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

Matthew Curtin and Kirill Morozov

April 16, 2022

1 Introduction

Homomorphic encryption realizes computation on encrypted data, without performing decryption. We refer the interested reader to an excellent survey on this topic [2]. For some cryptosystems, homomorphic properties seem to be difficult to achieve. One important example is the code-based McEliece cryptosystem [3] (and its dual counterpart, the Niederreiter cryptosystem [4]). These systems recently gained prominence as a basis of the Classic McEliece proposal [1]—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 Generic Homomorphic Encryption

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

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

For encrypting a plaintext $m \in \mathbb{Z}_n$, we generate $k \leftarrow_R \mathbb{Z}_n$ (that is 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 evaluation, perform the following. Given $(c'_1, c''_1,)$ and $(c'_2, c''_2,)$, compute $(c'_1, c'_2, c''_1 + c''_2)$. It is easy to check that the decryption algorithm devised in the previous paragraph results in correct decryption. It is easy to see that performing homomorphic addition with more ciphertexts preserves the general structure $(\{KEM\ components\ list\}, \{DEM component\})$, and hence consistency of decryption is preserved as well.

It is possible that above mentioned construction appears in previous works as a folklore concept, but the authors are not currently aware of its formal treatment in the existing literature.[1]

2.1 Discussion

First of all, we note that the proposed scheme is not *strongly homomorphic*, *non-compact* and not *circuit private* (see [2] for the formal discussion). Despite having these substantial disadvantages, we

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

would like to argue that the proposed approach is non-trivial and may be useful for some applications. For the lack of space, we will only provide some specific examples and will deter a detailed discussion to the full version of this paper. A well-known Rothblum's argument [5] states that a trivial construction may collect ciphertexts, decrypt them, and then performs computation on the decrypted results. Note that in this case, the receiver learns the respective plaintexts. However, in our construction, this is not the case: the receiver will obtain $c_1'' + c_2'' = m_1 + m_2 + k_1 + k_2$ for $m_1, m_2 \leftarrow_R \mathbb{Z}_n$ and $k_1, k_2 \leftarrow_R \mathbb{Z}_n$, from which they would learn $m_1 + m_2$ after decrypting the KEM componenets—as it would happen in the case of conventional homomorphic encryption.

A clear issue with our construction is that the ciphertext size is growing with computation, and in the worst case, this growth is exponential in the number of operations. However, a small number of operations will be easily handled by our design.

Another problem is that the ciphertext size will reveal the number of operations performed on the ciphertexts (in the general case). Here, we would like to note that for some applications this issue will not matter. As a specific example, let us consider the following semi-honest voting protocol. Encode a no vote as 0, a yes vote as 1. The number of parties is less than n. Each party uses our 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 (we will discuss this topic later), the aggregator does not learn the individual votes, and so does not the committee. Note that the distribution of the ciphertexts (specifically, the number of KEM components) will be known in advance by all parties, and hence it will not pose a security issue.

3 Code-Based Homomorphic Encryption

We consider two approaches for achieving code-based homomorphic encryption. The non-compact approach involves the ciphertext growing in size with each computation. This approach allows for infinite computations without loss of information, but requires more space. The compact approach involves noise in the ciphertext increasing with each computation. Ciphertext will not grow in size, but the number of computations is limited due to the growing number of errors per each computation. The costs and benefits to each approach will be explored in the following sections.

References

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

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

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

[4] 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)

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

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