Code-Based Cryptography in the Post-Quantum Era

Mahmoud Namazi
mnamazi@umail.ucsb.edu

June 17, 2018

Abstract

With the discovery of Shor's algorithm, previous encryption algorithms, such as RSA and Diffie-Hellman, once deemed secure are now in danger of being cracked by quantum computers in the future. This future danger has led to the development of a multitude of different approaches for encryption in the post-quantum era. The class of lattice-based methods are quite popular, but they are not the only post-quantum encryption algorithms. Code-based encryption algorithms, which are based on the theory of error-correcting codes, offer another way to keep information secure in the post-quantum era. In this survey, we discuss the reasons for the development of post-quantum encryption and code-based encryption algorithms (in particular the McEliece cryptosystem).

1 Modern Cryptography and Shor's Algorithm

Modern cryptography appears everywhere, from online banking to keyless cars and instant messaging services, such as Whatsapp. The algorithms utilized in most modern cryptographic systems revolve on being difficult to crack due to one of the following three computationally difficult mathematical problems: integer factorization (RSA), the discrete logarithm problem (ElGamal encryption, Diffie-Hellman key exchange), or the elliptic curve discrete logarithm problem (Elliptic Curve Cryptography). However, currently quantum computers are under development which, using Shor's algorithm, will be able to quickly solve these difficult mathematical problems and therefore breaking many modern cryptographic systems. Shor's algorithm allows for fast integer factorization, while modern computers take exponential time to solve the integer factorization problem, on the order of $O(e^{1.9(logN)^{\frac{1}{3}}(loglogN)^{\frac{2}{3}}})$ [5], quantum computers compute at order $O((logN)^2(loglogN)(logloglogN))$ [2] which is significantly faster.

This will usher in a post-quantum cryptography age which will necessitate the development of new cryptographic systems which rely on different concepts for their security. Already many new systems have been suggested, these include: lattice-based cryptography, multivariate cryptography, hash-based cryptography, elliptic curve isogeny

cryptography, and code-based cryptography. In this paper, I will be introducing code-based cryptography which relies on concepts from error correcting codes and coding theory to create cryptographic systems which are impervious to quantum computers.

2 Linear Codes

In this section, we introduce some basic concepts from error-correcting codes, specifically with regards to linear codes in order to better prepare the reader with regards to code-based cryptography.

The key idea behind error-correcting codes is repetition. The more repetition that exists, the more resiliency to errors the message will have as it moves through a noisy channel. However, at the same time, the greater the repetition, the greater the code size (message) which can have negative impacts in areas such as communication bandwidth.

One type of error-correcting code are linear codes. Linear codes have several properties which include:

- Closure under addition and multiplication
- Generator polynomial, P(X), which can be used for systematic coding of the message
- P(X) can be represented as a k x n generator matrix, G

Three numbers can be used to distinguish a particular linear code, based on its properties. They are N (the size of the code), K (the size of the message), and T (maximum error resiliency, in other words the max number of bit errors that can be correct). The error resiliency can be calculated from the size of the code and size of the message by the following equation.

$$t \leq \lfloor \frac{n-k}{2} \rfloor$$

3 The McEliece Cryptosystem

The McEliece cryptosystem was the first code-based encryption system. It was created in 1978 by Robert McEliece [8]. It is asymmetric, so it involves both a public key and a private key. In the original paper, as well as the many follow-up papers, Goppa codes are used. These are a class of linear codes based on algebraic curves. There is nothing particularly special about these codes, the only requirement is that linear codes which have difficult to break decoding algorithms are used.

The algorithm relies on the three parameters that we can use to distinguish a linear code's properties (N, K, T which are described in the previous section).

3.1 Key Generation

Key generation involves three matrices. A generator matrix G which is based on a particular binary linear code (which relies on parameters N and K, it is capable of correcting T encoding errors). A random n x n permutation matrix, P, and k x n random non-singular matrix, S, are also generated. The public key is given by ($\hat{G}$, T), where $\hat{G}$ is a k x n matrix given by $\hat{G}$ = SGP. The private key consists of the consituent matrices, (S, G, P).

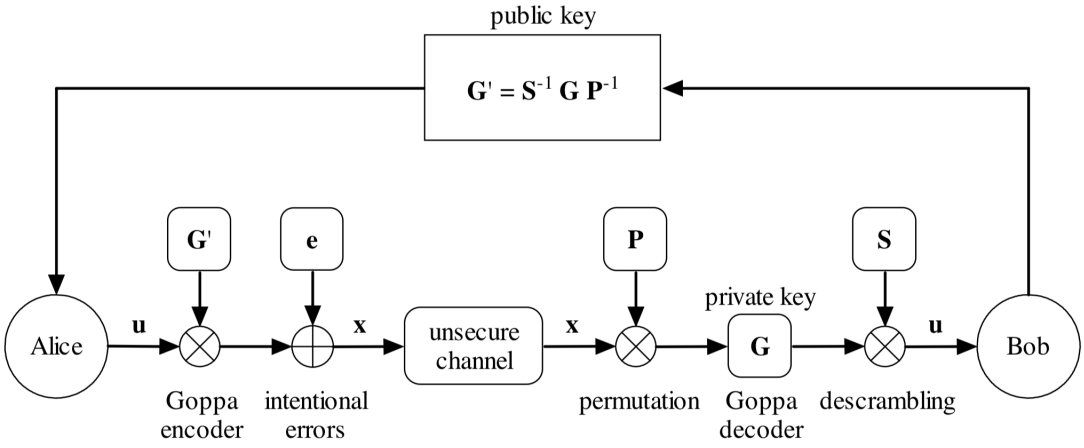

Figure 1: The McEliece cryptosystem, as first published using Goppa linear codes (S^{-1} and P^{-1} used in the public key, simply entails using S and P instead of inverses when decoding). Image taken from [1].

3.2 Encryption

In order to encrypt our message, we take our binary message, m (which is a vector of length k), and multiply it by the public key giving the encrypted message, e' = m$\hat{G}$. Then, we take a random vector of length n, r, with up to T ones, and add it to e' (mod 2), giving the final encrypted message, e = e' + r. The random vector, r, is essentially introducing errors into the encoded message. One thing to consider, however, is that the encrypted message may pick up additional, unwanted errors along the way (due to a noisy channel). Therefore, we may want to pick a random vector with less than T ones in order to account for this.

3.3 Decryption

For decryption, we take the inverse of P, P^{-1}, and compute $\hat{e} = eP^{-1}$. We then use the decoding algorithm to get $\hat{m}$ from $\hat{e}$. This gives $\hat{m} = mS$. To compute the original message, we compute $m = \hat{m} = mSS^{-1}$. Decryption, as further analyzed below, is difficult to break due to the large matrices and the random error vector.

4 Possible Attacks and Issues

As McEliece discusses in his paper [8], the cryptosystem is difficult to break for two reasons (corresponding to the two types of possible attacks). The first type of attack involves recovering G from $\hat{G}$ (the public key). Using a large N and K (McEliece suggested N = 1024 and K = 524, resulting in massive matrices), there are an astronomical number of choices for the matrices, G, S, and P. The attacker would first have to succeed in properly guessing these matrices before they could proceed to decrypt the message. This attack is essentially infeasible.

The second possible attack is to try to decode the message, m, where G is not known and there are T errors. This is the general decoding problem and it is considered NP-complete [3]. The larger the linear code used, the more infeasible this attack is. McEliece does a calculation in his paper using a Goppa code with properties N = 1024, K = 524, and T = 50 and finds that the probability of a random vector successfully decoding the message is $2^{-215.59}$.

McEliece's cryptosystem does have some downsides. The first is that the public key and private key are essentially very large matrices. In [4], the authors analyze the previously mentioned attack on a variant of the McEliece cryptosystem, called the Niederreiter cryptosystem [9]. They propose changes to the algorithm which make it more secure, but allow for smaller linear codes to be used. This reduces the memory and bandwidth load when using this form of encryption, additionally, as memory and communications technology improves, this problem becomes negligible.

5 Hardware Implementations

Some hardware implementation has been done for variants of the McEliece cryptosystem. Two of these works involve FPGA implementations [7, 6], while [10] involves a novel architecture which is tested using an FPGA. There could be space in this area for more novel architectures. Additionally, it would be interesting to look at the ability of GPUs and novel architectures to break McEliece cryptosystems, this would likely not be difficult using the parameters McEliece suggested in 1978 considering the large improvements in computing power. Future architectures will probably have to optimize for power and speed as modern mobile devices involve significant communication, so encryption would have to be fast, but at the same time encryption cannot consume a significant amount of power which is involved in the computation done with large matrices.

References

[1] Marco Baldi, Marco Bianchi, Franco Chiaraluce, Joachim Rosenthal, and Davide Schipani. Enhanced public key security for the mceliece cryptosystem. *Journal of Cryptology*, 29(1):1–27, 2016.

[2] David Beckman, Amalavoyal N Chari, Srikrishna Devabhaktuni, and John Preskill. Efficient networks for quantum factoring. *Physical Review A*, 54(2):1034, 1996.

[3] Elwyn Berlekamp, Robert McEliece, and Henk Van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[4] Daniel J Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the mceliece cryptosystem. In *International Workshop on Post-Quantum Cryptography*, pages 31–46. Springer, 2008.

[5] Don Coppersmith. Modifications to the number field sieve. *Journal of Cryptology*, 6(3):169–180, 1993.

[6] Thomas Eisenbarth, Tim Güneysu, Stefan Heyse, and Christof Paar. Microeliece: Mceliece for embedded devices. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 49–64. Springer, 2009.

[7] Stefan Heyse, Ingo Von Maurich, and Tim Güneysu. Smaller keys for code-based cryptography: Qc-mdpc mceliece implementations on embedded devices. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 273–292. Springer, 2013.

[8] Robert J Mceliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978.

[9] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Prob. Control and Inf. Theory*, 15(2):159–166, 1986.

[10] Abdulhadi Shoufan, Thorsten Wink, Gregor Molter, Sorin Huss, and Falko Strentzke. A novel processor architecture for mceliece cryptosystem and fpga platforms. In *Application-specific Systems, Architectures and Processors, 2009. ASAP 2009. 20th IEEE International Conference on*, pages 98–105. IEEE, 2009.