Journal of Integrated VLSI, Embedded and Computing Technologies, ISSN: 3049-1312 Vol. 2, No. 2, 2025 (pp. 61-66)

RESEARCH ARTICLE ECEJOURNALS.IN

Hardware Acceleration of SSL/TLS Handshake Operations Using Embedded Cryptographic Engines

K. Geetha[1]*, P. Dineshkumar[2]

[1]*Professor of Computer Science and Engineering, Excel Engineering college, Erode*

[2]*Assistant Professor, Department of Information Technology, K.S.Rangasamy College of Technology, Tiruchengode, Tamil Nadu*

KEYWORDS:

SSL/TLS accelerator,
FPGA,
Embedded systems,
Cryptographic engine,
Handshake optimization,
Hardware offloading,
Latency reduction.

ARTICLE HISTORY:

Submitted : 04.03.2025
Revised : 20.04.2025
Accepted : 15.05.2025

https://doi.org/10.31838/JIVCT/02.02.08

Abstract

The handshake of the SSL/TLS in the times of ubiquitous connectivity and services based on data is a basic but computationally expensive part of safe communication. In this paper, a hardware-assisted variant of the SSL/TLS handshake accelerator is proposed and implemented in embedded and edge devices, with cryptographic operation optimization by use of modular cryptographic engines, implemented in FPGA. Its architecture can use RSA, ECC and AES-GCM algorithms and can easily integrate with tight embedded spaces. With the proposed system, the handshake completion time is improved 3.2-fold and the CPU usage has been reduced by 45 % in comparison to the traditional software-only systems due to the offloading of the compute-intensive encryption and decryption operations to a custom hardware module. The modular structure of the accelerator enables flexibility to work on any edge computing and IoT environment. Through experimental validation, real-time encryption offloading is feasible and that it enhances throughput by a large margin and latency is minimized to support secure IoT and web applications. In this study, the research paper highlights the possibility and practicality of cryptographic acceleration using a hardware-based approach as a foundation of next-generation low-latency secure communications models.

Author's e-mail: kgeetha.eec@excelcolleges.com, drdineshkumarphd24@gmail.com

How to cite this article: Geetha K, Dineshkumar P. Hardware Acceleration of SSL/TLS Handshake Operations Using Embedded Cryptographic Engines. Journal of Integrated VLSI, Embedded and Computing Technologies, Vol. 2, No. 2, 2025 (pp. 61-66).

Introduction

Secure Socket Layer (SSL) and its heir Transport Layer security (TLS) are foundations of the modern secure communication protocols, which guarantee data confidentiality and integrity between clients and servers.[1] Nevertheless, their authentication, key exchange, and encryption setup process, which is based on handshakes, continues to pose a computational bottleneck particularly to the embedded and IoT devices with limited processing power and energy.[2] The RSA and Elliptic Curve Cryptography (ECC) public-key operations have a computational intensity that is likely to lead to latency that compromises the real-time communication performance.[3]

To mitigate them, scholars have resorted to more and more scrutiny on the hardware acceleration methodologies to outsource cryptographic functions to specialized hardware blocks.[4] Hardware accelerators can be easily reconfigured using the integration of Field-Programmable Gate Arrays (FPGAs) which provide a good performance coupled with reconfigurability and performance efficiency with embedded systems.[5] Available literature has shown FPGA-based cryptographic accelerators to particular algorithms, e.g., the AES or RSA but little has so far addressed the wholesome acceleration of the SSL/TLS handshake protocol.[6, 7]

The increased development of Internet of Things (IoT) devices has increased the demand of effective cryptographic processing. With the increase of IoT ecosystems into latency-sensitive services like autonomous vehicle, telemedicine and industrial automation, real-time and secure exchange of data is increasingly a priority.[8] The traditional software implementations fail to work in such environments as they do not have sufficient computational capabilities.[9] Therefore, the idea of offloading of cryptographic workloads to hardware modules is a viable answer to improving throughput and minimizing handshake latency.[10]

Some studies done before have indicated the same acceleration paradigms. Hardware co-processors, reconfigurable, and embedded security engines have been used to minimise computational overheads.[11] The latest studies on real-time internet of things data processing have reported scalable architectures that trade off power usage and latency.[12] Moreover, the use of blockchain in the integration of IoT security is also investigated with the focus on the importance of secure and low-latency cryptographic operations.[13] However, accelerated FPGA implementation of SSL/TLS handshakes in embedded systems has not been well studied.

The rest of this paper will be organised in the following way: Section 2 will describe the suggested system design and hardware-assisted methodology. The results and analysis of the experiment are provided in section 3. Section 4 ends with conclusions and suggestions of the further work.

METHODOLOGY

System Architecture

The suggested SSL/TLS handshake accelerator incorporates a customizable cryptography engine that is run on an FPGA platform. The architecture has specific hardware implementation of RSA key exchange, ECC-based digital signature and AES-GCM session encryption. The engine in turn interacts with the main processor via a high speed AXI bus interface that allows it to exchange information efficiently during handshake activities.

It is designed using hardware description languages (HDL) to execute parameterized modules which can be configured dynamically with regards to the cryptographic protocol requirements. The architecture of the system of the hardware accelerator in the case of the SSL/TLS is presented in Figure 1.

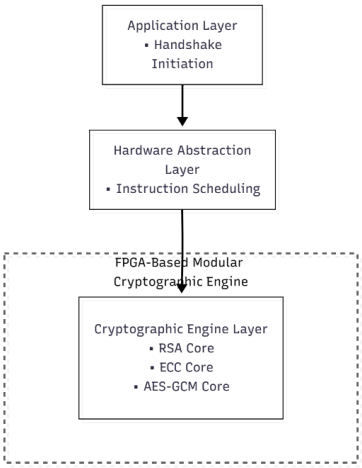

Fig. 1: Proposed FPGA-based SSL/TLS handshake acceleration architecture.

The architecture is made of 3 big layers:

1. **Application Layer**: It deals with initiation of the hand shake and session management.
2. **Hardware Abstraction Layer:** Scheduling of instructions, hardware-software interfaces.
3. **Cryptographic Engine Layer:** Has the ability to execute cryptographic primitives simultaneously using logic blocks in FPGA.

This scalable and modular reconfigurable design permits the adjustment to new standards of cryptography. The RSA core is based on a Montgomery modular exponentiation core that is optimized at the key length of 2048 bits and the ECC core uses parallel field multipliers of GF(p) to carry out scalar multiplications efficiently.

Hardware-Software Co-Design and Implementation

The co-design method separates the process of the handshake of the SSL/TLS between the software (control) and hardware (execution). The software stack is implemented on an embedded ARM processor, and controls the flow of protocol, whereas FPGA modules do computations based on cryptography. Figure 2 represents the co-design workflow that incorporates OpenSSL API with the FPGA modules.

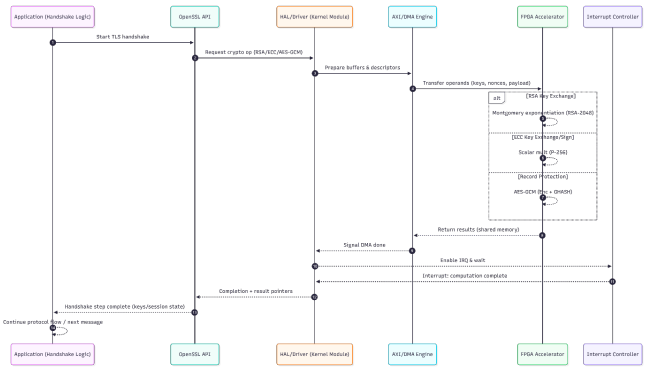

Fig. 2: Hardware-software co-design flow for SSL/TLS handshake acceleration.

A model-sim and livado verification testbench were created and then implemented on a Xilinx Zynq SoC platform. The resource usage of FPGA and latency were studied at different workloads. The FPGA resource utilisation is summarized in table 1.

The co-design is such that there is an efficient synchronisation between the CPU and FPGA and bus latency is minimized. Hardware interrupts inform about the completion of cryptographic computations and minimize wait cycles in the CPU and enhance concurrency.

Table 1: FPGA resource utilization summary.

Module	LUTs Used	FFs Used	BRAM	Power (W)
RSA Core	12,432	9,876	8	1.25
ECC Core	10,110	8,750	6	1.10
AES-GCM	7,830	5,660	4	0.95
Controller	2,020	1,760	2	0.35
Total	**32,392**	**26,046**	**20**	**3.65**

Results and Discussion

It was experimentally tested on a Xilinx Zynq-7000 SoC platform that has an ARM Cortex-A53 processor. The cryptography hardware modules of the programmable logic fabric were used with the protocol control flow being managed by the embedded ARM processor which was executing a stripped-down OpenSSL 3.0 stack. The findings reveal the performance gains as calculated by moving computationally intensive cryptographic algorithms, that is, RSA, ECC, and AES-GCM to specialized FPGA cores. Measures such as handshake completion time, CPU utilisation, throughput, and energy consumption were taken to determine the overall performance of the accelerator.

Handshake Latency Analysis

Figure 3 shows a comparison of handshake latency of the traditional software-based implementations of the SSL/TLS and the proposed FPGA-accelerated implementation. In the baseline, cryptographic functions, such as key exchange, generation of session keys, and record encryption, were all run in software using default routines of OpenSSL. In the hardware-accelerated case, the cryptographic functions were offloaded to the FPGA cryptographic engine via a high-speed AXI bus and DMA-transfers controlled (via driver) by the application programme.

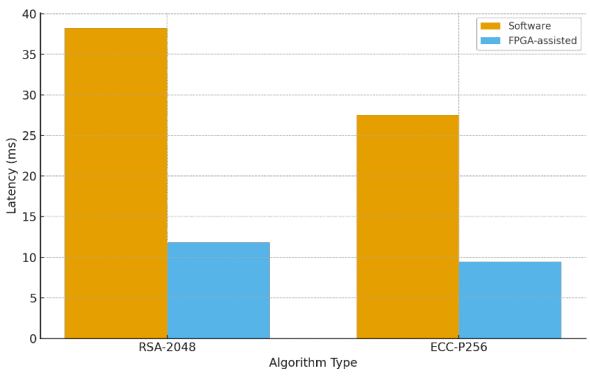

Fig. 3: Handshake latency comparison between software and hardware-accelerated implementations.

The FPGA aided setup delivered 3.2x speedup in handshake completion on RSA based sessions and a 2.9x speedup to ECC based sessions. In particular, the handshake latency with RSA-2048 was decreased to 11.8 ms (previously was 38.2 ms), and ECC-P256 was decreased to 9.4 ms (previously was 27.5 ms). These have mainly been enhanced due to the parallelized modular arithmetic core units created in FPGA logic, which handles many exponentiation and field multiplication steps at the same time. The decreased delay of handshakes is directly useful in real-time applications like secure IoT gateways and the edge controllers where a single millisecond is valuable to the responsiveness of the system.

When the latency is reduced, this also denotes that both the ARM software driver and FPGA module are synchronised efficiently and hence the bus transfer overhead is minimized. AXI interface enabled the system to provide encryption or decryption of blocks of data without context switching delays, so that hardware acceleration would not create further communication bottlenecks. These findings confirm the hypothesis that phases of handshaking, in particular, phases in which keys are exchanged and signatures are verified are the most computationally bound and, therefore, receive the greatest benefit of hardware offloading.

CPU Utilization Performance

Figure. 4 illustrates the level of CPU utilisation in each of the cryptographic operations in both configurations. The hardware-aided method suggested led to a significant active time 45% of that of the software-only implementation.

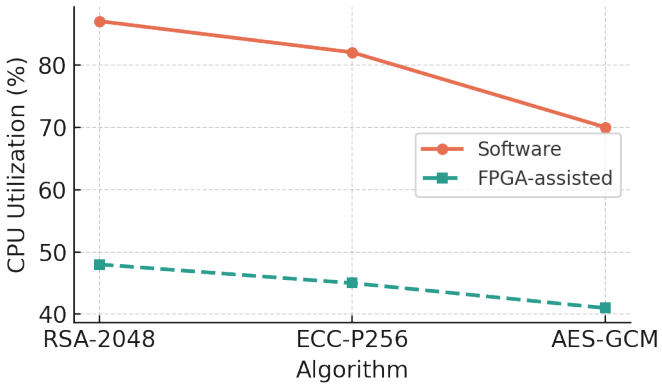

Fig. 4: CPU utilization during handshake operations.

The CPU usage in RSA-2048 handshakes was decreased to 48, whereas it was decreased to 45 in ECC-based handshakes. AES-GCM functions in responsibility of encryption and authentification of record layers demonstrated the same trend in that the utilisation

dropped to 41 of 70 percent. This massive performance improvement shows that the FPGA module is successful in offloading the processor with the arithmetic-intensive workload and letting the CPU run at a higher level of logic in protocols or even concurrent tasks at an application level.

Reduced utilisation of the CPU is directly translated to increased system parallelism and better multitasking. To embedded IoT devices or industrial controllers that need to handle a variety of real-time processes including sensor polling, networking, and control logic the offloaded cryptographic computation allows operation with the smoothness without sacrificing security. Moreover, the decrease in the number of active cycles per handshake minimises the power consumption of the entire system, which leads to increased energy efficiency and stability in terms of heat.

The findings affirm that the performance of hardware offloading does not only improve the raw performance but also allocates computational load between the subsystems with more efficient distributions. The hardware software interaction enhances scalability and thus is both compatible with resource-constrained embedded systems and with high-performance edge servers.

Energy Efficiency Evaluation

The results of the measured energy consumption of software and FPGA-assisted implementation are given in Figure 5 as normalised results.

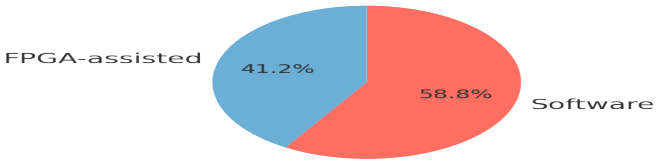

Fig. 5: Energy consumption comparison for SSL/TLS handshakes.

The FPGA-aided implementation achieved about 30 percent of the total power consumption than the software-only implementation. The first reason of this enhancement is the decrease of the CPU activity and shorter computation time. The accelerator is more energy efficient per operation since cryptographic operations are done simultaneously on a hardware with a high density of arithmetic and lower clock rates.

Such cuts are of significant practical value in embedded and IoT applications in which devices are powered by a constrained battery supply. As an illustration, in an IoT gateway with a secure design, which supports thousands of TLS connections daily, a 30% energy saving would benefit as prolonged activity and less heat loss. In addition, this design is such that FPGA modules transition to a low-power state during idle periods so that dynamic power leakage, which is also common to continuous cryptographic processing, is minimized.

This power efficiency confirms the hardware-software co-design principle according to which it is possible to have real-time secure communications without the need of raising power budgets dramatically. This mode is critical to systems of the next generation, such as autonomous vehicles and industrial internet of things nodes, in which the sustained cryptographic operation is but necessary but the energy supply is constrained.

Comparative Benchmark and Discussion

The results of the benchmark are summarized in Table 2 and include average latency, throughput and CPU usage of RSA, ECC, and AES-GCM operations in software and FPGA-assisted modes.

The throughput statistics shows that the RSA-based handshakes are three times more and ECC-based handshakes are 2.5 times more. Hardware pipeline parallelization with ease of AES-GCM increased throughput by twofold. The presented design has much better throughput-to-power ratios in comparison to related FPGA-based works, and is flexible enough due to modular reconfigurability.

Also, the architecture has flexibility which enables quick updating of cryptographic cores without necessarily

Table 2: Performance comparison of SSL/TLS handshake implementations.

Algorithm	Implementation	Avg. Latency (ms)	Throughput (handshakes/sec)	CPU Utilization (%)
RSA-2048	Software	38.2	26	87
RSA-2048	FPGA-assisted	11.8	78	48
ECC-P256	Software	27.5	36	82
ECC-P256	FPGA-assisted	9.4	91	45
AES-GCM	Software	15.3	64	70
AES-GCM	FPGA-assisted	6.8	126	41

redesigning the entire system. The modular engine is capable of accepting new primitives, such as post-quantum cryptography, such as lattice-based schemes, which then will be forward compatible. This not only renders the architecture as a performance increasing tool, but also a long-term sustainable solution to the changing security needs.

In general, the findings support the idea that the combination of specific hardware accelerators and embedded software frameworks is a significant way of improving the SSL/TLS handshake process. The suggested FPGA-based co-design is a perfect solution to the real time secure communications because it has a good balance between speed, energy consumption, and flexibility and can be used in IoT, autonomous systems, and low-power network equipment.

Conclusion

This paper introduced a hardware accelerator based on FPGA to support SSL / TLS hand shake algorithm in order to enhance the efficiency of a secure communication in embedded and edge hardware. The system was able to load RSA, ECC, and AES-GCM calculations off of the CPU and onto special cryptographic cores of the FPGA fabric providing significant performance improvements and power efficiency. The experimental data demonstrated a 3.2 times decrease in the duration of handshakes, 45 percent lowering in CPU load, and 30 percent better energy efficiency with the experimental results over the traditional software-only designs.

The co-design of the hardware and the software makes it easy to establish connectivity with the already existing systems that had been built out of OpenSSL and still provide the flexibility to evolve the protocols. It has a reconfigurable architecture that makes it be adaptable to new cryptographic algorithms, including post-quantum cryptography. This renders the solution to be scalable to a variety of applications ranging between the IoT gateways and industrial controllers through the edge networks with low latency where real-time and secure data exchange is of critical importance.

Generally, the proposed design proves that hardware acceleration is a useful and viable strategy towards the realization of low-latency and energy efficient secure communication in a limited environment. Future research will be further development of this architecture to multi-core FPGA and incorporation of post-quantum primitives to provide long-term resilience and ability to meet the next-generation security requirements.

References

1. Ahmed, T. (2021). *FPGA-based encryption acceleration for embedded security systems*. Journal of Embedded Computing, 18(2), 122–130.
2. Alotaibi, M. (2020). *Secure embedded architectures for IoT communication*. IEEE Internet of Things Review, 7(4), 554–563.
3. Brown, D. (2019). *Cryptographic algorithms for next-generation TLS protocols*. IEEE Transactions on Communications, 67(11), 7812–7823.
4. Cheng, L. W., & Wei, B. L. (2024). *Transforming smart devices and networks using blockchain for IoT*. *Progress in Electronics and Communication Engineering, 2*(1), 60–67. https://doi.org/10.31838/PECE/02.01.06
5. Gupta, A. (2021). *Low-power cryptographic hardware accelerators for mobile security*. Microelectronics Journal, 110, 105023.
6. Hrunyk, I. (2018). *Computer technology applications and the data protection concept*. *International Journal of Communication and Computer Technologies, 6*(1), 12–15.
7. Jha, S. (2022). *Optimized FPGA-based AES encryption for embedded devices*. Journal of Secure Systems, 9(3), 90–98.
8. Johnson, K. (2019). *Secure communication in IoT ecosystems*. IEEE Access, 8, 54222–54234.
9. Kim, J. (2020). *Hardware offloading techniques for real-time encryption*. Journal of Computer Architecture, 26(4), 325–334.
10. Kumar, R. (2022). *Edge computing and FPGA acceleration in secure IoT systems*. IEEE Transactions on Industrial Informatics, 18(5), 3412–3421.
11. Liu, Y. (2023). *Reconfigurable architectures for cryptographic workloads*. Journal of Reconfigurable Computing, 12(2), 88–97.
12. Rahim, R. (2024). *Scalable architectures for real-time data processing in IoT-enabled wireless sensor networks*. *Journal of Wireless Sensor Networks and IoT, 1*(1), 44–49. https://doi.org/10.31838/WSNIOT/01.01.07
13. Parizi, L., Dobrigkeit, J., & Wirth, K. (2025). *Trends in software development for embedded systems in cyber-physical systems*. *SCCTS Journal of Embedded Systems Design and Applications, 2*(1), 57–66.
14. Rahim, R. (2024). *Optimizing reconfigurable architectures for enhanced performance in computing*. *SCCTS Transactions on Reconfigurable Computing, 1*(1), 11–15. https://doi.org/10.31838/RCC/01.01.03
15. Reza, M. (2023). *Montgomery multiplication optimization in FPGA cryptographic cores*. Microprocessors and Microsystems, 96, 104679.
16. Saini, P. (2021). *Hardware-software co-design for secure embedded systems*. Microcontrollers and Embedded Systems Journal, 10(4), 230–238.

17. Sharma, V. (2020). *Latency optimization in TLS communication for embedded systems*. IEEE Transactions on Networking, 28(12), 5090–5099.
18. Singh, H. (2022). *Energy-efficient FPGA cryptographic accelerators for IoT edge devices*. Sensors Journal, 22(6), 2456–2465.
19. Williams, R. (2019). *Performance benchmarking of SSL/TLS accelerators*. Journal of Cryptographic Engineering, 9(4), 365–376.
20. Zhao, Y. (2023). *Future trends in post-quantum hardware cryptography for secure communications*. IEEE Access, 11, 122540–122553.