THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 21, Number 1/2020, pp. 53–60

A QUANTUM SAFE ANALYSIS OF HELIOS VOTING SYSTEM

Mihail-Iulian PLEŞA

Military Technical Academy, Bucharest
Corresponding author: Mihail-Iulian Pleşa, E-mail: mihai.plesa@mta.ro

Abstract. In this paper, we evaluate how quantum computers affect an electronic voting system in general but we also provide a detailed analysis of the Helios voting system. We analyze the two major cryptographic components of Helios that are vulnerable to quantum computers: the encryption scheme and the decryption proof. We investigate the use of Niederreiter cryptosystem as a replacement for the ElGamal encryption scheme and proposed a new decryption proof for this scheme.

Key words: electronic voting, post-quantum cryptography, zero-knowledge proofs.

1. INTRODUCTION

1.1. General considerations

Electronic voting follows the implementation of the electoral process by electronic means. There are three major types of electronic voting system: paper-based, direct-recording (DRE) and public network DRE. Paper-based electronic voting involves using paper for ballots but counting the votes electronically. Direct-recording implies using an electronic machine to register votes instead of paper. The electronic machine is placed inside a poll station. Public network DRE implies a remote voting process: every eligible voter can transmit his vote remotely, over a public network to a central authority. This type of vote is often called online voting.

In an online voting system, the security problems are difficult to address. For example, we cannot guarantee that the device used by the end-user is tampered proof since he uses his personal device. Obviously, to give every registered user a special device to vote will generate enormous costs. The solutions to most of the security problems raised by an online voting system are addressed using cryptographic means. We use encryption in order to hide the data over a public network, digital signatures in order to guarantee the integrity of data, MACs for authenticity and much more. All these cryptographic primitives that are commonly used, are based on hard mathematical problems such as integer factorization [1], discrete logarithm problem [2] or elliptic curve Diffie-Hellman problem [3]. All these problems are in fact instances of the Hidden Subgroup Problem (HSP) [4]. More exactly, they are instances of the HSP problem on finite abelian groups [5]. So far, no classical efficient algorithm has been found to solve these problems. On the other side, on a quantum computer, things are not the same. In 1996, Peter Shor proposed a quantum polynomial-time algorithm for integer factorization and discrete logarithms.

The complexity of Shor's algorithm is $O\left((\log N)^2(\log\log N)(\log\log\log N)\right)$ [6]. This complexity does not reflect the running time but the number of elementary quantum gates required for the algorithm. Given the applicability of this algorithm for integer factorization, Shor's algorithm is exponentially faster than the best classical known factorization algorithm which has the complexity $O\left(\mathrm{e}^{1.9(\log N)^{1/3}(\log\log N)^{2/3}}\right)$. This complexity reflects the number of steps taken by the classical algorithm. So far, we cannot compute the actual running time of Shor's algorithm to compare it with the running time of the classical algorithm. When we will have a quantum computer with a sufficiently large number of qubits, we will be able to compare the real running time. The hypothesis from which we start is that quantum computers will become a threat for current cryptographic primitives. NIST post-quantum initiative is based on the same hypothesis but the truth is that we do not have mathematical proof for this statement. At present, there is no mathematical proof that there is no classical algorithm that would equate the complexity of Shor's quantum algorithm. Although such a proof does not exist at the moment, the progress in the development of quantum computers is very fast: in

2016 IBM had a 5-qubit system and at the end of 2017 they developed a 50-qubit quantum computer [7]. However, until we have a mathematical proof that there are algorithms in the BQP complexity class that are not in the P complexity class, we cannot be absolutely sure of the superiority of quantum computers.

1.2. About this paper and related work

In this paper we explore the problems rise by an online voting system from the perspective of quantum computers. Concretely, we explore Helios, an open source online voting system [8]. In this work, we do not analyze the software vulnerabilities of the Helios system, but how this system is affected by the development of quantum computers. Specifically, each cryptographic primitive used by Helios was analyzed and for all those primitives for which an efficient quantum algorithm is known that would affect their security, another post-quantum primitive was proposed. Our express contribution is a new zero-knowledge decryption proof for the Niederreiter cryptosystem.

There are many papers on the electronic voting system from a quantum perspective but from our knowledge none of these address the specific problem of Helios. Most papers proposed another electronic voting scheme based on quantum computations [9,10,11,12,13]. We have to point out that there is a major difference between post-quantum and quantum cryptography. Post-quantum cryptographic algorithms are primitives that are based on mathematical problems for which no polynomial-time algorithm for solving is known. These algorithms are designed to run on a classical computer (although every procedure that runs on a classical computer may also run on a quantum one). Quantum cryptographic algorithms are those that use quantum effects in order to solve some computational problems. One example of a quantum encryption algorithm is [14].

The paper is structured as follow: in section 2 we consider the architecture and design of the Helios system and see what cryptographic primitives it uses. In section 3 we talk about quantum-resistant versions of those primitives that are not safe and present our proposed proof. Section 4 is left for the conclusions.

2. HELIOS VOTING SYSTEM

In this section, we describe the Helios voting system. The Helios protocol is made from two parts. The first part is ballot preparation. In this step, the voter only completes the ballot and the systems encrypt it and commits to that encryption. In the second step, the voter seals the ballot, meaning that cannot change it later. The flowchart of the Helios protocol is presented in Figure 1. The system interrogates the user on the voting questions. After the user responds to all the questions, the system will encrypt the answers and commits to that encryption by displaying the hash of the ciphertext. In this way, if the ciphertext is altered by some malicious party, the voter will know. The voter may choose to audit the encryption made by the system. In this case, the system presents the user with all the necessary information to re-encrypt the ballot and verify that he obtains the same ciphertext as the system.

2.1. The encryption scheme

Helios system uses the ElGamal encryption scheme [15]. Supposed the Alice wants to send a message m to Bob, $0 \le m \le p-1$, where p is some prime. Let α be a primitive root modulo p and x_B the secret key of Bob. Let $y_B = \alpha^{x_B}$ be the public Diffie-Hellman key of Bob. Alice will generate a random number k uniformly between 0 and $p-1$. Alice will then compute the number $K \equiv y_B^k \bmod p$ and sends to Bob to messages: $c_1 \equiv \alpha^k \bmod p$ and $c_2 \equiv Km \bmod p$. The pair (c_1, c_2) is the ciphertext. Upon receiving, Bob will calculate $K \equiv c_1^{x_B} \bmod p$ and recover m by computing $m \equiv K^{-1} c_2$.

This cryptosystem is based on the discrete logarithm problem, which is known to be vulnerable to Shor's algorithm.

2.2. Voter anonymity in Helios

As we have stated in the previous section, the vote must not be correlated with the voter. Helios was designed to ensure user anonymity using homomorphic encryption or mix-nets.

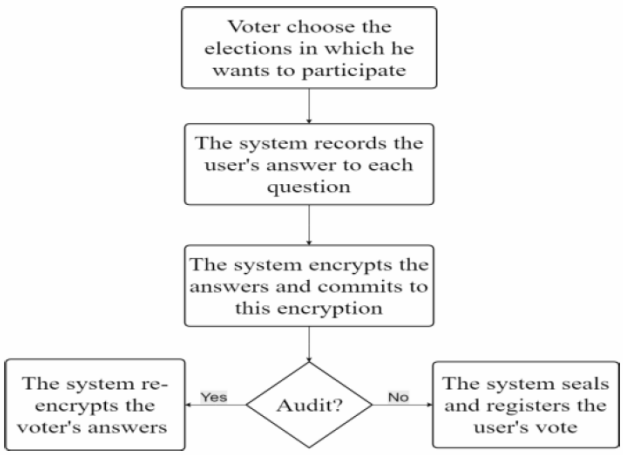

Fig. 1 – Helios voting system.

By using homomorphic encryption, the votes will never be decrypted, and the results of the election will be calculated using homomorphic operations. ElGamal encryption scheme is homomorphic with respect to multiplication. It can be easily checked that $(c_{11} * c_{12}, c_{21} * c_{22})$ is the encryption of $m_1 * m_2$. The scheme can be made homomorphic with respect to addition by constructing a new message m' from the message m as $m' = \alpha^m$ and then encrypting m' with ElGamal. The decryption implies solving the discrete logarithm problem which is hard in general but if the messages are small it can be done using a lookup table [16].

The second option for assuring voters anonymity involved using a mix-net. In this variant, Helios uses a mix-net that will re-encrypt and shuffle all the registered votes. At the end of the election process, the votes are decrypted, and the tally is calculated.

2.3. Voter privacy in Helios

In order to preserve voter privacy, the system does not display the name of the voter or his id on the public board. Rather, what is displayed is the hash of one of this information. As the author says, this is not very good protection because if one knows enough details about the voter, it can deduce the information displayed on the board. In the perspective that in the future, the cryptographic primitives used in Helios will be easily broken, the user may choose to use a voter alias. The user can still be identified in the system, but no data that leads to his identity will be used.

2.4. Auditability in Helios

There are two types of audits in Helios: single ballot and election. In a single ballot audit, the user may choose to audit his encrypted vote before sealing it. If the user chooses to do this, the system will return the randomness used in creating the ciphertext so that the voter can verify that the encryption was correct. The voter may choose to audit the system any number of times before sealing the vote. One important fact is that the system must display the ciphertext and only after that the user will choose whether to audit or seal. In this way, the user may verify that the system is correct. The system may trick the user if it correctly guesses what the voter intention is. If the user chooses to audit, the system must display the correct ciphertext so that the user doesn't realize that he was fooled. If the user chooses to seal the vote, only then the system may display a ciphertext that doesn't encrypt the user choices. Every time when the system has to guess the user intention, it has a $\frac{1}{2}$ probability to correctly guess it. That means, the system has a probability of $\frac{1}{2^n}$ to correctly guess n successive times. This implies that after n audits, the probability that the system is correct is $1-\frac{1}{2^n}$, which is very high for n large enough.

Auditing an election is done via zero-knowledge proof protocols denoted further with ZK. There are three types of ZK used by Helios: ZK range proof, ZK decryption proof, and ZK shuffling proof. The ZK range proof is based on ZK decryption proof, thus we have to analyze only two ZK protocols [17].

2.4.1. ZK shuffling proof

The idea of a mix-net is to randomly permute the encrypted votes so that the correlation between the vote and the voter will be destroyed. In order to do that, the mix-net must use a random permutation. In the case that the mix-net does not output a random permutation, an attacker has more chances to guess what that permutation is and thereupon to find the correlations between the votes and the voters. This is an intuitive explication why the mix-net must output random permutations. In Helios design, the mix-net will also re-encrypt every vote before shuffling. The ZK shuffling will prove that the permutation generated by mix-net was a random permutation without revealing it and that the re-encryption of the vote is valid. That means that the mix-net has re-encrypted the correct encrypted votes and not some other data (for example, other encrypted votes that were not expressed by the voters). In order to create proof that the mix-net has correctly permuted the registered votes, the system will create a secondary mix-net. The user will then query the system to reveal either the secondary permutation with the corresponding encryption factors or the difference between secondary and primary mix-net (the permutation that will transform the secondary mix-net into the primary one). After n such queries, the probability that the primary mix-net is incorrect is 2^{-n}. This kind of proof does not use any problems vulnerable to any quantum attack.

2.4.2. ZK decryption proof

A ZK decryption proof involves a prover (A) who wants to prove to a verifier (B) that a certain ciphertext has been decrypted to a certain plaintext. This kind of ZK proof has a direct application in electronic voting systems that use homomorphic encryption. In Helios, a solution to ensure the anonymity of the vote is the used of the homomorphic property of the ElGamal scheme. After all the votes are collected and the election is over, the system will compute the homomorphic sum of the votes, thus obtaining the encrypted sum of the votes. Let denote this encrypted sum by c. The ciphertext c will be decrypted by the system into the plaintext m, the result of the election. Since the votes are not decrypted separately, the voters have no reason to believe that m is indeed the result of the election. A ZK decryption proof will prove that the decryption of the ciphertext c is the plaintext m without revealing any information about the private key. In Helios, Chaum-Pedersen protocol is used as a ZK-decryption proof [18].

Suppose we have an ElGamal ciphertext (c_1, c_2) and we want to prove that the corresponding plaintext of this ciphertext is m. The notations used here are the same as those from 2.1. To prove that m is the corresponding plaintext for the ciphertext (c_1, c_2) we have to prove the equality of two discrete logarithms, more exactly we have to prove that $\log_{\alpha} y_B = \log_{c_1}(c_2 m^{-1})$. Intuitively, we must prove that the private exponent used in public key y_B is the same as the private exponent used in shared secret K. Chaum-Pedersen protocol goes as follow (as usual, A is the prover and B, the verifier):

1. A generates a random $w \in Z_p$ and computes $Q_1 = \alpha^w$ and $Q_2 = c_1^w$;
2. A sends to B the numbers Q_1 and Q_2;
3. B transmits a challenge $c \in Z_p$;
4. A responds with $t = w + xc$;
5. B verifies that $\alpha^t = Q_1 y_B^c$ and $c_1^t = Q_2 \left(\frac{c_2}{m} \right)^c$.

2.4.3. ZK range proof

The idea of a ZK range proof is to show that given the encryption of some numerical value, this is in some range $0...max$. The necessity for this type of ZK proof comes from the following problem. Suppose that a voter has sealed his votes and submit it to the system. Of course, the vote is encrypted using ElGamal cryptosystem. When the vote is over, the system will compute the homomorphic sum of those ciphertexts and announce the results. To understand the motivation of this type of ZK proof into a voting system, let us consider the following example: suppose in a referendum the only possible answers that a voter can register are 0 or 1. The total number of voters is denoted by N. After all the encrypted votes are registered, the system will compute the homomorphic sum of the votes and decrypt the result. If the decrypted result if

greater or equal than $\frac{N}{2}$ it is considered that the referendum has passed. If one of the voters encrypts, for example, the value $-N$, the referendum will not pass no matter how the participants have voted. The system must have some mechanism to assures itself that the plaintexts corresponding to those ciphertexts that it received are well-formatted (i.e. in some range). The solution that Helios approaches is the following: for a user to prove that he has encrypted value in some range, he will use a ZK-or proof composed of multiple ZK decryption proof. ZK-or proofs can be constructed using Σ-protocol.

To describe what a ZK-or proof is, suppose we have two Σ-protocols, σ_1 and σ_2. σ_1 is proving some claim c_1 and σ_2 is proving some other claim c_2. One of these two claims is true, but the prover does not want to reveal which one. A ZK-or proof is a protocol that allows the prover to do exactly that. For the prover, the fact that one of the claims is false is meaning that for one σ_i, $i \in \{1,2\}$, the prover will not be capable of computing the response after the commitment. The prover needs to compute the commitment value after receiving the challenge. Suppose that for σ_1, the prover (A) knows that the claim is true and for σ_2 he knows that the claim is not true. In order to prove to the verifier (B) that one of these statements is true without revealing which one, it will use the CDS-94 technique [19]:

1. A chooses a random challenge, $ch2$, computes a commitment $com2$ and a response $res2$ such that the sigma protocol $(com2, ch2, res2)$ is valid.

2. A sends to B two commitments: $com1$ and $com2$. $com1$ is the commitment for σ_1 which proves a true claim.

3. B sends a challenge ch.

4. A computes $ch1 = ch' - ch2$ and sends to B $(ch1, com1, res1)$ and $(ch2, com2, res2)$.

5. B verifies that both sigma protocols are valid and that $ch1 + ch2 = ch'$.

In this way, the verifier (B) will know that one of the sigma protocols represents a true claim (since B sends the challenge ch') but he will not know which one it is because he cannot determine which commitment was calculated before the prover receives the challenge. This is a general construction for a ZK-or proof, regardless of what sigma protocols prove and can be easily generalized for proving the "OR" of n Σ-protocols.

Coming back to the original problem, in Helios, the voter must prove that the plaintext value which he encrypted is in some range $0...max$. In order to do that, the voter will compute a ZK-or protocol on $max+1$ ZK-decryption protocols: ZK-decryption(0), ZK-decryption(1)… ZK-decryption(max). If the value encrypted is in the range, $0...max$ then only one of the ZK-decryption proofs will be true and so the verifier will know that the value is in the required range.

3. THE POST-QUANTUM SOLUTION

Now that we have detailed how Helios works, we can see that from a quantum perspective there are two main problems with the system. Both the encryption scheme used, and the ZK-decryption proof are based on the hardness of discrete logarithm problem. Although this is secure given classical computers, the primitive will not be secure in the context of quantum computers as we have detail in the introduction. For the Helios to work, we must find an encryption scheme that is homomorphic and, we must find a ZK-decryption protocol for that scheme. The scheme that is in our opinion the most suitable is the classical McEliece scheme [20]. The scheme has entered in round 2 submissions for NIST post-quantum initiative and has resisted 4 decades of cryptanalysis.

3.1. General McEliece cryptosystem

The McEliece cryptosystem is not bounded to some special type of linear, but rather is a general methodology for constructing a cryptosystem from a linear code.

Let consider a linear $[n,k,d]$ code capable of correcting t errors, G its generator matrix. Let us consider two matrices: a $k \times k$ non-singular matrix S and a $n \times n$ permutation matrix P. Let the matrix $\hat{G}$

be a $k \times n$ matrix defined as $\hat{G} = SGP$. Then, the public key is $\left(t, \hat{G}\right)$ and the private key is (S, G, P). Intuitively, we hide the generator matrix using a scrambling matrix respective a permutation one. To encrypt a message m one calculates $c' = m\hat{G}$. After that, it generates a random vector z of length n and weight t. The ciphertext c is defined as $c = c' + z$. Basically, the message m is encrypted as a codeword altered by some error. To decrypt a message, one can calculate $\hat{c} = cP^{-1}$, thus obtaining $\hat{c} = mSG + e'$, where $e' = eP^{-1}$. Decoding $\hat{c}$ (i.e. eliminating the error) we can find out the message $\hat{m} = mS$. From $\hat{m}$ we can easily calculate the original message m as $m = \hat{m}S^{-1}$.

The idea of the cryptosystem is simple and powerful. Although it is not depending on some linear code, choosing a good code is crucial. The official submission at NIST is using binary Goppa codes [21]. There were other attempts for construction of McEliece cryptosystem using Reed-Solomon codes, but those variants of the cryptosystem were eventually broken [22].

The drawback of this cryptosystem is that it does not allow the construction of a signature scheme. Another security equivalent version of this cryptosystem that permits the construction of signatures is presented in the next section.

3.2. Niederreiter cryptosystem

The Niederreiter cryptosystem is a slightly modified version of McEliece cryptosystem that allows the construction of a signature scheme. This cryptosystem does not use the generator matrix but the parity check matrix H.

Let $y \in F_2^n$ be a message. The syndrome of y is defined as $s = yH^T$. Suppose we have a message $m = x + e$ where x is the actual information and e is the error. The syndrome of m is $mH^T = (x+e)H^T = xH^T + eH^T$. But as we have stated in section 3.1, the syndrome for a codeword (is the codeword before it doesn't have any errors) is 0 so the syndrome of m becomes: $mH^T = eH^T$. So, the syndrome of the message is equal to the syndrome of the error. That means that if we can calculate the syndromes of all possible errors, we can deduce the error itself. More exactly, if one receives a message m with errors, it can calculate the syndrome of it and after looking into a pre-calculated table of error syndromes, it can find the information unaltered by computing $x = m - e$, where e is the error extracted from the table. Of course, calculating the syndromes of all possible values is not possible for long messages. For a n bit message and a code capable of correcting up to t errors we have $2^{C_n^t}$ possible errors. However, if one does have an efficient syndrome decoding algorithm, it can calculate the error without precomputing all possible combinations. That is the idea of Niederreiter cryptosystem. The model is identical with the McEliece, but this time, the message is being encrypted as the syndrome. The decryption is done with an efficient decoding algorithm. As in the case of McEliece cryptosystem, the binary Goppa codes are used. These codes provide an efficient decoding algorithm [33].

Let us consider a binary code $[n, k, d]$ capable of correcting up to t errors. From the generator matrix, G one calculates the parity check matrix H. As in McEliece cryptosystem, after that, it generates $(n-k)\times(n-k)$ non-singular matrix S and a $n \times n$ permutation matrix P. The public key is $\left(H^{\text{pub}} = SHP, t\right)$ and the private key is (S, H, P). To encrypt a message, one calculates $c = H^{\text{pub}} m^T$. For decryption, one does calculate $S^{-1}c = HPm^T$ and apply to this result the syndrome decoding algorithm, so it remains only with Pm^T. From this point, it can easily recover the message by computing $m^T = P^{-1}Pm^T$. For signing a message m, the hash of the message will be calculated and decrypted as it were a ciphertext.

One other fact that needs to be verified is the homomorphism of Niederreiter cryptosystem. Suppose we have two messages m_1 and m_2. Let $c_1 = H^{\text{pub}} m_1^T$, $c_2 = H^{\text{pub}} m_2^T$ and $c_3 = H^{\text{pub}}\left(m_1^T + m_2^T\right)$. Following the encryption algorithm, we have $c_3 = H^{\text{pub}}\left(m_1^T + m_2^T\right) = H^{\text{pub}} m_1^T + H^{\text{pub}} m_2^T = c_1 + c_2$ thus the encryption of the sum of plaintexts is the sum of the ciphertexts corresponding to those plaintexts. We can conclude that the scheme is homomorphic with respect to the operation of addition. Of course, that, in this variant, the Niederreiter cryptosystem is not secure against chosen plaintexts attacks. This can be quickly solved by randomizing the encryption. One solution, for example, is using some padding scheme on m before

encrypting it. Niederreiter can be converted to an IND-CPA or IND-CCA2 scheme using the techniques described in [23] and [24].

We proposed the replacement of ElGamal encryption from the Helios system with Niederreiter cryptosystem. Since this cryptosystem allows a signature scheme, all proofs of private key possession based on a signature can now be made using Niederreiter signature scheme.

3.3. Decryption proof for Niederreiter cryptosystem

If Niederreiter cryptosystem possesses a valid scheme that allows doing ZK-decryption proof, then we can easily make Helios quantum resistant. Morozov and Takagi proposed an interactive Σ-protocol for ZK-decryption proof in [25]. Our scheme is not an interactive one and does not have the structure of a Σ-protocol. As usual, we denote the prover by A and the verifier by B. A wants to prove B that a ciphertext c corresponds to a plaintext m. The idea behind our protocol is to let the verifier check whether the relation $S^{-1}c = HPm^T$ is true without revealing the private matrices S, H and P. The proposed protocol is the following:

1. A will generate a random $(n-k)\times(n-k)$ non-singular matrix X and transmits to B two matrices $M_1 = SX^{-1}$ and $M_2 = XHP$;
2. B will accept the proof only if the following two conditions are met simultaneously:
   i) $M_1 M_2 = H^{\text{pub}}$;
   ii) $M_1^{-1}c = M_2 m^T$.

Let analyze in more detailed the proposed protocol. If c is the corresponding ciphertext for the plaintext m, then we have that $S^{-1}c = HPm^T$. But if this relation is true, then for every matrix X of dimensions $(n-k)\times(n-k)$ the relation $XS^{-1}c = XHPm^T$ is also true. The matrix multiplication is valid from the perspective of dimensions because S has dimension $(n-k)\times(n-k)$ and so does S^{-1}, c has dimension $(n-k)\times 1$ so the product $S^{-1}c$ has dimension $(n-k)\times 1$. Thus, the product $XS^{-1}c$ has dimension $(n-k)\times 1$. In the right term, we have: H has dimension $(n-k)\times n$, P has dimension $n\times n$, thus the product HP has dimension $(n-k)\times n$ and the product HPm^T has dimension $(n-k)\times 1$ hence the product $XHPm^T$ has dimension $(n-k)\times 1$ as the left-hand side. Transmitting the matrices XS^{-1} and XHP to the verifier does not mean that the prover has indeed the private keys.

We claim the verifier cannot find any information about the private key and the prover cannot cheat. This statement is described as more rigorous in propositions 1 and 2, for which we present a sketch of the proof.

PROPOSITION 1. *Given the ciphertext c, the plaintext m, the public key H^{pub} and the matrices M_1 and M_2, the verifier cannot find any information about the matrices S, H or P.*

Sketch of proof. Given an IND-CPA secure variant the Niederreiter, the verifier cannot extract any information about the private key given the ciphertext c and the plaintext m, Given the matrix M_1, for the verifier to find the matrix S he must possess the secret matrix X. If this matrix is randomly generated, the probability that the verifier correctly guesses the matrix X is $2^{-(n-k)^2}$.

PROPOSITION 2. *Given the ciphertext c, the plaintext m, the private key (S,H,P) and the public key H^{pub}, the prover cannot find two matrices M_1 and M_2, such that $M_1 M_2 = H^{pub}$ and $M_1^{-1}c = M_2 m'^T$, where $m' \neq m$.*

Sketch of proof. Suppose that the prover finds two matrices M_1 and M_2 such that $M_1 M_2 = H^{pub}$ and $M_1^{-1}c = M_2 m'^T$. Then $c = M_1 M_2 m'^T$. But $M_1 M_2 = H^{pub}$, thus $c = H^{pub} m'^T$. The problem is reduced to finding a plaintext m' given a ciphertext c. This problem is the well-known general decoding problem of linear codes.

We do not claim that our proofs are rigorous nor that the proposed protocol is secure. The problem of whether this protocol is secure or not remains a cryptanalysis problem that will find its answer in the future.

4. CONCLUSIONS AND FURTHER DIRECTION OF RESEARCH

In this paper, we presented a general overview of electronic voting and a detailed analysis of the Helios system. Although much of the cryptographic primitives are still safe to use, we must consider that quantum computers are becoming a reality. From our perspective, we must slowly turn our attention in rewriting the existing software using post-quantum primitives. Helios is one of that system. In this paper, we have analyzed how Helios depends on the existing cryptographic primitives and with what they can be replaced. We found a post-quantum cryptographic scheme, namely the McEliece cryptosystem that sounds promising. This cryptosystem has resisted over 40 years of cryptanalysis and recently has entered in round 2 of NIST post-quantum challenge. Our express contribution was a decryption proof protocol. A further direction of research will be the construction of a rigorous mathematical proof for our proposed protocol in section 3.4. We choose the Helios system for our analysis because it is a well-known open-source electronic voting system and one of the most impacted things by quantum computers will be our electronic voting systems. It will not impact only our privacy but our way of organizing a democratic society and hence all the other freedoms.

REFERENCES

1. R.L. RIVEST, A. SHAMIR, L. ADLEMAN, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, **21**, *2*, pp. 120-126, 1978.
2. W. DIFFIE, M. HELLMAN, *New directions in cryptography*, IEEE Transactions on Information Theory, **22**, *6*, pp. 644-654, 1976.
3. J. HOFFSTEIN, J. PIPHER, J.H. SILVERMAN, *An introduction to mathematical cryptography*, Springer, New York, 2014.
4. F. WANG, *The hidden subgroup problem*, arXiv preprin, p. 1008.0010, 2010.
5. P.W. SHOR, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM Review, **41**, *2*, pp. 303-332, 1999.
6. D. BECKMAN, A.N. CHARI, S. DEVABHAKTUNI, J. PRESKILL, *Efficient networks for quantum factoring*, Physical Review A, **54**, *2*, p. 1034, 1996.
7. S.J. DEVITT, *Performing quantum computing experiments in the cloud*, Physical Review A, **94**, *3*, p. 032329, 2016.
8. "Helios Voting", available online: https://heliosvoting.org/ (accessed February 5, 2019).
9. R.D. Pino, V. LYUBASHEVSKY, G. NEVEN, G. SEILER, *Practical quantum-safe voting from lattices*, Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security – CCS 17, 2017.
10. N. BAO, N.Y. HALPERN, *Quantum voting and violation of Arrow's impossibility theorem*, Physical Review A, **95**, *6*, p. 062306, 2017.
11. CERN Courier, "Quantum voting", available online: https://cerncourier.com/quantum-voting/ (accessed February 6, 2019).
12. Y. LI, G. ZENG, *Quantum anonymous voting systems based on entangled state*, Optical Review, **15**, *5*, pp. 219-223, 2008.
13. P. XUE, X. ZHANG, *A simple quantum voting scheme with multi-qubit entanglement*, Scientific Reports, **7**, *1*, pp. 1-4, 2017.
14. M.-I. PLESA, T. MIHAI, *A new quantum encryption scheme*, Advanced Journal of Graduate Research, **4**, *1*, pp. 59-67, 2018.
15. T. ELGAMAL, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions on Information Theory, **31**, *4*, pp. 469-472, 1985.
16. B. KACSMAR, S. PLOSKER, R. HENRY, *Computing low-weight discrete logarithms*, International Conference on Selected Areas in Cryptography – SAC 2017, Lecture Notes in Computer Science, **10719**, pp. 106-126, 2017, Springer.
17. I. DAMGARD, *On Σ-protocols*, Lecture Notes, University of Aarhus, Department for Computer Science, 2010, v.2.
18. D. CHAUM, T.P. PEDERSEN, *Wallet databases with observers*, Advances in Cryptology – CRYPTO' 92, Lecture Notes in Computer Science, **740**, pp. 89-105, 1992, Springer.
19. R. CRAMER, I. DAMGÅRD, B. SCHOENMAKERS, *Proofs of partial knowledge and simplified design of witness hiding protocols*, Advances in Cryptology – CRYPTO '94, Lecture Notes in Computer Science, pp. 174-187, 1994, Springer.
20. A. VALENTIJN, *Goppa codes and their use in the McEliece cryptosystems*, BSc Thesis, Syracuse University, US, 2015.
21. K.O. CHUNG, *Goppa codes*, Technical report, Department of Mathematics, Iowa State University, US, December 2004.
22. G. LANDAIS, J.-P. TILLICH, *An efficient attack of a McEliece cryptosystem variant based on convolutional codes*, International Workshop on Post-Quantum Cryptography, Lecture Notes in Computer Science, **839**, pp. 102-117, 2013, Springer.
23. R. NOJIMA, H. IMAI, K. KOBARA, K. MOROZOV, *Semantic security for the McEliece cryptosystem without random oracles*, Designs, Codes and Cryptography, **49**, *1-3*, pp. 289-305, 2008.
24. N. DOTTLING, R. DOWSLEY, J. MULLER-QUADE, A.C.A. NASCIMENTO, *A CCA2 secure variant of the McEliece cryptosystem*, IEEE Transactions on Information Theory, **58**, *10*, pp. 6672-6680, 2012.
25. K. MOROZOV, T. TAKAGI, *Zero-knowledge protocols for the McEliece encryption*, Australasian Conference on Information Security and Privacy, Lecture Notes in Computer Science, **7372**, pp. 180-193, 2012, Springer.

Received August 9, 2019