8-2016

Hardware accelerated authentication system for dynamic time-critical networks

Ankush Singla
Purdue University

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Ankush Singla

Entitled
HARDWARE ACCELERATED AUTHENTICATION FOR DYNAMIC TIME-CRITICAL NETWORKS

For the degree of Master of Science

Is approved by the final examining committee:

Elisa Bertino Co-chair	
Ioannis Papapanagiotou Co-chair	
Samuel S. J. Wagstaff	

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Elisa Bertino

Approved by: Eugene H. Spafford 7/19/2016
Head of the Departmental Graduate Program Date

HARDWARE ACCELERATED AUTHENTICATION SYSTEM FOR DYNAMIC TIME-CRITICAL NETWORKS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Ankush Singla

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

August 2016

Purdue University

West Lafayette, Indiana

Dedicated to my parents who have always encouraged and facilitated me in my endeavors.

ACKNOWLEDGEMENTS

I wish to gratefully acknowledge Dr. Ioannis Papapanagiotou and Prof. Attila Yavuz for their continued support and guidance throughout the duration of the project for this thesis. I would also like to acknowledge the insightful comments, feedback and the logistical support provided by Prof. Elisa Bertino which have helped shape the direction of the project. Also notable is the contribution and support of my co-researcher Anand Mudgerikar.

TABLE OF CONTENTS

Page

LIST OF ABBREVIATIONS

IoV – Internet of Vehicles

RA – Rapid Authentication

HAA – Hardware Accelerated Authentication

CPU – Central Processing Unit

GPU – Graphical Processing Unit

SoC – System on Chip

CUDA – Compute Unified Device Architecture

IOT – Internet of Things

GPGPU - General Purpose Graphic Processing Units

SIMD - Single Instruction Multiple Data

BSM – Basic Safety Message

AES – Advanced Encryption Standard

ABSTRACT

Singla, Ankush. M.S., Purdue University, August 2016. Hardware Accelerated Authentication System for Dynamic Time-Critical Networks. Major Professor: Elisa Bertino.

The secure and efficient operation of time-critical networks, such as vehicular networks, smart-grid and other smart-infrastructures, is of primary importance in today's society. It is crucial to minimize the impact of security mechanisms over such networks so that the safe and reliable operations of time-critical systems are not being interfered.

Even though there are several security mechanisms, their application to smart-infrastructure and Internet of Things (IoT) deployments may not meet the ubiquitous and time-sensitive needs of these systems. That is, existing security mechanisms either introduce a significant computation and communication overhead, or they are not scalable for a large number of IoT components. In particular, as a primary authentication mechanism, existing digital signatures cannot meet the real-time processing requirements of time-critical networks, and also do not fully benefit from advancements in the underlying hardware/software of IoTs.

As a part of this thesis, we create a reliable and scalable authentication system to ensure secure and reliable operation of dynamic time-critical networks like vehicular networks through hardware acceleration. The system is implemented on System-On-Chips (SoC) leveraging the parallel processing capabilities of the embedded Graphical Processing Units (GPUs) along with the CPUs (Central Processing Units). We identify a set of cryptographic authentication mechanisms, which consist of operations that are highly parallelizable while still maintain high standards of security and are also secure against various malicious adversaries. We also focus on creating a fully functional prototype of the system which we call a “Dynamic Scheduler” which will take care of scheduling the messages for signing or verification on the basis of their priority level and the number of messages currently in the system, so as to derive maximum throughput or minimum latency from the system, whatever the requirement may be.

CHAPTER 1. INTRODUCTION

1.1 Introduction

According to National Highway Traffic Safety Administration (NHTSA) there were 32,479 highway fatalities in the year 2011 (Harding, et al., 2014). Another NHTSA study which accesses the readiness of Vehicle-to-Vehicle (V2V) communication technology for practical application, estimates that just two of the many possible V2V applications Intersection Movement Assist (IMA) and Left Turn Assist (LTA) can save 49 to 1,083 lives by preventing these kind of accidents. These kind of applications can harness vehicular networks to communicate with other vehicles as well as control-centers through roadside infrastructure.

The secure and efficient operation of time-critical networks, such as the aforementioned vehicular networks as well as smart-grid and other smart-infrastructures, is of primary importance in today's society. It is crucial to minimize the impact of security mechanisms over such networks so that the safe and reliable operations of time-critical systems are not being interfered. For instance, if the delay introduced by the crypto operations negatively affects the time available for braking the car before a collision, a car may not be able to safely stop in time. Similarly, smart-grid networks require phasor measurement units to authenticate security sensitive measurements with a very high throughput (e.g., 1000-2000 messages per second). Finally, the security mechanisms for time-critical networks must be highly scalable, since these systems have very large numbers of connected components.

Even though there are several security mechanisms, their application to smart-infrastructure and Internet of Things (IoT) deployments may not meet the ubiquitous and time-sensitive needs of these systems. That is, existing security mechanisms either introduce a significant computation and communication overhead, or they are not scalable for a large number of IoT components. In particular, as a primary authentication mechanism, existing implementations of digital signatures cannot meet the real-time processing requirements of time-critical networks, and also do not fully benefit from advancements in the underlying hardware/software of IoTs.

The goal of this research is to identify a suite of extremely fast digital signatures and implement them using hardware-acceleration, to ensure delay-aware authentication in time-critical networks. This project analyzes and implements real-time digital signature schemes, and then pushes the performance to the edge via cryptographic hardware-acceleration.

The end result of this research would be to show significantly better throughput and lower latency of the accelerated authentication algorithms when compared to existing implementations, along with an ability to handle and schedule messages according to their priority level.

1.2 Scope

The scope of this thesis includes the following:

1. Identify viable authentication mechanisms for hardware-acceleration and implement these algorithms on server-grade GPUs using their parallel processing capabilities.
2. Implement these algorithms on System-On-Chips containing GPUs.
3. Create a Dynamic Scheduler to schedule and manage incoming messages and more importantly leverage the processing capabilities of both CPUs and GPUs at the same time.
4. Create graphs and charts showing performance comparisons of these implementations with state-of –the art authentication implementations.

1.3 Assumptions

For successful implementation of our system in the real world we are assuming the following:

1. There exists a working fault-tolerant and high-bandwidth wireless network for these authentication messages and signatures to travel from one node to others.
2. There exists a fully functional and reliable key distribution mechanism already in place and the nodes have their respective private-public keys in place before our system starts working.

3. The authentication system is the highest priority program running on the system. All other programs, for e.g. entertainment or other communication systems can take a back-seat when the authentication program needs the hardware.

1.4 Limitations

In addition to relying upon the above assumptions to be in place before our system is operational, it suffers from the following limitations:

1. We do not focus on encrypting the messages while in transit, so as to protect them from the eyes of a malicious adversary.
2. The receiving nodes may not have a way to get the public keys of the transmitting nodes while they are moving at a fast speed.
3. Fault tolerance needs to be built into the system. There is no way now for knowing whether the message was received by the intended node and passed verification or not.

CHAPTER 2. REVIEW OF RELEVANT LITERATURE

2.1 Introduction

In the following sections, we give an understanding of the state-of-the art authentication algorithms being used and a preliminary analysis including the benefits and the shortcomings of the same. In the section 2.3, we describe the existing hardware-accelerated implementations of various cryptographic mechanisms.

2.2 Existing Authentication Mechanisms

We outline the advantages and limitations of authentication mechanisms that are most relevant to our work.

Message Authentication Codes and Standard Digital Signatures: Symmetric crypto based authentication mechanisms rely on Message Authentication Code (MAC) (Menezes, Oorschot, & Vanstone, 1996). Despite their simplicity and computational efficiency, MAC-based methods are not practical for broadcast authentication in large-scale distributed systems [(Luk, Perrig, & Whillock, 2006), (Boneh, Lynn, & Shacham, 2004)], as they require a pairwise key distribution requirement among all the signers and verifiers. These methods also cannot achieve non-repudiation and public verifiability. Digital signatures (e.g., RSA (Rivest, Shamir, & Adleman, 1978), ECDSA (American

Bankers Association, 1999)) rely on Public Key Infrastructures (PKIs) (Menezes, Oorschot, & Vanstone, 1996), which make them publicly verifiable and scalable for large systems. Hence, they are considered as a primary authentication mechanism for large-scale delay-aware systems such as vehicular networks and smart-grid systems. For instance, the vehicular WAVE architecture mandates (IEEE, 2014) the use of PKI mechanisms to digitally sign critical messages [(IEEE, 2013), (Vinel, Campolo, Petit, & Koucheryavy, 2011), (Mammeri, Petit, & Zoubir, 2013)]. Smart-grids require ubiquitously deployed phasor measurement units to authenticate sensitive measurements with a very high throughput (e.g., up to 120 messages per second) [(Robin Berthier, 2013), (IEEE, 2011)]). Despite their scalability, standard digital signature schemes require several expensive operations such as modular exponentiation and pairing (e.g., BLS (Boneh, Lynn, & Shacham, 2004)). Therefore, they are not suitable for time-critical authentication. It has been shown that they introduce significant delays which for safety reasons are unacceptable in time-critical networks such as vehicular networks [(Mammeri & Zoubir, 2010), (Mammeri, Petit, & Zoubir, 2013), (Vinel, Campolo, Petit, & Koucheryavy, 2011)].

Delayed Key Disclosure and Amortized Signatures: Delayed key disclosure methods [(Perrig, Canetti, Song, & Tygar, 2000) , (Perrig, Canetti, Song, & Tygar, 2001), (Perrig, Canetti, Song, & Tygar, 2002)] are efficient and compact as they introduce 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). Signature amortization methods (e.g., (Song, Zuckerman, & Tygar, 2002), (Lysyanskaya, Tamassia, & Triandopoulos, 2004), (Wong & Lam, 1999), (Miner & Staddon, 2001)) compute a signature over a set of messages instead of individual message. Hence, the cost of signature generation and verification is amortized over multiple messages. However, these methods require packet buffering and introduce packet loss risk due to the use of hash-chains.

Cryptographic Hardware Acceleration with GPUs: Symmetric ciphers and RSA for SSL have been implemented, accelerated and benchmarked using General Purpose GPUs (GPGPUs) [(Gilger, Barnickel, & Meyer, 2012), (Jang, Han, Han, Moon, & Park, 2011)]. AES related GPU based acceleration techniques have been investigated in the CUDA framework [(Li, Zhong, Zhao, Mei, & Chu., 2012), (Iwai, Kurokawa, & Nisikawa, 2010)]. There are numerous other studies that leverage Graphic Processing Units (GPU)s, but we limit the discussion because of space constraints. Discrete GPUs have several limitations such as high operating power usage and size, hence they have not been deployed in cars. NVIDIA, as a major GPU manufacturer and our collaborator in this work, has moved towards implementing the Tegra SoCs, which are being deployed in newer Audi and Tesla models. In [(Yan, Shi, & Fei, 2009), (Mane, Judge, & Schaumont, 2011)], modular arithmetic accelerations with embedded Digital Signal Processing (DSP) cores in SoC have been developed. In (Pabbuleti, Mane, Desai, Albert, & Schaumont, 2013), an ECC implementation for the Venom (NEON) co-processor in Qualcomm’s Scorpion (ARM) Central Processing Unit (CPU), with Streaming SIMD Extensions (SSE2) instruction-set in Intel’s Atom CPU, was developed. However, none of these studies has explored the collaboration between GPU and CPU in modern SoC co-processor architectures. To the best of our knowledge, only the work in (Wang, Xiong, Yun, & Cavallaro, 2013), which is outside of the crypto domain, has explored the GPU/CPU coordination. In (Glas, Sander, Stuckert, M¨uller-Glaser, & Becker, 2011), an ECDSA acceleration on reconfigurable hardware has been proposed, which offers some performance improvements. However, such approaches do not incorporate the ARM-based hardware design architecture of a SoC. Our proposed methods in SoCs are built using systems already existing in cars, and also show better performance.

In (Singla, Mudgerikar, Papapanagiotou, & Yavuz, 2015), we describe Hardware-Accelerated Authentication (HAA) derived from RA (Rapid Authentication) scheme (Yavuz, 2014) and showed its application to vehicular networks. HAA significantly improves the performance of offline-online constructions. That is, HAA-2048 (GPU) is x18, x6, and x3 times faster than the current CPU implementation of RSA, ECDSA and RA (Yavuz, 2014), respectively, for the same level of security. Figures 3, 4 and 5 indicate results when these optimizations are performed on an Nvidia Tegra K1 SoC with 192 Nvidia CUDA Cores on its GPU. Figures 6, 7 and 8 indicate results when these optimizations are performed on an Nvidia Tesla K40c server-grade GPU containing 2880 Nvidia CUDA Cores. Figure 9 illustrates the results when the CPU cores and GPU are used in conjunction to divide the processing workload.

2.3 Cryptographic Hardware Acceleration with GPUs

Symmetric ciphers and RSA for SSL have been implemented, accelerated and benchmarked using General Purpose Graphic Processing Units (GPGPUs) [30, 43]. AES-related GPU-based acceleration techniques have been investigated for the Compute Unified Device Architecture (CUDA) framework in (Li, Zhong, Zhao, Mei, & Chu., 2012) and (Iwai, Kurokawa, & Nisikawa, 2010). Discrete GPUs have several limitations such as high operating power usage and size, hence they have not been deployed in cars. NVIDIA, as a major GPU manufacturer, has moved towards implementing the Tegra SoCs, which are being deployed in Audi and Tesla models. None of these prior studies have explored the collaboration between GPU and CPU in modern SoC co-processor architectures. To the best of our knowledge, only the work in (Wang, Xiong, Yun, & Cavallaro, 2013), which is outside of the crypto domain, has explored the GPU/CPU coordination in an older version of SoC. In (Glas, Sander, Stuckert, M¨uller-Glaser, & Becker, 2011), an ECDSA acceleration on reconfigurable hardware has been proposed, which offers some performance improvements.

CHAPTER 3. FRAMEWORK AND METHODOLOGY

The proposed authentication system involves a network of individual nodes communicating among each other and with a central command-center. The authentication algorithm will be processed on System-On-Chips (SoC) installed inside the nodes. For the purposes of this section, we will talk about the vehicular networks, but any similar dynamic and time-critical network can replace them. We also discuss the parallelization techniques used in our recent paper describing Hardware Accelerated Authentication (Singla, Mudgerikar, Papapanagiotou, & Yavuz, 2015) using Rapid Authentication (Yavuz, 2014)

3.1 The Network Nodes

The authentication apparatus described is perfectly suitable for deployment in real-time and critical networks, such as Vehicular Networks. The vehicles will serve as individual nodes in the vast peer-to-peer internetwork of vehicles, command-centers, traffic lights and other relevant Internet of Thing devices in the immediate vicinity. The vehicles will have to obtain some kind of a cryptographic certificate for identification of an authorized user. This will be provided by some existing Public Key Infrastructure (PKI) authority, the details of which are outside the purview of this thesis.

The vehicles in the network are expected to generate messages at a really high rate due to the advanced sensing technologies and latest interconnected services being mainstreamed. As and when the autonomous capabilities are increased in these vehicles, the reliance on these sensors will increase manifold to navigate the roads and highways safely and quickly. This will introduce algorithms, which rely on the spatial positioning of the vehicles relative to each other, so as to avoid any collisions, bottlenecks in traffic and minimize the journey duration. This will further increase the rate of generation of messages and will burden any authentication schemes deployed to process such huge number of messages, almost in real-time.

3.2 System-on-Chip requirements

Big car manufacturers like Audi, Volkswagen, and BMW have already started rolling out car models with Graphics Processing Unit (GPU) enabled System-on-chip (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 and much more.

We will particularly focus on the Nvidia Tegra K1 SoC for the purposes of this research. Nvidia Tegra is the most logical first choice due to its easy to program CUDA API, development tools integration into Visual Studio and Eclipse, results visualizer to locate the performance bottlenecks and widespread availability. These SoCs have added benefits of being energy efficient, having a small form factor and quite sturdy.

3.3 Hardware Accelerated Rapid Authentication Scheme

The Hardware Accelerated Authentication scheme a.k.a. HAA (Singla, Mudgerikar, Papapanagiotou, & Yavuz, 2015) is derived from RA (Rapid Authentication) scheme (Yavuz, 2014). RA exploits already existing structures in the vehicular communication messages to enable pre-computation for signature schemes like RSA. The main idea of RA is to leverage the fact that the numbers of possible sub-messages in a command and control message are limited. Hence, it is possible to pre-compute and store a RSA signature on each of those sub-messages during the offline phase. When a message is to be signed in the online phase, the signer combines individual RSA signatures of relevant sub-messages via Condensed-RSA (Tsudik & Mykletun, 2006), which requires only a few modular multiplications. The verification of this signature is also efficient, as it requires a standard RSA signature verification plus a few modular multiplications.

The Rapid Authentication scheme is modified in order to be implemented on GPUs to produce the faster and more scalable HAA. Various optimizations in both the RA scheme and the RSA algorithm have been made to maximize the number of parallel operations, in order to harness the power of the GPUs.

The other optimization techniques used in the implementation are described as follows:

3.3.1 Inter Message Parallelization

In our HAA scheme, message components are processed in batches both in the online and offline phase. Each message as defined in the RA scheme is processed in parallel. This means that multiple threads perform the signing and verifying operations for multiple messages at once concurrently in the GPU.

3.3.2 Intra Message Parallelization

Besides processing messages in batches, RSA signature and verification algorithms are optimized to improve performance in GPUs by using the Chinese remainder theorem (only signing) and other optimizations like Montgomery multiplication and Sliding Window techniques. These optimizations have been implemented in (Jang, Han, Han, Moon, & Park, 2011) for RSA encryption and decryption. These techniques form the building blocks for the HAA implementation.

3.3.3 Restoring depleted random masks using Offline Stage while signing

In (Yavuz, 2014) and most of the online/offline schemes, it is assumed that the messages generated in the offline stage are pre-computed and it will not affect the performance of the online stage. This is not true for a real world scenario where the pre-computed random masks included with each message might get exhausted really quickly which will then have to be generated in real time in the online phase. The HAA scheme addresses this problem by generating the pre-computed messages (offline stage) on the GPUs in batches in real time using GPU acceleration. This will significantly improve the performance of the scheme in real world scenarios where the number of messages generated per unit time is pretty high.

3.3.4 Other Hardware Acceleration 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 algorithm. We also used optimizations specific to the architecture of the GPU to realize the full potential of the available cores.

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~\cite{mp} 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 (Koç, 1995). With the binary square-and-multiply method, we can process these windows and reduce the number of modular multiplications, making the exponentiation algorithm faster.

3.4 Dynamic Scheduler

We propose a “Dynamic Scheduler” prototype, which is a system capable of authenticating messages on-the-fly according to the priority requirements of the individual messages. The components of the prototype are detailed below:

3.4.1 Message Structure

SAE J2735, published in November 2009 specifies Basic Safety Message (BSM) as a standard message structure for vehicular networks.

According to a recent NHTSA report on readiness for V2V technology (Harding, et al., 2014) “the BSM is used to exchange safety data regarding vehicle state. The message is broadcast routinely to surrounding vehicles with a variety of data content. The BSM is split into two parts to guarantee that the core information for vehicle safety (Part I) has priority and is transmitted more often. It also minimizes the amount of data communicated (most of the time) between devices, helping to reduce channel congestion”. A message structure is shown in the Fig. 2 in Appendix.

3.4.2 Concurrent Priority Queue

A Concurrent Priority Queue will be created for storing the messages as they come and feed them to the scheduler when ready. The queue would be a FIFO (First-in-First-out) queue but dependent on the priority levels of the respective messages. Since there will be a huge number of messages generated per second in a real-time vehicular network, there will still be certain latency (however small) introduced by the authentication operations. Some kinds of messages like a certain vehicle crash, losing steering control, brake failure cannot afford being buffered and waiting for being processed. While some other not so critical messages like location updates, weather details, news services can afford a certain minimal amount of latency without affecting any safety of the vehicle.

So, a priority queue is being proposed which will contain the messages according to their priority. The queue as described in Figure 1 will be a First-in-First-out (FIFO) data structure where the processor(s) just pick up the next message from the front of the queue and run the authentication algorithm on it. The incoming messages however get inserted at their respective positions in the priority queue according to their priority. The messages will contain a priority field inside the definition which will be predefined. The immediate messages will have priority assigned as 0. The other non-immediate messages can be assigned priorities according to their urgency. The lower the priority number the higher the priority.

As and when the messages are generated the system will look for the suitable location to insert the message. The messages will be inserted after the messages of priority equal to its own but before the messages having priority lesser than it. This will allow non-pre-emptive preferential processing of the messages according to their urgency and critical nature. The queue should be able to:

a. Take messages from multiple sources.
b. Keep a count of the number of messages.
c. Keep a count of immediate/non-immediate messages.
d. The queue should put in messages according to their priority level. The highest priority messages should be added in the front.

3.4.3 Scheduler

We define a scheduler for the process, which will decide which processor CPU/GPU will process the messages in the queue and the amount of messages to be fed to the GPU at once. We identify a threshold value \tau as the minimum number of messages fed to the GPU at once, when it starts outperforming the CPU in terms of the throughput obtained while cryptographically processing the messages. This threshold value will be different for each model of a SoC because the different CPUs and GPUs embedded into those will give different performance results according to their computing capabilities. Once the threshold value is generated, it will be fixed for the lifetime of that SoC, as this will not change.

The dynamic scheduler proposed will check the number of the non-immediate (priority ≠ 0) messages in the queue. If it comes out to be greater than \tau, the scheduler will hand over all of these to the GPU to authenticate these messages in a batch. This decision will be required to be made on 2 occasions:

1. When the GPU has processed the message batch assigned to it and has just become idle.
2. Whenever a new non-immediate message is inserted.

There are a few things to consider here. The CPU no matter what will always process the immediate messages. We assume that the immediate messages, for e.g., vehicle crash, losing steering control, brake failure will be a rare occurrence and never exceed the \tau value at one time even under extreme circumstances. The other thing to consider is that the length of the queue will have to be defined so as to accommodate any number of messages that may be generated in a certain period of time. This will also depend on the execution speed of the processors deployed in the vehicle but it can be taken as a fairly huge constant value. A scheduler capable of taking the messages from the queue and should be able to:

a. Make a decision on the appropriate processor to feed them to the e.g. CPU, GPU based on an optimum decision tree.
b. Keep in view the minimum number of messages after which the GPU starts giving faster throughput than the CPU.
c. Schedule the immediate messages for minimum latency.
d. Schedule the messages on the basis of priority level.

e. Schedule the messages for Signature or Verification algorithm based on the requirement.

3.4.4 Network Sender/Receiver

A Network Sender/Receiver for sending and listening the messages over the network. It should be able to:

a. Take messages from the scheduler and send the messages to the desired recipient.
b. Listen to the messages over a designated port and pass it along to the priority queue for signature verification.
c. Scale for the type of networks it is in i.e. should be able to handle high volume of messages at a given time without any loss.

CHAPTER 4. RESULTS AND ANALYSIS

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 (ECRYPT, II, 2007), 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 exceeds 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/message while ECDSA with pre-computation incurs an end-to-end delay of 1.18 msec/message (Shamus, MIRACL). 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/message seconds.

We focus on comparing the delay and throughputs between HAA and RA, as we found out that RA is the best scheme among the existing alternatives in terms of end-to-end delay (0.69 msec/message). 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.

Test Infrastructure: One is a server configuration with Nvidia Tesla K40c GPU with 2880 cores and a Base clock rate of 745 Mhz. It has an Intel i7-5930K CPU with a clock rate of 3.50 GHz. The Tesla K40c has the Kepler architecture and a CUDA compute capability of 3.5.

The other is a System-On-Chip (SoC) configuration with the Nvidia Jetson TK1 development kit, which has an Nvidia Tegra K1 chipset. The Tegra K1 chipset has an embedded Kepler GPU with 192 CUDA cores and a base clock rate of 852 Mhz. It also has a 4-Plus-1 quad-core ARM Cortex A15 CPU with clock rate of 2.3 Ghz. The GPU in the Tegra is based on the Kepler architecture, which is the same architecture as the Tesla K40c used for the server configuration. It has a compute capability of 3.2.

TABLE 1: Average end-to-end crypto delay comparison (signature generation plus verification time).

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

4.1 HAA (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 up to x7 times. In the verify stage, the gain is around x1.3 times. These results are outlined in Figures: 6, 7 and 8 respectively.

In terms of execution time, the GPU can process 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 CPU for message batches greater than 32, 224 and 900 respectively.

4.2 HAA (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 up to 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 Appendix Figures: 3, 4 and 5.

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.

4.3 Dynamic Scheduler (Server)

In the offline sign stage, for 4096 messages, we achieve x4 times more throughput with 8 CPU threads instead of 1. For 16 CPU threads we get around x5 times the throughput as compared to just 1 CPU thread. This is expected, as we are using an Intel i7-5930K CPU with 6 physical and 12 logical cores. Multithreading allows us to put all of those cores to use. This performance gain almost plateaus after 16 threads, as there are only so many CPU cores for processing. In the online sign stage, we achieve similar throughput gains of x4 and x5 for 8 and 16 CPU threads respectively as compared to just a single CPU thread. The verification stage results are also similar. These results are outlined in Figures: 9, 10 and 11 respectively.

Moving on to the results where we use CPU and GPU at the same time we gain more performance benefits as expected. At its highest level i.e. when using 16 CPU threads and GPU together we get x9 times the performance as compared to a single CPU thread and x3 times the performance when compared to only GPU in the offline stage. We get almost similar results for the online signing and the verification stage.

BIBLIOGRAPHY

BIBLIOGRAPHY

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

Boneh, D., Lynn, B., & Shacham, a. H. (2004). Short signatures from the Weil pairing. *Journal of Cryptology*, 297-319.

ECRYPT, II. (2007). *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.

Gilger, J., Barnickel, J., & Meyer, U. (2012). GPU-acceleration of block ciphers in the OpenSSL cryptographic library. *Information Security* (pp. 338–353). Springer.

Glas, B., Sander, O., Stuckert, V., M¨uller-Glaser, K. D., & Becker, J. (2011). Prime field ECDSA signature processing for reconfigurable embedded systems. *International Journal of Reconfigurable Computing.*

Harding, J., Powell, G. R., Yoon, R., Fikentscher, J., Doyle, C., Sade, D., . . . Wang, J. (2014). *Vehicle-to-vehicle communications: Readiness of V2V technology for application (Report No. DOT HS 812 014).* Washington DC: NHTSA.

IEEE. (2011). IEEE standard for synchrophasor measurements for power systems. *IEEE Std C37.118.1-2011*.

IEEE. (2013, April). 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), pages 1-289*.

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

Iwai, K., Kurokawa, T., & Nisikawa, N. (2010). AES encryption implementation on CUDA GPU and its analysis. *First International Conference on Networking and Computing (ICNC)* (pp. 209-214). IEEE.

Jang, K., Han, S., Han, S., Moon, S. B., & Park, K. (2011). SSLShader: Cheap SSL Acceleration with Commodity Processors. *NSDI.*

Koç, C. K. (1995). Analysis of sliding window techniques for exponentiation. *Computers & Mathematics with Applications*, 17-24.

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

Luk, M., Perrig, A., & Whillock, B. (2006). Seven cardinal properties of sensor network broadcast authentication. *Proceedings of 4th ACM workshop on security of ad hoc and sensor networks, SASN '06* (pp. 147-156). New York, USA: ACM.

Lysyanskaya, A., Tamassia, R., & Triandopoulos, N. (2004). Multicast authentication in fully adversarial networks. *IEEE Symposium on Security and Privacy*, (pp. 241-253).

Mammeri, J. P., & Zoubir. (2010). Analysis of authentication overhead in vehicular networks. *Wireless and Mobile Networking Conference (WMNC), 2010 Third Joint IFIP*, (pp. 1-6).

Mammeri, Petit, J., & Zoubir. (2013). Authentication and consensus overhead in vehicular ad hoc networks. *Telecommunication Systems, 52(4)*, (pp. 2699-2712).

Mane, S., Judge, L., & Schaumont, P. (2011). An integrated prime-field ECDLP hardware accelerator with high-performance modular arithmetic units. *International Conference on Reconfigurable Computing and FPGAs (ReConFig)* (pp. 198–203). IEEE.

Menezes, A., Oorschot, P. C., & Vanstone, S. (1996). *Handbook of Applied Cryptography ISBN: 0-8493-8523-7*. CRC Press.

Miner, S., & Staddon, J. (2001). Graph-based authentication of digital streams. *Proceedings of the IEEE Symposium on Security and Privacy*, (pp. 232-246).

Pabbuleti, K., Mane, D., Desai, A., Albert, C., & Schaumont, a. P. (2013). SIMD acceleration of modular arithmetic on contemporary embedded platforms. *High Performance Extreme Computing Conference (HPEC)* (pp. 1–6). IEEE.

Perrig, A., Canetti, R., Song, D., & Tygar, a. D. (2000, May). Efficient authentication and signing of multicast streams over lossy channels. *Proceedings of the IEEE Symposium on Security and Privacy*.

Perrig, A., Canetti, R., Song, D., & Tygar, D. (2001, February). Efficient and secure source authentication for multicast. *In Proceedings of Network and Distributed System Security Symposium*.

Perrig, A., Canetti, R., Song, D., & Tygar, D. (2002). The TESLA broadcast authentication protocol. *RSA Cryptobytes*.

Rivest, R., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* (pp. 120-126). ACM.

Robin Berthier, J. G. (2013). Reconciling security protection and monitoring requirements in advanced metering infrastructures. *IEEE SmartGridComm* (pp. 450-455). IEEE.

Rogaway, P., & Bellare, M. (1996). The exact security of digital signatures: How to sign with RSA and Rabin. *Proceedings of the 15th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '96)* (pp. 399-416). Springer-Verlag.

Shamus. (MIRACL). *Multiprecision Integer and Rational Arithmetic C/C++ Library(MIRACL)*. Retrieved from https://github.com/miracl/MIRACL.

Singla, A., Mudgerikar, A. A., Papapanagiotou, I., & Yavuz, A. A. (2015). HAA: Hardware-accelerated authentication for internet of things in mission critical vehicular networks. *IEEE International Conference for Military Communications, 2015 (MILCOM '15)* (pp. 1-7). IEEE.

Song, D., Zuckerman, D., & Tygar, J. (2002). Expander graphs for digital stream authentication and robust overlay networks. *IEEE Symposium on Security and Privacy*.

Tsudik, G., & Mykletun, E. (2006). Aggregation queries in the database-as-a-service model. *Proceedings of the 20th IFIP WG 11.3 working conference on Data and Applications Security, DBSEC'06* (pp. 89-103). Springer-Verlag.

Vinel, A., Campolo, C., Petit, J., & Koucheryavy, Y. (2011). Trustworthy broadcasting in IEEE 802.11p/WAVE vehicular networks: Delay analysis. *Communications Letters, IEEE, 15(9)* (pp. 1010-1012). IEEE.

Wang, G., Xiong, Y., Yun, J., & Cavallaro, J. R. (2013). Accelerating computer vision algorithms using opencl framework on the mobile GPU-a case study. *In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* (pp. 2629-2633). IEEE.

Wong, C. K., & Lam, S. S. (1999). Digital signatures for flows and multicasts. *IEEE Transaction on Networks, 7(4)*, (pp. 502–513).

Yan, H., Shi, Z. J., & Fei, Y. (2009). Efficient implementation of elliptic curve cryptography on DSP for underwater sensor networks. *7th Workshop on Optimizations for DSP and Embedded Systems (ODES-7)*, (pp. 7-15).

Yavuz, A. A. (2014). An efficient real-time broadcast authentication scheme for command and control messages. *IEEE Transactions on Information Forensics and Security* (pp. 1733-1742). IEEE.

APPENDIX

APPENDIX

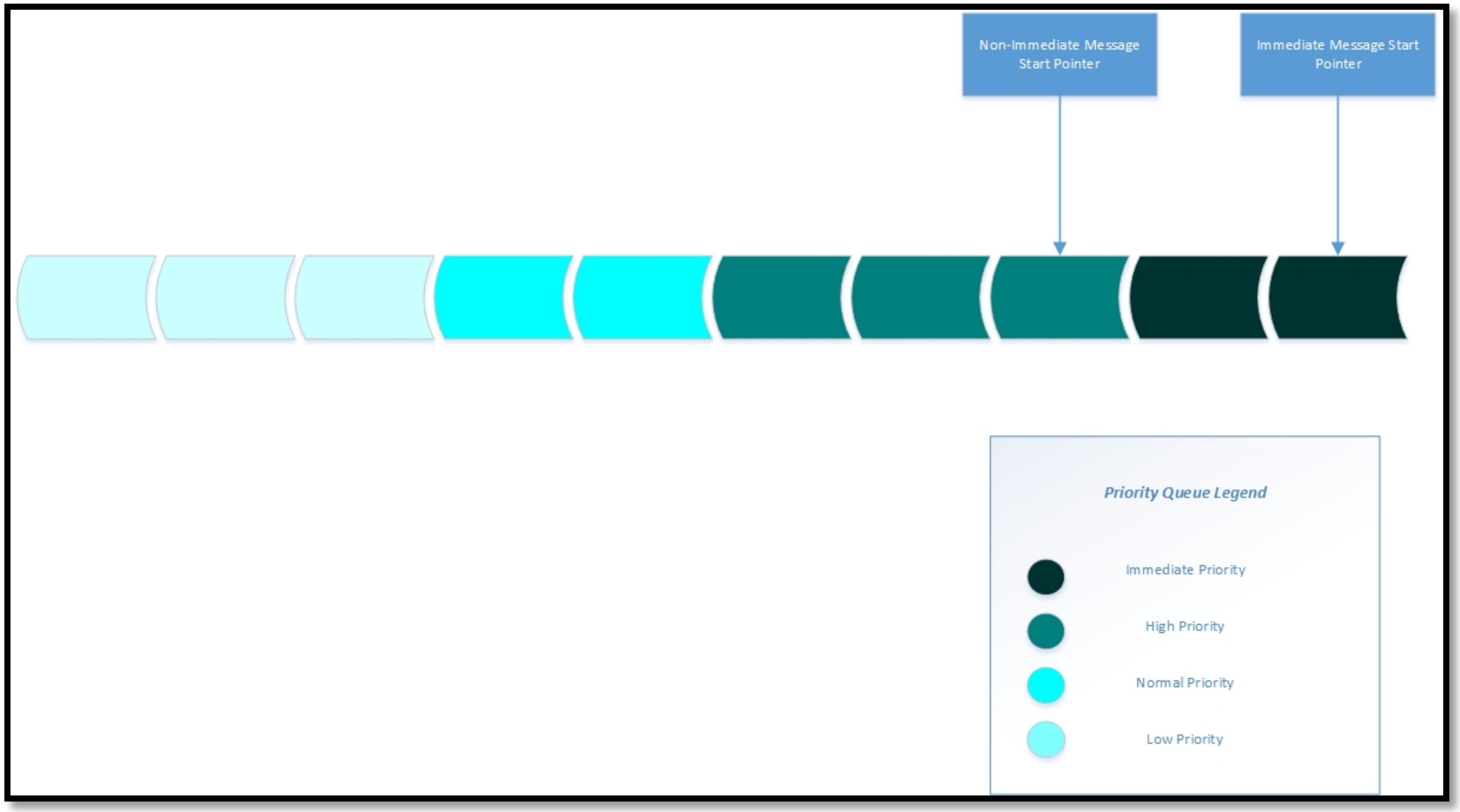
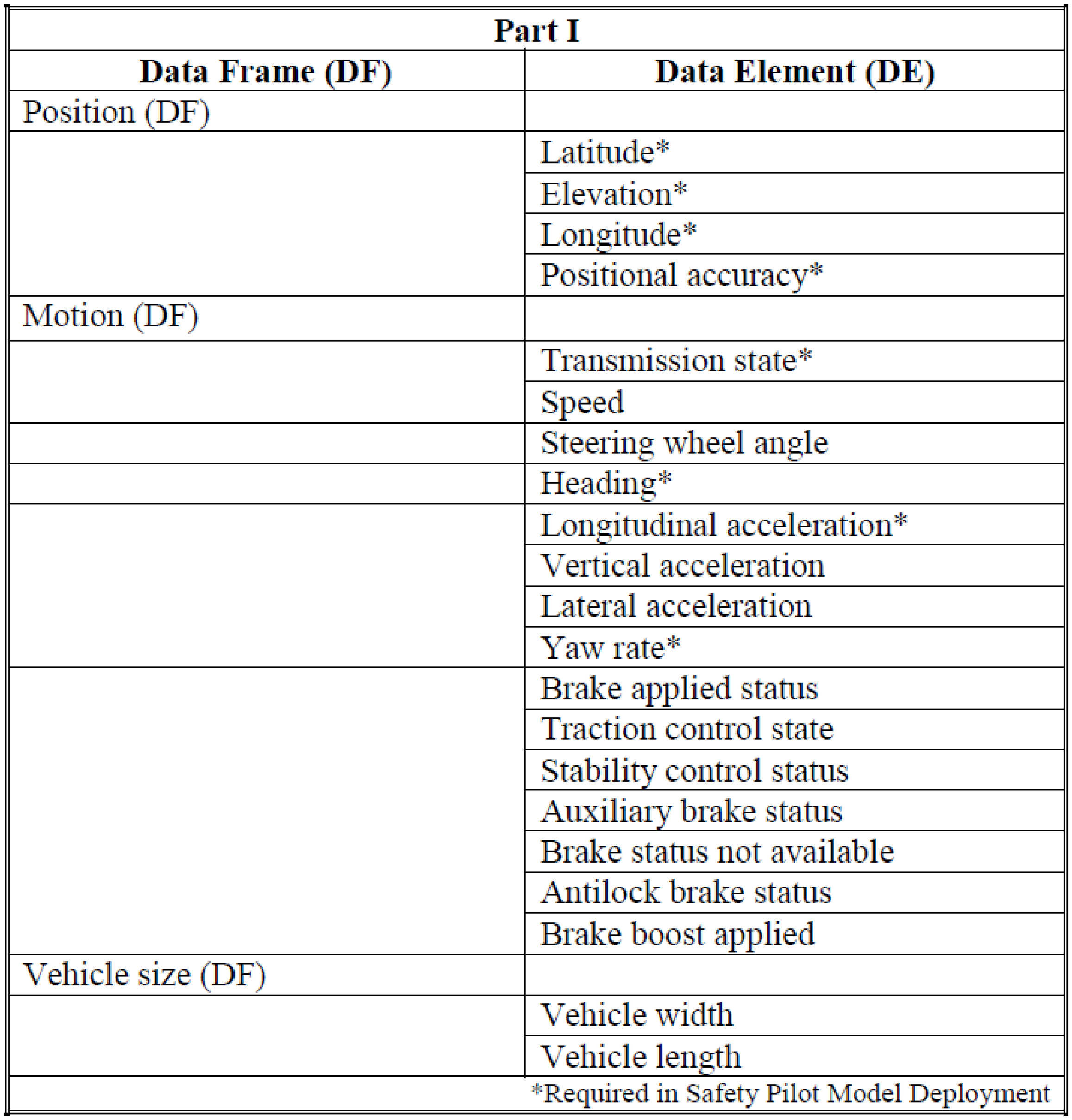


Fig 1: Concurrent Priority Queue



Part I	
Data Frame (DF)	**Data Element (DE)**
Position (DF)	
	Latitude*
	Elevation*
	Longitude*
	Positional accuracy*
Motion (DF)	
	Transmission state*
	Speed
	Steering wheel angle
	Heading*
	Longitudinal acceleration*
	Vertical acceleration
	Lateral acceleration
	Yaw rate*
	Brake applied status
	Traction control state
	Stability control status
	Auxiliary brake status
	Brake status not available
	Antilock brake status
	Brake boost applied
Vehicle size (DF)	
	Vehicle width
	Vehicle length
	*Required in Safety Pilot Model Deployment

Fig 2. Part 1 of Basic Safety Message (SAE)

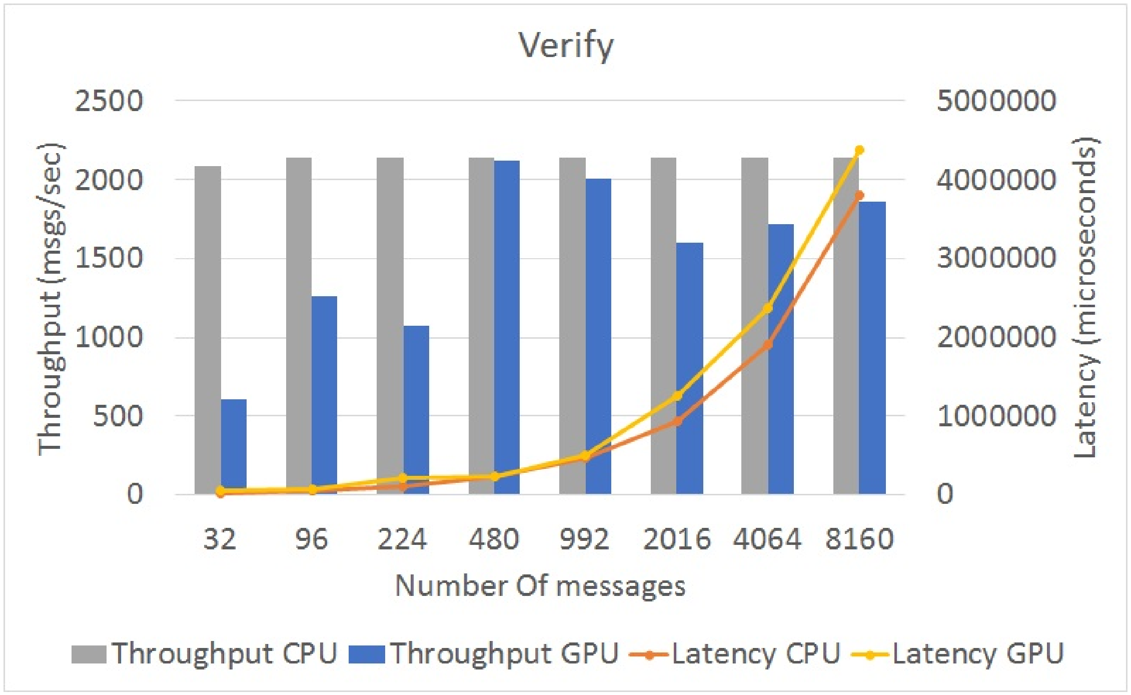


Fig 3. Verification Results for HAA-SOC

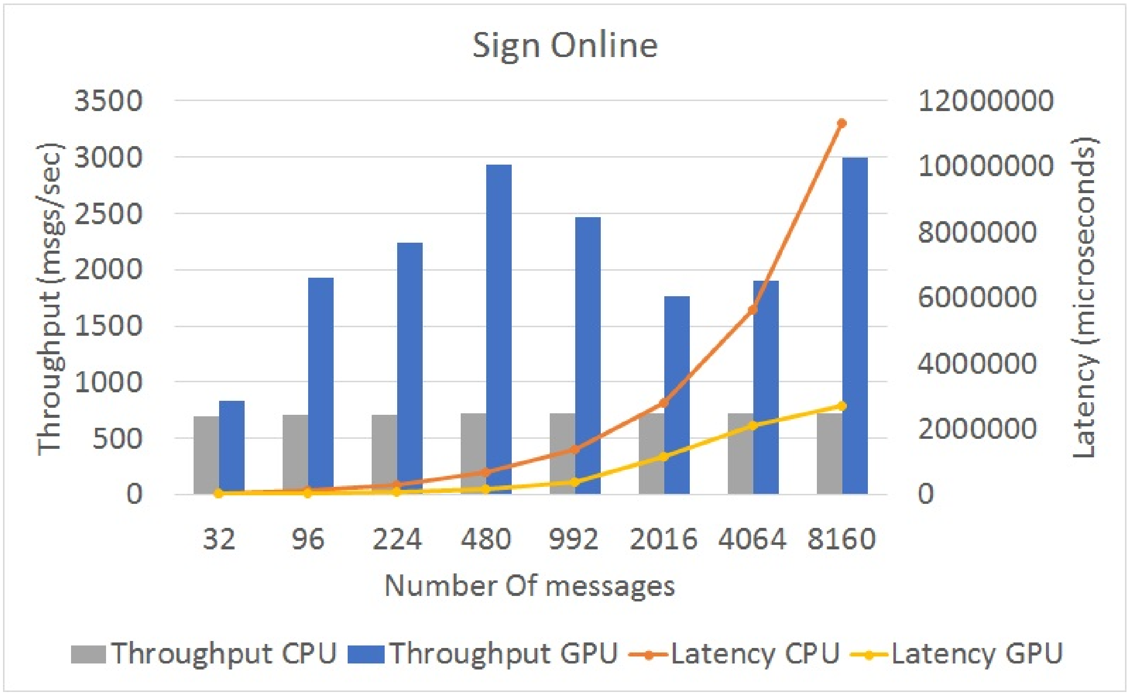


Fig 4. Sign-online results for HAA-SOC

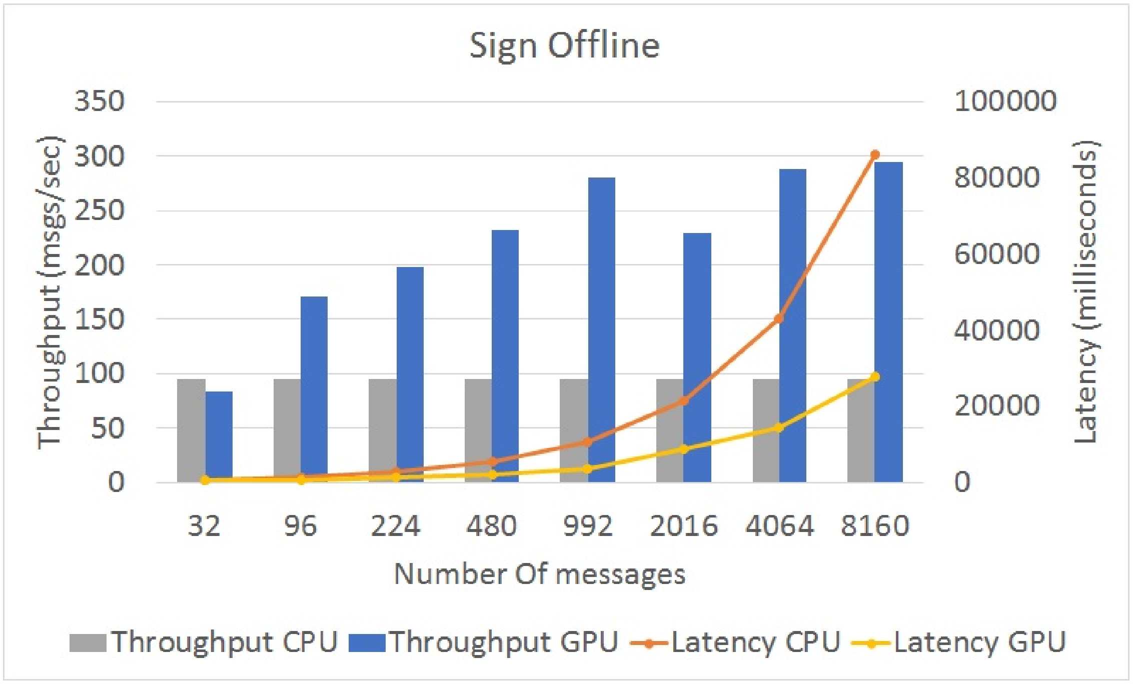


Fig. 5: Sign Offline results for HAA-SOC

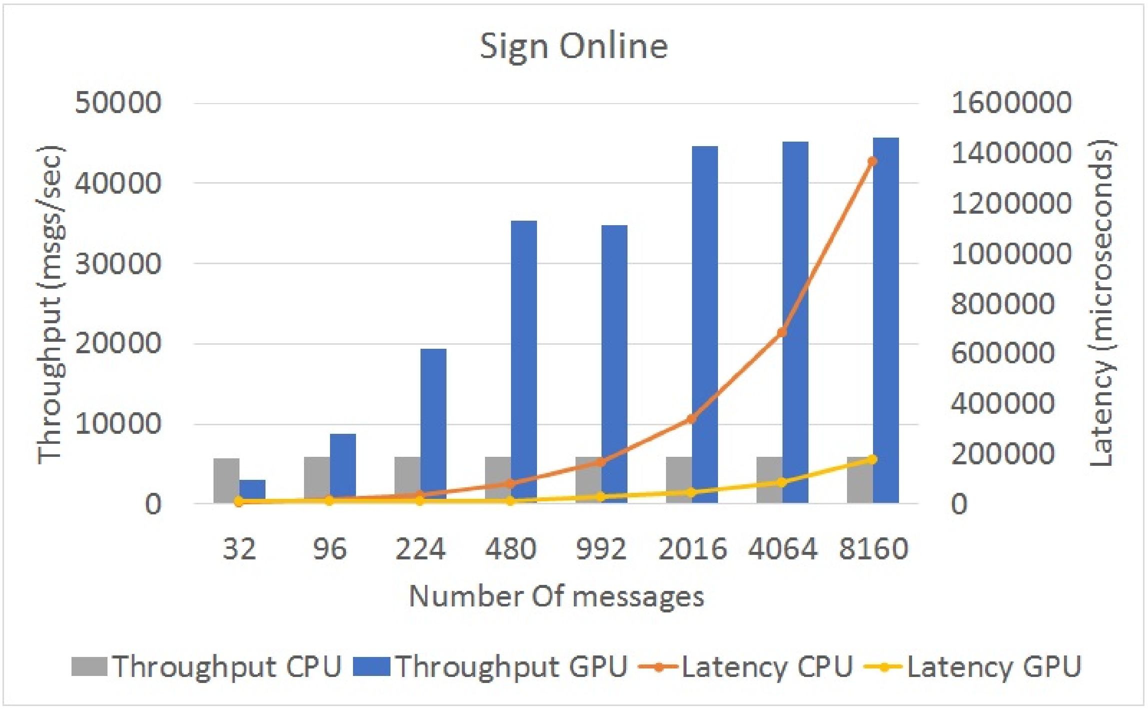


Fig. 6: Sign Online results for HAA

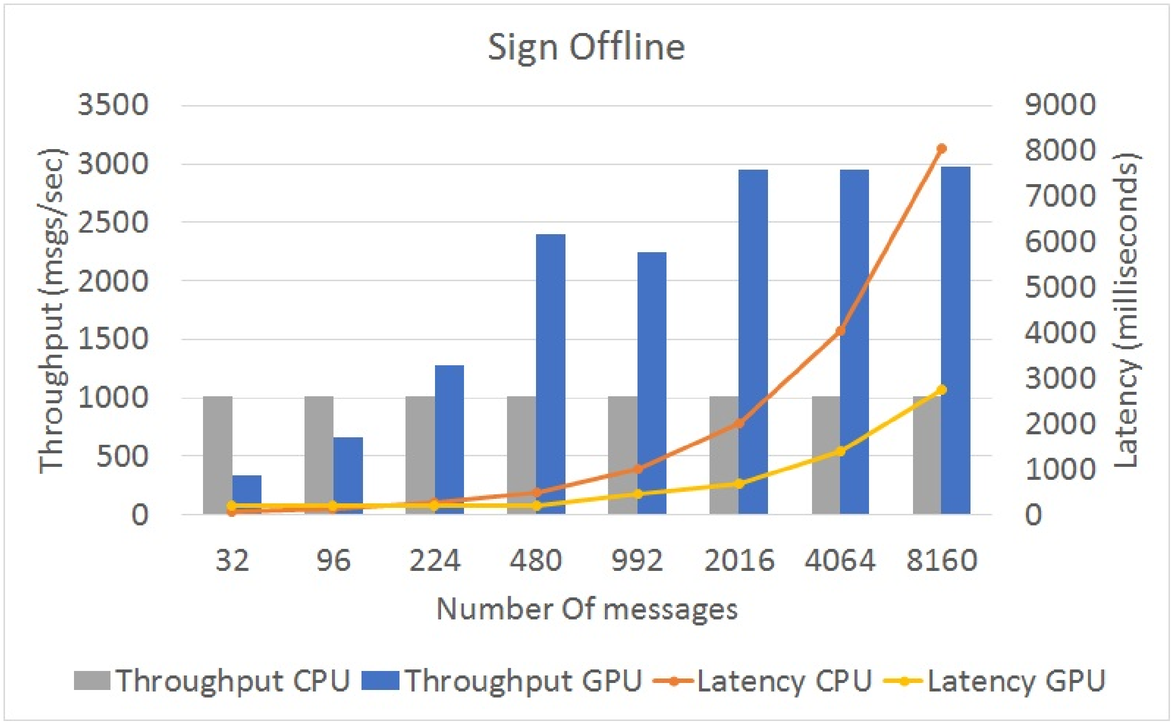


Fig. 7: Sign Offline results for HAA

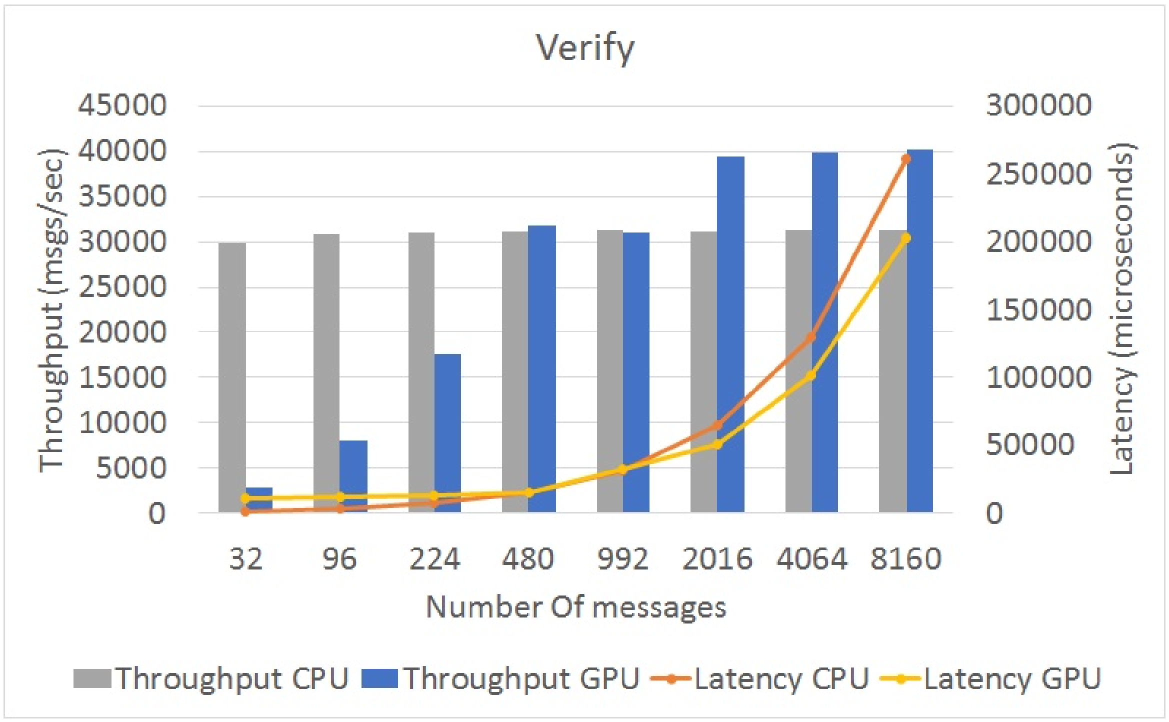
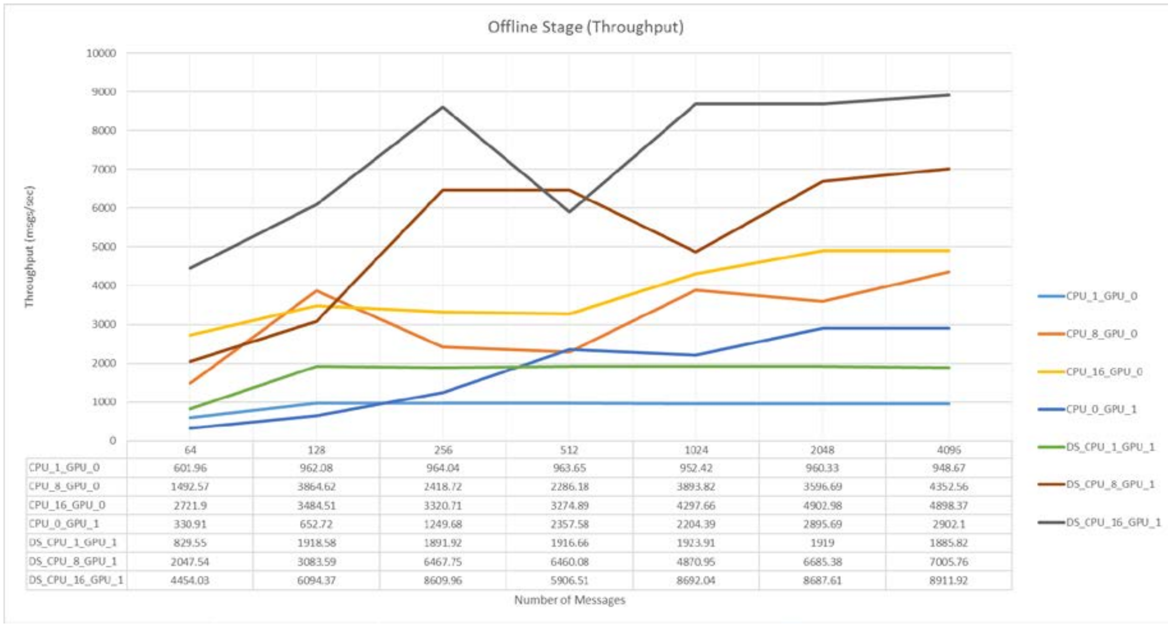
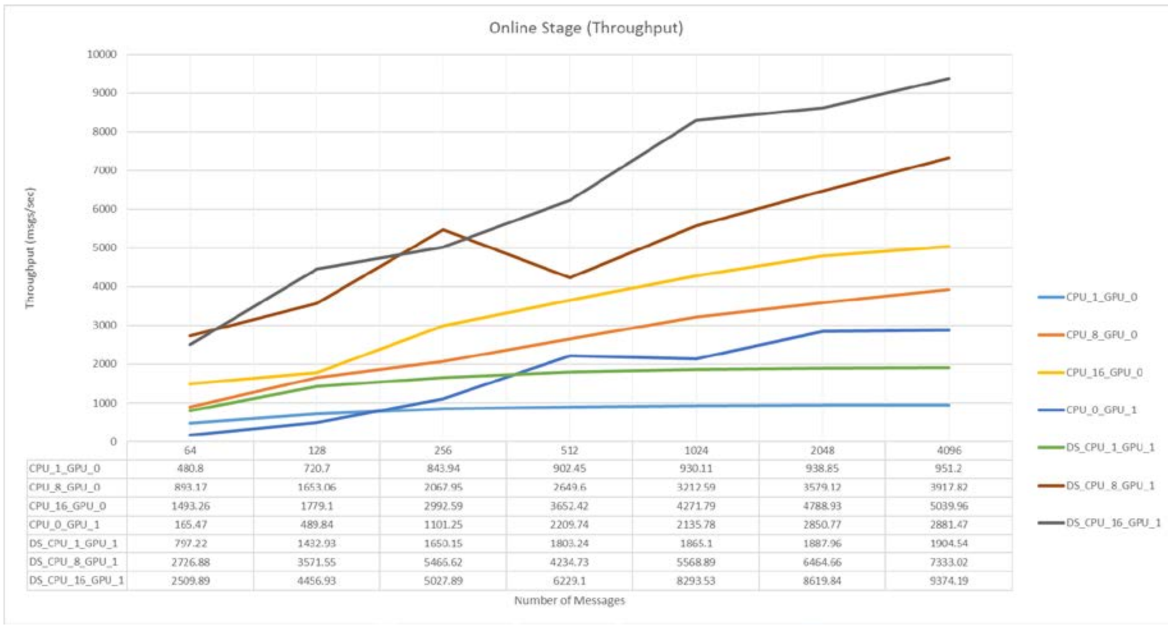


Fig. 8: Verification results for HAA



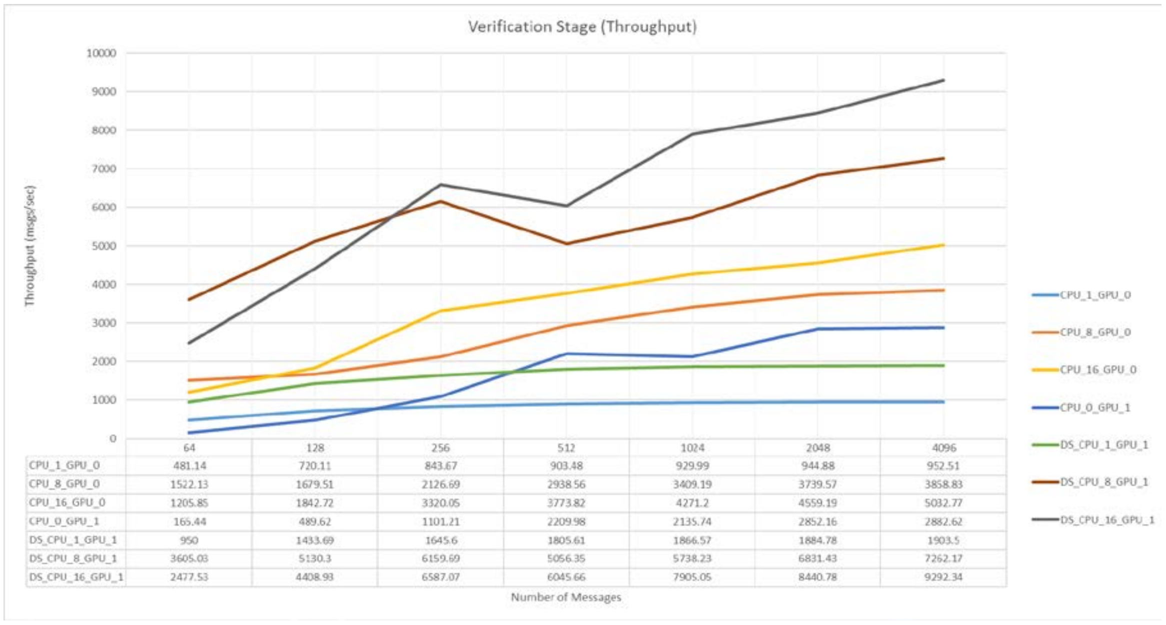
	64	128	256	512	1024	2048	4096
CPU_1_GPU_0	601.96	962.08	964.04	963.65	952.42	960.33	948.67
CPU_8_GPU_0	1492.57	3864.62	2418.72	2286.18	3893.82	3596.69	4352.56
CPU_16_GPU_0	2721.9	3484.51	3320.71	3274.89	4297.66	4902.98	4898.37
CPU_0_GPU_1	330.91	652.72	1249.68	2357.58	2204.39	2895.69	2902.1
DS_CPU_1_GPU_1	829.55	1918.58	1891.92	1916.66	1923.91	1919	1885.82
DS_CPU_8_GPU_1	2047.54	3083.59	6467.75	6460.08	4870.95	6685.38	7005.76
DS_CPU_16_GPU_1	4454.03	6094.37	8609.96	5906.51	8692.04	8687.61	8911.92

Fig. 9: Offline Stage results for DS (CPU+GPU)



	64	128	256	512	1024	2048	4096
CPU_1_GPU_0	480.8	720.7	843.94	902.45	930.11	938.85	951.2
CPU_8_GPU_0	893.17	1653.06	2067.95	2649.6	3212.59	3579.12	3917.82
CPU_16_GPU_0	1493.26	1779.1	2992.59	3652.42	4271.79	4788.93	5039.96
CPU_0_GPU_1	165.47	489.84	1101.25	2209.74	2135.78	2850.77	2881.47
DS_CPU_1_GPU_1	797.22	1432.93	1650.15	1803.24	1865.1	1887.96	1904.54
DS_CPU_8_GPU_1	2726.88	3571.55	5466.62	4234.73	5568.89	6464.66	7333.02
DS_CPU_16_GPU_1	2509.89	4456.93	5027.89	6229.1	8293.53	8619.84	9374.19

Fig. 10: Online Stage results for DS (CPU+GPU)



	64	128	256	512	1024	2048	4096
CPU_1_GPU_0	481.14	720.11	843.67	903.48	929.99	944.88	952.51
CPU_8_GPU_0	1522.13	1679.51	2126.69	2938.56	3409.19	3739.57	3858.83
CPU_16_GPU_0	1205.85	1842.72	3320.05	3773.82	4271.2	4559.19	5032.77
CPU_0_GPU_1	165.44	489.62	1101.21	2209.98	2135.74	2852.16	2882.62
DS_CPU_1_GPU_1	950	1433.69	1645.6	1805.61	1866.57	1884.78	1903.5
DS_CPU_8_GPU_1	3605.03	5130.3	6159.69	5056.35	5738.23	6831.43	7262.17
DS_CPU_16_GPU_1	2477.53	4408.93	6587.07	6045.66	7905.05	8440.78	9292.34

Fig. 11: Verification Stage results for DS (CPU+GPU)

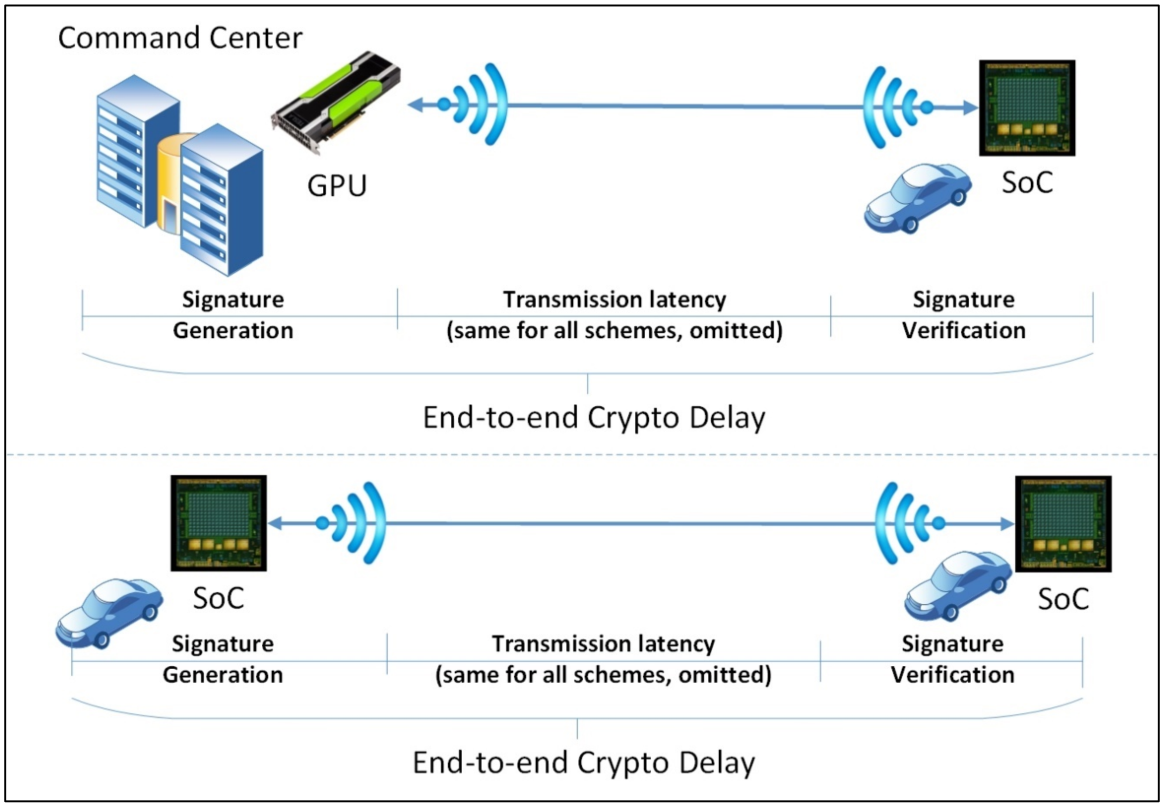


Fig. 12: Depiction of the vehicular network structure