Resource Centric Analysis of RSA and ECC Algorithms on FPGA

Deeksha Sudarshan[1], *Chirag* Khandelwal[2], *Linge Gowda* B M[3], *Kiran Kumar* Bijjaragi[4], and *Rekha* S S[5]

[1,2,3,4,5]Electronics and Communication Engineering, PES University, Bengaluru, India

Abstract. The electronics industry's shadow side is counterfeiting, and the doom is growing. Almost every business in the supply chain is impacted by the issue, including component suppliers, distributors, Electronics Manufacturing Services (EMS) providers, Original Design Manufacturers (ODMs), Original Equipment Manufacturers (OEMs), and their clients. In fact, any electronics firm that wishes to benefit from the cheap costs associated with globalization must be aware that someone along the supply chain may be persuaded to acquire fake items and sell them as genuine. A thorough grasp of chip designs, including partitioning and prioritizing data transit and storage, as well as a range of obfuscation techniques and activity monitoring, is necessary to reduce the danger of future hardware breaches. To battle this problem, we need to enforce various security measures at different levels of the supply chain. The recent methods include implementing cryptographic ciphers into the devices. The commonly used ciphers are the hard ciphers. But owing to the advancements and increase in the number of low power and resource constrained devices, there has been a dire need to design ciphers that support such devices. This paper talks about the advantages of lightweight ciphers, aiming to secure low power devices and other embedded devices. This work mainly compares two algorithms, RSA(hard cipher) and ECC(light cipher) in terms of their device utilization and power consumption on a Kintex-7. The presented results are justified from simulations performed on the Vivado design suite.

Keywords. Lightweight cipher, RSA, ECC, Asymmetric key, Cryptography

1 Introduction

The widespread outsourcing of semiconductor manufacturing has led to a rise in counterfeit semiconductor devices, IP infringement and cloning, posing a significant security threat. The Internet of Things (IoT) is an imminent application area that substantially utilizes semiconductors, and security has become a paramount concern in this field. The counterfeit devices can be infiltrated into the IoT network, which can have disastrous consequences. Over time counterfeiting has evolved into a very lucrative business. Widespread instances of counterfeiting have had a significant impact on the supply chain of electrical components [1]. The inadequacies in the current legal system, the reduced risk of criminal punishment

compared to other crimes, and the inadequate enforcement of laws are some of the factors contributing to the increase in popularity of counterfeiting [2].

It is essential to establish the identity of a device to ensure that it is a trusted member of the network. It is also imperative that manufacturers, policymakers, and stakeholders work together to address these issues and ensure the security and integrity.

Counterfeiting mainly takes place at either the source end (design house, foundry, and testing centre) or at the e-waste collection end (recycling and relabelling). Since making chips requires a lot of capital, the majority of semiconductor producers have shut down their fabs and switched to a fabless strategy. The companies prefer to finalize the design and this follows outsourcing manufacturing of chips to fabrication plants. Intellectual property owners must provide foundries access to their whole IC design as well as test patterns and test results for them to be able to construct and test the integrated circuits (ICs). Due to the high expense of IP development, people involved in IC manufacture and testing are in a position where they may profit, in an illicit manner from the IP they have been handed. A prime instance of this is when a foundry fabricates more integrated circuits (ICs) than required, it allows them to sell the surplus ICs for a low material cost rather than having to pay the high cost of IP development. A different illustration is, a foundry choosing to sell its faulty ICs rather than discarding them [13].

This switch to a fabless model has dawned paramount distress to the entire semiconductor industry. IP theft, counterfeiting, insertion of malicious data (e.g. hardware Trojans) are some of the risks involved with the fabless model. Numerous studies have been conducted to counteract these attacks on the supply chain of electronic components by unreliable fabs. Some of the methods to deal with these threats include – providing unique electronic chip ID (ECID), Ending Piracy of Integrated Circuits (EPIC), logic obfuscation, Secure Split Test (SST) and active metering. These methods have control over the quantity of ICs produced and the target market.

False ICs may be located by giving each one a unique ID and then cross-referencing that ID with a database. Physical Unclonable Functions (PUFs) are a class of silicon hardware elements that, depending on the particular ICs they are used in, can provide a variety of outputs. Using a challenge-and-response method, ring oscillator (RO) based PUF (RO-PUF) and Arbiter PUF may generate the similar type of static, yet distinct and trustworthy IDs [1]. By allowing the IP owner to maintain a secret key to identify ICs, that only they are aware of, makes it harder for counterfeiters to alter or fabricate the identification. Each chip must be activated using an external key in the EPIC method, which can only be manufactured by the IP rights holder and cannot be replicated.

An innovative combinational locking technique, automatically generated chip IDs, and a creative use of public-key cryptography are the foundations of EPIC. In their IC design process that takes counterfeiting into account, RTL standards are expanded to include support for on-chip TRNG and public-key cryptography. Particularly, at startup, every produced IC is expected to generate its own random public and private keys. The public Master Key and the bare minimum of circuitry required to implement the combinational locking mechanism are also included into RTL. As a result, none of the most recent additions are currently connected to the original reasoning. Using customary logic synthesis, technology mapping, and circuit design, the improved RTL is transformed into a gate-level netlist. One may connect the anti-piracy logic without interfering with the circuit's critical channels now that they have been identified. [18]

A series of security protocols known as hardware metering allow the design house to have post-fabrication control over the assembled integrated circuits. As each chip and/or its functionality can be uniquely identified by hardware metering, the design house is able to differentiate between multiple ICs made using the same masks. Both passive and active hardware metering methods can be used. Using challenge-response pairs, passive techniques

uniquely identify each IC and register it. Afterwards, the appropriate registration of suspicious ICs purchased from the market is verified. Nevertheless, active metering strategies lock each IC until the IP holder unlocks it. The ICs are typically initialised to a locked condition upon power up. [3]. The IP owner is given remote control over each IC using the Active metering approach. A new set of states and transitions that are added to the exclusive finite state machine (FSM) are used to lock all ICs. The specific ID produced for each IC serves as the foundation for the locking mechanism. Only the IC owner has access to the transition table, and they are the only ones who can use this ID to unlock the IC. While the above techniques have been proposed to address the issue, they have limitations in terms of their effectiveness. Basic IC identification techniques are vulnerable to being forged, while advanced methods like PUF-based identification can detect counterfeits but are unable to prevent their production. Active metering techniques are capable of preventing counterfeit production, but their effectiveness is constrained by the fact that the IC must be activated before testing. This can allow unscrupulous foundries to market faulty ICs that have already been turned on by the IP holder. Moreover, some foundries may ask for more keys than necessary by citing a low yield, resulting in an excess of functional ICs in the market. Similar issues may arise during the assembly, testing, and shipping of the ICs. Therefore, there is a need for more robust and comprehensive approaches to address the problem of counterfeit ICs. To address these shortcomings, the Secure Split Test (SST) was introduced that can identify and stop the production and sale of ICs that are overproduced, flawed, or out of specification by unauthorised foundries and assemblies.

2 Background

2.1 Secure Split Test

The process of IC fabrication involves a lot of high costs due to which the majority of the semiconductor companies have shut down their foundries and shifted to a model wherein the entire process of chip fabrication is outsourced. This has its own advantages and disadvantages. The most obvious advantage being the companies itself getting rid of all the hassles involving chip fabrication while the disadvantages can be many ranging from IP theft, faulty IC's entering the market to counterfeiting of chips.

By including an additional SST structure and enhancing communication between IP owners and foundries, it can help prevent cloning by aiding IP owners to safeguard and measure their IPs by securing the manufacturing test process. SST adds hardware elements for cryptography and blocks an IC's proper operation until the IP owner activates it. Having the entire control over deciding which IC enters the supply chain and which does not, the chances of counterfeit products entering the supply chain decrease substantially. To prevent the IC from being triggered without the IP owner's key, SST is made to be resistant to many forms of intrusions. With SST, the IP owner is re-involved in the production test procedure. SST is intended to stop several forms of counterfeit ICs, including cloned, overproduced, faulty, and out-of-spec ICs.

SST inserts two main blocks to the design. i.e. scan- locking block and the functional-locking block. The functional-locking block's primary purpose is to make sure that only unlocked ICs are functionally sound. ICs can only work correctly provided the IP owner gives the appropriate functional key. This block contains true random number generator (TRNG), RSA cryptographic algorithm block and an XOR mask to perform functional locking. The XOR block here can either act as a transparent/buffer block or as an inverter depending upon the inputs. The XOR block functions transparently if the inputs coming in from the TRNG and the RSA block are the same; otherwise, it behaves as an inverter. This is placed in the non-

critical paths of the circuit. TRNGs are used because the numbers they generate cannot be foretold even if one has the access to the designs. This is because they make use of physical aspects of the IC such as temperature, power supply, noise etc. They also provide unique outputs each time they are accessed as opposed to PUFs. The IP owner becomes aware of when to activate the IC only when the correct the input is received from RSA, which makes the circuit functional. The IP owner generates the first set of keys (test keys) by encrypting the modified random numbers generated for each die. Following this, the desired and generated outputs are compared to determine which die has passed or failed the tests. After the functional and verified dies are finalized, another set of keys are produced (functional keys) by encrypting the original random number, which activates all the functionalities of the ICs.

The scan-locking block ensures the functional results and other patterns of the IC's are not taped out/recorded. The scan-locking block ensures that an untrusted individual cannot influence the SST hardware by acquiring information on the 3functional results from an IC. The scan-locking block will prevent any attacker from using patterns and tracking the IC's responses even if the chip has received its functional key and is operating as intended [1], [3].

This guarantees that the IP owner has complete control over the overall chip creation process as he can decide all the passed and failed cases and what can enter the market.

In terms of communication, each IC's random number and test signature must be sent to the IP owner by the testing facility or foundry. The IP owner must then provide keys, along with pass and fail situations based on the test findings. After this, the IP owner needs time to decide and confirm whether the findings are what was expected before finalising the list of passed ICs.

2.2 RSA

Rivest, Shamir, Adleman (RSA) Cryptography is an asymmetric cryptographic algorithm that makes use of large prime numbers. Being an asymmetric cryptosystem, RSA makes use of both public as well as private keys. The strength of encryption is dependent on the key size and chosen prime numbers. This algorithm is based on the Integer Factorization Problem [16]. The strength of the encryption can be increased by tripling or doubling the key size. Although RSA keys are commonly 1024 or 2048 bits in length, experts believe that 1024-bit keys may soon be rendered unusable.[4]

The three significant steps involved in this algorithm are as follows: key generation, encryption, and decryption [6],[7],[8]. For a detailed understanding, let's look at an example.

To generate the keys, consider two prime numbers of equal length p=17 and q=19 as inputs. Calculate modulus n, where $n = p*q = 323$. Choose e, a positive integer in the range $1<e<\phi(n)$ such that gcd $(e, \phi(n)) = 1$ and $\phi(n)$ is the Euler's totient function of n, $\phi(n) = (p-1)*(q-1)$. In this case $\phi(n) = 288$ and $e = 5$ as it is a coprime to $\phi(n)$ and satisfies the gcd condition. Here e is the public/encryption key. Now to compute the private/decryption key, d, use the expression, $d*e = \text{mod } \phi(n)$, which is the multiplicative inverse of e; here $d = 173$.

Next, encryption is performed using the public key (e,n) and plaintext m such that $0<=m<n$. The cipher text, c, is obtained from the equation $c = m^e \pmod n$. Consider $m = 123$ and $c = 123^5 \pmod{323} = 55$. Now to decrypt this cipher text c, we need the private key (d,n). Using the expression $m = c^d \pmod n = 55^{173} \pmod{323}$ gives us the original plaintext $m = 123$.

The computational intensity involved in factoring the two large numbers and the underlying concept of 'trap-door one-way permutation' makes RSA favorable in most use cases. RSA can be used for digital signatures as well making it versatile. This can be used in various areas such as Secure Sockets Layer, Transport Layer Security and VPNs.

Even though RSA is now the most widely used cryptographic technique, given it's a hard cipher, it poses certain level of challenges when being deployed on IOT devices, which leads us to look for other light ciphers to accommodate low power needs.

2.3 Elliptic Curve Cryptography

The elliptic curve discrete logarithm problem (ECDLP) is a supposition made by algorithms that use elliptic curves, that it is difficult to compute the discrete logarithm of a random elliptic curve element with respect to a base point that is open to the public. The security of elliptic curve cryptography is determined by the ability to compute a point multiplication and the inability to compute the multiplicand given the original and product points. Smaller bits key values, which translate to quicker performance and less computational complexity, allow ECC to offer the same level of security as RSA. The degree of security for a 160-bit ECC key is equivalent to a 1024-bit RSA key. This reason has drawn a lot of attention and is being widely used in various mobile applications. Being an asymmetric cryptosystem, ECC also makes use of both public and private keys. To generate the key pairs using the ECC algorithm, we need a two-step process. The first step involves generating a set of domain parameters that are considered valid for use in ECC. These parameters define the curve that will be used for encryption and decryption, as well as other important properties of the ECC algorithm. The process of generating these parameters must be done carefully to ensure that they are secure and can resist attacks.

The second step of the process involves generating the actual key pair that will be used for encryption and decryption using the domain parameters generated in the first part. This key pair consists of a private key that is kept secret and a public key that can be shared with others. The private key is generated randomly and is used to decrypt data, while the public key is used to encrypt it. These underlying concepts of ECC pose a great intractable challenge to the attackers who try to breach the system.

EC is purely calculated over finite fields, and is based on the Weierstrass curve equation, $y^2 = x^3 + ax + b$. This equation has three distinct roots, where the constants a and b satisfy $4a^3 + 27b^2 \neq 0$. This algorithm operates based on the point functions done on selected coordinate points on the curve. Figure 1 shows a basic example of an elliptic curve.

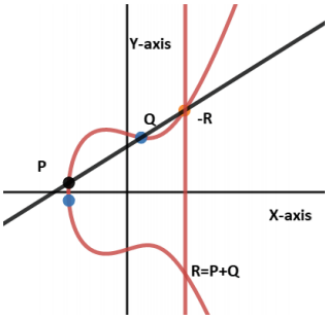

Fig. 1. Elliptic Curve Representation [10]

Since all the coordinates on the elliptic curve are integers, we need to perform all the arithmetic operations modulo a prime number to maintain the precision and consistency. Using mod p also ensures the resulting points are finite. p is chosen based on the size of the elliptic curve and the required level of security, and usually it is a large prime number. There are four, point functions that we will look into [5], [11], [12].

i. Point Addition – Take two points on the curve, say $P(x_1, y_1)$ and $Q(x_2, y_2)$. Now the line that joins P and Q intercepts the curve at point – R and R is the x-axis

reflection of it. To get the coordinates of R, find the slope of line connecting P,Q and -R. Therefore, the x coordinate of R is given by

$$x_3 = \left(\left(\frac{y_2-y_1}{x_2-x_1}\right)^2 - x_1 - x_2\right) \quad (1)$$

mod p and the y coordinate are given by

$$y_3 = \left(\left(\frac{y_2-y_1}{x_2-x_1}\right) \times (x_1 - x_3) - y_1\right) \bmod p \quad (2)$$

ii. Point Doubling – Take any two same points on the curve, P(x_1, y_1) and Q (x_1, y_1), such that R = P + P (or Q + Q) = 2P. Now the coordinates of R are as follows;

$$x_3 = \left(\left(\frac{3x_1^2+a}{2y_1}\right)^2 - x_1 - x_2\right) \bmod p \quad (3)$$

$$y_3 = \left(\left(\frac{3x_1^2+a}{2y_1}\right) \times (x_1 - x_3) - y_1\right) \bmod p \quad (4)$$

iii. Point Multiplication – Choose any point, say P on the elliptic curve and a scalar value k which represents the number of times P must be added to itself. This results in another point Q = kP. This Q will have the same coordinates as P, but is uniformly scaled by the value k. Given Q and P is it nearly impossible to determine k and therefore this increases security.

iv. Zero point/Point at infinity – Take two points P(x_1, y_1) and Q(x_1, $-y_1$). These are said to be in additive inverse and in this case R=0 as the line joining P and Q does not intercept the curve.

The ECC algorithm can be understood using an example [9], [10]. Let Alice be the sender and Bob be the receiver. Consider P to be the public key used to encrypt message M. Let x and y be the private keys of Alice and Bob respectively. Alice computes xP and Bob computes yP and both are made public. To encrypt M, the following must be done. Multiply Bob's public key, yP with Alice's private key, x and this results in xyP. The cipher text is obtained by performing the operation: M + xyP. Next to decrypt this cipher text, Bob's private key, y is multiplied with Alice's public key, xP which results in yxP. Now yxP is subtracted from the cipher text; M + xyP – yxP, which gives us our original message M. The encryption module has two multiplier blocks and one addition block and the decryption module contains one multiplier block, negation operation and one addition block. When implementing this algorithm using Verilog HDL, we require mainly three modules, viz., controller, multiplier and adder. Figure 2 represents the block diagram of all the modules implemented using Verilog. The controller decides when the multiplier and adder modules are to be enabled. It has several input signals such as clock, reset, coordinates of the message M, coordinates of the public keys, select signal to perform either encryption or decryption and a private key which is a randomly generated large integer. It also has an enable signal sent to the multiplier and adder blocks.

The multiplier module has four inputs viz., k which is a randomly generated integer, x and y coordinates of the point on the curve to be multiplied with k and an enable signal. This gives x and y coordinates of the result as the output. Finally, the adder module has five inputs, two sets of x and y coordinates of two points on the curve and the enable signal. Output from this are the coordinates of the result. The final output will have the coordinates of either the cipher text or the decrypted/original text based on the selection.

ECC is widely used because of its wide range benefits over hard ciphers like that of RSA. Some of the benefits include a low-key size, exponential time challenge for the attacker to breach the system, quick cryptographic operations resulting in lesser time needed for both encryption as well as decryption. ECC uses the concept of trapdoor function which makes it simple to compute in one particular direction (i.e., Encryption) but really difficult to compute in the other direction (i.e., Decryption). This makes ECC a very viable option from a security point of view.[5]

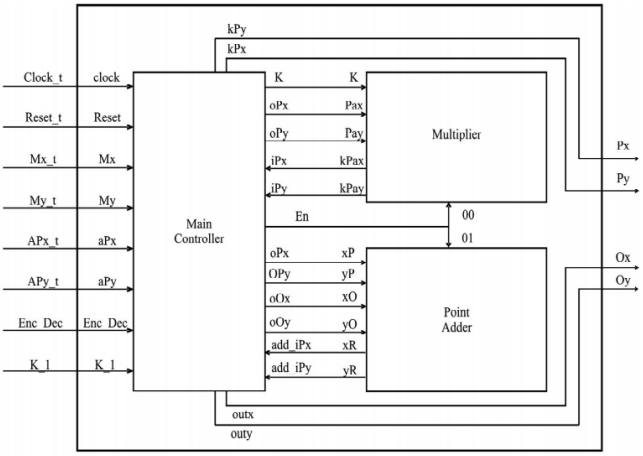

Fig. 2. Block Diagram Representation of the ECC architecture [9]

3 Comparative study – RSA vs ECC

As discussed earlier, RSA is used in the SST architecture for cryptographic operations. [1] talks about the impact RSA has when used in SST. Since RSA can only encrypt data of size equal to the key size and produces an output of the same size, the physical structure of the individual blocks depends on the key size. This also requires the TRNG to produce a random number of size same as the key size. All of this leads to a massive area overhead. And in case any optimization is required, and the RSA structure cannot be changed to a great extent, the rest of the blocks must be examined and required changes must be implemented while also maintaining the right functionality and this becomes a very time intensive process.

Due to these short comings [14] presents a set of comparative analysis of the RSA and ECC algorithms based on various metrics. One key finding was that ECC tends to outperform RSA in terms of operational efficiency and security. This is particularly evident when comparing ECC-160 to RSA-1024 that were used to compute point multiplication operation on the elliptic curve. It was tested on two computers with 8-bit processors, and in both cases, ECC shows increased efficiency, making it a desirable alternative for applications requiring less computing power. Another important aspect of this comparison is the risk associated with key usage. According to studies, using a 1024-bit RSA key up until 2014 presented a little danger, but a 160-bit ECC key over a prime field may be used for much longer without risk. In light of this, ECC may provide longer-term applications a more secure alternative. To support these claims, [14] further provided various case studies. A few case studies are discussed below.

A study aimed to compare the performance of three encryption algorithms: RSA, ElGamal Elliptic Curve Encryption, and Menezes-Vanstone Elliptic Curve Encryption. In order to do this, the FlexiProvider library for ECC and the FlexiProvider implementation of RSA were utilised to create the elliptic curve versions of the ElGamal algorithm in Java. The researchers evaluated the efficiency of the algorithms based on several parameters such as encryption, decryption, encrypted data size and key generation. The findings showed that elliptic curve-based solutions outperformed RSA in all pertinent areas. In conclusion, the study demonstrated that ECC was a better alternative to RSA as it involved fewer overheads.

Another novel approach was suggested in a different study [17] with the intention of lowering the number of memory accesses necessary for public-key cryptography operations. The algorithm was used, and it produced three important findings. Firstly, cryptography

dealing with public keys can function without hardware acceleration on small devices. They observed 0.81 seconds for 160-bit ECC point multiplication on an Atmel ATmega128 running at 8 MHz and 0.43 seconds for an RSA-1024 operation with exponent $e=2^{16}+1$. Secondly, as key size grows and processor word size decreases, ECC point multiplication outperforms RSA modular exponentiation in terms of relative performance. Finally, elliptic curves over fields utilizing pseudo-Mersenne primes, as defined by NIST and SECG, allow for high-performance implementations and show no performance degradation in comparison to prime or extension fields specifically selected for a certain processor architecture. On the Atmel AVR platform, their proposed algorithm improved ECC point multiplication performance by 25%. Their measurements and analyses produced the following key findings: As the word size of the processor reduces, ECC performs more favorably than RSA. This results from the fact that, as the word size decreases, the reduction of Montgomery increases quadratically whereas the reduction that is optimized on the basis of sparse pseudo-Mersenne primes and complexity of subtraction and addition, increases linearly. As a consequence, the performance of ECC point multiplication on small devices has become at par with RSA public-key operations, and for big key sizes, they anticipate it to be much higher.

In terms of security, [16] claims that ECC can employ substantially less parameters than RSA, while maintaining equal security levels. The authors also state that, although ECC consumes very less time for decryption but significant amount of time for encryption and vice-versa in case of RSA, the total time taken for both encryption and decryption is remarkably low in case of ECC. This study was conducted using three different input bit sizes (8, 64 and 256 bits) and four different security level bit sizes (80, 112, 128, 144 bits) and they conclude that ECC proves to be favorable for various memory constraint devices.

Based on the aforementioned studies and analysis, this paper compares RSA and ECC in terms device utilization, power consumption and timing reports to prove that ECC outperforms RSA across all comparable metrics and is therefore a more viable option for deployment in most of the current day applications. This holistic study can be extended and used a base to confidently replace RSA with ECC in the SST architecture and thereby making SST more accessible for majority of the applications.

4 Testing Methodology

The designs and testbenches for the RSA and ECC algorithms were implemented in Verilog HDL for a frequency of 100MHz (clock period of 10ns). The Xilinx Vivado tool was used for simulating the sub modules and top module of the RSA and ECC algorithm. After verification and viewing the elaborated designs, synthesis was performed on the Kintex-7 xc7k70tfbv676-1 board. Table 3 shows the device utilization of RSA and ECC algorithms for key sizes 32 and 64 bits. Table 1 and 2 shows the power consumption of the algorithms and timing reports respectively. Our implementations of ECC and RSA algorithms on the target FPGA device show that the resource utilization for ECC algorithm in terms of LUTs (Look Up Tables) is reduced by 15.32% for 32 bit key size and by 68.43% for 64 bit key size, when compared to RSA. And in terms of power, ECC consumes 52.31% lesser power for 32 bits key size and 51.85% lesser power for 64 bits key size, compared to RSA. For the same clock period of 10ns (100MHz) the ECC implementation has greater slack compared to RSA by 2.32ns for 32 bits key size and by 0.51ns for 64 bits key size. This proves that ECC is an efficient alternative to RSA encryption in the SST architecture due to its lesser overhead, low power consumption and better slack.

Table 1. Power consumption of RSA and ECC

Cryptographic Algorithm	Key Size	Power Consumption in Watts
ECC	32	1.089
RSA		2.283
ECC	64	3.092
RSA		6.421

Table 2. Timing report of RSA and ECC

Cryptographic Algorithm	Key Size	Slack in ns
ECC	32	4.65
RSA		2.33
ECC	64	1.40
RSA		0.89

Table 3. Device utilization analysis of RSA and ECC

Key Width (bits)	Cryptographic Algorithm	LUT			FF			DSP		
		Available	Utilized	%	Available	Utilized	%	Available	Utilized	%
32	ECC	134600	4651	3.46	269200	556	0.21	740	90	12.16
	RSA	134600	25123	18.78	269200	754	0.28	740	44	5.95
64	ECC	134600	8493	6.31	269200	724	0.27	740	224	30.27
	RSA	134600	100600	74.74	269200	2273	0.84	740	152	20.54

5 Conclusion

During message transmission, security is of the utmost importance. Secure messaging is offered using cryptographic methods. Asymmetric-key cryptography was developed by Diffie-Hellmen to address the issues with key management and distribution while also guaranteeing the secrecy and integrity of a message. Over the course of a product's existence, businesses must evaluate the performance of their anti-counterfeiting tactics. The amount of security risk that each firm is prepared to bear in the supply chain must be weighed against the cost of incorporating security measures into the chip. Therefore, early in the design process, designers must take extra precautions to safeguard their original designs. This work presented a comparative analysis of RSA and ECC in terms of their device utilization and power consumption on FPGA. Due to the size and performance advantages of elliptic curve cryptography, networks and communication devices have started to choose ECC over other encryption options and are widely used in wireless communication based applications. [15]

Elliptic curve cipher is perfect for the smaller, less capable devices that the majority of people use today to access network services, since it requires very few keys and is computationally highly efficient. Elliptic curve encryption is more difficult to use than RSA. While RSA employs only one encryption method, ECC may be implemented in a variety of ways. ECC uses arithmetic algorithms as its main aim operations for high-level security features including encryption for maintaining secrecy and a digital signature for authentication. ECC can be used with both software and hardware. According to the standard ECC protocol, each user generates their own public and private keys, and the parties agree on publicly accessible data items. Compared to RSA, ECC is less vulnerable to attacks. This is because RSA uses the prime factorization problem, which is easier to crack than the mathematical attributes of elliptic curves, on which ECC depends. The attack surface for potential attackers is also reduced due to the smaller key sizes implemented in ECC. Key management is made simpler and more effective by ECC, as it requires less storage space for keys. ECC keys can also be generated faster than RSA keys, which can shorten the time needed for key management. While both RSA and ECC offer secure encryption options, the

advantages of ECC such as smaller key sizes, faster key generation, and higher resistance to attacks have made it a preferred choice for many communication and network devices.

Further to expand on this and to cater to various needs of the industry, we can investigate creating or modifying cryptographic algorithms to protect multiple IPs on a single device with only one private key which lock and unlocks all the IPs on the device and acts as the master key, which only the final product or device owner will have access to. This will further make key handling way more efficient. Other applications include Transport Layer Security (TLS), Bitcoin and Secure Shell (SSH) [12].

References

[1] G. K. Contreras, M. T. Rahman and M. Tehranipoor, "Secure Split-Test for preventing IC piracy by untrusted foundry and assembly," *2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, Dec (2013).

[2] Livingston, Henry, "Avoiding Counterfeit Electronic Components. Components and Packaging Technologies", *IEEE Transactions on. 30. 187 - 189. 10.1109/TCAPT.2007.893682,* Apr (2007).

[3] S. Kumar, H. Rao, S. R. Sahoo and K. Mahapatra, "Secure split test techniques to prevent IC piracy for IoT devices.," *Integration VLSI, **vol. 58*, **pp. 390-400**, June (2017).

[4] Nisha, Shireen and F.Mohammed. "RSA Public Key Cryptography Algorithm – A Review". *International Journal of Scientific & Technology Research*. **6**. 187-191.,Jul (2017).

[5] S Patel and Banerji "ECC Based Encryption Algorithm for Lightweight Cryptography", *Advances in Intelligent Systems and Computing, **vol 940**. Springer, Cham,* Mar (2017).

[6] S D Thabah, M Sonowal, R U Ahmed and P Saha, "Fast and area efficient implementation of RSA algorithm," *Proceedings of the International Conference on Recent Trends in Advanced Computing (ICRTAC)*, (2019).

[7] Ko, Kaya. "RSA Hardware Implementation." (1995).

[8] R. L. Rivest, A. Shamir, and L. Adleman. "A method for obtaining digital signatures and public-key cryptosystems". Commun. ACM 21, 2 (Feb. 1978), 120–126. doi.org/10.1145/359340.359342. 1978.

[9] ARCHITECTURE OF ELLIPTIC CURVE CRYPTOGRAPHY USING VERILOGHDL, thesis submitted by S. Latha Shanmuga Vadivu

[10] S Banerjee and A Patil. "ECC Based Encryption Algorithm for Lightweight Cryptography." *International Conference on Intelligent Systems Design and Applications* (2018).

[11] Singh, Laiphrakpam Dolendro and Khumanthem Manglem Singh. "Implementation of Text Encryption using Elliptic Curve Cryptography." *Procedia Computer Science* **54** (2015): 73-82.

[12] M A Saadi and B Kumar. "A Review on Elliptic Curve Cryptography", *International Journal of Future Generation Communication and Networking,* vol.3, no.3, pp. 1597-1601, (2020)

[13] Ujjwal Guin, Daniel Dimase, and Mohammad Tehranipoor. "Counterfeit Integrated Circuits: Detection, Avoidance, and the Challenges Ahead." *J. Electron. Test*. 30, 1 (February 2014), 9–23. doi.org/10.1007/s10836-013-5430-8

[14] D. Mahto and D. K. Yadav, "RSA and ECC: A Comparative Analysis," *International Journal of Applied Engineering Research*, vol. 12, no. 19, pp. 9053-9061, (2017).

[15] F. Mallouli, A. Hellal, N. Sharief Saeed and F. Abdulraheem Alzahrani, "A Survey on Cryptography: Comparative Study between RSA vs ECC Algorithms, and RSA vs El-Gamal Algorithms," 2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/ 2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), Paris, France,

(2019), pp. 173-176, doi: 10.1109/CSCloud/EdgeCom.2019.00022.

[16] D. Mahto, D A Khan and D. K. Yadav. "Security Analysis of Elliptic Curve Cryptography and RSA." (2016).

[17] N. Gura, A. Patel, A. Wander, H. Eberle, and S.C. Shantz, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs," in Cryptographic Hardware and Embedded Systems - CHES 2004, M. Joye and JJ. Quisquater, Eds. Springer, Berlin, Heidelberg, (2004), pp. 119-132.

[18] J. A. Roy, F. Koushanfar and I. L. Markov, "EPIC: Ending Piracy of Integrated Circuits," 2008 Design, Automation and Test in Europe, Munich, Germany, (2008), pp. 1069-1074, doi: 10.1109/DATE.2008.4484823.