HAA: Hardware-Accelerated Authentication for Internet of Things in Mission Critical Vehicular Networks

Ankush Singla, Anand Mudgerikar, Ioannis Papapanagiotou and Attila A. Yavuz

***Abstract*—Modern vehicles are being equipped with advanced sensing and communication technologies, which enable them to connect to surrounding entities. In military vehicular networks, it is vital to prevent adversaries from manipulating critical messages via cryptographic protection (e.g., digital signatures) and at the same time to minimize the impact introduced by crypto operations (e.g., delay). Hence, their communication must be delay-aware, scalable and secure.**

In this paper, we developed *Hardware-Accelerated Authentication* (HAA) that enables practical realization of delay-aware signatures for vehicular networks. Specifically, we developed a cryptographic hardware-acceleration framework for Rapid Authentication (RA) [1], which is a delay-aware offline-online signature scheme for command and control systems. We showed that HAA can significantly improve the performance of offline-online constructions under high message throughput, which is an important property for vehicular networks. *HAA-2048 (GPU)* is x18, x6, and x3 times faster than the current CPU implementation of RSA, ECDSA and RA, respectively, for the same level of security.

***Index Terms*—Authentication; hardware-acceleration; digital signatures; vehicular networks.**

1. Introduction

Secure vehicular networks will play a key role in tactical military systems by providing mobile and ad-hoc communication in battlefields [2]. To ensure reliable operation of such networks, the security must be guaranteed in a real-time and scalable manner [3]. In particular, achieving the immediate authentication of Command & Control (C&C) messages is a vital requirement to prevent adversaries from disrupting the network and inflicting catastrophic damage [1], [4].

- *A. Singla, Anand A Mudgerikar and I. Papapanagiotou are with the Center for Education and Research in Information Assurance and Security, Purdue University, West Lafayette, IN, 47907.*
- *I. Papapanagiotou is also with the Computer and Information Technology , Purdue University, West Lafayette, IN, 47907. E-mail: {amudgeri,asingla,ipapapan}@purdue.edu*
- *A. A. Yavuz is with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331. E-mail: attila.yavuz@oregonstate.edu*

Scheme	End-to-end Crypto Delay (msec)
RSA-2048 (CPU)	4
ECDSA-256 (CPU)	1.18
RA-2048 (CPU)	0.69
RA-2048 (CPU on SoC)	7.1
HAA-2048 (GPU)	**0.21**
HAA-2048 (GPU on SoC)	**2.6**

TABLE 1: Average end-to-end crypto delay comparison (signature generation plus verification time).
Implementation details are explained in detail in Section 6.

A broadcast authentication scheme permits each receiver in a large group to verify if the received message is intact and originated from the claimed sender [5]. Hence, it is a fundamental security service to protect C&C messages in such mission-critical vehicular networks [3], [6]. However, broadcast authentication is a challenging problem, especially for large and dynamic networks with strict delay-aware requirements [1].

Digital signatures are the most essential tools for broadcast authentication, since they provide scalability, public verifiability, and non-repudiation properties. Therefore, existing vehicular communication standards (e.g., [7], [8]) and many other critical applications (e.g., smart-grid [9]) rely on digital signatures. However, traditional signatures (e.g., ECDSA [10]) incur significant computation overhead, and therefore cannot meet the delay-aware requirements of mission-critical applications [11], [12]. Several specialized digital signatures and variants have been proposed to address this issue. They offer different computation, communication, storage, scalability and immediate verification trade-offs (we discuss related work in Section 2).

The offline-online signatures (e.g., [13]) offload costly signature generation operations to the offline phase, which permits a fast signature generation during the online phase. This is done by pre-computing and storing K signature tokens, which can be generated without knowing the actual message to be signed. These tokens are then used to sign K messages without performing any expensive operations. Hence, they are used for applications that require low end-to-end crypto delay but with small/moderate number of burst messages. Recently, an offline-online signature scheme called *Rapid Authentication* (RA) [1] was proposed, and was specifically designed for the authentication of C&C messages. RA exploits already existing semi-structures in certain

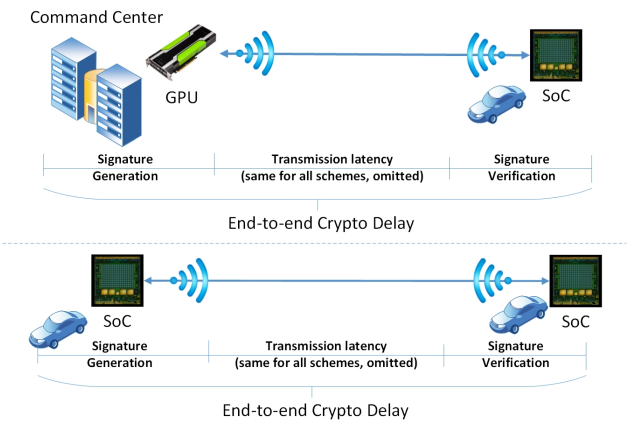

Figure 1: Vehicular Network

C&C messages and achieves the lowest end-to-to-end authentication delay among existing alternatives. Hence, RA is an ideal candidate for delay-aware applications but only with low/moderate message throughput.

Despite their great potential, offline-online signatures and RA have performance limitations if they are directly adapted to vehicular networks, which generally require very high message throughput (e.g., 3000 messages per sec according to IEEE WAVE standard [7]): If the application requires signing more than K messages per online cycle then the offline-online signature increases the computational cost in the incoming online cycle. That is, as opposed to traditional offline-online signature settings, high throughput applications may deplete pre-computed tokens at higher rates and therefore offline phase impedes the performance of online phase continuously. Despite considered as an ideal primitive for real-time authentication, this limitation of offline-online signatures must be mitigated to be used in vehicular networks having high message throughput. Hence, it is a challenging but important task to enable practical realization of these methods on mission critical vehicular networks.

In this paper, we address this problem by first analyzing the behavior of offline-online signatures under high throughput applications, and then developing a comprehensive cryptographic hardware-acceleration framework for RA that we call Hardware-Accelerated Authentication (HAA). HAA offers significant performance improvements over the prior realizations of RA as well as some standard signatures (e.g., ECDSA, RSA) for high throughput applications. We summarize our contributions as below:

• *Exploit the benefits of offline-online signatures for high throughput applications*: We investigate ways to optimize the algorithmic computations of the offline-online class of signature schemes towards achieving the highest possible bandwidth. Our main focus is to mitigate the impact of fast exhaustion of offline signatures, through leveraging software and hardware optimizations. We speficically target RA, as an exemplar offline-online signature scheme with applications to military vehicular networks.

• *A new end-to-end hardware acceleration framework for RA*: We identified that cryptographic hardware acceleration, with GPUs present on servers and Systems-on-Chip (SoC) in vehicles, are ideal tools to mitigate the aforementioned limitations of offline-online signatures. We evaluate and validate this claim by developing a cryptographic hardware acceleration framework for RA, which is called HAA. Table 1 demonstrates that HAA offers significant performance improvements over previous implementations of some standard signatures and RA on CPUs. For example, *HAA-2048 (GPU)* is approximately x18, x6, and x3 times faster than the CPU only implementation of RSA, ECDSA and RA, respectively, for the same level of security.

• *Special Hardware-Acceleration and Optimization Techniques*: We have employed various optimization techniques to accelerate the RA on GPUs in servers and in SoCs. We enable parallelism via algorithms such as Chinese Remainder Theorem and Montgomery Multiplication [14]. We leverage techniques like Constant Length Non-zero Window Technique, On-the-fly Token regeneration and batch processing to further accelerate the authentication.

The rest of the paper is organized as follows. Section 2 discusses related work and its limitations. Section 3 gives the primitives used by HAA. Section 4 describes our system, threat and security models. Section 5 introduces our proposed hardware-acceleration and optimization techniques. Section 6 presents our performance evaluation and comparison. Finally, Section 7 concludes this paper.

2. Related Work

Delay-Aware Broadcast Authentication: Message Authentication Code (MAC)s [14] rely on symmetric cryptography and are computationally efficient. However, they are not practical for large and distributed systems [15] due to key management issues (e.g., key distribution requirement). Standard digital signatures (e.g., [10], [16], [17]) achieve scalability and therefore are the security backbone of various vehicular communication standards (e.g., [7]). However, these schemes require expensive operations such as modular exponentiation or cryptographic pairing. Therefore, they are not suitable for time-critical authentication. Indeed, it has been shown that they introduce significant delay, which creates safety problems [11], [12].

Delayed key disclosure methods (e.g., [18]) achieve computationally efficiency and small tag sizes by introducing an asymmetry between signer and verifier via a time factor. However, these methods require packet buffering, and therefore cannot achieve immediate verification (which is vital for delay-aware authentication). One-Time Signatures (OTSs) (e.g., [3], [19]) offer very

fast signature generation and verification. However, they incur extremely large signature and public key sizes, and also public keys must be renewed frequently. Hence, OTSs are impractical for vehicular networks. Various customizations of traditional signatures (along with crypto pairing [17]) and OTSs for vehicular networks have been studied (e.g., [20]). However, these schemes suffer from computationally inefficiency (due to expensive pairing operations) and public key distribution issues (OTS-based approaches). The advantages and limitations of offline-online signatures (e.g., [13]) and RA [1] have been discussed in Section 1. While some drawbacks of TESLA have been discussed in [18], it requires use of a digital signature for multiple packages, and hence requires PKC unlike original TESLA would need. Also, since multiple packets are signed together, this still requires packet buffering and introduces packet loss risks. Therefore, TESLA is not ideal for time-critical applications, as it requires time sync., packet buffering and vulnerable to packet loss, despite use of hash chains up to certain degree.

Cryptographic Hardware Acceleration: Symmetric ciphers and RSA for SSL have been implemented, accelerated and benchmarked using GPUs [21], [22]. AES related GPU based acceleration techniques have been investigated in the CUDA framework [23]. GPUs require significant power and their size is relatively big for vehicular applications. Evidently, this is one of the reasons that development is moving towards integrating GPUs in SoCs like the Tegra K1 in modern vehicles like Audi and Tesla. To the best of our knowledge, there has been no prior work done on hardware acceleration of cryptographic methods in SoCs.

3. Preliminaries

We now present cryptographic primitives and hardware-acceleration methods that are used by our proposed methods.

3.1. Cryptographic Primitives

Notation: Operators $||$ and $|x|$ denote the concatenation operation and the bit length of variable x, respectively. $x \xleftarrow{\$} \mathcal{S}$ denotes that variable x is randomly and uniformly selected from set $\mathcal{S}$. $|\mathcal{S}|$ denotes the cardinality of set $\mathcal{S}$. $\{x_i\}_{i=0}^{l}$ denotes $(x_0, \ldots, x_l)$.

Rapid Authentication: Our proposed methods focus on RA, which is described in [1]. Due to space constraints, we only give a brief definition of RA algorithms and refer curios reader to [1] for the details.

1) $(sk, PK) \leftarrow RA.Kg(1^\kappa)$: Given the security parameter 1^κ, first generate a RSA private/public key pair as (sk', PK'). That is, randomly generate two large primes (p, q) and compute $n \leftarrow p \cdot q$. The public and secret exponents $(e, d) \in Z_n^*$ satisfy $e \cdot d \equiv 1 \bmod \phi(n)$, where $\phi(n) = (p-1)(q-1)$. Set $sk' \leftarrow (p, q, d)$ and $PK' \leftarrow (e, n)$. Also generate global randomness $\overline{r} \xleftarrow{\$} \{0,1\}^\kappa$. Set RA private/public key pair as sk=sk' and $PK \leftarrow (PK', \overline{r})$, respectively.

2) $\overline{sk} \leftarrow RA.Offline\text{-}Sig(sk, \overrightarrow{M})$: It takes a set of message components $\overrightarrow{M} \leftarrow \{M_0, \ldots, M_{L-1}\}$ and sk as the input. Each message component $M_{0 \le i \le L-1}$ is comprised of a set of sub-messages $m_{i,j}$ for $i = 1, \ldots, L-1$ and $j = 0, \ldots, |M_i| - 1$. The sub-messages of component M_0 are dedicated to time-stamp. The rest of components may include sub-messages such as C&C messages in M_1, receiver IPs in M_2 and so on. Given $(sk, \overrightarrow{M})$, the offline phase outputs a signature-message table $\overline{sk} = (\Gamma, \overrightarrow{\beta})$ as follows:

i) Given $m_{i,j} \in M_i$, compute a signature on $m_{i,j}$ as $s'_{i,j} \leftarrow [H(m_{i,j}||i)]^d \bmod n$ for $i = 0, \ldots, L-1$ and $j = 0, \ldots, |M_i| - 1$. The corresponding signature table of M_i is $\beta_i = \{s'_{i,j}\}_{i=0,j=0}^{L-1,|M_i|-1}$.

ii) Compute $\gamma_j \leftarrow [H(r_j||\overline{r})]^d \bmod n$, where $r_j \xleftarrow{\$} \{0,1\}^\kappa$ for $j = 0, \ldots, l'$. The random number/signature pair table is $\Gamma = \{r_j, \gamma_j\}_{j=0}^{l'}$.

iii) The private key of the online phase is $\overline{sk} \leftarrow (\Gamma, \overrightarrow{\beta})$, where $\overrightarrow{\beta} \leftarrow (\beta_0, \beta_1, \ldots, \beta_{L-1})$.

3) $\sigma \leftarrow RA.Online\text{-}Sig(\overline{sk}, \overrightarrow{m})$: Given an online message $\overrightarrow{m} = (m_0, \ldots, m_l) \in \overrightarrow{M}$ $(0 < l < L-1)$, fetch the corresponding signatures of sub-messages $(m_0, \ldots, m_l)$ from $\beta_0, \ldots, \beta_l$ as $(s'_0, \ldots, s'_l)$. Randomly fetch a pair (r, γ) from Γ and erase the selected pair from Γ. The signature on $\overrightarrow{m}$ is computed as:

$$s \leftarrow \gamma \cdot (\prod_{i=0}^{l} s'_i) \ , \ \sigma \leftarrow (r, s)$$

4) $c \leftarrow RA.Ver(PK, \overrightarrow{m}, \sigma)$: Given $(\overrightarrow{m}, \sigma)$, the verifier computes $m' \leftarrow [H(r||\overline{r}) \cdot \prod_{i=1}^{l} H(m_i||i)] \bmod n$ and $c \leftarrow \sigma^e \bmod n$. If $|r| = \kappa$ and $c = m'$ hold then return 1, else return 0.

Chinese Remainder Theorem (CRT): We leverage CRT [14] to accelerate RA on GPUs. We split a k-bit signature σ into two $k/2$ bit signatures σ_1 and σ_2.

$$\sigma_1 = M^{d \bmod p-1} \bmod p, \quad \sigma_2 = M^{d \bmod q-1} \bmod q$$

, where M is the message and (p, q) are the primes used in RSA. Later, we use the mixed radix conversion algorithm [24] to combine the two parts and recover the signature σ as $\sigma = \sigma_2 + [(\sigma_1 - \sigma_2).(q^{-1} \bmod p)].q$ These two parts are processed on separate threads in the GPU, which is significantly faster than the k-bit modular exponentiation.

Montgomery Multiplication: Modular multiplication is inefficient in the GPU implementations since it requires a trial division to determine the result and is not parallelizable. Montgomery multiplication algorithm is suitable for implementation in a GPU, since it does not require trial division and can be implemented in

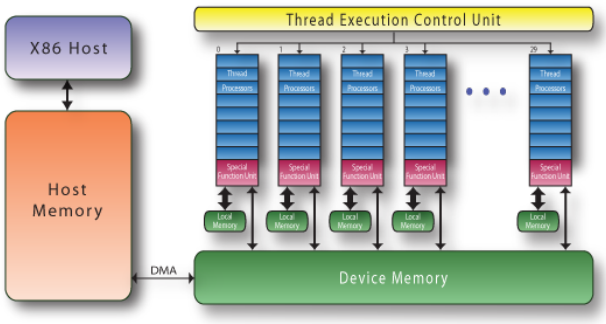

Figure 2: GPU architecture

parallel on separate words of the message. That is, given $a \cdot b \bmod n$, we first find two integers r^{-1} and n' using the Extended Euclidean Algorithm such that $rr^{-1} - nn' = 1$. We then transform $\overline{a} = ar \bmod n$ and $\overline{b} = br \bmod n$. Later, we compute $a \cdot b \bmod n$ by using montgomery reduction [14].

3.2. Hardware-Acceleration

Car manufacturers like Audi, Volkswagen, and BMW have already started rolling out car models with Graphics Processing Unit (GPU) enabled SoC capabilities. The most noted SoCs being used for providing automotive solutions are the Nvidia Jetson (with Nvidia Tegra GPU) and Qualcomm Snapdragon (with Adreno GPU). These are currently being used for various services like interactive HUD displays, navigation map services, entertainment services etc. These SoCs are leveraged for execution of the described offline-online authentication algorithm. These SoCs have added benefits of being energy efficient, having a small form factor and quite sturdy.

We implemented RA on the Nvidia Tesla K40c GPU with 2880 computing cores with 12 GB of GDDR5 device memory and memory bandwidth 288GB/sec. Our base system is equipped with Intel i7-5930K CPU/Clock Speed 3.5Ghz and 16GB DDR4 2400 MT/s (PC4-19200). This infrastructure represents a powerful command center setting. We also implemented the RA on an Nvidia Tegra K1 SoC [25], which has a 4-Plus-1 quad-core ARM Cortex A15 CPU with clock rate of 2.3 Ghz and an embedded GPU with 192 computing cores. This represents the remote vehicular environment. We specifically used the Tegra K1 as it is a common SoC in several commercial vehicles. A basic architecture diagram of an NVIDIA GPU is shown in Figure 2 [26].

4. Models

System Model: In our system model, there two types of entities: (i) Central entities such as static C&C centers or satellites, which are resourceful and equipped with GPUs. (ii) Mobile entities such as vehicles, which are equipped with SoC. The C&C messages can be broadcasted from central entities to vehicles (i.e., infrastructure-to-vehicle setting) or from one vehicle to other surrounding vehicles (i.e., vehicle-to-vehicle setting). In the first setting, the signature generation and verification phases are accelerated with server GPU and SoC-GPU, respectively. In the second setting, both phases are accelerated with SoC-GPUs.

Threat Model and Security Model: We assume the traditional PKC-based broadcast authentication threat and security model as considered in RA [1] as well as in many other signature schemes (e.g., [27]).

Threat Model: We assume a resourceful but Probabilistic Polynomial Time (PPT) bounded adversary $\mathcal{A}$ with the following properties: (i) passive attacks against output of cryptographic operations, (ii) active attacks including packet interception/modification. (iii) $\mathcal{A}$ aims to produce an existential forgery against the digital signatures broadcasted by the sender.

Security Model: Our methods boost the performance of RA via hardware-acceleration without modifying its security properties. Hence, they achieve the same level of security with RA, which is a strong security notation called Existential Unforgeability under Chosen Message Attack ($EU\text{-}CMA$) [28]). $EU\text{-}CMA$ security notation captures the threat model described above. As in prior work, we assume a public key infrastructure and certificates of public keys are all in place.

5. HAA Optimization Techniques

To accelerate the RA, we leveraged the parallel processing and optimization capabilities of GPUs both on server and embedded in the SoCs. We have made several optimizations to parallelize the individual steps of RA algorithms. We also used optimizations specific to the architecture of the GPU to realize the full potential of the available cores.

On-the-fly Token regeneration: HAA addresses the problem of exhaustion of pre-computed tokens by generating them in batches in real-time via GPU acceleration. When the pre-computed tokens are about to be exhausted, the offline stage of the RA is run on the GPU to the replenish them. This significantly improves the performance of the RA, especially for vehicular networks that have very high message throughput.

Batch Processing: Message components are processed in batches. That is, the crypto operations for multiple messages are performed concurrently in the GPU. This requires that a batch of messages be passed to the GPU, instead of a single message, for signing or verification.

Breakup of components into words: To optimize the throughput on the GPU, each message component is divided into words of size 32/64 bit, depending on the GPU capabilities. Each operation being run on a single thread is run over words rather than entire message components. We use standard multi-precision

algorithms [29] to represent and perform operations between large integers.

GPU warp size utilization: Warps are set of threads (generally 32) that are considered as one single execution unit inside a CUDA block. To gain maximum throughput from the GPU, it is necessary to attain the maximum number of active warps per streaming multiprocessor which is 64 in our case. We achieve this by adjusting the number of threads per block to the optimal value.

Memory latency vs GPU Occupancy: The size of the shared memory can limit the number of active warps on the GPU at a particular point in time by reducing the occupancy of the Streaming Multiprocessors (SM). The other limiting factor in the performance output is the number of reads and writes on the global memory on the device. We attain an optimum balance between the SM occupancy and the Global memory read/write latency through testing various permutations of memory allocations among the shared and global memory.

Constant Length Non-zero Window Technique: We scan the bits of the exponent from the least significant to the most significant. At each step, we compute a zero window or a non-zero window [30]. With the binary square-and-multiply method, we can process these windows and reduce the number of modular multiplications, making the exponentiation algorithm faster.

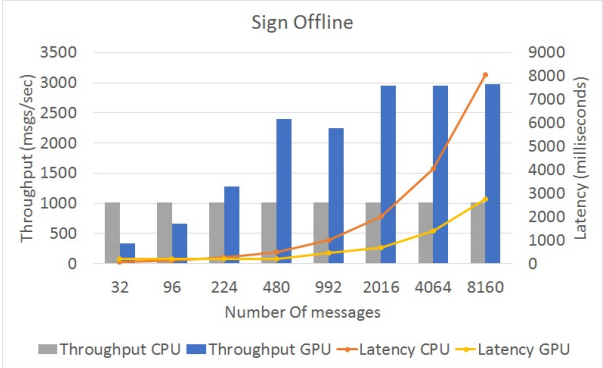

Figure 3: Server - Sign Offline

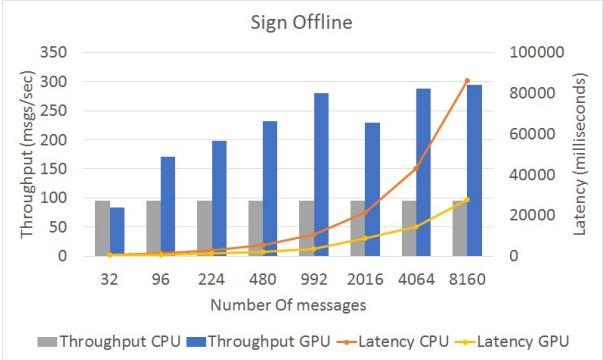

Figure 4: SoC - Sign Offline

6. Analysis and Comparison

We compare our scheme with some standard signature schemes such as RSA (2048-bit with $e = 2^{16} + 1$) and ECDSA (256-bit ECC-based signature) in terms of the end-to-end crypto delay in Table 1. According to BSI standards [31], 2048-bit RSA provides the same level of security as 256-bit ECDSA. We assume that we can pre-compute and store 4096 signatures. If the number of messages exceed 4096 then the offline tokens are replenished. This is a fair assumption in vehicular networks which have an average message throughput of 3000 messages per sec. For the processing of 8192 messages, RSA incurs an end-to-end delay of 4 msec/msg while ECDSA with pre-computation incurs an end-to-end delay of 1.18 msec/msg [32]. HAA outperforms both these schemes, x18 times better than RSA and x6 times better than ECDSA, respectively, with an end-to-end delay of 0.21 msec/msg seconds.

We focus on comparing the delay and throughputs between HAA and RA, as we find that RA is the best scheme among the existing alternatives in terms of end-to-end delay (0.69 msec/msg). Each component size is fixed to be 512 bytes for our experiments. We have used two configurations for our experiments, server and SoC settings. We have shown the evaluation results by comparing the GPU results to their CPU counterparts for the offline sign, online sign and verify stages of RA. We present the results in Figures: 3 - 8 both on the server and the SoC configuration with parameters $a = 32$, $e = 2^{16} + 1$ and $|n| = 2048$.

Performance analysis on Command Server: In the offline sign stage, for 8160 messages, we achieve x3 times more throughput with our GPU optimizations compared to CPU only implementations. In the online sign stage, we achieve high throughput gains upto x7 times. In the verify stage, the gain is around x1.3 times. These results are outlined in Figures: 3, 5 and 7, respectively.

In terms of execution time, the GPU can process

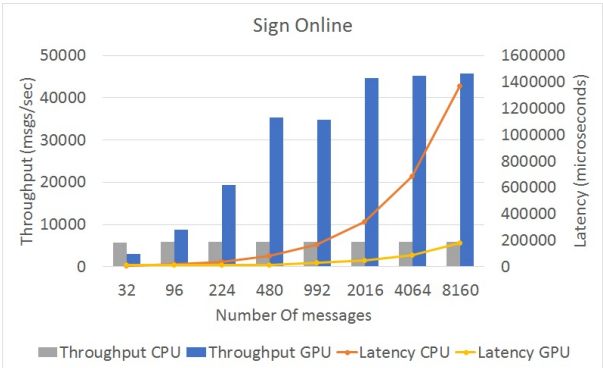

Figure 5: Server - Sign Online

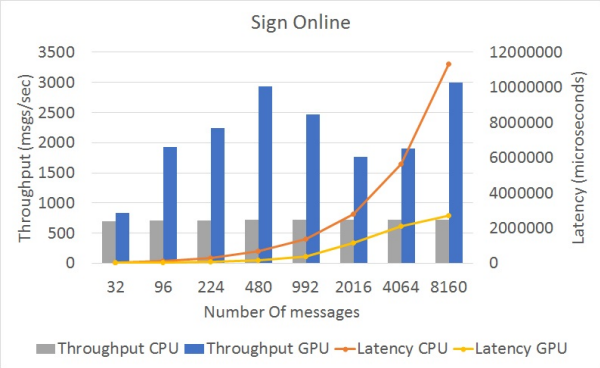

Figure 6: SoC - Sign Online

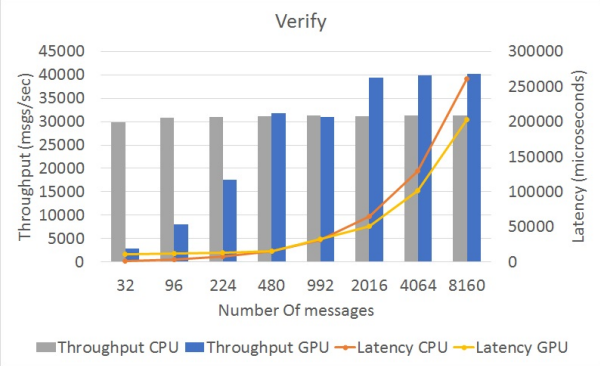

Figure 7: Server - Verify

a message in 0.337, 0.021, 0.024 milliseconds for the offline, online and verify stages of the algorithm respectively. This is approximately x2.91, x7.67, x1.28 times faster than the corresponding CPU execution times. The GPU gives a worse performance than the CPU if we are processing a very small number of messages. This is mainly due to the low clock speeds of the GPU cores as compared to the CPU and also due to the time taken to copy the data to the GPU memory from the CPU memory and back. We find that all the three stages Online, Offline and verify perform faster in GPU than

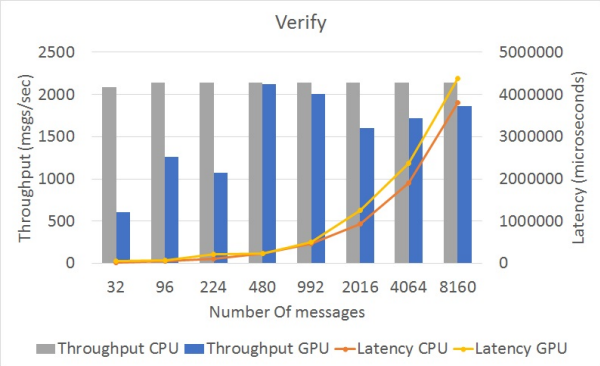

Figure 8: SoC - Verify

CPU for message batches greater than 32, 224 and 900 respectively.

Performance analysis on SoC: In the offline sign stage, for 8160 messages, we achieve x3.1 times more throughput with our GPU optimizations compared to CPU only implementations. In the online sign stage, we achieve high throughput gains upto x4.1 times. In the verify stage, the throughput of the GPU implementation hovers around the CPU throughput albeit a little less than it. The reason is explained later in the section. These results are outlined in Figures: 4, 6 and 8.

In terms of execution time, the Tegra GPU can process a message in 3.40, 0.33, 0.53 milliseconds for the offline, online and verify stages of the algorithm respectively. This is approximately x3.09, x4.16, x0.86 times the corresponding CPU execution times. We find the stages Online sign and Offline sign perform faster in GPU than CPU for message batches greater than 96 and 32 respectively. The GPU verify stage performs worse than the CPU on the Tegra for all message batch sizes. The reason for the lower gains in the verify stage for the GPU optimizations are as follows.

Double the copy operations in verify stage: In the verify stage, two GPU kernels (units of execution in GPU) are being executed, modular multiplication and modular exponentiation, as opposed to the online and offline stages where there is only a single GPU kernel being executed. Due to two GPU kernel being executed one after the other, there is a waiting time between memory copy operations from host memory to device memory and then back. This adversely impacts the overall execution time of the verify stage in GPU.

Modular exponentiation with public key exponent: RSA public key exponent is generally selected small (e.g., $e = 2^{16} + 1$) to enable fast signature verification. In this case, since $e << d$, the optimizations made in GPUs for speeding up modular exponentiation are less significant.

7. Conclusion

In this paper, we have introduced HAA, Hardware-Accelerated Authentication for mission critical vehicular networks. HAA exploits the limitations of prior offline-online signature schemes and reduces their end-to-end cryptographic delay by leveraging various optimization techniques realized on commercially available embedded hardware. That is, HAA harnesses the power of the thousands of cores available in GPUs, both at the C&C as well of smaller GPUs available in SoCs, to accelerate the performance of RA. Our experimental results demonstrates the potential of HAA, as it provides a speedup of x18, x6 and x3 than the corresponding CPU implementations of RSA, ECDSA and RA, respectively. Hence, HAA is a suitable choice for delay-aware authentication in military vehicular networks. In the future, we plan to explore the potential of HAA in

other application domains that may have similar delay-aware authentication needs such as smart-grids and drone swarms as a part of Internet of Things.

Acknowledgment

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU and Tesla K1 used for this research.

References

[1] A. A. Yavuz, "An efficient real-time broadcast authentication scheme for command and control messages," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 10, pp. 1733–1742, Oct 2014.

[2] C. Wilson, "Network centric operations: Background and oversight issues for congress, march 2007," http://www.globalsecurity.org/military/systems/ground/win-t.htm.

[3] Q. Wang, H. Khurana, Y. Huang, and K. Nahrstedt, "Time valid one-time signature for time-critical multicast data authentication," in *INFOCOM 2009, IEEE*, April 2009.

[4] J. Petit and Z. Mammeri, "Analysis of authentication overhead in vehicular networks," in *Wireless and Mobile Networking Conference (WMNC), 2010 Third Joint IFIP*, Oct 2010, pp. 1–6.

[5] A. Perrig, R. Canetti, D. Song, and D. Tygar, "Efficient authentication and signing of multicast streams over lossy channels," in *Proceedings of the IEEE Symposium on Security and Privacy*, May 2000.

[6] H. Guo, Y. Wu, F. Bao, H. Chen, and M. Ma, "UBAPV2G: A unique batch authentication protocol for vehicle-to-grid communications," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 707 –714, December 2011.

[7] "IEEE guide for wireless access in vehicular environments (WAVE) - architecture," *IEEE Std 1609.0-2013*, pp. 1–78, March 2014.

[8] "IEEE standard for wireless access in vehicular environments security services for applications and management messages," *IEEE Std 1609.2-2013 (Revision of IEEE Std 1609.2-2006)*, pp. 1–289, April 2013.

[9] Z. Lu, X. Lu, W. Wang, and C. Wang, "Review and evaluation of security threats on the communication networks in the smart grid," in *Military Communication Conference (MILCOM)*, November 2010.

[10] *ANSI X9.62-1998: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, American Bankers Association, 1999.

[11] J. Petit and Z. Mammeri, "Authentication and consensus overhead in vehicular ad hoc networks," *Telecommunication Systems*, vol. 52, no. 4, pp. 2699–2712, 2013.

[12] A. Vinel, C. Campolo, J. Petit, and Y. Koucheryavy, "Trustworthy broadcasting in IEEE 802.11p/WAVE vehicular networks: Delay analysis," *Communications Letters, IEEE*, vol. 15, no. 9, pp. 1010–1012, September 2011.

[13] A. Shamir and Y. Tauman, "Improved online/offline signature schemes," in *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '01. London, UK: Springer-Verlag, 2001, pp. 355–367.

[14] A. Menezes, P. C. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996, ISBN: 0-8493-8523-7.

[15] M. Luk, A. Perrig, and B. Whillock, "Seven cardinal properties of sensor network broadcast authentication," in *Proceedings of 4th ACM workshop on security of ad hoc and sensor networks*, ser. SASN '06. New York, NY, USA: ACM, 2006, pp. 147–156.

[16] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[17] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *Journal of Cryptology*, vol. 14, no. 4, pp. 297–319, 2004.

[18] A. Perrig, R. Canetti, D. Song, and D. Tygar, "Efficient and secure source authentication for multicast," in *Proceedings of Network and Distributed System Security Symposium*, February 2001.

[19] L. Reyzin and N. Reyzin, "Better than BiBa: Short one-time signatures with fast signing and verifying," in *Proceedings of the 7th Australian Conference on Information Security and Privacy (ACIPS '02)*. Springer-Verlag, 2002, pp. 144–153.

[20] X. Fan and G. Gong, "Accelerating signature-based broadcast authentication for wireless sensor networks," *Ad Hoc Networks*, vol. 10, no. 4, pp. 723–736, June 2012.

[21] J. Gilger, J. Barnickel, and U. Meyer, "GPU-acceleration of block ciphers in the OpenSSL cryptographic library," in *Information Security*. Springer, 2012, pp. 338–353.

[22] K. Jang, S. Han, S. Han, S. Moon, and K. Park, "Sslshader: Cheap SSL acceleration with commodity processors."

[23] Q. Li, C. Zhong, K. Zhao, X. Mei, and X. Chu, "Implementation and analysis of AES encryption on GPU," in *IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012*. IEEE, 2012, pp. 843–848.

[24] C. K. Koc, "High-speed rsa implementation," Technical Report, RSA Laboratories, Tech. Rep., 1994.

[25] "NVIDIA Jetson TK1 development kit," https://developer.nvidia.com/jetson-tk1, accessed: January 2015.

[26] M. Wolfe, "The pgi accelerator programming model on nvidia gpus," Tech. Rep., Jun. 2009. [Online]. Available: https://www.pgroup.com/lit/articles/insider/v1n1a1.htm

[27] A. A. Yavuz and P. Ning, "Self-sustaining, efficient and forward-secure cryptographic constructions for unattended wireless sensor networks," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1204–1220, 2012.

[28] J. Katz and Y. Lindell, *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.

[29] E. K. Donald, "The art of computer programming," *Sorting and searching*, vol. 3, pp. 426–458, 1999.

[30] C. K. Koç, "Analysis of sliding window techniques for exponentiation," *Computers & Mathematics with Applications*, vol. 30, no. 10, pp. 17–24, 1995.

[31] I. ECRYPT, "Yearly report on algorithms and keysizes. european network of excellence in cryptology ii," ICT-2007-216676. Available at http://www. ecrypt. eu. org/documents/D. SPA. 17. pdf, Tech. Rep.

[32] Shamus, "Multiprecision integer and rational arithmetic c/c++ library(MIRACL)," http://www.certivox.com/miracl/miracl-download/, 2014, [Online; accessed September 2014].