FPGA-based Cloud Security Solutions for 5G Networks

Marios Papadopoulos
Electrical and Computer Engineering
University of the Peloponnese
Patras, Greece
E-mail: m.papadopoulos@go.uop.gr

Kostas Lampropoulos
Electrical and Computer Engineering
University of Patras
Patras, Greece
E-mail: klamprop@ece.upatras.gr

Paris Kitsos
Electrical and Computer Engineering
University of Patras
Electrical and Computer Engineering
University of the Peloponnese
Patras, Greece
E-mail: pkitsos@upatras.gr

Abstract— The primary goal of this work is to introduce possible hardware solutions regarding cryptographic key management and generation, which ensure uniqueness, high security standards, and of course applicability in multi-tenancy environments over 5G networks. It is also proposed to design an integrated system that combines elliptic curve key generation, client content encryption, and decryption process with FPGA devices, and a user interface that can be implemented with conventional web implementation methods. This paper will describe a cryptographic key generation system with FPGA that can work on Zynq or ultra-scale devices. Essentially this is a hardware IP block that performs the computation of the scalar multiplication [k]P on any elliptic curve defined in a finite field of characteristic p > 3. Intended as a hardware accelerator embedded in a silicon matrix to ensure highly reliable and yet efficient elliptic curve signature authentication and secure Diffie-Hellman key exchange.

Keywords— FPGA, Cloud Encryption, elliptic curve cryptography, Montgomery multiplication, True Random Number Generator, Multitenancy System, 5G networks.

I. INTRODUCTION

The advent of 5G networks has revolutionized the landscape of wireless communication, promising unprecedented speed, reduced latency, and the capacity to connect a vast number of devices simultaneously. This technological leap has opened new avenues for various applications, including cloud services.

One of the most important problems that arise is the security of data traffic in cloud services in such a way that there are clear guarantees that the user is the only key to access this data and that the systems are completely safe and reliable. It is important to mention that security issues concern all networks used today for data transfer, especially those that are wireless and accessible by anyone such as 5G networks.

Also, a promising approach to enhancing security in FPGA cloud environments is the utilization of elliptic curve cryptography (ECC). ECC is renowned for its high security per bit, making it an ideal candidate for resource-constrained environments like FPGAs. The mathematical properties of elliptic curves provide strong encryption with smaller key sizes compared to traditional methods, thus offering both security and efficiency.

This paper explores the intersection of FPGA cloud security, elliptic curve cryptography, and 5G networks. It aims to provide a comprehensive overview of the current state of FPGA cloud security, examine the specific challenges posed by 5G connectivity, and evaluate the potential of ECC in addressing these challenges. The discussion will include an analysis of the security vulnerabilities inherent in FPGA deployment over 5G, the role of ECC in mitigating these risks, and the practical considerations for implementing ECC on FPGAs within a cloud infrastructure.

By investigating these aspects, this paper seeks to contribute to the development of robust security frameworks that can leverage the strengths of FPGAs and ECC, while harnessing the transformative capabilities of 5G networks. Ultimately, the goal is to pave the way for secure and efficient FPGA-based cloud solutions that can meet the demands of the next generation of wireless communication.

II. ELIPTIC CURVES IN A MULTITENANCY SYSTEM

An elliptic curve over a field K is formed by the set of points $P = (x, y) \in K$ satisfying the short-Weierstrass equation:

E/K: $y2 = x3 + ax + b$. to which one must also add the point at infinity, denoted O. The numbers x and y are the affine coordinates of point P. In this solution, the field K denotes a prime field of characteristic $p > 3$, so $K = Fp$. A necessary and sufficient condition of the existence of E/K is that quantity $\Delta /= 0$, where $\Delta = -16(4a3 + 27b2)$. An elliptic curve presents the mathematical structure of an additive group. What makes elliptic curves particularly useful in cryptographic applications [1] is the fact that the discrete logarithm problem in this case is particularly tight compared to other systems that have been studied. As a result, with shorter key lengths, comparable levels of security can be achieved. The basic operation in elliptic curve cryptography is the scalar multiplication [15], that is, given a point $P \in E/Fp$ called base point, one has to compute: $Q = [k]P = P + P + \cdots + P$. (K times) [2].

A complete Key -Aggregation - Cryptosystem (KAC) that can offer both bitstream encryption/decryption for different tenants and efficient key management. is described in Fig. 1.

Each plaintext message is associated with a unique identity Id encrypted with a common master public key (mpk), generated by the system administrator [12 - 13], The administrator, which is part of the cloud security system frontend and interacts with both the cloud security system and the user environment, also generates a master secret key (msk),

that is used to generate decryption keys for the plaintexts. The basic advantage of KAC is its ability to generate constant-size aggregate decryption keys corresponding to identities Id1, Id2, · · ·, Idn. Then, it is possible to generate a constant-size aggregate decryption key ski for the i-th ciphertext. For example, in Fig. 1 the individual secret keys sk1 and skn for the identities Id1 and Idn are compressed into a single aggregate-key sk1 n, which has the same size as either of sk1 and skn, that can decrypt the ciphertexts C1 and Cn, but not C2. KAC operation is based on Elliptic Curves. [5].

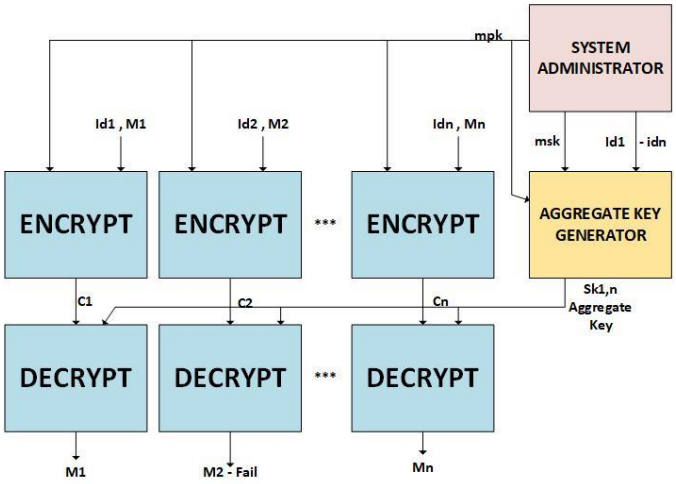

Fig. 1. Key-Aggregation Cryptosystem

Comparing the two main key management systems, elliptic curve, and RSA, the following main conclusions were drawn [2 - 3] (see Table. 1 and Table. 2):

TABLE 1: Advantages between Elliptic Curves and RSA

Characteristics	***ECC***	***RSA***
Very fast key generation	*	
Smaller keys, cipher-texts, and signatures.	*	
Fast signatures	*	
Signatures can be computed in two stages, allowing latency much lower.	*	
Moderately fast encryption and decryption.	*	
Than inverse throughput.	*	
Right protocols for authenticated key exchange (FH-ECMQV et al.).	*	
Better US government support.	*	
Binary curves are fast in hardware.	*	
Unique curves with bilinear pairings allow new-fangled crypto	*	
Signature generation is faster with RSA.	*	
More comfortable to implement than ECC.		*
Easier to understand.		*
Signing and decryption are similar, encryption and verification are similar.		*
Widely deployed, better industry support.		*

TABLE 2: Disadvantages between Elliptic Curves and RSA

Characteristics	***ECC***	***RSA***
Complicated and tricky to implement securely, mainly the standard curves.	*	
Standards aren't state-of-the-art, particularly ECDSA, which is a hack compared to Schnorr signatures.	*	
Newer algorithms could theoretically have unknown weaknesses. Binary curves are slightly scary.	*	
Signing with a broken or compromised random number generator compromises the key.	*	
It still has some patent problems, especially for binary curves. It might be costly...	*	
Public key operations (e.g., signature verification, as opposed to signature generation) are slow with ECC.	*	
Very slow key generation.		*
Slow signing and decryption, which are slightly tricky to implement securely.		*
The two-part key is vulnerable to GCD attack if poorly implemented.		*
Public key operations (e.g., signature verification, as opposed to signature generation) are faster with RSA (8000 ECDSA verifications per second, vs. 20000 RSA verifications per second).		*

According to the above comparative data, most of the disadvantages of the ECC system are related to the difficulty of understanding and implementing such a system and less to issues concerning the reliability of its operation and its performance, in any case ECC excels significantly in speed and reliability, also some of the disadvantages of ECC do not affect its implementation in FPGA Cloud Security [4].

III. Basic Implementation

The initial design of the proposed system includes the ECC generator, i.e. the core based on the ZYNQ platform like 7020-G.

A. Core

The IP is divided into several structural elements where functionally one is based on the other, as the Fig. 2 shows. The top-level component ecc_axi handles the AXI interface, holds the register bank, and services requests issued by software. Below is the ecc_scalar that handles the main hardware state machine. ecc_curve which is a tiny CPU operation only, runs from a dedicated microcode memory called ecc_curve_iram. The content of this memory is read-only and set at synthesis time (though debug mode allows to patch it at runtime) through

the static link of a bunch of assembly source files. CPU is very simple, it has a 3-stage pipeline, no stack or stack pointer, no general_–purpose registers or register window, no cache, no memory hierarchy, no I/O, no calling convention, no exception, no interrupt, no level privileges. Below the ecc_curve element is the ecc_fp element which can be thought of as the Arithmetic Logic Unit (ALU) of ecc_curve. The instructions are retrieved from the microcode memory with ecc_curve decoding and then passed to the ecc_fp to be executed from its internal memory (called "IP internal large number memory", this is the ecc_fp_dram component) which by default can hold 32 large numbers (each with a size of nn bits). There is no memory management unit, so all addresses are physical, both for data and instructions. The commands supported by ecc_curve fall into two categories: arithmetic operations and branch entries.

The main arithmetic operations are addition (NNADD) subtraction (NNSUB), Montgomery multiplication (FPREDC) left and right bit shifts (NNSLL and NNSRL), bitwise XOR (NNXOR). There is also an instruction named NNRND which is used to generate any large random number. Arithmetic is signed and uses two's complement representation. The instructions NNADD and NNSUB should be considered to operate only on integers while FPREDC operates on field elements. In other words, neither NNADD nor NNSUB perform direct reduction, while the Montgomery multiplication performed by the FPREDC instruction by default preserves the reduced form of its inputs in its output. Whenever the result of an addition or subtraction requires a decrement, it is executed with a supplementary conditional statement of subtraction and/or addition, but always in constant time. This is made possible by a hardware mechanism called on-the-fly correction implemented inside ecc_curve. All instructions are executed synchronously (ie the destination operand of instruction i is updated in memory before instruction i+ 1 is decoded) with the exception of Montgomery multiplications. The FPREDC instruction is handled asynchronously, which means that the REDC function is published by ecc_fp to a dedicated hardware Montgomery Multiplier, that is the mm_ndsp element however, support is for mm_ndsp instances 1 through 4 of mm_ndsp however Co-Z arithmetic allows to reduce this to 2 without creating any wait-state that could be caused by dependency between intermediate terms. The number of mm_ndsp instances can still be reduced to one if you want to reduce design area, obviously at the cost of a reduced performance.

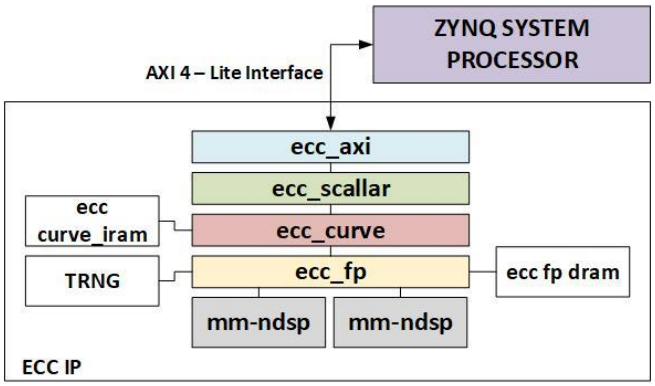

Fig. 2. ECC hardware architecture

The number of multiplier-accumulators (usually named «DSP-blocks» in the FPGA ecosystem) inside each component mm_ndsp is also configurable at synthesis time so as to allow you to set your own cursor in the area-speed trade-off. If software requires synchronization between instructions, it can use the special BARRIER instruction to guarantee strict ordering.

IPECC incorporates its own True Random Number Generator, based on the ES–TRNG [14] design from KU-Leuven No post-processing functionality is provided with IP, although the HDL structure was made to facilitate the integration of a custom one in a straightforward manner. A FIFO set is used to temporarily store the random data. The raw random bits are initially stored in a raw random FIFO. They are then pulled back and assembled to form internal random numbers of different word sizes and then pushed back into one of 4 possible downstream FIFOs, according to an arbitration schedule. If you are familiar with the standard AIS31 terminology from BSI, the first FIFO stores what are called raw random bits while the four downstream FIFOs store what are called internal random numbers (IRNs). But IPECC provided as is does not include any post-processing functions, so there is no real difference between raw random bits and internal random numbers other than their width. If post-processing is applied (which is recommended) it should be considered that this is a cryptographic hardware design, which usually requires the design of a hash function. The new logic should then be placed between the raw FIFO random numbers and the internal random numbers. Each of the 4 FIFOs serves its random numbers to a randomness-based IP element (acting as a client) to implement one of the side-channel countermeasures. These components are: ecc_axi (for in-place scaling of the scalar), ecc_curve (for internal shuffling of the 4 X and Y coordinates of R0 and R1, a feature closely related to the patching mechanism described above), ecc_fp_dram_sh (this is the which replaces ecc_fp_dram when there is a randomization countermeasure in the IP to implement full randomization of the large number memory) and ecc_fp (to implement the NNRND instruction). The RTL code will then automatically build a binary routing tree that includes all instances of the raw TRNG in order to pool their different binary outputs and multiplex their input into the raw random FIFO. In order to allow evaluation of the entropy of a particular plan, IP allows software to read the contents of the raw random FIFO (only when configured in debug mode), a feature that is supposed to be interesting only for FPGA targets.

IPECC is wrapped as AXI4-lite compatible IP, which in practice means that it can connect directly to any AXI-on-a-chip based interfacing system, such as an ARM or RISC-V application processor. Zedboard includes ARM CORTEX – A9 as a powerful System Processing Unit. IPECC then becomes a cryptographic peripheral memory mapped to the system's overall physical address space and used in conjunction with a software driver running on the CPU, can be programmed to perform [k]P acceleration calculations on behalf of the processor without requiring further CPU resource. Given the relatively large computation time required to perform scalar multiplication, and given the small amount of data required to

transfer input and output data to and from IP, no DMA interface is required, and therefore IPECC is presented only as an AXI slave/secondary interface, IPECC supports any elliptic curve in primary fields (prime or complex order) as long as it is defined by the Weierstrass equation. There is a one-to-one correspondence between all elliptic curves and their description by the Weierstrass equation, however, it is not always trivial to translate curves used in some standard elliptic curve protocols without being explicitly defined by the Weierstrass equation. the (a, b) representation required by IPECC. For such curves, one should consider using IPECC together with the highly flexible libecc software library (and for other curves as well), as it automatically and transparently performs the mathematical conversion to Weierstrass form. Additionally, the highest level of elliptic curve operations that IPECC can do is to calculate [k]P. It does not implement any Elliptic Curve Signature Protocol or Elliptic Curve Key Exchange Protocol. If you need to either generate or verify elliptic curve signatures based on any of the EC*DSA type signature schemes, you should consider using a hash function on top of the [k]P calculation, either designing software or using an accelerated IP with hardware hash function. In fact, this is all the more reason why libecc should be considered as the main software component over IP [19].

B. Implementation

Fig. 3 shows the internal Block diagram and their interconnection.

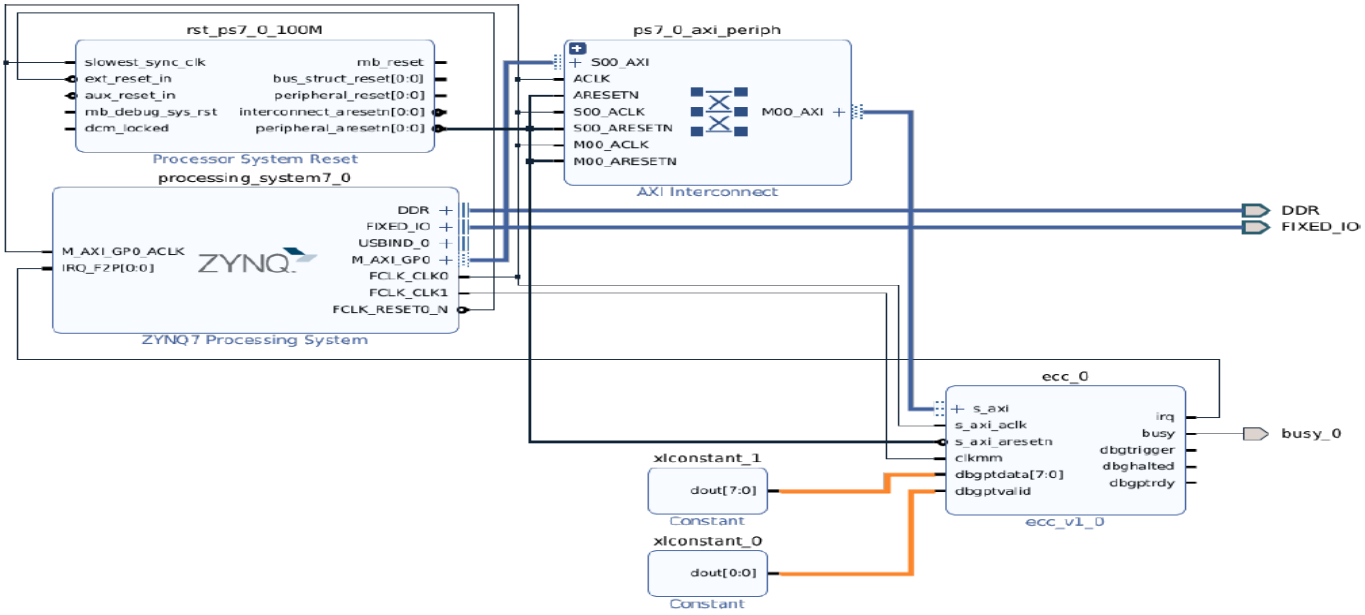

Fig. 3. Block Diagram

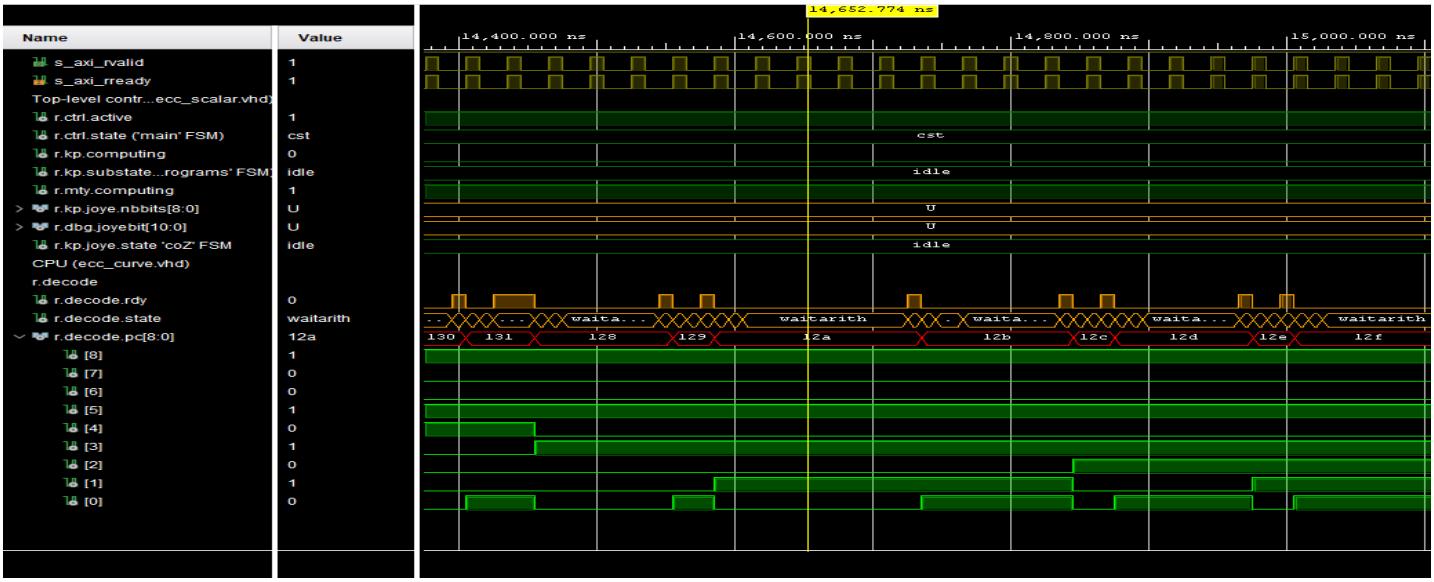

Fig. 4. Simulation

Fig. 4 shows the results of the simulation using the Vivado 2023.2 environment software. The random number generator was simulated using a text input file with random number values.

Fig. 5 shows the simulation results in the log file. The stages of calculations in each cycle of the system are clearly visible, the input data and the results both in each operating cycle as well as in the final stage.

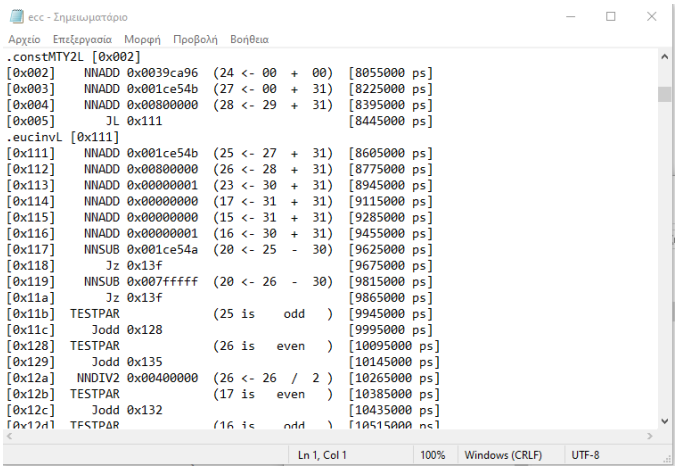

Fig. 5. Simulation Log

The implementation results are shown in Table 3. The analysis describes the FPGA ZYNQ 7020 – G system resources that were committed.

TABLE 3: Hardware Resources Zynq 7020 – G

Site Type	*Used*	*Available*	*Util%*
Slice LUTs	4957	53200	9.31
LUT as Memory	206	17400	1.18
Slice Registers	5226	106400	4.91
Slice	1863	13300	14.00
LUT as Logic	4751	53200	8.93
Block RAM tile	13	140	9.28

As the Table 3 shows the hardware implementation of the proposed system cover a very small number of FPGA resources.

C. Use Case

The use of a key management system with the Elliptic Curves technique analyzed in this work could be integrated into an encryption/decryption system as it has been implemented today [5], combined with a friendly user interface that can be created with the traditional programming tools for web applications, and to constitute as a frontend between a communication system using 5G technology and cloud services (see Fig. 6).

The deployment of Field Programmable Gate Arrays (FPGAs) in cloud environments over 5G networks offers significant advantages in terms of performance, flexibility, and scalability. However, the integration of FPGAs in these settings also introduces new security challenges. This proposal outlines a security framework to enhance the protection of FPGA-based cloud services using Elliptic Curve Cryptography (ECC), leveraging the high performance and low latency capabilities of 5G networks. The rapid adoption of 5G technology facilitates the widespread use of cloud services, including FPGA-as-a-Service (FaaS).

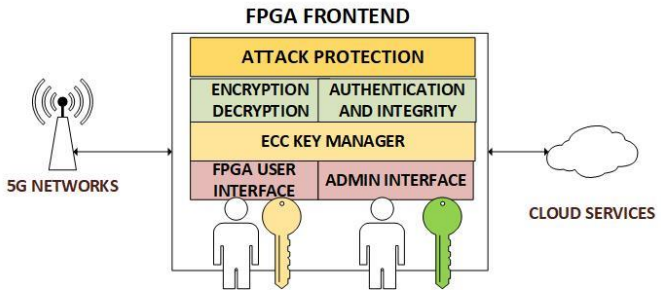

Fig. 6. A. complete FPGA Frontend Security Cloud System over 5G with User and Admin control Interface

However, traditional security models are inadequate for the dynamic and resource-constrained nature of FPGAs in multi-tenant cloud environments. ECC, known for its strong security per bit and efficiency, is well-suited for addressing these challenges. Security Challenges in 5G FPGA Cloud Environments are : a) Multi-Tenancy Risks: In multi-tenant settings, malicious tenants can exploit shared resources to launch attacks such as side-channel and fault injection attacks [9 - 11], b) Data Confidentiality and Integrity: Ensuring that data processed by FPGAs is not exposed or tampered with during transmission over 5G networks is crucial and c) Bitstream Protection: Protecting FPGA bitstreams from unauthorized access and modification is essential to prevent IP theft and malicious reconfigurations. The main points of the proposal to implement such a system based on the key system with elliptic curves are a) ECC's smaller key sizes provide strong security with less computational overhead, suitable for FPGA environments, b) Implement ECC-based digital signatures to ensure the authenticity and integrity of data and bitstreams. Reinforcement techniques could be integrated into the system with FPGA systems that can offer services such as a) Intrusion Detection and Monitoring, and b) Implementing monitoring mechanisms to detect anomalies and potential attacks based on unusual patterns of FPGA resource usage.

IV. Conclusion

In this work, the implementation of an elliptic curve-based cryptographic key generation system that can be combined with frontend encryption/decryption systems for files in cloud environments is detailed. The core of the system is based on the Zynq FPGA platform, on which the SoC was implemented and

supports unique key pair generation functions. Additionally fortify the security of FPGA deployments in 5G cloud environments using ECC, addressing critical vulnerabilities while maintaining performance efficiency. By implementing a robust security architecture, we can leverage the full potential of FPGAs and 5G technology securely and reliably.

ACKNOWLEDGEMENT

This article describes work undertaken in the context of the SAND5G project, “Security Assessments for Networks anD Services in 5G” which has received funding from the European Union’s Digital Europe programme under grant agreement No 101127979 and is supported by European Cybersecurity Competence Center. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

[1] V. S. Miller, "Use of elliptic curves in cryptography," in Advances in Cryptology, H. C. Williams, Ed. Berlin, Germany: Springer-Verlag, 1986, pp. 417-426.
[2] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.
[3] C. P. Schnorr, "Efficient identification and signatures for smart cards," in Advances in Cryptology—Crypto '89, G. Brassard, Ed., Lecture Notes in Computer Science, vol. 435. Berlin, Germany: Springer-Verlag, 1990, pp. 239-252.
[4] C.-P. Schnorr, "Efficient signature generation by smart cards," J. Cryptol., vol. 4, no. 3, pp. 161-174, 1991.
[5] M. Papadopoulos and P. Kitsos, "FPGA-based encryption system for cloud security," 2023.
[6] É. Brier and M. Joye, "Weierstraß elliptic curves and side-channel attacks," in International Conference on Theory and Practice of Public Key Cryptography, 2002.
[7] R. R. Goundar, M. Joye, and A. Miyaji, "Co-z addition formulæ and binary ladders on elliptic curves," 2010.
[8] M. Joye, "Highly regular right-to-left algorithms for scalar multiplication," 2007.
[9] I. Giechaskiel et al., "Gotcha! I know what you are doing on the FPGA cloud: Fingerprinting co-located cloud FPGA accelerators via measuring communication links," 2023.
[10] H. Englund and N. Lindskog, "Secure acceleration on cloud-based FPGAs - FPGA enclaves," in IEEE International Parallel and Distributed Processing Symposium Workshops, 2020.
[11] M. K. Ahmed, J. Mandebi, S. K. Saha, and C. Bobda, "Multi-tenant cloud FPGA: A survey on security," 2022.
[12] A. Bag, S. Patranabis, D. Basu Roy, and D. Mukhopadhyay, "Cryptographically secure multi-tenant provisioning of FPGAs," in Security, Privacy, and Applied Cryptography Engineering, SPACE 2020, Lecture Notes in Computer Science, vol. 12586. Berlin, Germany: Springer-Verlag, 2020, pp. 59-76.
[13] S. Patranabis, Y. Shrivastava, and D. Mukhopadhyay, "Provably secure key-aggregate cryptosystems with broadcast aggregate keys for online data sharing on the cloud," IEEE Trans. Comput., vol. 66, no. 5, pp. 891-904, May 2017.
[14] B. Yang, V. Rožić, M. Grujić, N. Mentens, and I. Verbauwhede, "ES-TRNG: A high-throughput, low-area true random number generator based on edge sampling," IACR Trans. Cryptogr. Hardw. Embed. Syst., vol. 2018, no. 3, pp. 267-292, 2018.
[15] A. Venelli and F. Dassance, "Faster side-channel resistant elliptic curve scalar multiplication," 2010.
[16] I. Blake, G. Seroussi, and N. Smart, Elliptic Curves in Cryptography, London Mathematical Society Lecture Note Series. Cambridge, U.K.: Cambridge Univ. Press, 1999.
[17] R. R. Goundar, M. Joye, A. Miyaji, M. Rivain, and A. Venelli, "Scalar multiplication on weierstraß elliptic curves from co-z arithmetic," J. Cryptogr. Eng., vol. 1, no. 2, pp. 161-176, 2011.
[18] D. Hankerson, A. Menezes, and S. Vanstone, Guide to Elliptic Curve Cryptography. Berlin, Germany: Springer-Verlag, 2004.
[19] N. Meloni, "New point addition formulae for ECC applications," in Arithmetic of Finite Fields, C. Carlet and B. Sunar, Eds. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 189-201.