Implementation of (Somewhat) Fully Homomorphic Encryption on a Field Programmable Gate Array

Joseph Kirk

September 5, 2017

1 Motivation

As more and more organisations turn to cloud-based computing for their business needs, sensitive data sees more opportunity for breaches in privacy than ever before. For example, computation providers allow other companies to make use of their hardware remotely for more intense computations than the companies would otherwise be able to perform; however, any data computed on must be provided in an unencrypted form, disallowing the use of this service on private data. Homomorphic encryption aims to allow computation on encrypted data, with computations performed on ciphertext being reflected in the plaintext, thereby allowing for computation providers to work with sensitive data without being able to decrypt it. Furthermore, homomorphic cryptosystems allow for the evaluation of encrypted functions on encrypted data, useful where algorithms themselves may be considered sensitive (e.g. in MRI machines).

Although the theoretical usefulness of fully homomorphic encryption is high, the practicality of it is less so. Computations on ciphertext require exponentially more time than the counterpart operation on plaintext, making real-time computation with FHE impossible until advances are made in increasing performance. This proposal considers the various approaches to making a practical implementation of homomorphic encryption a reality.

2 Background

In order to explore the background of fully homomorphic encryption, the terms somewhat homomorphic and partially homomorphic encryption must be established. The distinction is as follows:

Partially homomorphic encryption refers to encryption schemes with homomorphic properties in either addition or multiplication. Without homomorphism in both operations, these encryption schemes are severely limited in terms of functions they are able to evaluate.

Somewhat homomorphic encryption refers to encryption schemes with homomorphic properties in both addition and multiplication, but with error vectors that increase with each subsequent addition or multiplication. This limits the scheme to evaluating only simple functions, as too large an error vector prevents accurate decryption.

Fully homomorphic encryption expands upon somewhat homomorphic encryption by introducing methods of reducing error vector blow-up. By controlling for the errors in somewhat homomorphic encryption schemes, fully homomorphic encryption allows for the evaluation of arbitrarily complex functions.

The first fully homomorphic encryption scheme was proposed by Craig Gentry.[1] The scheme used a polynomial ring and ideal lattices in order to establish homomorphism in both addition and multiplication, and introduced the concept of *bootstrapping* to prevent the error vector from blowing up. Bootstrapping allows for the scheme's Eval function, which evaluates arbitrary circuits on a given ciphertext homomorphically, to evaluate its own decryption circuit; in doing so, it allows for a given ciphertext to be *recrypted* into a ciphertext with a smaller error vector. `[Deepayan: Sorry for the lacking explanation - I don't fully understand how this works yet.]`

As of yet, however, bootstrapping is considered prohibitive. The original scheme, as implemented by Gentry and Halevi, is stated by the authors to be practically secure with 2^{13} or 2^{15} dimensions; the recryption process for a scheme with these dimensions takes 2.8 min or 31 min, respectively. Furthermore, the respective public key sizes for the schemes of these sizes are 284MB and 2.25GB, respectively.[2] Clearly such a system is far from practical in lightweight or time-sensitive operations. For this reason, research into homomorphic encryption has divided into two branches with similar goals yet different approaches. Some research attempts to implement bootstrapping in ever more efficient and lightweight forms, while other research focuses instead on improving the operation limits of somewhat homomorphic encryption schemes in other ways, such as optimising operations to reduce the increase in noise.

The original scheme proposed by Gentry is based upon ideal lattices; this gives it post-quantum security but increases the complexity of using the system to prohibitive levels for non-experts. Improvements have been made to the ideal lattice FHE scheme by Gentry and others, such as speeding up the `KeyGen` function[2] or reducing the size of the keyspace without sacrificing security.[3] An additional three FHE scheme archetypes are found in research:

Over Integers FHE may be regarded as the simplest scheme to understand from a non-expert perspective. First proposed by Van Dijk et al.,[4] it encrypts a message as follows:

$$E(m) = m + 2r + pq; \tag{1}$$

where p and q are large primes, and r is a random number much smaller than p. In this form, p is a symmetric secret key, and c may be decrypted as:

$$D(c) = (c \bmod p) \bmod 2. \tag{2}$$

It may be plainly seen from the above equations that the scheme is bounded. In order to deal with this, Van Dijk et al. introduced bootstrapping, much in the same manner as Gentry.

References

[1] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on theory of computing*, ser. STOC '09. ACM, May 2009, pp. 169–178.

[2] C. Gentry and S. Halevi, "Implementing gentry's fully-homomorphic encryption scheme." in *EUROCRYPT*, vol. 6632. Springer, pp. 129–148.

[3] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes." Springer.

[4] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2010, pp. 24–43.