Secure FPGA as a Service - Towards Secure Data Processing by Physicalizing the Cloud

Mark A. Will
Cyber Security Lab
The University of Waikato
Hamilton, New Zealand
Email: mark.will@waikato.ac.nz

Ryan K. L. Ko
Cyber Security Lab
The University of Waikato
Hamilton, New Zealand
Email: ryan.ko@waikato.ac.nz

***Abstract*—Securely processing data in the cloud is still a difficult problem, even with homomorphic encryption and other privacy preserving schemes. Hardware solutions provide additional layers of security and greater performance over their software alternatives. However by definition the cloud should be flexible and adaptive, often viewed as abstracting services from products. By creating services reliant on custom hardware, the core essence of the cloud is lost. FPGAs bridge this gap between software and hardware with programmable logic, allowing the cloud to remain abstract. FPGA as a Service (FaaS) has been proposed for a greener cloud, but not for secure data processing. This paper explores the possibility of Secure FaaS in the cloud for privacy preserving data processing, describes the technologies required, identifies use cases, and highlights potential challenges.**

***Index Terms*—Secure Processing; Privacy Preserving Data Processing; FPGA as a Service; Homomorphic Encryption; Green Computing; Secure Multi-Party Computation;**

I. Introduction

The holy grail of cryptography, fully homomorphic encryption, is too slow for computing a single operation [1]. Therefore users and businesses storing data in the cloud need to choose between functionality and security when selecting a cloud service. Most pick functionality, because they trust the service to protect their data. However, in reality the only entity that can be truly trusted is the user themselves. A physical approach to privacy preserving data processing is with Field-Programmable Gate Arrays (FPGAs) and custom hardware, which bestow greater security and privacy over their software alternatives [2][3]. We propose Secure FPGA as a Service (SFaaS) to leverage these security properties of FPGAs to harden the cloud against both insider and outsider threats.

The terminology FPGA as a Service (FaaS) has been proposed for a cleaner and greener cloud environment [4][5][6], because of greater performance per unit of power. FPGAs are already used in cloud architectures such as Microsoft for performance and configurability [7][8]. Google also uses custom hardware to accelerate machine learning tasks in the cloud, but are not configurable [8]. Until recently, customers did not have access to FPGAs in the cloud for their own designs. Amazon are aiming to change this with their EC2 F1 Instances [9], and are now publicly available to preview. However initially this service is targeted at massively parallel and computational intensive tasks. The feasibility of low latency applications is currently unclear as the FPGA would ideally need direct access to a networking interface. However the limitation for EC2 F1 Instances to support SFaaS is encrypting the FPGA definition such that only the FPGAs themselves can decrypt it.

A FPGA bridges the gap between hardware and software, by providing performance closer to that of an Application-Specific Integrated Circuit (ASIC), while having the reconfigurability of a microprocessor. They contain a finite amount of programmable logic, also known as reconfigurable logic, that can be used to implement digital circuits by applying a bitstream file to the device. This bitstream file is analogous to a compiled program in software, but where programs contain machine instructions, a bitstream file contains a sequence of bits which configure circuits and logical functions. For secure processing this bitstream file must be protected from all entities other than the customer and FGPA device, and is discussed in Section II where the proposed service is given. This section also discuses task switching, encryption schemes for data entering and leaving the FPGA, and attack vectors.

Processing data in parallel can give better performance, however it is usually achieved by computing the same function over chucks of data at the same time. FPGAs offer a slightly different form of parallel processing by pipelining a design. This allows different stages of a function to be processed in parallel such that the data flows through the function. This parallelisation allows proposed use cases in Sections III and IV to have greater performance as well as security.

This paper proposes a simple service model, and provides a survey of previous work for the use of FPGAs, concluding with some benefits and challenges of providing SFaaS in Sections V and VI. As discussed in Section III-A, SFaaS offers greater protection to plaintext data processing in the cloud, over traditional software approaches. Finally, even greater security can be achieved by combining SFaaS with other privacy preserving schemes.

II. Proposed Service

Figure 1 shows the general model for the cloud service, where each virtual machine has access to a FPGA – this is the current approach by Amazon. In this section the idea of FaaS is first given, before how it can extended to SFaaS for greater security and privacy in the cloud.

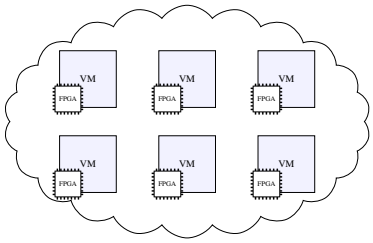

Fig. 1. Virtual Machines having access to FPGAs in the cloud.

A. Booking Method

There are many possible options for booking methods, which can be categorised into 3 groups: $24/7$ use, fixed time, or shared. The $24/7$ option would be like renting server hardware for an undetermined length of time, and is the easiest to be implemented. This is the current offering from Amazon. Booking a FPGA for a fixed time, for example 1 hour, is similar in terms of complexity as the $24/7$ option, but requires more management. Finally, the shared option would be the most complex to implement as task switching is required. Which model would be cheapest for the customer is unknown due to the complexity of the shared option.

B. Task Switching

In order for the FPGA pool to be shared, task switching is necessary to stop resources being blocked. Central Processing Units (CPUs) can switch tasks hundreds a time a second, however a FPGA cannot match this because of the reconfiguration time. Average times are low milliseconds, which will be made worse with bitstream file encryption. This is not including swapping out data, which will require encryption before leaving the FPGA.

Each cloud service needs to define the input and output wires for every FPGA design to meet to order to be compatible with their service. Naturally there are the data input and outputs for receiving data from the virtual machine. However the interface for task switching is important as well. There are two approaches which could be offered: (1) when the task is being switched, all working data is lost, and (2) a few register like variables are encrypted before being saved. The encryption key could be seperate to the input and output data, but would need to be specified in the bitstream file.

After the input wire signalling the task is to be swapped goes high, a predefined number of clock signals would pass before the task actually replaced. This allows the task to clear any data that is sensitive to prevent the next task possibly being able to see it, encrypt and output any data, and making use of the saved registers (by outputting encrypted data for later use). Other options for task switching where the FPGA is only used for a short period of time, is each design specifies how long it will require the FPGA, or the design can signal when to be switched. For example if a few encrypted values need to be averaged, then once the computation is finished it can signal to be switched. Thus saving the customer money and freeing resources for the service to sell to others.

C. Attack Vectors

In Figure 1 each virtual machine has its own FPGA. In this case each virtual machine could store an encrypted bitstream file that only its FPGA could decrypt. However in most cases this would not be guaranteed, as FPGAs would be shared. Given a pool of FPGAs, the client could upload an encrypted bitstream file for each FPGA. It is essential each FGPA has a different key so that in the event of one being compromised, it can be removed easily from the pool. Another challenge with this approach is stopping a malicious user from copying the encrypted bitstream file and manually running it on the intended FPGA. However the design should always encrypt data leaving the FPGA, mitigating this attack.

The primary attack is physically stealing the bitstream file decryption keys from within the FGPAs. If the functionality accessible from the cloud environment is limited to bitstream file loading, data in/out and a few control signals, then a remote attacker cannot gain access to the keys. For the most part an employee of a cloud service should be trusted to protect customers data. However a Google administrator was fired in 2010 for abusing his rights to access teenage girls data in order to stalk them [10]. One could argue that physical attacks would defer and be too difficult for many employees, certainly more so than current software solutions. Combined with limiting access to data-centres, video surveillance, and locked chassis (especially the FGPA housing) adds to the difficulty.

Apart from black box attacks, all other attack vectors require physical access, including readback attacks, side-channel attacks and reverse engineering the bitstream [11].

1) Black Box Attacks: A common attack for systems where all possible input combinations are tried, with the output revealing the inner design [11]. This attack is not feasible given that the input and output data must be encrypted, where the public key may not even be known. The design should however handle incorrect input, for example a value not encrypted with the correct public key.

2) Readback Attacks: For debugging, FPGAs often have a readback feature to allow values to be read from the FPGA, for example the decryption keys through a special interface. Methods off disabling this functionality exist [11], however for a production FPGA deployed in a cloud service, this functionality should not exist or be physically disabled in the chip once the chip has passed production tests.

3) Side-Channel Attacks: These attack vectors are viable, and involve analysing physical properties of the FPGA while in operation, for example power consumption [12] or electromagnetic radiation [13].

4) Reverse Engineering the Bitstream: With enough time and effort, the design of a bitstream file (once decrypted) can be reverse engineered [11]. However an attacker only need focus of finding the decryption key for the input data, which could be easier depending on how obfuscated the design is.

D. Securing the Bitstream File

Because the service needs to be able to reconfigure the FPGAs, secure FPGA switch types cannot be used, for example

Antifuse [14]. Therefore SRAM is required which needs a bitstream file to define the connections. A major challenge in the proposed system is keeping the bitstream file protected such that it is only exposed on the FPGA. This is a similar to the protection of intellectual property for designs [15]. Mechanisms exist to encrypt bitstream files [16][17], but often suffer limitations: (1) symmetrical encryption key, (2) battery needed for key and (3) each FPGA using the same key. For SFaaS, asymmetric encryption should be used so that any customer can encrypt their design without being able to decrypt others [18].

Each FPGA should have a seperate key pair as discussed in Section II-C, where no entity has knowledge of the private key. In order to create a key pair where no entity has knowledge of the decryption key is a current open challenge. One technique is that the FPGAs are hardcoded with a private key which is generated automatically, and the public key is printed on the chip or included in the box. The keys should never be saved, and only visible for a brief moment during manufacturing. The manufacturer could also sign all public keys, allowing customers to verify the key was generated by the manufacturer, not by a malicious entity. However the private key is visible at some point in time, so is not the ideal solution.

E. Physical Unclonable Function

Recent work on Physical Unclonable Functions (PUFs) is allowing for secrets to be derived from complex physical characteristics of the silicon (a physical one-way function) rather than storing the secrets in memory [19][20][21]. Guajardo *et al.* proposed the feasibility of using PUFs for intellectual property protection by encrypting the bitstream file using elliptic curve cryptography [22]. PUFs are naturally noisy which combined with varying temperatures and ageing could affect the reliability [23]. However the key only needs to be generated or regenerated once per power cycle. Therefore after the FPGA is powered on and the private key has passed a test against the known public key, any PUF issues will not be encountered until it is reset.

F. Programmed Decryption Key

With a protected bitstream file, the decryption key for data processing can remain protected, as it will only be exposed within the FPGA. Three cryptography schemes will be analysed for usage in a FPGA in terms of logic size and performance. Note that logic size depends on the FPGA used, and are given as an approximation.

1) AES: The Advanced Encryption Standard (AES) is widely used in secure FPGA designs and offers very fast performance ($>$ 20Gbps) [24][25][26]. Designs can also be tailored to use less logic, from thousands of slices down to a few hundred while still achieving megabits per second [25][26]. The advantage of AES over other cryptography schemes in terms of implementation on a FPGA is the simplicity of the algorithm. Hence the flexible performance versus logic required. The only main limitation of AES is that it uses a symmetric key. Even with the key protected in the bitstream file, data sharing and multiple data sources remains an issue. However AES could be used for data storage and processing, while all uploaded or outgoing data could be encrypted with a public key cryptosystem.

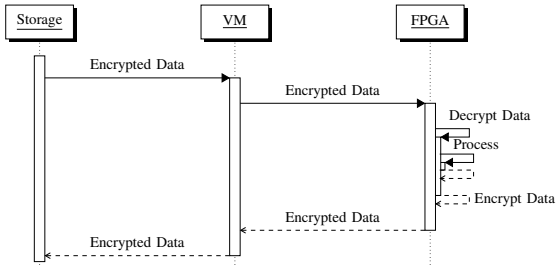

Fig. 2. SFaaS data flow for secure processing.

2) RSA: A public key encryption scheme designed on the factoring problem, RSA computes the plain-text or cipher base value to an exponent within a modulo [27], and can use Montgomery modular multiplication [28][29]. RSA is more expensive in terms of area and performance compared to AES for FPGAs. For example a slower implementation still requires thousands of slices, where faster implementations can require tens of thousands [30]. Performance can vary between megabits per second down to kilobits per second [29].

3) ECC: Also a public key cryptosystem, Elliptic Curve Cryptography (ECC) was proposed independently by Neal Koblitz and Victor Miller in 1985, and its cryptographic strength comes from the elliptic curve discrete logarithm problem being hard [31]. An early FPGA implementation in [32] showed a $\times 30$ speedup over software implementations, using only a few thousand slices. A recent survey in 2007 showed the varying difference between state-of-the-art implementations [33], and in 2008 a 33.05μs solution on a 163 bit binary field was proposed [34].

III. Use Cases

A. Plaintext Processing

The state-of-the-art secure processor, AEGIS [2], was designed to only reveal the data inside the processor. Therefore any data leaving the processor is encrypted. This protects against a range of software and physical attacks. Similarly a recent proposal for a secure MIPS processor in a FPGA by Songhori *et al.* also allows for development of secure applications or functions in a high-level language [3]. However instead of full processor designs, secure functions could be implemented in an FPGA and compute over plaintext with their input and output encrypted, similar to Figure 2. For example a hardware secured cloud voting application [35].

AEGIS still has security vulnerabilities in the form of side-channel-attacks [36][37]. This attack vector analyses information "leaked" from the physical execution of a program, for example power consumption [12] or electromagnetic radiation [13]. The other limitation of a custom secure processor was the practicality of deployment in the cloud, but with SFaaS could be a made a reality. Even with existing attack vectors, FPGA secure processors would have greater security over existing cloud techniques implemented in software.

B. Homomorphic Encryption

Given that FPGAs are still vulnerable to attack (more challenging than traditional cloud attacks), schemes can be implemented on SFaaS to further protect data. Processing data securely can also be achieved using homomorphic encryption, which exists in 2 flavours: Partially Homomorphic Encryption (PHE) and Fully Homomorphic Encryption (FHE). PHE supports a single operation, for example, addition or multiplication. Where FHE can support many operations computed over encrypted data.

Cryptographic schemes supporting single homomorphic operations have been around since RSA was proposed in 1978 [27]. For some applications, only one operation is required, and in these cases PHE is an ideal solution [38][39][40][35]. FHE was only proven plausible by Gentry as late as 2009 [41], many years after PHE. Wang *et. al.* [42] showed performance results of a revised FHE scheme by Gentry and Halevi [43] in 2015 for the recrypt function. CPU and GPU implementations took 17.8 seconds and 1.32 seconds respectively, using a small dimension size of 2048 [42]. A medium dimension size of 8192 took 96.3 seconds and 8.4 seconds for the same function [42]. The current limitation for FHE is that implementations [1] of FHE schemes for FPGAs cannot give practical processing times, however a speedup is given over software implementations. Using SFaaS, smaller keys could be used for FHE, and protected by stronger encryption outside of the FPGA. This would make extracting data through side-channel attacks more difficult, as plaintext values cannot be directly discovered.

C. Privacy Preserving Processing with MPC

An alternative privacy processing technique is secure Multi-Party Computation (MPC) [44]. An example scheme is FRIBs (FRagmenting Individual Bits) proposed in [45], where data is fragmented across different service providers such that the data can only be visible once the fragments are joined together. Arbitrary processing can still occur using *NAND* operations on the fragments. Proof-of-concept results show a massive performance gain over fully homomorphic encryption, closer to that of partially homomorphic encryption.

Figure 3 shows a system model, where each server could be implemented in an FPGA. The security of FRIBs is achieved through distribution where each fragment server is hosted on different providers, reducing the risks of zero-day and insider attacks. Therefore implementing FRIBs on FPGAs would further enhance its security and reduce these risks (as discussed in Section V-B), by making it harder to gain access to the fragments, and to learn patterns on reduction requests. Then by using SFaaS, the fragments are encrypted outside of the FPGA, reducing the risks even more. The FPGAs could have access to their own network interface, or pass requests through the virtual machine.

An example FPGA module is given in Figure 4, which takes two encrypted sets of fragments (A and B) and an instruction (for example *add*). The module decrypts A and B using AES, and begins computing the addition operation. More

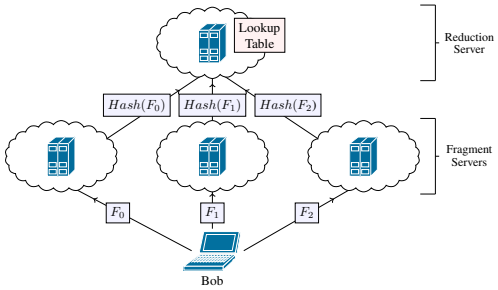

Fig. 3. An example of FRIBs with 3 Fragment Servers and 1 Reduction Server [45].

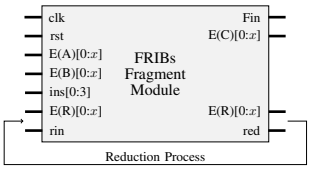

Fig. 4. Example of the interface for a fragment module for FRIBs in a FPGA.

details on how the addition is achieved with FRIBs is given in [45]. Implementing a half-adder in FRIBs on a FPGA would take a few clock cycles to combine and get the obfuscated lookup value for all 32-bits, where with a CPU this process needs to be repeated 32 times. The module encrypts the reduction request to be sent, and upon receiving the encrypted response, continues processing. The main challenge for a MPC implementation with SFaaS is that the booking time needs to be the same for each service, as each server needs to be running at the same time.

D. Random Number Generator

Many security protocols depend on either real or pseudo random number generators. Real random number generation requires some form of entropy. With virtual machines in the cloud, the guest operating system can request the entropy from the physical machine. This request can be slow, and can result in an entropy pool being reused. Kohlbrenner *et al.* proposed a true random number generator for a FPGA which achieved an output of 0.5Mbits per second [46]. This design used two ring oscillators, and a sampler circuit to extract the jitter contained in the signals. Tsoi *et al.* proposed FPGA implementations for a true and pseudo random number generator in [47]. Both designs used less than a few hundred slices, and passed tests for random number generation [48].

Connected to a virtual machine would give enough random bits for most security applications. With SFaaS the connection between the FPGA and virtual machine would remain secure. They also have the ability of being joined with another design in the FPGA. For example some encryption algorithms need random numbers. Potential issues are with multiple FPGAs from the same wafer, in a bank near each other, and exposed to the same heat and electronic radiation, whether they remain truely random when compared to each other. Another is sharing of FPGA resources, and whether this exposes any weaknesses.

E. Low Latency

Fast performance allows the develop of low latency applications and services. However if the FPGA has access to a network interface, low latency network requests and responses can be achieved. For example financial exchanges provide updates instantly, requiring processing in the sub-millisecond latency range [49], leading to high frequency trading [50] – latency is proportional to profit. In terms of security, low latency is critical for firewalls with deep packet inspection [51][52]. FPGAs allow hardware rules to updated easily when compared to dedicated appliances. The ability to deploy a custom packet inspector in the cloud within a FPGA will also give performance advantages over software alternatives. Better performance allows for sophisticated inspection, and gives customers more control over their security. Therefore keeping their cloud applications responsive, even with high traffic volumes flowing through the deep packet inspector.

IV. Other Use Cases

In this section use cases are given where FaaS would provide a benefit in the cloud. SFaaS could also improve many of these use cases for greater security and privacy.

A. Big Data Processing and Deep Learning

In a survey on big data platforms by Singh *et al.* FPGAs scored highly for data throughput, but low for the size of data supported when compared to other technologies [53]. Therefore there is a balance between throughput and data size, because even if a FPGA can process small data really quickly, if it cannot hold enough in the chip, some big data analytics may be difficult. Another aspect of big data processing in FPGAs is deep learning [54]. Farabet *et al.* implemented a convolutional neural network [55] in a CPU, GPU, and two FPGAs [56]. The FPGAs split the performance of the GPU, with the CPU the slowest. Similar results were achieved in [57] where a FPGA out performed a CPU both in terms of execution time, and power consumed. Deep learning could also have data encrypted outside of the FPGA, resulting in secure searching for tasks like object recognition.

B. FFT

The fast Fourier transform (FFT) is used for a range of applications, from computer science to geology. FPGAs offer a faster alternative to software for computing the FFT [58][59], including low powered devices [60]. Therefore having faster computation for FFTs would improve the performance and possibly the overall power consumed for many cloud applications.

C. Image Processing

There is a shift between FPGAs and graphics processing units (GPUs) in which one gives better performance and energy efficiency for image processing [61], including computing the FFT [62]. However SFaaS can offer private image processing, while FPGAs are more universal than GPUs for many applications and functions.

V. Benefits and Challenges

A. Power Consumption

FaaS and FPGAs in the cloud were primarily proposed for greener computing [4][5][6]. For example with an implemented convolutional neural network, Zhang *et al.* showed an FPGA used much less energy compared to a software implementation [57]. This has positives for both customer and service provider, as less energy results in lower operating costs while reducing infrastructure investment for green energy sources.

B. Security

FPGAs and hardware implementations in general suffer from weaknesses and have many attack vectors, some of which are discussed in Section II-C. However many require physical access to the device, where many threats for a cloud environment come from outside attacks. If physical access is required to try and uncover the decryption keys to cipher data of some user, then outside attacks become more limited when compared to software implementations. However some challenges are how reliable the PUFs would be, as they are needed to protect the bitstream file, and transferring public keys to customers.

Insider attacks from cloud employees and administrators [63] are a threat, and arguably the bigger thread to customer data [64][65][66]. A survey by Kaspersky and B2B International revealed that 73% of companies have had internal information security incidents, and state that the single largest cause of confidential data loss is by insiders (42%) [66]. Physically discovering decryption keys from a hardware device requires advanced skills and tools than software implementations, and is more physically noticeable through swipe card access to datacenters, security cameras, and security guards. Therefore reducing the risk of insider attacks.

C. Development

Designs for FPGAs requires using a hardware description language (HDL), for example Verilog or VHDL. These are similar to software programming languages, such as defining functions with inputs and outputs. However the key difference between software and hardware design is that software is sequential, where one operation is performed before another. However in a FPGA many operations happen in parallel for each clock cycle, for example when setting a register, this should not be read until the next clock cycle. So defining hardware requires a different developer mindset than software, as the clock and the amount of parallelism of offer needs to be considered.

Development time is something that might also be an issue, with simulation, testing and debugging more difficult on FPGAs than with software. However recent advancements in development tools are making the process easier. Current application development in the cloud is primarily software based, therefore the ratio between software and hardware designers would be relatively high in this domain. With the introduction of FaaS this could change, as more hardware designer could

be required for cloud development. However there are tools emerging for software developers to create designs for FPGAs without requiring extensive knowledge [67][68].

D. Copying Data

Even though FPGAs can process data very quickly, transferring data to and from an external processor can expensive [59][60][69]. With encrypted data flowing in and out of the device, the amount of data will be greater for many cryptography schemes than plaintext. However if encryption/decryption can occur at line-rate, for secure data processing this added latency will not be problematic. Only in low latency designs will the transfer latency be felt.

VI. Concluding Remarks

Amazon's EC2 F1 Instances have proven the possibility of FaaS, and only once it is fully available will its true capabilities be realised. For SFaaS to become reality, further features will be needed, especially for bitstream file encryption. If the FPGA is used only for one customer, then encryption such that only the FPGA has the decryption key is not as critical, if the customer trusts Amazon and their staff 100%. In this case the bitstream file would never be saved in the cloud. However for near true secure processing, the bitstream file will need to be secured through to the FPGA. The option to task switch is something currently not offered by Amazon, and results in customers only paying for the time they actually use.

There are many other related works and applications where SFaaS could provide extra performance and additional layers of security. FaaS can offer security through deep packet inspection if a network interface is directly accessible, but SFaaS offers much more as all data outside of the FPGA is encrypted. Even though data processing in a FPGA is not guaranteed to be fully secure, it improves upon existing cloud techniques by making both insider and outsider attacks more difficult. Combined with a weak fully homomorphic scheme, or the distributed approach of MPC, gives even greater levels of data privacy.

With secure FPGA as a service, users would gain greater privacy and security when using third-party cloud services, enabling them to have more control over their data.

Acknowledgements

This research is supported by STRATUS (Security Technologies Returning Accountability, Trust and User-Centric Services in the Cloud) (https://stratus.org.nz), a science investment project funded by the New Zealand Ministry of Business, Innovation and Employment (MBIE).

References

[1] X. Cao, C. Moore, M. O'Neill, N. Hanley, and E. O'Sullivan, "High-speed fully homomorphic encryption over the integers," in *Financial Cryptography and Data Security*, pp. 169–180, Springer, 2014.

[2] G. E. Suh, C. W. O'Donnell, and S. Devadas, "AEGIS: A single-chip secure processor," *Information Security Technical Report*, vol. 10, no. 2, pp. 63–73, 2005.

[3] E. M. Songhori, S. Zeitouni, G. Dessouky, T. Schneider, A.-R. Sadeghi, and F. Koushanfar, "Garbledcpu: a mips processor for secure computation in hardware," in *Proceedings of the 53rd Annual Design Automation Conference*, p. 73, ACM, 2016.

[4] O. Yanovskaya, M. Yanovsky, and V. Kharchenko, "The concept of green cloud infrastructure based on distributed computing and hardware accelerator within fpga as a service," in *Proceedings of IEEE East-West Design Test Symposium (EWDTS 2014)*, pp. 1–4, Sept 2014.

[5] S. A. Fahmy and K. Vipin, "A case for FPGA accelerators in the cloud," *ACM SoCC (Poster)*, 2014.

[6] Altera Corporation, "Altera FPGAs Achieve Compelling Performance-per-Watt in Cloud Data Center Acceleration Using CNN Algorithms." Online http://www.prnewswire.com/news-releases/altera-fpgas-achieve-compelling-performance-per-watt-in-cloud-data-center-acceleration-using-cnn-algorithms-300039440.html (Accessed 12/03/17), February 2015.

[7] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, *et al.*, "A reconfigurable fabric for accelerating large-scale datacenter services," in *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, pp. 13–24, IEEE, 2014.

[8] B. H. Frank, "Microsoft Azure networking is speeding up, thanks to custom hardware." Online http://www.pcworld.com/article/3124927/microsoft-azure-networking-is-speeding-up-thanks-to-custom-hardware.html (Accessed 13/03/17), September 2016.

[9] Amazon Web Services, Inc, "Amazon EC2 F1 Instances (Preview)." Online https://aws.amazon.com/ec2/instance-types/f1/ (Accessed 15/03/17).

[10] A. Chen, "GCreep: Google Engineer Stalked Teens, Spied on Chats." Online. http://gawker.com/5637234/gcreep-google-engineer-stalked-teens-spied-on-chats (Accessed 16/01/17), September 2010.

[11] T. Wollinger and C. Paar, *How Secure Are FPGAs in Cryptographic Applications?*, pp. 91–100. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[12] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO'99*, pp. 388–397, Springer, 1999.

[13] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Cryptographic Hardware and Embedded Systems—CHES 2001*, pp. 251–261, Springer, 2001.

[14] S. Brown and J. Rose, "FPGA and CPLD architectures: A tutorial," *IEEE design & test of computers*, vol. 13, no. 2, pp. 42–57, 1996.

[15] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Robust FPGA intellectual property protection through multiple small watermarks," in *Design Automation Conference, 1999. Proceedings. 36th*, pp. 831–836, IEEE, 1999.

[16] T. Kean, "Secure configuration of field programmable gate arrays," in *International Conference on Field Programmable Logic and Applications*, pp. 142–151, Springer, 2001.

[17] L. Bossuet, G. Gogniat, and W. Burleson, "Dynamically configurable security for sram fpga bitstreams," *International Journal of Embedded Systems*, vol. 2, no. 1-2, pp. 73–85, 2006.

[18] S. Malipatlolla and S. A. Huss, "A novel method for secure intellectual property deployment in embedded systems," in *Programmable Logic (SPL), 2011 VII Southern Conference on*, pp. 203–208, IEEE, 2011.

[19] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM conference on Computer and communications security*, pp. 148–160, ACM, 2002.

[20] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th annual Design Automation Conference*, pp. 9–14, ACM, 2007.

[21] M.-D. M. Yu and S. Devadas, "Secure and robust error correction for physical unclonable functions," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 48–65, 2010.

[22] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "Physical unclonable functions and public-key crypto for FPGA IP protection," in *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*, pp. 189–195, IEEE, 2007.

[23] A. Maiti, L. McDougall, and P. Schaumont, "The Impact of Aging on an FPGA-based Physical Unclonable Function," in *Field Programmable Logic and Applications (FPL), 2011 International Conference on*, pp. 151–156, IEEE, 2011.

[24] A. Hodjat and I. Verbauwhede, "A 21.54 gbits/s fully pipelined aes processor on fpga," in *Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th Annual IEEE Symposium on*, pp. 308–309, IEEE, 2004.

[25] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J.-D. Legat, "Compact and efficient encryption/decryption module for fpga implementation of the aes rijndael very well suited for small embedded applications," in *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, vol. 2, pp. 583–587, IEEE, 2004.

[26] T. Good and M. Benaissa, "Aes on fpga from the fastest to the smallest," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 427–440, Springer, 2005.

[27] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, 1978.

[28] M. K. Hani, T. S. Lin, and N. Shaikh-Husin, "Fpga implementation of rsa public-key cryptographic coprocessor," in *TENCON 2000. Proceedings*, vol. 3, pp. 6–11, IEEE, 2000.

[29] A. Daly and W. Marnane, "Efficient architectures for implementing montgomery modular multiplication and RSA modular exponentiation on reconfigurable logic," in *Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*, pp. 40–49, ACM, 2002.

[30] A. Cilardo, A. Mazzeo, L. Romano, and G. P. Saggese, "Exploring the design-space for FPGA-based implementation of RSA," *Microprocessors and Microsystems*, vol. 28, no. 4, pp. 183–191, 2004.

[31] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[32] K. Leung, K. Ma, W. K. Wong, and P. H. W. Leong, "Fpga implementation of a microcoded elliptic curve cryptographic processor," in *Field-Programmable Custom Computing Machines, 2000 IEEE Symposium on*, pp. 68–76, IEEE, 2000.

[33] G. M. de Dormale and J.-J. Quisquater, "High-speed hardware implementations of elliptic curve cryptography: A survey," *Journal of systems architecture*, vol. 53, no. 2, pp. 72–84, 2007.

[34] W. N. Chelton and M. Benaissa, "Fast elliptic curve cryptography on fpga," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 16, no. 2, pp. 198–205, 2008.

[35] M. A. Will, B. Nicholson, M. Tiehuis, and R. K. Ko, "Secure Voting in the Cloud using Homomorphic Encryption and Mobile Agents," in *2015 International Conference on Cloud Computing Research and Innovation (ICCCRI)*, pp. 173–184, IEEE, 2015.

[36] B. Yang, K. Wu, and R. Karri, "Scan based side channel attack on dedicated hardware implementations of data encryption standard," in *International Test Conference, 2004. Proceedings. ITC 2004.*, pp. 339–344, IEEE, 2004.

[37] B. Köpf and D. Basin, "An information-theoretic model for adaptive side-channel attacks," in *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 286–296, ACM, 2007.

[38] M. Hirt and K. Sako, "Efficient receipt-free voting based on homomorphic encryption," in *Advances in Cryptology—EUROCRYPT 2000*, pp. 539–556, Springer, 2000.

[39] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: protecting confidentiality with encrypted query processing," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pp. 85–100, ACM, 2011.

[40] D. Boneh, C. Gentry, S. Halevi, F. Wang, and D. J. Wu, "Private database queries using somewhat homomorphic encryption," in *Applied Cryptography and Network Security*, pp. 102–118, Springer, 2013.

[41] C. Gentry *et al.*, "Fully homomorphic encryption using ideal lattices.," in *STOC*, vol. 9, pp. 169–178, 2009.

[42] W. Wang, Y. Hu, L. Chen, X. Huang, and B. Sunar, "Exploring the feasibility of fully homomorphic encryption," *Computers, IEEE Transactions on*, vol. 64, no. 3, pp. 698–706, 2015.

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

[44] A. C. Yao, "Protocols for secure computations," in *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pp. 160–164, IEEE, 1982.

[45] M. A. Will, R. K. Ko, and I. H. Witten, "Privacy Preserving Computation by Fragmenting Individual Bits and Distributing Gates," in *Trustcom/BigDataSE/ISPA, 2016 IEEE*, vol. 1, pp. 900–908, IEEE, 2016.

[46] P. Kohlbrenner and K. Gaj, "An embedded true random number generator for fpgas," in *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, pp. 71–78, ACM, 2004.

[47] K. H. Tsoi, K. Leung, and P. H. W. Leong, "Compact fpga-based true and pseudo random number generators," in *Field-Programmable Custom Computing Machines, 2003. FCCM 2003. 11th Annual IEEE Symposium on*, pp. 51–61, IEEE, 2003.

[48] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," tech. rep., DTIC Document, 2001.

[49] G. W. Morris, D. B. Thomas, and W. Luk, "Fpga accelerated low-latency market data feed processing," in *High Performance Interconnects, 2009. HOTI 2009. 17th IEEE Symposium on*, pp. 83–89, IEEE, 2009.

[50] J. W. Lockwood, A. Gupte, N. Mehta, M. Blott, T. English, and K. Vissers, "A low-latency library in fpga hardware for high-frequency trading (hft)," in *High-Performance Interconnects (HOTI), 2012 IEEE 20th Annual Symposium on*, pp. 9–16, IEEE, 2012.

[51] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood, "Deep packet inspection using parallel bloom filters," in *High performance interconnects, 2003. proceedings. 11th symposium on*, pp. 44–51, IEEE, 2003.

[52] Y. H. Cho and W. H. Mangione-Smith, "Deep packet filter with dedicated logic and read only memories," in *Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th Annual IEEE Symposium on*, pp. 125–134, IEEE, 2004.

[53] D. Singh and C. K. Reddy, "A survey on platforms for big data analytics," *Journal of Big Data*, vol. 2, no. 1, p. 8, 2015.

[54] J. Zhu and P. Sutton, "Fpga implementations of neural networks–a survey of a decade of progress," in *International Conference on Field Programmable Logic and Applications*, pp. 1062–1066, Springer, 2003.

[55] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[56] C. Farabet, Y. LeCun, K. Kavukcuoglu, and E. Culurciello, "Large-scale fpga-based convolutional networks," *Scaling up Machine Learning: Parallel and Distributed Approaches*, pp. 399–419.

[57] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 161–170, ACM, 2015.

[58] K. S. Hemmert and K. D. Underwood, "An analysis of the double-precision floating-point fft on fpgas," in *Field-Programmable Custom Computing Machines, 2005. FCCM 2005. 13th Annual IEEE Symposium on*, pp. 171–180, IEEE, 2005.

[59] C. Cullinan, C. Wyant, T. Frattesi, and X. Huang, "Computing performance benchmarks among cpu, gpu, and fpga," 2012.

[60] M. A. Will, "Real-Time Image Processing," Honours Thesis, The University of Waikato, 2013.

[61] B. Cope, P. Y. Cheung, W. Luk, and S. Witt, "Have gpus made fpgas redundant in the field of video processing?," in *Field-Programmable Technology, 2005. Proceedings. 2005 IEEE International Conference on*, pp. 111–118, IEEE, 2005.

[62] B. Duan, W. Wang, X. Li, C. Zhang, P. Zhang, and N. Sun, "Floating-point mixed-radix fft core generation for fpga and comparison with gpu and cpu," in *2011 International Conference on Field-Programmable Technology*, pp. 1–6, Dec 2011.

[63] A. Chen, "Google Engineer Stalked Teens, Spied on Chats." Online [Accessed 26/08/14] http://gawker.com/5637234/gcreep-google-engineer-stalked-teens-spied-on-chats, Gawker, September 2010.

[64] M. Theoharidou, S. Kokolakis, M. Karyda, and E. Kiountouzis, "The insider threat to information systems and the effectiveness of iso17799," *Computers & Security*, vol. 24, no. 6, pp. 472–484, 2005.

[65] N. Giandomenico and J. de Groot, "Insider vs. Outsider Data Security Threats: What's the Greater Risk?." Online. https://digitalguardian.com/blog/insider-outsider-data-security-threats (Accessed 10/03/17), January 2017.

[66] Kaspersky Lab and B2B International, "Over 5,500 IT specialists were surveyed from 26 countries around the world," 2015.

[67] D. Pellerin and S. Thibault, *Practical FPGA programming in C*. Prentice Hall Press, 2005.

[68] XILINX INC, "Vivado high-level synthesis." Online https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html (Accessed 12/03/17).

[69] T. El-Ghazawi, E. El-Araby, M. Huang, K. Gaj, V. Kindratenko, and D. Buell, "The promise of high-performance reconfigurable computing," *Computer*, vol. 41, no. 2, 2008.