The McEliece Code-based Public Key Cryptosystem

Palmer Adonis Lao
laopa@clarkson.edu

Sean P. Lyons
lyonssp@clarkson.edu

Abstract

As the world draws closer to the era of quantum computing, there is a fear on the minds of cryptographers and those who depend on the cryptographic technologies in place in everyday life. Shor's algorithm, once it is useful in practice, has the ability to break systems, like RSA and ElGamal, that protect the world's most sensitive data. This paper presents a system developed by Robert McEliece. This code-based, public key cryptosystem, known as the McEliece cryptosystem, is one of few existing classic cryptosystems that cannot be broken by Shor's algorithm or other extensions of the Hidden Subgroup Problem. In our paper we will present some necessary exposition on linear codes, explanation of the hardness of decoding a linear code, and the details of the cryptosystem itself.

1 Error-correcting Codes

Common data transmissions are ridden with potential for error. These errors can take shape as a result of all kinds of physical imperfections in the channel over which data is being sent. Regardless, we don't seem to have any issues sending text messages over immense distances or receiving those messages when impurities effect them. This is a result of **error-correcting codes**. A common method of providing some facility to correct transmission errors is redundancy. As an example, suppose I want to send a single bit, 0. Now suppose as a way of correcting any error, I simply append two extra copies of the string to itself to get the message 000. Any corruption that occurs when I send this bit string across a channel will reflect on the string as flipped bits. Because the original message has been tripled, it is clear that 100 is not a valid bit-string, because there is no string that you can triple-duplicate to get 100. By the same token, 110 is also an invalid string. If the receiver sees behavior like this, they know there was an error in transmission. It is clear from the example that our error-correcting scheme can only **detect** up to 2 errors. That is, if all three bits get flipped due to transmission error, the receiver would be unaware because 111 is a valid bit string.

Now let's turn to the issue of **correcting** the error. If a single bit is flipped, rendering the bit string 100 on the receiving end, can the error be corrected? The answer is yes. If we assume that there has only been one transmission error, then we know that the original string only consisted of the bits that are the majority bits in the corrupted string. However, there is an issue here on the receiving end. How does the receiver know that the original string was not 111 and the received string does not reflect 2 transmission errors? The error-correcting scheme described is said to only **correct** 1 error, because once there is a chance that a second error may have occured, there is no way to recover the original message. A couple of the properties hidden in the example above actually generalize to some important properties of linear codes of arbitrary length. Let us state some key definitions before proceeding.

Definition 1.1. A **linear code** of length n and rank k is a linear subspace, C, or $\mathbb{F}_q^n$, where $\mathbb{F}_q$ is the finite field on q elements.