Homomorphic Encryption for Secure Computation on Big Data

Roger A. Hallman, Mamadou H. Diallo, Michael A. August and Christopher T. Graves
United States Department of Defense, SPAWAR Systems Center Pacific, San Diego, Ca, U.S.A.

Keywords: Big Data, Homomorphic Encryption, Data Privacy, Secure Computation.

Abstract: With the ubiquity of mobile devices and the emergence of Internet of Things (IoT) technologies, most of our activities contribute to ever-growing data sets which are used for big data analytics for a variety of uses, from targeted advertising to making medical and financial judgments and beyond. Many individuals and organizations adopt this new big data paradigm without giving any consideration to privacy and security when they create this data and voluntarily give it up for aggregation. Data breaches have become such a common occurrence that it is easy to despair that concepts like privacy and security are antiquated and we should simply accept data leakage as a new normal. Homomorphic Encryption (HE) is a method of secure computation which allows for calculations to be made on encrypted data without decrypting it and without giving away information about the operations being done. While HE has historically been plagued by computational inefficiencies, the field is rapidly advancing to a point where it is efficient enough for practical use in limited settings. In this paper, we argue that, with sufficient investment, HE will become a practical tool for secure processing of big data sets.

1 INTRODUCTION

The integration of current and emerging technologies into every aspect of human activity is contributing to the generation of ever-increasing data sets about both individuals and organizations. This data is aggregated for big data analytics that are put to a variety of uses, such as targeted advertising or contributing to medical or financial decisions. "Big data analytic" refers to the ability to ingest extremely large quantities of data, efficiently process and analyze it, and make conclusions and inferences from it (Boyd and Crawford, 2012). A major challenge for organizations that collect, transmit, store, and process on these data sets is how to preserve the security and privacy of the data when processing it. The problems of protecting data in transit and at rest have been extensively studied (). Currently, cryptographic based security protocols exist, which securely and efficiently protect data sets of any size when transiting on the network and when being stored in data store. In fact, the U.S. Federal government has supported the development of AES (Miller et al., 2009) and selected it as the recommended solution for protecting data. The remaining outstanding issue is how to efficiently and security protected data; in particular, big data, while in processing.

Due to this lack of practical techniques for protecting security and privacy of data when being processed remotely (e.x, in the cloud), news of data security breaches, sometimes strikingly severe and at other times seemingly trivial, is a commonplace occurrence. Examples of particularly serious data breaches include the 2017 Equifax data breach that affected more than 143,000,000 people, primarily in the United States (Ng and Musil, 2017). Much of the data that was stolen from Equifax was in plaintext (Newman, 2017), presumably for computational convenience. Another particularly severe event was the 2015 breach at the United States Office of Personnel Management, where records of more than 21.5 million people were stolen (Zengerle and Cassella, 2015). Other data breaches were less impactful but still high profile.

Homomorphic encryption has been recognized as one of the ideal approaches to securing and processing data in remote servers including the cloud. HE enables operations to be performed directly on encrypted data without ever using the decryption key. Using HE, data is encrypted on the client side, pushed into the cloud, securely processed, and results are sent back to the client for decryption. A sample prototype application making use of HE technology is *CallForFire*, a defense application that we developed, which implements the "call for indirect fire" protocol by securely outsourcing computations to the cloud using HE (Diallo et al., 2016). Another sample application using HE is a cloud-based secure VOIP

Hallman, R., Diallo, M., August, M. and Graves, C.
Homomorphic Encryption for Secure Computation on Big Data.
DOI: 10.5220/0006823203400347
In *Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security (IoTBDS 2018)*, pages 340-347
ISBN: 978-989-758-296-7

application on mobile devices (Rohloff et al., 2017) (Secure VOIP). Other potential applications are listed in (Archer et al., 2017), including some key attributes of the applications such as latency, data volume, and practicality.

HE solves an important problem in data security and privacy. While some applications with limited scope, such as *CallForFire* and *Secure VOIP*, can make use of HE, currently it is not mature enough to be a general solution for secure computation in real world applications. In particular, the performance of HE technology is not sufficient for performing general purpose computations on big data sets in a practical amount of time, preventing it from being used in applications doing big data analytics. Most modern applications could not be run on a HE computation platform owing to limitations in performance, which result from the following: the specific HE scheme used, the implementation of the scheme within the HE software library, and the limitations of the hardware used to run the applications. Furthermore, while there is much interest in enhancing the performance of HE technology, the HE community today is still small but is growing.

In this paper, we argue that, with sufficient investment, HE will become a practical tool for secure processing of big data sets. The extrapolation of trends in the HE community since 2009 indicate that there is a growing interest in the development of practical HE libraries. The time it takes for the widespread adoption of HE to occur is a function of the degree of research and development investment made into the technology. This investment will enable the necessary performance enhancements to both the software and hardware, including enhancements to the HE schemes used, their implementation in the form of software libraries, and the underlying hardware on which the libraries are run, among other things. Currently, there is a need for more organizations, including government, academia, and industry, to further invest in research and development of HE technology. Active participation by various organizations will help to accelerate the maturity and adoption of HE.

2 BACKGROUND ON HOMOMORPHIC ENCRYPTION

In this section, we present Homomorphic Encryption. We first provide a brief high-level background, covering the development of important cryptosystems and implementations. We then introduce the foundational mathematical problem for the most practical Homomorphic Encryption schemes. Finally, we present the current state of Homomorphic Encryption research, including standardization efforts and major research initiatives.

2.1 A Brief History of Homomorphic Encryption

Fully Homomorphic Encryption, or more simply Homomorphic Encryption (HE), refers to a secure computation technique that was first theorized by Rivest, Adleman, and Dertouzos in 1978 (Rivest et al., 1978), allowing computation on encrypted data by taking advantage of certain mathematical properties of various encryption schemes. The computed results remain in an encrypted state and can be further computed on or decrypted. In fact, many cryptographic schemes theoretically allow varying degrees of homomorphic properties (e.g., RSA, based on exponentiation, allows for an additive homomorphism) however these desirable properties are lost in the process of real world system implementations. Partially Homomorphic Encryption, allowing for a limited class of secure computations on encrypted data, have been in use for several decades (ElGamal, 1985; Paillier et al., 1999). The first working HE scheme was introduced in 2009 by Gentry (Gentry, 2009), which was built on `NAND` gates to evaluate low-degree polynomials over encrypted data and used a "bootstrapping" function capable of evaluating its own decryption circuit and at least one more function. Gentry's scheme was revolutionary, though terribly inefficient. This inefficiency led other researchers to build new HE systems that would preserve the desirable functionality of Gentry's work while improving the computation efficiency. The most successful of these newer HE implementations are based off of the Ring Learning With Errors (RLWE) Problem (Regev, 2010) and include the Brakerski-Gentry-Vaikuntanathan (BGV) (Brakerski et al., 2011) and Fan-Vercauteren (FV) (Fan and Vercauteren, 2012) cryptography schemes.

One of the earliest implementations of any HE scheme was Shoup and Halevi's HELib[1], which was an implementation of BGV. One implementation of the FV cryptography scheme is Microsoft Research's Simple Encrypted Arithmetic Library (SEAL)[2]. There have been hardware-based

[1]https://github.com/shaih/HElib

[2]http://sealcrypto.org/

optimization efforts that have improved upon the efficiency of RLWE-based HE implementations as well. For instance, (Cousins et al., 2017a) demonstrated the efficacy of FPGA architectures and (Diallo et al., 2016; Diallo et al., 2017) demonstrated GPGPU-based optimization. The CUDA Homomorphic Encryption Library has further pursued GPGPU-based HE optimization (Dai and Sunar, 2015).

2.2 Mathematical Foundations of Homomorphic Encryption

Regev (Regev, 2010) describes the **Learning With Errors Problem** as follows: Choose a parameter $n \geq 1$, a modulus $q \geq 2$, and an 'error' probability distribution χ on $\mathbb{Z}_q$. Let $A_{\mathbf{s},q}$ on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ be the probability distribution obtained by choosing a vector $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing an error $e \in \mathbb{Z}_q$ according to χ and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$, with additions performed in $\mathbb{Z}_q$. An algorithm solves the Learning With Errors Problem with modulus q and error distribution χ if for any $\mathbf{s} \in \mathbb{Z}_q^n$, and given an arbitrary number of samples from $A_{\mathbf{s},q}$, it outputs $\mathbf{s}$ with high probability. The Learning With Errors Problem is provably hard (Albrecht et al., 2015), however large key sizes (e.g., $O(n^2)$) are required for cryptographic schemes built on the regular Learning With Errors Problem; RLWE-based cryptographic schemes achieve linear key sizes. The RLWE Problem is achieved by replacing $\mathbb{Z}_q^n$ with the polynomial ring $\mathbb{Z}_q[x]/\langle x^n + 1 \rangle$, where n is a power of 2. The hardness of RLWE under worst-case assumptions has similarly been proven (Lyubashevsky et al., 2013).

2.3 Recent Investment and Efforts in Homomorphic Encryption

Gentry's initial HE scheme and the successive cryptographic systems that followed it have spawned multiple implementations across academia and industry, the most successful of which are based on BGV and FV. While they are functionally similar, the implementations available are quite diverse and a standardization effort has begun to bring current and future HE implementations under a common framework (Lauter et al., 2017). The demand for easily available, scalable secure computation technology will only grow as cloud computing services become less expensive and more powerful. This demand will likely be met by developers who are not cryptography experts. Standardization is imperative to simplify and uniformize HE implementation APIs, provide a clear and straightforward understanding of security properties, and pinpoint appropriate use cases for HE in the real world.

2.3.1 API Standardization for Homomorphic Encryption

API standardization efforts revolve around two main thrusts (Brenner et al., 2017):

1. Storage Model APIs, which mostly follow existing design patterns.
    - Cryptographic context includes information on the state of the homomorphic encryption/evaluation session, instructs the compiler on circuit generation, and determines which cryptographic library provides the instruction set. Cryptographic context is, in a sense, metadata for the information being stored and processed.
    - Payload Representation of the keys, plaintexts, and ciphertexts enables parameters and data to be serialized, transported, and deserialized consistently across various platforms and HE implementations.
2. HE Assembly Language
    - Circuit Description information, which includes information on the circuits being executed may include low-level calls to library functions, and be library-specific.

2.3.2 Homomorphic Encryption Security

HE offers a guarantee of data security to sectors where it is adopted, and standardization will increase its adoption and utility. A key element of this standardization process will be a consensus on security parameter levels across diverse implementations and systems. An HE scheme has three security properties (Chase et al., 2017):

1. Indistinguishability under chosen-plaintext attack (IND-CPA).
    - No adversary has an advantage in guessing whether a given ciphertext is an encryption of two different messages. Encryptions are randomized to ensure that no two encryptions of the same message appear the same.
2. Compactness.
    - Homomorphic operations on the ciphertexts do not expand the length of the ciphertexts.
3. Efficient decryption.

- Decryption runtime does not depend on the functions which were evaluated on the ciphertexts.

Parameter generation requires four inputs:

- A security level λ (i.e., $\lambda = 128$-bit security).
- A plaintext modulus P.
- A dimension K of all vectors to be encrypted.
- An auxiliary parameter B for controlling the complexity of circuits that can be run to process encrypted data.

These parameters are used to generate a public key, a secret key, and an evaluation key. The public key can be used by anyone to encrypt a message while the secret key is used for decryption. The generated evaluation key is given to an entity that will compute on the encrypted data.

2.3.3 Use Cases for Homomorphic Encryption

HE has shown potential for a diverse array of use cases, from the medical field to financial industries and beyond (Archer et al., 2017). It is expected that in the next decade or two a significant fraction of the world population will have full genome sequences, which will prove to be a powerful tool in the study of biology and medicine. Human DNA and RNA sequences are unique biometric identifiers, which may convey medically significant or socially sensitive information and, once released, can never be retrieved or retracted. The adoption of HE will allow genomic data sets to be uploaded to cloud data centers where they can be used to provide precision medicine. Geneticists have begun to recognize the potential impact of HE on their field to the point that the iDASH Privacy & Security Workshop[3] routinely has a task for machine learning on encrypted data.

3 CURRENT TRENDS IN USING HOMOMORPHIC ENCRYPTION TO PROCESS BIG DATA

A number of specific applications have been developed to evaluate the security, usability, and performance of HE applied to big data. In particular, a number of Machine Learning techniques have been investigated to analyze how they can take advantage of HE to securely process data. Below, we survey a number of such techniques.

[3]http://www.humangenomeprivacy.org/

Graepel, et al. (Graepel et al., 2012), showed early results that machine learning (e.g., Linear Means and Fisher's Linear Discriminant) on smaller, homomorphically encrypted data sets could be practical. Gilad-Bachrach, et al. (Gilad-Bachrach et al., 2016), showed a dramatic improvement in the application of neural network operations on homomorphically encrypted data. Specifically, they used artificial feed-forward neural networks for predictions on a data set consisting of 60,000 hand-written digits, with each image in a 28×28 array. Using a training data set of 50,000 images and a test set of the remaining 10,000, they were able to achieve nearly 59,000 predictions per hour with about 99% accuracy.

Aslett, et al. (Aslett et al., 2015), conducted an extensive review of software tools for statistical machine learning on homomorphically encrypted data. They evaluated a sample of HE implementations from the perspective of statistical analysis and commented on the constraints of working on homomorphically encrypted data. Constraints that are mentioned for almost all implementations that they look at include:

- Simple mathematical operations that have not yet been implemented in HE libraries (i.e., square root, division and comparison). These constraints typically require computation to be completed in the clear.
- The inability of HE implementations to handle extensive computation on floating point numbers. Because all HE implementations require fixed points for computation, it is common for floating point numbers to be truncated and rescaled (Diallo et al., 2015).

Esperanca, et al. (Esperanca et al., 2017), demonstrate that efficient and scalable statistical analysis of homomorphically encrypted data sets requires specially tailored and customized computational methods that may be very different from state-of-the-art methods for unencrypted environments. Aono, et al. (Aono et al., 2016), implemented logistic regressions using multiple HE schemes and determined that this operation was scalable over homomorphically encrypted data, tolerating data sets with 10^8 elements.

Yonetani, et al. (Yonetani et al., 2017), used an homomorphic cryptosystem for visual learning, including facial recognition and detecting places where sensitive information could be accessible. The Paillier Cryptosystem (Paillier et al., 1999) was used to create a doubly-permuted homomorphic encryption, which allows high-dimensional classifiers to be updated securely and efficiently. Sparsity constraints are enforced on classifier updates, and updated weights are decomposed into a small number of non-zero values. Only the non-zero values are homomorphically encrypted. The facial recognition algorithm was trained on a data set of more than 162,000 images with a test set of about 20,000 images. The facial recognition task involved classifying 40 different attributes and achieved 84% accuracy. Detecting sensitive places (e.g., a lavatory, a personal computer terminal, etc.) was achieved with a training data set of more than 131,000 images of places and about 7,200 test images with about 72% accuracy.

There have been notable achievements in using HE for machine learning tasks such as the *k*-nearest neighbors problem and building convolutional neural networks. (Shaul et al., 2018) recently used HElib to demonstrate a scalable and efficient solution to the *k*-nearest neighbors problem using encrypted data in near real time. Given a query point, *q*, they were able to find the nearest 20 points in a set of over 1,000 in under an hour. They showed that efficient and scalable statistical algorithms are achievable in HE environments. Additionally, they implemented a “coin toss” algorithm and an effective “comparison” algorithm. In their experiments, they used their algorithms to find the nearest hotels to a given query point in Boston, Massachusetts. (Juvekar et al., 2018) have recently developed a low latency framework for secure neural networks that utilizes a packed additive HE scheme in tandem with Yao’s Garbled Circuits, a multi-party computation technique. This framework, labeled “GAZELLE”, enables clients to classify private images using a convolutional neural network while not revealing their input to the server. Within these convolutional neural networks, HE is used for computation within the linear network layers and garbled circuits for computation in the non-linear network layers. They were able to attain at least 3 orders of magnitude lower latency and 2 orders of magnitude lower bandwidth than previous HE-based approaches to machine learning on neural networks.

As can be seen above, several researchers are looking at how to take advantage of HE for performing Machine Learning-based computations on big data sets within specific domains. The trend of using HE approaches for analyzing big data is promising. This trend can serve as a vehicle for motivating the research, development, and adoption of HE for secure computation on big data.

4 CURRENT TRENDS IN HARDWARE OPTIMIZATION FOR HOMOMORPHIC ENCRYPTION

There are several trends in using hardware-based approaches to accelerate HE operations. In particular, the use of FPGAs, ASICs, GPUs, CPU extensions, and clustering, are all approaches that have been investigated. These approaches can be grouped into two categories: offloading computation to a HE co-processor (FPGAs, ASICs, GPUs), and using CPU vector computation extensions.

The approach of offloading computation to a HE co-processor enhances the performance of certain HE operations by performing them directly in hardware, which leads to significant overall performance improvements when performing HE computations. One approach is the implementation of FPGA modules for large integer multiplication and modulus reduction for speeding up these HE operations (Cao et al., 2014). They demonstrated a speedup of 44x compared to a software-based approach. In (Cousins et al., 2017b), the authors offloaded certain transforms and ring operations to the FPGA and demonstrated speedup in a sample problem utilizing encrypted string comparisons. Another approach which offloads computations makes use of an ASIC to implement large integer multiplication and modulus reduction modules to significantly improve performance of HE operations while reducing circuit footprint (i.e. gate count) (Dorz et al., 2015).

An approach for speeding up the performance of the FPGA is to pipeline the circuits implemented in it. This has the advantage of optimizing the utilization of the FPGA board, which can result in increased throughput and reduced latency. These approaches can be extended by combining clustering with offloading computation to an FPGA, as proposed in (Roy et al., 2015). The approach of using GPUs to accelerate HE operations leverages the highly parallelized and high throughput memory architecture of GPUs. For instance, an implementation of HE primitives that makes use of a GPU realized speedup of 7.68x for encryption and 7.4x for decryption compared to a CPU implementation (Wang et al., 2012). The approach of using accelerated CPU vector computation makes use of CPU instruction set extensions (e.g., AVX, SSE, NEON) to accelerate vector computations using SIMD. Such a technique is used by (Migliore et al., 2017). A fully optimized solution may require combining these different approaches into a hybrid solution.

5 CURRENT TRENDS IN HOMOMORPHIC ENCRYPTION LIBRARIES

The success of homomorphic encryption in practical use depends on how well the community can deliver libraries that are efficient and simple to use. Today, there exists a number of open source HE libraries such as HElib, NFLlib, PALISADE, and SEAL. However, these libraries provide mostly low-level homomorphic operations including arithmetic addition, subtraction, multiplication, and comparison. These operations are useful in demonstrating that the libraries work on toy problems, but that they are less useful when building real-world systems that need to be deployed in the cloud. Any attempt at building a system using these libraries will require enormous effort from the programmer to figure out how the capabilities of the libraries can fit their need, and to figure out what the limitations of the libraries are. For example, HElib doesn't support floating point computations. If a system requires floating point numbers, then the task of extending the library to handle floating point numbers is left to the programmer. Likewise, the programmer is left with the task of finding parameter settings that lead to optimal performance of the library. We argue that, in order to promote the widespread adoption of HE, these libraries and any subsequent ones, need to be matured by providing APIs that abstract out the complexity of the libraries. In addition, the libraries need to provide APIs that can be used to easily integrate the libraries into systems by both cryptography experts and non-experts.

Considering the complexity of defining and implementing a homomorphic encryption scheme, the adoption of HE may well start with solutions that are tailored to specific problems. The HE libraries can be optimized to efficiently solve specific problems. This observation is based on the fact that HE libraries can perform well with some types of operations, and not so well with other operations. For instance, most libraries perform well with addition and subtraction operations, but perform poorly with multiplication. Our own experience with HElib shows that multiplication is slower than addition.

CallForFire. Our experience with HElib shows that it can be used in practice with certain types of applications including interactive applications (Diallo et al., 2016). Recall that CallForFire is a mission-critical, cloud-based defense application that implements the indirect call for fire protocol used during combat operations when an infantry unit is impractical for engagement with a target. The call for indirect fire protocol involves essentially a few players. A *Forward Observer* is deployed in advance in the field to look for *Targets* to be destroyed by *Firing Units* directed by a *Fire Direction Center*. Note that this protocol is interactive. When the *Forward Observer* detects a *Target*, he/she sends the *Target*'s location information to the *Fire Direction Center*, which then selects a *Firing Unit* to destroy the *Target*. In this application, the number of messages to be exchanged between the players is minimal. Furthermore, only two computations need to be performed by the fire direction center: the location of the target, and the distance between the firing unit and the target. More importantly, these computations deal only with addition and multiplication on integers in the Military Grid Reference System (MGRS). MGRS represents a location with a five-digit integer on a fixed grid. Due to all these characteristics of the call for fire protocol, we were able to successfully implement CallForFire using HElib and deploy it in the cloud to securely process the operations of the protocol.

Secure VoIP. Prior to the development of CallForFire, a practical VoIP teleconferencing mobile application using end-to-end homomorphic encryption based on NTRU had been developed. In this application, the voice data is sampled, encoded and encrypted on the mobile device clients that are communicating with each other, then sent over a generic network such as the open Internet to the encryption-enabled cloud-based server, and mixed at the server without decrypting the VoIP data. The encrypted results of the mixing are then sent back to the client mobile devices for decryption, decoding and playback to the end-users (Rohloff et al., 2017). Note that the only homomorphic operation used in this application is addition. In addition, the number of operations to be performed for any call is low. The above characteristics of the VoIP application make it an ideal candidate for practical implementation using homomorphic encryption.

6 CONCLUSION AND FUTURE WORK

In this paper, we explore the trends underlying advances in HE development, focusing particularly on big data. We argue that, with sufficient investment, HE will become a practical tool for secure processing of big data sets. The current trends in the use of homomorphic encryption to secure big data analysis highlights the growing interest in the adoption of HE. This trend is also complemented by research in enhancing the performance of HE through novel hardware-based approaches. The various hardware-based approaches shows the growing research activity in this area. In addition, the trends in the development of HE libraries demonstrate that libraries can be tailored to address specific big data analytics. In the future, we will perform a comprehensive survey of HE libraries and hardware-based optimizations to evaluate their performance.

REFERENCES

Albrecht, M. R., Player, R., and Scott, S. (2015). On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203.

Aono, Y., Hayashi, T., Trieu Phong, L., and Wang, L. (2016). Scalable and secure logistic regression via homomorphic encryption. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pages 142–144. ACM.

Archer, D., Chen, L., Cheon, J. H., Gilad-Bachrach, R., Hallman, R. A., Huang, Z., Jiang, X., Kumaresan, R., Malin, B. A., Sofia, H., Song, Y., and Wang, S. (2017). Applications of homomorphic encryption. Technical report, HomomorphicEncryption.org, Redmond WA.

Aslett, L. J., Esperança, P. M., and Holmes, C. C. (2015). A review of homomorphic encryption and software tools for encrypted statistical machine learning. *arXiv preprint arXiv:1508.06574*.

Boyd, D. and Crawford, K. (2012). Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon. *Information, communication & society*, 15(5):662–679.

Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (2011). Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277. Available at https://eprint.iacr.org/2011/277.

Brenner, M., Dai, W., Halevi, S., Han, K., Jalali, A., Kim, M., Laine, K., Malozemoff, A., Paillier, P., Polyakov, Y., Rohloff, K., Savaş, E., and Sunar, B. (2017). A standard api for rlwe-based homomorphic encryption. Technical report, HomomorphicEncryption.org, Redmond WA.

Cao, X., Moore, C., O'Neill, M., Hanley, N., and O'Sullivan, E. (2014). High-speed fully homomorphic encryption over the integers. In Böhme, R., Brenner, M., Moore, T., and Smith, M., editors, *Financial Cryptography and Data Security*, pages 169–180, Berlin, Heidelberg. Springer Berlin Heidelberg.

Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Hoffstein, J., Lauter, K., Lokam, S., Moody, D., Morrison, T., Sahai, A., and Vaikuntanathan, V. (2017). Security of homomorphic encryption. Technical report, HomomorphicEncryption.org, Redmond WA.

Cousins, D., Rohloff, K., and Sumorok, D. (2017a). Designing an fpga-accelerated homomorphic encryption co-processor. *IEEE Transactions on Emerging Topics in Computing*.

Cousins, D. B., Rohloff, K., and Sumorok, D. (2017b). Designing an fpga-accelerated homomorphic encryption co-processor. *IEEE Transactions on Emerging Topics in Computing*, 5(2):193–206.

Dai, W. and Sunar, B. (2015). cuhe: A homomorphic encryption accelerator library. Cryptology ePrint Archive, Report 2015/818.

Diallo, M. H., August, M., Hallman, R., Kline, M., Au, H., and Beach, V. (2015). Nomad: A framework for developing mission-critical cloud-based applications. In *Availability, Reliability and Security (ARES), 2015 10th International Conference on*, pages 660–669. IEEE.

Diallo, M. H., August, M., Hallman, R., Kline, M., Au, H., and Beach, V. (2016). Callforfire: A mission-critical cloud-based application built using the nomad framework. In Clark, J., Meiklejohn, S., Ryan, P. Y., Wallach, D., Brenner, M., and Rohloff, K., editors, *Financial Cryptography and Data Security*, pages 319–327, Berlin, Heidelberg. Springer Berlin Heidelberg.

Diallo, M. H., August, M., Hallman, R., Kline, M., Au, H., and Slayback, S. M. (2017). Nomad: a framework for ensuring data confidentiality in mission-critical cloud-based applications. In *Data Security in Cloud Computing*, Security, pages 19–44. Institution of Engineering and Technology.

Dorz, Y., ztrk, E., and Sunar, B. (2015). Accelerating fully homomorphic encryption in hardware. *IEEE Transactions on Computers*, 64(6):1509–1521.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472.

Esperanca, P., Aslett, L., and Holmes, C. (2017). Encrypted accelerated least squares regression. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 334–343, Fort Lauderdale, FL, USA. PMLR.

Fan, J. and Vercauteren, F. (2012). Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144. Available at https://eprint.iacr.org/2012/144.

Gentry, C. (2009). A fully homomorphic encryption scheme. phd thesis, stanford university, 2009.

Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. (2016). Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 201–210, New York, New York, USA. PMLR.

Graepel, T., Lauter, K., and Naehrig, M. (2012). Ml confidential: Machine learning on encrypted data. In

International Conference on Information Security and Cryptology, pages 1–21. Springer.

Juvekar, C., Vaikuntanathan, V., and Chandrakasan, A. (2018). Gazelle: A low latency framework for secure neural network inference. *arXiv preprint arXiv:1801.05507*.

Lauter, K., Laine, K., Rohloff, K., Chen, L., and Zimmermann, R. (2017). Homomorphic encryption standardization: An open industry/government/academic consortium to advance secure computation. Available at http://homomorphicencryption.org/.

Lyubashevsky, V., Peikert, C., and Regev, O. (2013). On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43:1–43:35.

Migliore, V., Seguin, C., Real, M. M., Lapotre, V., Tisserand, A., Fontaine, C., Gogniat, G., and Tessier, R. (2017). A high-speed accelerator for homomorphic encryption using the karatsuba algorithm. *ACM Trans. Embed. Comput. Syst.*, 16(5s):138:1–138:17.

Miller, F. P., Vandome, A. F., and McBrewster, J. (2009). *Advanced Encryption Standard*. Alpha Press.

Newman, L. H. (2017). 6 fresh horrors from the equifax ceo's congressional hearing. Available at https://www.wired.com/story/equifax-ceo-congress-testimony/.

Ng, A. and Musil, S. (2017). Equifax data breach may affect nearly half the us population. Available at https://www.cnet.com/news/equifax-data-leak-hits-nearly-half-of-the-us-population/.

Paillier, P. et al. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *Eurocrypt*, volume 99, pages 223–238. Springer.

Regev, O. (2010). The learning with errors problem. *Invited survey in CCC*, page 15.

Rivest, R. L., Adleman, L., and Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180.

Rohloff, K., Cousins, D. B., and Sumorok, D. (2017). Scalable, practical voip teleconferencing with end-to-end homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 12(5):1031–1041.

Roy, S. S., Järvinen, K., Vercauteren, F., Dimitrov, V., and Verbauwhede, I. (2015). Modular hardware architecture for somewhat homomorphic function evaluation. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 164–184. Springer.

Shaul, H., Feldman, D., and Rus, D. (2018). Scalable secure computation of statistical functions with applications to k-nearest neighbors. *arXiv preprint arXiv:1801.07301*.

Wang, W., Hu, Y., Chen, L., Huang, X., and Sunar, B. (2012). Accelerating fully homomorphic encryption using gpu. In *2012 IEEE Conference on High Performance Extreme Computing*, pages 1–5.

Yonetani, R., Boddeti, V. N., Kitani, K. M., and Sato, Y. (2017). Privacy-preserving visual learning using doubly permuted homomorphic encryption. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2059–2069.

Zengerle, P. and Cassella, M. (2015). Estimate of americans hit by government personnel data hack skyrockets. Available at https://www.reuters.com/article/us-cybersecurity-usa/millions-more-americans-hit-by-government-personnel-data-hack-idUSKCN0PJ2M420150709.