Code-based Key Encapsulation from McEliece’s Cryptosystem

Edoardo Persichetti
Florida Atlantic University

Abstract

In this paper we show that it is possible to extend the framework of [10] and create a secure KEM based on the McEliece protocol. This provides greater flexibility in the application of coding theory as a basis for cryptographic purposes.

I. INTRODUCTION

A Hybrid Encryption scheme is a cryptographic protocol that uses public-key encryption as means to securely exchange a key, while delegating the task of encrypting the body of the message to a symmetric scheme.

The public-key component is known as Key Encapsulation Mechanism (KEM). The first code-based KEM was presented by Persichetti in [10] and utilized the Niederreiter framework [8]. In this paper, we expand on Persichetti’s work and prove that if we use the McEliece approach [6] we are still able to obtain a secure KEM. This is a novel construction, with a great potential impact, especially considering NIST’s recent call for papers for secure post-quantum primitives [1].

II. PRELIMINARIES

A. The McEliece Cryptosystem

We present here a more “modern” version compared to R. J. McEliece’s original cryptosystem [6]. In the following description (Table I), we consider families of codes to which is possible to associate an efficient decoding algorithm; we denote this with \( \text{Decode}_\Delta \), where \( \Delta \) is a description of the selected code that depends on the specific family considered. For instance, in the case of binary Goppa codes, the associated algorithm is Patterson’s algorithm [9] and \( \Delta \) is given by a Goppa polynomial \( g(x) \) and its support \( (\alpha_1, \ldots, \alpha_n) \). For MDPC codes [7], decoding is given by Gallager’s bit-flipping algorithm [5] and \( \Delta \) is a sparse parity-check matrix \( H \) for the code. Also, we denote with \( \mathbb{W}_{q,n,w} \) the set of words of \( \mathbb{F}_q^n \) with Hamming weight \( w \).

The security of the scheme follows from the two following computational assumptions.

Assumption 1 (Indistinguishability) The \( k \times n \) matrix \( G \) output by \( \text{KeyGen} \) is computationally indistinguishable from a uniformly chosen matrix of the same size.

Assumption 2 (Decoding Hardness) Let \( G \) be a generator matrix for an \( [n, k] \) linear code \( C \) over \( \mathbb{F}_q \) and \( y \) a word of \( \mathbb{F}_q^n \). It is hard to find a codeword \( c \in C \) with \( d(c, y) \leq w \).

Table I: The McEliece cryptosystem.

| Setup | Fix public system parameters \( q, n, k, w \in \mathbb{N} \), then choose a family \( \mathcal{F} \) of \( w \)-error-correcting \([n, k]\) linear codes over \( \mathbb{F}_q \). |
|---|---|
| \( \mathcal{K} \) | \( \mathcal{K}_{\text{publ}} \) the set of \( k \times n \) matrices over \( \mathbb{F}_q \). |
| \( \mathcal{K}_{\text{priv}} \) | the set of code descriptions for \( \mathcal{F} \). |
| \( \mathcal{P} \) | The vector space \( \mathbb{F}_q^n \). |
| \( \mathcal{C} \) | The vector space \( \mathbb{F}_q^n \). |
| \( \text{KeyGen} \) | Generate at random a code \( C \in \mathcal{F} \) given by its code description \( \Delta \) and compute a public \( \mathcal{G} \) generator matrix \( G \). Publish the public key \( G \in \mathcal{K}_{\text{publ}} \) and store the private key \( \Delta \in \mathcal{K}_{\text{priv}} \). |
| \( \text{Enc} \) | On input a public key \( G \in \mathcal{K}_{\text{publ}} \) and a plaintext \( \phi = x \in \mathcal{P} \), choose a random error vector \( e \in \mathbb{W}_{q,n,w} \), then compute \( y = xG + e \) and return the ciphertext \( \psi = y \in \mathcal{C} \). |
| \( \text{Dec} \) | On input the private key \( \Delta \in \mathcal{K}_{\text{priv}} \) and a ciphertext \( \psi \in \mathcal{C} \), compute \( \text{Decode}_\Delta(\psi) \). If the decoding succeeds, return its output \( \phi = x \). Otherwise, output \( \perp \). |
Assumption \[2\] is also known as the General Decoding Problem (GDP), which was proved to be NP-complete in \[2\], and it is believed to be hard on average, and not just on the worst-case instances (see for example Sendrier \[11\]).

**B. Encapsulation Mechanisms and the Hybrid Framework**

A Key Encapsulation Mechanism (KEM) is essentially a Public-Key Encryption scheme (PKE), with the exception that the encryption algorithm takes no input apart from the public key, and returns a pair \((K, \psi_0)\). The string \(K\) has fixed length \(\ell_K\), specified by the KEM, and \(\psi_0\) is an “encryption” of \(K\) in the sense that \(\text{Dec}_{sk}(\psi_0) = K\). The key \(K\) produced by the KEM is then passed on to a Data Encapsulation Mechanism (DEM), which is in charge of encrypting the actual message. The formulation of a DEM, that normally comprises additional tools for security such as Message Authentication Codes (MAC), is outside the scope of this paper, and we refer the reader to \[4\] for more details.

Formally, a KEM consists of three algorithms, KeyGen, Enc, Dec, which we present in Table \[II\]. A KEM is required to be sound for at least all but a negligible portion of public key/private key pairs, that is, if \(\text{Enc}_{pk}(\ ) = (K, \psi_0)\) then \(\text{Dec}_{sk}(\psi_0) = K\) with overwhelming probability.

**Table II: Key Encapsulation Mechanism.**

| Algorithm | Description |
|-----------|-------------|
| KeyGen    | A probabilistic key generation algorithm that takes as input a security parameter \(\lambda\) and outputs a public key \(pk\) and a private key \(sk\). |
| Enc       | A probabilistic encryption algorithm that receives as input a public key \(pk\) and returns a key/ciphertext pair \((K, \psi_0)\). |
| Dec       | A deterministic decryption algorithm that receives as input a private key \(sk\) and a ciphertext \(\psi_0\) and outputs either a key \(K\) or the failure symbol \(\bot\). |

The security notions for a KEM are similar to the corresponding ones for PKE schemes. The one we are mainly interested in (representing the highest level of security) is presented below.

**Definition 1** The adaptive Chosen-Ciphertext Attack game for a KEM proceeds as follows:

1) Query a key generation oracle to obtain a public key \(pk\).
2) Make a sequence of calls to a decryption oracle, submitting any string \(\psi_0\) of the proper length. The oracle will respond with \(\text{Dec}_{sk}(\psi_0)\).
3) Query an encryption oracle. The oracle runs \(\text{Enc}_{pk}^{KEM}\) to generate a pair \((\tilde{K}, \tilde{\psi}_0)\), then chooses a random \(b \in \{0, 1\}\)
   and replies with the “challenge” ciphertext \((K^*, \psi_0)\) where \(K^* = \tilde{K}\) if \(b = 1\) or \(K^*\) is a random string of length \(\ell_K\) otherwise.
4) Keep performing decryption queries. If the submitted ciphertext is \(\psi_0^*\), the oracle will return \(\bot\).
5) Output \(b^* \in \{0, 1\}\).

The adversary succeeds if \(b^* = b\). More precisely, we define the advantage of \(A\) against KEM as

\[
\text{Adv}_{KEM}(A, \lambda) = \left| \Pr[b^* = b] - \frac{1}{2} \right|.
\]

We say that a KEM is secure if the advantage \(\text{Adv}_{KEM}\) of any polynomial-time adversary \(A\) in the above CCA attack model is negligible.

It has then been proved that, given a CCA adversary \(A\) for the hybrid scheme (HY), there exist an adversary \(A_1\) for KEM and an adversary \(A_2\) for DEM running in roughly the same time as \(A\), such that for any choice of the security parameter \(\lambda\) we have \(\text{Adv}_{HY}(A, \lambda) \leq \text{Adv}_{KEM}^*(A_1, \lambda) + \text{Adv}_{DEM}(A_2, \lambda)\). See Cramer and Shoup \[4\] Th. 5] for a complete proof.

\[1\]While the original version proposes to use scrambling matrices \(S\) and \(P\) (see \[8\] for details), this is not necessary and alternative methods can be used, depending on the chosen code family.
C. Other Cryptographic Tools

In this section we introduce another cryptographic tool that we need for our construction.

Definition 2 A Key Derivation Function (KDF) is a function that takes as input a string $x$ of arbitrary length and an integer $\ell \geq 0$ and outputs a bit string of length $\ell$.

A KDF is modelled as a random oracle, and it satisfies the entropy smoothing property, that is, if $x$ is chosen at random from a high entropy distribution, the output of KDF should be computationally indistinguishable from a random length-$\ell$ bit string.

Intuitively, a good choice for a KDF could be a hash function with a variable (arbitrary) length output, such as the new SHA-3, Keccak [3].

III. The New KEM Construction

The KEM we present here follows closely the McEliece framework, and is thus based on the hardness of SDP. Note that, compared to the original Niederreiter scheme, a slight modification is introduced in the decryption process. As we will see later, this is necessary for the proof of security.

Table III: The McEliece KEM.

| Setup | Fix public system parameters $q, n, k, w \in \mathbb{N}$, then choose a family $F$ of $w$-error-correcting $[n, k]$ linear codes over $\mathbb{F}_q$. |
|-------|----------------------------------------------------------------------------------------------------------------------------------|
| KeyGen | Choose a code $C \in F$ given by its code description $\Delta$ and compute a generator matrix $G$. Publish the public key $G$ and store the private key $\Delta$. |
| Enc   | On input a public key $G$ choose random words $x \in \mathbb{F}_q^n$ and $e \in \mathbb{W}_{q,n,w}$, then compute $K = \text{KDF}(x|e, \ell_K)$, $\psi_0 = xG + e$ and return the key/ciphertext pair $(K, \psi_0)$. |
| Dec   | On input a private key $\Delta$ and a ciphertext $\psi_0$, compute $\text{Decode}_\Delta(\psi_0)$. If the decoding succeeds, use its output $(x, e)$ to compute $K = \text{KDF}(x|e, \ell_K)$. Otherwise, choose a random permutation $\pi$, then set $K = \text{KDF}(\pi(\psi_0), \ell_K)$. Return $K$. |

If the ciphertext is correctly formed, decoding will always succeed, hence the KEM is perfectly sound. Furthermore, it is possible to show that, even if with this formulation $\text{Dec}^{\text{KEM}}$ never fails, there is no integrity loss in the hybrid encryption scheme thanks to the check given by the MAC.

We prove the security of the KEM in the following theorem.

Theorem 1 Let $A$ be an adversary in the random oracle model for the Niederreiter KEM as in Definition II. Let $\theta$ be the running time of $A$, $n_{\text{KDF}}$ and $n_{\text{Dec}}$ be two bounds on, respectively, the total number of random oracle queries and the total number of decryption queries performed by $A$, and set $N = n_{\text{KDF}} \cdot |\mathbb{W}_{q,n,w}|$. Then there exists an adversary $A'$ for SDP such that $\text{Adv}^{\text{KEM}}(A, \lambda) \leq \text{Adv}^{\text{SDP}}(A', \lambda) + n_{\text{Dec}}/N$. The running time of $A'$ will be approximately equal to $\theta$ plus the cost of $n_{\text{KDF}}$ matrix-vector multiplications and some table lookups.

Proof We replace KDF with a random oracle $\mathcal{H}$ mapping elements of the form $(x, e) \in \mathbb{F}_q^k \times \mathbb{W}_{q,n,w}$ to bit strings of length $\ell_K$. To prove our claim, we proceed as follows. Let’s call $G_0$ the original attack game played by $A$, and $S_0$ the event that $A$ succeeds in game $G_0$. We define a new game $G_1$ which is identical to $G_0$ except that the game is halted if the challenge ciphertext $\psi_0 = x^*G + e^*$ obtained when querying the encryption oracle had been previously submitted to the decryption oracle: we call this event $F_1$. Since the number of valid ciphertexts is $N$, we have $\text{Pr}[F_1] \leq n_{\text{Dec}}/N$. It follows that $\text{Pr}[S_0] - \text{Pr}[S_1] \leq n_{\text{Dec}}/N$, where $S_1$ is the event that $A$ succeeds in game $G_1$. Next, we define game $G_2$ which is identical to $G_1$ except that we generate the challenge ciphertext $\psi_0$ at the beginning of the game, and we halt if $A$ ever queries $\mathcal{H}$ at $(x^*|e^*)$: we call this event $F_2$. By construction, since $H(x^*|e^*)$ is undefined, it is not possible to tell whether $K^* = K$, thus we have $\text{Pr}[S_2] = 1/2$, where $S_2$ is the event that $A$ succeeds in game $G_2$. We obtain that $\left|\text{Pr}[S_1] - \text{Pr}[S_2]\right| \leq \text{Pr}[F_2]$ and we just need to bound $\text{Pr}[F_2]$.

We now construct an adversary $A'$ against GDP. $A'$ interacts with $A$ and is able to simulate the random oracle and the decryption oracle with the help of two tables $T_1$ and $T_2$, initially empty, as described below.
Key Generation: On input the instance \((G, y^*, w)\) of GDP, return the public key \(\text{pk} = G\).

Challenge queries: When \(A\) asks for the challenge ciphertext:
1) Generate a random string \(K^*\) of length \(\ell_K\).
2) Set \(\psi_0^* = y^*\).
3) Return the pair \((K^*, \psi_0^*)\).

Random oracle queries: Upon \(A\)’s random oracle query \((x, e) \in \mathbb{F}_q^k \times \mathbb{W}_{q,n,w}\):
1) Look up \((x, e)\) in \(T_1\). If \((x, e, y, K)\) is in \(T_1\) for some \(y\) and \(K\), return \(K\).
2) Compute \(y = xG + e\).
3) If \(y = y^*\) then \(A’\) outputs \(c = xG\) and the game ends.
4) Look up \(y\) in \(T_2\). If \((y, K)\) is in \(T_2\) for some \(K\) (i.e. the decryption oracle has been evaluated at \(y\)), return \(K\).
5) Set \(K\) to be a random string of length \(\ell_K\) and place \((x, e, y, K)\) in table \(T_1\).
6) Return \(K\).

Decryption queries: Upon \(A’\)’s decryption query \(y \in \mathbb{F}_q^n\):
1) Look up \(y\) in \(T_2\). If \((y, K)\) is in \(T_2\) for some \(K\), return \(K\).
2) Look up \(y\) in \(T_1\). If \((x, e, y, K)\) is in \(T_1\) for some \(x, e\) and \(K\) (i.e. the random oracle has been evaluated at \((x, e)\) such that \(y = xG + e\)), return \(K\).
3) Generate a random string \(K\) of length \(\ell_K\) and place the pair \((y, K)\) in \(T_2\).
4) Return \(K\).

Note that, in both random oracle and decryption queries, we added the initial steps to guarantee the integrity of the simulation, that is, if the same value is queried more than once, the same output is returned. A fundamental issue is that it is impossible for the simulator to determine if a word is decodable or not. If the decryption algorithm returned \(\perp\) if and only if a word was not decodable, then it would be impossible to simulate decryption properly. We have resolved this problem by insisting that the KEM decryption algorithm always outputs a hash value. With this formulation, the simulation is flawless and \(A’\) outputs a solution to the GDP instance with probability equal to \(\Pr[\mathbb{F}_2]\). \(\square\)

IV. Conclusions

In this paper, we have introduced a key encapsulation method based on the McEliece cryptosystem. This novel approach enjoys a simple construction and a tight security proof as for the case of the Niederreiter KEM presented in [10]. We believe that our new construction will offer an important alternative while designing quantum-secure cryptographic primitives.

REFERENCES

[1] National Institute of Standards and Technology: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process (call for proposal) (December 2016). http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/call-for-proposals-final-dec-2016.pdf.
[2] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. IEEE Transactions on Information Theory, 24(3):384 – 386, May 1978.
[3] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. http://keccak.noekeon.org/.
[4] R. Cramer and V. Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. SIAM J. Comput., 33(1):167–226, January 2004.
[5] R. G. Gallager. Low-Density Parity-Check Codes. M.I.T. Press, 1963.
[6] R. McEliece. A Public-Key Cryptosystem Based on Algebraic Coding Theory. Technical report, NASA, 1978.
[7] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. MDPC-McEliece : New McEliece Variants from Moderate Density Parity-Check Codes. In Cryptology ePrint Archive, Report 2012/409, 2012.
[8] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. Problems of Control and Information Theory, 15(2):159–166, 1986.
[9] N. Patterson. The algebraic decoding of Goppa codes. IEEE Transactions on Information Theory, 21(2):203 – 207, March 1975.
[10] Edoardo Persichetti. Secure and Anonymous Hybrid Encryption from Coding Theory. In Philippe Gaborit, editor, Post-Quantum Cryptography: 5th International Workshop, PQCrypto 2013, Limoges, France, June 4-7, 2013. Proceedings, pages 174–187, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
[11] N. Sendrier. The tightness of security reductions in code-based cryptography. IEEE Information Theory Workshop (ITW), pages 415–419, October 2011.