Hardware Support for Secure Stream Processing in Cloud Environments

Jeff Anderson
Department of Electrical and Computer Engineering
The George Washington University
jeffa@gwu.edu

Tarek El-Ghazawi
Department of Electrical and Computer Engineering
The George Washington University
tarek@gwu.edu

ABSTRACT

Many-core microprocessor architectures are quickly becoming prevalent in data centers, due to their demonstrated processing power and network flexibility. However, this flexibility comes at a cost; co-mingled data from disparate users must be kept secure, which forces processor cycles to be wasted on cryptographic operations. This paper introduces a novel, secure, stream processing architecture which supports efficient homomorphic authentication of data and enforces secrecy of individuals' data. Additionally, this architecture is shown to secure time-series analysis of data from multiple users from both corruption and disclosure. Hardware synthesis shows that security-related circuitry incurs less than 10% overhead, and latency analysis shows an increase of 2 clocks per hop. However, despite the increase in latency, the proposed architecture shows an improvement over stream processing systems that use traditional security methods.

CCS CONCEPTS

•Networks → Network security; •Hardware → Signal processing systems;

KEYWORDS

stream processing, secure processing, network-on-a-chip, homomorphic MAC

ACM Reference format:
Jeff Anderson and Tarek El-Ghazawi. 2017. Hardware Support for Secure Stream Processing in Cloud Environments. In *Proceedings of CF'17, Siena, Italy, May 15-17, 2017,* 4 pages.
DOI: http://dx.doi.org/10.1145/3075564.3075592

1 INTRODUCTION

Recent trends in computing have used the availability of data centers to enable computation in the cloud. Several operational models have been developed, such as software-as-a-service (SaaS) and infrastructure-as-a-service (IaaS), which allow users to submit data to a data center who then runs software specially tailored to consume the data in real-time. This has enabled cloud-based stream processing systems, which are capable of processing "big data".

CF'17, Siena, Italy

978-1-4503-4487-6/17/05…$15.00
DOI: http://dx.doi.org/10.1145/3075564.3075592

As data centers continue to grow to keep pace with demand, computing efficiency has become a major concern. To improve efficiency while continuing to increase computing power, data centers are turning to microprocessors based on network-on-a-chip (NoC) technology [4]. This architecture allows the overall network to be viewed at a finer level of granularity, with more processor cores accessible to the network and, therefore, individual users.

NoC-based network architectures and the spreading of calculations among different cores introduce a new concern to the cloud computing paradigm; that of securing different users' data which may be co-mingled on a single chip. Required separation of data is not a new phenomenon, as digital design flows have been developed by field programmable gate array (FPGA) vendors which enforce separation of data in the final design for information assurance and safety [2]. NoCs, however, are unable to comply with the requirement for separation of data, as they are based on an architecture which allows packets of data to take any path required, or several paths for large packets, to arrive at its destination [12].

Information assurance best practice mandates the use of cryptography and verification to protect data from disclosure and corruption [16]. However, these are both processing intensive and iterative, each decreasing the overall efficiency of the system [3, 13]. This is most noticeable in stream processing systems, where the requirement for parallelization of computation is in direct conflict with the serial nature of data validation [16]. These conflicting requirements result in data taking unnatural paths through the system, where data is first decrypted and validated as one monolithic structure, and then split and sent to multiple cores for processing.

An architecture is needed which can support the inherent parallelism and security required by cloud-based stream processing systems. It would allow the introduction of data to the system in a confidential manner and support parallel, and out-of-order, validation of subsets of the data, without introducing unnecessary delays into the system.

2 CONTRIBUTIONS

The proposed architecture is capable of securing data from disclosure and corruption in such a way that it allows stream processing systems to execute efficient computations in a NoC environment. Data confidentiality is achieved with secret splitting using Random Linear Network Coding, and validation is achieved with an efficient implementation of a homomorphic message authentication code (MAC). A homomorphic validation scheme was deemed desirable due to its independence from packet sequence, a critical property considering the varying path delay in a NoC that results from parallel data taking different paths to a destination. A homomorphic MAC scheme developed by Agrawal and Boneh [1] was levied and

modified to meet the needs of a NoC environment. Security was achieved through minimal modifications to a baseline NoC router.

3 BACKGROUND

3.1 Related Work

Just as computer security research, in general, has increased, so has security research directed at NoC-based architectures and stream processing. The main thrust of secure stream processing has been in the filtering domain [15]; and the main thrusts of secure NoC research have, thus far, been in three directions: side channels within shared resources, intrusion detection and data filtering.

In [7, 10, 19] the authors identify timing side channels within shared resources. Shared resources, by definition, are available to all NoC routers and, therefore, can be snooped upon. Given enough side channel information, inferences can be made of the data being passed between processing elements. The authors of [19] and [7] propose arbitration and static limiting mechanisms to combat the use of shared resources as side channels.

Proposed solutions for malware-infected NoCs monitor the activity of the NoC and act as an IDS for its network [10, 14]. While an IDS works reasonably well for detecting anomalous network behavior, it does not detect copying of data by an intermediate router.

Proposed solutions for data separation in NoCs allow data to be mixed while on the network, but use firewalls to separate data from unauthorized cores [8, 9]. A firewall uses rules to govern network traffic, which causes network packets to be filtered from unauthorized cores. However, elaborate firewall rules increase the latency of network traversal, as each hop must check several rules before allowing data to pass to the next hop.

However, if data in the NoC network were kept secret, then an infected component able to copy data would be rendered useless, eliminating the need for firewalls. End-to-end encryption of network data is the most frequently used method to avoid disclosure of data on a network [16]. However, this solution is inappropriate for securing data in a stream processing system due to the overhead associated with iterative decryption of data prior to processing.

Our motivation to propose a secure stream processing architecture is to ensure timely delivery of network packets, while maintaining their confidentiality and integrity. The proposed solution is uniquely different from other solutions because it does not restrict network traffic to ensure security; instead, it allows traffic to flow naturally through the network, enabling the system to operate efficiently.

3.2 Linear Network Coding

Linear network coding is an encoding technique used in network communications that support multicast. Linear network coding uses multicast to combine information from multiple sources on a channel, thereby achieving the max-flow bound on the information transmission rate [18]. This is done by allowing nodes inside of the network to perform algebraic operations inside the network.

More generally, each internal node generates a message X_k from the linear combination of received messages $\{m_i\}_{s,i=1}$ by the relation,

$$x_i = \sum_{i=1}^{s} g_k^i \cdot M_i \tag{1}$$

where values $g_{i,k}$ are coefficients selected from GF(2^8) [7].

In addition to advantages regarding channel bandwidth usage, the use of linear network coding can also enable data confidentiality. Secret sharing, an idea introduced by Shamir in 1979 [17], can be implemented using a network coding scheme. Instead of encoding one channel with data from two channels, data from one channel is masked with a random bitstring and both the masked data and the random bitstring are sent over different channels to their destination.

3.3 Homomorphic Validation

Work by Boneh has shown that linear network coding schemes are vulnerable to some data pollution attacks, forcing the sink node to authenticate the data prior to consumption. Additionally, Boneh's work has shown that standard signature schemes are inapplicable to the integrity verification of network coded data [6].

Homomorphic MAC functions differ from traditional hash-based MACS in that for any two blocks b_i and b_j, $h_G(b_i+b_j) = h_G(b_i)h_G(b_j)$. The same holds true for all blocks of message b, enabling greater efficiencies in data verification through parallelization of hashing operations over each block, and then combining the results of each block hash using a collective reduction operation. Agrawal and Boneh introduced the concept of a homomorphic MAC to increase the performance of homomorphic signature validation to the point where it can be used at wireline speeds [1]. This is of great benefit to the signal processing community, as data is often expected to be consumed in real-time. Additional benefits can be gained when considering the commutative properties of homomorphic MACs; network packets are free to arrive in any order during validation without forcing the entire operation to sit idle.

4 PROPOSED SECURE STREAM PROCESSING ARCHITECTURE

The proposed secure, many-core architecture enables secure stream processing by efficiently supporting data confidentiality and validity. It consists of a static-routed NoC architecture, with routers that have been modified to efficiently execute the homomorphic MAC scheme introduced by Agrawal.

4.1 Secure Stream Processor

A notional many-core, secure, stream processing architecture implementing Linear Network Coding and homomorphic MAC is shown in Figure 1.

This distributed-memory architecture consists of processing elements connected in a grid by routers; with each router having 4 virtual channels and being capable of both XY and YX static routing. Each processing element consists of a signal processor and local memory. Processing elements (PE) are free to communicate with

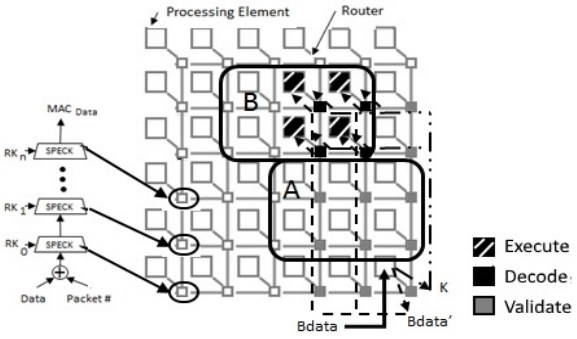

Figure 1: Many-Core secure signal processing architecture. Data enters through PE(6,6) and is linear encoded with a random string prior to being sent to the destination using both XY and YX routing.

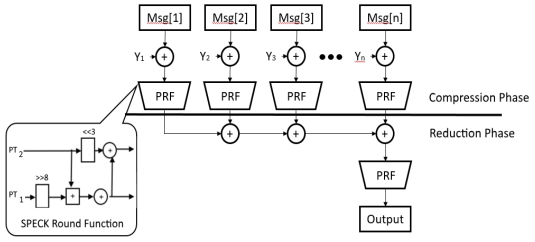

Figure 2: Homomorphic validation with a PHASH reduction phase and SPECK for the compression phase.

each other over the network. Select processing elements are relegated to data offloading and serve as network interface controllers (NIC).

Take two users, Alice and Bob, each requiring data confidentiality during time series stream processing. As a user requests processing for a given batch of data, a NoC dispatcher assigns a cluster of processing elements based on the required computation and available resources.

Note the overlapping security boundaries (virtual boundaries which show the dataflow of protected data for each user of the system) for users Alice and Bob. Notice that a subset of data intended for Bob flows through the security boundary of Alice, based on a static XY or YX NoC routing protocol. This data can be snooped by various means [5, 7, 10], and therefore requires securing.

As data enters the NoC through a NIC, as seen in Figure 1, the entrypoint router generates a keystream K_i using a PRG and combines them with the incoming data B_{data} using the exclusive-or operation, forming $B_{data'}$. It can be shown that data encoded in this manner with a secret key, or Vernam Cipher, provides a higher level of confidentiality than modern block ciphers [4]. $B_{data'}$ is then forwarded to its destination using a static XY routing protocol and K_i is forwarded to the same destination using static YX routing, ensuring that coefficients and encoded data never arrive at the same intermediate node.

4.2 Efficient Homomorphic MAC

There are multiple ways to construct MACs using hash functions or block ciphers [16]. A block cipher-based MAC (CMAC) was deemed most appropriate due to its input block size being equal to the cipher's block size. This allows single flits to be validated without the need to pad to a fixed size. Lightweight cryptographic algorithms are favored to others as pseudorandom functions (PRF), due to their greatly reduced gate complexity and support for small block sizes. SPECK, introduced by the United States National Security Agency and shown in Figure 2, was selected over other lightweight algorithms, such as SIMON and SKINNY, due to its round strength, simple key schedule and lack of S-Boxes.

Latency can further be reduced by unrolling the SPECK round functions, as seen in Figure 1, and housing each round function in an individual router. This is made possible by the NoC dataflow structure; that is, data is handled by many routers on the way to its destination node, and each router can continue the MAC function started by the previous router. The destination router finalizes the last round of the MAC for a data packet, and then combines the results with the results of other MACs using a reduction operation.

There are multiple ways to structure the reduction phase of the parallelizable MAC, as shown in [3, 13]. Each offers tradeoffs between security and performance. Reduction phases introduced by Rogaway, Kaminski, Bellare and Damgard were considered for the homomorphic MAC implementation. Ultimately, Rogaway's PMAC approach, shown in Figure 2, was deemed better suited due to Kaminski's time-complexity being dependent on both the total volume of data and the size of each hash block [13].

4.3 Secure Router Implementation

The proposed secure router is based on a static NoC router developed by the Integrated Systems group at the Technological University of Munich (TUM-LIS). It was parameterized such that it contained 5 ports, with two virtual channels per port, a flit width of 32 bits and a flit depth of 8 flits per packet.

The secure router proceeds to follow the standard NoC router pipeline [11]; however, some modifications were required to support data confidentiality and validation:

(1) **Dual RC stage** — the RC stage was modified to calculate two next hops, one using static XY routing and the other using static YX routing. Splitting the data and keystream using different routing protocols ensures that no intermediate router would come in contact with both.
(2) **Encryption-enabled VA stage** — keystream generation and linear network coding was added to the VA stage. Keystream generation was done using a PRF, and data was encrypted by taking the *exclusive-or* of data and keystream. The VA stage then fills the appropriate virtual channels with encrypted data and its associated keystream.
(3) **Data-validating XB stage** — prior to exiting the router at any port, a SPECK engine executes a single round of SPECK on encrypted data. Additionally, local output is responsible for CMAC reduction and data recovery.

Since the NoC environment cannot guarantee synchronized reception of network packets, a random-access buffer must be added

to the output port of the router that stores either data or keystream packets until both have been received. This reduces any backpressure which may occur in high-traffic conditions, without eroding security of the received data. To reduce the size of the required buffer, the smaller keystream packets are transmitted through the network with the priority bit set, increasing the chance that they will arrive prior to the larger data packets.

5 RESULTS

The proposed NoC architecture from Figure 2 was simulated with Modelsim using gate-level RTL of the network and behavioral models of processing elements programmed to calculate parallelized, Discrete Fourier Transforms (DFT) of time series data. The instruction-accurate models were developed by compiling DFTs, Message Passing Interface, and AES-128 for both ECB decryption and CMAC validation functions, and then disassembling with Ida Pro. 64 kilowords of data was then generated and sent to different 6-core clusters in the NoC for processing. During data processing, a Byzantine node sent data into a cluster to disrupt the stream computations.

5.1 Security Simulation Results

Simulations show that data remained confidential during transit between clusters. Additionally, the Byzantine node was unable to forge data and insert it into the data stream.

5.2 Synthesis Results

The secure router was synthesized using Xilinx Vivado for their Artix-7 family of FPGAs. The additional security features resulted in a 5% increase in LUTs and a 2.5% increase in flipflops. Further increases in router size are due to the additional random-access buffer, whose size is dependent on the required number of packets to be stored. Each key stored in a buffer resulted in approximately a 70-flipflop increase and a 100-LUT increase. We suspect that more hardware-efficient secure architectures are possible.

5.3 Performance Simulation Results

Network packet overhead is increased 100% with the addition of MAC data, resulting in a 50% reduction in throughput. However, overall efficiency is still higher than that of an architecture secured by traditional means.

6 CONCLUSION

Efforts to bolster efficiency through the use of massively parallel architectures are being countered by requirements for ad-hoc network architectures and the information assurance requirements levied upon data centers by their customers. In this work, we have proposed a many-core, secure stream processing architecture capable of securing computations, while taking a minimal performance penalty. Experimental results show that attacks to disclose and corrupt data are, indeed, countered and system efficiency is greater than that achieved by standard methods of securing data.

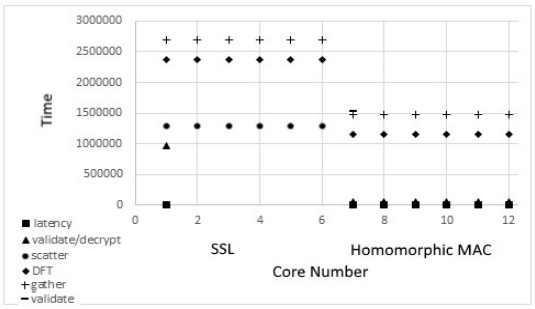

Figure 3: Simulation shows that parallelism enabled by Homomorphic MAC increases performance when compared to traditional validation and decryption schemes.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Vikram Narayana for providing useful comments on the paper and research.

The authors would also like to thank the anonymous referees, Dr. Kathy Potter and Anne Anderson for their valuable comments.

REFERENCES

[1] Shweta Agrawal and Dan Boneh. 2009. Homomorphic MACs: MAC-Based Integrity for Network Coding. In *ACNS '09. Proceedings of the 7th International Conference on Applied Cryptography and Network Security*. ACM, 292–305.
[2] Altera Corporation 2009. *AN 567: Quartus II Design Separation Flow*. Altera Corporation.
[3] John Black and Phillip Rogaway. 2002. A Block-Cipher Mode of Operation for Parallelizable Message Authentication. In *EUROCRYPT '02 Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology*. Springer-Verlag, London, UK.
[4] Smalley E. 2011. Data Center On A Chip. *IEEE Spectrum* (Spring 2011).
[5] Cezar Reinbrecht et al. 2007. Gossip NoC – Avoiding Timing Side-Channel Attacks through Traffic Management. In *NOCS 2007. First International Symposium on Networks-on-Chip*. IEEE.
[6] Dan Boneh et al. 2009. Signing a Linear Subspace: Signature Schemes for Network Coding. In *PKC '09. Irvine Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography*. Springer-Verlag, Berlin, Heidelberg.
[7] Hassan M.G. Wassel et al. 2014. Networks on Chip with Provable Security Properties. *IEEE Micro* 34, 3 (June 2014), 57–68.
[8] Jean-Philippe Diguet et al. 2016. NOC-centric Security of Reconfigurable SoC. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE Computer Society, Washington DC.
[9] Leandro Fiorin et al. 2008. Implementation of a reconfigurable data protection module for NoC-based MPSoCs. In *IPDPS 2008. IEEE International Symposium on Parallel and Distributed Processing*. IEEE.
[10] Martha Johanna Sepulveda et al. 2015. NoC-Based Protection for SoC Time-Driven Attacks. *IEEE Embedded Systems Letters* 7, 1 (March 2015), 7–10.
[11] Sheng Ma et al. 2014. *Networks-on-Chip 1st ed.* Morgan Kaufmann, Burlington, MA.
[12] K.G. Shin John Hall and Jennifer Rexford. 1996. A Router Architecture for Real-Time Point-to-Point Networks. In *23rd Annual International Symposium on Computer Architecture*. IEEE.
[13] Alan Kaminsky and Stanislaw P. Radziszowski. 2008. A case for a parallelizable hash. In *MILCOM 2008. Military Communications Conference Proceedings*. IEEE.
[14] Gianluca Palermo Leandro Fiorin and Cristina Silvano. 2008. A security monitoring service for NoCs. In *CODES+ISSS '08. Proceedings of the 6th IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis*. ACM, New York, NY, 197–202.
[15] Indrakshi Ray Raman Adaikkalavan and Xing Xie. 2011. Multilevel Secure Data Stream Processing, In Data and Applications Security and Privacy XXV. DBSec 2011. *Lecture Notes in Computer Science* 6818 (2011), 122–137.
[16] Bruce Schneier. 1995. *Applied Cryptography: Protocols, Algorithms, and Source Code in C 2nd ed.* John Wiley & Sons, Inc., New York, NY.
[17] Adi Shamir. 1979. How to share a secret. *Communications of the ACM Magazine* 22, 11 (November 1979), 612–613.
[18] R.W. Yeung S.Y.R. Li and Ning Cai. 2003. Linear network coding. In *IEEE Transactions on Information Theory*, Vol. 49. IEEE, 371–381.
[19] Yao Wang and G. Edward Suh. 2012. Efficient Timing Channel Protection for On-Chip Networks. In *NOCS '12. Proceedings of NOCS 2012*. IEEE, 142–151.