Generic Homomorphic Encryption and Its Application to Code-Based Cryptosystems (Draft)

Matthew Curtin[1] and Kirill Morozov[1]

[1]University of North Texas, USA

April 27, 2022

Abstract

Homomorphic encryption realizes computation on encrypted data, without performing decryption. For some cryptosystems, homomorphic properties seem to be difficult to achieve. One important example is the code-based McEliece cryptosystem (and its dual counterpart, the Niederreiter cryptosystem). These systems recently gained prominence as a basis of the Classic McEliece proposal—the third-round finalist of the NIST Post-Quantum Cryptography standardization project. In this work, we explore the options of arming the above code-based cryptosystems with (partial) homomorphic properties using a generic approach to homomorphic encryption. Specifically, we show how to construct a cryptosystem, which supports some group operation—e.g., addition or multiplication—hence achieving partially homomorphic property in a restrictive setting. This approach may be of independent interest when applied to other cryptographic systems and protocols.

Keywords: Homomorphic encryption, code-based cryptosystem, KEM/DEM paradigm

1 Introduction

Homomorphic encryption realizes computation on encrypted data, without performing decryption. We refer the interested reader to an excellent survey on this topic [5]. For some cryptosystems, homomorphic properties seem to be difficult to achieve. One important example is the code-based McEliece cryptosystem [6] (and its dual counterpart, the Niederreiter cryptosystem [7]). These systems recently gained prominence as a basis of the Classic McEliece proposal [2]—the third-round finalist of the NIST Post-Quantum Cryptography standardization project. In this work, we explore the options of arming the above code-based cryptosystems with (partial) homomorphic properties using a generic approach to homomorphic encryption. Specifically, we show how to construct a cryptosystem, which supports some group operation—e.g., addition or multiplication—hence achieving partially homomorphic property in a restrictive setting. This approach may be of independent interest when applied to other cryptographic systems and protocols.

2 Preliminaries

Let us introduce some notation first. Bitwise exclusive-or is represented as $x + y$ for $x, y \in \mathbb{F}_2^n$. The expression $(x|y) \in \mathbb{F}_2^{k_0+k_1}$ represents the concatenation of two vectors $x \in \mathbb{F}_2^{k_0}$ and $y \in \mathbb{F}_2^{k_1}$. We denote the Hamming weight of a vector $x \in^n$ as $|x|$. Let W_n^t define a subset of vectors in ^{n} of the Hamming weight t, that is $W_n^t := \{x \in : |x| = t\}$. We denote by $x \leftarrow_R X$ a uniformly random sampling of an element x from its domain X.

2.1 Randomized Niederreiter PKE

The Randomized Niederreiter PKE [9] is a variant of the Niederreiter PKE [7] with random padding of a plaintext, which provides IND-CPA security. In this subsection, we present a modified variant of the former PKE, where the plaintext is concatenated to the error vector as is, rather than being encoded as a low-weight vector. This modification immediately enables homomorphic computation in the setting described in Sec. 4.1. At the same time, it severely limits the plaintext length. We consider this to be acceptable, since the setting described in 4.1 is a pure proof-of-concept. Also, it is easy to see that such the modification preserves the IND-CPA security of the construction. It is worth noting that we will use a somewhat high-level and old-fashioned description style in this subsection, which suits the purpose of this presentation. For a more modern and detailed presentation of code-based PKEs, we refer the reader to [2].

The Randomized Niederreiter PKE consists of a triplet of PPT algorithms (Gen, Enc, Dec) defined as follows.

- Key generation algorithm Gen works as follows:
  1. Generate $H \in \mathbb{F}^{(n-k)\times n}$ which is a parity-check matrix of an irreducible binary Goppa code correcting up to t errors. Denote by $Decode$ the respective decoding algorithm.
  2. Generate a random non-singular matrix $S \in \mathbb{F}^{(n-k)\times(n-k)}$.
  3. Generate a random permutation matrix $P \in \mathbb{F}^{n\times n}$.
  4. Let $H^{pub} = SH'P$ and output $pk = (H^{pub}, t, n_1)$, where n_1 is the plaintext length, and $sk = (S, H, P)$.
- Encryption algorithm Enc takes a plaintext $m \in^{n'}$ and pk as input and outputs a ciphertext $c = H^{pub}e^T$, where $e = (e_0|m)$, $e_0 \leftarrow_R W^{t-n'}_{n-n'}$.
- Decryption algorithm Dec, given a ciphertext c and secret key sk, works as follows:
  1. Compute $S^{-1}c = H'e^T$, where S^{-1} denotes the inverse matrix of S.
  2. Compute $Pe^T = Decode(S^{-1}c)$.
  3. Compute $e^T = P^{-1}(Pe^T)$ and output m.

3 Generic Homomorphic Encryption

In a nutshell, our proposed construction is as follows: use the KEM/DEM paradigm for encryption, and for evaluation, keep the KEM components (that is prepend them to the ciphertext) and perform the computation on the DEM components. In particular, for simplicity, we may consider the DEM component to be the one-time pad (over a certain group), and the KEM component to be an encryption of the one-time key using some public-key encryption scheme (PKE).

In order to clarify our proposal, let us consider the following specific example. Let (pk, sk) be a key pair and denote the encryption and decryption algorithms of some PKE scheme as $Enc(pk, \cdot)$ and $Dec(sk, \cdot)$, respectively. As a DEM, we will use the one-time pad over $\mathbb{Z}_n$ for some integer n.

When encrypting a plaintext $m \in \mathbb{Z}_n$, we generate k as a uniformly random element of $\mathbb{Z}_n$ and compute $c = (Enc(pk, k), m + k) = (c', c'')$. All the operations on DEM components are performed over $\mathbb{Z}_n$. Decryption is performed in a straightforward manner: decrypt the KEM component(s), and subtract the resulting key(s) from the DEM part.

For homomorphic evaluation, given ciphertexts $(c'_1, c''_1,)$ and $(c'_2, c''_2,)$, compute the resulting ciphertext as $(c'_1, c'_2, c''_1 + c''_2)$. Note that we have $c''_1 + c''_2 = m_1 + m_2 + k_1 + k_2$. It is easy to see that the decryption algorithm devised in the previous paragraph results in a correct decryption.

Also, it is easy to check that performing homomorphic addition with more ciphertexts preserves the general structure ($\{KEM\ components\ list\}, \{DEM\ component\}$), so that consistency of decryption is preserved as well.

We can imagine that the above construction could appear in previous works whether explicitly or implicitly, but the authors are not currently aware of its formal treatment in the existing literature.[1]

Discussion. We note that the proposed scheme is not *strongly homomorphic*, not *compact* and generally not *circuit private*—see [5] for discussion on these notions. Despite these substantial disadvantages, we would like to argue that the proposed approach is non-trivial and that it may be useful for some applications. First, we observe that a well-known Rothblum's argument [10] does not apply to our construction. It states that a trivial construction may collect ciphertexts, decrypt them, and then perform computation on the decrypted results. Note that in that case, the receiver learns the respective plaintexts, however this does not happen in our construction. Specifically, in our example shown above, the receiver learns only a sum of the respective pliantexts $m' + m''$ but not their individual values—the same way as it would happen in the case of conventional homomorphic encryption.

In order to address a potential circuit privacy issue, let us consider the following toy protocol for semi-honest voting. Encode a yes/no vote as 0/1 (an integer). Each party uses our proposed scheme to encrypt their vote on the committee's public key. An aggregator receives the ciphertexts, sums them together and passes the result to the committee. If the scheme is CPA secure (discussed next), the aggregator will not learn the individual votes, and so will not the committee. Note that the distribution of the ciphertexts (specifically, the number of KEM components) will be known in advance by all parties in both stages, and hence it will not pose a security issue.

Security. We conjecture that if the KEM components are IND-CPA secure and the DEM component is unconditionally secure (as the one-time pad), then our generic homomorphic scheme is IND-CPA secure. The intuition is that we can depart from the standard IND-CPA security result for the KEM/DEM paradigm, and extend it via the hybrid argument for an arbitrary number of KEM components. We state this result in the form of a conjecture because we do not have the formal written proof at the moment. At the same time, we foresee no technical difficulties for constructing such the proof.

Generalizations. The KEM component may also be implemented using a symmetric cipher. This way, any IND-CPA cipher can be made partially homomorphic in this setting. The DEM component can be implemented as a one-time pad over an arbitrary abelian group, hence we can use, e.g., summation or multiplication but not both. An open question is to use both of these operations at the same time, hence extending this construction to fully homomorphic encryption.

4 Code-Based Homomorphic Encryption

Let us apply our approach to code-based cryptosystems. We will limit our focus to the Niderreiter PKE, because it is actually used in the current version of the Classic McEliece proposal [2]. At the same time, we note that our results extend to the McEliece PKE as well.

For simplicity, let us select the parameter set mceliece348864 from [2] which corresponds to 128-bit classical security (Category 1 in the NIST terminology). We have the ciphertext size of 128 bytes, and the plaintext size of 32 bytes. Since [2] implements an IND-CCA2 secure KEM in the random oracle model (ROM), it suffices for our construction. See [**?**] for more detailed discussion on its security and ways to avoid ROM.

[1]At the time of submission, as of April 16, 2022.

The above parameters give us a sense of scaling for our proposal. Consider the toy voting protocol from the previous section. For the case of 10 parties, we will obtain the final encryption of the tally with 10 KEM components, which will take $10 \cdot 128 = 1280$ bytes in total, plus a DEM component of 32 bytes, resulting in the ciphertext size of 1312 bytes.

We have implemented our proposed construction and published the results in Github [3].

4.1 Compact Additively Homomorphic Niederreiter Encryption

For comparison, we considered an alternative (straightforward) approach to additively homomorphic Niederreiter encryption. The idea is to use error vectors of lower weight as compared to the standard scheme. This approach results in a *compact* scheme. However, the number of additions to be handled by the scheme needs to be known in advance. Moreover, this approach very quickly results in impractically large parameters when the number of additions is growing.

Let us present this approach through the following toy example: we construct a Niedderreiter scheme which supports one addition (bitwise XOR) of 3-bit plaintexts. In fact, this is the example used in our proof-of-concept implementation [3].

Let $m_1, m_2 \in^{n'}$ be the plaintexts. In practice, we anticipate n' to be a small number up to about 5. According to the description in Section 2.1, we will have the respective ciphertexts $c_i = H^{pub} e_i^T$, where $e = (e_{0,i} | m_i)$ and $e_{0,i} \leftarrow_R W_{n-n'}^{t-n'}$, for $i = 1, 2$. Differently from Section 2.1, we will set $|e_{0,i}| = t/2 - n'$ for $i = 1, 2$. Note that $c_i = H^{pub} e_i^T = H_e^{pub} e_0^T +_m m_i^T$, where $H_e^{pub} \in \mathbb{F}^{n-k \times n-n'}$ and $H_e^{pub} \in \mathbb{F}^{n-k \times n'}$ are the submatrices corresponding to the random error vector and the plaintext, respectively. Note that both terms in this sum are $(n-k)$-bit vectors.

Now, by bitwise XORing these ciphertexts, we will have

$$c = c_1 + c_2 = H^{pub} e_1^T + H^{pub} e_2^T = H^{pub} (e_{0,1} | m_1)^T + H^{pub} (e_{0,2} | m_2)^T = H^{pub} (e_{0,1} + e_{0,2} | m_1 + m_2)^T.$$

It is easy to see that the decryption will work correctly because $|e_{0,1} + e_{0,2} | m_1 + m_2| \leq t$ by construction.

We may argue that this design will be secure, as long as the parameters $(n, k, t/2 - n')$ provide for a secure Niederreiter encryption.

Discussion. We may extend the above idea in a straightforward manner in order to allow more additions by reducing the initial weight of the error vector. Since the security of the Niederreiter PKE depends on this weight in a crucial manner, this approach appears to be extremely "anti-asymptotic".

5 Implementation

We implemented both compact and non-compact variants of partially homomorphic Niederreiter encryption in C [3].

We used the Classic McEliece round 3 submission package to NIST [2] as our Niederreiter implementation. Although it is referred to as "McEliece", it actually the Niederreiter's dual approach. The submission package consists of a reference implementation and an optimized implementation. We used the optimized implementation for better performance. The NIST submission package has several implementations with different parameter sets. These sets control the (m, n) and t parameters, where m is the degree of a field extension. For simplicity, we pick the minimal parameters for our proof-of-concept implementation. The NIST submission uses Keccak shake256 function for hashing, so we used the eXtended Keccak Code Package [4] for this function.

The NIST submission package includes a Known Answer Test program to demonstrate the KEM functionality. We modified this file, makefile, build, and run files as a start to our implementation. The KAT program's methods for pseudorandom number generation, public and private key generation, and printing byte strings were used in our implementation.

Hardware Specifications. The implementations were developed and tested on a Intel 2.30GHz Xeon Gold 6140 CPU. The operating system was Ubuntu version 18.04. The timings were measured using these specifications.

5.1 Non-Compact Implementation

For the non-compact implementation, we directly used the Classic McEliece KEM [2] and binary one-time pad as DEM. The plaintext size is 3 bits.

Timing. We measured the timing of the preceding implementation by timing how long encryption, decryption, and homomorphic computation took on average over 100 trials. Encryption here consisted of encapsulating a key using the KEM and encrypting a random message using one time pad with the generated symmetric key. Computation consisted of performing one time pad on two ciphertexts and appending their encapsulated symmetric keys to a list. Decryption consisted of de-encapsulating all symmetric keys in the list and performing one time pad with them and the ciphertext.

	CPU Cycles	Time (milliseconds)
Encryption	459	0.460
Decryption	52879	52.879
Computation	17	0.017

Note, the timing for decryption in the above table is for a ciphertext with one encapsulated key. A ciphertext with multiple computations on it will have multiple encapsulated keys. Thus, the timing for a ciphertext to be decrypted grows linearly in regards to how many computations there are. If decryption for a ciphertext with one computation on it takes 50,000 CPU cycles on average, then a decryption for a ciphertext with 10 computations on it will take 500,000 CPU cycles on average.

5.2 Compact Implementation

The compact implementation uses variable error vector weight in order to achieve homomorphic encryption. The compact implementation uses the mceliece6688128 parameter set as opposed to mceliece348864 [2]. The mceliece6688128 because it is expected to provide a reasonable security level for $t' = t/2$. Specifically, we use the crude bound $-log_2(1 - k/n)t$ to provide an estimate for a workfactor of the best attack against the parameter set (m, n, k, t) [1]. Note that we take $k = n - mt$ as in [2].

Parameter Set	m	n	t	k	$-log_2(1 - k/n)t$	$t' = \lfloor t/2 \rfloor$	k'	$-log_2(1 - k'/n)t'$
mceliece348864	12	3488	64	2720	139	32	3104	101
mceliece460896	13	4608	96	3360	180	48	3984	138
mceliece6688128	13	6688	128	5024	256	64	5856	192
mceliece6960119	13	6960	119	5413	258	59	6193	187
mceliece8192128	13	8192	128	6528	294	64	7360	211

5.2.1 Timing

Timing for the compact approach was measured over 100 trials. Each trial consisted of:

- The encryption of 3-bit plaintexts m_1, m_2 using the compact approach described in Sec. 4.1, resulting in c_1, c_2.

- The decryption of c_1, c_2.
- The summation $c = c_1 + c_2$.
- The decryption of c, resulting in $m = m_1 + m_2$.

Encryption and decryption of the original ciphertexts, and decryption of the evaluated ciphertext were timed over the 100 trials. The average time each stage took can be seen in the table below.

	CPU Cycles	Time (milliseconds)
Encryption	243	0.243
Decryption (original ciphertext)	625075	625.075
Decryption (evaluated ciphertext)	6251	6.251

It is interesting to note that decryption of the original ciphertexts takes about 10 times more than that of the evaluated ciphertexts. Currently, we explain it by suboptimality of the error weight parameter $t/2$. Further investigation will be made on this point.

References

[1] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, Paolo Santini: A Finite Regime Analysis of Information Set Decoding Algorithms. Algorithms 12(10): 209 (2019)

[2] Classic McEliece. NIST Post-Quantum Cryptography Project Submission `https://classic.mceliece.org/`

[3] Mattew Curtin. Github project: Partially-Homomorphic-McEliece. `https://github.com/MattCurtin12/Partially-Homomorphic-McEliece/`

[4] Keccak Team. eXtended Keccak Code Package `https://github.com/XKCP/XKCP`

[5] Halevi, S. (2017). Homomorphic Encryption. In: Lindell, Y. (eds) Tutorials on the Foundations of Cryptography. Information Security and Cryptography. Springer, Cham.

[6] McEliece, R.J.: A Public-Key Cryptosystem Based on Algebraic Coding Theory. Deep Space Network Progress Report (1978)

[7] Niederreiter, H.: Knapsack-type Cryptosystems and Algebraic Coding Theory. Problems of Control and Information Theory, vol. 15, no. 2, pp. 159-166, Russian Academy of Sciences (1986)

[8] NIST Post-Quantum Cryptography Standardization Project. `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`

[9] Nojima, R., Imai, H., Kobara, K., Morozov, K.: Semantic Security for the McEliece Cryptosystem without Random Oracles. Designs Codes and Cryptography, vol. 49, no. 1-3, pp. 289–305 (2008)

[10] Ron Rothblum: Homomorphic Encryption: From Private-Key to Public-Key. TCC 2011: 219-234