FACTA UNIVERSITATIS
Series: **Electronics and Energetics** Vol. 32, N° 1, March 2019, pp. 129-145
https://doi.org/10.2298/FUEE1901129V

FPGA IMPLEMENTATION OF MODIFIED ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM

Kamalakannan Venkataraman, Tamilselvan Sadasivam

Department of Electronics and Communication Engineering,
Pondicherry Engineering College, Pillaichavady, Puducherry, India

Abstract. With rapid deployment of Internet-of-Things (IoT) devices, security issues related to data transmitted between the devices increases. Thus the integrity of perceptual layer devices is of utmost importance to secure the information being transmitted between the devices. In a secured information system, digital signature generation and verification processes are entirely different from data encryption and decryption processes. Digital signatures are rapidly emerging due to the problems related to data integrity thus playing a crucial role in the authentication process by enabling the sender to attach a signature to the encrypted message. Based on the devices it is beneficial to select an algorithm showing favorable behavior, therefore Keccak-f [1600] algorithm is best suited for devices having area and cost constraints. In this paper, implementation of the original Elliptic Curve Digital Signature Algorithm and its variants are considered and evaluated in terms of the security level and computational cost. Here the modified ECDSA scheme concepts related to signature generation and verification are similar to the original ECDSA scheme. The computational cost of the Modified ECDSA is reduced by removing inverse operation in key generation and signing phase, also problems related to signature being forged are resolved using hidden generator point concept. Hence the Modified ECDSA is more secure with less computational cost when implemented on FPGA using Verilog HDL. Therefore, this algorithm can be applied for the devices being connected in perceptual layer of the IoT.

Key words: *Internet of Things, Elliptic Curve Cryptography, Elliptic Curve Digital Signature Algorithm, Secured Hash Algorithm, Keccak*

1. INTRODUCTION

The Internet of Things (IoT) represents a network of independent devices interconnected globally. In the IoT enormous amounts of information have to be communicated, stored, processed and analyzed securely. Securing these pieces of information is one of the fundamental challenges in the IoT. Many IoT products consist of inexpensive components

Received June 20, 2018; received in revised form September 26, 2018
Corresponding author: Kamalakannan Venkataraman
Department of Electronics and Communication Engineering, Pondicherry Engineering College, Pillaichavady, Puducherry, India
(e-mail: vkamalakannan@pec.edu)

with limited memory and computational resources. Such devices might be unable to support the computationally intense cryptographic functions of asymmetrical cryptography. If designers considered the privacy implications of unencrypted data, they have limited options for encryption because of this hardware platform. Therefore the designers have to create their own security protocols. As it is known cryptography is the branch of cryptology dealing with the design of algorithms for encryption and decryption, intended to ensure the secrecy and/or authenticity of message [1]. In 1985 Neal Koblitz and Victor S. suggested that the public key cryptography relates to the algebraic structure of Elliptic Curve (EC) over finite fields. The Digital Signature Algorithm (DSA) was proposed in August 1991 by the U.S. National Institute of Standards and Technology (NIST) and was specified in a U.S. Government Federal Information Processing Standard (FIPS) 186 called the Digital Signature Standard (DSS). In 1992 Scott Vanstone proposed Elliptic Curve Digital Signature Algorithm (ECDSA) because NIST requested public comment related to DSS proposal. In 1998 it was accepted as ISO 14888-3 standard by International Standards Organization (ISO) and in 1999 it was accepted as ANSI X9.62 standard by American National Standard Institute (ANSI). In 2000 it was accepted as IEEE 1363-2000 standard by Institute of Electrical and Electronics Engineers (IEEE) and FIPS 186-2 standard by Federal Information Processing Standards (FIPS). Elliptic Curve Cryptography (ECC) is a public cryptography that has a mathematical advantage when compared to Rivest Shamir Adleman (RSA) as it requires full exponential time for solving Elliptic Curve Discrete Logarithmic Problem (ECDLP). The ECDLP is distributed over points on the Elliptic Curve (EC). A digital signature is generally an authentication process that enables the sender to attach a signature to a message, thus comprises of digital signature generation and digital signature verification processes [2]. The integrity of the message is guaranteed because the digital signatures detects and stops unauthorized users from modifying the data and also authenticates the identity of the signatory. Generally, the ECDSA is an Elliptic Curve variant of the DSA and it gives cryptographically strong digital signatures due to ECDLP concept. Here Keccak-f [1600], recognized as a new Secure Hash Algorithm-3, i.e. SHA-3 by NIST is considered in the digital signature generation and the digital signature verification processes [3]. Here the complexity of digital signature generation and digital signature verification is also analyzed to stop the attacker attempting to forge the signature. In this paper the analysis of the original ECDSA and its variants are considered and evaluated in terms of the implementation area, security level and speed of execution. Based on these analysis a modified ECDSA method is designed and implemented on FPGA for IoT devices. This section gives a brief introduction about the paper. Section 2 and section 3 give an overview of Elliptic Curve Cryptography and Secured Hash Algorithm-3 Keccak. Section 4 gives a detailed description of original ECDSA scheme, its security proofs and an attack possible on original ECDSA scheme. Section 5 describes modified ECDSA suitable for signer with limited computation capability and a method to solve forging problem using initialization and authorization stage. Section 6 provides comparison of ECDSA schemes, whereas Section 7 explains implementation and synthesis of ECDSA. The result analysis of ECDSA and its variant are given in section 8 with conclusions drawn in section 9 followed byreferences.

2. ELLIPTIC CURVE CRYPTOGRAPHY

Elliptic Curves have been studied for centuries by mathematicians, therefore have a very rich history. ECC is the foremost choice in public key schemes due to its smaller key size [4], [13]. The key length of ECC is considerably shorter than that of RSA, but it achieves the same level of security of RSA. Generally, EC are of two finite fields; fields of odd characteristic F_p, where p is a large prime number, and fields of characteristic two $F_2{}^m$, where 2^m is a large binary value. When the distinction is not important, denote both of them as F_q, where n = p or n = 2^m. An EC is the set of solutions (x, y) to Weierstrass equation. An elliptic curve E over a field K is defined by an Eq. (1) as

$$E(K): y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (1)$$

where the coeffients a_1, a_2, a_3, a_4, $a_6 \in K$.

The curve E is nonsingular or smooth and is an elliptic curve if and only if the discriminant of E, ΔE is nonzero. The Weierstrass equation has been transformed to the elliptic curve called a short Weierstrass curve, where a, b $\in$ K. We assume that the characteristic of K $\neq$ 2, 3 and the discriminant of short Weierstrass curve is given in Eq. (2) as

$$\Delta = -(4a^3 + 27b^2) \quad (2)$$

3. SECURE HASH ALGORITHM-3

Keccak has a different structure when compared to other hash functions. Secure Hash Algorithm-3 Keccak was selected because in 2004 SHA-1 was found to be weak, and the threat was carried to SHA-2 also. Successful attacks have been reported in the algorithms SHA-0 and SHA-1, which generate collisions, which influences the principle of hash functions, which is to ensure the information integrity. The function SHA-2 is currently still safe, but as sharing a similar structure with its predecessor, the SHA-1, becomes suspicious and raises doubts about its safety stimulated the scientific community to search a successor more robust and secure. The sha-3 was focused in the information secure area and was more robust and secure. The Keccak architecture is as shown in the Fig. 1 consisting of pre-processing and the sponge construction [5].

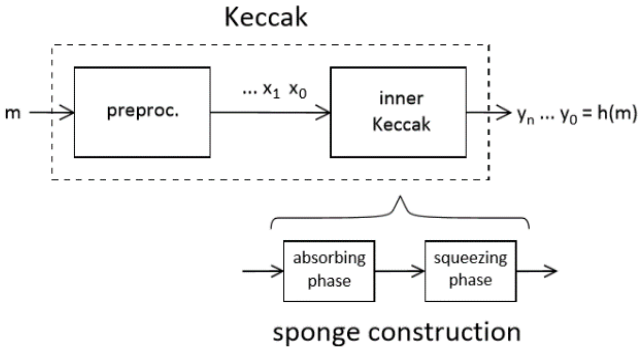

Fig 1 High-level view on Keccak

In the pre-processing construction the message is spliced into blocks with necessary padding. In the sponge construction absorbing (or input) phase and squeezing (or output) phases are present as shown in Fig. 2.

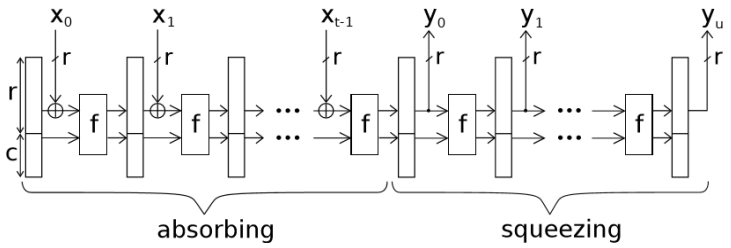

Fig. 2 Absorbing and squeezing phases of the sponge construction

In the absorption phase the block data are applied to the algorithm for processing. In the squeezing phase the processed data is squeezed out based on the configurable length. The function Keccak-f is used in both phases. It reads the input blocks x_i, and generates the output blocks y_j allowing arbitrary-length outputs $y_0 \cdots y_u$. The security level of Keccak has to be configured with several parameters related to the input and output sizes. The parameter b to be configured is the width of the state depending on the exponent l i.e., $b = r + c = 25(2^l)$, where l = 0,1,..,6, having width of $b \in \{25,50,100,200,400,800,1600\}$, r is the bit rate and c is called the capacity. The function Keccak-f referred to as Keccak-f permutation is the main part in hash algorithm and is used in both absorbing phase and squeezing phase. The Keccak-f structure is shown in Fig 3. There are n_r rounds in the function, here each round has an input b bits. The parameter l influences the number of rounds specified in Eq. (3) as

$$n_r = 12+2l \quad (3)$$

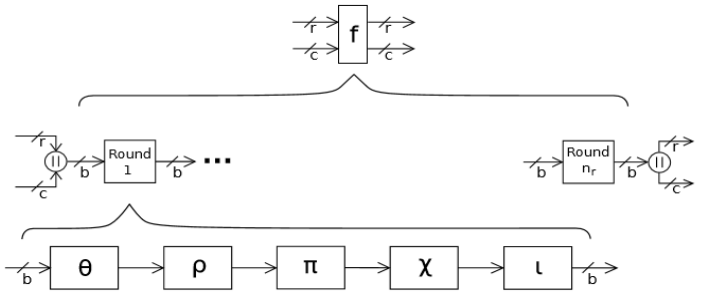

Fig. 3 Internal structure of function Keccak

The number of rounds required for the respective state width is provided in Table 1. Any instance of the Keccak sponge function family makes use of one of the seven Keccak-f permutations, denoted Keccak-f[b], where $b \in \{25, 50, 100, 200, 400, 800, 1600\}$ is the width of the permutation.

Table 1 Number of rounds within Keccak-f

State width b [bits]	# rounds n_r
25	12
50	14
100	16
200	18
400	20
800	22
1600	24

These Keccak-f permutations are iterated constructions consisting of a sequence of almost identical rounds. The number of rounds n_r depends on the permutation width, and is given by $n_r = 12 + 2\ell$, where $2^{\ell} = b/25$. This gives 24 rounds for Keccak-f [1600]. Thus referring the Table 1 the SHA-3 Keccak repeats 24 rounds, each round consists of five steps in sequence manipulating the entire state

- Step 1 -θ step

This function consists of three equations involving simple XOR and bitwise cyclic shift operations.

$$A[x] = S[x,0] \oplus S[x,1] \oplus S[x,2] \oplus S[x,3] \oplus S[x,4] \quad (4)$$

$$B[x] = A[x-1] \oplus ROTL^{1}(A[x+1]) \quad (5)$$

$$C[x,y] = A[x,y] \oplus B[x] \quad (6)$$

Theta step involves XOR-ing between the input state matrix from Eq. (4) and output lanes obtained from Eq. (5) to generate Eq. (6).

- Step 2 -ρ and π step

$$D[y, 2x+3y] = ROTL^{[x,y]}(C[x,y]) \quad (7)$$

Here steps Rho (ρ) and Pi (π) together calculates a 5x5 array "B". The operation of Rho (ρ) and Pi (π) take the state array "C" and perform circular rotation on each of the 25 lanes by a fixed number to obtain array "D" in Eq. (7).

- Step 3 -χ step

$$S[x,y] = D[x,y] \oplus (-D[x+1,y] \wedge D[x+2,y]) \quad (8)$$

In this step operation on the lanes, the D array obtained in the previous steps is manipulated and the results are replaced in the state array "S" illustrated in the Eq. (8).

- Step 4 -τ step

In the Iota (τ) step specified in Eq. (9) the XOR operation is performed for RC round constant specific for each of the 24 rounds of Keccak-f[1600] with the lane at location [0, 0] of the new state matrix "S".

$$S[0,0] = S[0,0] = S[0,0] \oplus RC \quad (9)$$

4. Elliptic Curve Digital Signature Algorithm

In 1992 Scott Vanstone proposed ECDSA because NIST requested public comment related to DSS proposal [6]. In 1998 it was accepted as ISO 14888-3 standard by International Standards Organization (ISO). The ECDSA is an Elliptic Curve variant of

the DSA and because of ECDLP generates a cryptographically strong digital signatures. The integrity plays a critical role to safeguard data inside the network as shown in Fig 4. Sender Bob generates a signature to be added with the message before transmission. At the other end receiver Alice verifies the signature, in order to receive the message [7].

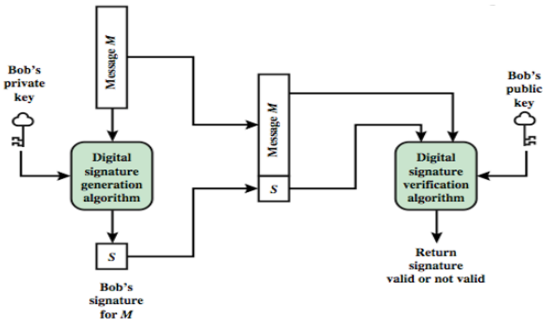

Fig. 4 Digital signature process

ECDSA has been established as an efficient algorithm against cyberattacks and are characterized by their speed to generate and verify the signature. ECDSA consists of 3 phases: key generation, signature generation and signature verification. These three phases are explained in the following sub-sections.

4.1. ECDSA Key Generation

To generate a public and private key sender performs the following steps
Step 1: Select a random integer $d_A \in [1, p\text{-}1]$
Step 2: Computes the public key $Q_A = d_A G$.

4.2. ECDSA Signature Generation

Using the sender's private key d_A and public key Q_A
Step 1: Select an integer $K \in [1, p - 1]$
Step 2: Compute h = HASH (M) = SHA-3 (M)
Step 3: Calculate $KG = (x_1, y_1)$
Step 4: Compute $r = x_1 \pmod p$, If r = 0, go to step 2
Step 5: Compute $s = K^{-1}(h + d_A r) \pmod p$. If s = 0, go to step 2
The signature pair generated is (r, s)

4.3. ECDSA Signature Verification

Using public key Q_A and sender's signature (r, s)
Step 1: Verify that r and $s \in [1, p - 1]$. If not, the signature is invalid
Step 2: Compute h = HASH (M) = SHA-3 (M)
Step 3: Compute $w = s^{-1} \pmod p$
Step 4: Compute $u_1 = hw \pmod p$ and $u_2 = rw \pmod p$
Step 5: Compute $(x_2, y_2) = u_1 G + u_2 Q_A$
Step 6: Compute $v = x_2 \pmod p$

4.4. Proof of ECDSA Scheme

Step 1: Compute $s = K^{-1}(h + d_A r)$ mod p on rearranging
Step 2: Compute $K = s^{-1}(h + d_A r)$
Step 3: Compute $KG = s^{-1}(h + d_A r)\, G = (x_1, y_1)$
Step 4: Compute $KG = s^{-1} hG + s^{-1} d_A r\, G$
Step 5: Compute $KG = w\, h\, G + r\, w\, Q_A$ where $w = s^{-1} \pmod p$ and $Q_A = d_A\, G \pmod p$
Step 6: Compute $KG = u_1\, G + u_2\, Q_s = (x_2, y_2)$ where $u_1 = hw \pmod p$ and $u_2 = rw \pmod p$

Therefore

LHS= $KG=(x_1, y_1)$ and $r = x_1 \pmod p$
RHS=$u_1G + u_2Q_A = (x_2, y_2)$ and $v= x_2 \pmod p$

Hence v=r

The signature is valid if v = r valid, invalid otherwise.

In this algorithm if the same key K is being used for signing each and every messages, then there is an issue of the secret key being found by the intruder. This is explained in the following example, where the same secret K is applied for two different messages m_1 and m_2. In this process two signatures (r, s_1) and (r, s_2) are generated from the Eq. (10) and Eq. (11) as

$$s_1 = K^{-1}(h_1 + d_A r) \tag{10}$$
$$s_2 = K^{-1}(h_2 + d_A r) \tag{11}$$

where h_1 = SHA-3 (m_1); h_2 = SHA-3 (m_2)
Knowing s_1 *and* s_2 it is possible to find the secret key K using the Eq. (12)

$$K = (h_1 - h_2)/(s_1 - s_2) \tag{12}$$

From the equation $K\, s_1 - K\, s_2 = h_1 + d_A r - h_2 - d_A r$
Thus knowing *K, r, s* and *h* in the encryption concept, it is possible to find d_A *by* Eq. (13)

$$d_A = (Ks - h)/r \tag{13}$$

Hence different key should be used for signing different messages, otherwise the private key d_A can be sensed by the intruder. The ECDSA is modified to solve the above problem by considering inverse operation only in verification phase. In this method there is no need of inverse operation in the key generation and signing phase there is no need of inverse operation. The scheme processes are discussed in the following sub-sections having the same key pair generation algorithm.

4.5. ECDSA Scheme 2 Signature Generation

Using the sender's private key d_A and public key Q_A
Step 1: Compute h = HASH (M) = SHA-3 (M)
Step 2: Select a random integer K from [1, $p - 1$]
Step 3: Compute $KG = (x_1, y_1)$
Step 4: Compute $r = x_1 \pmod p$, If $r = 0$, go to step 2
Step 5: Compute $s = (Kh + (r\ xor\ h)d_A)\, G \pmod p$. If s = 0, go to step 2
The signature pair generated is (r, s)

4.6. ECDSA Scheme 2 Signature Verification

Using the sender's private key d_A and public key Q_A
Using public key Q_A and sender's signature (r, s)
Step 1: Verify that r and s are integers in [1, $p - 1$]. If not, the signature is invalid
Step 2: Compute h = HASH (M) = SHA-3 (M)
Step 3: Compute $w = h^{-1} \pmod p$
Step 4: Compute $u = (r \; xor \; h) \pmod p$
Step 5: Compute $(x_2, y_2) = w(s - uQ_A)$
Step 6: The signature is valid if $v = x_2 \pmod n = r$, invalid otherwise

4.7. Proof of ECDSA Scheme 2

Step 1: Compute $s = (Kh + (r \; xor \; h)d_A) \; G = (Kh + u \; d_A) \; G = KhG + ud_AG$
Step 2: Compute $sw = KhwG + uwd_AG$
Step 3: Compute $sw = KG + uwQ_A$ where $w = h^{-1} \pmod p$ and $Q_A = d_A \; G \pmod p$
Step 4: Compute $KG = sw\text{-} \; uwQ_A = w \; (s - u \; Q_s)$
Therefore
LHS= KG=(x_1, y_1) and $r = x_1 \pmod p$
RHS= $w(s - u \; Q_A) = (x_2, y_2)$ and $v = x_2 \pmod p$
Hence v=r

In the ECDSA Scheme 2, an intruder can forge the signature by knowing the public parameters (G, n, p, Q_s) and transmit the wrong information to the receiver. The receiver receives the signature and verifies the signature to authenticate the sender's signature. This is been explained as follows

If an intruder 'T' is forges the signature by knowing the public parameters (G, n, p, Q_s) for a false message 'M' in the following steps

Step 1: For signing a message 'M' by sender, using private key d_A and public key $Q_S = d_AG$
Step 2: Calculate h = HASH (M) = SHA-3 (M)
Step 3: Select a random integer K_T from [1, $p - 1$]
Step 4: Compute $K_T G = (x_T, y_T)$
Step 5: Calculate $r_T = x_T \pmod p$, If $r_T = 0$, go to step 2
Step 6: Calculate $s_T = (K_T h + (r_T \; xor \; h) \; Q_S \pmod p$. If $s_T = 0$, go to step 2

Thus the signature pair (r_T, s_T) is transmitted with the false message 'M'

The receiver obtains an authenticated copy of sender's signature pair with the false message 'M' and verifys the authenticity of sender's signature (r_T, s_T) using public parameters (G, n, p, Q_s) for message 'M' by performing the following steps:

Step 1: Verify that r_T and s_T are integers in [1, p − 1]. If not, the signature is invalid
Step 2: Calculate h = HASH (M) = SHA-3 (M)
Step 3: Calculate $w = h^{-1} \pmod p$
Step 4: Calculate $u = (r_T \; xor \; h) \pmod p$
Step 5: Calculate $(x_T, y_T) = w(s_T - uQ_S)$
Step 6: The signature is valid if $v_T = x_T \pmod p = r_T$, invalid otherwise

If the forged signature is validated, then intruder can successfully send false information, hence digital signature schemes are not secure. To solve this drawback public parameters being shared are reduced.

5. MODIFIED ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM

While comparing the original ECDSA and its variants, it is found that original ECDSA is vulnerable to attack if the same key is used for different messages. Scheme 2 is useful for verifier with limited compute apparatus as there is no inverse calculation in key generation and signing phase, but anyone can use legitimate user's public-key to forge the signature of any information. Thus in the Modified ECDSA scheme hidden generator point concept is applied to authenticate the encrypted message communicated between the devices connected in the perceptual layer of IoT. The normal ECDSA are configured with the points on the Elliptic Curve, a generator point '*G*' is selected publicly available and distributed over the network by the Certificate Authority (CA) [11]. In this scheme, the requirement of CA makes it difficult to implement security. The information shared by the CA can be breached by the intruders, making the network susceptible to MIM attack [12].

Hence to elucidate this exposure and to secure the network against MIM attacks, maintaining the security for each session of communication between the two nodes without a common generator point is suggested. Therefore a generator point is shared only between the devices being connected to communicate. This concept is implemented in the ECDSA has two stages; initialization stage and authorization stage.

5.1 Initialization Stage

Let us consider two nodes represented in Fig. 5 in the WSN. It is assumed that both nodes, i.e. sender and receiver, select their generator points, G_S and G_R individually apart from the private keys, K_S and K_R.

The inverse of the private keys K_S^{-1} and K_R^{-1} are also computed. Once the inverse of the private keys are computed, the sender generates its public key P_{SA} using the Eq. (14), whereas the receiver generates its public key P_{RA} using the Eq. (15)

$$P_{SA} = K_S^{-1} G_S \tag{14}$$

$$P_{RA} = K_R^{-1} G_R \tag{15}$$

Both the public keys P_{SA} and P_{RA} are exchanged between sender and receiver after multiplying it with the inverses of their private keys. The resultant key is transmitted to the receiver as is specified in the Eq. (16), and the resultant key received by the sender is specified in the Eq. (17)

$$P_{SB} = P_{RA} K_S^{-1} = K_R^{-1} G_R K_S^{-1} \tag{16}$$

$$P_{RB} = P_{SA} K_R^{-1} = K_S^{-1} G_S K_R^{-1} \tag{17}$$

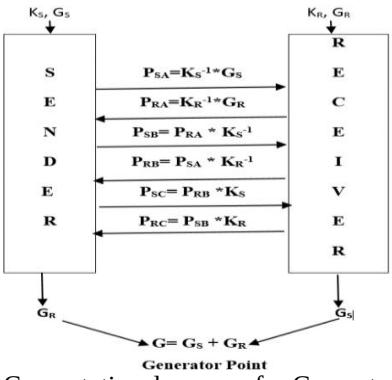

Fig. 5 Computational process for Generator point

These received keys are multiplied again by the sender and the receiver to generate P_{SC} and P_{RC} as specified in Eq. (18) and Eq. (19)

$$P_{SC}=P_{RB}K_S=K_S^{-1}G_SK_R^{-1}K_S=G_SK_R^{-1} \quad (18)$$
$$P_{RC}=P_{SB}K_R=K_R^{-1}G_RK_S^{-1}K_R=G_RK_S^{-1} \quad (19)$$

When P_{SC} and P_{RC} received by the individual sender and receiver, they are multiplied with K_S and K_R to obtain G_R and G_S.
The sender computes the receiver's generator point in Eq. (20) as

$$P_{RC}*K_S=K_S^{-1}*G_R*K_S=G_R \quad (20)$$

The receiver computes the sender's generator in Eq. (21) as

$$P_{SC}*K_R=K_R^{-1}*G_S*K_R=G_S \quad (21)$$

These generator points G_S and G_R are added to generate a common generator points for the sender and receiver given in Eq. (22) as

$$G = G_S + G_R \quad (22)$$

Hence the sender and receiver exchanges information between them and generated using 'G' and computing P, $2P$..... kP.

5.2 Authorization Stage

Let us consider two nodes in the WSN. The public key and the private keys of the transmitter are P_S and K_S, whereas for receiver it is P_R and K_R. The key has to be generated by the process shown in Fig. 6 for every session of transmission between the sender and receiver. Thus authorization has to be provided for each transmission.

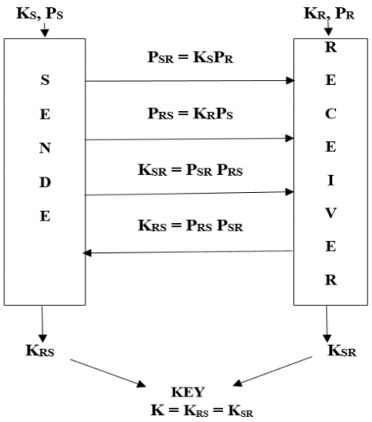

Fig. 6 Computational process for Key

Both the public keys P_{SR} and P_{RS} are exchanged after multiplying it with private keys. The key transmitted to the receiver is specified in the Eq. (23), and the key received by the sender is specified in the Eq. (24)

$$P_{SR} = P_R K_S \quad (23)$$
$$P_{RS} = P_S K_R \quad (24)$$

The keys of the sender and the receiver are multiplied again to generate K_{SR} and K_{RS} given in Eq. (25) and Eq. (26) as

$$K_{SR} = P_{SR} P_{RS} = P_R K_S P_S K_R \quad (25)$$
$$K_{RS} = P_{RS} P_{SR} = P_R K_S P_S K_R \quad (26)$$

When P_{SR} and P_{RS} received by the individuals, the Key K_{SR} and K_{RS} are generated by the sender and receiver individually which are equal, thus commonly referred as Key 'K' in the implementation of ECDSA.

The sender and receiver in the WSN have individual generator points, G_S and G_R with their unique private keys, K_S and K_R. After initializing the keys generation process, both devices exchange the generator points G_S and G_R and generate a common generator point by the initialization process explained in subsection 5.1. Hence the sender and receiver exchange information between them by considering common generator point *'G'* and computing *P*, 2*P, 3P….. kP*.

The sensor nodes must securely share a key before encryption. The shared secret key is generated and refreshed between the sender and receiver. The public key of sender and receiver are P_S and P_R. are exchanged using DHKE process and a key is generated by the method explained in the sub section 5.2.

Considering the generator point *'G'* and key *'K'*, scalar multiplication is performed to compute *KG* provided in Eq. (27), to be applied for signature generation and signature verification process.

$$KG = (K_1, K_2) = (K_X, K_Y) \quad (27)$$

From the initialization and authorization stage, the values of K and G are known. This scheme processes are discussed in the following steps.

5.3. Modified ECDSA Signature Generation

To generates the signature for message *M* the signer using the values of *K* and *G* by performing the following steps:

Step 1: Calculate h = HASH (M) = SHA-3 (M)
Step 2: Compute $KG = (x_1, y_1)$
Step 4: Compute $r = x_1 \pmod p$
Step 5: Compute $s = (K + (r \text{ xnor } h))\ G \pmod p$.
The signature pair thus generated is (r, s).

5.4. Modified ECDSA Signature Verification

The verifier verifies the signature using K and G from the initialization and authorization stage for message M by performing the following steps:

Step 1: Verify that s is integers in $[1, p-1]$. If not, the signature is invalid
Step 2: Compute $KG = (x_1, y_1)$
Step 3: Compute $r = x_1 \pmod p$
Step 4: Compute $u = (r \text{ xnor } h) \bmod (\text{mod } p)$
Step 5: $(x_2, y_2) = (s - uG)$
Step 6: The signature is valid if $v = x_2 \pmod p = r$, invalid otherwise.

5.5 Proof of Modified ECDSA Scheme

Signature send by sender to receiver is (r, s) and s can be generated only by Sender because of its private key.

Step 1: Compute $s = (K + (r \; xnor \; h)\; G$

Step 2: Compute $s = (K + u)\; G$ where $u = (r$ xnor h)

Step 3: Compute $s = KG + uG$

Step 4: Compute $s - uG = KG = (x_2, y_2)$

Therefore

LHS = KG = (x_1, y_1) and $r = x_1 \pmod{p}$

RHS = $(s - uG) = (x_2, y_2)$ and v = $x_2 \pmod{p}$

Hence v=r

The improved ECDSA scheme reduces the computational cost while keeping the same security as original ECDSA. They are suitable for the users who have limited computing capacity.

6. Comparison of Elliptic Curve Digital Signature Algorithm

The original ECDSA and proposed ECDSA are compared and represented in the Table 2. While comparing the original ECDSA and the proposed ECDSA, it is found that the original ECDSA consists of inverse operations in signature generation and signature verification and hence is more complex as needs more point multiplication operation. The improved scheme, the initialization stage and authorization stage are introduced to share the values of K and G between the sender and receiver, thus reducing the computational cost as no inverse operations are required for signature generation and signature verification, while keeping the same security as original ECDSA.

Table 2 Comparison of ECDSA variants

Algorithm	Signature Generation	Signature Verification	Attack	Inverse in Key generation	Inverse in Signing	Inverse in Verification
Original ECDSA	$S=k^{-1}(h + d_A r)$	$u_1=hs^{-1}$ $u_2=rs^{-1}$ $u_1G + u_2Q_A$	Vulnerable	No	Yes	Yes
Proposed ECDSA	s = (K + (r xnor h) G	u = (r xnor h) (s - uG)	Not Vulnerable	No	No	No

7. Implementation and Synthesis Elliptic Curve Digital Signature Algorithm

The original ECDSA signature generation and signature verification was realized in Verilog HDL and simulation was carried out using ISim simulation tool available in XILINX 14.3 for verifying its functional correctness. The RTL block schematic of the ECDSA signature generation is illustrated in Fig. 7 and ECDSA signature verification is illustrated in Fig. 8.

The ECDSA signature generation and signature verification were synthesized and the device utilization summary, timings summary and memory utilization are tabulated in Table 3. The hardware implementation of ECDSA signature generation was performed on Virtex-5 5XC5VLX50T-1FF1136 FPGA Development board by XILINX to evaluate the area and speed. It was found that the ECDSA signature generation operated at a maximum frequency of 13.180 MHz whereas the ECDSA signature verification operated at a maximum frequency of 13.210 MHz.

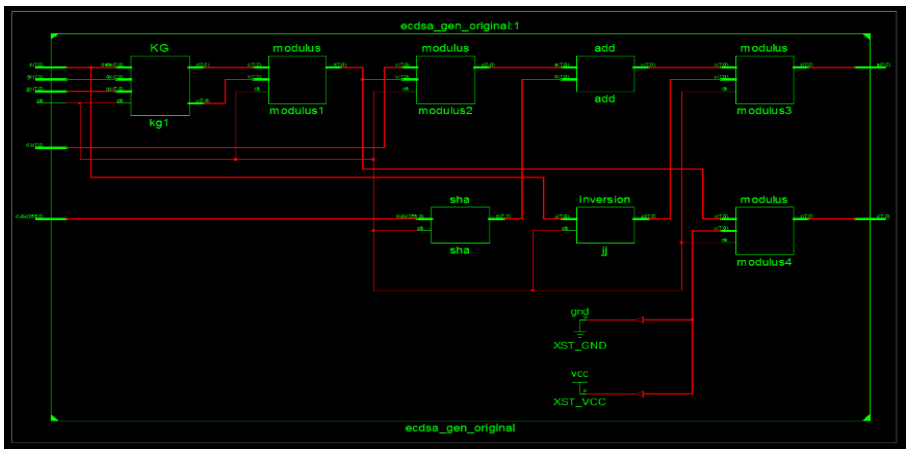

Fig. 7 RTL Block Schematic of ECDSA signature generation

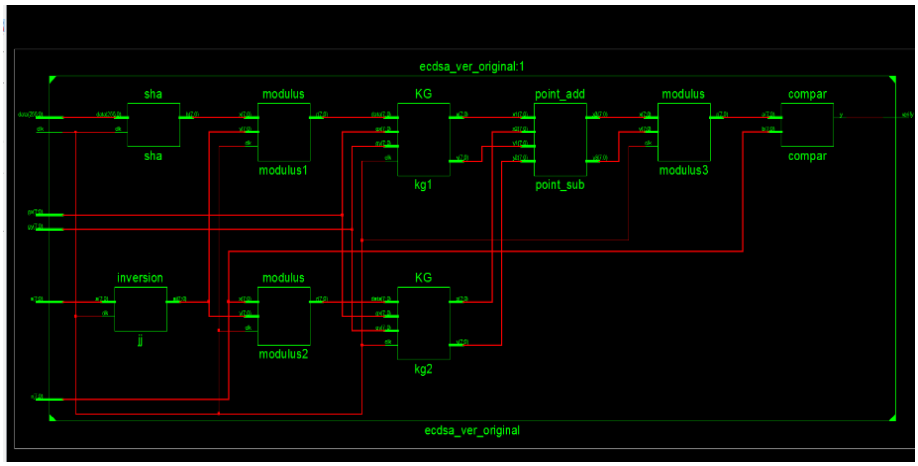

Fig. 8 RTL Block Schematic of ECDSA signature verification

Table 3 Synthesis Summary for ECDSA

Parameters	Signature Generation	Signature Verification
Slice Registers	6701	6790
Slice LUTs	16370	22734
LUT-FF pairs	4884	4996
Bonded IOBs	226	226
Real Time	2627.00 secs	890.00 secs
CPU Time	2626.99 secs	890.19 secs
Maximum Frequency	13.180 MHz	13.210 MHZ

The modified ECDSA signature generation and signature verification was realized in Verilog HDL and the simulation was carried out using ISim simulation tool in XILINX for verifying its functional correctness. The RTL block schematic of the Modified ECDSA signature generation is illustrated in Fig. 9, and Modified ECDSA signature verification is illustrated in Fig. 10.

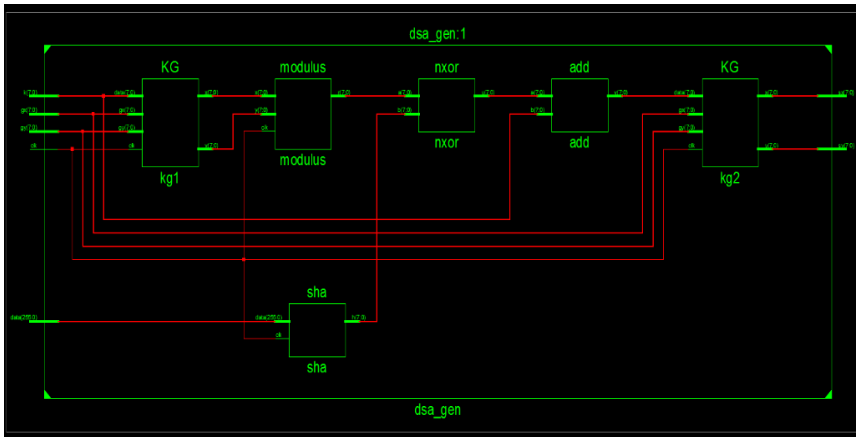

Fig. 9 RTL Block Schematic of modified ECDSA signature generation

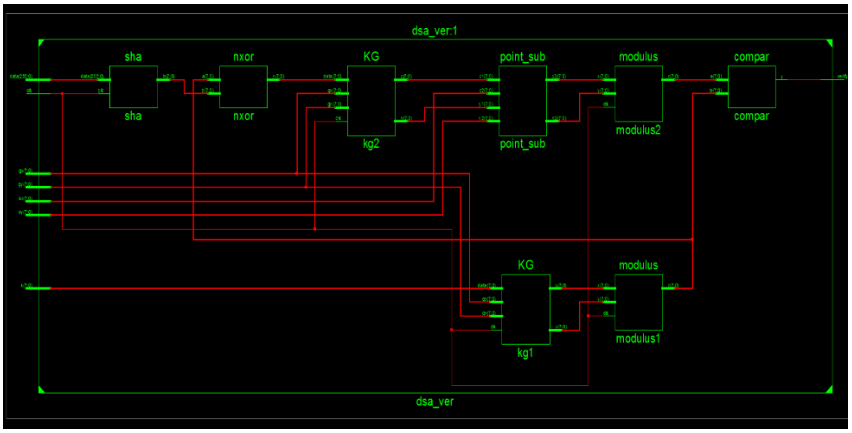

Fig. 10 RTL Block Schematic of Modified ECDSA signature verification

The Modified ECDSA signature generation and signature verification are synthesized and the device utilization summary, timings summary and memory utilization are tabulated in the Table 4.

Table 4 Synthesis Summary for modified ECDSA

Parameters	Signature Generation	Signature Verification
Slice Registers	198	454
Slice LUTs	7853	16387
LUT-FF pairs	152	346
Bonded IOBs	41	34
Real Time	924.00 secs	910.00 secs
CPU Time	923.69 secs	910.09 secs
Maximum Frequency	13.469 MHz	13.156 MHz

8. Result Analysis of Elliptic Curve Digital Signature Algorithm

The ECDSA and its variants are synthesized and analyzed using XILINX tool. The Table 5 and Table 6 illustrate the values obtained after synthesizing Original ECDSA and modified ECDSA Scheme for signature generation and signature verification. Comparison was performed related to maximum frequency and number of Slice LUTs.

Table 5 Comparison of Synthesis results of ECDSA Signature Generation

Parameters	Original ECDSA	Proposed ECDSA
Number of Slice LUTs	16370	7853
Max. Frequency(MHz)	13.180	13.469

Table 6 Comparison of Synthesis results ECDSA Signature Verification

Parameters	Original ECDSA	Proposed ECDSA
Number of Slice LUTs	22734	16387
Max. Frequency(MHz)	13.210	13.156

The outcomes obtained show that the modified ECDSA scheme is better suitable for resource constrained devices. The maximum achievable frequency of 13.469 MHz is achieved for signature generation and maximum achievable frequency of 13.156 MHz is achieved for signature verification on Virtex-5 (XC5VLX50T-1FF1136) FPGA board is better than the existing ECDSA schemes. Based on the design metric such as Frequency (MHz) and Area (Slices/ALUTs), the modified ECDSA outperforms the existing ones in terms of time for execution and Slice LUTs required in FPGA device.

9. CONCLUSION

Elliptic Curve Digital Signature Algorithm (ECDSA) is one of the primitives of Elliptic Curve Cryptography (ECC). The SHA-3 algorithms like Keccak provide better security and proves beneficial wherever security constraints have to be achieved. Here a variant of Keccak-f [1600] having five steps (θ step, ρ and π step, χ step and τ step) repeated 24 times were applied to generate hashed output. Generally, Modular Inversion is computed using Montgomery's method which consists of a GCD operations. The GCD operation utilizes more number of arithmetical operations. Thus computational cost increases when implemented on FPGA as number of operations increases. From the analysis, it is found that ECDSA is vulnerable to MIM attack when the same key is applied for all messages. At the same time if computational cost is reduced, then there are chances of signature being forged by the intruder. Therefore, the modified ECDSA scheme keeps the mathematical structure of ECDSA and security the same as the original ECDSA scheme, but reduces the computational cost by reducing the inverse operation being applied in the key generation and signing phase. Also this scheme solves the problems related to signature forging due to the available public parameters (G, n, p, Q_S). These are achieved by using hidden generator concept. Hence this scheme has more security with less computational cost, therefore can be implemented in the perceptual layer devices in IoT i.e., the ECDSA can be applied for securing the information communicated by devices such as WSNs, RFIDs, etc., having limited memory and computational capacity. Since FPGAs are used as end products, the design of ECDSA is fine-tuned for FPGA implementation. The work can be extended by considering advanced FPGAs where parallelism can be exploited in the architecture to reduce the delay in the asymmetrical cryptography.

REFERENCES

[1] N. Koblitz, A. J. Menezes, and S. A. Vanstone, "The state of elliptic curve cryptography", *Design, Codes, and Cryptography*, vol. 19, Issue 2-3, pp.173-193, 2000.

[2] V. Miller, "Use of elliptic curves in cryptography", Advances in Cryptography-Crypto '85. LNCS 218, Springer Verlag, 1986, pp. 417-426.

[3] G. Provelengios, P. Kitsos, N. Sklavos, and C. Koulamas, "FPGA-Based Design Approaches of Keccak Hash Function," In Proceedings of the 15th Euromicro Conference, 2012, pp. 648-653.

[4] D. Manel, O. Raouf, H. Ramzi and A. Mtibaa, "Hash Function and Digital Signature based on Elliptic Curve", In Proceedings of the 14th international conference on Sciences and Techniques of Automatic control & computer engineering - STA'2013, Sousse, Tunisia, December 20-22, 2013 pp. 388-392.

[5] K. Latif, M. M. Rao, A. Aziz, and A. Mahboob, "Efficient Hardware Implementations and Hardware Performance Evaluation of SHA-3 Finalists," In Proceeding of 3rd SHA-3 Candidate Conference, March 2012.

[6] S. P. Raj, A. P. Renold, "An Enhanced Elliptic Curve Algorithm for Secured Data Transmission in Wireless Sensor Network", In Proceedings of Global Conference on Communication Technologies (GCCT 2015), pp. 891-896.

[7] A. Khalique, K. Singh, S. Sood, "Implementation of Elliptic Curve Digital Signature Algorithm", *International Journal of Computer Applications*, vol. 2, no. 2, pp. 21-27, May 2010.

[8] E. Wajih, B. Noura, M. Mohsen & T. Rached, "Low Power Elliptic Curve Digital Signature Design for Constrained Devices", *International Journal of Security (IJS)*, vol. 6, no.2, pp. 1-14, April 2012.

[9] G. Sarath, D. C. Jinwala and S. Patel, "A Survey on Elliptic Curve Digital Signature Algorithm and its Variants", *Computer Science & Information Technology (CS & IT) –CSCP*, 2014, pp. 121–136.

[10] A. I. Ali, H. P. Isitc, "Comparison and Evaluation of Digital Signature Schemes Employed in NDN Network", *International Journal of Embedded systems and Applications (IJESA)*, vol. 5, no. 2, pp. 15-29, June 2015,

[11] H. Junru, "The improved elliptic curve digital signature algorithm", In Proceedings of the International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011, pp. 257-259.

[12] B. Panjwani, D. C. Mehta, "Hardware-Software Co-design of Elliptic Curve Digital Signature Algorithm over Binary Fields", In Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2015, pp. 1101-1106.

[13] X. Zhang, S. Ma, W. Shi, and D. Han, "Implementation of Elliptic Curve Digital Signature Algorithm on IRIS Nodes", In Proceedings of the International Conference on Estimation, Detection and Information Fusion (ICEDIF 2015), pp. 403-406.