A New Post-Quantum Key Agreement Protocol and Derived Cryptosystem Based on Rectangular Matrices

HUGO DANIEL SCOLNIK [1, 2, 3, 4] hugo@dc.uba.ar, hscolnik@gmail.com

JUAN PEDRO HECHT [3] phecht@dc.uba.ar, qubit101@gmail.com

[1]*Instituto de Ciencias de la Computación, Universidad de Buenos Aires and CONICET, Buenos Aires, Argentina.*

[2]*Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Buenos Aires, Argentina.*

[3]*Maestria en Seguridad Informática – Facultades de Ciencias Económicas, Ciencias Exactas y Naturales, Ingeniería, Universidad de Buenos Aires, Buenos Aires, Argentina.*

[4]*Corresponding author*

Abstract. In this paper, we present an original algorithm to generate session keys and a subsequent generalized ElGamal-type cryptosystem. The scheme presented here has been designed to prevent both linear and brute force attacks using rectangular matrices and to achieve high complexity. Our algorithm includes a new generalized Diffie-Hellmann scheme based on rectangular matrices and polynomial field operations.

Keywords: key-exchange protocol, noncommutative algebraic cryptography, postquantum cryptography, rectangular matrices, GDHP.

Statements and declarations: the authors declare no competing financial interests. No funding was received for conducting this study.

1. Introduction

It is well known that generating secure key exchange algorithms is a priority for implementing symmetric protocols [16]. The idea of public key cryptography goes back to the work of James Ellis [8] and the seminal work of Diffie-Hellman [7] and its variants, which were the first practical solutions universally used in SSL, TLS, SSH, IPsec, PKI, Signal, etc. On the other hand, the imminent appearance of quantum computers able to implement Shor's and Grover's algorithms [2], which seriously affect the currently used cryptographic methods, led to the current research efforts in post quantum cryptography (PQC).

This paper was inspired by E. Stickels's proposals [22], which were cryptanalyzed by V. Shpilrain [18] and C. Mullan [17]. More recently, S. Kanwal and R. Ali [11] published an interesting protocol, but it was also cryptanalyzed by J. Liu et al. [14]. A natural alternative was to use rank-deficient matrices, but this has been cryptanalyzed by F. Virdia using Jordan canonical forms[23]. More recently, Daniel Brown [4] presented a promising attack on our original algorithm which led to this updated version.

It is worthwhile to note that in the NIST competition for standardization of postquantum protocols [19], there is none based on the use of noncommutative algebraic systems [18], those dedicated to key exchange protocols (KEP) and their canonical asymmetric cryptosystems, derived using a generalized ElGamal scheme. This paper aims to provide an alternative solution in this regard.

2. Paper organization

First, we present an overall description of the proposed algorithms and the corresponding protocols, the proof that Alice and Bob will derive a common key, security considerations, and finally some experimental results and a discussion.

3. The notation used in this work

dim: integer (rows of a square matrix), Z_p: set of non-negative residuals mod p, products in Z_p (represented by dots), $\oplus$: field sum of integers or matrices in F_{256} , $\odot$: field product of integers or matrices in F_{256}, $X^{\odot k}$: field k-power of a X-matrix in F_{256}, ||: concatenation Per[A]: the permanent of matrix A, A(i, j): matrix component of the i-th row and j-th column, $\in_{rand}$: random uniform selection in a closed interval.

4. Overview of the key exchange algorithm

It is a key exchange algorithm (KEM) that operates on rectangular matrices, mixing conventional linear operations with operations on the polynomial ring of matrices:

$R\{\ M(\mathbb{F}_{256}, dim),\ \oplus,\ \odot\ \}$. The entries of the matrices are elements of the polynomial field $F_{256} / x^8+x^6+x^3+x^2+1$, and therefore each one is a byte.
The algorithm requires a double exchange between Alice and Bob, although if a single one is required this can be achieved by sharing a priori a high order generator matrix in the ring.
The algebraic security of the protocol is based on the difficulty of solving a class of the generalized DH problem (GDHP)[16], that is on the difficulty of solving discrete log problems over matrix powers that use field operations since no classical or quantum P-class is known to solve it [16, 18].

5. Diagram of each cycle of the key exchange algorithm

ALICE	BOB
Setup $F_{256} / x^8+x^6+x^3+x^2+1$ / <x+1> operations dim $\in_{random}$ $\mathbb{Z}$; cols=dim; rows < cols; dim≥ 16; h ≥ 64	
A1 (rows x cols) $\in_{random}$ $\mathbb{Z}_{256}$ B1 (cols x rows) $\in_{random}$ $\mathbb{Z}_{256}$ Pa = A1 . B1 (mod 256) $\longrightarrow$ Pb $\longleftarrow$	A1 (rows x cols) $\in_{random}$ $\mathbb{Z}_{256}$ B1 (cols x rows) $\in_{random}$ $\mathbb{Z}_{256}$ $\longrightarrow$ Pa $\longleftarrow$ Pb = A2 . B2 (mod 256)

core = Pa $\odot$ Pb
expoA $\in_{random}$ $[2^{h-1}, 2^h]$
U = $core^{\odot expoA}$ $\longrightarrow$
V $\longleftarrow$
	core = Pa $\odot$ Pb
	expoB $\in_{random}$ $[2^{h-1}, 2^h]$
	$\longrightarrow$ U
	$\longleftarrow$ V = $core^{\odot expoB}$

Ka = $V^{\odot expoA}$ compact Ka = Per [Ka] (mod 2^h)	Kb = $U^{\odot expoB}$ compact Kb= Per[Kb] (mod 2^h)

6. Diagram of the coupled El Gamal cipher algorithm

ALICE	BOB
Setup F_{256} / $x^8+x^6+x^3+x^2+1$ / <x+1> operations H(): SHA3-512; h ≥ 64	

(C, D) $\longleftarrow$	msg =”any secret here” (padded to 512-bits)
	C = V = $core^{\odot expoB}$
	D = H(compact Kb) ⊕ msg
	(C,D)

Ka = $C^{\odot expoA}$
compact Ka = Per [Ka] (mod 2^h)
msg = D ⊕ H(compact Ka)	

7. Key exchange algorithm

ALGORITHM 1: PQC multiKEP

COMMENTS
The key exchange algorithm (KEP) uses several internal cycles as defined below and is therefore defined here as a multiKEP. **INPUT:** see the initial configuration. **OUTPUT:** shared session key of 512 bits.
INITIAL CONFIGURATION (PUBLIC VALUES):
dim: integer (proposed ≥ 16)
rows[X|, columns[X]: dimensions of the matrices X:{A, B}, where rowsA=columnsB=dim, columnsA=rowsB and rowsA > columnsA.
h: integer (proposed ≥ 64)
t: number of iterations (proposed ≥ 10)
H( ): hashing SHA3-512.

ALICE

```
 1. for k=1 to t
 2.     for i=1 to rowsA
 3.         for j=1 to columnsA
 4.             $A1_k(i,j) \in_{rand} \mathbb{Z}_{256}$
 5.         next j
 6.     next i
 7.     for i=1 to rowsB
 8.         for j=1 to columnsB
 9.             $B1_k(i,j) \in_{rand} \mathbb{Z}_{256}$
10.         next j
11.     next i
12.     $Pa_k = A1_k . B1_k$ (mod 256)
13. next k
14. Send the vector $Pa = (Pa_1, \ldots, Pa_t)$ to Bob and receive vector Pb
```

```
15. for k=1 to t
16.     $core_k = Pa_k \odot Pb_k$
17.     $expoA_k \in_{rand} [2^{h-1}, 2^h]$
18.     $U_k = core_k^{\odot expoAk}$
19. next k
20. Send the vector $U = (U_1, \ldots, U_t)$ to Bob and receive vector V
```

BOB

```
21. for k=1 to t
22.     for i=1 to rowsB
23.         for j=1 to columnsB
24.             $A2_k(i,j) \in_{rand} \mathbb{Z}_{256}$
25.         next j
26.     next i
27.     for i=1 to rowsA
28.         for j=1 to columnsB
29.             $B2_k(i,j) \in_{rand} \mathbb{Z}_{256}$
30.         next j
31.     next i
32.     $Pb_k = A2_k . B2_k \pmod{256}$
33. next k
34. Send the vector $Pb = (Pb_1, \ldots, Pb_t)$ to Bob and receive vector Pa
35. for k=1 to t
36.     $core_k = Pa_k \odot Pb_k$
37.     $expoB_k \in_{rand} [2^{h-1}, 2^h]$
38.     $V_k = core_k^{\odot expoBk}$
39. next k
40. Send the vector $V = (V_1, \ldots, V_t)$ to Bob and receive vector U
```

SESSION KEY OBTAINED BY ALICE

```
41. for k=1 to t
42.     $A\text{-}KEY_k = Per[V_k^{\odot expoAk}] \pmod{2^h}$
43. next k
44. $A\text{-}CONCAT = A\text{-}KEY_1 \| A\text{-}KEY_2 \| \ldots \| A\text{-}KEY_t$
45. $KEY_{alice} = H(A\text{-}CONCAT)$
```

SESSION KEY OBTAINED BY BOB

```
46. for k=1 to t
47.     $B\text{-}KEY_k = Per[U_k^{\odot expoBk}] \pmod{2^h}$
48. next k
49. $B\text{-}CONCAT = B\text{-}KEY_1 \| B\text{-}KEY_2 \| \ldots \| B\text{-}KEY_t$
50. $KEY_{bob} = H(B\text{-}CONCAT)$
```

8. Algorithm 1: keys equality proof

Lemma 1:
The keys given by Algorithm 1 are equal, $\text{A-KEY}_{alice} = \text{B-KEY}_{bob}$

Proof:
The result follows from the fact that the matrix powers of equal base commute in the matrix ring, and hence U = $core^{\odot expoA}$ *and* V = $core^{\odot expoB}$ satisfy $U \odot V = V \odot U = core^{\odot expoA.expoB} = core^{\odot expoB.expoA}$ ■

9. Derived cipher algorithm

ALGORITHM 2: PQC multiKEP + ElGamal cipher

Observation: Bob sends a message to Alice. Vector U was received by Bob

Insert here algorithm 1

ELGAMAL (C, D):

```
1.    Select msg string padded to 512-bits
2.    C = V
3.    D = H(compact Kb) ⊕ msg
4.    Send (C, D) to Alice
```

ALICE RECOVERS THE MESSAGE FROM BOB

```
5.    Ka = $C^{\odot expoA}$
6.    compact Ka = Per[Ka]  (mod $2^h$)
7.    msg = D ⊕ H(compact Ka)
```

10. KEP and ELGAMAL cipher protocols

It is necessary to define a protocol allowing for an interchange of information between Alice and Bob asynchronously to achieve the following objectives:

- deferred communications
- check the integrity of the exchanged information
- mutual authentication to avoid attacks from active adversaries (e.g., man-in-the-middle)
- block replay attacks
- availability of the exchanged information
- perfectly defined formats

The following protocol aims to fulfill these requirements.

PROTOCOL: KEP AND CIPHER PUBLIC DATA EXCHANGES

INPUT: any kind of data to be exchanged between entities.
OUTPUT: encapsulated message (msg).
INITIAL CONFIGURATION:
msg: any kind of information to be exchanged between entities.
Universal-Keyed Message Authentication Code (UMAC): here proposed to assure strong symmetric authentication [3, 10].
ID: any elsewhere predefined and sender-receiver shared identification tag.
K: sender-receiver shared key.
Tag: a smart label that can store any sort of information from identification numbers as a brief description for each entity. Here, the tag is Tag = HMAC-SHA3-512 (HM || Nonce). See Fig 1 and more in [3]
HM: $NH_K(msg_1)$ || $NH_K(msg_2)$ || · · · || $NH_K(msg_r)$ || Len; see the NH definition in [3].
Nonce: pseudorandom and unique number that changes with each generated tag.
Timestamp: formatted date and time.

MESSAGE AUTHENTICATION
1. **Acquire** K and msg
2. **Define** a fixed-length Nonce
3. **Generation of** UMAC/SHA3-512/ID/Data
4. **Encapsulate** the concatenation of msg || UMAC/SHA3-512 (HM || Nonce) || Timestamp into a file
5. **Send** the file and Nonce to the receiver

MESSAGE VALIDATION
1. **Acquire** at any time the sent file
2. **Recover** msg and UMAC/SHA3-512 (HM || Nonce)
3. **Verify** integrity and sender's identity using K, Nonce, and msg
4. **Accept** or **Dismiss** msg according to the verification result

11. Semantic security of algorithm 2

Here, we provide an informal view of this aspect based on concepts mostly derived from Bellare's work [1]. In summary, semantical security measures the resistance of any encryption algorithm to attacks using chosen plaintext or ciphertext selected by the attacker, who has access to the encryption and decryption modules working as oracles without knowledge of the key selected for enciphering [1]. The semantic security term is strongly related to other definitions: the one-way functions and the non-malleability of ciphertexts.
Indistinguishability under a chosen plaintext attack (represented as IND-CPA) is equivalent to the property of semantic security and is considered a basic requirement for most provably secure public-key cryptosystems [1]. One-way refers to bidirectional functions that have a probabilistic-polynomial time algorithm that converts domains into codomains, but no such algorithm is known that inverts the procedure. Non-malleability refers to the resistance to slightly modifying the ciphertext to obtain meaningful recovered

plaintext [1]. The next concept to define is the indistinguishability of different ciphertexts of two similar but different plaintexts; an attacker cannot assign a ciphertext of one of them to any one of the plaintexts. This feature is generally presented as a game between a challenger (the algorithm defender) and an adversary (the algorithm attacker) [1]. The challenger generates a key pair PK, SK (public key and secret key, respectively), based on any security parameter k (which can be the key size in bits), and publishes PK to the adversary. The challenger retains SK.

Here, we describe the adaptative version of the game. The adversary may perform any number of encryptions, decryptions, or any other operations. (The adversary is a probabilistic polynomial Turing Machine) [1]. Eventually, the adversary submits two distinct chosen plaintexts m_0 and m_1 to the challenger (of the same length). The challenger selects a bit $b \in \{0, 1\}$ uniformly at random and sends the challenge ciphertext C = E (PK, m_b) back to the adversary. The adversary is free to perform any number of additional computations or decryptions (except C, this step is the adaptative phase of the attack). Finally, its outputs in polynomial time a guess for the value of b [1].

The adversary wins the game if it guesses the bit b, and winning means the algorithm is not indistinguishable and secure; otherwise, the algorithm reaches the strongest available security level: IND-CCA2. (Indistinguishable chosen ciphertext adaptative attack). Formally, a cryptosystem is indistinguishable under an adaptative chosen ciphertext attack if no adversary can win the above game with probability p greater than $1/2+ \in_k$, where $\in_k \leq 1/\Pi^K$ (Π^K arbitrary polynomial function) and $\in_k$ is defined as a negligible function in the security parameter k [1]. For Algorithm 2, we prove the IND-CPA security level and explain how it could be easily adapted to reach the IND-CCA2 security level.

The use of the UMAC function [3] in our Protocol fills this need in such a way that the practical implementation of Algorithms 1 and 2 culminates with the desired provable-security level.

12. A toy numerical example

a. Setup

```
dim=8; rows=dim; cols=2; h=32
```

b. First exchange

$$A1 = \begin{matrix} 9 & 204 \\ 23 & 4 \\ 214 & 49 \\ 126 & 138 \\ 240 & 132 \\ 159 & 236 \\ 61 & 109 \\ 1 & 240 \end{matrix} \quad B1 = \begin{matrix} 67 & 152 & 149 & 3 & 19 & 61 & 159 & 42 \\ 254 & 187 & 152 & 44 & 74 & 225 & 166 & 152 \end{matrix}$$

$$Pa = \begin{matrix} 195 & 92 & 93 & 43 & 163 & 113 & 223 & 154 \\ 253 & 148 & 195 & 245 & 221 & 255 & 225 & 38 \\ 160 & 219 & 166 & 238 & 12 & 15 & 176 & 52 \\ 230 & 158 & 70 & 50 & 62 & 80 & 190 & 156 \\ 200 & 236 & 16 & 128 & 248 & 52 & 168 & 192 \\ 197 & 204 & 171 & 109 & 5 & 79 & 201 & 54 \\ 29 & 215 & 57 & 115 & 9 & 86 & 145 & 186 \\ 99 & 232 & 21 & 67 & 115 & 45 & 63 & 170 \end{matrix}$$

$$A2 = \begin{matrix} 100 & 159 \\ 172 & 11 \\ 117 & 6 \\ 59 & 108 \\ 60 & 215 \\ 192 & 71 \\ 204 & 75 \\ 61 & 155 \end{matrix} \quad B2 = \begin{matrix} 112 & 156 & 30 & 127 & 34 & 159 & 189 & 202 \\ 12 & 232 & 252 & 111 & 75 & 182 & 221 & 213 \end{matrix}$$

$$Pb = \begin{matrix} 52 & 8 & 60 & 141 & 221 & 38 & 23 & 51 \\ 196 & 200 & 252 & 25 & 17 & 166 & 123 & 223 \\ 120 & 188 & 158 & 165 & 76 & 239 & 143 & 80 \\ 224 & 212 & 58 & 25 & 122 & 109 & 203 & 106 \\ 84 & 104 & 172 & 253 & 245 & 30 & 231 & 59 \\ 84 & 88 & 100 & 9 & 77 & 186 & 11 & 147 \\ 196 & 72 & 188 & 185 & 17 & 6 & 91 & 95 \\ 244 & 164 & 186 & 120 & 131 & 21 & 216 & 25 \end{matrix}$$

c. **Second exchange**

$$\text{core (by Alice)} = \begin{matrix} 53 & 218 & 13 & 25 & 238 & 59 & 1 & 17 \\ 141 & 163 & 69 & 64 & 239 & 3 & 54 & 39 \\ 57 & 191 & 140 & 121 & 68 & 167 & 239 & 208 \\ 83 & 183 & 146 & 138 & 176 & 24 & 166 & 202 \\ 45 & 23 & 198 & 51 & 246 & 77 & 87 & 131 \\ 232 & 132 & 138 & 155 & 144 & 19 & 49 & 236 \\ 61 & 36 & 142 & 134 & 253 & 223 & 212 & 204 \\ 115 & 61 & 111 & 187 & 190 & 137 & 48 & 46 \end{matrix}$$

$$\text{core (by Bob)} = \begin{matrix} 53 & 218 & 13 & 25 & 238 & 59 & 1 & 17 \\ 141 & 163 & 69 & 64 & 239 & 3 & 54 & 39 \\ 57 & 191 & 140 & 121 & 68 & 167 & 239 & 208 \\ 83 & 183 & 146 & 138 & 176 & 24 & 166 & 202 \\ 45 & 23 & 198 & 51 & 246 & 77 & 87 & 131 \\ 232 & 132 & 138 & 155 & 144 & 19 & 49 & 236 \\ 61 & 36 & 142 & 134 & 253 & 223 & 212 & 204 \\ 115 & 61 & 111 & 187 & 190 & 137 & 48 & 46 \end{matrix}$$

$\text{expoA} = 2027723037$
$\text{expoB} = 1962405525$

$$U = \begin{matrix} 102 & 220 & 129 & 55 & 136 & 241 & 29 & 74 \\ 52 & 55 & 33 & 115 & 106 & 162 & 101 & 10 \\ 105 & 108 & 31 & 247 & 132 & 21 & 51 & 105 \\ 226 & 246 & 168 & 104 & 93 & 64 & 39 & 225 \\ 101 & 50 & 167 & 135 & 26 & 150 & 236 & 164 \\ 182 & 253 & 34 & 118 & 13 & 36 & 179 & 59 \\ 80 & 198 & 47 & 251 & 166 & 252 & 63 & 104 \\ 104 & 90 & 199 & 189 & 134 & 166 & 228 & 196 \end{matrix}$$

$$V = \begin{matrix} 48 & 178 & 46 & 2 & 54 & 35 & 43 & 255 \\ 1 & 30 & 231 & 89 & 249 & 71 & 188 & 58 \\ 44 & 228 & 56 & 63 & 230 & 132 & 93 & 100 \\ 107 & 27 & 77 & 104 & 144 & 137 & 196 & 215 \\ 155 & 237 & 134 & 215 & 69 & 212 & 249 & 86 \\ 231 & 186 & 190 & 211 & 113 & 221 & 174 & 159 \\ 101 & 11 & 194 & 13 & 233 & 227 & 212 & 74 \\ 159 & 51 & 241 & 233 & 93 & 164 & 90 & 127 \end{matrix}$$

d. **Session keys**

$$Ka = \begin{matrix} 8 & 142 & 81 & 79 & 249 & 129 & 125 & 84 \\ 51 & 137 & 68 & 84 & 161 & 209 & 209 & 30 \\ 107 & 49 & 60 & 164 & 147 & 238 & 81 & 129 \\ 32 & 51 & 28 & 250 & 76 & 1 & 17 & 175 \\ 42 & 220 & 216 & 21 & 210 & 96 & 202 & 249 \\ 224 & 224 & 52 & 153 & 94 & 78 & 7 & 68 \\ 51 & 177 & 26 & 22 & 54 & 248 & 90 & 246 \\ 181 & 51 & 246 & 202 & 188 & 192 & 33 & 35 \end{matrix}$$

$$Kb = \begin{matrix} 8 & 142 & 81 & 79 & 249 & 129 & 125 & 84 \\ 51 & 137 & 68 & 84 & 161 & 209 & 209 & 30 \\ 107 & 49 & 60 & 164 & 147 & 238 & 81 & 129 \\ 32 & 51 & 28 & 250 & 76 & 1 & 17 & 175 \\ 42 & 220 & 216 & 21 & 210 & 96 & 202 & 249 \\ 224 & 224 & 52 & 153 & 94 & 78 & 7 & 68 \\ 51 & 177 & 26 & 22 & 54 & 248 & 90 & 246 \\ 181 & 51 & 246 & 202 & 188 & 192 & 33 & 35 \end{matrix}$$

Compact Ka = 940671506
Compact Kb = 940671506

e. **Bob to Alice enciphered message**

PLAINTEXT msg = ***secret string*** (padded to 512-bita)
CIPHERTEXT C = V =
{{155,172,125,25,19,186,176,176}, {31,113,22,102,164,214,92,213},
{153,152,105,47,28,210,155,66}, {207,172,118,190,115,224,88,155},
{58,186,55,245,127,58,46,212}, {48,138,194,247,234,37,149,115},
{211,254,118,16,89,233,7,37}, {172,205,217,27,204,77,79,219}}
HASH (Kb)=
623f543dd1968404add50ce2a3c10a66a9ab182290414544fd0b
4f5218c9b1612487783a08c423b9c5c71e965ee9eb23ca9fdd95e4
5eee79d0759e6868f89163
CIPHERTEXT D =
{17,90,55,79,180,226,164,119,217,167,101,140,196,225,42,70,137,
139,56,2,176,97,101,100,221,43,111,114,56,233,145,65,4,167,88,26
,40,228,3,153,229,231,62,182,126,201,203,3,234,191,253,181,196,
126,206,89,240,85,190,72,72,216,177,67}

f. **Alice recovers the message**

Compact Ka = 940671506
RECOVERED msg = ***secret string***

13. Discussion

Algorithm 1 has been implemented in different computer languages and shows that extremely high complexity can be easily achieved on a standard processor. The fact that by modifying the input variables (dim, number of rows, columns, iterations), practically any security level can be easily obtained without resorting to multiple precision leads to very fast implementations. Depending upon the computer architecture and software implementation, larger *dim* values can be used for reaching higher complexity levels.
It is particularly important to use the UMAC function in the Protocol because it is similar to Merkle's trees for PQC digital signatures [2] and plays the role of achieving maximal semantic security [1] and simultaneously strengthening its postquantum character.

Note: as Black et al. [3] state, "*the security of UMAC is rigorously proven, in the sense of giving exact and quantitatively strong results which demonstrate an inability to forge*

UMAC-authenticated messages assuming an inability to break the underlying cryptographic primitive. (sic)"

14. **Conclusions**

The algorithms presented in this paper are such that very high complexity can be reached using small primes, normal precision, and small rectangular matrices, leading to very fast computer implementations.

Acknowledgments: to D. R. L. Brown, D. B. Szyld, L. Liberti, N. Gillis, J. Di Mauro, I. Córdoba, S. Barzola, G. Cucatti and Y. Alis for many interesting discussions, theoretical insights and computer implementations.

References

1. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In Annual International Cryptology Conference, Springer, Berlin, Heidelberg, pp. 26-45 (1998)
2. Bernstein D., Lange T.: Post-Quantum Cryptography, Nature, 149,188-194 (2017)
3. Black, J., Halevi, S., Krawczyk, H., Krovetz, T., & Rogaway, P.: UMAC: Fast and secure message authentication, Annual International Cryptology Conference, pp. 216-233, Springer, Berlin, Heidelberg (1999)
4. Brown, D .R. L.: private communication (2022)
5. Daemen, J., Rijmen, V.: AES Proposal: Rijndael, AES algorithm submission, September 3 (1999)
6. Di Mauro, J., Salazar, E. & Scolnik, H.D.: Design and implementation of a novel cryptographically secure pseudorandom number generator. J Cryptogr Eng, 12, 255–265 https://doi.org/10.1007/s13389-022-00297-8 (2022)
7. Diffie, W., Hellman, M.: New directions in cryptography", IEEE Transactions on Information Theory, 22, 6, 644-654 (1976)
8. Ellis, J. H.: The possibility of non-secret digital encryption, CESG Research Report (1970)
9. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes, in M. Wiener (Ed.): CRYPTO'99, LNCS 1666, Springer-Verlag (1999)
10. Internet Engineering Task Force (IETF), UMAC RFC 4418, https://datatracker.ietf.org/doc/rfc4418/ (2006). Accessed 15 September 2022
11. Kanwal, S., Ali, R.: A cryptosystem with noncommutative platform groups. Neural Computing and Applications, 29: 11, 1273-1278 (2018).
12. Lee, G.T.: Abstract Algebra, Springer Undergraduate Mathematics Series, https://doi.org/10.1007/978-3-319-77649-1_3 (2018)
13. Lidl, R., Niederreiter, H.: Introduction to Finite Fields and their Applications, Cambridge University Press, Cambridge (1997)
14. Liu, J., Jia, J., Zhang, H., Yu, R., Yu, Y., Wu, W.: Cryptanalysis of a cryptosystem with non-commutative platform groups. China Communications, 15(2), 67-73 (2018)
15. Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology, M. Naor (Ed.): TCC 2004, LNCS 2951, pp. 21–39, Springer-Verlag, (2004)
16. Menezes, A., Van Oorschot, P., Vanstone, S.: Handbook of applied cryptography. The CRC Press series on discrete mathematics and its applications, CRC-Press (1997)
17. Mullan, C.: Some Results in Group-Based Cryptography, Thesis submitted to the University of London for the Degree of Doctor of Philosophy (2020)
18. Myasnikov, A., Shpilrain, V., Ushakov A.: Non-commutative Cryptography and Complexity of Group-theoretic Problems, Mathematical Surveys and Monographs, AMS Volume 177 (2011)
19. NIST Computer Security Resource Center: Competition for standardization of post-quantum protocols (PQC), https://csrc.nist.gov/projects/post-quantum-cryptography (2017). Accessed 15 September 2022
20. NIST Computer Security Resource Center: Post-Quantum Security, https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-(evaluation-criteria) (2022). Accessed 15 September 2022
21. Rotman, J. J.: Advanced Modern Algebra, vol. 114, American Mathematical Soc. (2010)
22. Stickel, E.: A new method for exchanging secret keys. Proceedings of the Third International Conference on Information Technology and Applications (ICITA05), Contemporary Mathematics, IEEE Computer Society, 2, 426–430 (2005)
23. Virdia F.: private communication (2022)