On Algebraic Graphs of Large Girth and New Families of Message Authentication Codes*

Vasyl Ustimenko[1,2,†] and Oleksandr Pustovit[1,*,†]

[1] Royal Holloway University of London, United Kingdom, Egham Hill, TW20 0EX Egham, United Kingdom
[2] Institute of Telecommunications and Global Information Space, NAS of Ukraine, 13 Chokolivsky ave., 02186 Kyiv, Ukraine

Abstract

Several new families of graph-based key-dependent message authentication codes are proposed. The method allows for the generation of sensitive digests of electronic documents. Computer simulation justifies a high level of corresponding avalanche effect. Algorithms are based on walks on special algebraic graphs defined by equations over the arithmetic ring Z_q, $q = 2^r$, $r \geq 8$. The cryptographic stability of proposed key-dependent message authentication codes is connected with a hard algebraic problem in investigating the systems of algebraic equalities in n variables of linear degree cn for some positive constant c. Growth of n increases the cryptographic stability. If the file is presented in the form of word in alphabet Z_q of length n and length of the walk in the graph is fixed then the execution is a linear function of size $O(n)$. Proposed algorithms can work with data in the form of texts, audio and video files, and files with various extensions such as .avi, .tif, .pdf, etc. The speed for constant m is linearly dependent on variable n. Growth of n increases the cryptographic stability. Algorithms can generate a digest of a prescribed size for a potentially infinite digital document.

Keywords

key dependent, message authentication codes, highly nonlinear multivariate cryptography, graph-based cryptography, algebraic graphs of large girth, symbolic computations

1. Introduction

A simplified model of the global information space can be imagined as a large, time-growing network of registered virtual users (individuals or institutions) who exchange information and can store it in electronic repositories located in the network or isolated from it. The size of files for exchange (electronic documents) tends to increase. An important category of the information space is the trust in documents. Users can use a symmetric algorithm with a private key to encrypt documents and a key exchange protocol to maintain the security of the encoding procedure. Certified public key algorithms can also be used to change the key. These methods ensure the security of the exchange channels.

It is easy to see that even using reliable encryption tools does not ensure complete trust in documents. The above justifies the importance and relevance of the following questions. Has the original document already been damaged? Hash function generation technologies are used to answer these questions. A hash function is needed to generate a compensated form of the original document. Note that the general hash function does not require a key or password.

For document verification and authentication tasks, so-called key hash functions (key hash functions, message authentication codes, or MACs) are required that are password-dependent.

In this paper, we propose new graph-based fast algorithms for creating digests of electronic files to detect cyberattacks, computer viruses, or other corruptions and verify data integrity. The cryptographic stability of several graph-based key-dependent hash functions is related to studying the navigation problem of finding the path between two graph vertices. In the case of the Cayley graph of the group G this is about decomposing the group element into a composition of known

* CQPC 2025: Classic, Quantum, and Post-Quantum Cryptography, August 5, 2025, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.
vasyl.ustymenko@rhul.ac.uk (V. Ustimenko); sanyk_set@ukr.net (O. Pustovt)
0000-0002-2138-2357 (V. Ustimenko); 0000-0002-3232-1787 (O. Pustovt)

generators. Research on the message authentication codes based on Cayley graphs started from the case of Ramanujan graphs constructed by Lubotzky-Philips-Sarnak and Margulis [1, 2] and Pizer [3]. Charles, Lauter, and Goren [4] gave a construction of a hash function based on these graphs. This work was motivated by the successful use of Cayley Ramanujan graphs to construct low-density parity Check codes for satellite communications. Petit, Lauter, and Quisquater [5], Tillich and Zémor [6] investigated the properties of Message Authentication Codes. These results gave a full cryptanalysis of the scheme. Carvalho Pinto and Petit [7] also developed an improved path-finding algorithm for these graphs.

Cayley-Ramanujan graphs form a family of graphs of large girth. Another family $CD(n, q)$ of graphs with the fast growth of girth was suggested by Lazebnik, Ustimenko, and Woldar [8] (see also [9] where graphs $D(n, q)$ with connected components isomorphic to $CD(n, q)$ were introduced). Guinand and Lodge [10, 11] and other researchers used corresponding graphs to construct LDPC codes. Accordingly, McKay and Postol $CD(n, q)$ based graphs have better properties than Cayley graphs of large girth [12].

The first message authentication codes based on $CD(n, q)$ graphs were proposed by Polak and Zhupa [13] and Ustimenko and Pustovit [14]. In fact, the last paper uses a more general family of graphs $CD(n, K)$ defined over an arbitrary commutative ring K. If $K = F_q$, then $CD(n, K) = CD(n, q)$ and $D(n, K) = D(n, q)$.

Ustimenko [15] proves that the girth of $D(n, K)$ is at least $n + 5$ if K is an integral domain.

Message authentication codes of [14] are defined in terms of infinite graphs $D(n, K[x_1, x_2, ..., x_m])$ where K is a finite commutative ring. The cryptographical stability of these codes rests not only on the complexity of the corresponding navigation problem but also on the complexity of the general problem of solving a nonlinear system of algebraic equations.

In this paper, the authors present a family of new robust MACs based on several families of algebraic graphs, such as generalised Wenger graphs $W(n, K)$, $D(n, K)$, $A(n, K)$, and other graphs.

All graphs under consideration are so-called linguistic graphs of type $(1, 1, n - 1)$ (see [15] and further references) introduced as bipartite graphs with partition sets isomorphic to affine spaces K^n over a commutative ring K with unity for which the incidence relation *triangular equations give me*. In fact points and lines of the linguistic graph are tuples of kind $(x) = (x_1, x_2, ..., x_n)$ and $[y] = [y_1, y_2, ..., y_n]$ and (x) and $[y]$ are incident if and only if $a_i x_i - b_i x_i = f_i(x_1, x_2, ..., x_{i-1}, y_1, y_2, ..., y_{i-1})$, $i = 2, 3, ..., n$ where a_i and b_i are elements of multiplicative group K^* of the K and $f_i \in K[x_1, x_2, ..., x_{i-1}, y_1, y_2, ..., y_{i-1}]$. The first coordinates $\varrho(x) = x_1$ and $\varrho(y) = y_1$ define the colours of the point and line. The path in the graph formed by distinct vertices is the walk $v_0 I v_1 I v_2 ... v_{k-1} I v_k$ such that the colours of all points and lines are different.

If $F_m(K)$ is a family of linguistic graphs, then the growth of m to infinity defines the projective limit $F(K)$.

If $F_m(K)$ is one of the families of graphs investigated in [16], then the walk $v_0 I v_1 I v_2 ... v_{k-1} I v_k$ of $F(K)$ such that $\varrho(v_i)$-$\varrho(v_{i+2}) \in K^*$ for $I = 0, 1, ..., k - 2$ is the path.. Cayley-Ramanujan graphs form a family of graphs of large girth. Another family $CD(n, q)$ of graphs with the fast growth of girth was suggested by Lazebnik, Ustimenko, and Woldar [8] (see also [9] where graphs $D(n, q)$ with connected components isomorphic to $CD(n, q)$ were introduced). Guinand and Lodge used corresponding graphs.

In this paper, we select the case when $K = Z_q$, $q = 2^m$, $m \geq 2$, and the aforementioned linguistic graphs to define new families of message authentication codes. In Section 1, we introduce an algorithm for converting an arbitrary word $(\alpha_1 \alpha_2, ..., \alpha_n)$ written in the alphabet Z_q into the digest $(d_1, d_2, ..., d_k)$, which is the word of length k, $k < n$. These algorithms are defined in terms of arbitrary linguistic graphs. The complexity estimates are given in the Jordan-Gauss graphs defined in [16]. Section 2 is dedicated to the requirements on the cryptographical stability of message authentication codes. We discuss the general complexity of investigating a nonlinear system of equations, which justifies the security of the presented MACs. Section 3 introduces our algorithms regarding arbitrary linguistic graphs of type $(1, 1, n - 1)$. Section 4 describes our selected algebraic graphs defined by the sparce quadratic system of equations and discusses their complexity and

parameters of the corresponding algebraic systems of equations. Section 5 is dedicated to implementing MACs described in the previous section. Section 6 contains conclusive remarks.

2. Requirements for digesting

The cryptographically stable hash function f must provide the practical impossibility of selecting a pair of links x and z with the same hash function value. The digest of a document created with a key-dependent hash function (MAC) uses the HMAC symbol. When users want to exchange correspondence secretly, verifying the author of the letter and the absence of changes when forwarding, they choose a shared MAC. Additionally, they use a standard symmetric encryption scheme.

In addition to cryptographical stability, the execution speed and a high indicator of avalanche effect are important. The avalanche effect can be measured in the following way. The HMAC of the generated file has to be computed. After this step, a chosen character of the original file has to be changed to another symbol, and HMAC for the new file has to be computed.

Finally, a comparison of the characters of two HMACs has to be made, and the percentage of changed characters has to be computed. For practical usage of HMAC, it is necessary to show that a change of an arbitrarily used character leads to a change of at least 40% of bits independently of the size of the tested files.

The introduced approach of using walks on algebraic graphs defined over a finite commutative ring K is helpful for the development of stream ciphers of Symmetric Cryptography (see, for instance, [17], [18], and further references) and constructions of HMACs. Other applications of graph theory to Symmetric Cryptography are considered in [19]. The method of generation of nonlinear transformations of free modules over commutative rings described in terms of special graphs defined by algebraic equations (so-called linguistic graphs) can be used in Non-commutative Cryptography instead of methods of generators and equations [20].

Studies of message authentication codes and HMACs are a hot topic. A complete list of all published papers within this direction is impossible; we only refer to some recent papers [21–29].

Recall that non-commutative cryptography is an active field of cryptology that explores cryptographic primitives and systems based on algebraic structures such as groups, semigroups, and non-commutative rings.

One of the earliest applications of non-commutative algebraic structure for cryptographic purposes was using groups to develop cryptographic protocols.

Noteworthy that arbitrary hash functions such as MD5 or SHA1 can be used to compose HMACs corresponding to MD5 and SHA-1 message authentication codes, which are known as HMAC-MD5 and HMAC-SHA-1, respectively. HMAC's cryptographic performance depends on the cryptographic performance of the underlying hash function, the size of its hash output, and the size and quality of the key.

3. Description of HMACs based on linguistic graphs defined over Z_q, $q = 2^m$

Let us assume that $q = 2^m$ and ${}^nI$ is a bipartite linguistic graph with the partition sets formed by points $(x) = (x_1, x_2, \ldots, x_n)$, $x_i \in Z_q$ and lines $[y_1, y_2, \ldots, y_n]$, $y_i \in Z_q$. Recall that $(x)^nI[y]$ if and only if the following equations hold

$$\begin{gathered} a_2x_2 - b_2x_2 = f_2(x_1, y_1), \\ a_3x_3 - b_3x_3 = f_3(x_1, x_2, y_1, y_2), \\ \ldots \\ a_nx_n - b_nx_n = f_i(x_1, x_2, \ldots, x_{n-1}, y_1, y_2, \ldots, y_{n-1}). \end{gathered} \tag{1}$$

We define the operator $N_a(v)$ of taking the neighbour of colour a via the following rule. If vertex v is the point (p) then $N_a(p)$ is the neighbouring line of this point with the colour a, i. e line $[l] = [a, l_2, \ldots, l_n]$ where l_i, $i = 2, 3, \ldots, n$ are defined recurrently from the equations $a_2p_2 - b_2l_2 = p_1a$, $a_3p_3 - b_3l_3 = f_3(p_1, p_2, l_1, l_2), \ldots, a_ip_i - b_il_i = f_i(p_1, p_2, \ldots, p_{i-1}, l_1, l_2, \ldots, l_{i-1})$. If vertex v is the line $[l]$ then $N_a(l)$ is the neighbouring point of the line with the colour a, i.e., point $(p) = (a, p_2, \ldots, p_n)$ where p_i, $i = 2, 3, \ldots, n$ are defined recurrently from the equations $a_2p_2 - b_2l_2 = al_1$, $a_3p_3 - b_3l_3 = f_3(p_1, p_2, l_1, l_2), \ldots, a_ip_i - b_il_i = f_i(p_1, p_2, \ldots, p_{i-1}, l_1, l_2, \ldots, l_{i-1})$.

In the following algorithms, we assume that the commutative ring $K = Z_q$ and the graph $nI(K)$ internal parameters are among the input data. So, the operations of addition and multiplication of the ring are given by the loaded addition and multiplication tables. So we have embedded functions $x + y$ and $x \boxtimes y$ where x and y are taken from the set of residues mod q. We will use another embedded power function x^y, where x is from the domain of all odd residues and y is the residue modulo 2^{m-1}, which is the order of the multiplicative group K^*. Parameters give the graph $nI(K)$ a_i, b_i, $i = 2, 3, \ldots, n$, and a list of polynomials f_i is given in their standard forms, i.e., lists of monomial terms of f_i, $i = 2, 3, \ldots, n$ shown in the lexicographical order. The data described above is public. The input tuple of our algorithm is of the kind $(x_1, x_2, \ldots, x_n)$, $x_i \in K = Z_q$, $q = 2^{8s}$, $s \geq 1$. So we can treat each x_i as a tuple $(z_1, z_2, \ldots, z_s)$ from Z_b, $b = 256$ corresponding to the residue $z_1 + z_2b + \ldots + z_sb^{s-1}$ from Z_q, and use the standard one-to-one correspondence between elements of Z_b and *ASCII* codes of binary alphabet. So we can partition the text in binary alphabet into words of length s and treat each word as an element of the ring $K = Z_q$.

The passive password of our secret key of our key dependent message authentication codes is the tuples $(\alpha_2, \alpha_3, \ldots, \alpha_n)$, $(\gamma_2, \gamma_3, \ldots, \gamma_n)$ where $\alpha_j \in (Z_q)^*$ and $(\lambda_2, \lambda_3, \ldots, \lambda_n)$ where λ_i are even residues together with the tuple $((a(2), a(3), \ldots, a(n))$ formed by elements of $a(j) \in \{1, 2, \ldots, 2^{8s-1} - 1\}$. We refer to this list of parameters as a *passive password.* We additionally use *the format parameter* $k = O(n^t)$, $k < n$, where $0 < t \leq 1$.

The active password is formed by *depth parameter* m of size $O(1)$ together with tuples $(\beta(1), \beta(2), \ldots, \beta(m))$ and $(\mu(1), \mu(2), \ldots, \mu(m))$ from $((Z_q)^*)^m$.

We consider bijective transformations L of kind $x_1 \longrightarrow x_1 + \alpha_2x_2 + \ldots + a_nx_n$, $x_j \longrightarrow x_j$, $j = 2, 3, \ldots, n$ and the value of multivariate polynomial $Q(x_1, x_2, \ldots, x_n) = x_1(\lambda_2x_2 + \gamma_2)^{a(2)}(\lambda_3x_3 + \gamma_3)^{a(3)} \ldots (\lambda_nx_n + \gamma_n)^{a(n)} = Q$.

To control the speed of computation of Q we assume that ''almost all'' parameters a(i), I ≥ 2, coincide with 1, i.e., the sum of parameters a(i) which are distinct from 1 is the selected constant d.

HMAC algorithm 1 (two versions)

Let $(x_1, x_2, \ldots, x_n)$ be the intermediate point of the graph. We assume that it is the input of the algorithm.

1. Compute $y(1) = L(x) = (a, x_2, x_3, \ldots, x_n)$.
   Compute $N_Q(y(1)) = z(1) = [c, y_2, y_3, \ldots, y_n]$.
2. $y(2) = N_{a+\beta(1)}(z(1))$.
   $z(2) = N_{c+\mu(1)}(y(2))$.
3. $y(3) = N_{c+\beta(1)+\beta(2)}(z(2))$.
   $z(3) = N_{c+\mu(1)+\mu(2)}(y(3))$.

We continue this process with m-steps to get $y(m) = N_{a+\beta(1)+\beta(2)+\ldots+\beta(m-1))}(z(m-1))$ and $z(m) = N_{c+\mu(1)+\mu(2)+\ldots+\beta(m-1)}(y(m))$.

We have two versions of the algorithm via selection of $y(m)$ or $z(m)$ as the intermediate output $v = (v_1, v_2, \ldots, v_n)$.

Finally, we select parameter $k = O(n^t)$, $k < n$, where $0 < t \leq 1$, and announce that $(v_{n-k}, v_{n-k+1}, v_{n-k+2}, \ldots, v_n)$ is the output of the algorithm.

HMAC algorithm 2.

The first step is the same as the first computation of the previous algorithm. So we get $y(1)$ and $y(2)$. In each of the following steps, we permute two colours of the vertices.

2. $y(2) = N_{c+\mu(1)}(z(1))$, $z(2) = N_{a+\beta(1)}(y(2))$.
3. $y(3) = N_{c+\mu(1)+\mu(2)}(z(2))$, $z(3) = N_{a+\beta(1)+\beta(2)}(y(3))$.

We continue this process with m-steps to get $y(m) = N_{c + \mu(1) + \mu(2) + \ldots + \mu(m-1)}(z(m-1))$ and $z(m) = N_{a + \beta(1) + \beta(2) + \ldots + \beta(m-1)}(y(m))$.

Similarly, we have two versions of the algorithm via the selection of $y(m)$ or $z(m)$ as the intermediate output $v = (v_1, v_2, \ldots, v_n)$. Finally, we select parameter $k = O(n^t)$, $k < n$, where $0 < t \leq 1$, and announce that $(v_{n-k}, v_{n-k+1}, v_{n-k+2}, \ldots, v_n)$ is the output of the algorithm.

The multivariate maps $(x_1, x_2, \ldots, x_{n-k}) \longrightarrow (v_{n-k}(x_1, x_2, \ldots, x_n), v_{n-k+1}(x_1, x_2, \ldots, x_n), \ldots, v_n(x_1, x_2, \ldots, x_n))$ for the computation of the outputs in the two algorithms presented above have different degrees and densities.

HMAC algorithm 3

This algorithms will use the map $Q(x_1, x_2, \ldots, x_n)$ of K^n on K^n of kind $x_1 \longrightarrow x_1(\lambda_2 x_2 + \gamma_2)^{a(2)}(\lambda_3 x_3 + \gamma_3)^{a(3)} \ldots (\lambda_n x_n + \gamma_n)^{a(n)}$, $x_j \longrightarrow x_j$ and linear function $M = x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n$.

We take $(x) = (x_1, x_2, \ldots, x_n)$ and compute $Q(x) = (a, x_2, x_3, \ldots, x_n) = y(1)$, parameter $M(x_1, x_2, \ldots, x_n) = c$ and $z(1) = N_c(y(1))$.

Then we use recursive procedures of the MAC Algorithm 1 to compute recurrently $y(2)$, $z(2)$, $y(3)$, $z(3)$, …, $y(m)$, $z(m)$. For the selected parameter k, we take k last coordinates of $y(m)$ or $z(m)$ and use this tuple as the output.

4. On the selected graphs for the implementation

a. General Jordan-Gauss graphs

Accordingly to [16], we refer to the linguistic graph ${}^nI(K)$ given by equations of kind (1) as the Jordan-Gauss graph in the case when

$$\begin{gathered} f_2 = a_2(1, 1)x_1y_1, \\ f_3 = a_3(1, 1)x_1y_1 + a_3(1, 3)x_1y_2 + a_3(2, 3)x_2y_2, \\ f_4 = a_4(1, 1)x_1y_1 + a_4(1, 2)x_1y_2 + a_4(1, 3)x_1y_3 + a_4(2, 2)x_2y_2 + a_4(2, 3)x_2y_3 + a_4(3, 3)x_3y_3, \\ \ldots \\ f_n = a_n(1, 1)x_1y_1 + a_n(1, 2)x_1y_2 + \ldots + a_n(1, n-1)x_1y_{n-1} + a_n(2, 2)x_2y_2 + a_n(2, 3)x_2y_3 + \\ + a_n(2, n-1)x_2y_{n-1} + + \ldots + + a_n(n-1, n-1)x_{n-1}y_{n-1}. \end{gathered} \tag{1}$$

We refer to the graph as a dense Jordan-Gauss if all coefficients $a_i(k, l)$ as above are different from zero**.** If ${}^nI(K)$ is a dense Jordan-Gauss graph, we can keep it secret by adding the coefficients of the graph equations to the passive password. The theoretical complexity of the presented above algorithms is $O(n^2)$.

b. Special Jordan-Gauss graphs

We select for the implementation several well-known sparse Jordan-Gauss graphs for which the number of nonzero coefficients $a_i(k, l)$ is $O(n)$.

Example 4.2.1. Generalised Wenger graph.

This is a Jordan-Gauss graph with partition its isomorphic to K^n, such that point $(x_1, x_2, \ldots, x_n)$ is incident to line $[y_1, y_2, \ldots, y_n]$ if and only if the following equations hold

$$\begin{gathered} x_2 - y_2 = x_1y_1, \\ x_3 - y_3 = x_1y_2, \\ \ldots \\ x_n - y_n = x_1y_{n-1}. \end{gathered} \tag{1}$$

Wenger introduced this family of graphs in [30] for the case of $K = Z_p$ where p is prime.

Example 4.2.2.

Let K stand for an arbitrary commutative ring with unity. We consider the graph $D(K)$ with points and lines which are infinite tuples over K of kind $(p) = (p_1, p_2, \ldots, p_i, p_{i+1}, \ldots)$, $[l] = [l_1, l_2, \ldots, l_i,$

l_{i+1}, ...] with the following incidence relations. The point (p) is incident to line $[l]$ if the following equation holds.

$$\begin{gathered} l_2 - p_2 = l_1 p_1, \\ l_3 - p_3 = l_2 p_1, \\ l_4 - p_4 = l_1 p_2, \\ l_i - p_i = l_1 p_{i-2}, \\ l_{i+1} - p_{i+1} = l_{i-1} p_1, \\ l_{i+2} - p_{i+2} = l_i p_1, \\ l_{i+3} - p_{i+3} = l_1 p_{i+1} \text{ where } i \geq 5. \end{gathered} \tag{1}$$

Graphs $D(n, K)$, $n \geq 2$, were introduced in [15]. In fact, they are homomorphic images of $D(K)$ under a homomorphism sending (p) into $(p_1, p_2, \ldots, p_n)$ and $[l]$ into $[l_1, l_2, \ldots, l_n]$. So the incidence Jordan-Gauss graph $D(n, K)$ is defined by the first $n - 1$ equations in the list presented above.

The graph G's girth $g(G)$ is the size of its minimal cycle. The forest is the graph without a cycle. The connected forest is called the tree [31]. If K is an integrity domain, i.e., K does not contain zero divisors, then the girth $D(n, K)$ of is at least $n + 5$ [15]. These facts were established in [9] for the case of finite fields.

Example 4.2.3.

Infinite Jordan-Gauss graph $A(K)$ with points $(p) = (p_1, p_2, \ldots, p_i, p_{i+1}, \ldots)$ and lines $[l] = [l_1, l_2, \ldots, l_i, l_{i+1}, \ldots]$ with coordinates from commutative ring K is defined by equations

$$\begin{gathered} l_2 - p_2 = l_1 p_1, \\ l_3 - p_3 = p_1 l_2, \\ l_4 - p_4 = l_1 p_3, \\ l_5 - p_5 = p_1 l_4, \\ \ldots \end{gathered} \tag{1}$$

Graphs $A(n, K)$, $n \geq 2$ introduced in [15] are homomorphic images of $A(K)$ under a homomorphism sending (p) into $(p_1, p_2, \ldots, p_n)$ and $[l]$ into $[l_1, l_2, \ldots, l_n]$. So the incidence of Jordan-Gauss graph $A(n, K)$ is defined by the first $n - 1$ equations presented above.

If K is an integrity domain, then $A(K)$ is the forest (see [16] and further references), the girth of $A(n, K)$ is $\geq [(n + 2) / 2]$.

We can see that the quadratic monomial term on the right-hand side of the equations uniquely defines each equation of $D(K)$. Let D be the set of monomial terms of the equation of $D(K)$, and A be the set of monomial terms of the $A(K)$ equations. Note that the deletion of equations of $D(K)$ with monomial terms from $D - A$, together with corresponding equations, defines the homomorphism of the forest $D(K)$ onto $A(K)$.

Example 4.2.4

The equations of the graph $D(K)$ and corresponding quadratic monomial terms are ordered accordingly to the list (2). Let ${}^rB(K)$ be the homomorphic image of $D(K)$ obtained by the deletion of all equations with the number $> r$ which contain the monomials from D-A. Let ${}^rB(n, K)$, $n > r + 1$ be the graph with partition sets isomorphic to K^n defined via the first $n - 1$ equations of ${}^rB(K)$ (see [16] and further references).

In the case of Examples 4.2.1–4.2.4, the complexity of the generating procedure of the message authentication codes is $O(n)$.

In cases 4.2.1–4.2.3, we assume that the graphs are known to the public. In the case of the graph of Example 4.2.4, we can add the parameter n to the passive password.

5. Comments on the implementation

We investigate our algorithms in the commutative ring Z_{256}, which is the same size as the binary alphabet. We select the cases of sparse graphs described in Examples 4.2.1–4.2.4 for the implementation. We choose the parameters of the passive password as follows. Only $O(1)$ values of kind α_i, β_i, and $a(j)$ differ from 1, and only a selected finite number of λ_i differ from 2. The passive password remains the same during our computer simulation.

We refer to $d = 2m$ as the *length of the walk*. In the cases of 4.2.2–4.2.4, the change of active password leads to the shift in intermediate output $v = (v_1, v_2, ..., v_n)$. The walk's computation speed is the same in all cases 4.2.1–4.2.4 because Jordan-Gauss graphs have equations with the single quadratic monomial terms in each equation. Note that the time execution does not depend on parameter k, which defines the digest size.

Computer simulations demonstrate a high level of the avalanche effect. We can see that a single change of the character of the initial file or a single shift in the character of the active password leads to a change of 98% of the characters of the output. Not that this is essentially better than in the case of HMAC [32] when the change of characters of the initial file leads to the change of 47-50 characters of the output.

Our software is written in C++; therefore, it is portable and runs on many platforms, such as Unix/Windows. To evaluate our algorithm's performance, we use files of different sizes. We measure the time (Fig. 1) needed to produce the digest in milliseconds. And the file size of files in kilobytes for passwords of length d.

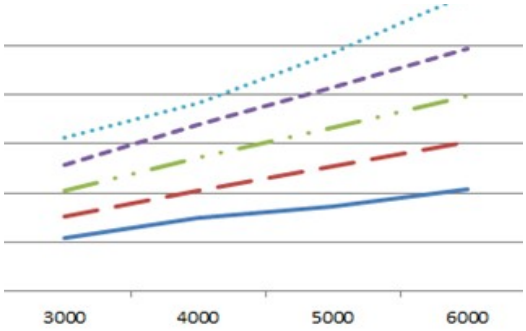

Figure 1: Run time for the digest generation

We use an average PC with a Pentium 3.00 GHz, 2GB of RAM, and Windows 7. The following diagram presents the time of the algorithm's execution in the case of graphs 4.2.2. on in the case of graphs 4.2.2.

Conclusions

The paper proposes new fast graph-based algorithms for creating sensitive digests of electronic files to detect cyberattacks, computer viruses, or other damages and check data integrity. These tools can be used to defend a virtual organization and audit all files after a registered intervention. Cryptographic stability of new key-dependent hash functions is associated with complex algebraic problems, such as the study of systems of algebraic equations of a considerable degree and the decomposition of a nonlinear transformation into the composition of given generators. These facts justify resistance of digests against adversary attacks with the use of algorithms in terms of a Turing machine or the theory of Quantum Computation.

Algorithms of digest generation use the idea of presentation of files in the form of sequences (words) of elements of the arithmetical ring modulo 2^m. The presented families of graphs form sequences that define the projective limit of finite graphs given by equations.

Affine transformations and polynomial maps of high degree are used to hide the transformation induced by the walk on the graph. This scheme is new,

Implemented according to this scheme, a family of fast algorithms was investigated via computer simulations on large real data sets. Changing a single document character in the binary alphabet causes the change of most characters of the produced digest (≥98%). This property and the evaluation of the time execution of software programs justify the potential of practical usage of the implemented algorithm for cybersecurity tasks.

Declaration on Generative AI

While preparing this work, the authors used the AI programs Grammarly Pro to correct text grammar and Strike Plagiarism to search for possible plagiarism. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

References

[1] A. Lubotzky, R. Phillips, P. Sarnak, Ramanujan graphs, Combinatorica 8(3) (1988) 261–277. doi:10.1007/BF02126799

[2] G. Margulis, Explicit constructions of graphs without short cycles and low density codes. Combinatorica 2 (1982) 71–78. doi:10.1007/BF02579283

[3] A. K. Pizer, Ramanujan graphs and Hecke operators. Bull. Amer. Math. Soc. 23(1) (1990) 127–137.

[4] D. X. Charles, K. E. Lauter, E. Z. Goren, Cryptographic hash functions from expander graphs, J. Cryptol. 22(1) (2009) 93–113. doi:10.1007/s00145-007-9002-x

[5] C. Petit, K. Lauter, J.-J. Quisquater, Full cryptanalysis of LPS and Morgenstern hash functions, in: Security Cryptography Networks, 2008, 263–277. doi:10.1007/978-3-540-85855-3_18

[6] J.-P. Tillich, G. Zémor, Collisions for the LPS expander graph hash function, in: Advances in Cryptology (EUROCRYPT), 4965, 2008, 254–269. doi:10.1007/978-3-540-78967-3_15

[7] E. Carvalho Pinto, C. Petit, Better path-finding algorithms in LPS Ramanujan graphs, J. Math. Cryptol. 12(4) (2018) 191–202.

[8] F. Lazebnik, V. Ustimenko, A. J. Woldar, A new series of dense graphs of high girth, Bull. Amer. Math. Soc. 32(1) (1995) 73–79. doi:10.1090/S0273-0979-1995-00569-0

[9] F. Lazebnik, V. Ustimenko, Some algebraic constructions of dense graphs of large girth and of large size, in: Discrete Mathematics and Theoretical Computer Science (DIMACS), 10, 1993, 75–93. doi:10.1090/dimacs/010/07

[10] P. Guinand, J. Lodge, Tanner type codes arising from large girth graphs, in: 1997 Canadian Workshop on Information Theory (CWIT), 1997, 5–7.

[11] P. Guinand, J. Lodge, Graph theoretic construction of generalized product codes, in: 1997 IEEE Int. Symposium on Information Theory (ISIT), 1997, 111. doi:10.1109/ISIT.1997.613026

[12] D. MacKay, M. Postol, Weakness of Margulis and Ramanujan margulis low dencity parity-check codes, Electron. Notes Theor. Comput. Sci. 74 (2003) 97–104. doi:10.1016/S1571-0661(04)80768-0

[13] M. Polak, E. Zhupa, Keyed hash function from large girth expander graphs, Albanian J. Math. 16(1) (2022) 25–39. doi:10.51286/albjm/1656414764

[14] V. Ustimenko, O. Pustovit, On new stream algorithms generating sensitive digests of computer files, in: Annals of Computer Science and Information Systems, 16th Conference on Computer Science and Intelligence Systems, 26, 2021, 117–121. doi:10.15439/2021f80

[15] V. Ustimenko, On linguistic dynamical systems, families of graphs of large girth, and cryptography, J. Math. Sci. 140(3) (2007) 461–471. doi:10.1007/s10958-007-0453-2

[16] T. Chojecki, et al., On affine forestry over integral do-mains and families of deep Jordan–Gauss graphs, Eur. J. Math. 11(10) (2025). doi:10.1007/s40879-024-00798-2

[17] V. Ustimenko, et al., On the constructions of new symmetric ciphers based on nonbijective multivariate maps of prescribed degree, Secur. Commun. Netw. (2019) 1–15. doi:10.1155/2019/2137561
[18] V. Ustimenko, T. Chojecki, Walks on algebraic small world graphs of large girth and new secure stream ciphers, in: Intelligent Systems and Applications (IntelliSys), Lecture Notes in Networks and Systems, 1067, 2024, 525–538. doi:10.1007/978-3-031-66431-1_37
[19] P. L. K. Priyadarsini, A survey on some applications of graph theory in cryptography, J. Discret. Math. Sci. Cryptogr. 18(3) (2015) 209–217. doi:10.1080/09720529.2013.878819
[20] V. Ustimenko, Graphs in terms of algebraic geometry, symbolic computations and secure communications in post-quantum world. UMCS Editorial House, Lublin, 2022.
[21] M. Cary, R. Venkatesam, A message authentication code based on unimodular matrix groups, Advances, in: Lecture Notes in Computer Science, 23rd Annual Int. Cryptology Conf., 2003, 500–512. doi:10.1007/978-3-540-45146-4_29
[22] M. Bellare, D. J. Bernstein, S. Tessaro, Hash-function based PRFs: AMAC and its multi-user security, in: Lecture Notes in Computer Science, 2016, 566–595. doi:10.1007/978-3-662-49890-3_22
[23] K. Yasuda. A double-piped mode of operation for MACs, PRFs and PROs: Security beyond the birthday barrier, in: Lecture Notes in Computer Science, 2009, 242–259. doi:10.1007/978-3-642-01001-9_14
[24] X. Wang, et al., Cryptanalysis on HMAC/NMACMD5 and MD5-MAC, in: Lecture Notes in Computer Science, 5479, 2009, 121–133. doi:10.1007/978-3-642-01001-9_7
[25] G. Leurent, T. Peyrin, L. Wang, New generic attacks against hash-based MACs, in: Advances in Cryptology—ASIACRYPT, 8270, 2013, 11–20.
[26] N. Koblitz, A. Menezes, Another look at HMAC, Cryptology ePrint Archive, Report 2012/074, 2012.
[27] Y. Dodis, et al., Message authentication, revisited, in: Advances in cryptology (EUROCRYPT), 7237, 2012, 355–374. doi:10.1007/978-3-642-29011-4_22
[28] A. Bessalov, V. Sokolov, S. Abramov, Efficient commutative PQC algorithms on isogenies of Edwards curves, Cryptogr. 8(3), iss. 38 (2024) 1–17. doi:10.3390/cryptography8030038
[29] A. Bessalov, et al., Multifunctional CRS encryption scheme on isogenies of nonsupersingular Edwards curves, in: Workshop on Classic, Quantum, and Post-Quantum Cryptography, 3504, 2023, 12–25.
[30] R. Wenger, Extremal graphs with no C4, C6 and C10s, J. Comb. Theory, Ser B., 52 (1991) 113–116. doi:10.1016/0095-8956(91)90097-4
[31] B. Bollobash, Extremal graph theory, Academic Press, London, 1978.
[32] S. Krendelev, P. Sazonova, Parametric hash function resistant to attack by quantum computer, in: 2018 Federated Conf. on Computer Science and Information Systems (ACSIS), 15, (2018) 387–390.