Practical Homomorphic Encryption: A Survey

Ciara Moore, Máire O'Neill, Elizabeth O'Sullivan
Centre for Secure Information Technologies (CSIT)
Queen's University Belfast
Northern Ireland
{cmoore50, maire.oneill, e.osullivan}@qub.ac.uk

Yarkın Doröz, Berk Sunar
Vernam Lab
Worcester Polytechnic Institute
Worcester, MA, USA
ydoroz@wpi.edu, sunar@wpi.edu

***Abstract*—Cloud computing technology has rapidly evolved over the last decade, offering an alternative way to store and work with large amounts of data. However data security remains an important issue particularly when using a public cloud service provider. The recent area of homomorphic cryptography allows computation on encrypted data, which would allow users to ensure data privacy on the cloud and increase the potential market for cloud computing. A significant amount of research on homomorphic cryptography appeared in the literature over the last few years; yet the performance of existing implementations of encryption schemes remains unsuitable for real time applications. One way this limitation is being addressed is through the use of graphics processing units (GPUs) and field programmable gate arrays (FPGAs) for implementations of homomorphic encryption schemes. This review presents the current state of the art in this promising new area of research and highlights the interesting remaining open problems.**

I. Introduction

Cloud computing offers many services to users, including the option to offload storage and computation of large amounts of data to the cloud. However to take advantage of cloud computing, users must trust and share their data with the cloud service providers. One way to ensure data privacy is to encrypt data before uploading to the cloud. However, if users wish to compute on the data, the data must be downloaded and decrypted, making the main advantages of using cloud services redundant. One solution for providing secure cloud computing on untrusted public clouds is the use of homomorphic encryption: a method of encryption which allows computations on encrypted data, without the need to fully decrypt the data on the cloud.

Partially homomorphic encryption schemes have been known for many years, offering the ability to carry out a certain type of operation on ciphertexts without decryption, for example addition *or* multiplication, such as the additively homomorphic Paillier [1] or the multiplicatively homomorphic ElGamal [2] cryptosystems. However the most commonly used computations, for example in the areas of signal processing, statistics or finance, include both the use of addition and multiplication and thus partially homomorphic schemes do not suffice.

The idea of a fully homomorphic encryption (FHE) scheme was first proposed by Rivest, Adleman and Dertouzos in 1978 [3] under the alternative title of privacy homomorphism and it has remained an open problem until 2009. The concept of homomorphic encryption (HE) has been revisited in recent years, due to the seminal Stanford PhD thesis [4] produced by Craig Gentry in 2009, in which he introduced the first plausible fully homomorphic encryption scheme. A FHE scheme is an encryption scheme which allows the efficient evaluation of an arbitrary depth circuit (composed of additions *and* multiplications) to be evaluated directly on encrypted data. Gentry provided a blueprint for constructing an FHE from a so-called somewhat homomorphic encryption (SHE) scheme, an encryption scheme which allows the evaluation of a *limited* depth circuit. From its introduction in 2009, three main branches of homomorphic encryption schemes have developed: lattice-based, integer-based and learning-with-errors (LWE) or ring-learning-with-errors (RLWE) based encryption.

Research into SHE and FHE could ultimately lead to very exciting developments in a wide variety of areas. The main application of FHE would be to enable secure storage and computation on the cloud. This would serve to further the use and development of cloud-based computing. The technique of homomorphic encryption also lends itself to other important cryptographic applications such as multi-party computation (MPC) [5], [6].

In spite of this potential, current homomorphic encryption schemes are still not efficient enough for real time applications, for example, key generation in Gentry and Halevi's lattice based scheme [7] takes from 2.5 seconds to 2.2 hours. Moreover, a recent implementation required 36 hours for a homomorphic evaluation of AES using FHE by Gentry *et al.* [8]. Another important limitation of FHE schemes is memory usage; very large ciphertext and public key sizes are required to guarantee adequate security to prevent against possible lattice-based attacks. Gentry and Halevi's FHE scheme uses public key sizes ranging from 17 MB to 2.25 GB [7]. Also in the previously mentioned evaluation of AES, memory is reported to be the main limiting factor in the implementation [8]. A further limitation to FHE schemes is the costly bootstrapping operation; bootstrapping is required to reduce and manage the noise that is incurred when homomorphic operations are carried out on ciphertexts. Recent research has therefore focused on optimisations to improve efficiency of the FHE schemes, such as minimising the need for the expensive bootstrapping operation [9] or employing batching techniques for multiple bit encryption [10]–[12]. Moreover

978-1-4799-3432-4/14/$31.00 ©2014 IEEE

the development of optimised FHE architectures targeting alternative platforms to software are being explored, such as for GPU technology [13] or FPGA technology [14]–[18] to try and increase the performance capabilities of these schemes.

II. Current Approaches

Initial research on FHE concentrated on lattice based schemes [7], [19], [20], involving relatively large public key sizes and ciphertext sizes. Lattice based FHE schemes, and indeed the other FHE schemes, base their security on hard problems associated with lattices, such as the sparse subset sum problem (SSSP) or the shortest vector problem (SVP). Single instruction, multiple data (SIMD) techniques were proposed by Smart and Vercauteren [10] for these schemes to perform tasks in parallel and thus improve efficiency.

The main focus of the theoretical cryptographic research community is currently on LWE and RLWE based FHE [21]–[23]. LWE was introduced by Regev [24], and has been shown to be as hard as the worst case lattice problems. This problem has been extended to work over rings [25], and this extension increases the efficiency of LWE. An analysis into the practicality of SHE schemes along with an implementation of a SHE scheme by Brakerski and Vaikuntanathan [26] was carried out by Lauter *et al.* [27].

Integer based schemes were introduced by van Dijk *et al.* [28] as a theoretically simpler alternative to lattice based schemes and have been further developed to offer similar performance to existing lattice based schemes [29], [30]. Moreover a proposed public key compression technique has reduced the size of the public key from over 2 GB down to a manageable 10 MB [29]. The performance of these integer based schemes has very recently been improved through a proposed batching technique [12] for the encryption of multiple plaintext bits into one ciphertext. The performance of evaluating AES using this batched scheme over the integers is comparable to the alternative implementation by Gentry *et al.* using the RLWE based FHE scheme [8].

Since the introduction of SHE and FHE in 2009 [4] there have been several optimisations in the field of homomorphic encryption to improve efficiency and speed. As mentioned in the introduction, bootstrapping is an expensive operation used to reduce the noise in the ciphertext by homomorphically decrypting a ciphertext using a secret key that is encrypted and hidden in the public key. There has been a lot of research into improving or avoiding the use of bootstrapping, such as the introduction of modulus switching by Brakerski *et al.* [9], which helps to minimise the noise generated when carrying out homomorphic operations.

There have been several recent software implementations of FHE schemes, such as [8], [12], and an open source software implementation, `hcrypt`, of FHE is available online [31]. The latest online introduction of a homomorphic encryption library (`HElib`), developed by Halevi and Shoup, features an optimised implementation of the FHE scheme proposed by Brakerski *et al.* [9]. It improves the performance of schemes, offering a speed up by a factor of 12 over previous implementations. It takes 3 hours instead of 36 hours for the homomorphic AES scheme [32], [33].

III. GPU and FPGA Implementations of Homomorphic Encryption Schemes

Research into the development of optimised FHE architectures targeting alternative platforms to software for implementations could prove useful in increasing the efficiency of SHE and FHE schemes. There are currently several research groups working towards this goal and recent developments in this area indicate the potential of hardware architectures to drastically increase the efficiency of homomorphic encryption schemes. In particular, the performance of the underlying crypto-primitives required in many of the FHE schemes, such as modular reduction and large multiplication, could be significantly improved through the use of GPU or FPGA technology. FPGA technology offers flexibility at a low cost, in comparison with application specific integrated circuit (ASIC) designs. In addition, modern FPGAs include embedded hardware blocks, which are optimised for multiply-accumulate (MAC) operations, and thus can be exploited when implementing large multiplications.

There has been some research already conducted into hardware implementations of LWE schemes. Hardware building blocks for the LWE cryptosystem were considered by Göttert *et al.* along with the use of the Fast Fourier Transform (FFT) [36]. An efficient hardware implementation of RLWE encryption is presented by Pöppelmann and Güneysu [37], as proposed by Lyubashevsky *et al.* [25] and Lindner and Peikert [25], is presented by Pöppelmann and Güneysu [37], which can fit on a low cost Xilinx Spartan-6 FPGA, and thus has relatively small resource usage. It compares favourably with the previously mentioned implementation of LWE by Göttert *et al.* in terms of hardware resource utilisation.

The recent work on hardware and GPU architectures for large-integer multiplication has improved the efficiency of FHE schemes. A hardware design of polynomial multiplication using FFT targeting lattice-based SHE and FHE schemes and implemented on FPGA was presented by Pöppelmann and Güneysu [38]. The performance of this multiplication operation implemented on a Spartan-6 FPGA is compared to the same operation with similar parameter sizes in the software implementation of a RLWE SHE scheme by Lauter *et al.* [27] and offers a speed up factor of 39 [38].

The first GPU implementation of a FHE scheme was presented by Wang *et al.* [13]. The authors implemented the small parameter size version of Gentry and Halevi's lattice-based FHE scheme [7] on an NVIDIA C2050 GPU using the FFT algorithm, achieving speed up factors of 7.68, 7.4 and 6.59 for encryption, decryption and the recryption operations, respectively. The FFT algorithm was used to target the bottleneck of this lattice-based scheme, namely the modular multiplication of very large numbers, and the authors took advantage of the highly parallelised GPU to accelerate the performance of the FHE scheme. However, the authors state that even with the speed up the implemented FHE scheme remains

TABLE I
OVERVIEW OF FHE IMPLEMENTATIONS: TIMINGS FOR MULT AND SMALL SIZE ENCRYPT

Design	Scheme	Platform	Mult.	Encrypt
FFT multiplier [34]	Gentry & Halevi's FHE scheme [7]	90nm TSMC	7.74 ms	2.09s
FFT multiplier / reduction [13]	Gentry & Halevi's FHE scheme [7]	NVIDIA C250 GPU	0.765 ms	1.69s
Optimised FFT multiplier / reduction [35]	Gentry & Halevi's FHE scheme [7]	NVIDIA GTX 690	0.583 ms	0.0062s
FFT multiplier [18]	Gentry & Halevi's FHE scheme [7]	Stratix V FPGA	0.125 ms	-
FFT multiplier / reduction [17]	Coron *et al.'s* FHE schemes [29], [30]	Xilinx Virtex7 FPGA	-	0.0130s

unpractical, as high latency is incurred in the encryption and recryption steps. An extension of the authors' work [35] involves the modification of arithmetic operations to decrease costly back and forth FFT conversions. This modified method, implemented on an NVIDIA GTX 690, achieves speed up factors of 174, 7.6 and 13.5 for encryption, decryption and the recryption operations, respectively, when compared to results of the implementation of Gentry and Halevi's FHE scheme [7] that runs on an Intel Core i7 3770K machine. A further FPGA implementation targeting large number multiplication was proposed by Wang and Huang [18]. The authors propose an architecture for a 768K-bit FFT multiplier using a 64K-point finite field FFT as the key component. This component was implemented on both a Stratix V FPGA and a NVIDA Tesla C2050 GPU and the implementation was around twice as fast on the FPGA as on the GPU [13].

Doröz *et al.* presented a full custom hardware implementation of very large integer multiplication [34]. Their design is based on Schönhage-Strassen's Number Theoretic Transform algorithm and includes a 768-KByte cache to decrease I/O transactions. A multiplication takes 7.74 ms for million-bit operands at 666 MHz using a TSMC 90 nm library with an area requirement of 26.7 million equivalent gates. The authors also mention that the performance estimates of Gentry and Halevi's FHE primitive operations match with the previously reported software implementations on a high end Intel Xeon processor [7], and only uses a tiny fraction of the area.

Furthermore, the authors also present the first custom hardware realisation of the Gentry-Halevi FHE scheme [39]. The design introduces new hardware blocks to the previously introduced multiplier to perform all the Gentry-Halevi FHE primitives. The primitives are implemented by using a similar approach as Wang *et al.* [35] to reduce the number of FFT conversions. The design achieves speed up factors of 1.24, 99.44 and 10.32 for decryption, encryption and recryption operations respectively, when compared to the software implementation by Gentry and Halevi [7]. In comparison with the results of the GPU implementation by Wang *et al.* [13], the custom hardware design achieves speed up factors of 0.15, 12.15 and 1.35 for encryption, decryption and the recryption operations, respectively. The authors also state that they achieve these speed up factors utilising a small area of 30 million equivalent gates whereas the NVIDIA Tesla C2050 GPU and the Intel Xeon processor contains 900 million and 205 million equivalent gates respectively.

Other current work has also focused on the use of large integer multiplications required in many FHE schemes [7], [29], [30], in order to maximise the potential speed up factor for hardware implementations. The use of a Comba multiplier [40], which can be implemented efficiently using the DSP slices of an FPGA [41], was proposed by Moore *et al.* [16] to improve the performance of integer based FHE schemes [28], [30]. An FPGA implementation of a RLWE SHE scheme was also targeted by Cousins *et al.* [14], [15], in which Matlab Simulink is used to design the FHE primitives.

Lastly, Cao *et al.* [17] proposed a large-integer multiplier using integer-FFT multipliers combined with Barrett reduction to target the multiplication and modular reduction bottlenecks featured in many FHE schemes. The encryption step in the proposed integer based FHE schemes by Coron *et al.* [29], [30] were designed and implemented on a Xilinx Virtex-7 FPGA. The synthesis results show speed up factors of over 40 are achieved compared to existing software implementations of this encryption step [17]. This speed up illustrates that further research into hardware implementations could greatly improve the performance of these FHE schemes. An overview of the above mentioned multipliers for different platforms is given in Table I.

IV. CONCLUSIONS AND FUTURE DIRECTIONS

Although there has been a lot of recent research in the area of homomorphic cryptography, there are many remaining open problems. In terms of the theoretical research, parameter selection for homomorphic encryption schemes is currently a complex process, as each scheme has specifically selected parameters, all of which are interlinked and these parameters are usually selected based on current possible lattice based attacks and their existing limits. An example of a weakness in this approach to parameter selection was exploited in an attack by Lee [42], where the parameter selection in Gentry and Halevi's scheme [7] was not conservative enough to prevent a lattice based attack exploiting the sparse subset sum problem. More research into parameter selection is needed to ensure the most suitable parameters are chosen to guarantee both efficiency and security.

In terms of practical FHE implementations, further research into suitable hardware designs and optimisations of existing schemes could provide a large speed up, as indicated in [16]. Optimisations at an algorithmic level are required; for example parameters must be optimised to maximise efficiency of implementations. Moreover, batching techniques proposed for FHE schemes, [10]–[12], could greatly improve performance of any implementation and should also be investigated further. Optimisations at an architectural level are also needed. One major bottleneck in the implementation of these schemes is memory storage. Large parameter sizes and very large

ciphertext sizes consume large amounts of memory, which requires memory management. Lastly, optimisations to target specific devices, such as using the embedded multipliers on an FPGA, are required. For possible FPGA implementations, the use of off-chip DDR3 memory is almost certainly required and therefore data transfer could become a performance issue. Thus, research into any optimisations reducing the memory requirements would be useful for future implementations.

In conclusion, the area of homomorphic cryptography remains an interesting area with a lot of scope for future research. Although further work on FHE schemes is needed to improve and optimise performance, the new avenue of targeting hardware or GPU technology for optimised FHE architectures also looks very promising, and brings the possibility of real time implementations of FHE a step closer.

ACKNOWLEDGMENT

Funding for this research was in part provided by the US National Science Foundation CNS Awards #1117590 and #1319130.

REFERENCES

[1] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, 1999, pp. 223–238.

[2] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[3] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation*, pp. 169–180, 1978.

[4] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009.

[5] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *STOC*, 2012.

[6] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "Cloud-assisted multiparty computation from fully homomorphic encryption," *IACR Cryptology ePrint Archive*, vol. 2011, p. 663, 2011.

[7] C. Gentry and S. Halevi, "Implementing Gentry's fully-homomorphic encryption scheme," in *EUROCRYPT*, 2011, pp. 129–148.

[8] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," *IACR Cryptology ePrint Archive*, vol. 2012, p. 99, 2012.

[9] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "Fully homomorphic encryption without bootstrapping," *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 18, p. 111, 2011.

[10] N. P. Smart and F. Vercauteren, "Fully homomorphic SIMD operations," *IACR Cryptology ePrint Archive*, vol. 2011, p. 133, 2011.

[11] Z. Brakerski, C. Gentry, and S. Halevi, "Packed ciphertexts in LWE-based homomorphic encryption," *IACR Cryptology ePrint Archive*, vol. 2012, p. 565, 2012.

[12] J.-S. Coron, T. Lepoint, and M. Tibouchi, "Batch fully homomorphic encryption over the integers," *IACR Cryptology ePrint Archive*, vol. 2013, p. 36, 2013.

[13] W. Wang, Y. Hu, L. Chen, X. Huang, and B. Sunar, "Accelerating fully homomorphic encryption using GPU," in *HPEC*, 2012, pp. 1–5.

[14] D. Cousins, K. Rohloff, R. Schantz, and C. Peikert. (2011) SIPHER: Scalable Implementation of Primitives for Homomorphic EncRyption. [Online]. Available: http://www.ll.mit.edu/HPEC/agendas/proc11/agenda.html

[15] D. Cousins, K. Rohloff, C. Peikert, and R. E. Schantz, "An update on SIPHER (scalable implementation of primitives for homomorphic encRyption)," in *HPEC*, 2012, pp. 1–5.

[16] C. Moore, N. Hanley, J. McAllister, M. O'Neill, E. O'Sullivan, and X. Cao, "Targeting FPGA DSP slices for a large integer multiplier for integer based FHE," *Workshop on Applied Homomorphic Cryptography*, vol. 7862, 2013.

[17] X. Cao, C. Moore, M. O'Neill, N. Hanley, and E. O'Sullivan, "High speed fully homomorphic encryption over the integers," in *Workshop on Applied Homomorphic Cryptography, to appear*, 2014.

[18] W. Wang and X. Huang, "FPGA implementation of a large-number multiplier for fully homomorphic encryption," in *ISCAS*, 2013, pp. 2589–2592.

[19] C. Gentry and S. Halevi, "Fully homomorphic encryption without squashing using depth-3 arithmetic circuits," *IACR Cryptology ePrint Archive*, vol. 2011, p. 279, 2011.

[20] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *Public Key Cryptography*, 2010, pp. 420–443.

[21] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *FOCS*, 2011, pp. 97–106.

[22] C. Gentry, S. Halevi, and N. P. Smart, "Better bootstrapping in fully homomorphic encryption," *IACR Cryptology ePrint Archive*, vol. 2011, p. 680, 2011.

[23] C. Gentry, S. Halevi, and N. P. Smart, "Fully homomorphic encryption with polylog overhead," *IACR Cryptology ePrint Archive*, vol. 2011, p. 566, 2011.

[24] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *STOC*, 2005, pp. 84–93.

[25] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," *IACR Cryptology ePrint Archive*, vol. 2012, p. 230, 2012.

[26] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," in *CRYPTO*, 2011, pp. 505–524.

[27] K. Lauter, M. Naehrig, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" *Cloud Computing Security Workshop*, pp. 113–124, 2011.

[28] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *EUROCRYPT*, 2010, pp. 24–43.

[29] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, "Fully homomorphic encryption over the integers with shorter public keys," in *CRYPTO*, 2011, pp. 487–504.

[30] J.-S. Coron, D. Naccache, and M. Tibouchi, "Public key compression and modulus switching for fully homomorphic encryption over the integers," in *EUROCRYPT*, 2012, pp. 446–464.

[31] H. Perl, M. Brenner, and M. Smith. (2011) hcrypt. [Online]. Available: http://www.hcrypt.com/scarab-library/

[32] S. Halevi and V. Shoup. (2012) HElib, homomorphic encryption library. https://github.com/shaih/HElib.

[33] S. Halevi. (2012) Performance of HElib. [Online]. Available: http://mpclounge.files.wordpress.com/2013/04/hespeed.pdf

[34] Y. Doröz, E. Öztürk, and B. Sunar, "Evaluating the hardware performance of a million-bit multiplier," in *Digital System Design (DSD), 16th Euromicro Conference on*, 2013.

[35] W. Wang, Y. Hu, L. Chen, X. Huang, and B. Sunar, "Exploring the feasibility of fully homomorphic encryption," *IEEE Transactions on Computers*, vol. 99, no. PrePrints, p. 1, 2013.

[36] N. Göttert, T. Feller, M. Schneider, J. Buchmann, and S. A. Huss, "On the design of hardware building blocks for modern lattice-based encryption schemes," in *CHES*, 2012, pp. 512–529.

[37] T. Pöppelmann and T. Güneysu, "Towards practical lattice-based public-key encryption on reconfigurable hardware," in *Selected Areas in Cryptography*, 2013, pp. 14–16.

[38] T. Pöppelmann and T. Güneysu, "Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware," in *LATINCRYPT*, 2012, pp. 139–158.

[39] Y. Doröz, E. Öztürk, and B. Sunar, "Accelerating fully homomorphic encryption in hardware," 2013, draft, Under Review. [Online]. Available: http://ecewp.ece.wpi.edu/wordpress/vernam/files/2013/09/Accelerating-Fully-Homomorphic-Encryption-in-Hardware.pdf

[40] P. G. Comba, "Exponentiation cryptosystems on the IBM PC," *IBM Systems Journal*, vol. 29, no. 4, pp. 526–538, 1990.

[41] T. Güneysu, "Utilizing hard cores of modern FPGA devices for high-performance cryptography," *J. Cryptographic Engineering*, vol. 1, no. 1, pp. 37–55, 2011.

[42] M. S. Lee, "On the sparse subset sum problem from Gentry-Halevi's implementation of fully homomorphic encryption," *IACR Cryptology ePrint Achive: Report 2011/567*, 2011.