{CDH}}(\mathcal{A})$ denotes the probability of success for adversary $\mathcal{A}$. The characteristic reflecting the complexity of solving CDH problem is the value of $\mathbf{Adv}_{\text{CDH}}(t) = \max_{\mathcal{A}\in\mathcal{A}(t)} \mathbf{Adv}_{\text{CDH}}(\mathcal{A})$. For given $X = xP$ and $Y = yP$ we will denote xyP as $\text{CDH}_P(X, Y)$. When P is clear in context it will be denoted as $\text{CDH}(X, Y)$.

Theorem 6.2. *The following inequality is valid:*

$$\begin{aligned}\mathbf{Adv}_{EKE_{KA}}(t, q_H, q_{send}, q_{exec}, q_{reveal}) \leqslant \\ \leqslant \frac{2q_{send}}{|\mathcal{D}|} + \frac{(2q_{exec} + q_{send})^2}{q} + 2q_H\mathbf{Adv}_{\text{CDH}}(t + 2\tau q_{exec}) + \\ + 2q_{send}\sqrt[4]{\sqrt{\mathbf{Adv}_{\text{CDH}}(8t + 8q_{S1}\tau + 2\Theta + O(q_H\tau))} + \frac{8q_H^4}{q}},\end{aligned}$$

where C is some constant, τ is the complexity of evaluation of multiple point in group E, q_{S1} is the number of queries to the oracle $\mathcal{O}_{send}$ in the form $(r, salt)$, Θ is the complexity of finding two equal elements in two sets of a size q_H^2.

Let us give a comment on the estimate presented in the theorem. As was shown above, the lower bound of the advantage $\mathbf{Adv}_{EKE_{KA}}(t, q_{send})$ is of the order $|\mathcal{D}|^{-1}$. The estimate obtained in the theorem shows that if the CDH problem is intractable then the advantage in the problem of distinguishing key from random string has the order $|\mathcal{D}|^{-1}$ (with limitations on the number of unsuccessful authentications).

The basic ideas of the proof are similar to ones in [1], adapted for the case of interacting subjects using one point Q.

6.2 The complete *EKE* protocol security estimate

As a final security result that is proven using the above statements we conclude with the following theorems.

Theorem 6.3. *The following inequality holds:*

$$\mathbf{Adv}_{K,EKE}(t, q_{exec}, q_{send}, q_{reveal}, q_H) \leqslant 2\mathbf{Adv}_{K,EKE_{KA}}(t, q_{exec}, q_{send}, q_{reveal}, q_H).$$

Theorem 6.4. *The following inequality holds:*

$$\mathbf{Adv}_{K,EKE_{KA}}(t, q_{exec}, q_{send}, q_{reveal}, q_H) \leqslant \frac{1}{2^n} + \mathbf{Adv}_{EKE_{KA}}(t, q_{exec}, q_{send}, q_{reveal}, q_H).$$

References

[1] Abdalla M., Pointcheval D. Simple Password-Based Encrypted Key Exchange Protocols. Proc. of Topics in Cryptology - CT-RSA 2005, LNCS 3376, pp. 191-208, Springer-Verlag.

[2] Bao F., Deng R.H., Zhu H. Variations of Diffie-Hellman problem. In: Qing, S., Gollmann, D., Zhou, J. (eds.) ICICS 2003. LNCS, vol. 2836, pp. 301–312. Springer, Heidelberg (2003).

[3] Bellare M., Pointcheval D., Rogaway P. Authenticated Key Exchange Secure against Dictionary Attacks. Advances in Cryptology — EUROCRYPT 2000, Lecture Notes in Computer Science Volume 1807, 2000, pp 139-155.

[4] Lochter M., Merkle J., Schmidt J-M., Schutze T. Requirements for Standard Elliptic Curves. Cryptology ePrint Archive, Report 2014/832.

[5] «Cryptographic Protection for Data Processing System», GOST 28147-89, Gosudarstvennyi Standard of USSR, Government Committee of the USSR for Standards, 1989.

[6] «Information technology. Cryptographic data security. Formation and verification processes of [electronic] digital signature», GOST R 34.10-2012, Federal Agency on Technical Regulating and Metrology,

[7] «Information technology. Cryptographic Data Security. Hashing function», GOST R 34.11-2012, Federal Agency on Technical Regulating and Metrology, 2012.

[8] «Password security using GOST algorithms», Technical Commitee for standardization «Cryptography and security mechanisms» (TC 26) (in Russian).

[9] «The password-based authentication protocol with key establishment», Standardization recommendations, first redaction draft, November 2014, rus-cp-eke-gost-00-rd (in Russian).

RSA-like Cryptosystem using Dedekind Rings

Sergey Tronin Kseniya Petukhova

Abstract

The aim of this paper is to establish necessary conditions for the maximum possible generalization of the RSA algorithm. We substitute ideals of a Dedekind ring for integers.

Keywords: RSA, Dedekind ring, ideal, Euler's function

1 Euler's φ-function for a Dedekind rings

Let us remember that the power of a quotient ring R/A is denoted by $N(A)$ and is called the norm of ideal $A \in R$.

Lemma 1. *Let R be a Dedekind ring, and $R/P < \infty$ for every maximal ideal P. Then $|R/A| = N(A) < \infty$ for every ideal A. With that if $A = P^n$, then $N(P^n) = N(P)^n$.*

Proof. Since $P^n/P^{n+1} \cong R/P$ ([6, p. 13,(5)]), $|P^n/P^{n+1}| = |R/P| = N(P)$.

Proof is performed by the induction on n. Inductive hypothesis is $|R/P^n| = N(P)^n$. Case $n = 1$ comes from the hypothesis of lemma. In case $n = 2$ we have $A \subset B \subset R$. Then there exists homomorphism $f : R/A \to R/B$, and $Ker(f) = B/A$. $f : x + A \to x + B$. Since $P^2 \subset P$, $f : R/P^2 \to R/P$ is surjective, $|Ker(f)| = |P/P^2| = N(P)$. Above $|R/P^2| = |R/P| \cdot |P/P^2|$, hence $N(P^2) = N(P)^2$. Suppose $|R/P^n| = N(P)^n < \infty$. From $P^{n+1} \subset P^n$ we have surjective homomorphism $f : R/P^{n+1} \to R/P^n$, and $Ker(f) = P^n/P^{n+1}$. Hence $|R/P^{n+1}| \cong |R/P^n| \cdot |P^n/P^{n+1}| = N(P)^n \cdot N(P) = N(P)^{n+1}$.

From this point we will consider Dedekind rings taking into account the following condition: $|R/M| < \infty$ for every maximal ideal. In particular all rings of algebraic integers satisfy this condition. Herewith assuming the above, we can define the Euler's function analog for ideals of ring R:

$$\varphi(A) = |U(R/A)|,$$

where $U(R/A)$ is a group of units. This definition is correct by lemma 1.

If $|R/M| < \infty$, where M is a maximal ideal, then φ-function is defined for every ideal of a Dedekind ring R. If $A = M_1^{k_1} \cdot M_2^{k_2} \cdot \ldots \cdot M_m^{k_m}$, then $\varphi(A) = \prod\limits_{i=1}^{m} \varphi(M_i^{k_i})$.

Lemma 2. *Let O is a ring, N is a maximal ideal and all elements from N is nilpotent. Then O is a local ring.*

Proof. Let $u \in O$, $u \notin M$, then $Ou + M = O$. Hence $1 = wu + z$, where $w \in R$, $z \in M$. Then $wu = 1 - z$. But since z is nilpotent, than $1 - z$ is invertible. Hence u is invertible and u is not element of M. This means that O is a local.

Theorem 1.*Let R be a Dedekind ring such that $|R/M| = N(M) < \infty$ for every maximal ideal, then $\varphi(M^n) = N(M)^n - N(M)^{n-1}$.*

Proof. The ring R/M^n contains a maximal ideal M/M^n and every element of M/M^n is nilpotent. Hence R/M^n is a local ring by the lemma 2. This means that $U(R/N^n) = R/M^n \setminus M/M^n$. Transferring to powers we have $\varphi(M^n) = |R/M^n| - |M/M^n|$. From above $|R/M^n| = N(M)^n$. We still have $|M/M^n| = N(M)^{n-1}$ to prove.

The proof is by induction on n.

Case $n = 2$. $M/M^2 \cong R/M$ by [6, page 13,(5)].

Case $n + 1$: $M^{n+1} \subset M^n \subset M$ and as in lemma 1 we have: there exists homomorphism $g : M/M^{n+1} \to M/M^n$ and it is surjection with kernel $M^n/M^{n+1} \cong R/M$. Then we have $|M/M^{n+1}| = |M/M^n| \cdot |M^n/M^{n+1}| = N(M)^n \cdot N(M)$.

Theorem 2. (Generalized Euler's Theorem) *Let R be a ring with above conditions, $m \in R$, $A \subset R$ be an ideal. If $Rm + A = R$ then $m^{\varphi(A)} \equiv 1$ (mod A).*

Proof. By the condition $m + A$ is in $U(R/A)$.

Special cases of generalized Euler's function can be found in [1] and [5].

2 The Main Result

Let R be a Dedekind ring with the following conditions:

1. For every maximal ideal M quotient-ring R/M is finite.

2. It's possible to find a set $W \subset R$ for some maximal ideals $M_1, M_2 \in R$ and $A = M_1 \cdot M_2$ such that:

2.1. Every coset R/A intersects with W and this intersection contains only one element (i.e. W consists of different elements from co-sets R/A that's why we have one-by-one correspondence between W and R/A).

2.2. It's possible to find some convenient for calculations one-by-one correspondence between W and $[0, T] = \{0, 1, 2, \dots, T-1, T\}$ with sufficiently large natural number T.

Suppose Alice wishes to enable anyone to send her secret messages, which can only be decrypted by her. First, she picks two maximal ideals $M_1 \neq M_2 \subset R$. Then Alice computes $A = M_1 \cdot M_2$. Alice also chooses an encrypting exponent e, which satisfies $gsd(E, \varphi(A)) = 1$. Now Alice's public key is the pair (A, e), which she can publish in a public directory. Then Alice computes the decryption exponent d such as $ed = 1 + \varphi(A)t$. Alice keeps secret her privat key, which is the triple (d, M_1, M_2).

Now suppose Bob wishes to encrypt a message for Alice. He first looks up Alice's public key and represents the message as an element $m \in W$. The ciphertext c is then produced by raising the message to the power of the public encryption exponent modulo the public modulus, i.e.

$$c = m^e \pmod A \in W.$$

Alice on receiving c can decrypt the ciphertext to recover the message by exponentiating the ciphertext by the private decryption exponent, i.e.

$$m = c^d \pmod A \in W.$$

Theorem 3. $m^{ed} \equiv m \pmod A$ *for all* $m \in W$.

Proof. Case $Rm + A = R$. We can use Theorem 2: $m^{\varphi(A)} \equiv 1 \pmod{A}$. From here $m^{\varphi(A)t} \equiv 1 \pmod{A}$ and $m^{ed} = m \cdot m^{\varphi(A)t} \equiv m \pmod{A}$.

Case $Rm + A \neq R$, $m \neq 0$. Then the ideal $Rm + A$ contains $A = M_1M_2$ and therefore $M_1M_2 = (Rm + A)M'$ for some ideal M'. So we have $Rm + A = M_1$ or $Rm + A = M_2$. Let $Rm + A = M_1$.

From $Rm + A = M_1$ we get $m \in M_1$, hence $m^{ed} \equiv m \pmod{M_1}$.

Since $m \in W$, $m \neq 0$, $m \in M_1$, then $m \notin M_2$. From this we have $Rm + M_2 = R$, and by Theorem 2 we get $m^{\varphi(M_2)} \equiv 1 \pmod{M_2}$. Next we calculate

$$\begin{aligned}&m^{\varphi(M_1)\varphi(M_2)} = m^{\varphi(A)} \equiv 1 \pmod{M_2},\\ &m \cdot m^{\varphi(A)t} \equiv m \cdot 1 = m \pmod{M_2},\\ &m^{1+\varphi(A)t} = m^{ed} \equiv m \pmod{M_2}.\end{aligned}$$

From $m^{ed} \equiv m \pmod{M_1}$ and $m^{ed} \equiv m \pmod{M_2}$ we get $m^{ed} \equiv m \pmod{M_1M_2}$ by Chinese Remainder Theorem for rings.

3 Some known results and further perspectives

The RSA cryptosystem for $R \neq \mathbf{Z}$ was introduced earlier only for case $R = \mathbf{Z}[i]$. This was made in a series of papers: [2], [3] and so on. The authors of these papers have argued that cryptographic security for case $R = \mathbf{Z}[i]$ is greater than cryptographic security of usual RSA cryptosystem for $R = \mathbf{Z}$. At the same time the authors of paper [4] assert that this is not the case. We agree with the last authors. Moreover:

Theorem 4. *Let R be a quadratic ring of algebraic integers. Then cryptographic security of RSA for this R does not exceed cryptographic security of usual RSA cryptosystem.*

Proof. As known from [7, § 4.5], for every ideal $I \subseteq R$, there exists $a, b, d \in \mathbf{Z}$ such that the following are true:

(a) $I = d(\mathbf{Z}a + \mathbf{Z}(-b + \delta))$, and

(b) $b^2 - tb + n \equiv 0 \pmod{a}$, or, equivalently, $a|N(-b + \delta)$.

Conversely, any $I \subseteq R$ satisfying (a) and (b) is an ideal.

Herewith the following proposition holds ([7, p.84, proposition 4.9.10]): Let $I = \mathbf{Z}a + \mathbf{Z}(-b+\delta)$ be an ideal of R. Let $a = p_1^{e_1} \cdot \dots \cdot p_r^{e_r}$ be the prime factorization of a in $\mathbf{Z}$. The factorization of I into prime ideals is given by

$$A = \prod_{i=1}^{r} (\mathbf{Z}p_i + \mathbf{Z}(-b+\delta))^{e_i}$$

Thus, the task to factorize the ideal is equivalent to the task of the natural numbers decomposition into the product of primes.

We may talk about the cryptographic security of the RSA for the Dedekind rings only when we define the form of ideals. In the previous theorem it was supposed that there exists a reasonably good way, which allows us to find a standard form for every ideal of R. So, if a question of finding a good algorithm, which will allow us to find a standard form of ideal in the quadratic ring can be successfully resolved, then the quadratic ring of algebraic integers (without $\mathbf{Z}$) will be unfit for being used as platform for the generalized RSA algorithm.

Some results of this paper were announced in [8].

This work was partially funded by the subsidy allocated to Kazan Federal University for the state assignment in the sphere of scientific activities (the registration number is 114 090 970 010).

References

[1] Cross J. T. The Euler φ-Function in the Gaussian Integers // The American Mathematical Monthly. — 1983. — V. 90, No 8. — P. 518–528.

[2] El-Kassar A. N., Haraty R., Awad Y. A.. Modified RSA in the domains of Gaussian integers and polynomials over finite fields //Proceedings of the ISCA 18th International Conference on Computer Applications in Industry and Engineering, Honolulu, 9-11 November 2005. — P. 298–303.

[3] Haraty R.A., El-Kassar A. N., Shibaro B. A Comparative Study of RSA Based Digital Signature Algorithms // Journal of Mathematics and Statistics. — 2006. — V. 2, No 1. — P. 354–359.

[4] Koval A., Verkhovsky B.S. Analysis of RSA over Gaussian Integers Algorithm // ITNG-2008, Information Technology: New Generations, Third International Conference. — P. 101–105.

[5] Rodosskii K.A. Euclidean Algorithm. (In Russian) — Moscow.: Nauka, Gl. red. fiz.-mat. lit., 1988. — 240 p.

[6] Swinnerton-Dyer H. P. F. A Brief Guide to Algebraic Number Theory. — Cambridge University press, 2001. — 145 p.

[7] Trifković M. Algebraic Theory of Quadratic Numbers. — Springer Science+Business Media, New York, 2013. — xi+197 pp.

[8] Tronin S.N., Petikhova K.A. RSA cryptosystem for Dedekind rings // International Conference “Algebra and Mathematical Logic: Theory and Applications” (Kazan, Russia, June 2–6, 2014) – Kazan: Kazan University, 2014. — P. 148–149.

Additively Homomorphic Encryption Using Matrix Polynomials with Approximate Perfect Security

Philipp Burtyka

Abstract

This paper proposes a new probabilistic additively homomorphic encryption (AHE) scheme based on univariate matrix polynomials over prime fields. The cryptosystem is simple and efficient. The analysis of its security properties shows that compared to predecessors it has so-called approximate perfect security in k-bounded KPA (known plaintext attack) model.

Keywords: Additively Homomorphic Encryption, Matrix Polynomials, Approximate Perfect Security.

1 Introduction

Additively homomorphic cryptosystem (AHC) allows to compute $Enc(x+y)$ using $Enc(x), Enc(y)$ without knowledge of a secret key sk. Additively homomorphic encryption (AHE) is of great interest today because of many practical applications in various areas, like for example: electronic voting [1], privacy - preserving data mining [9], private information retrieval [12], private matching and set operations [4, 8], securing mobile agents [13].

Today a great variety of AHCs exists. The most well-known AHC is celebrated Goldwasser-Micali's scheme [5] based on quadratic residuosity problem. Other AHCs based on similar problems can be founded for example in [6]. Also there are AHCs based on elliptic curves [14], and finally AHCs exploiting lattice framework exist, for example [11, 10].

All mentioned AHCs are asymmetric and to prove their security the authors use an approach based on complexity theory. More precisely AHCs are usually shown to be semantically secure against different types of attacks, if some mathematical problem is hard to solve (for example: factoring big numbers, finding the shortest lattice vector, etc.). This means that if problem is hard, any polynomial adversary can't distinguish between probability distributions $Pr\{m_0|C\}$ and $Pr\{m_1|C\}$, where m_0, m_1 – some plaintexts and C – any ciphertext.

But to the best of our knowledges efficient symmetric AHC with perfect security hasn't been proposed yet. And there is only one homomorphic cryptosystem (HC) [7] that is proven to be approximate perfectly secure in k-bounded KPA (known plaintext attack) model. This means that this cryptosystem is approximate perfectly secure if adversary has at most k pairs (plaintext, ciphertext), where k is linearly bounded in cryptosystem's parameters. However HC [7] is very inefficient (even if only its additive homomorphism is exploited). But in the light of quantum computing progress it's of key importance to build efficient homomorphic cryptosystems having perfect security even to some extent, because they may be much more resistant to different attacks.

Our contribution. In this paper we analyze a simplified version of HC from [2] based on matrix polynomials. In contrast to [2] HC presented here is not multiplicatively homomorphic. It is only AHC, and so for simplicity we call it $AHMP$ (additively homomorphic using matrix polynomials). Our simplification of [2] leads to possibility to prove that $AHMP$ has approximate perfect security in k-bounded KPA (known plaintext attack) model. $AHMP$ is the first AHC with such security property. Compared to [7] for $AHMP$ k is bounded quadratically in cryptosystem's parameters. Also it's important that in $AHMP$ all algorithms and ciphertexts sizes depends polynomially in its parameters, wherein in [7] they are exponential.

Organization of the paper. Section 2 provides some necessary background. Section 3 describes $AHMP$. And Section 4 deals with security of $AHMP$. Proofs of some technical lemmas can be found in Appendix.

2 Preliminaries

2.1 Notations

Natural numbers, integers and real numbers are denoted by $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{R}$ correspondingly, a prime field of characteristic q – by $\mathbb{F}_q$, the ring of square $n \times n$ matrices with $\mathbb{F}_q$ entries – by $\mathbb{F}_q^{n\times n}$. By lowercase letters we denote vectors (like $\vec{v}$) and by uppercase bold letters – matrices (like $\mathbf{M}$). The i^{th} row of matrix $\mathbf{M}$ is $\mathbf{M}_i$. Zero vector and zero matrix are $\vec{0}$ and $\mathbf{0}$ correspondingly.

The probability of event $\mathcal{X}$ is $Pr\{\mathcal{X}\}$, conditional probability of $\mathcal{Y}$ after $\mathcal{X}$ – $Pr\{\mathcal{X}|\mathcal{Y}\}$, marginal probability of $\mathcal{X}$ and $\mathcal{Y}$ – $Pr\{\mathcal{X}, \mathcal{Y}\}$. $x \overset{\$}{\leftarrow} R$ means that x is a random element sampled according to uniform distribution over ring R, $x_i \overset{\$}{\leftarrow} R, i = \overline{1,m}$ – random elements sampled uniformly and independently, $x \overset{\mathfrak{D}}{\leftarrow} R$ – an element sampled according to $\mathfrak{D}$.

2.2 Matrix Polynomials

Let's look at the expression

$$F(X) = \mathbf{F}_d X^d + \mathbf{F}_{d-1} X^{d-1} + ... + \mathbf{F}_1 X + \mathbf{F}_0,$$

where $\mathbf{F}_i, \in \mathbb{F}_q^{n\times n}, i = \overline{0,d}$ – some constant matrices, $\mathbf{F}_d \neq \mathbf{0}$, $X \in \mathbb{F}_q^{n\times n}$ is a matrix variable. This is a univariate matrix polynomial in variable X of degree d. The set of matrix polynomials forms a ring $\mathbb{F}_q^{n\times n}[X]$ with usual polynomial operations $+, \cdot$. For example the sum of $F_i(X) = \sum_{j=0}^{d_i} \mathbf{F}_{i,j} \cdot X^j, i = 1, 2,\ d_1 > d_2$, looks like $F_+(X) = \sum_{j=0}^{d_1} (\mathbf{F}_{1,j} + \mathbf{F}_{2,j}) \cdot X^j$, where $\mathbf{F}_{2,j}, j = \overline{d_2 + 1, d_1}$ are set to $\mathbf{0}$ and the product – $F_\cdot(X) = \sum_{j=0}^{d_1+d_2} \mathbf{F}_{\cdot,j} \cdot X^j$, where $\mathbf{F}_{\cdot,j} = \sum_{i+k=j} \mathbf{F}_{1,i} \cdot \mathbf{F}_{2,k}$.

Also we'll use the following notation $\mathbb{F}_{q,d}^{n\times n}[X] = \{F(X) \in \mathbb{F}_q^{n\times n}[X], deg(F(X)) \leq d\}$.

We say that $F(X) \in \mathbb{F}_q^{n\times n}[X]$ has a root if $\exists \mathbf{K} \in \mathbb{F}_q^{n\times n}$ s.t. $F(\mathbf{K}) = \mathbf{0}$. The subset of $\mathbb{F}_{q,d}^{n\times n}[X]$ containing all $F(X)$ having at least one root will be denoted by $\mathbb{F}_{q,d,zero}^{n\times n}[X]$.

2.3 Additive Homomorphic Encryption

Definition 1. *Symmetric cryptosystem CS is a tuple of non-empty finite sets $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \{e_{\mathcal{K}}\}, \{d_{\mathcal{K}}\})$, where $\mathcal{P}$ – plaintext set, $\mathcal{C}$ – ciphertext set, $\mathcal{K}$ – set of secret keys, $\{e_{\mathcal{K}}\}$ – set of encryption functions, i.e. parametrized mappings from $\mathcal{P}$ to $\mathcal{C}$, $\{d_{\mathcal{K}}\}$ – set of decryption functions, i.e. parametrized mappings from $\mathcal{C}$ to $\mathcal{P}$. And for $\forall \mathbf{sk} \in \mathcal{K}$, $\forall m \in \mathcal{P}$ the following property holds: $d_{\mathbf{sk}}(e_{\mathbf{sk}}(m)) = m$.*

Usually, cryptosystem CS has algorithms Keygen_{CS}, $\mathsf{Encrypt}_{CS}$, $\mathsf{Decrypt}_{CS}$ implementing CS. Their computational complexity must be polynomial.

For security analysis one needs the probability distributions $\mathfrak{K}$ and $\mathfrak{P}$ over $\mathcal{K}$ and $\mathcal{P}$ respectively. For further discussion we assume that $\mathfrak{K}$ is uniform and $\mathfrak{P}$ may be any distribution s.t. for $\forall m \in \mathcal{P}$ $Pr_{\mathfrak{P}}\{m\} > 0$.

For probabilistic encryption [5] for a fixed key $\mathbf{sk}$ and $m \in \mathcal{P}$ there exists a set of ciphertexts that correspond to m according to $\mathsf{Encrypt}_{CS}$. We will denote this set by $\mathcal{C}_{m,\mathbf{sk}}$.

Definition 2. *The cryptosystem CS is AHC if for $\forall m_1, m_2 \in \mathcal{P}$ and $\forall \mathbf{sk} \in \mathcal{K}$ $d_{\mathbf{sk}}(e_{\mathbf{sk}}(m_1)\ op_{\mathcal{C}}\ e_{\mathbf{sk}}(m_2)) = m_1 + m_2$, where $op_{\mathcal{C}}$ – some binary operation over $\mathcal{C}$ and the compactness property is satisfied, i.e.*

$$bitlength(e_{\mathbf{sk}}(m_1)\ op_{\mathcal{C}}\ e_{\mathbf{sk}}(m_2)) \leq$$

$$\leq max\{bitlength(e_{\mathbf{sk}}(m_1)), bitlength(e_{\mathbf{sk}}(m_2))\}.$$

2.4 Security definitions

For further discussion we need the notion of approximate perfect security in k-bounded KPA model. In this model adversary $\mathcal{A}$ trying to break cryptosystem is allowed to have $\leq k$ pairs (m_i, C_i) made on the same $\mathbf{sk}$ for fixed $k \in \mathbb{N}$ and $m_i \in \mathcal{P}, i = \overline{1,k}$. After receiving (m_i, C_i) $\mathcal{A}$ intercepts some random ciphertext $C \in \mathcal{C}$ and tries to find $m \in \mathcal{P}$ encrypted into C on $\mathbf{sk}$. CS is perfectly secure in k-bounded KPA model iff $Pr_{\mathfrak{P}}\{m\} = Pr\{m|C, C_1, ..., C_k\}$ for $\forall m \in \mathcal{P}$ (for brevity here and below we denote $Pr\{m|C, (m_1, C_1), ..., (m_k, C_k)\}$ by $Pr\{m|C, C_1, ..., C_k\}$). In turn approximate perfect security is a little bit weaker security requirement that can be formalized as follows.

Definition 3 ([7])**.** *Let $CS(t) = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \{e_k\}, \{d_k\})$ be a cryptosystem with a security parameter t, $\mathcal{F} = \{f : \mathcal{C}^{k+1} \mapsto \mathbb{R}\}$ be the set of all functions from $\mathcal{C}^{k+1}$ to $\mathbb{R}$ and $\delta : \mathcal{F} \times \mathcal{F} \mapsto \mathbb{R}$ be a metric (distance) on $\mathcal{F}$. Let's fix arbitrary $m \in \mathcal{P}$ and consider $Pr\{m|\ C, C_1, ..., C_k\}$ and $Pr_{\mathfrak{P}}\{m\}$ that are in fact functions $\in \mathcal{F}$ (the later one is a constant function in $C, C_1, ..., C_k \in \mathcal{C}^{k+1}$). We say, that $CS(t)$ has* ***approximate perfect security*** *in k-bounded KPA model, iff for $\forall$ probability distributions $\mathfrak{P}$ on $\mathcal{P}$ and $\forall m \in \mathcal{P}$*

$$\lim_{t \to \infty} \delta\big(Pr\{m \mid C, C_1, ..., C_k\}, Pr_{\mathfrak{P}}\{m\}\big) = 0 \ . \tag{1}$$

We will use the following metric δ.

Definition 4 ([7])**.** *Let $f, g \in \mathcal{F} = \{f : \mathcal{C}^{k+1} \mapsto \mathbb{R}\}$. Define a metric δ as*

$$\delta(f, g) = \frac{1}{|\mathcal{C}^{k+1}|} \cdot \sum_{c \in \mathcal{C}^{k+1}} (f(c) - g(c))^2.$$

3 Additively homomorphic cryptosystem

Let $q \in \mathbb{N}$ – prime, $n, d \in \mathbb{N}$, $\mathcal{P} = \mathbb{F}_q$, $\mathcal{K} = \mathbb{F}_q^{n\times n} \times \mathbb{F}_q^n \setminus \{\vec{0}\} \times \{1, ..., n\}$, $\mathcal{C} \subseteq \mathbb{F}_{q,d}^{n\times n}[X]$. The cryptosystem $AHMP$ may be described by Algorithms 1, 2, 3. Its correctness and the presence of homomorphism are proved in lemmas 1, 2 (proofs are placed in Appendix).

Algorithm 1: Setup(n, d, q).

```
Input: parameters $n, d, q \in \mathbb{N}$, $q$ is a prime
Output: secret key $\mathbf{sk} = (\mathbf{K}, \vec{k}, \bar{i})$, where $\mathbf{K} \in \mathbb{F}_q^{n\times n}$, $\vec{k} \in \mathbb{F}_q^n \setminus \{\vec{0}\}$
1 $\mathbf{K} \overset{\$}{\leftarrow} \mathbb{F}_q^{n\times n}$;
2 $\vec{k} \overset{\$}{\leftarrow} \mathbb{F}_q^n \setminus \{\vec{0}\}$;
3 $\bar{i} \overset{\$}{\leftarrow} \{1, ..., n\}$;
4 $\mathbf{sk} := (\mathbf{K}, \vec{k}, \bar{i})$ ;
5 return $\mathbf{sk}$;
```

Algorithm 2: Encrypt$(m, \mathbf{sk})$.

```
    Input: $m \in \mathcal{P}$, $\mathbf{sk} = (\mathbf{K}, \vec{k}, \bar{i})$
    Output: $C(X) \in \mathcal{C}$
1   $\mathbf{M} \xleftarrow{\$} \mathbb{F}_q^{n\times n}$ ;
2   $\bar{j} := \min\{j \in \{1, ..., n\}, s.t. k_j \neq 0\}$;
3   $M_{\bar{i},\bar{j}} := k_{\bar{j}}^{-1} \cdot [m - \sum_{t=1,t\neq\bar{j}}^{n} M_{\bar{i},t} \cdot k_t]$;
4   $F(X) \xleftarrow{\$} \mathbb{F}_{q,d}^{n\times n}[X]$;
5   $C(X) := F(X) - F(\mathbf{K}) + \mathbf{M} \in \mathcal{C}$;
6   return $C(X)$;
```

Algorithm 3: Decrypt$(\mathbf{sk}, c)$.

```
    Input: $\mathbf{sk} = (\mathbf{K}, \vec{k}, \bar{i})$,$C(X)$
    Output: $m \in \mathbb{F}_q$
1   $\mathbf{M} \leftarrow C(\mathbf{K})$;
2   $m := \mathbf{M}_{\bar{i}} \cdot \vec{k}$;
3   return $m$;
```

Lemma 1. *The AHMP cryptosystem has a correct decryption.*

Lemma 2. *The AHMP cryptosystem is additively homomorphic, i.e. if there are $C_1(X), C_2(X) \in \mathcal{C}$ encrypting $m_1, m_2 \in \mathcal{P}$, then $C_1(X) + C_2(X)$ encrypts $m_1 + m_2$.*

One may see that $AHMP$ is very similar to HC from [2]. The main difference is that in [2] there is $\mathbf{M} \cdot \vec{k} = m \cdot \vec{k}$ and this gives both multiplicative and additive homomorphism.

3.1 Computational complexity of $AHMP$

- The most complex part of Encrypt$(\mathbf{sk}, m)$ is computing of $F(\mathbf{K})$. Its computational cost is $O(d \cdot n^3)$ operations in $\mathbb{F}_q$. So the asymptotic complexity of Encrypt$(\mathbf{sk}, m)$ is $O(d \cdot n^3)$.
- The most complex part of Decrypt$(\mathbf{sk}, C(X))$ is computing of $C(\mathbf{K})$ and so the asymptotic complexity of Decrypt$(\mathbf{sk}, C(X))$ is $O(d \cdot n^3)$.

- $C_1(X), C_2(X) \in \mathbb{F}_{q,d}^{n\times n}[X]$ and therefore computing $C_1(X) + C_2(X)$ requires $O(d \cdot n^2)$ operations in $\mathbb{F}_q$.

4 Security Analysis

Theorem 1. *The cryptosystem $AHMP(n, d, q)$ achieves approximate perfect security in k-bounded KPA model for $k = n^2 + n$.*

To prove this theorem we need several additional lemmas.

Lemma 3. *For AHMP $\mathcal{C} = \mathbb{F}_{q,d}^{n\times n}[X]$.*

Lemma 4. *For $\forall \mathbf{sk} \in \mathcal{K}$ $Pr\{\mathbf{sk} \in \mathcal{K}\} = 1/(q^{n^2} \cdot (q^n - 1) \cdot n)$.*

Lemma 5. *Let's fix some arbitrary $\mathbf{sk} = (\mathbf{K}, \vec{k}, \bar{i}) \in \mathcal{K}$ and $m \in \mathcal{P}$. Then for a random ciphertext $C(X)$, s.t. $C(X) \xleftarrow{\$} \mathcal{C}$,*

$$Pr[\mathsf{Decrypt}(\mathbf{sk}, C(X)) = m] = 1/q.$$

Lemma 6. *Let's fix some $m \in \mathcal{P}$ and $\mathbf{sk} \in \mathcal{K}$, then $|\mathcal{C}_{m,\mathbf{sk}}| = q^{(d+1)\cdot n^2 - 1}$ and for $\forall C(X) \in \mathcal{C}_{m,\mathbf{sk}}$ $Pr\{C(X)|m, \mathbf{sk}\} = 1/q^{(d+1)\cdot n^2-1}$.*

Lemmas 5, 6, 3, 4 are proved in Appendix.

Finally to prove Theorem 1 we need a probability distribution induced by Algorithm 2 over $\mathcal{C}$ for randomly chosen $m \in \mathcal{P}$ and $\mathbf{sk} \in \mathcal{K}$. Unfortunately it's not uniform for $\forall \mathfrak{P}$ over $\mathcal{P}$. However it's close to it, more precisely it goes to uniform for $n \to \infty$. But due to the lack of space we omit the proof of this fact and for simplicity we'll use the following simplifying assumption to prove Theorem 1.

Assumption 1. $Pr_{\mathbf{sk} \xleftarrow{\$} \mathcal{K}}\{C\} = 1/|\mathcal{C}| = 1/q^{(d+1)\cdot n^2}$ *for $\forall C \in \mathcal{C}$ produced by the encryption procedure.*

Although we claim that the theorem can be proved without this simplification and it'll be shown in the extended version of the paper.

Proof of Theorem 1. We'll prove approximate perfect security of $AHMP$ (n, d, q) in k-bounded KPA model. We suppose $\mathcal{A}$ has some $(m_i, C_i), i =$

$\overline{1,k}$, s.t. $\mathsf{Decrypt}(\overline{\mathbf{sk}}, C_i) = m_i, i = \overline{1,k}$ for some unknown $\overline{\mathbf{sk}}$. And also $\mathcal{A}$ has $C \in \mathcal{C}$ obtained by encrypting with $\overline{\mathbf{sk}}$ an unknown plaintext $\overline{m} \in \mathcal{P}$ which $\mathcal{A}$ wish to recover. To show approximate perfect security one need to estimate a distance between $Pr_{\mathfrak{P}}\{m\}$ and $Pr\{m \mid C, C_1, ..., C_k\}$ for $\forall m \in \mathcal{P}$.

For any m the following holds

$$Pr\{m \mid C, C_1, ..., C_k\} = \frac{Pr\{m, C, C_1, ..., C_k\}}{Pr\{C, C_1, ..., C_k\}} = \tag{2}$$

$$= \frac{1}{Pr\{C, C_1, ..., C_k\}} \cdot \sum_{\mathbf{sk} \in \mathcal{K}} Pr\{m, C, C_1, ..., C_k \mid \mathbf{sk}\} \cdot Pr\{\mathbf{sk}\}\ . \tag{3}$$

Note, that $Pr\{C, C_1, ..., C_k\} = Pr\{C\} \cdot \prod_{i=1}^{k} Pr\{C_i\}$ because $C, C_1, ..., C_k$ are independent and therefore by assumption 1

$$Pr\{C, C_1, ..., C_k\} = 1/q^{(k+1)\cdot(d+1)\cdot n^2}.$$

By lemma 4 $Pr\{\mathbf{sk}\} = 1/(q^{n^2} \cdot (q^n - 1) \cdot n)$. And clearly if $\mathsf{Decrypt}(\mathbf{sk}, C) \neq m$ or $\mathsf{Decrypt}(\mathbf{sk}, C_i) \neq m_i$ for some i, then $Pr\{m, C, C_1, ..., C_k \mid \mathbf{sk}\} = 0$. Hence, one can sum in (3) using only keys $\mathbf{sk} \in A$, where $A = \{\mathbf{sk} \in \mathcal{K} \mid \mathsf{Decrypt}(\mathbf{sk}, C) = m, \mathsf{Decrypt}(\mathbf{sk}, C_1) = m_1, ..., \mathsf{Decrypt}(\mathbf{sk}, C_k) = m_k\}$. So we obtain

$$Pr\{m \mid C, C_1, ..., C_k\} = \frac{q^{(k+1)\cdot(d+1)\cdot n^2}}{q^{n^2} \cdot (q^n - 1) \cdot n} \sum_{\mathbf{sk} \in \mathcal{A}} Pr\{m, C, C_1, ..., C_k \mid \mathbf{sk}\}. \tag{4}$$

In the last formula for a fixed $\mathbf{sk} \in A$ $m, C, C_1, ..., C_k$ are independent (because m is fixed). Also m and the secret key $\mathbf{sk}$ are independent. And (4) may be rewritten as follows

$$\begin{aligned} &Pr\{m \mid C, C_1, ..., C_k\} = \\ =&\frac{(q^{(k+1)\cdot(d+1)\cdot n^2})}{q^{n^2} \cdot (q^n - 1) \cdot n} \sum_{\mathbf{sk} \in A} Pr_{\mathfrak{P}}\{m\} \cdot Pr\{C \mid \mathbf{sk}\} \cdot \prod_{i=1}^{k} Pr\{C_i \mid \mathbf{sk}\}. \end{aligned} \tag{5}$$

Because m now is fixed $Pr\{C \mid \mathbf{sk}\} = Pr\{C \mid m, \mathbf{sk}\} = 1/q^{(d+1)\cdot n^2-1}$, and also $Pr\{C_i \mid \mathbf{sk}\} = Pr\{C_i \mid m_i, \mathbf{sk}\} = 1/q^{(d+1)\cdot n^2-1}, i = \overline{1,k}$ (we use lemma 6). Therefore we have

$$\begin{aligned} &Pr\{m \mid C, C_1, ..., C_k\} = \\ =&\frac{q^{(k+1)\cdot(d+1)\cdot n^2}}{q^{n^2}\cdot(q^n-1)\cdot n}\sum_{\mathbf{sk}\in A} Pr_{\mathfrak{P}}\{m\}\cdot\frac{1}{q^{(k+1)\cdot((d+1)\cdot n^2-1)}} = \\ =&\frac{q^{k+1}}{q^{n^2}\cdot(q^n-1)\cdot n}\cdot Pr_{\mathfrak{P}}\{m\}\cdot|A|. \end{aligned} \tag{6}$$

Recall that $|A| = |\{\mathbf{sk} \in \mathcal{K} \mid \mathsf{Decrypt}(\mathbf{sk}, C) = m, \mathsf{Decrypt}(\mathbf{sk}, C_1) = m_1, ..., \mathsf{Decrypt}(\mathbf{sk}, C_k) = m_k\}|$. So $|A|$ is a random variable depending on $m, C, C_1, ..., C_k$. And below we'll use notation $|A| = \gamma(m, C, C_1, ..., C_k)$. Hence

$$\begin{aligned} &Pr\{m \mid C, C_1, ..., C_k\} = \\ =&\frac{q^{k+1}}{q^{n^2}\cdot(q^n-1)\cdot n}\cdot Pr_{\mathfrak{P}}\{m\}\cdot\gamma(m, C, C_1, ..., C_k)\ . \end{aligned} \tag{7}$$

For some fixed $\mathbf{sk} \in \mathcal{K}$ let's consider random variable $Y_{\mathbf{sk}}(m\ , C, C_1, ..., C_k)$ which $= 1$ iff $\mathbf{sk} \in A$ and $= 0$ otherwise, for $C, C_1, ..., C_k$ sampled uniformly at random. According to independence of $C, C_1, ..., C_k$ from each other and due to lemma 5 $Pr\{Y_{\mathbf{sk}} = 1/q^{k+1}\}$. It is clear that

$$\gamma(m, C, C_1, ..., C_k) = \sum_{\mathbf{sk}\in\mathcal{K}} Y_{\mathbf{sk}}$$

and this implies that $\gamma(m, C, C_1, ..., C_k)$ has the binomial distribution with parameters $|\mathcal{K}| = q^{n^2}\cdot(q^n-1)\cdot n$ and $\frac{1}{q^{k+1}}$, which is denoted by $B(q^{n^2}\cdot(q^n-1)\cdot n, \frac{1}{q^{k+1}})$. It is known, that $B(q^{n^2}\cdot(q^n-1)\cdot n, \frac{1}{q^{k+1}})$ has the expected value $E[\gamma] = \frac{1}{q^{k+1}}$ and the variance $Var[\gamma] = |\mathcal{K}|\cdot(1-1/q^{k+1})\cdot 1/q^{k+1} = \frac{q^{n^2}(q^n-1)}{q^{k+1}}\cdot\frac{q^{k+1}-1}{q^{k+1}}$.

Now we can evaluate the distance between $Pr_{\mathfrak{P}}\{m\}$ and $Pr\{m \mid C, C_1, ..., C_k\}$ for $\forall m \in \mathcal{P}$ with respect to δ from Definition 3 to show that $AHMP$ achieves approximate security in k-bounded KPA model.

$$\begin{aligned}
&\delta(Pr\{m \mid C, C_1, ..., C_k\}, Pr_{\mathfrak{P}}\{m\}) = \\
&= \frac{1}{|\mathcal{C}^{k+1}|} \cdot \sum_{C, C_1, ..., C_k \in \mathcal{C}^{k+1}} \Big(Pr\{m \mid C, C_1, ..., C_k\} - Pr_{\mathfrak{P}}\{m\} \Big)^2 = \\
&= \frac{1}{|\mathcal{C}^{k+1}|} \cdot \sum_{C, C_1, ..., C_k \in \mathcal{C}^{k+1}} \Big(\frac{q^{k+1}}{q^{n^2} \cdot (q^n - 1) \cdot n} \cdot Pr_{\mathfrak{P}}\{m\} \cdot \qquad (8) \\
&\cdot \gamma(m, C, C_1, ..., C_k) - Pr_{\mathfrak{P}}\{m\} \Big)^2 = \\
&= (Pr_{\mathfrak{P}}\{m\})^2 \cdot \frac{q^{2\cdot(k+1)}}{\big(q^{n^2} \cdot (q^n - 1) \cdot n\big)^2} \cdot
\end{aligned}$$

$$\cdot \underbrace{\sum_{C, C_1, ..., C_k \in \mathcal{C}^{k+1}} \frac{1}{|\mathcal{C}^{k+1}|} \Big(\gamma(m, C, C_1, ..., C_k) - \frac{q^{n^2} \cdot (q^n - 1) \cdot n}{q^{(k+1)}} \Big)^2}_{Var[\gamma(m, C, C_1, ..., C_k)]} = \qquad (9)$$

$$= (Pr_{\mathfrak{P}}\{m\})^2 \cdot \frac{q^{2\cdot(k+1)}}{\big(q^{n^2} \cdot (q^n - 1) \cdot n\big)^2} \cdot \frac{q^{n^2} \cdot (q^n - 1) \cdot n}{q^{k+1}} \cdot \frac{q^{k+1} - 1}{q^{k+1}} = \qquad (10)$$

$$= (Pr_{\mathfrak{P}}\{m\})^2 \cdot \frac{q^{k+1} - 1}{q^{n^2} \cdot (q^n - 1) \cdot n}. \qquad (11)$$

Now we only need to determine k s.t. $\lim_{n\to\infty} \frac{q^{k+1}-1}{q^{n^2}\cdot(q^n-1)\cdot n} = 0$. It's easy to obtain that k should satisfy the following inequality

$$k \leq n^2 + n. \qquad (12)$$

Therefore, we have shown that for $\forall m \in \mathcal{P}$ and any randomly intercepted $C, C_1, ..., C_k$

$$\lim_{n\to\infty} \delta(Pr\{m \mid C, C_1, ..., C_k\}, Pr_{\mathfrak{P}}\{m\}) = 0$$

for $k < n^2 + n$, $0 \leq a, b \leq 1$. $\square$

According to the proof of theorem 1 n may be considered as security parameter of $AHMP(q, n, d)$ (see definition 3) that handles the approximate perfect security property. And one may see that the degree d doesn't influence on the last one. So from this point of view d may be set to 1 in order to obtain lower computational complexity of $AHMP$. Let's comment on the obtained bound $k \leq n^2 + n$ for $d = 1$. If pairs $(m_i, C_i), i = \overline{1, k}$ are given, then one may compose n systems of quadratic polynomial equations with $n^2 + n$ variables of the form $[C_t(\mathbf{X})]_j \cdot \vec{y} = m_t, t = \overline{1, k}$. Here $j = \overline{1, n}$, $[C_t(\mathbf{X})]_j$ is j-th row of matrix $C_t(\mathbf{X})$ and $\mathbf{X}, \vec{y}$ – matrix and vector variables . The solution of one of this systems is a secret key $\mathbf{K}, \vec{k}$. It is a well known result [3] that a system of t quadratic equations with s variables over a finite field may be solved efficiently only if $t > s$. So this explains the bound $k \leq n^2 + n$ for approximate perfect security.

Remark. *The discussion above shows that $AHMP$ even for $d = 1$ is not a linear cryptosystem.*

In the case of $d > 1$ the system of equations introduced above has monomials of a degree > 3 and therefore becomes much harder to solve. To obtain its solution in reasonable time it may be necessary to have essentially larger number k of pairs than for $d = 1$. So we suppose that for $d > 1$ and $k > n^2 + n$ $AHMP$ may be still resistance to KPA. But it will be already a security based on a complexity theory. In a future work we are planning to investigate this problem in details.

Finally we'd like to note that $AHMP$ has similar security properties in k-bounded COA (ciphertext only attack) model. Precise bounds on k will be given in a future work. And also resistance of $AHMP$ against CPA (chosen plaintext attack) and CCA (chosen ciphertext attack) will be studied.

5 Conclusion

A novel symmetric additively homomorphic cryptosystem $AHMP(q, n, d)$ based on matrix polynomials was presented. Its main feature is the presence of such unusual property as approximate perfect security in k-bounded

KPA model, where $k \leq n^2 + n$. The last one means that the distance between distributions $Pr\{m \mid C, (m_1, C_1), ..., (m_k, C_k)\}$ and $Pr_{\mathfrak{P}}\{m\}$ goes to zero if $n \to \infty$ for any fixed $m \in \mathcal{P}$, $k \leq n^2 + n$ and intercepted pairs $(m_i, C_i), i = \overline{1, k}$. We'd like to point out that our bound on k is better then for approximate perfectly secure symmetric homomorphic cryptosystem from [7]. And also in comparison with [7] all algorithms of our $AHMP(q, n, d)$ depend polynomially on its parameters q, n, d. Ciphertexts sizes are also polynomially in q, n, d, wherein in [7] all algorithms and ciphertexts sizes are exponential.

The only advantage of [7] in comparison with $AHMP(q, n, d)$ is that presented construction is not only additively homomorphic. Also it has multiplicative homomorphism (however not efficient). So in future we are planning to check whether the extended version of $AHMP(q, n, d)$ presented in [2] possessing multiplicative homomorphism is approximate perfectly secure.

Also we'd like to stress out that of course the analysis of $AHMP(q, n, d)$ security while isn't full. It requires further investigations because for any finite q, n distributions $Pr\{m \mid C, C_1, ..., C_k\}$ and $Pr_{\mathfrak{P}}\{m\}$ may be very close, but nevertheless they are not identical. It seems necessary to correlate approximate perfect security property with different possible strategies to attack the cryptosystem. And based on this analysis parameters settings suitable for practical usage of $AHMP(q, n, d)$ should be derived.

6 Acknowledgements

The work is supported by grant RFBR 15-07-00597-a.

References

[1] Josh Daniel Cohen Benaloh. *Verifiable secret-ballot elections.* Yale University. Department of Computer Science, 1987.

[2] Philipp Burtyka and Oleg Makarevich. Symmetric fully homomorphic encryption using decidable matrix equations. In *Proceedings of the 7th*

International Conference on Security of Information and Networks, page 186. ACM, 2014.

[3] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in Cryptology EUROCRYPT 2000*, pages 392–407. Springer, 2000.

[4] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology-EUROCRYPT 2004*, pages 1–19. Springer, 2004.

[5] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[6] Antoine Guellier. *Can Homomorphic Cryptography ensure Privacy?* PhD thesis, Inria; IRISA; Supélec Rennes, équipe Cidre; Université de Rennes 1, 2014.

[7] Michal Hojsík and Veronika Půlpánová. A fully homomorphic cryptosystem with approximate perfect secrecy. In *Topics in Cryptology–CT-RSA 2013*, pages 375–388. Springer, 2013.

[8] Lea Kissner and Dawn Song. Privacy-preserving set operations. In *Advances in Cryptology–CRYPTO 2005*, pages 241–257. Springer, 2005.

[9] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Advances in Cryptology CRYPTO 2000*, pages 36–54. Springer, 2000.

[10] Carlos Aguilar Melchor, Guilhem Castagnos, and Philippe Gaborit. Lattice-based homomorphic encryption of vector spaces. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 1858–1862. IEEE, 2008.

[11] Carlos Aguilar Melchor, Philippe Gaborit, and Javier Herranz. Additively homomorphic encryption with d-operand multiplications. In *Advances in Cryptology–CRYPTO 2010*, pages 138–154. Springer, 2010.

[12] Rafail Ostrovsky and William E Skeith III. A survey of single-database private information retrieval: Techniques and applications. In *Public Key Cryptography–PKC 2007*, pages 393–411. Springer, 2007.

[13] Tomas Sander and Christian F Tschudin. Protecting mobile agents against malicious hosts. In *Mobile agents and security*, pages 44–60. Springer, 1998.

[14] Scott A Vanstone and Robert J Zuccherato. Elliptic curve cryptosystems using curves of smooth order over the ring z n. *Information Theory, IEEE Transactions on*, 43(4):1231–1237, 1997.

A Proofs of lemmas concerning cryptosystem correctness

Proof of Lemma 4. We have $\mathcal{K} = \mathbb{F}_q^{n\times n} \times \mathbb{F}_q^n \setminus \vec{0} \times \{1, ..., n\}$ and then $|\mathcal{K}| = q^{n^2} \cdot (q^n - 1) \cdot n$. Secret key $\mathbf{sk} \in \mathcal{K}$ is chosen uniformly at random. This proves the statement. $\square$

Proof of Lemma 1. We need to prove that if $C(X) = \mathsf{Encrypt}(m, \mathbf{sk})$ for any $m \in \mathcal{P}, \forall \mathbf{sk} = (\mathbf{K}, \vec{k}, \bar{\imath})$ then there is $\mathsf{Decrypt}(\mathbf{sk}, C(X)) = m$.

According to Algorithm 2 $C(X) = G(X) + \mathbf{M}$, where $G(X) \in \mathbb{F}_q^{n\times n}[X]$ is s.t. $G(\mathbf{K}) = \mathbf{0}$. Therefore $C(\mathbf{K}) = \mathbf{M}$ holds. $\mathbf{M}$ is built in such a way that $\mathbf{M}_{\bar{\imath}} \cdot \vec{k} = \sum_{t=1,t\neq\bar{\jmath}}^{n} M_{\bar{\imath},t} \cdot k_t + k_{\bar{\jmath}} \cdot k_{\bar{\jmath}}^{-1} \cdot [m - \sum_{t=1,t\neq\bar{\jmath}}^{n} M_{\bar{\imath},t} \cdot k_t] = m.$ $\square$

Proof of Lemma 2. We need to prove that if $C_i(X) = \mathsf{Encrypt}(m_i, \mathbf{sk}), i = 1, 2$ for any $m_i \in \mathcal{P}, i = 1, 2$ and $\mathbf{sk} = (\mathbf{K}, \vec{k}, \bar{\imath})$ then $\mathsf{Decrypt}(\mathbf{sk}, C_1 + C_2) = m_1 + m_2$ holds.

$C_1 + C_2 = G_+(X) + \mathbf{M}_+$, where $G_+(X) = G_1(X) + G_2(X)$, $\mathbf{M}_+ = \mathbf{M}_1 + \mathbf{M}_2$. Obviously $G_+(\mathbf{K}) = \mathbf{0}$ and therefore $(C_1 + C_2)(\mathbf{K}) = \mathbf{M}_+$, where $\mathbf{M}_{+,\bar{\imath}} \cdot \vec{k} = \sum_{t=1,t\neq\bar{\jmath}}^{n} [M_{1,\bar{\imath},t} + M_{2,\bar{\imath},t}] \cdot k_t + k_{\bar{\jmath}} \cdot k_{\bar{\jmath}}^{-1} \cdot [m_1 + m_2 - \sum_{t=1,t\neq\bar{\jmath}}^{n} [M_{1,\bar{\imath},t} + M_{2,\bar{\imath},t}] \cdot k_t] = m_1 + m_2$ holds. $\square$

B Proof of Lemmas on Security Analysis

Proof of Lemma 3. We need to show that $\mathcal{C} = \mathbb{F}_{q,d}^{n\times n}[X]$, i.e. that each $C(X) \in \mathbb{F}_{q,d}^{n\times n}[X]$ can be a ciphertext. Any $C(X) \in \mathcal{C}$ produced by algorithm 2 has a form of $C(X) = G(X)+\mathbf{M}$, where $G(X)$ may be an arbitrary polynomial $\in \mathbb{F}_{q,d,zero}^{n\times n}[X]$, i.e. s.t. $\exists \mathbf{K}\ G(\mathbf{K}) = \mathbf{0}$.

Let's look at $\mathbf{M} = \{M_{i,j}\}_{i=\overline{1,n},j=\overline{1,n}}$. It's independent on $G(X)$ and may be any matrix $\in \mathbb{F}_q^{n\times n}$. Indeed, like in algorithm 2 let's fix some arbitrary $M_{i,j} \in \mathbb{F}_q$ for $i = \overline{1,n}, j = \overline{1,n}$ and indices $\bar{i}, \bar{j}$. Element $M_{\bar{i},\bar{j}}$ is recounted due to formula $M_{\bar{i},\bar{j}} := k_{\bar{j}}^{-1} \cdot [m - \sum_{t=1,t\neq\bar{j}}^{n} M_{\bar{i},t} \cdot k_t]$ for plaintext m and secret vector $\vec{k} = (k_1, ..., k_n)$ s.t. $k_j \neq 0$. Recall that we assume that for any m $Pr_{\mathfrak{P}}\{m\} > 0$. And due to this assumption by varying $m \in \mathbb{F}_q$ and $\vec{k} \in \mathbb{F}_q^n \setminus \{\vec{0}\}$ one may obtain $\forall M_{\bar{i},\bar{j}} \in \mathbb{F}_q$ for any fixed values $M_{i,j}, (i,j) \neq (\bar{i},\bar{j})$. And this means that $\mathbf{M}$ may be any matrix $\in \mathbb{F}_q^{n\times n}$.

Now we would like to note that in fact $\forall F(X) \in \mathbb{F}_{q,d}^{n\times n}[X]$ may be represented as the sum $F(X) = G(X) + \mathbf{M}$, where $G(X) \in \mathbb{F}_{q,d,zero}^{n\times n}[X]$ and $\mathbf{M} \in \mathbb{F}_q^{n\times n}$. Indeed, one way simply write $F(X) = (F(X) - F(\mathbf{K})) + F(\mathbf{K})$ for arbirary $\mathbf{K} \in \mathbb{F}_q^{n\times n}$ and then $G(X) = F(X) - F(\mathbf{K})$, $\mathbf{M} = F(\mathbf{K})$.

Putting all things together we obtain $\mathcal{C} = \mathbb{F}_{q,d}^{n\times n}[X]$. □

Proof of Lemma 5. First let's take some $C_0(X) \in \mathcal{C}$ s.t.

$$\mathsf{Decrypt}(\mathbf{sk}, C_0(X)) = 0$$

and compose a matrix $\mathbf{A}^{\bar{i},\bar{j},\alpha} = \{A_{i,j}^{\bar{i},\bar{j},\alpha}\}_{i=\overline{1,n},j=\overline{1,n}} \in \mathbb{Z}_q^{n\times n}$, s.t. $A_{\bar{i},\bar{j}}^{\bar{i},\bar{j},\alpha} = k_{\bar{j}}^{-1}\cdot\alpha$ and $A_{i,j}^{\bar{i},\bar{j},\alpha} = 0$ for others (i,j), where $\bar{j} = min\{j \in \{1,...,n\} s.t. k_j \neq 0\}$, $\alpha \in \mathbb{F}_q$ – arbitrary constant. Adding $\mathbf{A}^{\bar{i},\bar{j},\alpha}$ to $C_0(X)$'s free coefficient gives $C_\alpha(X) \in \mathcal{C}$. And it's easy to verify that $\mathsf{Decrypt}(\mathbf{sk}, C_\alpha(X)) = \alpha$. This implies that by this approach one may construct the set of ciphertexts $\mathsf{C} = \{C_1(X), ..., C_{q-1}(X)\}$ corresponding to the zero encryption $C_0(X)$, where $C_i(X)$ – encryption of plaintext $i \in \mathbb{F}_q$ on $\mathbf{sk}$.

If there are encryptions of zero $C_{1,0}(X), C_{2,0}(X) \in \mathcal{C}$, $C_{1,0}(X) \neq C_{2,0}(X)$ then for sets $\mathsf{C}_i = \{C_{i,1}(X), ..., C_{i,q-1}\}, i = 1,2$ it's easy to see that $\mathsf{C}_1 \cap \mathsf{C}_2 = \emptyset$. The last one means that for a fixed $\mathbf{sk}$ one may split $\mathcal{C}$ into q subsets $\mathcal{C}_0, ..., \mathcal{C}_{q-1}$, where $\forall C(X) \in \mathcal{C}_i$ – encryption of i (and thus $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$

for $i \neq j$) and $|C_0| = ... = |C_{q-1}|$. This proves that for $C(X) \xleftarrow{\$} \mathcal{C}$, $Pr\{\mathsf{Decrypt}(\mathbf{sk}, C(X)) = m\} = 1/q$ holds for any $m \in \mathbb{F}_q$ and fixed $\mathbf{sk}$. □

Proof of Lemma 6. Any $C(X) \in \mathcal{C}$ produced by algorithm 2 for a fixed $\mathbf{sk} = (\mathbf{K}, \vec{k}, \bar{i})$ has the form of $C(X) = F(X) - F(\mathbf{K}) + \mathbf{M}$, where coefficients $F_i, i = \overline{0,d}$ are random independent variables s.t $F_i \xleftarrow{\$} \mathbb{Z}_q^{n\times n}$. So $C_i = F_i, i = \overline{1,d}$ and $C_0 = F_0 - F(\mathbf{K}) + \mathbf{M}$. Since $F_0 - F(\mathbf{K})$ is constant after sampling $F_i, i = \overline{1,d}$ the variability of C_0 is then fully determined by $\mathbf{M}$ that's independent of F_i and $\mathbf{K}$. Due to the encryption algorithm for fixed m and $\vec{k}$ $\mathbf{M}$ may be one of the q^{n^2-1} different possible matrix. Thus we obtain that $|\mathcal{C}_{m,\mathbf{sk}}| = q^{d\cdot n^2} \cdot q^{n^2-1} = q^{(d+1)\cdot n^2-1}$.

Also note that

$$Pr\{\mathbf{M}\} = (\prod_{(i,j),(i,j)\neq(\bar{i},\bar{j})} Pr\{M_{i,j}\})\cdot$$

$$\cdot Pr\{M_{\bar{i},\bar{j}} | M_{\bar{i},1}, .., M_{\bar{i},\bar{j}-1}, M_{\bar{i},\bar{j}+1}, .., M_{\bar{i},n}\}.$$

$Pr\{M_{\bar{i},\bar{j}} | M_{\bar{i},1}, .., M_{\bar{i},\bar{j}-1},\ M_{\bar{i},\bar{j}+1}, .., M_{\bar{i},n}\} = 1$ obviously holds and then

$$Pr\{\mathbf{M}\} = (1/q)^{n^2-1}.$$

Now let's estimate the probability distribution over $\mathcal{C}_{m,\mathbf{sk}}$ induced by algorithm 2 for fixed $m, \mathbf{sk}$. For $\forall C(X) \in \mathcal{C}_{m,\mathbf{sk}}$ we have

$$Pr\{C(X)|m, \mathbf{sk}\} = (\prod_{i=1}^{d} Pr\{C_i\}) \cdot Pr\{C_0|C_1, ..., C_d\} =$$

$$(1/q)^{d\cdot n^2} \cdot Pr\{C_0|C_1, ..., C_d\}.$$

And

$$Pr\{C_0|C_1, ..., C_d\} =$$

$$= \sum_{(F_0-F(K),M), F_0-F(K)+M=C_0} Pr\{F_0 - F(K)|F_1, ..., F_d\} \cdot Pr\{M\} = (1/q)^{n^2-1}.$$

The last one is true because $Pr\{F_0 - F(K)|F_1, ..., F_d\} = 1$ and clear there is only one matrix M in the sum for constant $F_0 - F(K)$. Therefore it implies $Pr\{C(X)|m, \mathbf{sk}\} = 1/q^{(d+1)\cdot n^2-1}$.

□

Optimizing Memory Cost of Multi-scalar Multiplication

Sergey Grebnev

Abstract

We present a modification of the non-adjacent form representation for multi-scalar multiplication which allows efficient performance-memory tradeoffs. Simultaneously we fix a feature of the original algorithm which prevented it from using all possible bases.

Keywords: multi-scalar multiplication, non-adjacent form, digital signature, performance evaluation

1 Introduction

Multi-scalar multiplication, i.e. computation of the form $k_1P + k_2Q$, where P and Q are both elements of a finite group and k_1, k_2 are integers, is an essential part of many cryptographic primitives. Consider, for example, Russian national digital signature standard scheme GOST R 34.10-2012. The scheme is built over a subgroup of points $< P >$ of an elliptic curve E over a prime field, where $\# < P > \approx 2^{256}$ or $\# < P > \approx 2^{512}$. The signature verification process is defined by the equation

$$x_{(sh(M) \bmod q)P-(rh(M) \bmod q)Q)} \pmod q = r,$$

where (r, s) is a signature, $h(M)$ is a hashed message, and $Q = dP$ is the public key; the signature is valid iff the equality holds. Thus, improving efficiency of multi-scalar multiplication is crucial for implementations of the signature scheme.

Our previous work [2] considered several possibilities of improving performance of GOST R 34.10-2012, including state-of-the-art choice of an elliptic curve representation and optimisation of scalar and multi-scalar multiplication. Recent paper deals with multi-scalar multiplication exclusively. We recall an algorithm by M. Kalinin which was sketched in [2], fix its minor features and present a novel variant of joint multibase form which allows for more flexible memory management.

2 Multi-scalar multiplication

The computation of k_1P+k_2Q may be performed as two successive scalar multiplications k_1P, k_2Q and an addition. However, it is possible to use so-called Shamir's trick, that is, jointly scan binary representations of k_1, k_2, performing at each step a doubling and additions of P, Q or $P+Q$, if the corresponding bits are set, and save on excessive multiplications. We can do even better, using specially optimized sparse representations to save on additions.

A common scheme for multi-scalar multiplication may be decribed as follows:

1. pre-compute a set of points iP, jQ;
2. build a special joint representation of the pair (k_1, k_2);
3. set the result to infinity, then "scan" the representation of exponents, adding a precomputed point depending on the corresponding digit and multiplying by the bases associated with the digit.

The most commonly used method is the joint sparse form, introduced in [9].

The development of elliptic curves cryptosystems gave rise to extensive study of their equivalent representations, such as Edwards curves, Hesse curves etc., see [3, 4]. Many of these enjoy faster basic operations, e.g. doubling and triplings. In order to use their advantages in scalar and mult-scalar multiplications, a number of special representations was designed, to mention joint double-base number system and joint double-base chains [1].

Window-based multibase representations, which were proposed in [6, 8] for scalar multiplications, were later adapted for mult-scalar multiplications in [5]. We proceed with their description.

A *multibase non-adjacent form with window w (wmbNAF)* of an integer k, denoted $(a_1, \dots, a_J)NAF_w(k) = \{d_1^{(a_1)} \dots d_m^{(a_m)}\}$, is a sequence of digits d_i, each one associated with a base a_i from a fixed finite set $A = \{a_1, \dots, a_J\}$ such that:

1. any positive d has a unique representation of the form $(a_1, \dots, a_J)NAF_w(d)$ for the set of coprime bases A and window w;
2. of any w consecutive digits, at most one is non-zero;
3. $d_i \in \{0, \pm 1, \pm 2, \dots, \pm\lfloor (a_1^w - 1)/2 \rfloor\} \backslash \{\pm a_1, \pm 2a_1 \dots, \pm \lfloor (a_1^{w-1} - 1)/2 \rfloor a_1\}$, $d_1 > 0$;
4. $k = (\dots((d_1 \cdot (a_2) + d_2) \cdot (a_3)) + \dots + d_{m-1}) \cdot (a_m) + d_m$; the latter formula, where (a_i) denotes the base associated with the i-th digit, gives a natural algorithm for scalar multiplication.

Please note that we keep the original notations from [6, 8] hereafter, so the term $d_i = d^{(a_j)}$ should be read as "set the i-th digit to the value d and set the associated base to a_j". The reader should not be confused by the notation (a_i), which denotes the base actually written at the i-th position of the wmbNAF, *not* its index in A.

A *joint multibase non-adjacent form with window w (JNAF)* of a pair of integers k_1, k_2 is actually a pair of $wmbNAF$s of length l, maybe left-padded with zeroes for a smaller integer, calculated for k_1, k_2 simultaneously in such a manner that the bases (a_i) associated with each position coincide:

$$(a_1, \dots, a_J)NAF_w(k_1, k_2) = \begin{pmatrix} (d_1^{(a_1)}, \dots, d_l^{(a_l)}) \\ (e_1^{(a_1)}, \dots, e_l^{(a_l)}) \end{pmatrix}, \text{ and}$$

$$k_1P + k_2Q = (a_{l-1})(\dots(a_3)((a_2)(d_1P + e_1Q) + d_2P + e_2Q) + \dots \\ + d_{l-1}P + e_{l-1}Q) + d_lP + e_lQ;$$

the latter formula gives a natural algorithm for multi-scalar multiplication.

The paper [5] presents algorithms to compute $JNAF$ in two slightly different ways, denoted

$$(a_1,\dots,a_J)\overrightarrow{NAF}_w(k_1,k_2) \quad (\text{“left-to-right”})$$

and

$$(a_1,\dots,a_J)\overleftarrow{NAF}_w(k_1,k_2) \quad (\text{“right-to-left”})$$

depending on the order in which the bases are treated. Since the paper is not widely availiable, we provide a sketch of the algorithm.

Algorithm 1.

Input: $k_1, k_2 \in \mathbb{N}$, $J \geq 1$, $A = \{a_1, a_2, \dots, a_J\}$ *– an ascending set of different prime bases,* $j = 1,\dots,J$*,a window* $w \in \mathbb{N}, w \geq 2$.

Output:

```
$(a_1,\dots,a_J)\overrightarrow{NAF}_w(k_1,k_2) = \begin{pmatrix} (d_1^{(a_1)},\dots,d_l^{(a_l)}) \\ (e_1^{(a_1)},\dots,e_l^{(a_l)}) \end{pmatrix}$
1. i=1;
2. **while** $k_1 > 0$ **or** $k_2 > 0$
2.1. **for** $j = 1$ **to** $J$ **do** (“Left-to-right” case)
**OR**
**for** $j = J$ **downto** $1$ **do** (“Right-to-left” case)
2.1.1. **if** (($k_1 \bmod a_j = 0$) **and** ($k_2 \bmod a_j = 0$)) **then**
2.1.1.1 $d_i = 0$, $k_1 = k_1/a_j$, $d_i = d_i^{(a_j)}$;
2.1.1.2 $e_i = 0$, $k_2 = k_2/a_j$, $e_i = e_i^{(a_j)}$;
2.1.1.3 **break**;
2.1.2. **else if** (($k_1 \bmod a_j = 0$)**and** ($k_2 \bmod a_j \neq 0$)) **then**
2.1.2.1 $e_i = k_2 \operatorname{mods} {a_1}^w$,
2.1.2.2 $k_2 = k_2 - e_i$;
2.1.2.3 **if** ($k_2 \bmod a_j = 0$) **then**
2.1.2.3.1 $d_i = 0$, $d_i = d_i^{(a_j)}$;
2.1.2.3.2 $e_i = e_i^{(a_j)}$;
2.1.2.3.3 $k_1 = k_1/a_j$, $k_2 = k_2/a_j$;
2.1.2.3.4 **break**;
```

```
2.1.2.4 else $k_2 = k_2 + e_i$;
2.1.3. else if $((k_1 \bmod a_j \neq 0)$ and $(k_2 \bmod a_j = 0))$ then
2.1.3.1 $d_i = k_1 \operatorname{mods} {a_1}^w$,
2.1.3.2 $k_1 = k_1 - d_i$;
2.1.3.3 if $(k_1 \bmod a_j = 0)$ then
2.1.3.3.1 $e_i = 0$, $e_i = e_i^{(a_j)}$;
2.1.3.3.2 $d_i = d_i^{(a_j)}$;
2.1.3.3.3 $k_1 = k_1/a_j$, $k_2 = k_2/a_j$;
2.1.3.3.4 break;
2.1.3.4 else $k_1 = k_1 + d_i$;
2.2 if $j > J$ then
2.2.1 $d_i = k_1 \operatorname{mods} {a_1}^w$;
2.2.2 $k_1 = k_1 - d_i$;
2.2.3 $e_i = k_2 \operatorname{mods} {a_1}^w$;
2.2.4 $k_2 = k_2 - e_i$;
2.2.5 for $k = 1$ to $J$ do ("Left-to-right" case)
OR
for $k = J$ downto $1$ do ("Right-to-left" case)
2.2.5.1 if $((k_1 \bmod a_k = 0)$ and $(k_2 \bmod a_k = 0))$ then
2.2.5.1.1 $d_i = d_i^{(a_k)}$;
2.2.5.1.2 $e_i = e_i^{(a_k)}$;
2.2.5.1.3 $k_1 = k_1/a_k$, $k_2 = k_2/a_k$;
2.2.5.1.4 break;
2.3 $i = i + 1$;
3. return $(a_1, \ldots, a_J)\overrightarrow{NAF}_w(k_1, k_2) = \begin{pmatrix} (d_1^{(a_1)}, \ldots, d_l^{(a_l)}) \\ (e_1^{(a_1)}, \ldots, e_l^{(a_l)}) \end{pmatrix}$
```

Remark 1. *Please note that description of the algorithm given in [2] was inaccurate, since in the case of "left-to-right" computation the cycle (step 2.1) was continuously performed for $j = 1$, thus, all bases greater then a_1 were ignored, and the output was always of the form $(2)\overrightarrow{NAF}_w(k_1, k_2)$. The version presented here fixes this feature.*

Given a JNAF, we considered two algorithms for computing $k_1P + k_2Q$, using either precomputed points iP, jQ or $iP \pm jQ$. As shown in [5], the digits of $(a_1, \dots, a_J)\overrightarrow{NAF}_w(k_1, k_2)$

$$d_i, e_i \in \{0, \pm 1, \pm 2, \cdots \pm \lfloor \frac{a_1^w - 1}{2} a_1 \rfloor\} \setminus \{\pm a_1, \pm 2a_1, \dots, \lfloor \frac{a_1^{w-1} - 1}{2} \rfloor\},$$

thus requiring either $a_1^w/2$ or $a_1^w/4$ precomputed points, depending on their form, while the digits of $(a_1, \dots, a_J)\overleftarrow{NAF}_w(k_1, k_2)$

$$d_i, e_i \in \{0, \pm 1, \pm 2, \cdots \pm \lfloor \frac{a_1^w}{2} \rfloor\} \setminus \{\pm 6k, k \in n\},$$

requiring either $\lfloor 5/12 a_1^w \rfloor$ or $\lfloor 25/144 a_1^w \rfloor^2$ precomputed points. Obviously, the algorithms with $iP \pm jQ$ precomputations require too much storage to be efficient in practice. Even for the tables with iP, jQ it may be desirable to reduce storage requirements while keeping performance characteristics. We proceed with the description of proposed solution.

3 Joint fractional multibase non-adjacent form

In 2008, Longa and Gebotys presented *fractional multibase non-adjacent form* which, given a window size w and a fixed set of digits $\mathcal{D}\{\pm 1, \pm 3, \cdots \pm (2t+1)\}$, produced a flexible encoding of an integer in a NAF_w-form, where all the digits are in $\mathcal{D}$. We describe an adjustment of this representation for the case of joint multibase non-adjacent form.

Suppose we have a set of J bases $\{2, a_2, \dots, a_J\}$ (that is $a_1 = 2$). Next, for $d \in \{\pm 1, \pm 3, \cdots \pm m\}$, where $m \geq 3$, m is an odd integer, we precompute $\{d_i P\}$, $\{d_i Q\}$. Fix a window w and let $m = 2^{w-2} + s$ for an odd $s \geq 1$.

Following [7], we write the recoding rule for our representation.

1. **if** $k \equiv 0 \pmod{a_1}$ **or** $k \equiv 0 \pmod{a_2}$ **or** $\cdots$ **or** $k \equiv 0 \pmod{a_J}$ **then** $k_i = 0$;
2. **else if** $0 < r \leq m$ **then** $k_i = r$;
3. **else if** $m \leq r < (3m - 4s)$ **then** $k_i = r - 2^{w-1}$;
4. **else if** $(3m - 4s) \leq r < 2^w$ **then** $k_i = r - 2^w$,

where $r = k \pmod 2)^w$.

Now we give a complete description of the algorithm.

Algorithm 2.

Input: *$k_1, k_2 \in \mathbb{N}$, $\mathcal{A} = \{a_1, a_2, \dots, a_J\}, J \geq 1$ – an ascending set of prime bases with $a_1 = 2$, a window $w \in \mathbb{N}$, $w \geq 2$, a set of digits $\mathcal{D} = \{\pm 1, \pm 3, \cdots \pm m\}$ such that $m = 2^{w-2} + s$, $2^{w-2} < m < 2^{w-1}$ for m, s – odd positive integers.*

Output: *joint fractional non-adjacent form*

$$(a_1, \dots, a_J)NAF_{w,t}(k_1, k_2) = \begin{pmatrix} (d_1^{(a_1)}, \dots, d_l^{(a_l)}) \\ (e_1^{(a_1)}, \dots, e_l^{(a_l)}) \end{pmatrix},$$

$d_i, e_i \in \{\pm 1, \pm 3, \cdots \pm m\}$.

```
1. $i = 0$;
2. while $(k1 > 0)$ or $(k_2 > 0)$ do
2.1. for $j = 1$ to $J$ do
2.1.1. if $((k_1 == 0 \pmod{a_j}))$and$(k_2 == 0 \pmod{a_j})))$ then
2.1.1.1. $d_i = 0^{(a_j)}$, $d_j = 0^{(a_j)}$
2.1.1.2. $k_1 = k_1/a_j$, $k_2 = k_2/a_j$
2.1.1.3. break
2.1.2. else if $((k_1 == 0 \pmod{a_j}))$and$(k_2! = 0 \pmod{a_j})))$ then
2.1.2.1. $r = k_2 \pmod 2)^w$
2.1.2.2. if $0 < r \leq m$ then $e_i = r$
2.1.2.3. else if $m < r \leq (3m - 4s)$ then $e_i = r - 2^{w-1}$
2.1.2.4. else if $(3m - 4s) < r < 2^2$ then $e_i = r - 2^w$
2.1.2.5. $k_2 - = e_i$
2.1.2.6. if $k_2 \equiv 0 \pmod{a_j}$ then
2.1.2.6.1. $d_i = 0^{(a_j)}$, $e_i = e_i^{(a_j)}$
2.1.2.6.2. $k_1 = k_1/a_j$, $k_2 = k_2/a_j$
2.1.2.6.3. break
2.1.2.7. else $k_2 = k_2 + e_i$
2.1.3. else if $((k_1! = 0 \pmod{a_j}))$and$(k_2 == 0 \pmod{a_j})))$ then
2.1.3.1. $r = k_1 \pmod 2)^w$
2.1.3.2. if $0 < r \leq m$ then $d_i = r$
```

```
2.1.3.3. else if $m < r \leq (3m - 4s)$ then $d_i = r - 2^{w-1}$
2.1.3.4. else if $(3m - 4s) < r < 2^2$ then $d_i = r - 2^w$
2.1.3.5. $k_1 -= d_i$
2.1.3.6. if $k_1 \equiv 0 \pmod{a_j}$ then
2.1.3.6.1. $d_i = d_i^{(a_j)}$, $e_i = 0^{(a_j)}$
2.1.3.6.2. $k_1 = k_1/a_j$, $k_1 = k_1/a_j$
2.1.3.6.3. break
2.1.3.7. else $k_1 = k_1 + d_i$
2.2. if $j > J$ then
2.2.1. $r = k_2 \pmod 2^w$
2.2.2. if $0 < r \leq m$ then $e_i = r$
2.2.3. else if $m < r \leq (3m - 4s)$ then $e_i = r - 2^{w-1}$
2.2.4. else if $(3m - 4s) < r < 2^2$ then $e_i = r - 2^w$
2.2.5. $k_2 -= e_i$
2.2.6. $r = k_1 \pmod 2^w$
2.2.7. if $0 < r \leq m$ then $d_i = r$
2.2.8. else if $m < r \leq (3m - 4s)$ then $d_i = r - 2^{w-1}$
2.2.9. else if $(3m - 4s) < r < 2^2$ then $d_i = r - 2^w$
2.2.10. $k_1 -= d_i$
2.2.11. for $l = 1$ to $J$ do
2.2.11.1. if $((k_1 == 0 \pmod{a_l})$and$(k_2 == 0 \pmod{a_l}))$ then
2.2.11.1.1. $d_i = d_i^{(a_k)}$, $e_i = e_i^{(a_k)}$
2.2.11.1.2. $k_1 = k_1/a_k$, $k_2 = k_2/a_k$
2.3. $i = i + 1$
3. Output $(a_1, \ldots, a_J)NAF_{w,t}(k_1, k_2) = \begin{pmatrix} (d_1^{(a_1)}, \ldots, d_l^{(a_l)}) \\ (e_1^{(a_1)}, \ldots, e_l^{(a_l)}) \end{pmatrix}$.
```

4 Performance analysis

We have studied performance of our algorithm according to the generally accepted framework described in [2]. That is, we consider certain efficient representations of a group of points of an elliptic curve E over

$GF(p)$, express complexity of basic operations in E, such as addition, doubling, tripling etc. in terms of multiplication in $GF(p)$), set a series of experiments to determine average characteristics of our representation and compute resulting complexity of the algorithm.

	$(2,3)\overrightarrow{NAF}_6$	$(2,3)\overleftarrow{NAF}_6$	$(2,3)NAF_{6,8}$	$(2,3)NAF_{6,15}$
Jacobian coordinates	6117.5	6115.7	6405	6136.1
Inverted Edwards	4661.8	4986	4832.8	4659.9
Inverted twisted Edwards	4639.5	4736	4837.6	4657.6

Table 1: Average complexity of multi-scalar multiplication in $E(GF(p))$, $p \approx 2^{512}$

No wonder that our algorithm for a window w behaves generally like $\overrightarrow{NAF}_w$. At the same time we are able to reduce memory costs twice, while performance loss is about 4%.

We conclude that the algorithm presented enjoys nice performance characteristics, allows for reduced memory cost and is, in general, more suited for implementation than the original variants of joint multibase NAF.

5 Acknoledgements

This work is supported by Academy of cryptography of the Russian Federation.

References

[1] Doche C., Kohel D.R., Sica F. Double-base nimber systems for multi-scalar multiplications. // Proc. EUROCRYPT 2009, LNCS 5479, 2009. — P. 502–519.

[2] Dygin D., Grebnev S. Efficient Implementation of the GOST R 34.10 Digital Signature Scheme using Modern Approaches to Elliptic Curve Scalar Multiplication // Mat. Vopr. Kriptografii. – 2013. — V. 4 No. 2 — P. 47-57.

[3] Explicit-formulas Database. — // http://hyperelliptic.org/EFD, 2012. Accessed 07.02.2014.

[4] Hisil H. Elliptic Curves, Group Law and Efficient Computation: Ph. D. thesis, Queensland University of Technology. — 2010. — http://qut.edu.au/.

[5] Kalinin M. An Implementation of the GOST R 34.10 Digital Signature Scheme using Novel Algorithms of Elliptic Curve Scalar Multiplication: Master's thesis (in Russian). // MSU, 2010. — 50 p.

[6] Longa P. Accelerating the Scalar Multiplication on Elliptic Curve Cryptosystems over Prime Fields. — http://eprint.iacr.org/2008/100/, 2008. Accessed 07.02.2014.

[7] Longa P., Gebotys C. Setting Speed Records with the (Fractional) Multibase Non-Adjacent Form Method for Efficient Elliptic Curve Scalar Multiplication. CACR technical report, CACR 2008-06. — 2008.

[8] Longa P., Miri A. New Multibase Non-Adjacent Form Scalar Multiplication and its Application to Elliptic Curve Cryptosystems. — http://eprint.iacr.org/2008/052/, 2008. Accessed 07.02.2014.

[9] J. A. Solinas. Low-weight binary representations for pairs of integers. CORR technical report 2001-41, University of Waterloo, — 2001.

A Timing Attack On CUDA Implementations Of An AES-Type Block Cipher

Denis Fomin

Abstract

This work presents a timing attack against an AES-type block cipher CUDA implementation. Our experiments show that it's possible to extract a secret AES 128-bit key with complexity of 2^{32} chosen plaintext encryptions. It's easy to show that all results can be applied for AES with other key sizes and even more: for any block cipher with a linear transform that is a composition of two types of a linear transformations on a substate.

Keywords: AES, Kuznyechik, Grasshopper, timing attack, cache attack, software timing attacks, CUDA, GPU

1 Introduction

A timing attack is a variant of a side-channel attack when an attacker exploits a correlation between the running times of the implementation of a cryptographic algorithm and the values of its input data (plaintext and key) to recover the secret key. Such attacks have been widely studied by researchers recently (for example: [1–4]).

Till now no timing attacks against GPU implementations of ciphers have been published. But there are a lot of different cache-timing attacks against CPU implementation. There exist two publications [8, 9] about timing attacks on CUDA implementations of AES and Blowfish, but the outcome of these works is that this type of attack is impossible, but at the same time these results show that information about bank conflicts in shared memory of NVIDIA GPU may leak some information about the data. In this work we use the same ideas to exploit our attack.

We show an attack against an AES-type block cipher CUDA implementation and also a theoretical possibility of this type of attacks. This attack is based on a NVIDIA GPU architecture, that differs from CPU architecture. In section 2 we explain CUDA architecture and the core idea of the attack. In section 5 we show experimental results for the implementation of our attack on AES-128 block cipher.

2 CUDA in brief

CUDA is a software and hardware architecture developed by NVIDIA company. It allows to produce non-graphical computing using GPUs.

Architecture of modern GPUs is different from CPU architecture. Every modern GPU has several streaming multiprocessors (from 2 to 30 SMX) with 192 CUDA cores (32 in old GPUs) in each of them. Each CUDA core works at low clock frequencies (< 1 GHz), has a register file (256KB in Kepler architecture GPUs or less in the older one) and three different cash types: texture, global (L2) and constant.

CUDA architecture uses their own model of parallelism called SIMT (single instructions multiple thread). Virtually all threads work in parallel and have the same priority of memory access. Threads are grouped into blocks. The global synchronization between threads in different blocks is generally impossible, and for one block of threads is performed through a special memory called shared memory. All threads in a block are divided into groups of 32 called warps. All threads in a warp at the same time perform the same instruction.

There are six types of memory in CUDA architecture: global, shared, registers, constant, texture and local.

Global memory is the largest memory in GPU: from 128MB to more than 20GB in Tesla K80. All GPUs have low access speed to this type of memory and cached in L2 global cache when accesses this type of memory. This type of memory is frequently used for data transfers between GPU and RAM.

Texture memory is a global memory with specific attributes, addressing to texture memory is produced with a special texture cache. It is good for working with textures and for big arrays with random access.

Shared memory is much faster than local and global memories. It is allocated per thread block, so all threads in the block have access to the same shared memory and it is always used as a user cache.

To achieve high memory bandwidth for concurrent accesses, shared memory is divided into equally sized memory modules (banks) that can be accessed simultaneously.

Shared memory of modern GPU has 32 banks consisting of successive 32 or 64 bits words. Each bank has a bandwidth of a word (32 or 64 bit long) per two clock cycles. If $b > 1$ addresses of memory requests fall in the same memory bank, there is a *bank conflict* and the access has to be serialized and takes b times longer. On the other hand, shared memory features a *broadcast* mechanism whereby a word can be read and broadcast to several threads simultaneously when servicing one memory read request. This reduces the number of bank conflicts when several threads read from an address within the same 32-bit word. *A common conflict-free case is when all threads of a warp read from an address within the same word* [5].

On devices with compute capability 2.x and 3.x each multiprocessor has 64KB of on-chip memory that can be partitioned between L1 cache and shared memory. For devices with compute capability 2.x, there are two settings, 48KB shared memory / 16KB L1 cache, and 16KB shared memory / 48KB L1 cache, on device with compute capability 3.x each multiprocessor also has settings with 32KB shared memory / 32KB L1 cache. By default the 48KB shared memory setting is used. This can be configured during runtime API.

Constant memory has a capacity of 64KB, and cached in a special cache of 8KB. Constant memory has short latency and high bandwidth when all threads access the same location. It is used for constant variables and can be read from each thread.

Registers is the fastest memory and have only thread scope. They are physically stored in each multiprocessor and can't be indexing. Local memory does not physically exist and is put in a global memory by the compiler in case a lot of registers or arrays in register memory are used. It is much slower than register memory. Therefore, the program must use a limited number of variables in order to run faster.

3 Description of AES-type ciphers

In this work we assume that an internal state of an AES-type block cipher is represented by two-dimensional arrays of words (like AES representation in [6]). Every internal state consists of N rows and N columns of words. Also we denote AES-type cipher as a block cipher with the following transformations:

- X – round key addition (`AddRoundKey`[1]),
- S – non-linear bijective transform – permutation (`SubBytes`[1]),
- L – linear transformation. Moreover L is a composition of rows transformation (`ShiftRows`[1]) and columns transformation (`MixColumns`[1]).

The core feature of a linear transformation is that it can be decomposed into two linear transformations. Further we consider AES with 128-bit key, but our technique is also applicable to AES with other key size.

Let N be equal to 4 and the internal state is a square matrix of bytes, that is stored row by row:

$$\begin{pmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{pmatrix}$$

Consider the following property of the linear transformation.

Proposition 1. *Let $x_{0,0}, \dots, x_{3,3}$ – an internal state of AES block cipher after the first* `AddRoundKey` *transformation and $a_{00}, \dots, a_{33}$ is an internal state after the first round — before the second* `AddRoundKey`*. Also let we set one value from $\{a_{i,0},\ i = 0, \dots, 3\}$ and any three values from the set $\{x_{i,i},\ i = 0, \dots, 3\}$. Then there exist only one value for the other variable $x_{i,i}$ for these fixed values.*

[1]in AES description [6]

Proof. Without loss of generality we set values for $a_{0,0}$ and $x_{0,0}$, $x_{1,1}$, $x_{2,2}$ and assume that there can be two $x'_{3,3} \neq x''_{3,3}$ values for $x_{3,3}$. Let π be a non-linear byte-permutation for S-transform and l – a matrix of `MixColumns` transformation. Let

$$(\pi(x_{0,0}), \pi(x_{1,1}), \pi(x_{2,2}), 0)\, l = (a_0, a_1, a_2, a_3)$$

and

$$\big(0, 0, 0, \pi(x'_{3,3})\big)\, l = (a'_0, a'_1, a'_2, a'_3)$$
$$\big(0, 0, 0, \pi(x''_{3,3})\big)\, l = (a''_0, a''_1, a''_2, a''_3).$$

So $a_0 \oplus a'_0 = a_0 \oplus a''_0 = a_{0,0}$ and

$$\big(0, 0, 0, \pi(x'_{3,3}) \oplus \pi(x''_{3,3})\big)\, l = 0.$$

The last equation is impossible because $x'_{3,3} \neq x''_{3,3}$ and l is an invertible linear transformation. □

In terms of proposition 1 we have 2^{24} values for $a_{i,0}$ for any fixed value i if we can set $x_{0,0}, x_{1,1}, x_{2,2}, x_{3,3}$ values.

Apparently we can do the same operation with other sets:

can set any value in one of:	when set values:
$\{a_{i,0},\ i = 0, \dots, 3\}$	$\{x_{0,0}, x_{1,1}, x_{2,2}, x_{3,3}\}$
$\{a_{i,1},\ i = 0, \dots, 3\}$	$\{x_{0,1}, x_{1,2}, x_{2,3}, x_{3,0}\}$
$\{a_{i,2},\ i = 0, \dots, 3\}$	$\{x_{0,2}, x_{1,3}, x_{2,0}, x_{3,1}\}$
$\{a_{i,3},\ i = 0, \dots, 3\}$	$\{x_{0,3}, x_{1,0}, x_{2,1}, x_{3,2}\}$

and can do it in 2^{24} different variants.

4 CUDA implementations of a block cipher

A GPU program generally consists of three major parts:

1. loading data to internal memory of the GPU;
2. evaluation on the GPU;
3. loading data from internal memory of the GPU.

Modern CUDA architecture makes it possible to remove the first and the third points of this list if using unified memory or page-locked host memory on some devices. But the implementation of an algorithm generally consists of these three parts.

To achieve maximal throughput of a block cipher implementation t-box lookup tables are used (merged an s-box and a linear transform) [7, 11–13]. This tables can be placed in a shared memory, texture memory or constant memory. Generally, the best place for lookup tables is a shared memory because a thread has a high access speed to this type of memory. Sometimes, when lookup tables can't be placed in a shared memory (they are too large) the implementation with texture memory can make the most efficient implementation [7].

Shared memory access has one important property. When we encrypt random data we have to read random data from the memory, so throughput of the implementation with a shared memory might be faster without bank conflicts. If we can reduce bank conflicts we can expand throughput of our implementation. For example in table 2 we present encryption throughput of three types of encryption:

- the same block encryption multiple times — without bank conflicts;
- random data encryption — a lot of bank conflicts;
- counter encryption — less bank conflicts than random data encryption because the highest bit of counter is changed very seldom.

In table 1 we present hardware and software specification that we used in this work for experiments.

In this article we propose that encryption time is a kernel working time (without moving data between GPU memory and RAM). As we can see in table 2 throughput of a counter mode encryption less than random data encryption in CBC mode and moreover evaluation time is different for this two modes of operations. The core idea of this work is based on the same idea: we can encrypt selected data and find a secret key, when the encryption takes less time.

Table 1: Hardware and software specification

OS	OpenSUSE 13.1	
CUDA compiler	nvcc ver. 6.0	
Graphics accelerator	NVIDIA GTX 285	NVIDIA GTX Titan
CUDA-cores	240	2688
GPU memory	1 GB	6 GB
Processor clock (MHz)	1 476	876
CPU	Intel Core i7-4770K	

Table 2: Encryption throughput of different modes of encryption (GTX Titan), MB/sec

Algorithm	Constant encryption	Random data encryption	Counter encryption
AES 128	31113	12878	14195
AES 256	26489	10960	11867

5 A timing attack on AES-128 block cipher

In [8, 9] authors show that bank conflicts may leak some information about the data. Also they say, that there exists a way to avoid it:

> If the lookup tables are small enough (as in the case of AES) we can create multiple copies of them in the cache and stripe them across bands to make sure that there is always one entire copy of the table available through each bank.([9])

From our point of view one could face the following problems while implementing this approach: the first one is that threads in a bank can evaluate only the same instructions at a time. The second one is the expansion of evaluation time. And the third one is that for $32 \cdot 4$ tables (without last round) we need 128KB of shared memory (32 is a bank quantity) more than have the latest GPUs.

There are a lot of implementations of AES block cipher with different encryption data, keys, t-box allocation, see [9, 11–13]. The fastest one has parameters that are presented in table 3.

Unfortunately we can't use ideas from [10] because for faster implementation we must use a lot of threads and we'll be allowed to use only several 32-bit registers. CUDA architecture doesn't allow to use 128-bit registers (only as a built-in vector of four 32-bit registers).

Table 3: The best AES implementation parameters

T-box allocation	Shared memory
Data allocation	Registers
Round keys allocation	Shared memory
Size of data in thread	128 bit/thread — each thread encrypts full data block

Timing attacks can be implemented when we can establish some sort of a correlation between the running times of the algorithm and the nature of the key and possibly data. This typically happens when the same operation takes different times for different inputs. As we showed earlier, data reading from a shared memory takes different amounts of time and it takes less time when there are no bank conflicts.

Let $K = \{k_{i,j},\ i,j = 0,\dots,3\}$ be a secret key. Assume that $k_{0,0} = 0, k_{1,1} = 0, k_{2,2} = 0$, and $k_{3,3} = 0$. In this way (in terms of proposition 1) we showed in section 3 that we can choose $x_{0,0}, x_{1,1}, x_{2,2}, x_{3,3}$ values 2^{24} times to set $a_{0,0} = 0$ (value and indexes presented as an example). So we can set encryption data array **Pt** as follows. Let array size equals to M plaintext blocks. Each byte of the plaintext block we set up by a random value except the first one. Each first byte we'll choose in such way that $a_{0,0} = 0$. In this way the first byte in each block after the second `AddRoundKey` transformation will be the same and the encryption time will diminish because in each warp all threads of a warp read from an address within the same word (see section 2). So we can get 32 bits of the key K: $k_{0,0} = 0, k_{1,1} = 0, k_{2,2} = 0, k_{3,3} = 0$ if the encryption time will be less than the encryption of M random plaintext blocks.

Let $\alpha, \beta, \gamma, \delta \in GF\left(2^8\right)$. If we add $\alpha, \beta, \gamma, \delta$ to each plaintext in **Pt** as follows:

add α to $(0,0)$ byte
add β to $(1,1)$ byte
add γ to $(2,2)$ byte
add δ to $(3,3)$ byte
we can specify 32 bits of a key $K = \{k_{i,j},\ i,j = 0,\dots,3\}$: $k_{0,0} = \alpha, k_{1,1} = \beta, k_{2,2} = \gamma, k_{3,3} = \delta$. So we can find 32 bits of a key if we can encrypt our array **Pt** (that is chosen for each encryption) 2^{32} times and find in what step the encryption time is the least. The number of the step with the minimal encryption time corresponds to the right choice of $\alpha, \beta, \gamma, \delta$ for 32 bits of the key.

Apparently, we can find other 12 bytes of the key independently and find all bytes of the key only for $3 \cdot 2^{32}$ array encryptions. So we can find all 128 bits of the key with complexity $4 \cdot 2^{32}$ array encryptions. More than that we can find all 128 bits of the key with only 2^{32} data encryptions, if we can encrypt specially $a_{0,0} = 0$, $a_{0,1} = 0$, $a_{0,2} = 0$, $a_{0,3} = 0$ for each of four key parts.

In table 4 we present the size of data array and encryption time with broadcast (**Pt** array in key find case) and without it (random data encryption time). Encryption time is a kernel working time (without moving data between GPU memory and RAM). As we can see in table 4 the minimum possible time to encrypt one array is about 2 ms.

To employ this attack an attacker must know the evaluation time. We propose that he has the access to the computer by malware or in some other manner but hasn't got a key. Also we propose that he can encrypt any data with an unknown AES key. 2014 year presents some new software bugs like Shellshock that can allow an attacker to gain unauthorized access to a computer system [14]. In our opinion it isn't too difficult to know a kernel working time (if you have access to a computer) because, for example, we can evaluate any program from NVIDIA Visual Profiler. There is no need to have root privileges to run a program from it and furthermore the profiler can be used in a command-line mode.

We run an experiment to find a secret key in the following way: we proposed that we knew 15 bytes of a key and tried to find the rest of it. We encrypted data with **Pt** array size that equals $M = 33\,554\,432$ on GTX

Table 4: Encryption time for different M with broadcast and without it, sec

GPU	encryption time by GTX Titan		encryption time by GTX 285	
M	the wrong choice of $\alpha, \beta, \gamma, \delta$	the right choice of $\alpha, \beta, \gamma, \delta$	the wrong choice of $\alpha, \beta, \gamma, \delta$	the right choice of $\alpha, \beta, \gamma, \delta$
524 288	—	—	0.00196643	0.00196088
1 048 576	0.00224006	0.0022334	0.00374532	0.00373562
2 097 152	0.00436898	0.00435408	0.00730337	0.00728249
4 194 304	0.00862449	0.00859434	0.0144648	0.0144242
8 388 608	0.0142342	0.0141866	0.02874	0.0286593
16 777 216	0.0283605	0.0282638	0.0572906	0.057132
33 554 432	0.0566222	0.0564296	—	—
67 108 864	0.113143	0.112767	—	—
134 217 728	0.226171	0.225415	—	—

Titan and $M = 16\,777\,216$ on GTX 285. We tried to find $(3, 3)$ byte of the key. As presented in figures 1 and 2 we could find the shortest encryption time and find the right key.

As we can see it is quite easy to implement this attack.

On February 2015 an attack recovering all 128 bits of a key were implemented on eight NVIDIA Tesla K10.G2.8GB that were allowed by the NVIDIA Technology Center. The whole key was recovered in less than 12 days.

6 Comparison with other LSX-block ciphers

As presented in this article we use the specific properties of AES block cipher: it has small lookup tables and easy linear transformation: it is a good property for implementations. If we have an LSX-block cipher with MDS-linear transform like in Kuznyechik (Grasshopper) [15] we can't use this attack: to set any value in the second round after the second X transform we must define all 16 bytes of internal state. But for AES we must set only 4 bytes. So, if we want to know the first 128-bit round key of Kuznyechik we must encrypt a specific array 2^{128} times and if we want to

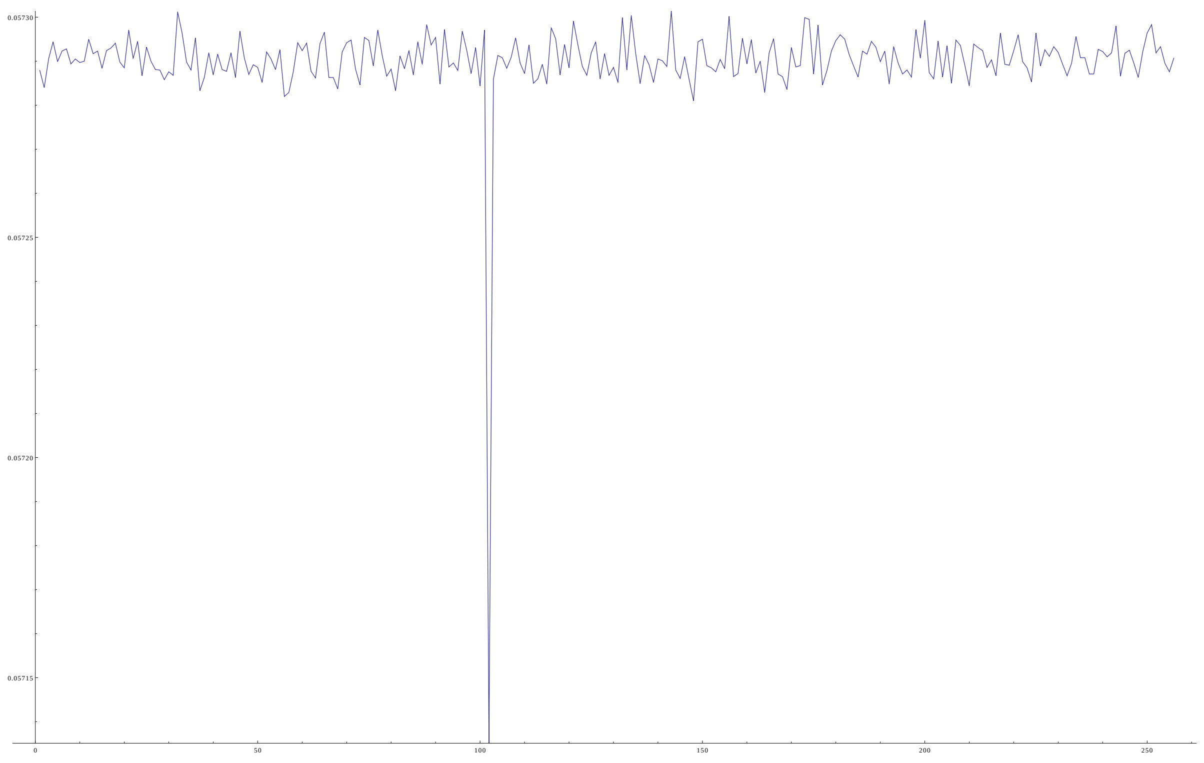

Figure 1: Timing attack on GTX 285 (256MB array size)

know the first 128-bit round key of an AES block cipher we must encrypt a specific array only 2^{32} times.

On the other hand if a linear transform of a LSX block cipher may be represented as a composition of two or more linear transforms on substate space the attack might be employed.

7 Conclusions

It this article we have shown a possibility of a timing attack on an AES-type block cipher CUDA implementations. We've shown that we can recover a secret 128-bit key with the complexity of 2^{32} specific data encryptions. We use specific properties of an AES block cipher: it has small lookup tables and a simple linear transformation. Also we analyse GPU architecture to show a theoretic capability of this type of attacks.

This attack can be applied also for AES with other key sizes and more

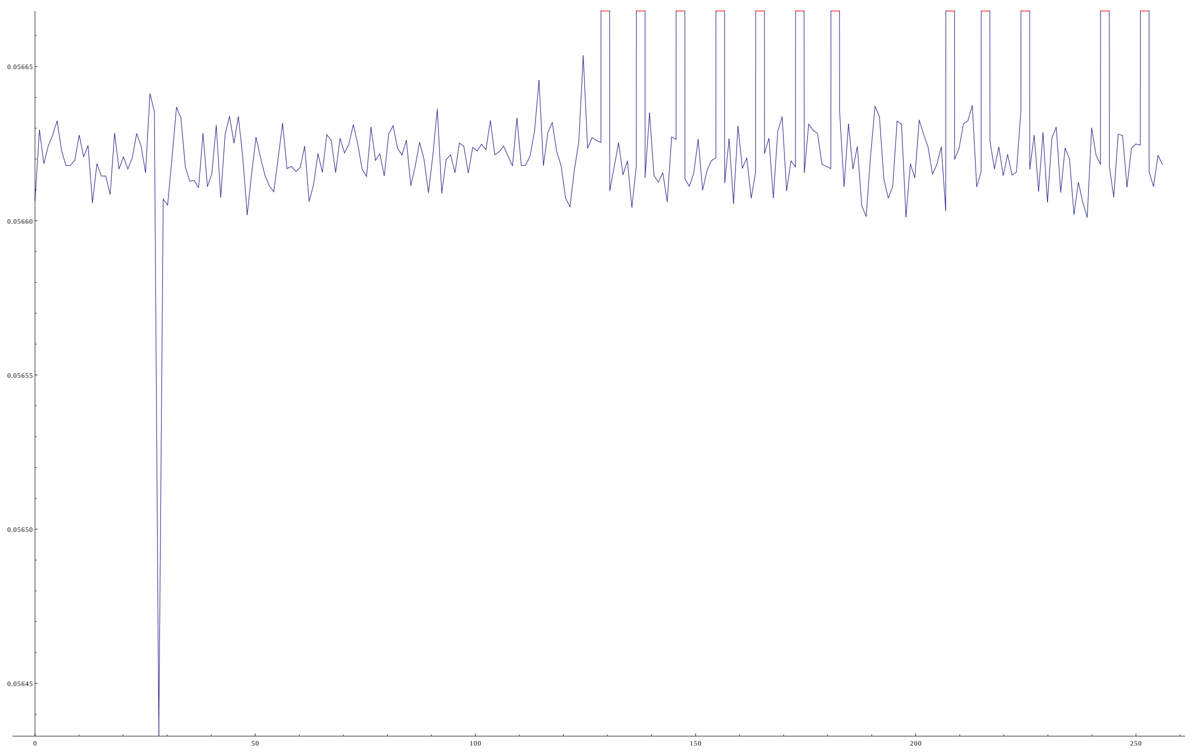

Figure 2: Timing attack on GTX Titan (512MB array size)

than that: for any AES-type block cipher. But if we have an LSX-block cipher with MDS-linear transform like in Kuznyechik [15] we can't mount this attack.

8 Acknowledgement of Reviewers

We would like to thank the reviewers for their helpful remarks.

References

[1] D. Page Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel. University of Bristol, 2002.

[2] D. J. Bernstein. Cache-timing attacks on AES. The University of Illinois at Chicago.

[3] P. Kocher: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. In: N. Koblitz (ed.): Crypto 1996, Springer, Lecture Notes in Com- puter Science 1109, Heidelberg 1996, 104–113.

[4] W. Schindler: A Timing Attack against RSA with the Chinese Remainder Theorem. In: C.K. Koc, C. Paar(eds.): Cryptographic Hardware and Embedded Sys- tems — CHES 2000, Springer, Lecture Notes in Computer Science 1965, Berlin 2000, 110–125.

[5] CUDA Toolkit documentation. — http://docs.nvidia.com/

[6] National Institute of Standards and Technology (NIST), "FIPS-197 : Advanced Encryption Standard (AES)", 2001.

[7] D. Fomin. Implementation of an XSL block cipher with MDS-matrix linear transformation on NVIDIA CUDA, CTCrypt'14, 2014.

[8] R. Mukherjee et al. Fast, Scalable, and Secure encryption on the GPU. — Gachibowli, Hyderabad, India

[9] R. Mukherjee. A Performance Prediction Model for the CUDA GPGPU Platform, Ph.D. Thesis // International Institute of Information Technology Hyderabad, India: 2010.

[10] E. Käsper, Peter Schwabe: Faster and Timing-Attack Resistant AES-GCM. CHES 2009: 1-17

[11] K. Iwai, N. Nishikawa, T. Kurokawa. Acceleration of AES encryption on CUDA GPU. International Journal of Network and Computing, vol. 2, num. 1, — P. 131-145, January 2012

[12] M. Kipper, J. Slavkin, D. Denisenko. Implementing AES on GPU. University of Toronto, April 20, 2009

[13] S. A. Manavski. CUDA COMPATIBLE GPU AS AN EFFICIENT HARDWARE ACCELERATOR FOR AES CRYPTOGRAPHY. IEEE International Conference on Signal Processing and Com-

munications (ICSPC 2007), 24-27 November 2007, Dubai, United Arab Emirates

[14] L. Seltzer (29 September 2014). ”Shellshock makes Heartbleed look insignificant”. ZDNet. Retrieved 29 September 2014. — http://www.zdnet.com/article/shellshock-makes-heartbleed-look-insignificant/, 08.10.2014.

[15] V. Shishkin et al. Low-Weight and Hi-End: Draft Russian Encryption Standard (invited talk), CTCrypt’14, 2014.

Fault Analysis of Kuznyechik

Riham AlTawy Onur Duman Amr M. Youssef

Abstract

Kuznyechik is an SPN block cipher that has been chosen recently to be standardized by the Russian federation as a new GOST cipher. In this paper, we present two fault analysis attacks on two different settings of the cipher. The first attack is a differential fault attack which employs the random byte fault model, where the attacker is assumed to be able to fault a random byte in rounds seven and eight. Using this fault model enables the attacker to recover the master key using an average of four faults. The second attack considers the cipher with a secret sbox. By utilizing an ineffective fault analysis in the byte stuck-at-zero fault model, we present a four stage attack to recover both the master key and the secret sbox parameters. Our second attack is motivated by the fact that, similar to GOST 28147-89, Kuznyechik is expected to include the option of using secret sbox based on the user supplied key to increase its security margin. Both the presented attacks have practical complexities and aim to demonstrate the importance of protecting the hardware and software implementations of the new standard even if its sbox is kept secret.

Keywords: Kuznyechik, Differential fault analysis, Ineffective fault analysis, GOST-Grasshopper.

1 Introduction

A draft for a new block cipher called Kuznyechik (Grasshopper in Russian) was presented at CTCrypt 2014 [19]. This new cipher is the result of a project for a new standard for block cipher encryption algorithm [2] published by the Russian Federation. Kuznyechik is intended to accompany the current Russian encryption standard GOST 28147-89 [1] as a new member of the GOST family of ciphers [2]. Although the current standard is considered a lightweight cipher [17], and only theoretical attacks on the full round cipher have been presented [12, 10], it operates on 64-bit blocks of data which is not sufficient for the current requirements [19]. Hence, the need arose for a new standard with a larger block length which is intended to supersede the current GOST 28147-89 cipher in the future. Recently, a meet in the middle attack on a reduced round version of Kuznyechik was presented in [4]. In this paper, we analyze the resistance of the cipher to fault analysis attacks where the standard sbox is used in the first case and a secret sbox is employed in the second one.

Fault analysis is an implementation dependent attack where the attacker applies some kind of physical intervention during the computation of the internal state of the primitive which corrupts random or known bits in the state. Consequently, the attacker observes the correct and faulty outputs and performs her analysis to gain non negligible information about the secret material embedded in the hardware. Fault injection can be done in many ways which include power glitches, clock pulses, and laser radiation [20, 9].

Fault analysis was first introduced when Boneh *et al.* showed how the private key of the RSA-CRT-algorithm can be successfully recovered by observing the correct ciphertext and then injecting a fault and acquiring the faulty ciphertext [6]. Afterwards, the idea was generalized by Biham and Shamir with the introduction of differential fault analysis (DFA) [5]. DFA combines fault analysis with differential cryptanalysis where the difference between faulty and genuine ciphertexts is exploited. DFA attacks have been widely used for the analysis of block ciphers and hash functions (e.g., see [11, 21, 14, 3]). In particular, AES has received a lot of attention with regards to DFA where some of the works used fault injection in the encryption process [21, 11], and others attacked the key schedule [13].

Contrary to DFA, ineffective fault analysis (IFA) [8] is another form of fault analysis which deduces information about the secret material when the induced fault has no effect on the output. In other words, we consider a fault injection successful when both the faulty and original ciphertexts are equal. Accordingly, one knows that the value of the faulted data is similar to the genuine one. The use of IFA is particularly interesting because a common countermeasure to detect fault injections is the use of dual execution branches where the encryption process is executed twice and the output is withheld if a difference in the two ciphertexts is detected. Accordingly, using IFA in our analysis easily bypasses this countermeasure because we gain knowledge of a fault injection only when the two ciphertexts are equal, which is the case that is not detected by the use of dual execution branches. Additionally, the stuck-at-zero fault model assumes that multiple bits are reset to zero by a fault injection. The practical feasibility of bit-reset fault injections has been demonstrated in a set of experiments [18, 15]. In fact, during experimenting with laser fault injection, the rate of occurrence of multiple bit-reset faults was reported to be much higher than that of bit-flip faults [18].

In addition to its use in the analysis of IDEA [7], IFA has been used to reverse engineer AES with secret parameters in [8]. The idea of reverse engineering using fault analysis to recover the adopted secret sbox has also been applied to the current Russian standard GOST 28147-89 [22], where the authors presented three algebraic fault analysis attacks to recover different combinations of its secret parameters.

Fault analysis attacks vary in the number of required faults depending on the employed fault model. Generally, all models assume that the attacker has access to the physical device, and is able to reset it to the same unknown initial settings as often as needed. Furthermore, different assumptions with respect to the amount of control the attacker has over the position and the Hamming weight of the induced faults are employed.

In this work, we present two fault analysis attacks on Kuznyechik. The first attack is a differential fault analysis attack that adopts the random byte fault model which is considered the most practical fault model. Using this model, the attacker is assumed to be able to fault a random state byte in rounds seven and eight. In this attack, we adapt the attack presented on AES in [16] to analyze Kuznyechik by using an equivalent representation of the last round which enables us to bypass the optimal diffusion effect of the last linear transformation. Our tweak enables a practical and efficient retrieval of the last round key and hence, one can peel off the last round and retrieve the round key used in the second to last round. The knowledge of the last two round keys allows us to invert the key schedule and recover the master key. The second attack is an ineffective fault analysis attack that considers Kuznyechik with a secret sbox. This attack employs a stuck-at-zero fault model where the attacker is assumed to be able to rest the value of a state byte to zero and observe the output of the cipher. Accordingly, one can verify that the value of the genuine state byte is zero when both the faulty and original outputs are equal. Our attack utilizes some of the approaches used to analyze AES with secret parameters [8]. However, unlike AES where the sbox is derived by utilizing the relations between different round keys, we propose a four stage approach where we employ an iterative stage in which we efficiently solve a system of linear equations in $GF(2^8)$ to retrieve multiple sets of candidate parameters. Afterwards, for each candidate set we recover the first two round keys which subsequently enables the retrieval of a set of candidate master keys. Finally, we filter the acquired set of master keys by testing them with a known plaintext-ciphertext pair to recover the right master key and the secret sbox entries. Our second attack demonstrates that trying to increase the security of implementations by having transformations with private parameters is not an adequate measure for protecting the implementation against fault analysis attacks. This fact is particularly interesting for Kuznyechik which, similar to GOST 28147-89, is expected to allow deployment with a secret sbox.

The rest of the paper is organized as follows. In the next section, the description of the Kuznyechik block cipher along with the notation used throughout the paper are provided. Afterwards, in section 3, we provide a detailed description of our differential fault analysis of the cipher. Our four stage ineffective fault analysis attack on Kuznyechik with a secret sbox is given in section 4. Finally, the paper is concluded in section 5.

2 Specification of Kuznyechik

Kuznyechik [19, 2] is an SPN block cipher that operates on a 128-bit state. The cipher employs a 256-bit key which is used to generate ten 128-bit round keys. As depicted in Figure 1, the encryption procedure updates the 16-byte state by iterating the round function for nine rounds. The round function consists of:

- SubBytes (S): A nonlinear byte bijective mapping.

- Linear Transformation (L): An optimal diffusion operation that operates on a 16-byte input and has a branch number = 17. This transformation can also be seen as a row left multiplication by a 16×16 byte matrix whose coefficients $\alpha_{i,j}$ denote the coefficient at row, i, and column, j, for $i, j = 0, 1, \cdots, 15$.
- Xor layer (X): Mixes round keys with the encryption state.

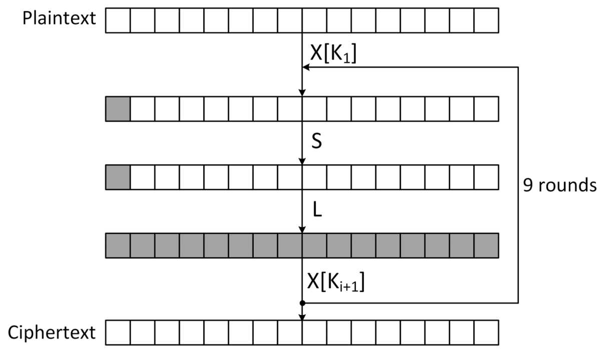

Figure 1: Encryption procedure

Additionally, an initial XOR layer is applied prior to the first round. The full encryption function where the ciphertext C is evaluated from the plaintext P is given by:

$$C = (X[K_{10}] \circ L \circ S) \circ \cdots \circ (X[K_2] \circ L \circ S) \circ X[K_1](P)$$

In our first attack, we use an equivalent representation of the last round function. The representation exploits the fact that both the linear transformation, L, and the Xor operation, X, are linear and thus, their order can be swapped. One has to first Xor the data with an equivalent round key, then apply the linear transformation, L, to the result. We evaluate the equivalent round key after the last round r by $EK_{r+1} = L^{-1}(K_{r+1})$. For further details regarding the employed sbox and linear transformation, the reader is referred to [19].

Key schedule: The ten 128-bit round keys are derived from the 256-bit master key by undergoing 32 rounds of a Feistel structure function. The first two round keys, K_1 and K_2, are derived directly from the master key, K, as follows: $K_1 \parallel K_2 = K$. As depicted in Figure 2, each pair of subsequent round keys is extracted after eight rounds of execution. During each round, the same round function used in the encryption procedure is applied to the right half of the input to the Feistel round. However, round constants are used with the X operation instead of round keys. The 128-

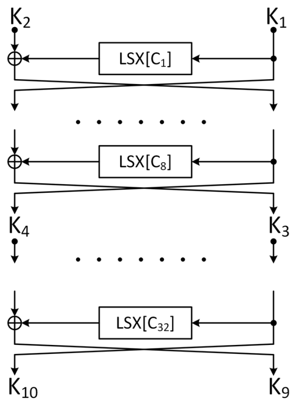

Figure 2: Key schedule

bit round constants C_i are defined as follows: $C_i = L(i),\ i = 1, 2, \cdots, 32$. Let $F[C](a, b)$ denote $(LSX[C](a) \oplus b, a)$, where C, a, and b are 128-bit inputs. The rest of the round keys are derived from the first two round keys, K_1 and K_2, as follows:

$$(K_{2i+1}, K_{2i+2}) = F[C_{8(i-1)+8}] \circ \cdots \circ F[C_{8(i-1)+1}](K_{2i-1}, K_{2i}),\ i = 1, 2, 3, 4.$$

Notation The following notation is used throughout the paper:

- x_i, y_i, z_i: The 16-byte state after the X, S, L operation, respectively, at round i.
- $x_i[j]$: The j^{th} byte of the state x_i, where $j = 0, 1, \cdots, 15$, and the bytes are indexed from left to right.
- Δx_i, $\Delta x_i[j]$: The difference at state x_i, and byte $x_i[j]$, respectively.
- (C, C'): A pair of ciphertexts where C denotes the original ciphertext and C' denotes the faulty one.

- $P[j]$: The j^{th} byte of the plaintext.

3 A Differential Fault Analysis Attack on Kuznyechik

In this attack, we adopt a random byte fault model where the attacker is assumed to be able to fault a random byte in the states before the linear transformation in round eight to a random value. As depicted in Figure 3, a successful fault injection occurs in either x_8 or y_8. The exact position of the fault cannot be determined from the observed ciphertexts due to the optimal diffusion properties of L and it has no added value to the steps of the attack. The attack starts by building a table for all the

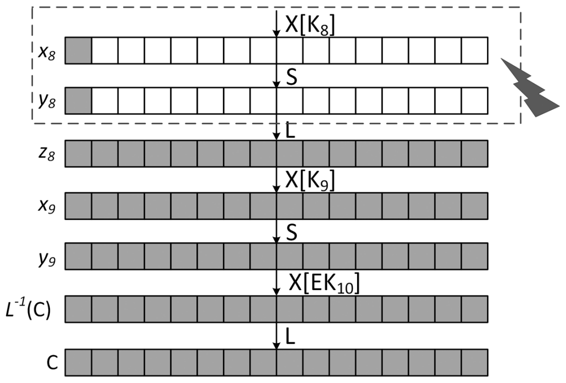

Figure 3: Fault injection in round eight

possible 16×255 differences in z_8 which result from propagating a random error at any of the 16 byte positions in either x_8 or y_8 through the linear transformation layer. Then using the observed ciphertext pair (C, C'), one guesses the last round key, and evaluates the difference at x_9. The evaluated difference is then checked against the differences in the stored table. Successful key candidates result in a match and are consequently stored in another table for further filtration with another (C, C') pair. Let n denote the number of bytes in the state, then the expected number of remaining candidate keys after trying N ciphertext pairs (C, C') is given by $256^n(n \times 255^{1-n})^N = 256^{16}(16 \times 255^{-15})^N$ [16]. Accordingly, two ciphertext pairs are required to retrieve the last round key. A naive implantation of this attack requires guessing the 128-bits of the last round key when testing the first ciphertext pair which renders its complexity unpractical. However, if the last round does not contain a linear transformation, one can guess independent key bytes which reduces the complexity as in the case AES and Khazad [16]. Kuznyechik employs a linear transformation in its last round. Accordingly, as depicted in Figure 3, we adopt an equivalent representation by which we evaluate an equivalent key, $EK_{10} = L^{-1}(K_{10})$ and swap the order of the linear transformation and the key mixing operations. Hence, we can apply a two fault practical attack and recover the master key. In what follows, we give the steps of the attack.

1. Store in a table T all the possible 16×255 differences in z_8.
2. Consider two ciphertext pairs (C_1, C'_1), and (C_2, C'_2), for each ciphertext pair, compute $EC_i = L^{-1}(C_i)$, and $EC'_i = L^{-1}(C'_i)$, for $i = 1, 2$.
3. For each value of the possible 2^{16} values of $EK_{10}[0]||EK_{10}[1]$, compute

   $$S^{-1}(X[EK_{10}[0]||EK_{10}[1]](EC_1[0]||EC_1[1])) \oplus S^{-1}(X[EK_{10}[0]||EK_{10}[1]](EC'_1[0]||EC'_1[1])),$$

   $$S^{-1}(X[EK_{10}[0]||EK_{10}[1]](EC_2[0]||EC_2[1])) \oplus S^{-1}(X[EK_{10}[0]||EK_{10}[1]](EC'_2[0]||EC'_2[1])).$$

   Match the resulting two differences from both ciphertext pairs with the two left most bytes of the differences in T. If a match occurs, add $EK_{10}[0]||EK_{10}[1]$ to another table T_k.
4. For each $EK_{10}[0]||EK_{10}[1]$ in table T_k:
   - Remove $EK_{10}[0]||EK_{10}[1]$ from T_k and extend it by one byte $EK_{10}[2]$.
   - For all the 2^8 values of $EK_{10}[2]$, compute

     $$S^{-1}(X[EK_{10}[1]||EK_{10}[2]](EC_1[1]||EC_1[2])) \oplus S^{-1}(X[EK_{10}[1]||EK_{10}[2]](EC'_1[1]||EC'_1[2])),$$

     $$S^{-1}(X[EK_{10}[1]||EK_{10}[2]](EC_2[1]||EC_2[2])) \oplus S^{-1}(X[EK_{10}[1]||EK_{10}[2]](EC'_2[1]||EC'_2[2])).$$

   - Match the resulting two differences from both ciphertext pairs with the second and third bytes of the differences in T. If a match occurs, add $EK_{10}[0]||EK_{10}[1]||$ $EK_{10}[2]$ to T_k.
5. Repeat step 4 until the length of the candidate keys in T_k is 16 bytes.

6. Using the two equivalent ciphertext pairs, exhaustively verify which of the remaining keys in T_k produces a difference that matches any of the ones in the precomputed table T.

We have simulated the procedure for the last round key recovery using 100 randomly generated keys. The use of two faults resulted in an avarage of 462.86 remaining candidates in T_k after step 5 of the above procedure. Then the remaining keys were exhastively tested using the same original-faulty ciphertext pairs to recover K_{10}. The attack requires two faults injected in either x_8 or y_8 to recover K_{10}. Afterwards, using the knowledge of K_{10}, one can peel off the last round and repeat the attack by injecting an additional two faults in either x_7 or y_7 to recover K_9. Finally, the knowledge of K_9 and K_{10} allows us to invert the key schedule and compute the master key.

4 Ineffective Fault Analysis Attack on Kuznyechik with a Secret Sbox

In this analysis, we consider the case when Kuznyechik is deployed with a secret sbox. Additionally, we assume that the same sbox is used in the key schedule operation. A similar setting is employed with the current standard GOST 28147-89 and it is expected that users will be allowed to use secret sboxes with Kuznyechik as well. Utilizing secret parameters is assumed to increase the security margin of the employed primitive and makes it harder to cryptanalyze. For that reason, customized primitives with secret parameters are used in military products, gaming systems, and pay TV.

Our attack applies an ineffective fault analysis using stuck-at-zero faults on Kuznyechik. The adopted fault model assumes that the attacker is able to reset a given state byte to zero, hence the attacker can verify if the original byte is zero or not by checking if both the original and faulty ciphertexts are equal. In other words, a successful fault injection takes place when the observed genuine and faulty ciphertexts are similar (i.e., $C = C'$).

Our attack recovers the master key and the sbox secret parameters in four stages. The first stage recovers the value of K_1 relative to the value of $S^{-1}(0)$. Afterwards, in the second stages, we form a system of equations and derive the values of 2^{16} candidates for the right sbox corresponding to all the possible values of $S^{-1}(0)||K_2[0]$. In the sequel, using the 2^{16} candidate sboxes, we recover the values of 2^{16} candidates for K_2 in the third stage . Finally, in the fourth step, we filter the 2^{16} candidate master keys and their corresponding sboxes using a known plaintext-ciphertext pair. In what follows we give the details of our four stage approach.

Recovery of $K_1 \oplus S^{-1}(0)$: This stage recovers the value of the i^{th} byte on $K_1[i]$ up to the constant value $S^1(0)$, for $i = 0, 1, \cdots, 15$, as follows:

1. Iteratively exhaust $P[i]$ and fault byte $y_1[i]$ until an ineffective fault is observed. The occurrence of the IF indicates that the original value of $y_1[i] = 0$.
2. Accordingly, the value of $K_1[i] \oplus S^{-1}(0) = P[i]$.

Applying the above two steps for all the values of i, we recover all of the bytes of K_1 up to the constant $S^{-1}(0)$. This step requires about $2^8 \times 16 = 2^{12}$ fault injections.

Retrieving 2^{16} candidates for the sbox: In this stage, we iteratively assumes all the possible values $S^{-1}(0)$ and $K_2[0]$ to derive the values of the secret parameters of 2^{16} candidate sboxes. The procedure for each $S^{-1}(0)$ and $K_2[0]$ guess is described as follows:

1. Let $a = S^{-1}(0)$ and hence $S(a) = 0$.
2. Evaluate the candidate value of K_1 by Xoring the value of $K_1[i] \oplus S^{-1}(0)$ recovered in the first stage of the attack, by the guessed value of $S^{-1}(0)$.
3. Repeat the following steps 2^8 times.
   (a) Let $P[0] = m_0 + K_1[0]$, $P[1] = a + K_1[1]$, and $P[i] = K_1[i] + S^{-1}(0)$, for $i = 2, 3, \cdots, 15$.

(b) Iteratively exhaust m_0 and fault byte $y_2[0]$ until an ineffective fault is observed. The occurrence of the IF indicates that the original value of $y_2[0] = 0$.

(c) Accordingly, after applying the $X[K_1]$, S, L, and $X[K_2]$ transformations, the value of the first byte of x_2 is given by $\alpha_{0,0}S(m_0) + \alpha_{1,0}S(a) + K_2[0]$.

(d) Due to the IF which resulted by the current choice of m_0, we know that the value of $x_2[0] = S^{-1}(0)$. Accordingly, one can compute the value of $S(m_0)$ by

$$S(m_0) = [K_2[0] + S^{-1}(0) + \alpha_{1,0}S(a)]\alpha_{0,0}^{-1}$$

(e) Set $a = m_0$, and go to step 3 to find another m_0.

The obtained equations from the above procedure correspond to a linear system of equations over $GF(2^8)$ where the unknowns are the sbox entries. According to our experimental results, the system obtained using $P[0]$ and $P[1]$ (or any other pairs) is not a full rank and exhaustively enumerating all its possible solutions is computationally prohibitive. However, because of its structure, we are able to evaluate the values of the sbox entries that are uniquely determined by these equations using the above iterative procedure. When we use three pairs, the system was full rank for 99 out of 100 experiments. Consequently, to recover all the 256 entries of the candidate sbox, the above algorithm is repeated with three different plaintext byte positions. More precisely, after exhausting $P[0]$ and iteratively setting $P[1]$ to the recovered value of a for 2^8 times, we repeat the exact procedure with $P[0]$ and $P[2]$, and finally with $P[1]$ and $P[2]$. However, in the latter two cases, we start the procedure from step 3, so we use the last recovered value of a and not the first point of entry as the first procedure. All in all, for the 2^{16} candidate sboxes, the attack requires $3 \times 2^{16} \approx 2^{17}$ faults and a time complexity of $2^{16}(3 \times 2^{16}) \approx 2^{33}$.

Recovering the rest of K_2: The previous two stages resulted in 2^{16} candidate sboxes with their corresponding candidate K_1 and $K_2[0]$ values. Accordingly, in this stage, for each sbox out of the retrieved 2^{16} candidates, we recover the remaining fifteen bytes of K_2 as follows: for each byte i, for $i = 1, 2, \cdots, 15$.

1. Let $P[0] = m_0 + K_1[0]$, and $P[i] = K_1[i] + S^{-1}(0)$.
2. Iteratively exhaust m_0 and fault byte $y_2[i]$ until an ineffective fault (IF) is observed. The occurrence of the IF indicates that the original value of $y_2[i] = 0$.
3. Evaluate $K_2[i] = \alpha_{0,i} S(m_0) + S^{-1}(0)$.

This stage requires $2^8 \times 15 \approx 2^{12}$ fault injections and a time complexity of $2^{16} \times 2^8 \times 15 \approx 2^{28}$ to recover the 2^{16} candidate for the remaining fifteen bytes of K_2.

Recovering the master key: In this stage, we test the 2^{16} candidate sets parameters, where each set consists of an sbox and its corresponding master key $K_1||K_2$ against a known plaintext-ciphertext pair. More precisely, using a candidate sbox and its corresponding $K_1||K_2$, one can encrypt a given plaintext and compare the computed ciphertext to the one generated by the attacked device. This stage requires a time complexity of 2^{16}.

Our four stage approach has a practical complexity which was verified by our simulation where the retrieval of the secret parameters of the sbox and corresponding keys require about $2^{12} + 2^{17} + 2^{12} \approx 2^{17}$ ineffective faults and a time complexity of $2^{12} + 2^{33} + 2^{28} + 2^{16} \approx 2^{33}$. This complexity is justified considering that the use of a secret 8-bit sbox and a 256-bit key increases the size of the secret information to about 1940 bit. Indeed, the security level of Kunyechik in this setting is expected to be very high and consequently the practicality of our attack proves its worthiness.

5 Conclusion

In this paper, we have presented fault analysis attacks on the new draft of the Russian encryption standard, Kuznyechik, in two different settings.

Our first attack is a differential fault analysis attack that utilizes the random byte fault model. In this attack, we employ an equivalent representation of the last round by which we bypass the effect of the optimal diffusion of the last linear transformation. This tweak enables an efficient and practical recovery of the master key using four faults. Our second attack is a four stage ineffective fault analysis of Kuznyechik when employing secret sbox. We first recover several sets of candidate secret sbox parameters and their corresponding master key. In the sequel, we filter these sets by testing them against a known plaintext-ciphertext pair to recover the right secret sbox and master key.

Our attack works when we assume that the same secret sbox is used in the key schedule operation. It is interesting to investigate how this approach can be extended in different cases where different sboxes are employed for each byte position, or when the sbox used in the key schedule is different than the one used in the encryption process.

While these attacks may not present direct threat to the theoretical security of Kuznyechik, they serve as a cautionary example to demonstrate the importance of protecting different implementations of the new standard, even if its sbox is not publicly known.

References

[1] GOST 28147-89. Information Processing Systems. Cryptographic Protection. Cryptographic Transformation Algorithm. *(In Russian).*

[2] The National Standard of the Russian Federation GOST R 34._-20_. Russian Federal Agency on Technical Regulation and Metrology report, 2015. `http://www.tc26.ru/en/standard/draft/ENG_GOST_R_bsh.pdf`.

[3] AlTawy, R., and Youssef, A. M. Differential fault analysis of Streebog. In *ISPEC* (2015), J. Lopez and Y. Wu, Eds., vol. 9065 of *Lecture Notes in Computer Science*, Springer, pp. 35–49.

[4] AlTawy, R., and Youssef, A. M. Meet in the middle attacks on reduced round Kuznyechik. Cryptology ePrint Archive, Report 2015/096, 2015. `http://eprint.iacr.org/`.

[5] Biham, E., and Shamir, A. Differential fault analysis of secret key cryptosystems. In *CRYPTO* (1997), J. Kaliski, BurtonS., Ed., vol. 1294 of *Lecture Notes in Computer Science*, Springer, pp. 513–525.

[6] Boneh, D., DeMillo, R., and Lipton, R. On the importance of checking cryptographic protocols for faults. In *EUROCRYPT* (1997), W. Fumy, Ed., vol. 1233 of *Lecture Notes in Computer Science*, Springer, pp. 37–51.

[7] Clavier, C., Gierlichs, B., and Verbauwhede, I. Fault analysis study of IDEA. In *CT-RSA* (2008), T. Malkin, Ed., vol. 4964 of *Lecture Notes in Computer Science*, Springer, pp. 274–287.

[8] Clavier, C., and Wurcker, A. Reverse engineering of a secret AES-like cipher by ineffective fault analysis. In *IEEE workshop on Fault Diagnosis and Tolerance in Cryptography* (2013), pp. 119–128.

[9] Courbon, F., Loubet-Moundi, P., Fournier, J. J., and Tria, A. Adjusting laser injections for fully controlled faults. In *Constructive Side-Channel Analysis and Secure Design* (2014), E. Prouff, Ed., Lecture Notes in Computer Science, Springer, pp. 229–242.

[10] Dinur, I., Dunkelman, O., and Shamir, A. Improved attacks on full GOST. In *FSE* (2012), A. Canteaut, Ed., vol. 7549 of *Lecture Notes in Computer Science*, Springer, pp. 9–28.

[11] Giraud, C. DFA on AES. In *AES* (2005), H. Dobbertin, V. Rijmen, and A. Sowa, Eds., vol. 3373 of *Lecture Notes in Computer Science*, Springer, pp. 27–41.

[12] Isobe, T. A single-key attack on the full GOST block cipher. In *FSE* (2011), A. Joux, Ed., vol. 6733 of *Lecture Notes in Computer Science*, Springer, pp. 290–305.

[13] Kim, C., and Quisquater, J.-J. New differential fault analysis on AES key schedule: Two faults are enough. In *CARDIS* (2008), G. Grimaud and F.-X. Standaert, Eds., vol. 5189 of *Lecture Notes in Computer Science*, Springer, pp. 48–60.

[14] Kircanski, A., and Youssef, A. M. Differential fault analysis of Rabbit. In *SAC* (2009), M. J. Jacobson, V. Rijmen, and R. Safavi-Naini, Eds., vol. 5867 of *Lecture Notes in Computer Science*, Springer, pp. 197–214.

[15] Moro, N., Dehbaoui, A., Heydemann, K., Robisson, B., and Encrenaz, E. Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller. In *IEEE workshop on Fault Diagnosis and Tolerance in Cryptography* (2013), pp. 77–88.

[16] Piret, G., and Quisquater, J.-J. A differential fault attack technique against SPN structures, with application to the AES and Khazad. In *CHES* (2003), C. Walter, C. Koç, and C. Paar, Eds., vol. 2779 of *Lecture Notes in Computer Science*, Springer, pp. 77–88.

[17] Poschmann, A., Ling, S., and Wang, H. 256 bit standardized crypto for 650 GE GOST revisited. In *CHES* (2010), S. Mangard and F.-X. Standaert, Eds., vol. 6225 of *Lecture Notes in Computer Science*, Springer, pp. 219–233.

[18] Roscian, C., Sarafianos, A., Dutertre, J.-M., and Tria, A. Fault model analysis of laser-induced faults in SRAM memory cells. In *IEEE workshop on Fault Diagnosis and Tolerance in Cryptography* (2013), pp. 89–98.

[19] Shishkin, V., Dygin, D., Lavrikov, I., Marshalko, G., Rudskoy, V., and Trifonov, D. Low-Weight and Hi-End: Draft Russian Encryption Standard. In *CTCrypt* (2014), pp. 183–188.

[20] Skorobogatov, S., and Anderson, R. Optical fault induction attacks. In *CHES* (2003), B. Kaliski, C. Koç, and C. Paar, Eds., vol. 2523 of *Lecture Notes in Computer Science*, Springer, pp. 2–12.

[21] Tunstall, M., Mukhopadhyay, D., and Ali, S. Differential fault analysis of the Advanced Encryption Standard using a single fault. In *Information Security Theory and Practice* (2011), C. Ardagna and J. Zhou, Eds., vol. 6633 of *Lecture Notes in Computer Science*, Springer, pp. 224–233.

[22] Zhao, X., Guo, S., Zhang, F., Wang, T., Shi, Z., and Gu, D. Algebraic fault analysis on GOST for key recovery and reverse engineering. In *IEEE workshop on Fault Diagnosis and Tolerance in Cryptography* (2014), pp. 29–39.

An Attack on 6 Rounds of Khazad

Dmitry Burov Boris Pogorelov

Abstract

This paper reports new attacks on the 64-bit block cipher Khazad. These attacks use some structural properties of round function. As a result we find 14 new classes of weak keys for 5 and 6 rounds of Khazad. Particularly we show that Khazad has 7 classes of weak keys for 5 and 6 rounds. The cardinality of each class is 2^{64}. The time complexity of weak keys recovering is 2^{35} S-box lookups for 5 rounds and 2^{43} S-box lookups for 6 rounds and the data complexity is 2^{32} chosen plaintexts. We also show that Khazad has 7 classes of weak keys for 5 and 6 rounds. The cardinality of each class is 2^{64}. The time complexity of weak key recovering is 2^{35} S-box lookups for 5 rounds and 2^{43} S-box lookups for 6 rounds and the data complexity is 2^{32} chosen ciphertexts.

Keywords: block cipher, Khazad, invariant subspaces, reducible linear transformation.

1 Description of Khazad

Khazad is an 8-round SP-network with 64 bit block length and 128 bit key length [1]. It was proposed by V. Rijmen and P. Barreto specially for the NESSIE project and was chosen as its finalist.

We represent the field $GF\left(2^8\right)$ as $GF\left(2\right)\left[x\right] / \, p(x)$, where $p\left(x\right) = x^8 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$ – a primitive polynomial over $GF(2)$. A polynomial $a\left(x\right) = a_0 \oplus a_1 x \oplus \ldots \oplus a_7 x^7 \in GF\left(2\right)\left[x\right]$, where $a_i \in GF\left(2\right)$ for all $i \in \{0, \ldots, 7\}$, is denoted by a numerical value $\sum_{i=0}^{7} a_i 2^i$, and written in decimal notation. Further a 64-bit cipher state is represented as a vector in $V_8\left(2^8\right)$.

Round function f_k consists of three transformations: $\tilde{s}$, h, v_k.

Function $\tilde{s} : V_8\left(2^8\right) \to V_8\left(2^8\right)$ consists of a parallel of application of a nonlinear involution substitution box $s : GF\left(2^8\right) \to GF(2^8)$:

$$\tilde{s} : \left(a_0, \ldots a_7\right) \mapsto \left(a_0^s, \ldots, a_7^s\right),$$

where $a_i \in GF\left(2^8\right)$ for all $i \in \{0,\dots,7\}$.

The diffusion layer is presented as a multiplication by matrix $h = |||h_{i,j}||| \in GL_8(2^8)$:

$$h = \begin{pmatrix} 1 & 3 & 4 & 5 & 6 & 8 & 11 & 7 \\ 3 & 1 & 5 & 4 & 8 & 6 & 7 & 11 \\ 4 & 5 & 1 & 3 & 11 & 7 & 6 & 8 \\ 5 & 4 & 3 & 1 & 7 & 11 & 8 & 6 \\ 6 & 8 & 11 & 7 & 1 & 3 & 4 & 5 \\ 8 & 6 & 7 & 11 & 3 & 1 & 5 & 4 \\ 11 & 7 & 6 & 8 & 4 & 5 & 1 & 3 \\ 7 & 11 & 8 & 6 & 5 & 4 & 3 & 1 \end{pmatrix}.$$

The linear transformation h is an involution.

The key addition $v_k : V_8\left(2^8\right) \to V_8\left(2^8\right)$ is a bitwise addition of a key vector $k \in V_8\left(2^8\right)$:

$$v_k : \alpha \mapsto \alpha \oplus k.$$

The key schedule expands the cipher key $k = (k_{-2},\ k_{-1}) \in V_{16}\left(2^8\right)$ into a sequence of round keys $k_0,\dots,k_8$, where $k_i \in V_8\left(2^8\right)$ for all $i \in \{0,\dots,8\}$. The sequence of round keys is evaluated by means of a Feistel iteration:

$$k_i = (k_{i-1})^{\tilde{s}h} \oplus c_i \oplus k_{i-2},$$

where $c_i = (c_{i,0},\dots,c_{i,7})$ defined as

$$c_{i,j} = (8i+j)^s,\ i \in \{0,\dots,8\},\ j \in \{0,\dots,7\}.$$

A round function is given as

$$f_k : \alpha \mapsto \alpha^{v_k \tilde{s} h}.$$

The full encryption function for r iteration is defined as

$$g_{k_0,\dots,k_r} = f_{k_0} \cdot \ldots \cdot f_{k_{r-2}} v_{k_{r-1}} \tilde{s} v_{k_r}.$$

2 Previous results on Khazad

Several attacks where applied on Khazad in a single-key model. There is an integral attack on 4 rounds of Khazad [1]. The attack requires 2^9 chosen plaintexts and has time complexity 2^{91}. In [2] Biryukov finds a class of 2^{64} weak keys for which Khazad can be broken with 2^{43} S-box lookups using 2^{38} chosen plaintexts. An attack on 5 rounds of Khazad was proposed by Muller [5]. This attack requires 2^{64} known plaintexts and have the time complexity 2^{91}. In [10] attack on 6 rounds of Khazad is presented. The data complexity is 2^{64} chosen plaintexts and the time complexity is not presented.

In [3] attack on 7 rounds of Khazad in the related-key model and attack on 8 rounds in the chosen-key model are presented.

Note that single-key attacks are more powerful than related-key and chosen-key attacks.

3 Structural properties of Khazad round function

In this section we investigate some properties of Khazad round function. These properties are caused by reducibility of the linear transformation h. Note that the transformation h is an optimal diffusion transformation. Invariant subspaces under $\tilde{s}$ and h are described in proposition 1.

Let G be a group generated by the linear transformation h and the group $\{v_\alpha | \alpha \in V_{64}\}$. The properties of the group G must influence on choice of S-box. Otherwise a round function may be approximated by imprimitive transformations [7], [8] or isometric transformations [6], [9]. Also a round function may save any subspaces [4]. This fact is used to attack Khazad.

Proposition 1. *The following subspaces are invariant under $\tilde{s}$ and h:*

1. $W^{(1)} = \left\{ (a, a, b, b, e, e, d, d) \, | a, b, e, d \in GF\left(2^8\right) \right\}$;
2. $W^{(2)} = \left\{ (a, b, a, b, e, d, e, d) \, | a, b, e, d \in GF\left(2^8\right) \right\}$;
3. $W^{(3)} = \left\{ (a, b, b, a, e, d, d, e) \, | a, b, e, d \in GF\left(2^8\right) \right\}$;
4. $W^{(4)} = \left\{ (a, b, e, d, a, b, e, d) \, | a, b, e, d \in GF\left(2^8\right) \right\}$;

5. $W^{(5)} = \{(a, b, e, d, b, a, d, e) \,|a, b, e, d \in GF(2^8)\}$;
6. $W^{(6)} = \{(a, b, e, d, e, d, a, b) \,|a, b, e, d \in GF(2^8)\}$;
7. $W^{(7)} = \{(a, b, e, d, d, e, b, a) \,|a, b, e, d \in GF(2^8)\}$;
8. $U^{(1)} = \{(a, a, a, a, b, b, b, b) \,|a, b \in GF(2^8)\}$;
9. $U^{(2)} = \{(a, a, b, b, a, a, b, b) \,|a, b \in GF(2^8)\}$;
10. $U^{(3)} = \{(a, a, b, b, b, b, a, a) \,|a, b \in GF(2^8)\}$;
11. $U^{(4)} = \{(a, b, a, b, a, b, a, b) \,|a, b \in GF(2^8)\}$;
12. $U^{(5)} = \{(a, b, a, b, b, a, b, a) \,|a, b \in GF(2^8)\}$;
13. $U^{(6)} = \{(a, b, b, a, a, b, b, a) \,|a, b \in GF(2^8)\}$;
14. $U^{(7)} = \{(a, b, b, a, b, a, a, b) \,|a, b \in GF(2^8)\}$;
15. $Z^{(1)} = \{(a, a, a, a, a, a, a, a) \,|a \in GF(2^8)\}$.

Proof. The proof is straightforward. $\square$

Further we assume $W \in \{W^{(i)}, U^{(i)}, Z^{(1)} | i \in \{1, \ldots, 7\}\}$. The existence of $\tilde{sh}$-invariant subspaces may be a potentially weakness of a block cipher. Only transformation v_k doesn't save this invariance. However $W^{v_k} = W$, if $k \in W$.

4 An attack on 5 rounds of Khazad

In this section we describe new weak keys classes for 5 rounds.

From the key schedule it follows that there are exactly $|W|^2$ encryption keys such that round keys k_1, k_2 belong to subspace W. We have $k_0 \in W \oplus c_2$, $k_3 \in W \oplus c_3$ because $k_2 = k_1^{\tilde{sh} v_{c_2}} \oplus k_0 \in W$, $k_1, k_2 \in W$ and $W^{\tilde{sh}} = W$. Hence,

$$(W \oplus c_2)^{f_{k_0} f_{k_1} f_{k_2} v_{k_3}} = W \oplus c_3.$$

For any set $X \subset V_8(2^8)$ and for each $i \in \{0, \ldots, 7\}$ we define a multiset

$$X_i = \{a \in GF(2^8) | (x_0, \ldots, x_{i-1}, a, x_{i+1}, \ldots, x_7) \in X\}.$$

In this context a multiset is a set where a value is allowed to appear several times. For each multiset X_i, $i \in \{0,\dots,7\}$, we define vectors $\mu(X_i) = \left(\mu_a(X_i) \,|\, a \in GF\left(2^8\right)\right)$ and $\nu(X_i) = (n_0(X_i), n_1(X_i), \dots, n_{2^8}(X_i))$, where

$$\mu_a(X_i) = \left|\{x \in X_i |\; x = a\}\right|,$$
$$n_j(X_i) = \left|\left\{a \in GF\left(2^8\right) \mid \mu_a(X_i) = j\right\}\right|.$$

Suppose

$$A{=}(W \oplus c_3)^{\tilde{s}h} = \{\alpha_i = (a_{i,0},\dots,a_{i,7}) \;\mid\; i \in \{0,\dots,|W|-1\}\}.$$

It is evident that a vector $\mu(A_i)$ is equal accurate within permutation to a vector $\mu\left(\left(A^{v_{k_4}\tilde{s}v_{k_5}}\right)_i\right)$ for all $i \in \{0,\dots,7\}$, k_4, $k_5 \in V_8\left(2^8\right)$. Using computer calculations we find that $\mu_a(A_i) \neq \mu_b(A_i)$ for all $i \in \{0,\dots,7\}$, $a, b \in GF\left(2^8\right)$ such that $a \neq b$. Hence we have that $\mu_x(A_i) = \mu_a(B_i)$ if and only if $(x \oplus k_{4,i})^s \oplus k_{5,i} = a$.

Algorithm 1.

Input: vectors $\mu(A_i)$ for all $i \in \{0,\dots,7\}$, set $B{=}\{\alpha^{g_{k_0,\dots,k_5}} | \alpha \in W \oplus c_2\}$.

Output: keys k_4, k_5.

Step 1. For all $i \in \{0,\dots,7\}$ do next steps.

Step 2. Choose arbitrary $a, b \in GF\left(2^8\right)$ such that $a \neq b$.

Step 3. Find $x, y \in GF(2^8)$ such that $\mu_a(B_i){=}\mu_x(A_i)$, $\mu_b(B_i){=}\mu_y(A_i)$.

Step 4. Find $k_{4,i}$, $k_{5,i} \in GF\left(2^8\right)$ by solving the following system of equations

$$\begin{cases} (x \oplus k_{4,i})^s \oplus k_{5,i}{=}a, \\ (y \oplus k_{4,i})^s \oplus k_{5,i}{=}b. \end{cases} \tag{1}$$

If system (1) has more than one solution, then we can choose $c \in GF\left(2^8\right)$, $c \notin \{a, b\}$, and z such that $\mu_c(B_i) = \mu_z(A_i)$. Further we can add a new equation to system (1) for discarding wrong keys.

The time complexity of this method equals to the time complexity of computing vectors $\mu(A_i)$ for all $i \in \{0,\dots,7\}$. This time complexity is $8 \cdot |W|$ S-box lookups. The data complexity is $|W|$ chosen plaintexts. Hence,

1. if $W \in \left\{W^{(i)} | i \in \{1,\dots,7\}\right\}$, then the time complexity is 2^{35} S-box lookups, the data complexity is 2^{32} chosen plaintext and we have 2^{64} weak keys;

2. if $W \in \left\{U^{(i)} | i \in \{1, \dots 7\}\right\}$, then the time complexity is 2^{19} S-box lookups, the data complexity is 2^{16} chosen plaintexts and we have 2^{32} weak keys;

3. if $W = Z^{(1)}$, then the time complexity is 2^{11} S-box lookups, the data complexity is 2^8 chosen plaintexts and we have 2^{16} weak keys.

If $W \in \left\{W^{(i)} | i \in \{1, \dots, 7\}\right\}$, then the time complexity and the data complexity of this method is less than the time complexity and the data complexity of the method represented in [2]. Moreover we have 7 weak keys classes opposed against 1 weak keys class in [2].

5 Membership tests for weak keys

In this section we present membership tests for weak keys of section 4. Membership tests are attacks designed not to recover the unknown key, but to determine if the key is a member of a set of weak keys. Notice that there is no membership test for weak keys in [2].

In this section we suppose that $W \in \left\{W^{(1)},\ U^{(1)}\right\}$ for simplicity. For other subspaces from proposition 1 all results are true accurate within indices.

Proposition 2. *Assume $i \in \{0, 1, 2, 3\}$. If $W = W^{(1)}$, then*

$$A_{2i} = A_{2i+1}. \tag{2}$$

If $W = U^{(1)}$, then

$$A_0 = A_1 = A_2 = A_3, \tag{3}$$

$$A_4 = A_5 = A_6 = A_7. \tag{4}$$

Proof. Let φ, φ' be transformations from $V_4(2^8)$ to $GF(2^8)$. By definition, put $\varphi : (a, b, e, d) \mapsto x$, $\varphi' : (a, b, e, d) \mapsto x'$, where

$$x = (a \oplus c_{3,0})^s h_{0,2i} \oplus (a \oplus c_{3,1})^s h_{1,2i} \oplus (b \oplus c_{3,2})^s h_{2,2i} \oplus (b \oplus c_{3,3})^s h_{3,2i} \oplus$$
$$\oplus (e \oplus c_{3,4})^s h_{4,2i} \oplus (e \oplus c_{3,5})^s h_{5,2i} \oplus (d \oplus c_{3,6})^s h_{6,2i} \oplus (d \oplus c_{3,7})^s h_{7,2i},$$

$$x' = (a \oplus c_{3,0})^s h_{0,2i+1} \oplus (a \oplus c_{3,1})^s h_{1,2i+1} \oplus (b \oplus c_{3,2})^s h_{2,2i+1} \oplus$$
$$\oplus (b \oplus c_{3,3})^s h_{3,2i+1} \oplus (e \oplus c_{3,4})^s h_{4,2i+1} \oplus (e \oplus c_{3,5})^s h_{5,2i+1} \oplus$$
$$\oplus (d \oplus c_{3,6})^s h_{6,2i+1} \oplus (d \oplus c_{3,7})^s h_{7,2i+1}.$$

From the definitions of A_{2i} and A_{2i+1} it follows that $(V_4(2^8))^{\varphi} = A_{2i}$, $(V_4(2^8))^{\varphi'} = A_{2i+1}$. Note that for all $j \in \{0,1,2,3\}$ we have $h_{2j,2i} = h_{2j+1,2i+1}$. Hence for all vectors $(a,b,e,d) \in V_4(2^8)$ we have

$$(a,b,e,d)^{\varphi} = (a \oplus c_{3,0} \oplus c_{3,1}, b \oplus c_{3,2} \oplus c_{3,3}, e \oplus c_{3,4} \oplus c_{3,4}, d \oplus c_{3,6} \oplus c_{3,7})^{\varphi'}.$$

Therefore we prove equality (2). Equalities (3) and (4) can be proved similarly. □

From proposition 2 the next corollary is followed.

Corollary 3. *For all $i \in \{0,1,2,3\}$, k_4, $k_5 \in V_8(2^8)$ we have*

$$\nu\left(\left(A^{v_{k_4}\tilde{s}v_{k_5}}\right)_{2i}\right) = \nu\left(\left(A^{v_{k_4}\tilde{s}v_{k_5}}\right)_{2i+1}\right).$$

Based on corollary 3 we present the following algorithm for identification of weak keys.

Algorithm 2.

Input: set B=$\{\alpha^{g_{k_0,\dots,k_5}} | \alpha \in W^{(1)} \oplus c_2\}$.

Output: unconclusive (key is weak with great probability) or no (key is not weak).

Step 1. For all $i \in \{0,1,2,3\}$ check

$$\nu(B_{2i}) = \nu(B_{2i+1}). \qquad (5)$$

Step 2. If for some $i \in \{0,1,2,3\}$ (5) is not satisfied, then output: «key is not weak», otherwise output «unconclusive».

The time complexity of algorithm 2 is 2^{35} S-box lookups. If algorithm 2 gives us an unconclusive answer, then the key is weak with great probability.

Vectors $\nu(A_i)$ may be computed for Khazad easily. Therefore algorithm 2 may be improved.

Algorithm 3.

Input: set B=$\{\alpha^{g_{k_0,\dots,k_5}} | \alpha \in W \oplus c_2\}$, vectors $\nu(A_i)$ for all $i \in \{0,\dots,7\}$.

Output: unconclusive (key is weak with great probability) or no (key is not weak)..

Step 1. For all $i \in \{0,\dots,7\}$ compute vectors $\nu(B_i)$.

Step 2. If there is $i \in \{0, \ldots, 7\}$ such that $\nu(B_i) \neq \nu(A_i)$, then output: «key is not weak», otherwise output «unconclusive».

The time complexity of algorithm 3 is 2^{35} S-box lookups. If algorithm 3 gives us an unconclusive answer so the key is weak with probability greater than such in algorithm 2.

6 An attack on 6 rounds of Khazad

In this section we describe an attack on 6 rounds of Khazad. This attack is similar to the attack of section 4.

The attack uses the property of the set $A{=}(W \oplus c_3)^{\tilde{s}h}$.

The next result may be obtained by computer computation.

Proposition 4. *Assume that* $W \in \{W^{(i)} | i \in \{1, \ldots, 7\}\}$. *For all* $a, b \in GF(2^8)$, $j \in \{0, \ldots, 7\}$ *we have*

$$\mu_a(A_j) \equiv \mu_b(A_j) \mod 2. \tag{6}$$

Note that proposition 4 is not true for $W \in \{U^{(i)},\ Z^{(1)} | i \in \{1, \ldots, 7\}\}$.

Suppose

$$C{=}A^{v_{k_4}\tilde{s}hv_{k_5}}{=}\{\gamma_i{=}(x_{i,0}, x_{i,1}, \ldots, x_{i,7}) | i \in \{0, \ldots, 2^{32}{-}1\}\}.$$

Proposition 5. *For all* $j \in \{0, \ldots, 2^{32} - 1\}$ *we have*

$$\sum_{i=0}^{2^{32}-1} x_{i,j}{=}0, \tag{7}$$

Proof. It is clear that equality (7) doesn't depend on k_5.

Suppose

$$Y = A^{v_{k_4}\tilde{s}} = \{(y_{i,0}, y_{i,1}, \ldots, y_{i,7}) | i \in \{0, \ldots, 2^{32} - 1\}\}.$$

The sum (7) can be represented in the form

$$\sum_{j=0}^{2^{32}-1} x_{i,j} = \sum_{i=0}^{2^{32}-1} \sum_{t=0}^{7} y_{i,t} h_{t,j} = \sum_{t=0}^{7} h_{t,j} \sum_{i=0}^{2^{32}-1} y_{i,t}.$$

From proposition 4 it follows that the sum $\sum_{i=0}^{2^{32}-1} y_{i,t}$ is equal to 0 for all $t \in \{0, \ldots, 7\}$. □

Suppose

$$D=\{\alpha^{g_{k_0,\ldots,k_6}}|\alpha \in W \oplus c_2\} = \{\delta_i=(d_{i,7},\ldots,d_{i,0}),\ i \in \{0,\ldots,2^{32}-1\}\}.$$

Using proposition 5 we present the following algorithm for recovering key k_6 for 6 rounds of Khazad.

Algorithm 4.
Input: the set D.
Output: the set of possible keys k_6 containing the right key.
Step 1. For all $i \in \{0,\ldots,7\}$ do the following steps.
Step 2. For all $k_{6,i} \in GF(2^8)$ check

$$\sum_{j=0}^{2^{32}-1} (d_{j,i} \oplus k_{6,i})^{s^{-1}} = 0. \tag{8}$$

Step 3. If (8) is not satisfied, then $k_{6,i}$ is wrong, otherwise $k_{6,i}$ is a possible variant for the right key byte.

Algorithm 4 doesn't discard 2^8 keys on average. The data complexity of algorithm 4 is 2^{32} chosen plaintexts and the time complexity is 2^{43} S-box lookups, the probability of discarding the right key is equal to 0.

7 Chosen ciphertext attacks

In this section we show that chosen ciphertext attacks can be applied similarly above-stated chosen plaintext attacks.

Suppose $W \in \{W^{(i)}, U^{(i)}, Z^{(1)}|i \in \{1,\ldots,7\}\}$. In the same way we have $k_6 \in W \oplus c_6$, $k_3 \in W \oplus c_5$, if $k_4, k_5 \in W$, and $k_5 \in W \oplus c_5$, $k_2 \in W \oplus c_4$, if $k_3, k_4 \in W$. There are exactly $|W|^2$ encryption keys such that $k_4, k_5 \in W$. There are exactly $|W|^2$ encryption keys such that $k_3, k_4 \in W$. Using chosen ciphertext $W \oplus c_5$ instead of chosen plaintext $W \oplus c_2$ we can apply attacks similar to attacks in sections 4, 5. Using computer computations we obtained that proposition 4 is also true for the sets $(W \oplus c_5)^{\tilde{s}}$ where $W \in \{W^{(i)}|i \in \{1,\ldots,7\}\}$. Therefore using chosen ciphertext $W \oplus c_6$ instead of chosen plaintext $W \oplus c_2$ we can apply attacks similar to attacks in sections 6.

Hence for 5 rounds of Khazad we have:

1. there are 7 classes of weak keys with cardinality 2^{64}, the time complexity is 2^{35} S-box lookups, the data complexity is 2^{32} chosen ciphertexts;
2. there are 7 classes of weak keys with cardinality 2^{32}, the time complexity is 2^{19} S-box lookups, the data complexity is 2^{16} chosen ciphertexts;
3. there is 1 class of weak keys with cardinality 2^{16}, the time complexity is 2^{11} S-box lookups, the data complexity is 2^{8} chosen ciphertexts.

For 6 rounds of Khazad we have that there are 7 classes weak keys, cardinality of each of them is 2^{64}. The time complexity is 2^{43} S-box lookups, the data complexity is 2^{32} chosen ciphertext.

8 Comparison of weak keys

In this section we compare the described classes of weak keys for 5 rounds with a class of weak keys from [2]. We briefly describe the idea of the method from [2]. The encryption function of 5 rounds of Khazad can be represented in the form

$$g_{k_0,\ldots,k_5}=v_{k_0}\tilde{s}v_{k'_1}h\tilde{s}hv_{k_2}\tilde{s}v_{k'_3}h\tilde{s}hv_{k_4}\tilde{s}v_{k_5},$$

where $k'_1 = k_1^h$, $k'_3 = k_3^h$. Suppose $k_2 = k'_3$. Since $\tilde{s}$, h are involutions the transformation $h\tilde{s}hv_{k_2}\tilde{s}v_{k'_3}h\tilde{s}h$ is an involution. This property is useful to apply slide attack for 5 rounds of Khazad. So the method from [2] uses property $k_2 = k_3^h$. It is clear that k_2, k_3 belong to the same coset over subspace W. On the other hand for our weak keys we have $k_2 \in W$, $k_3 \in W \oplus c_3$, $c_3 \notin W$ (chosen plaintext) and $k_3 \in W$, $k_2 \in W \oplus c_4$, $c_3 \notin W$ (chosen ciphertext).

Therefore our sets of weak keys for 5 rounds don't intersect with the set from [2].

9 Conclusions

In this paper we investigated structural properties of the round function of Khazad. As a result we found 14 new weak key classes for 5 and 6 rounds. The cardinality of each class is 2^{64}. For recovering weak keys from 7 of

14 classes we need 2^{32} chosen plaintexts. The time complexity is 2^{35} S-box lookups for 5 rounds and 2^{45} S-box lookups for 6 rounds. For recovering weak keys from other 7 classes we need 2^{32} chosen ciphertexts. The time complexity is the same. Moreover we obtained the subclasses of these weak keys classes. For recovering weak keys from these subclasses less data and operations are required.

References

[1] Barreto P., Rijmen V., The Khazad Legacy-Level Block Cipher. In First Open NESSIE Workshop, KU-Leuven, 2000. Submissionto NESSIE.

[2] Biryukov A., Analysis of Involutional Ciphers: Khazad and Anubis. Fast Software Encryption - 2003, LNCS. Springer, 2003.

[3] Birukov A., Nikolic I., Automatic Search for Related Characteristics in Byte-Orientes Block Ciphers: Application to AES, Camellia, Khazad and Others. Advances in Cryptology - EUROCRYPT 2010. - Springer Berlin Heidelberg, 2010, pp. 322-324.

[4] Leander G., Abdelraheem M., Alkhzaimi H., Zenner E., A Cryptanalysis of PRINT cipher: The Invariant Subspace Attack. Advances in Cryptology - CRYPTO 2011, LNCS, 2011, volume 6841, pp.206-221.

[5] Muller F., A new attack against Khazad. International Conference on the Theory and Application of Cryptology and Information Security, pp. 347-358, 2003.

[6] Pogorelov B.A., Permutation groups. Part 1, general questions. Moscow, 1986, 316 p. (Russian)

[7] Pogorelov B.A., Pudovkina M.A., On the distance from permutations to imprimitive groups for a fixed system of imprimitivity. Discrete Mathematics and Applications. volume 24, issue 2, p.p. 95-108, 2014.

[8] Pogorelov B.A., Pudovkina M.A., Factor structures of transformations. Mathematical Aspects of Cryptography, 2012, volume 3, no. 3, pp. 81-104 (Russian).

[9] Pogorelov B.A., Pudovkina M.A., Combinatorical characterization of XL-layers. Mathematical Aspects of Cryptography, 2013, vol. 4, no. 3, pp. 99-129 (Russian).

[10] Yonglong T., New Cryptanalysis on 6-round Khazad. Advances in Information Sciences and Service Sciences, 2013, vol. 5 issue 1.