processes

Article

Unforgeable Digital Signature Integrated into Lightweight Encryption Based on Effective ECDH for Cybersecurity Mechanism in Internet of Things

Adel A. Ahmed * and Omar M. Barukab

Information Technology Department, Faculty of Computing and Information Technology-Rabigh, King Abdulaziz University, Jeddah 25729, Saudi Arabia
* Correspondence: aaaabdullah1@kau.edu.sa; Tel.: +966-563884738

Abstract: Cybersecurity protocols enable several levels of protection against cyberattacks (digital attacks) that spread across network devices, platform programs, and network applications. On the Internet of Things (IoT), cyberattacks are generally intended to access and change/destroy sensitive information, which may reduce IoT benefits. Moreover, recent IoT systems are experiencing a critical challenge in designing a lightweight and robust cybersecurity mechanism on resource-constrained IoT devices. The cybersecurity challenges facing the IoT that should be taken into consideration are identifying compromised devices, data/service protection, and identifying impacted IoT users. This paper proposes an unforgeable digital signature integrated into an effective lightweight encryption (ELCD) mechanism that utilizes the secure key distribution in an elliptic curve Diffie–Hellman (ECDH) and resolves the weak bit problem in the shared secret key due to the Diffie–Hellman exchange. The ELCD mechanism proposes a secure combination between the digital signature and encryption, and it uses fast hash functions to confidentially transfer a shared secret key among IoT devices over an insecure communication channel. Furthermore, the ELCD mechanism checks the true identity of the sender with certainty through the proposed digital signature, which works based on a hash function and three steps of curve-point inspection. Furthermore, the security of ELCD was mathematically proven using the random oracle and IoT adversary models. The findings of the emulation results show the effectiveness of ELCD in terms of CPU execution time, storage cost, and power consumption that are less by 53.8%, 33–17%, and 68.7%, respectively, compared to the baseline cryptographic algorithms.

Keywords: IoT; ECDH; digital signature; random oracle model

check for updates

Citation: Ahmed, A.A.; Barukab, O.M. Unforgeable Digital Signature Integrated into Lightweight Encryption Based on Effective ECDH for Cybersecurity Mechanism in Internet of Things. *Processes* **2022**, *10*, 2631. https://doi.org/10.3390/pr10122631

Academic Editors: Fabricio Napoles-Rivera and Yo-Ping Huang

Received: 3 November 2022
Accepted: 2 December 2022
Published: 7 December 2022

1. Introduction

The Internet of Things (IoT) facilitates communication capabilities to the electronic devices and a variety of objects/things that can access the Internet. Working devices can be configured with a unique IP address to implement numerous smart applications without human intervention. Moreover, IoT devices are extremely heterogeneous, differ in capabilities, and have very limited resources in terms of connectivity, source of energy, processing and memory capabilities, and input/output hardware features [1]. The general architecture of the IoT is illustrated in Figure 1. In this figure, the event area depends on real applications that support civilian and military environments such as industrial, medical, smart home, and transportation. Wireless sensor nodes are attached to electronic devices or objects (called IoT devices) to collect measurement data from the environment and transfer it to an IoT gateway or to actuators. Upon receiving a command from the gateway or sensors, the actuators can intervene to change physical conditions such as by controlling equipment, switching lights on/off, and increasing/decreasing engine rotation speed. Moreover, a gateway provides connectivity between sensors/actuators and the remote IoT device, and it also facilitates data compression, secure data transmission, and compatibility communication between the event area sensors/actuators and the IoT remote devices [2–6].

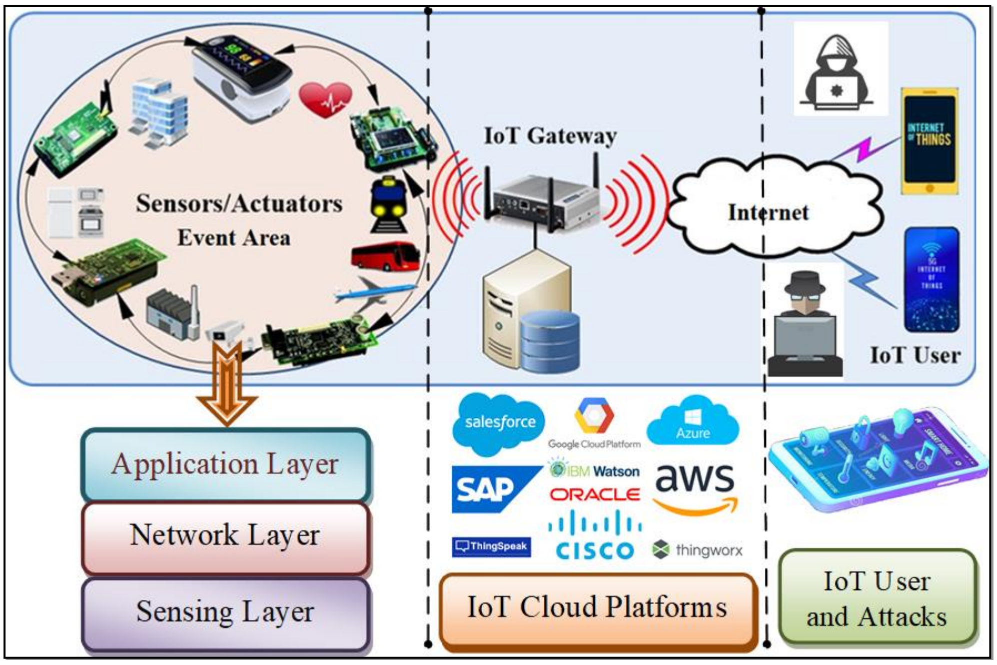


Figure 1. IoT general architecture.

The cybersecurity issue remains a significant barrier to IoT adoption and deployment due to the vulnerability of software privacy and hardware cyberattacks. Generally, cyberattacks use the internet for destroying, disrupting, disabling, and gaining unauthorized access to critical IoT information. Regardless of network-structure layers, the IoT is susceptible to numerous types of cyberattack at the sensing layer, the application layer, and the network layer. For instance, cyberattacks cause numerous effects on the IoT, which might include sensor capture, stolen-verifiers and controlled information, known secure key, denial of service (DoS), link sniffing, man-in-the-middle, forced delay, session hijacking, as well as brute force and dictionary attacks [7–10].

Encryption and digital signatures are the two essential solutions for cyberattacks on the IoT. Recent IoT cryptographic tools use two types of encryptions: the symmetric (private key) and the asymmetric (public key) encryption methods. Symmetric key algorithms apply the same key at the source and at the destination of the cryptographic process. Indeed, the strong-point of symmetric key encryption relies on distribution of the private key among IoT devices. Nevertheless, asymmetric key algorithms use two unique keys that are public and private. The private key is kept secret and never distributed while the public key can be sent through a secure channel to valid entitled devices. The main advantage of asymmetric key cryptography is the construction of a digital signature, which is used to verify the original sender, prevent the sender from disowning the message, and prove the integrity of the message. However, the drawback of standard digital signatures is the uncertainty in confirmation of the sender's identity. This means the sender private key could be counterfeited by an attacker private key, which can be used on the digital signature to mimic the identity of the true sender. Moreover, the implementation of standard asymmetric key algorithms in resource-constrained devices (e.g., IoT) is more complex and experiences more energy consumption and latency compared to symmetric key algorithms. Therefore, symmetric key cryptography that uses an efficient key distribution scheme can provide a promising lightweight cybersecurity solution for IoT networks. Furthermore, symmetric key cryptography uses the hash function to provide sender authenticity and digital fingerprint (integrity) for the distributed keys and plaintexts.

Efficient key distribution is the dilemma of symmetric key cryptography, and it becomes the most significant challenge in resource-constrained devices such as in IoT systems. One of the practical solutions is to use an elliptic curve Diffie–Hellman (ECDH) scheme, which is considered an appropriate solution for resource-constrained devices. The ECDH has a smaller key size, a more efficient source code and lower power consumption compared with the Rivest–Shamir–Adleman (RSA) cryptosystem. The common key can be directly derived from the shared secret key and used to encrypt subsequent data flow between source and destination using a symmetric key cipher.

The cryptosystem's standard solutions (e.g., RSA, AES, DES) require heavy computational overhead and cause longer processing latency. Hence, they cannot be applied directly to most resource-constrained devices such as IoT devices or sensors. Thus, it is a challenging task to develop lightweight, fast, and efficient secure cryptographic mechanisms for the IoT that can verify the identity of the sender and prevent the revelation of sensitive information by unauthorized attackers [11–16].

1.1. Research Problem Statement

As IoT devices have limited power capacity and processing capability, they are not capable of implementing complex mathematical operations. Therefore, the complicated security approaches that rely on long key size and complex encryption and decryption processing (e.g., RSA) are not a good solution for resource-constrained IoT devices [5]. The main challenge of developing an unforgeable digital signature comprises the consideration of resource-constrained of IoT devices, verification of the signer's true identity, and resolving the problems of key distribution and weak bits in the shared secret key.

1.2. Research Contribution

The following contributions are reported in this research:

- It proposes an efficacious digital signature, which confirms the true identity of the sender with certainty using a hash function and the three steps of curve-point inspection based on the ECDH scheme.
- It proposes a secure combination between encryption and digital signature, and it studies the weaknesses of other combinations.
- It proposes a lightweight encryption mechanism that can resolve the weak bit problems in the shared secret key due to the Diffie–Hellman exchange.
- It proposes secure key distribution among a group of IoT devices that can confidentially transfer a shared secret over an insecure communication channel. The shared group secret key (SGSK) is an ephemeral (dynamic) entity and is calculated on the basis of the ECDH scheme. Moreover, perfect forward secrecy, which is recommended by the RFC8442 standard, can be achieved using the ephemeral shared key.
- A comprehensive probabilistic mathematical cryptanalysis using the random oracle model (ROM) is employed to prove the security of the proposed digital signature and the encryption in the IoT.
- Finally, several simulation experiments have been conducted to evaluate the performance of the proposed mechanisms in terms of processing time, storage cost, and energy consumption. Overall, the proposed system offers faster processing times, less memory and energy consumption, and an effective method of key distribution.

The rest of this paper is arranged as follows: Section 2 presents the related works on encryption and digital signature based on the IoT platform. Section 3 describes the construction of system design for all models in the ELCD mechanism. Section 4 explains cybersecurity analysis based on ROM for ELCD mechanisms. Section 5 explains the emulation experiments and evaluation of the ELCD mechanism based on the IoT system. Finally, the conclusion and future work are explained in Section 6. All the notation used in the ELCD mechanism is summarized in Table 1.

Table 1. Frequently used notation.

Notation	Meaning	Notation	Meaning
C	Ciphertext	M	Plaintext message
CCA	Chosen cipher attack	m	The integer number of M
CPA	Chosen plaintext attack	n	Order of G
d	Private key	O	An extra point at infinity of the curve
D	Destination node	P	Modular prime
DS	Digital signature	Pb	Random point in the curve
ECC	Elliptic curve cryptography	$Pb.X_1$	X coordinate of Pb
ECDH	Elliptic curve Diffie–Hellman	Q	Public key
ELCD	Effective, lightweight cryptographic and digital signature	S	Source node
G	Base point generator	SGSK	Shared group secret key

2. Related Works on Encryption and Digital Signature

As the IoT is still a relatively new technology, so far only a limited number of algorithms in the literature have been developed to fit resource-constrained devices such as actuators, sensors, etc. In an earlier work presented by A. A. Ahmed called ESSC_DC [17], a digital certificate authority was used to verify the true identity of the sender by linking a public key to its owner. The advantages of the proposed algorithm in this paper compared to the previous research in [17] can be described as follows:

- The new proposed combination between digital signature and encryption has the ability to verify the true identity of the sender and receiver, which is considered the main function of the digital certificate in [17].
- The verification process in the establishment the digital certificate necessitates tedious work, consumes excessive power, and introduces extra processing delay if used in the IoT platform.
- The QoS performance of the proposed algorithm outperforms the QoS performance of the digital certificate in [17]. This is mainly due to the fact that the digital certificate authority, which is responsible for verifying the digital certificate, consumes excessive power and increases the end-to-end delay as will be explained in Section 5.

Thus, the works related to this paper emphasize the use of lightweight digital signature and cryptographic algorithms in the IoT network.

2.1. Lightweight Digital Signatures on IoT

Elliptic curve cryptography (ECC) has been used in numerous encryption algorithms, including public key algorithms such as the elliptic curve digital signature algorithm (ECDSA) [18]. This latter algorithm was recognized as an ISO standard, an ANSI standard, and as a combined IEEE and NIST standard in 1998, 1999, and 2000, respectively. However, the ECDSA has technical problems summarized in its slowness, design flaws and insufficiently defensive implementations of the random number generator. Arif et al. [19] proposed the shortened complex digital signature algorithm (SCDSA), which secures the transmission channel between the sender and the receiver in a human-centered IoT. Moreover, Park et al. [20] proposed an enhanced version of the elliptic-curve Qu–Vanstone (ECQV) protocol, which uses the idea of a certificate-issuing protocol to resolve the problems of an implicit certificate for establishing lightweight security association on IoT systems. However, this requires protecting the certificate request between the sensor device and the certificate authority. Furthermore, Atefeh et al. [21] proposed a secure data-sharing mechanism for device-to-device communication on 5G mobile systems. In their research, the virtual check concept was applied as an encouragement system to stimulate manipulator participation in the development of data sharing. Moreover, Adeel et al. [22] proposed a secure authentication mechanism that relies on elliptic ElGamal encryption. In [22], researchers integrated the elliptic curve cryptosystem (ECC) with a public key infrastructure (PKI) to produce a shared key pair that is exchanged between IoT devices.

Moreover, the research offered by Yasir et al. [23] integrated the ECC and ElGamal schemes over a public key infrastructure (EEoP). Furthermore, Adel et al. [24] proposed an effective multifactor authentication (CMA) system that utilizes the idea of the combination of several hash functions and geolocation authentication over IoT. Likewise, Scincalepore et al. [25] developed a key management protocol (KMP), which combines the ECDH scheme with digital certificates to authenticate the key generation.

2.2. Lightweight Encryption Algorithms on IoT

V. Shoup proposed the elliptic curve integrated encryption (ECIES) mechanism, which is integrated with an advanced standard encryption scheme of called ECIES_AES. Moreover, ECIES is integrated with rabbit encryption as described in the RFC4503 standard, which is called ECIES_Ra. A lightweight authenticated encryption with associated data (AEAD) has been suggested by NIST, which was proposed to work with a resource-constrained device (e.g., IoT system) [26]. The AEAD scheme offers a cipher and a tag, which can be considered a message authentication code (MAC). Hence, AEAD offers data authentication, confidentiality, and data integrity. For example, Seok et al. [27] developed secure device-to-device communication using the idea of AEAD and ECC to fit an IoT resource-constrained system. Khan et al. [28] developed a secure ECC-based authentication and encryption method that utilizes user credentials and biometric parameters to improve user authentication. Muhammad et al. [29] proposed secure IoT (SIT), which uses the idea of combining a Feistel 64-bit key cipher and a uniform substitution permutation. The integration of authentication and cryptography based on the Diffie–Hellman scheme was presented in Shah et al. [30]. The authentication uses multifactor authentication to share a secret key over the IoT system. Hammi et al. [31] proposed a one-time password (OTP) that relies on ECC and isogeny to guarantee IoT security. However, the randomness of the OTP based on ECC is not ensured. Rangwani et al. [32] proposed a secure system with privacy and authentication based on a three-factor authentication protocol for the industrial IoT (IIoT).

The limitations of the previous literature studies [18–32] are summarized in Table 2. In this table, the main limitations can be classified into three facts: Firstly, most of the research studies did not consider the combination of encryption and a digital signature, resource-constrained hardware, and the outstanding architecture of the IoT. Secondly, the vulnerabilities of ECDH (i.e., weak bits and chosen ciphertext attack) have not been investigated. Finally, the resource-constrained hardware issue has not been carefully taken into consideration in the context of digital signature and encryption mechanism design.

Table 2. Summary of related works.

Approaches	Year of Publication	Methodology and Features	Limitations
ECDSA [18]	2001	Proposed an elliptic curve-based digital signature algorithm.	Slowness, design flaws, and insufficiently defensive.
SCDSA and MPS-SCDSA [19]	2018	Secured communication between smart devices in IoT.	Needs high processing resource and consumes extra energy.
C.S. Park et al. [20]	2018	Proposed an enhanced version of the elliptic curve Qu–Vanstone (ECQV) certificate issuance protocol.	Consumes more power and latency due to verification of certificate at the certificate authority.
Adeel et al. [22]	2019	Merged two algorithms: ECC to manage public key infrastructure (PKI), and ElGamal to implement encryption.	Lacks adversary mode analysis.
Yasir et al. [23]	2017	Developed a tiny cryptographic system that depends on ECC and ElGamal.	The cryptanalysis was not studied.

Table 2. *Cont.*

Approaches	Year of Publication	Methodology and Features	Limitations
KMP [25]	2017	Integrated the ECDH exchange with a digital certificate to authenticate the key generation.	Does not fit IoT resource-constrained limitations due to power consumption of implied certificate.
B. Seok et al. [27]	2020	Proposed a secure device-to-device communication using the idea of AEAD and ECC to fit an IoT resource-constrained system.	The cryptanalysis was not studied.
M. Ayoub et al. [28]	2020	Developed a secure ECC based authentication and encryption that utilizes user credentials and biometric parameters to improve user authentication.	Does not fit IoT resource-constrained limitations due to vulnerability to error of biometric parameters.
SIT [29]	2017	Used the idea of combining Feistel 64-bit key and a uniform substitution permutation.	Does not fit IoT resource-constrained limitations due to power consumption.
Shah et al. [30]	2017	Combined the authentication and the cryptography based on Diffie–Hellman to distribute a secret key among IoT devices.	Does not prove the security of the combination.
B. Hammi et al. [31]	2020	Proposed OTP that relies on elliptic curve cryptography and isogeny.	The randomness of the OTP based on ECC is not ensured.

3. System Design of the ELCD Algorithm

The proposed ELCD mechanism mainly contains three functions: key management based on the ECDH scheme, encryption algorithms with a random padding scheme, and digital signatures based on the multifactor of a hash function. The former two functions have been combined to guarantee a high degree of security strength against cyberattacks on IoT systems. The following assumptions have been used to design the three proposed functions throughout this paper:

- The IoT gateway has a strong security protection system, and it is extremely hard to compromise.
- Each IoT device (sensor, actuator, remote IoT user, etc.) has two secure keys: a public key, which is available to all involved IoT devices, and a private key, which is not known publicly.
- The domain parameters of the ECDH are embedded and uploaded into all IoT devices during the programming session, which means that the ELCD mechanism is very suitable in industrial IoT (IIoT) applications.

The following sections explain the details of the proposed ELCD and how it functions on IoT networks.

3.1. The Key Management Algorithm

The main problem in traditional symmetric key cryptography is the exchange of the common key between the IoT devices over an insecure communication channel, which makes IoT devices susceptible to many attacks. Thus, the proposed key management algorithm uses the ECDH scheme with a dynamic shared key calculation to securely come to agreement with a fresh new secret key for each session between IoT devices (i.e., forward secrecy). Both ECC and ECDH schemes have been utilized to generate a shorter, secure shared key that is more appropriate for IoT devices. The elliptic curve is a set of points that are identified by solving the following equation:

$$\begin{gathered} E = \left\{(x,y)\middle|y^2 = x^3 + ax + b\right\} \cup \{O\}, \\ \textit{where } a,b \in K(\mathbb{Z}/p\mathbb{Z}) \textit{ satisfy } \left(4a^3 + 27b^2\right) \neq 0 \end{gathered} \quad (1)$$

where K defines a finite field of integer numbers over a modular prime P. In order to perform a mathematical operation such as adding a point to itself, an extra point at infinity (e.g., O) has been added to the curve. Let us consider S and D as a source and a destination that could be a sensor, an actuator, or remote IoT user. Primarily, the domain parameters are p (the prime of the base finite field), a, b, G (the base point generator), n (the order of G), and h (the subgroup cofactor usually is 1), which demonstrate the agreed information upon S and D to utilize the ECDH key exchange protocol. The S and D should obtain the private key d, which is determined using the random generator function between 1, and $n - 1$. Figure 2 illustrates the ECDH key's establishment and the process of exchanging the public key between two IoT devices in order to compute a shared secret key over an insecure communication channel. For the public key, a point Q is calculated as a scalar multiplication of d and G (e.g., $Q = d \times G$). Let the S key-pair be (d_S, Q_S) and the D key-pair be (d_D, Q_D). Each party of connection has to receive the other party's public key prior to the implementation of the ECDH protocol. Hence, S computes point $K(X_K, Y_K) = d_S \times Q_D$ and D computes point $K(X_K, Y_K) = d_D \times Q_S$. As a result, the shared secret key is X_K, which represents the x coordinate of the point K. The shared secret key that is calculated by both parties is equal because $d_S \times Q_D = d_S \times d_D \times G = d_D \times d_S \times G = d_D \times Q_S$. It is interesting to note that " $\times$ " is used to denote elliptic curve scalar multiplication. Moreover, the public key Q, the private key d, and shared secret key (X_K) in the proposed algorithm are ephemeral (dynamic), which means that they change with every new session between S and D.

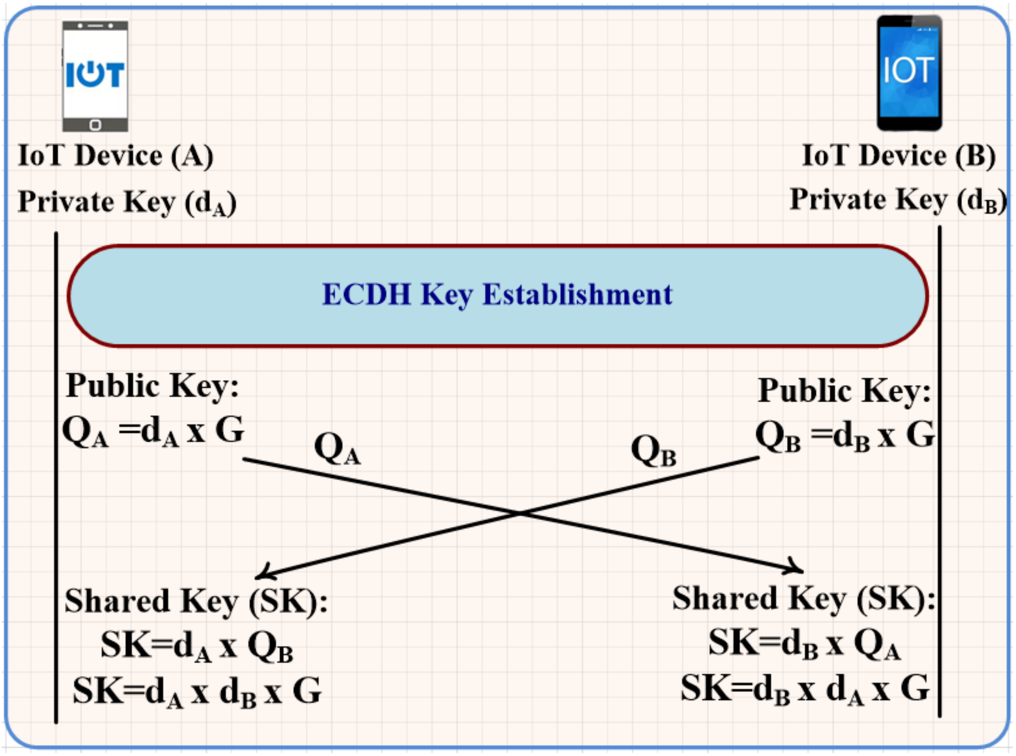


Figure 2. The ECDH key establishment process.

- **Shared Group Secret Key (SGSK)**

The proposed key distribution mechanism between a pair of IoT devices was modified to be applicable to a group of IoT devices. This eventually provides an advantage for the ELCD scheme compared to ECDH. For instance, the shared secret key can be calculated for five IoT devices (*IoT_Ds*) as shown in Figure 3 as follows:

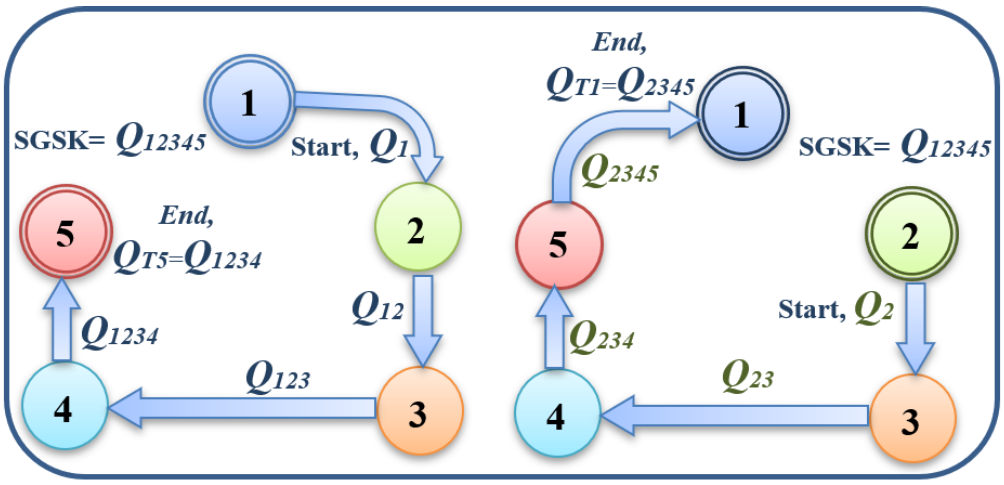


Figure 3. Shared group secret key.

- ***IoT_D*$_1$** creates the first part of the group public key as $Q_1 = d_1 \times G$.
- ***IoT_D*$_1$** sends Q_1 to the next *IoT_D* (e.g., *IoT_D*$_2$), which creates the second part of the group public key as $Q_{12} = d_2 \times Q_1$.
- This scenario continues until the last *IoT_D* (e.g., *IoT_D*$_5$) receives the previous parts of the group public key (Q_{1234}), which is considered as the total group public key created at *IoT_D*$_5$ ($Q_{T5} = Q_{1234}$). The total group public key at *IoT_D* number v can be generalized for any number of nodes in the group as:

$$Q_{Tv} = G\prod_{i=1}^{N}(d_i|d_v) \tag{2}$$

where N represents the total number of *IoT_D* in the group. For example, we can calculate Q_{T3} based on five *IoT_D*s as $d_1 \times d_2 \times d_4 \times d_5 \times G$.

- Finally, the SGSK can be calculated at the *IoT_D* number v as $(X_K, Y_K) = d_v \times Q_{Tv}$. For example, the SGSK at *IoT_D* number *3* can be calculated as $d_3 \times Q_{T3}$ while the SGSK at *IoT_D* number *4* can be calculated as $d_4 \times Q_{T4}$. It is interesting to note that $d_3 \times Q_{T3} = d_4 \times Q_{T4}$, because $d_3 \times Q_{T3} = d_3 \times d_1 \times d_2 \times d_4 \times d_5 \times G = d_4 \times d_1 \times d_2 \times d_3 \times d_5 \times G = d_4 \times Q_{T4}$. Each *IoT_D* should verify the received Q_{Tv} before performing any calculation. Therefore, if a malicious node exists during the key distribution phase, it does not know the domain parameters (i.e., n and G) and then it cannot calculate the appropriate Q_{Tv}. Therefore, each IoT device must receive Q_{Tv} before creating the SGSK and performing the digital signature and cryptographic algorithms.

3.2. The Lightweight Encryption Algorithm

An important function that prevents the disclosure and unauthorized reading of the digital signature is encryption. Therefore, confidentiality of sent messages including the IoT data and digital signature should be designed as a complete system. The first stage in the proposed system is performing the encryption algorithm at the source node using the following steps:

- Calculate a hash function for the shared secret key X_K such as E = StrToInt(Hash(X_K)), where the hash function represents a cryptographic hash function such as CMA [24] or SHA-256 [33].
- Calculate the curve point $Pb(X_1, Y_1) = E \times G$, which is hard to reverse because the scalar multiplication in the ECC has a one-way function property.
- Calculate the ciphertext $C = (m \times X_1) \bmod n$, where m is obtained by converting M to an integer number using a padding scheme, which should be an agreed-upon reversible protocol. In this paper, each M has been parsed to multiple chunks based

on the message size in an elliptic curve (e.g., Secp192r1) [34]. This means that the maximum length of each chunk is 127 bytes, and the minimum length is 24 bytes.

The decryption steps of ELCD at the destination node upon receiving C is performed as follows:

- Calculate E = StrToInt(Hash(X_K)), where Hash represents the corresponding cryptographic hash used in the authentication code calculation.
- Calculate the curve point $Pb(X_1, Y_1) = E \times G$.
- Calculate the integer number of the chunk $m = (C \times {X_1}^{-1}) \bmod n$, where ${X_1}^{-1} \bmod n$ can be resolved using a modular multiplicative inverse.
- Convert the integer number m to the parse (i) of plaintext (M), where i is the parse number. The concatenation of all parses should recreate the original message M.

3.3. The Proposed Digital Signature Algorithm

The traditional digital signature has a serious drawback in verifying the true identity of the sender. For instance, an adversary could intercept the transmitted message along with the digital signature and she/he could create her/his own set of public and private keys using the sender's identity. After that, the adversary would pretend to be a legitimate IoT sender and create a fictitious message with a different digital signature. Upon receiving the message and the digital signature, the receiver would unknowingly retrieve the imposter public key (thinking it belonged to the sender) and decrypt it. This problem can be solved using the digital certificate system, which can verify the true identity of the sender with certainty; however, a digital certificate requires more power consumption and causes more latency due to third party verification (digital certificate authority). In contrast, the proposed system resolves this problem using the three steps of curve-point inspection and the integration idea between the encryption and the digital signature as described in Figure 4. The proposed digital signature consists of six phases, three at the source IoT node and three at the destination IoT device.

The source node implements the following steps:

- **Phase 1, Create Digest.**
  (1) Convert the plaintext message (M) to an integer number m using an agreed-upon reversible protocol identified as a padding scheme.
  (2) Calculate the digest for m as Z = StrToInt(Hash(m)) mod n, where the Hash represents a cryptographic hash such as CMA [24] or SHA-256 [33].
- **Phase 2, Create DS.**
  (1) Encrypt the digest with the sender private key d_S to calculate the digital signature as $DS = ({d_S}^{-1} \times Y_K \times Z) \bmod n$, where Y_K represents the y-coordinate of the shared secret key.
  (2) Since d_S is a random number (ephemeral) that changes every session, DS also changes in each session.
- **Phase 3, Encrypt Message with DS.**
  (1) Encrypt the concatenation of m and DS with the shared secret key X_K. The proposed lightweight encryption algorithm is used to encrypt ($m + DS$), and the final output is the ciphertext (C).
  (2) Calculate E = StrToInt(Hash(X_K)).
  (3) Calculate the curve point $Pb(X_1, Y_1) = E \times G$.
  (4) Calculate $C = ((m + DS) \times X_1) \bmod n$.

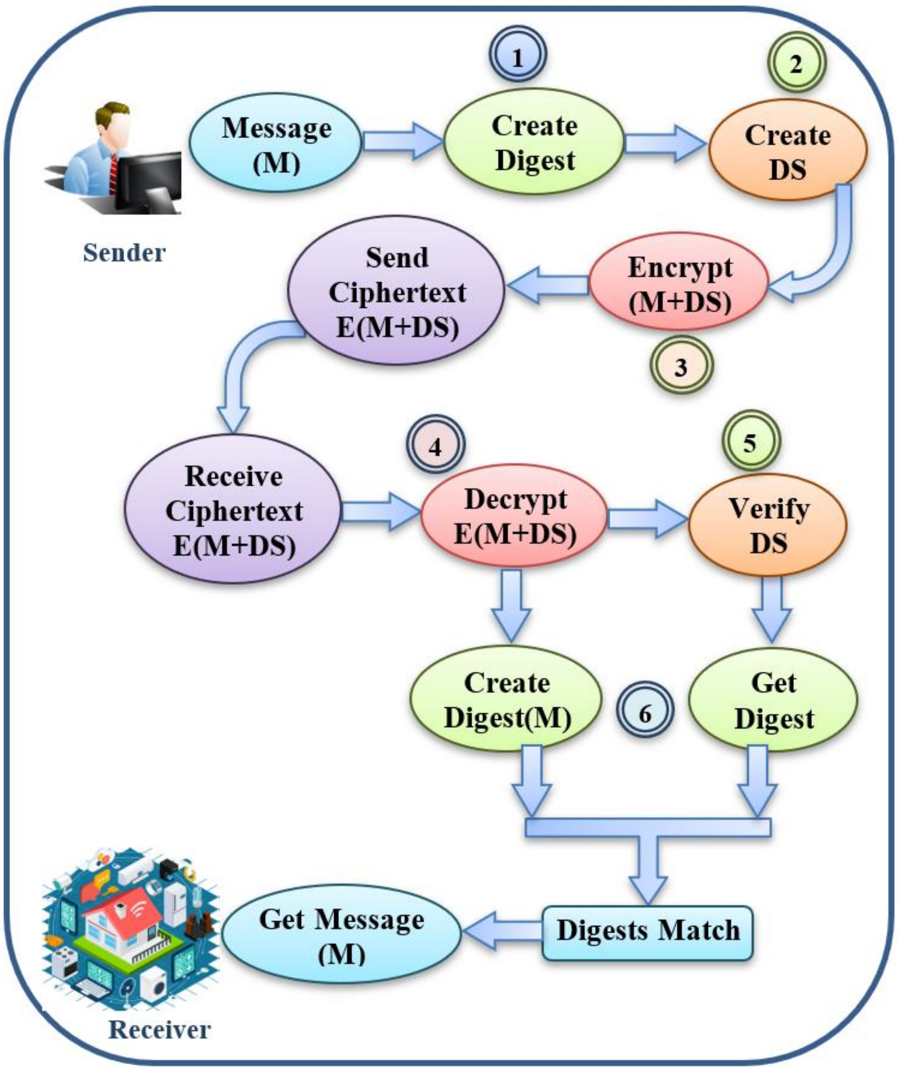


Figure 4. The six phases in the proposed digital signature.

In order to authenticate the digital signature, the following steps are implemented at the destination:

- **Phase 4, Decrypt Cyphertext.**
  (1) Calculate E = StrToInt(Hash(X_K)).
  (2) Calculate the curve point $Pb(X_1, Y_1) = E \times G$.
  (3) Calculate the concatenation of m and DS: $(m + DS) = (C \times {X_1}^{-1}) \bmod n$.
- **Phases 5 and 6, DS Verification and Obtaining the Message.**
  (1) Verify the true identity of the sender that is used in signing the plaintext (Y_K) using three steps of curve-point inspection: 1. Check that Q_S is not equal to the identity element O, 2. Check that Q_S lies on the curve 3. Check that $n \times Q_S = O$.
  (2) Retrieve the digest from the received DS as $U1 = (DS \times {Y_K}^{-1} \times Q_S) = Z \times G$. This works because $(DS \times {Y_K}^{-1}) \times Q_S = (({d_S}^{-1} \times Y_K \times Z) \times {Y_K}^{-1} \times (d_S \times G)$ $= Z \times G$, since the product of an element's inverse and the element itself is the identity.
  (3) Create the digest for the received m as Z^{-} = StrToInt(Hash(m)) mod n.
  (4) If the created $Z^{-} \times G$ = the received $Z \times G$, the received DS is valid, otherwise the received DS is invalid.
  (5) If the received DS is valid, accept the received message m and covert it back to M.

The advantages of the proposed lightweight digital signature can be summarized as follows:

(a) Domain parameters are not publicly exchanged between IoT devices; rather, they are uploaded into all devices during the programming session. This means that the attacker cannot create a valid public key Q.
(b) The verification of DS in the proposed method can verify the true identity with certainty. This means that the attacker needs to solve the elliptic curve discrete logarithm problem (ECDLP), which makes it extremally hard to reverse DS and obtain the private key for making a fake private key.
(c) The padded message m is hashed and digitally signed using the ECC and inverse modular multiplication of the sender private key, which is considered extremely hard to reverse and obtain the original message from.
(d) More importantly, the CMA hash function [24] creates a random digest for any two similar input messages. This is mainly due to the fact that the CMA is designed based on a time-enhanced-based one-time password (TEOTP) and it includes a salt random string to create the random digest for similar inputs.

Table 3 shows the proposed pseudocode. After the public key is calculated in each party of the IoT system, it is sent to all involved IoT devices that can calculate the shared secret key. The digital signature scheme is used in the first message of each communication session to verify the authenticity and the genuineness of those devices. If the first message with the digital signature is verified by any device, the rest of the received messages are decrypted to obtain the original plaintext. Otherwise, the received messages are discarded.

Table 3. The ELCD pseudocode.

```
                         ELCD at IoT Sender (S)
   Input: Secp192r1 domain parameters $p, a, b, G, n, h$;
   Output: $Q_S$, $DS$, $C$;//$Q_S$: public key of S, $DS$: digital signature; $C$: ciphertext
   Start Algorithm (ELCD)
1  | While (new session start) do
2  |     Pick private key ($d_S$);//$1 \leq d_S \leq n$
3  |     $Q_S$ = ($d_S \times G$);
4  |     Send_Public_key ($Q_S$);//Send the public key to destination
5  |     Receive_Public_key($Q_D$);//Receive the public key of D
6  |     K($X_K$,$Y_K$) = $d_S \times Q_D$;//calculate the shared key
                         Phase 1, Create Digest
7  |     $m$ = StrToInt($M$);//convert the plaintext to an integer.
8  |     $Z$ = StrToInt(Hash($m$)) mod $n$;//hash fun. for integer $m$
                         Phase 2, Create DS
9  |     if ($m$ is the first message) //first message of the session
10 |         $DS$ = ($d_S{}^{-1} \times Y_K \times Z$) mod $n$;//$DS$: digital signature
                         Phase3, Encrypt ($m$ + $DS$)
11 |     $E$ = StrToInt(Hash($X_K$)) mod $n$;//$E$: the hash fun. of key $X_K$
12 |     $Pb$($X_1$,$Y_1$) = $E \times G$;
13 |     $C$ = (($m$ + $DS$) $\times X_1$) mod $n$;//$C$: the ciphertext
14 |     Send("C");//The source sends "C" only to D
15 |     End;//if Statement
16 |   End;//While loop
17 End;//Algorithm
                         ELCD at IoT Receiver (D)
   Input: Secp192r1 domain parameters $p, a, b, G, n, h$;
   Output: $Q_D$, $DS$, $C$;//$Q_D$: Public key of D
```

Table 3. *Cont.*

ELCD at IoT Sender (S)

```
Start Algorithm (ELCD)
| While (new session start) do
|     Pick private key ($d_D$);//$1 \leq d_D \leq n$
|     $Q_D$ = ($d_D \times G$);
|     Send_Public_key ($Q_D$);//Send the public key to source node
|     Receive_Public_key($Q_S$);//Receive the public key from S
|     $K(X_K,Y_K) = d_D \times Q_S$; //if $Q_S$ is a valid curve point, the shared key will be calculated
|     Foreach ($C$ received and Flag == true) do
|       if (first message received) //Receive the first message
                              Phase 4, Decrypt C
|         Get($C$);//Receive the ciphertext ($C$)
|         $E$ = StrToInt(Hash($X_K$)) mod $n$;
|         $Pb(X_1,Y_1) = E \times G$;
|         $m + DS = (C \times {X_1}^{-1})$ mod $n$;
                 Phase 5 and 6, DS Verification & Obtain Message
|       Verify_Public_key($Q_S$);//Receiver will verify $Q_S$
|       $U1 = (DS \times {Y_K}^{-1} \times Q_S) = Z \times G$ mod $n$;
|       $Z^{\bar{}}$ = StrToInt(Hash($m$)) mod $n$;//$Z^{\bar{}}$ is digest for rec. m
|         if ($U1 == Z^{\bar{}} \times G$)
|               The signature is valid, and the source is legitimate;
|               Get($m$);//Obtain the m and
|         else
|               The signature is invalid, and the source is illegitimate;
|                   Flag = false;
|         End;// if Statement
|       else
|         $E$ = Hash($X_K$) mod $n$;
|         $Pb(X_1,Y_1) = E \times G$;
|         $m = (C \times {X_1}^{-1})$ mod $n$;
|       End;// if Statement
|             $M$ = Convert_IntToStr($m$);//convert $m$ to $M$.
|     End;// for loop
|   End;//While loop
End;//Algorithm
```

4. Cybersecurity Analysis

An adversary model was developed to measure the security performance of ELCD as explained in this section.

4.1. Adversary Model for ELCD on the IoT

The detriment of adversary cyberattack on the IoT is mainly focused on disrupting the control function of the IoT using one or more vulnerabilities that can be exploited by a malicious adversary to compromise the security system of the IoT environment [35–37]. The adversary is assumed to have the capability to read, transmit and forge IoT network traffic, which might dispute the sensed data, the privacy of IoT devices, and the control management of the gateway. The most crucial adversary attacks on ELCD are described as follows:

- **Spoofing attack:** The adversary intercepts or eavesdrops on IoT network traffic to obtain an IoT device's credential, which is used to gain access to the sensed information.
- **A man-in-the-middle attack:** The malicious adversary has the ability to listen to all traffic on a network and initiate a connection with any IoT devices. Furthermore, if the adversary acts as an active man-in-the-middle, it can modify the content of captured messages and resend them to the recipient.

- **A replay attack:** Instead of sending a message directly to the recipient, a replay attack makes a copy of that message and then uses it later. This is carried out by an adversary who intercepts the messages and delays, replays, or retransmits those messages.
- **A brute force attack:** The malicious adversary tries every possible mixture of letters, numbers, and characters to crack the shared secret key even if the domain parameters that are used in the ECDH scheme by both parties are extremely hard to obtain.
- **A sensor capture attack:** The impostor adversary captures a sensor node and steals the domain parameters and shared secret key to implement illegal actions on the IoT network.
- **A stolen-verifier attack:** The impostor adversary who has stolen the shared secret key from an IoT device can pretend to be an authorized device in order to launch attacks against other IoT devices, steal data, or bypass access controls.

4.2. Cryptoanalysis of ELCD

The random oracle model was developed to study the impact of the most common cryptanalysis attacks described as follows:

- **Chosen plaintext attack (CPA):** The adversary is assumed to obtain the ciphertexts for any plaintexts of their choice. Moreover, the adaptive CPA (CPA2) means that the adversary has the ability to choose the new input to the encryption of ELCD ($ELCD_E$) based on the analysis of her/his previously selected plaintext queries and their corresponding ciphertexts [38]. The definition of CPA can be represented mathematically by assuming that an adversary A gains access to an encryption oracle with any pair of equal-length messages (m_1, m_2) as input. The oracle returns a ciphertext as output.

Definition 1. *Let* $ELCD_E$ = (K , E, D) *be an encryption mechanism in ELCD, E is encryption, D is decryption, and K is the space of all keys. The advantage of indistinguishability chosen-plaintext attack (IND-CPA) of A is defined as*:

$$\begin{aligned} Adv^{in\text{-}cpa}_{ELCD_E}(A) \quad &= P_r[k \leftarrow K; C \leftarrow E_k(m_1) : A(C) = 1] \\ &-P_r[k \leftarrow K; C \leftarrow E_k(m_2) : A(C) = 1] \end{aligned} \tag{3}$$

The above equation shows that ELCD is secure if the advantage of IND-CPA is negligible, which means that A is not doing well. In contrast, ELCD_E is not secure if the advantage of IND-CPA is non-negligible, which means that A is doing well.

- **Chosen ciphertext attack (CCA):** The adversary is assumed to obtain the decryption of any ciphertext(s) of their choice. Moreover, the adaptive CCA (CCA2) means that the adversary has the ability to choose the new input to the decryption of ELCD based on the analysis of her/his previously selected queries [39].

Definition 2. *Let* $ELCD_E$ = (K , E, D) *be an encryption mechanism in ELCD, and A is an adversary who has the ability to access the encryption (E) and decryption (D) oracle. The advantage of IND-CCA of A is defined as:*

$$\begin{aligned} Adv^{in\text{-}cca}_{ELCD_E}(A) = P_r[k \leftarrow K; C \leftarrow E_k(m_b); b \leftarrow \{0,1\}; \\ b' \leftarrow A(E_k(.), D_k(.)) : b' = b] \end{aligned} \tag{4}$$

The above definition shows that the adversary has the right to unlimited access of the decryption oracle using any ciphertext C except one restriction, which is the previous returned query of their encryption oracle. Consequently, ELCD_E can be considered secure against IND-CCP if the adversary who given access to the oracles can find negligible advantage in distinguishing the two events of b (0/1).

4.3. The ELCD Cybersecurity Analysis

The ELCD scheme can offer significant security properties such as perfect forward secrecy (PFS), and it has impersonation resilience against key compromise. Since the hash function can be considered a random oracle function, ELCD uses a hash function to create a pseudorandom function (PRF). As explained in Section 3, the hash function in ELCD (i.e., CMA) utilizes the shared secret key (X_K) as an input and produces a secure random parameter ($H(X_K)$), which goes through scalar multiplication with the base point (G) to produce a random point *Pb*. This means *Pb.X1* (i.e., the x coordinate of *Pb*) is a random value that is periodically changed to defend against IND-CPA and replay attacks.

4.3.1. Proven Security for ELCD in the Random Oracle Model

The length of the shared secret key ($X_K \in \{0,1\}^L$) can be represented as $L = |X_K| = |n| = |p|$, which equals the length of used elliptic curve Secp192r1 (e.g., 192 bits). The proven security of ELCD uses ROM to instantiate the hash function as $H(.) : \{0,1\}^* \to \{0,1\}^L$.

Theorem 1. *If Pb is a* (t, ϵ)*-pseudorandom function (PRF), then the ELCD$_E$ is secure against IND-CPA.*

- **Methodology of Proof:** The contradiction methodology is used to prove Theorem 1. Let us assume an adversary *A* that runs in *PPT* exists, who breaks the security of *ELCD$_E$*. The algorithm *A* constructs a *PPT* distinguisher *B* that distinguishes the output of *Pb* from a random number with non-negligible cost. Since *Pb* is *PRF*; this contradicts the previous conclusion that *Pb* is a random function. Therefore, the original assumption is false and the *ELCD$_E$* must be secure.

Proof. Let us assume *A* attacks *ELCD$_E$* in the sense of IND-CPA and two messages M_0, M_1 are used as follows:

$$\left| \begin{array}{c} P_r[H(X_K) \leftarrow \mathbb{Z}_n^*; Pb \leftarrow H(X_K) \times G; C \leftarrow M_0 \times Pb.X1 : A(C) = 0] \\ -P_r[H(X_K) \leftarrow \mathbb{Z}_n^*; Pb \leftarrow H(X_K) \times G; C \leftarrow M_1 \times Pb.X1 : A(C) = 0] \end{array} \right| = \gamma(L) \qquad (5)$$

where $\gamma(L)$ is non-negligible. The algorithm *B* was constructed to distinguish *Pb* from the random function. This can be performed using the ability of *B* to call *Pb* to distinguish whether it is *PRF* or a completely random function. *B* works as follows: (1) pick a random $b \in \{0,1\}$; (2) *B* computes *C* = *Pb.X1* × M_b *mod n*; (3) run the experiment *A(C)* to obtain b', which denotes *A*'s guess of which the message encrypted. *A* guessed correctly (If $b = b'$) means *B* guesses *PRF* and thus can be represented by the *B* result "1". However, *A* does not guess correctly (If $b \neq b'$) if *B* guesses a random function and this can be represented by the *B* result "0". The algorithm *B* distinguishes the output of *Pb.X1* as:

$$\left| \begin{array}{r} P_r[H(X_K) \leftarrow \mathbb{Z}_n^*; Pb \leftarrow (H(X_K) \times G); y \leftarrow Pb.X1 : B(y) = 1] \\ - P_r[y \leftarrow \mathbb{Z}_n^* : B(y) = 1] \end{array} \right| \qquad (6)$$

We study each of these terms separately as: $P_1 \overset{\text{def}}{=} P_r[H(X_K) \leftarrow \mathbb{Z}_n^*; Pb \leftarrow (H(X_K) \times G); y \leftarrow Pb.X1 : B(y) = 1]$, and $P_2 \overset{\text{def}}{=} P_r[y \leftarrow \mathbb{Z}_n^* : B(y) = 1]$. In Step 3, the algorithm *B* does the following:

$$\begin{array}{r} P_1 = P_r[H(X_K) \leftarrow \mathbb{Z}_n^*; Pb \leftarrow (H(X_K) \times G); y \leftarrow Pb.X1 : \\ b \in \{0,1\}; b' \leftarrow A(Pb.X1 \times M_b) : b' = b] \end{array} \qquad (7)$$

By using the condition of b we obtain:

$$\begin{array}{c} P_1 = P_r[H(X_K) \leftarrow \mathbb{Z}_n^*; y \leftarrow Pb.X1 : A(Pb.X1 \times M_0) = 0] \times P_r[b = 0] \\ +P_r[H(X_K) \leftarrow \mathbb{Z}_n^*; y \leftarrow Pb.X1 : A(Pb.X1 \times M_1) = 0] \times P_r[b = 1] \end{array} \tag{8}$$

with applying the fact:
$P_r[b = 0] = P_r[b = 1] = \frac{1}{2}$ and

$$\begin{array}{c} P_r[H(X_K) \leftarrow \mathbb{Z}_n^*; y \leftarrow Pb.X1 : A(Pb.X1 \times M_1) = 1] = \\ 1 - P_r[H(X_K) \leftarrow \mathbb{Z}_n^*; y \leftarrow Pb.X1 : A(Pb.X1 \times M_1) = 0] \end{array} \tag{9}$$

we obtain:

$$P_1 = \frac{1}{2} + \left[\frac{1}{2} \times \left(\begin{array}{c} P_r[H(X_K) \leftarrow \mathbb{Z}_n^*; y \leftarrow Pb.X1 : A(Pb.X1 \times M_0) = 0] \\ -P_r[H(X_K) \leftarrow \mathbb{Z}_n^*; y \leftarrow Pb.X1 : A(Pb.X1 \times M_1) = 0] \end{array} \right) \right] = \frac{1}{2} + \left(\frac{1}{2} \times \gamma(L) \right) \tag{10}$$

P_2 is calculated as:

$$P_2 = P_r[y \leftarrow \mathbb{Z}_n^* : b \in \{0,1\}; b' \leftarrow A(Pb.X1 \times M_b) : b' = b] \tag{11}$$

As before, we eventually obtain:

$$P_2 = \frac{1}{2} + \left[\frac{1}{2} \times \left(\begin{array}{c} P_r[y \leftarrow \mathbb{Z}_n^* : A(Pb.X1 \times M_0) = 0] \\ -P_r[y \leftarrow \mathbb{Z}_n^* : A(Pb.X1 \times M_1) = 0] \end{array} \right) \right] \tag{12}$$

Since y is completely random and $Pb = H(X_K) \times G$, the probability of A winning when breaking the one-time pad is 0. Therefore, P_2 is $1/2$. The final result after using all parameters together provides:

$$\begin{array}{c} \left| \begin{array}{l} P_r[H(X_K) \leftarrow \mathbb{Z}_n^*; Pb \leftarrow (H(X_K) \times G); \\ y \leftarrow Pb.X1 : B(y) = 1] - P_r[y \leftarrow \mathbb{Z}_n^* : B(y) = 1] \end{array} \right| = |P_1 - P_2| \\ = \left| \frac{1}{2} + \frac{\gamma(L)}{2} - \frac{1}{2} \right| = \frac{\gamma(L)}{2} \end{array} \tag{13}$$

Since $\gamma(L)$ was non-negligible, the term $\frac{\gamma(L)}{2}$ is also non-negligible. This leads to the fact that A has a non-negligible advantage in breaking $ELCD_E$ and hence B has a non-negligible advantage in distinguishing the Pb from the random result. Nevertheless, this contradicts the fact that Pb is *a* (t, ϵ)-*PRF* and such A does not exist. Hence, $ELCD_E$ is secure against *IND-CPA*. □

Theorem 2. *For all PPT adversaries, the IND-CCA advantage when attacking* $ELCD_E$ *is negligible.*

- **Methodology of Proof:** The adversary guessing methodology is used to prove Theorem 2. Let us assume A is a *PPT* adversary algorithm that breaks $ELCD_E$ in the sense of *IND-CCA* for which $Adv^{in\text{-}cca}_{ELCD_E}(A) = 1$. To break $ELCD_E$, A gains access to an encryption oracle with any pair of equal-length messages (m_0, m_1) as input. The encryption oracle E_K (m_b) takes this input, and returns an encryption of either (m_0, m_1). The goal of A is to determine the value of *b. If A* guesses correctly, then $ELCD_E$ is not secure, otherwise $ELCD_E$ is secure against *IND-CCA.*

Proof. Let us assume that A queries E_K (m_b) with pair of messages (m_0, m_1) and the output of E_K (m_b) will be C_b. The challenge of A is to determine the value of $b \in \{0,1\}$. Therefore, A can solve this challenge using the following mechanism. Firstly, A flips the bits of C_b to get C_b^- and inputs the valid query C_b^- to the decryption oracle to obtain the message M. Finally, A can flip the bits of M at the same position that flipped in C_b to obtain M^-. As a result, if M^- is equal to either one of the queried messages (m_0, m_1), A guesses correctly and $ELCD_E$ is not secure. Otherwise, A guesses incorrectly and $ELCD_E$ is secure against *IND-CCA*. The procedure that A uses to break $ELCD_E$ can be described as follows:

$$\begin{array}{l} A(E(m_b),\ D\,(\cdot))\ \{ \\ m_0 \longleftarrow 0^{\mathrm{n}}; m_1 \longleftarrow 1^{\mathrm{n}}; C_b\ \longleftarrow\ E_K((m_0, m_1), b) \\ C_b^{-}\ \longleftarrow C \oplus 1^{\mathrm{n}};\ M \longleftarrow\ D_K\left(C_b^{-}\right);\ M \longleftarrow\ M \oplus 1^{\mathrm{n}}; \\ \mathit{If}\ M^{-} = m_0 \text{ then return 1 else return 0}\} \end{array}$$

Let us study the advantage of A in attacking $ELCD_E$ with *IND-CCA* in more detail as follows:

$$Adv_{ELCD_E}^{in\text{-}cca}(A) = P_r[\mathrm{Exp}_{ELCD_E}^{ind\text{-}ccp\text{-}1}(A)] - P_r[\mathrm{Exp}_{ELCD_E}^{ind\text{-}ccp\text{-}0}(A)] \tag{14}$$

We will study each part of Equation (14) individually. The first part is $\mathrm{Exp}_{ELCD_E}^{ind\text{-}ccp\text{-}1}(A)$ (b = 1) as: $C_1 = E_K(m_1, b) = Pb.X_1 \times m_1\ mod\ n$. If the *i*th bit of C_1 has been flipped, resulting in a new ciphertext C_1^- and the decryption oracle with C_1^- is queried as:

$$M = \left(C_1^-\right) = \times[(Pb.X_1 \times m_1\ mod\ n) \oplus 1^n]\ mod\ n. = \times[(Pb.X_1 \times 1^n\ mod\ n) \oplus 1^n]\ mod\ n. \tag{15}$$

The modular multiplication cannot be distributed over XOR (e.g., (5× (2⊕ 9))≠ (5× 2) ⊕ (5 × 9)), and it also cannot be associated over XOR (e.g., (5 × (2 ⊕ 9)) ≠ (5 × 2) ⊕ 9)). Hence, if A flips the *i*th bit of M to obtain M^-, they cannot guess correctly with (m_1) and the returned value is 0, which means $P_r[\mathrm{Exp}_{ELCD_E}^{ind\text{-}ccp\text{-}1}(A)]$ is 0. Similarly, the other part of Equation (14) can be proven as before in which $P_r[\mathrm{Exp}_{ELCD_E}^{ind\text{-}ccp\text{-}0}(A)$ is 0. Putting the two parts of Equation (14) together results in $Adv_{ELCD_E}^{in\text{-}cca}(A)$ = 0. Thus, the advantage of $\boldsymbol{A}$ in attacking $ELCD_E$ with *IND-CCA* is negligible. □

4.3.2. Proven Security for Proposed Digital Signature in ROM

The security advantage of the proposed digital signature consists of two levels: unforgeable digital signature integrated into lightweight encryption. The popular methods to implement combination between the digital signature and encryption can be described as follows:

- **Method 1**: Encrypt-and-Sign (EAS), which means that data should be encrypted using K_1 as $C = E_{k1}(M)$ and the digital signature should be calculated using K_2 as $D = DS_{k2}(M)$. The sending message is the pair (C, D) that should be sent separately.
- **Method 2:** Sign-then-encrypt (STE), which means that D is first calculated, and then the original data and D are concatenated and encrypted together. The sending message is $C = E_{k1}(M + D)$, where $D = DS_{k2}(M)$.
- **Method 3:** Encrypt-then-Sign (ETS), which means the original data M is first encrypted using K_1 as $C = E_{k1}(M)$, and then the D is calculated over C. The sending message is the pair (C, D), where $D = MAC_{k2}(C)$.

Method 1 and Method 3 are not secure because the adversary can eavesdrop on the communication channel between the sender and the receiver, capture all messages, strip off the sender's signature, sign the ciphertext with the adversary's own key, and send it to the receiver to gain access to IoT devices even though the adversary does not know the content of the messages. The following description shows that Method 2 is the more secure combination of digital signature and encryption.

Let us assume that the digital signature function is σ = *Sign(d, M)*, where $d \in \{0,1\}^n$ is the private key, M is the plaintext that should be signed, and $\sigma \in \{0,1\}^n$ is the output of the digital signature function. The verifying function is *VerfySign(Q, M, σ)*, which outputs 1 if the signature is valid or 0 if it is invalid. Let the symbol $Q \in E_{a,b}(\mathbb{Z}_n)$ denote the public key and let $H(M) :\ \{0,1\}^n \leftarrow D_Q$ (recall D_Q is the domain of $ELCD_{DS}(Q)$) be a hash function that is modeled as a random oracle function. Therefore, to generate a digital signature for M, $\sigma = Sign(d, H(M))$ is output. Correspondingly, to verify the digital signature σ on M, the $Sign^{-1}(Q,\sigma) \overset{?}{=} H(Ms)$ should be checked.

Theorem 3. *Let $ELCD_E$ = (K, E, D) be the encryption of ELCD that is secure under IND-CPA, and if $ELCD_{DS}$ = (K, D, V) is (t, qϵ)-secure (unforgeable against adaptive CPA). Then, ELCD = ($\overline{K}$, $\overline{E}$, $\overline{D}$, $\overline{V}$) created by the DS-then-encrypt is a secure combination between $ELCD_E$ and $ELCA_{DS}$. (Where t is the upper bound for the adversary's running time, q is the maximum number of queries to the random oracle H, and ϵ is the maximum probability that the adversary does the experiment).*

- **Methodology of Proof:** The proof of Theorem 3 is divided into two parts: the proof that $ELCD_{DS}$ is (t, $q\epsilon$)-secure against adaptive *CPA*, and the proof *ELCD* = ($\overline{K}$, $\overline{E}$, $\overline{D}$, $\overline{V}$) created using DS-then-encrypt is a secure combination between $ELCD_E$ and $ELCA_{DS}$. The contradiction methodology is used to prove the two parts of Theorem 3. The methodology of the first proof part of Theorem 3 can be described as follows:
    1. Assume an adversary $\boldsymbol{A}$ that runs in *PPT* exists, who has the ability to generate a forgery digital signature for the original message M with a probability δ.
    2. If the probability δ is proven not negligible, this means that A is doing well and $ELCD_{DS}$ is not secure.
    3. However, if the probability δ is proven negligible, this means A is not doing well and $ELCD_{DS}$ should be secure.

Proof of the First Part: Assume that A is used to construct an algorithm B that has the ability to reverse the trapdoor permutation. B receives Q and a random digital signature $\sigma = y$, and tries to obtain the digest of a signed message $x = H(M)$ such that $Sign^{-1}(Q,x) = y$. We assume that before A ever asks for a signature on message m, it has already queried $H(M)$. $B(Q, y)$ works as follows:

- B chooses a random index $i^* \in \{1, \ldots, q\}$.
- B can only have one query for the random oracle H.
- B receives the ith query from A to H and responses as follows:
    - ○ Let m_i represent the ith query from A to H: if $i = i^*$, this means $m_i = m$ and H will return y; otherwise, H chooses a random $r_i \leftarrow D_Q$.
    - ○ Calculate $Out_i = Sign^{-1}(Q, r_i)$, return Out_i.
- ▪ If A sends a signature request query on message m, B chooses i such that $m_i = m$; if $i = i^*$, abort; otherwise, return r_i as the signature.
- ▪ When A generates its forgery (m, σ), if $m = m_{i^*}$ then B outputs σ; otherwise, abort.

It is interesting to note that every response to signature queries from B is certainly a correct signature. B is able to respond to all signature queries except if A asks for a signature on m_{i^*}. Therefore, the forgery output (m, σ) is valid if A never queried a signature on m. In that case, m should be equaled to m_j for query j, and it must be the case that there is at least one index j for which A never requests a signature on m_j. Subsequently, because i^* is randomly chosen, this means the probability that $j = i^*$ is at least $1/q$. Hence, if A outputs a valid forgery ($j = i^*$), then σ is definitely the inverse of y, which means that B succeeds in reversing the *Sign* function. Nevertheless, this contradicts the fact that the *Sign* function is *a* (t, ϵ)-secure and our assumption must be wrong. With this we can conclude that the proposed digital signature should be secure, and no such A can exist. Since the probability of success to invert the *Sign* function is at least $1/q$ times the probability of generating a valid forgery by A (δ), we deduce that the probability of success for inverting the *Sign* function is at least δ/q. Nevertheless, the *Sign* function is assumed to be *a* (t, ϵ)-secure, the probability of successfully inverting it is $\delta/q \leq \epsilon$ or $\delta \leq q\epsilon$), which means it is negligible. This ends the proof of the first part of Theorem 3. □

- **Methodology of Second Part Proof:** The methodology of proof the second part of Theorem 3 can be described as follows:
    1. Assume the existence of adversary *A* that gains access to three random oracle functions: sign oracle function (SOF), encryption oracle function (EOF) and decryption oracle function (DOF). The three functions simulate $ELCA_{DS}$ and $ELCD_E$.
    2. First, *A* queries SOF with any pair of equal-length messages (m_0, m_1) as input. The output of SOF is a pair of bits (σ_0, σ_1).
    3. Second, *A* flips a single bit in either (σ_0, σ_1). Let us assume that the single bit of σ_0 has been flipped to be $\overline{\sigma_0}$.
    4. Third, *A* queries EOF with a pair of equal-length digital signatures ($\overline{\sigma_0}$, σ_1). The output of EOF is a pair of (C_0, C_1) = C_b.
    5. Fourth, *A* flips a single bit in C_b at the same position that was flipped in $\overline{\sigma_0}$ to obtain C_b^-. After that, *A* queries DOF with C_b^-.
    6. The goal of *A* is to determine the value of *b*. If *A* guesses correctly, then combination between $ELCD_E$ and $ELCA_{DS}$ in *ELCD* is not secure, otherwise it is secure against *IND-CCA*.

Proof of the Second Part. The procedure that *A* uses to break the combination between $ELCD_E$ and $ELCA_{DS}$ in *ELCD* can be described as follows:

```
$A$(SOF, EOF, DOF {
$m_0 \leftarrow 0^n$; $m_1 \leftarrow 1^n$; $\sigma_b \leftarrow DS_{K2}((m_0, m_1), b)$;
$\overline{\sigma}_b \leftarrow \sigma_b \oplus 1^n$; $C_b \leftarrow E_{K1}((\overline{\sigma}_0, \sigma_1), b)$
$C_b^- \leftarrow C_b \oplus 1^n; \sigma_b \leftarrow D_{K1}(C_b^-)$.
If $\sigma_b = m_0$ then return 1 else return 0}
```

Let us study the advantage of *A* in attacking $ELCD_E$ with *IND-CCA* in more detail as follows:

$$Adv_{ELCD}^{in\text{-}cca}(A) = P_r[\text{Exp}_{ELCD}^{ind\text{-}ccp\text{-}1}(A)] - P_r[\text{Exp}_{ELCD}^{ind\text{-}ccp\text{-}0}(A)] \tag{16}$$

We will study each part of Equation (16) individually. The first part is $\text{Exp}_{ELCD}^{ind\text{-}ccp\text{-}1}(A)$ (b = 1) as: $\sigma_1 = DS_{K2}(m_1, b) = (d_S{}^{-1} \times Y_K \times H(m_1))$ mod *n*. If the *i*th bit of σ_1 has been flipped, this result in a new digital signature $\overline{\sigma}_b$. Furthermore, if *A* queries EOF with a pair of equal-length digital signatures ($\overline{\sigma_0}$, σ_1). The output of EOF is a pair of (C_0, C_1) = C_b as follows:

$C_b = Pb.X_1 \times \sigma_b \ mod\ n$. *A* flips a single bit in C_b at the same position that was flipped in $\overline{\sigma_1}$ to obtain C_b^-. After that, *A* queries DOF with C_b^- as:

$$\sigma\text{b} = D_{K1}(C_b^-) = Pb.X_1{}^{-1} \times [(Pb.X_1 \times \sigma_b \ mod\ n) \oplus 1^n] \ mod\ n. = \times[(Pb.X_1 \times 1^n \ mod\ n) \oplus 1^n] \ mod\ n \tag{17}$$

Similar to Theorem 2, the modular multiplication cannot be distributed over XOR. Hence, if *A* flips the *i*th bit of σ_1 to obtain $\overline{\sigma}_b$, they cannot guess correctly with (σ_1) and the returned value is 0, which means $P_r[\text{Exp}_{ELCD}^{ind\text{-}ccp\text{-}1}(A)]$ is 0. Similarly, the other part of Equation (16) can be proven as before, in which case $P_r[\text{Exp}_{ELCD}^{ind\text{-}ccp\text{-}0}(A)$ is 0. Putting the two parts of Equation (16) together results in $Adv_{ELCD}^{in\text{-}cca}(A)$ = 0. Thus, the advantage of *A* in attacking the *ELCD* scheme with *IND-CCA* is negligible. □

Theorem 4. *The weak bits problem (weak bits are certain bits of information that can be correctly predicted with non-negligible advantage.) in the shared secret key due to the Diffie–Hellman exchange has been solved by ELCD.*

Proof. Let us assume that an adversary exposes the vulnerability of the communication channel to implement sniffing (eavesdropping) or a man-in-the-middle attack on an IoT network. Furthermore, let us assume that the parties of IoT devices select easier domain parameters. Thus, an adversary who uses brute force and sensor capture attacks can collect enough residue of the public keys (Q_S, Q_D), digital signatures DS, and ciphertexts C to derive the private keys (d_S, d_D) and shared secret keys (X_K, Y_K) of the IoT parties. This is a well-known problem, and it is called the weak bits problem. Let us assume that X_K has been calculated, which contains a weak bit problem, ELCD uses CMA (SHA-256 can be used), which combines three levels of hash function and random string (Salt) as:

$$H(H1, H2, Salt) = (H1(K1 \oplus X_K)||H2(K2 \oplus L)) \oplus Salt \tag{18}$$

where $H1(K_1, C_i) = (K_1 \oplus C_i) mod\ (2^{31} - 1)$; $H2(K_2, C_i)$ = (Geo_loc $\oplus K_2$) mod $(2^{31} - 1)$. Moreover, the variable Salt is a combination of random characters that are appended to the digest to make the dictionary and brute force attacks very much slower and limit the impact of a rainbow table attack. Three levels of hash function and random string guarantee the unpredictability of X_K and ensure strong, robust properties (preserve the collision resistance (CR), pseudo-randomness (PRF), message authentication code (MAC), and one-wayness (OW)). Hence, ELCD uses multihash functions (e.g., Hash(m) and Hash(X_K)) to remove the weak bits caused by the Diffie–Hellman exchange even if the communication protocol is vulnerable to sniffing attacks. □

4.4. Countermeasures against Replay and Man-in-the-Middle Attacks

The secure combination in ELCD has the ability to prevent man-in-the-middle and replay attacks from gaining access or replicating the digital signature. This is primarily due to the fact that the digital signature is protected by encryption (sign-then-encrypt), and the shared secret key Y_K that is used in the digital signature represents the true identity of the signer. Moreover, ELCD rejects the received message from the replay attacker due to the following reasons:

- The sender authentication based on Y_K in the digital signature should be checked before processing a message from a man-in-the-middle attacker.
- The digital signature is calculated based on the private key of the sender, which is protected by a hash function and encryption after a digital signature is applied to the plaintext.
- Replay attacks need to implement three steps before resending the intercepted message. These steps are shared secret key calculation, digital signature, and message encryption, which are very difficult to gain access to without breaching the hash function and the shared secret key.

4.5. Countermeasures against Brute Force Attacks

Since the shared secret key is ephemeral and must change every communication session, the ELCD resolves the weak bits problem and provides perfect forward secrecy. Furthermore, a brute force attacker needs to resolve the elliptic curve discrete logarithm problem (ECDLP) that requires $0.886 \times \sqrt{k}$ steps. This means that the security strength is 96, which is likely to be quite computationally intensive [34,40].

4.6. Countermeasures against Session Hijacking and Spoofing Attacks

The shared secret key in ELCD is encrypted using a secure hash function such as SHA-2 and CMA. This process leads to the generation of a random number (e.g., a digest of shared secret key after using the hash function), which can be utilized in the creation of session identity. Thus, if the attacker succeeds in breaking the session ID, they need to calculate the digital signature to gain access to the communication channel between the IoT parties. This is essentially due to the digital signature between the IoT sender and the receiver of the session required in the verification process. Furthermore, The ELCD mechanism can defend against key spoof attacks using the shared secret key calculation, which means it will not be sent through the channel between the parties of the IoT system. Therefore, the intruders have no chance to spoof the key.

4.7. Countermeasures against Device Capture and Stolen-Verifier Attacks

The ELCD cryptographic scheme can defend against IoT device capture and stolen-verifier attacks using built-in multifactor hash functions (e.g., CMA) that are built inside all IoT devices during the programming session. The multifactor hash functions that are used in ELCD are flashed and converted into low level source code language as explained in the assumption. Accordingly, a stolen key will not work without breaking the hash functions, which means that the intruder will not gain access to any of the captured IoT device's secure information.

5. Implementation and Performance Evaluation of ELCD on the IoT

The security software in the IoT platform should be evaluated based on resource constraints in terms of computational cost, storage usage, and power consumption. Consequently, ELCD uses the idea of ECDH for sharing the secret key, which is recommended by SECG/NIST, namely Secp192r1 [34]. The advantages of using the Secp192r1 standard elliptic curve in ELCD can be described as follows:

- The size of encryption and authentication keys is 24 bytes (192 bits) and the experimental processing latency that has been estimated for the ECDH to create and exchange the secret key is 0.576 s [28].
- The optimally recognized algorithm for resolving the k-size of ECDLP requires $0.886 * \sqrt{k}$ steps. Generally, a k-bit security strength can be achieved if the security system practices at least $2 \times k$-bit key size. Therefore, ELCD prefers to use the Secp192r1 curve, which can provide 96-bit security strength [34,40].
- The maximum message size of the IoT device is 127 bytes and it can be implemented based on the 6LowPAN protocol (40 bytes header), which is used to create a connection association between the IoT device and the sensor nodes [41].

The evaluation scenarios use the Mininet-IoT emulation software to implement and test the performance of ELCD because it has the ability to simulate the IoT hardware and communication description [42]. The experimental IoT network topology consists of one IoT gateway (BaseST1), eight static IoT sensors (sensor1 to sensor8), two intruders (Intrudr6 and Intrudr7), and one mobile IoT device (IoTDev5) as can be seen in Figure 5. The role of the intruders is mainly to implement the adversary model that has been discussed in the previous section. All IoT hardware boards contain pairs of network interface cards: communication with the IoT base station using IPv4 and IPv6 (i.e., 6LowPAN). Moreover, the proposed ELCD software is uploaded into all sensors, IoTDev5, and BaseST1. Moreover, the exchange of public keys and secure packets between all valid IoT devices is executed using client–server socket programming that combines with the ELCD's code. BaseST1 implements the server code while the client code is executed in all sensors and IoTDev5. Table 4 illustrates the experiment's parameters and configuration. In Mininet-IoT, 802.15.4_hwsim and 802.11_hwsim wireless models are used to perform the 6LowPAN protocol on the TCP/IP model. Moreover, the propagation model of the wireless signal is configured based on the shadowing model, which reflects the actual signal degradation due to impairments of the signal such as attenuation, noise, and interference. The mobility model of the mobile devices in the experiment is established using random movement on a grid network area of 1000 m $\times$ 900 m. All experimental running time has been set to 1000 s to study the impact of ELCD against the intruders when they implement dictionary and brute force attacks.

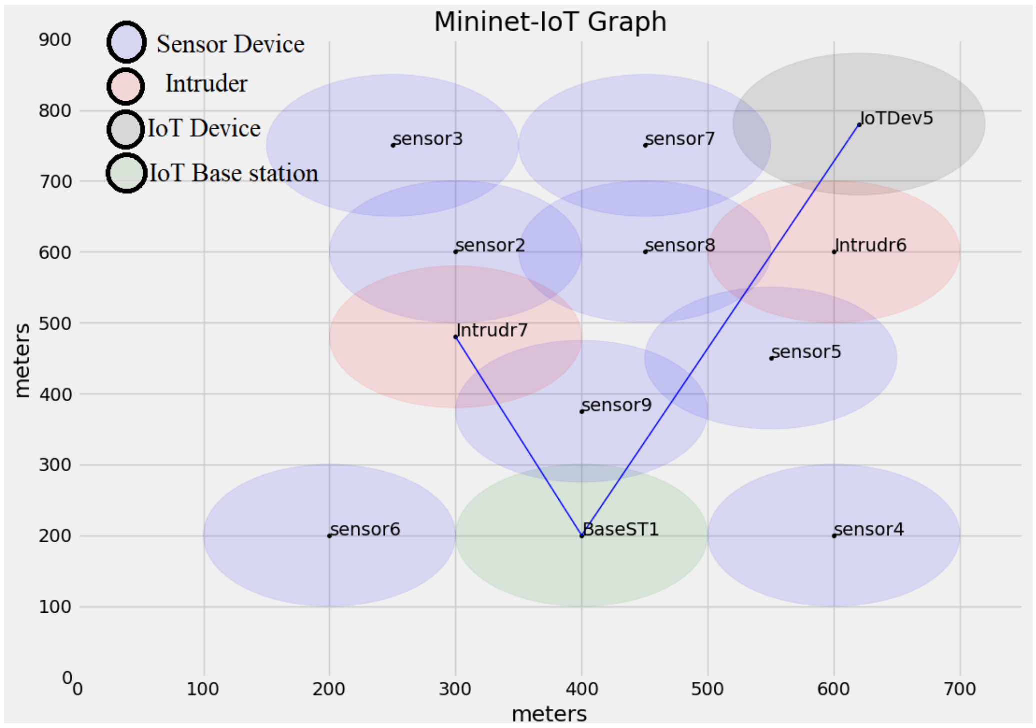


Figure 5. The IoT mesh topology.

Table 4. Experiment configuration.

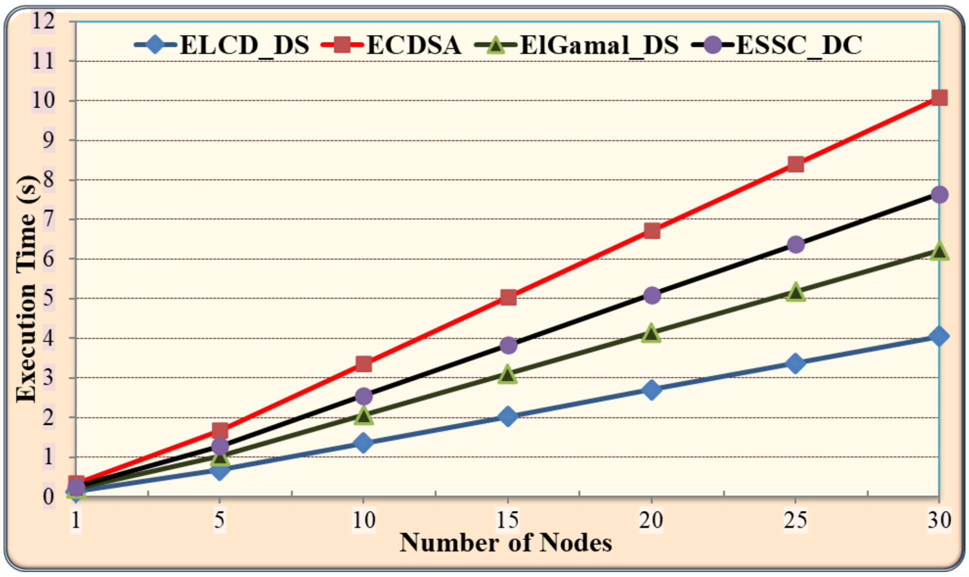
Parameter	Values
MAC and PHY	802.15.14_hmsim and 802.11_hmsim
Propagation model	Shadowing
Path loss exponent	3.0
Shadowing deviation (dB)	3.0
Event area	(1000 m × 900 m)
Cover of IoT device	150 m
Cover range of BaseST1	250 m
Traffic emulator	TCP Socket client/server; 1000 messages.
Performance metrics	CPU execution time, storage cost, and energy consumption
ECDH curve	Secp192r1
Message size	127 bytes
Key size	192 Bits
Emulation duration	1000 s

5.1. Performance Evaluation and Results Discussion

The performance evaluation of the proposed integration of encryption and authentication (e.g., ELCD) was analyzed in terms of the CPU execution time, memory usage, and power consumption cost. The comparison of performance analysis was investigated for the three methods of combination between authentication and encryption as presented in Figure 2. Furthermore, the performance of ELCD was compared with three benchmark security algorithms, namely ECIES_AES, ESSC_DC, and ECIES_Ra [RFC4503]). All of the source code is written in the Python programming language and implemented in the Mininet-IoT emulator. Moreover, the main source code of each of the baseline algorithms was downloaded from the security website [43]. Many scenarios were simulated, all subsequent testbeds were recured 10 times, and for each testbed 1000 packets were exchanged. Finally, the average results were calculated with the confidence interval reaching 95% based on a mean value and a standard deviation, as 5% of variation errors in the sample were accepted. Furthermore, the cProfile and memory_profiler program provided deterministic cost profiling of ELCD and the baseline mechanisms. The memory_profiler program can be used to measure the execution time of an algorithm, its storage cost, and its energy consumption. The total cost of the CPU's execution time can be estimated as the multiplication of the CPU execution time and the number of steps per execution (s/e). Moreover, the storage cost in each IoT device can be calculated as the total cost of communication (sent/received message) data, sensed information, and the cost of the source code in a time unit. Furthermore, the total energy consumption (mJ) in the IoT devices can be estimated as the total energy consumption for packet overhead that is used to execute the source code of the security algorithm [44].

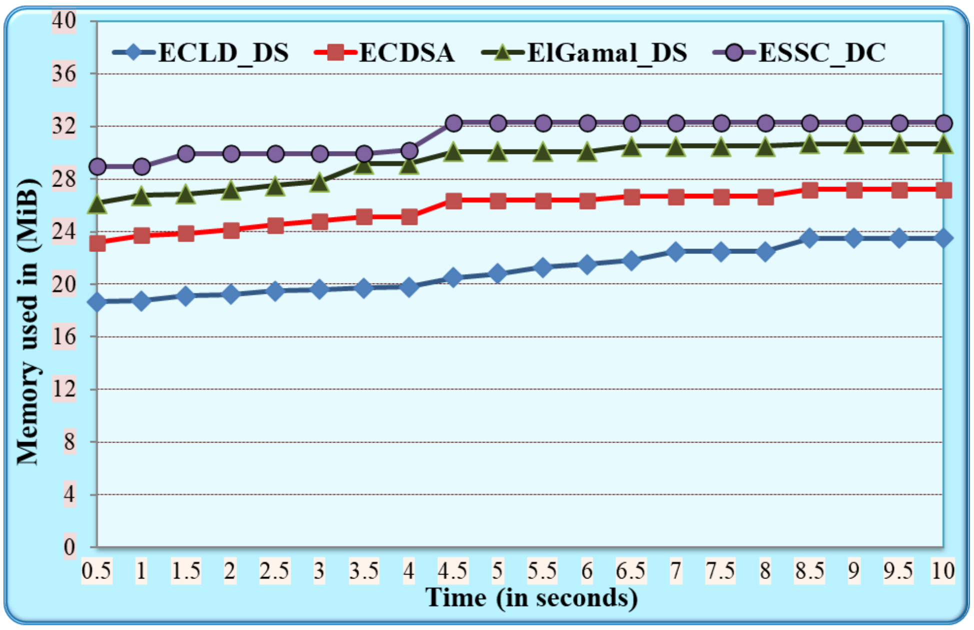
5.1.1. Performance Comparison between ELCD Digital Signature and Baseline Algorithms

The performance of using an ELCD digital signature (ELCD_DS) has been evaluated and compared with ECDSA, ESSC_DC, and ElGamal_DS. As can be shown in Figure 6a, the ELCD_DS experiences on average 88.9% less execution time compared to ESSC_DC, 53.8% less execution time compared to ECDSA, and it experiences on average 33.5% less execution time compared to ElGamal_DS. Moreover, Figure 6b illustrates that ELCD_DS experiences on average 37.02% less memory usage compared to ESSC_DC, 17.1% less memory usage compared to ECDSA, and it experiences on average 29.8% less memory usage compared to ElGamal_DS. Additionally, Figure 6c shows that ELCD_DS consumes on average twofold less energy compared to ESSC_DC, 68.7% less energy consumption compared to ECDSA, and it consumes on average 44.4% less energy compared to ElGamal_DS. The results presented in Figure 6 show the superiority of the ELCD_DS algorithm, which is mainly achieved due to the following reasons: Firstly, ELCD_DS uses a lightweight and secure calculation based on ECDH and a hash function to create a random digest based on the private key. In contrast, ESSC_DC uses a certificate authority to verify all digital certification processes, which requires extra resource in terms of energy, memory, and processing delay. Furthermore, ECDSA consumes more resources in terms of energy consumption, storage cost, and execution time due to the higher execution and communication overhead in the frequent use of scalar multiplication and inverse modular multiplication. Moreover, ElGamal_DS does not provide a certain solution; however, it provides four solutions, which is not suitable in IoT network. Finally, the lightweight hash (one-way direction) functions in ELCD_DS require less energy consumption, storage cost, and CPU time.

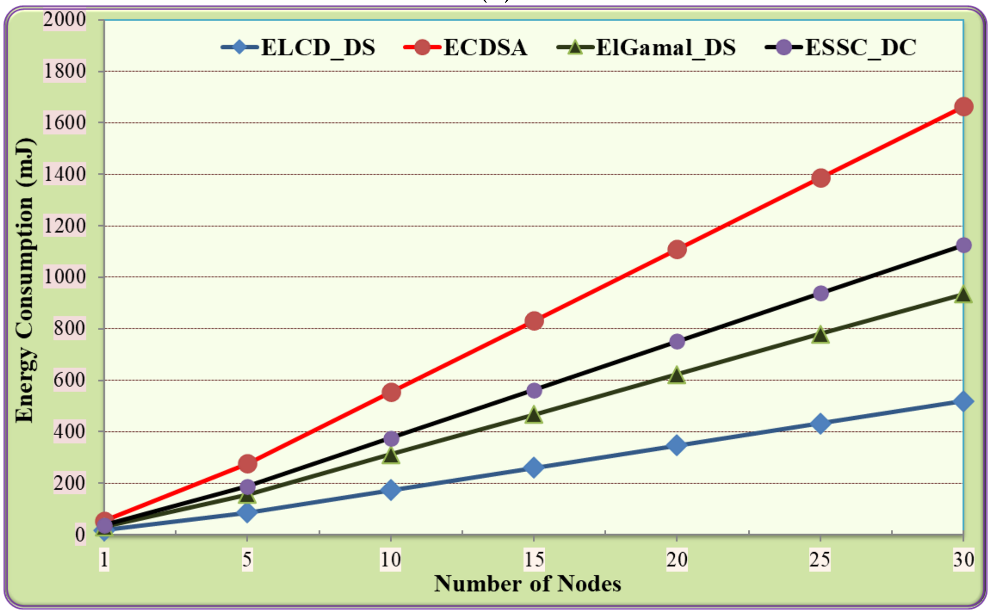


(**a**)

Figure 6. *Cont.*



(**b**)

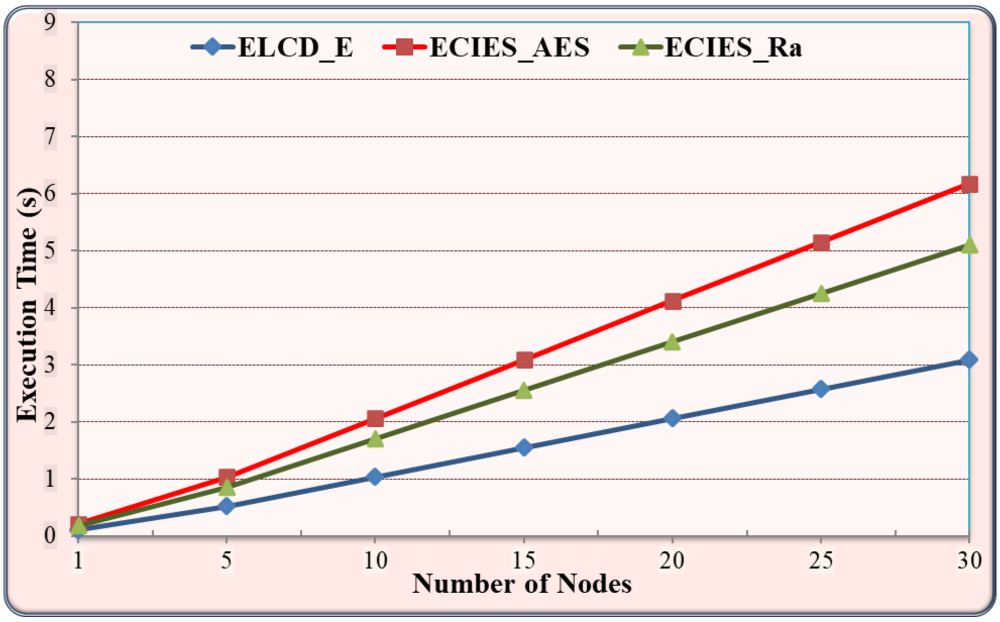


(**c**)

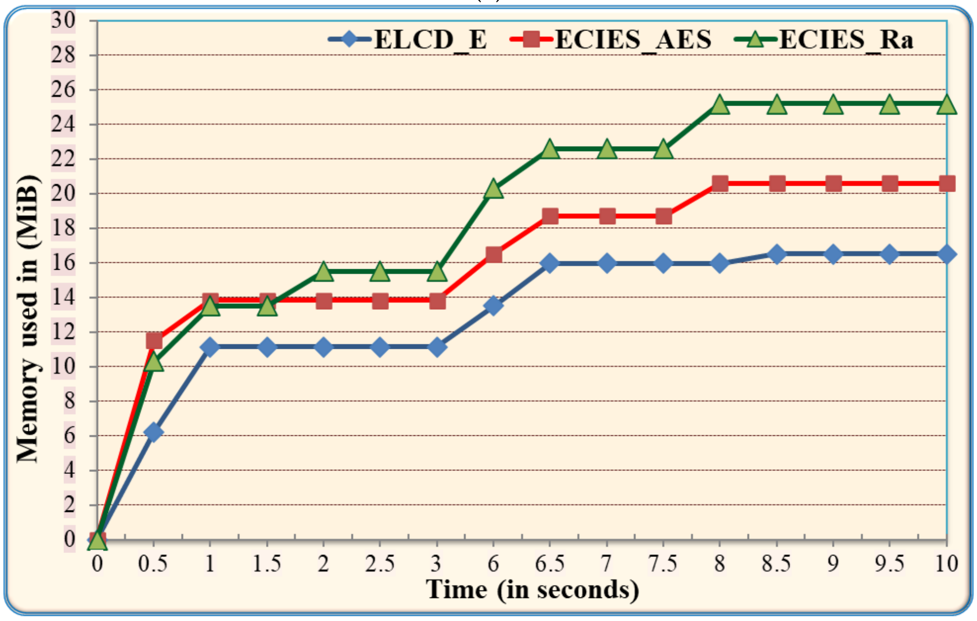
Figure 6. Performance comparison between ELCD digital signature and baseline algorithms on IoT (**a**) execution time; (**b**) storage cost; (**c**) energy consumption.

5.1.2. Performance Comparison between ELCD Cryptographic and Baseline Algorithms

The performance of ELCD encryption (ELCD_E) was evaluated and compared with ECIES_Ra and ECIES_AES. As shown in Figure 7a, the ELCD_E experiences on average 50% less execution time compared to EDIDS_AES and on average 39.4% less execution time compared to ECIES_Ra. Furthermore, Figure 7b depicts that the ELCD_E experiences on average 19.6% and 32% less memory usage compared to ECIES_AES and ECIES_Ra, respectively. Moreover, Figure 7c shows that ELCD_E consumes on average 41.2% less energy compared to the energy consumption in ECIES_AES, and it consumes on average 32.6% less energy compared to ECIES_Ra. The above results show that the ELCD_E outperforms ECIES_AES and ECIES_Ra in terms of CPU-time execution, storage cost, and energy consumption. This is primarily due to the following reasons: Firstly, ELCD_E consumes less energy and processing time in the encryption and decryption process, which is implemented based on an efficient mathematical random function. ELCD_E creates an ephemeral shared secret key for each session between IoT devices, which guarantees perfect forward secrecy of the encrypted message. Secondly, the ELCD_E consumes less storage cost due to the small number of functions called and the fewer execution steps per function. Finally, ECIES_AES and ECIES_Ra use complex and less effective encryption and decryption methods compared to ELCD_E. Overall, the findings of the experimental results show that the proposed integration of authentication and encryption in ELCD is effective, lightweight, and provides outstanding performance in terms of the CPU execution time, the storage cost, and energy consumption. More importantly, it resolves the problem of key distribution in symmetric key cryptography, and it resolves the problem of verifying the sender's identity in the digital signature.

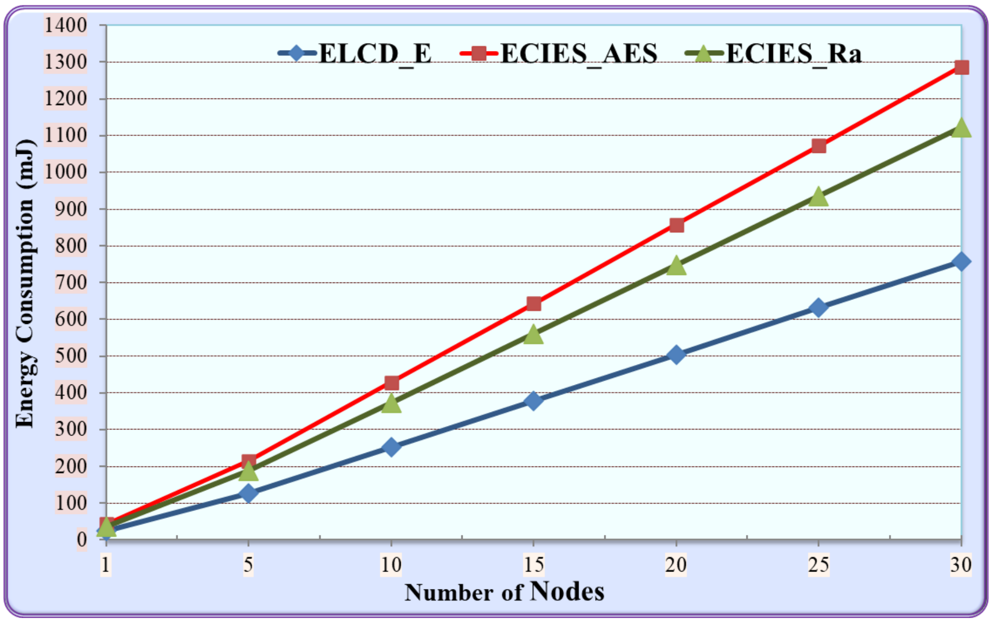


(**a**)



(**b**)

Figure 7. *Cont.*



(**c**)

Figure 7. Comparison between ELCD encryption (ELCD_E) and baseline cryptographic algorithms on IoT (**a**) execution cost; (**b**) storage cost; (**c**) energy consumption.

6. Conclusions and Future Work

The proposed ELCD algorithm was presented and compared with standard lightweight cryptographic and digital signature schemes. The ELCD mechanism utilized ECDH to develop a pair and a group of shared secret keys on an IoT network. The ELCD mechanism integrates a digital signature with secure encryption, which confirms the true identity of the sender with certainty and provides perfect forward secrecy. Furthermore, the security of the ELCD was proven mathematically and cyberattacks were investigated using the random oracle model. The performance of the ELCD outperforms the baseline digital signature in terms of CPU execution time, which is less by 53.8%; storage cost, which is less by 32–17%; and energy consumption, which is less by 68.7%. Future work regarding this research will focus on enhanced the performance a tiny digital certificate based on the ECDH in IoT networks.

Author Contributions: Conceptualization, O.M.B.; methodology, O.M.B.; software, A.A.A.; validation, A.A.A. and O.M.B.; formal analysis, A.A.A.; investigation, A.A.A. and O.M.B.; resources, A.A.A.; data curation, A.A.A.; writing—original draft preparation, A.A.A.; writing—review and editing, A.A.A.; visualization, A.A.A.; supervision, A.A.A.; project administration, A.A.A.; funding acquisition, A.A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University (KAU), Jeddah, Saudi Arabia, grant number G-039-830-1443 And The APC was funded by G:039-830-1443.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The Deanship of Scientific Research (DSR) at King Abdulaziz University (KAU), Jeddah, Saudi Arabia has funded this Project under grant no. (G:039-830-1443).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sarker, I.H.; Khan, A.I.; Abushark, Y.B.; Alsolami, F. Internet of Things (IoT) Security Intelligence: A Comprehensive Overview, Machine Learning Solutions and Research Directions. *Mob. Netw. Appl.* **2022**, 1–17. [CrossRef]
2. Sciancalepore, S.; Piro, G.; Vogli, E.; Boggia, G.; Grieco, L.; Cavone, G. LICITUS: A lightweight and standard compatible framework for securing layer-2 communications in the IoT. *Comput. Netw.* **2016**, *108*, 66–77. [CrossRef]
3. Kittur, A.S.; Pais, A.R. A trust model based batch verification of digital signatures in IoT. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *11*, 313–327. [CrossRef]
4. Li, S.; Zhang, T.; Yu, B.; He, K. A Provably Secure and Practical PUF-Based End-to-End Mutual Authentication and Key Exchange Protocol for IoT. *IEEE Sens. J.* **2021**, *21*, 5487–5501. [CrossRef]
5. Arne, B.; Le, N.; Dominik, S.; Stephan, S.; Lars, C.W. Security Properties of Gait for Mobile Device Pairing. *IEEE Trans. Mob. Comput.* **2019**, *19*, 697–710.
6. Diro, A.A.; Chilamkurti, N.; Kumar, N. Lightweight Cybersecurity Schemes Using Elliptic Curve Cryptography in Publish-Subscribe fog Computing. *Mob. Netw. Appl.* **2017**, *22*, 848–858. [CrossRef]
7. Khasawneh, S.; Kadoch, M. Hybrid Cryptography Algorithm with Precomputation for Advanced Metering Infrastructure Networks. *Mob. Netw. Appl.* **2018**, *23*, 982–993. [CrossRef]
8. Bu, L.; Isakov, M.; Kinsy, M.A. A secure and robust scheme for sharing confidential information in IoT systems. *Ad Hoc Netw.* **2019**, *92*, 101762. [CrossRef]
9. Hendaoui, F.; Eltaief, H.; Youssef, H. UAP: A unified authentication platform for IoT environment. *Comput. Netw.* **2021**, *188*, 107811. [CrossRef]
10. Vidya, R.; Prema, K.V. Lightweight hashing method for user authentication in Internet-of-Things. *Ad Hoc Netw.* **2019**, *89*, 97–106.
11. Chuang, Y.-H.; Lo, N.-W.; Yang, C.-Y.; Tang, S.-W. A Lightweight Continuous Authentication Protocol for the Internet of Things. *Sensors* **2018**, *18*, 1104. [CrossRef] [PubMed]
12. de Fuentes, J.M.; Gonzalez-Manzano, L.; Lopez, J.; Peris-Lopez, P.; Choo, K.-K.R. Editorial: Security and Privacy in Internet of Things. *Mob. Netw. Appl.* **2019**, *24*, 878–880. [CrossRef]
13. Riad, K.; Huang, T.; Ke, L. A dynamic and hierarchical access control for IoT in multi-authority cloud storage. *J. Netw. Comput. Appl.* **2020**, *160*, 102633. [CrossRef]
14. Alexander, J.M.; Kueffer, C.; Daehler, C.; Edwards, P.J.; Pauchard, A.; Seipel, T.; Arévalo, R.J.; Cavieres, L.A.; Dietz, H.; Jakobs, G.; et al. NETRA: Enhancing IoT Security Using NFV-Based Edge Traffic Analysis. *IEEE Sens. J.* **2019**, *19*, 4660–4671. [CrossRef]
15. Zhou, M.; Han, L.; Lu, H.; Fu, C. Intrusion Detection System for IoT Heterogeneous Perceptual Network. *Mob. Netw. Appl.* **2021**, *26*, 1461–1474. [CrossRef]
16. Alamer, A. An efficient group signcryption scheme supporting batch verification for securing transmitted data in the Internet of Things. *J. Ambient. Intell. Humaniz. Comput.* **2020**, 1–18. [CrossRef]
17. Ahmed, A.A. Lightweight Digital Certificate Management and Efficacious Symmetric Cryptographic Mechanism over Industrial Internet of Things. *Sensors* **2021**, *21*, 2810. [CrossRef]
18. Johnson, D.; Menezes, A.; Vanstone, S. The elliptic curve digital signature algorithm (ECDSA). *Int. J. Inf. Secur.* **2001**, *1*, 36–63. [CrossRef]
19. Mughal, M.A.; Luo, X.; Ullah, A.; Ullah, S.; Mahmood, Z. A lightweight digital signature based security scheme for human-centered Internet of Things. *IEEE Access* **2018**, *6*, 31630–31643. [CrossRef]
20. Park, C. A Secure and Efficient ECQV Implicit Certificate Issuance Protocol for the Internet of Things Applications. *IEEE Sens. J.* **2017**, *17*, 2215–2223. [CrossRef]
21. Mohseni-Ejiyeh, A.; Ashouri-Talouki, M.; Mahdavi, M. An Incentive-Aware Lightweight Secure Data Sharing Scheme for D2D Communication in 5G Cellular Networks. *ISeCure* **2018**, *10*, 15–27.
22. Abro, A.; Deng, Z.; Memon, K.A. A Lightweight Elliptic-Elgamal-Based Authentication Scheme for Secure Device-to-Device Communication. *Future Internet* **2019**, *11*, 108. [CrossRef]
23. Javed, Y.; Khan, A.S.; Qahar, A.; Abdullah, J. EEoP: A lightweight security scheme over PKI in D2D cellular networks. *J. Telecommun. Electron. Comput. Eng.* **2017**, *9*, 99–105.
24. Ahmed, A.A.; Ahmed, W.A. An Effective Multifactor Authentication Mechanism Based on Combiners of Hash Function over Internet of Things. *Sensors* **2019**, *19*, 3663. [CrossRef]
25. Sciancalepore, S.; Piro, G.; Boggia, G.; Bianchi, G. Public Key Authentication and Key Agreement in IoT Devices with Minimal Airtime Consumption. *IEEE Embed. Syst. Lett.* **2017**, *9*, 1–4. [CrossRef]
26. NIST Computer Security Resource Center. Lightweight Cryptography Project. Available online: https://csrc.nist.gov/projects/lightweight-cryptography (accessed on 27 November 2022).
27. Seok, B.; Sicato, J.C.S.; Erzhena, T.; Xuan, C.; Pan, Y.; Park, J.H. Secure D2D Communication for 5G IoT Network Based on Lightweight Cryptography. *Appl. Sci.* **2020**, *10*, 217. [CrossRef]
28. Khan, M.A.; Quasim, M.T.; Alghamdi, N.S.; Khan, M.Y. A Secure Framework for Authentication and Encryption Using Improved ECC for IoT-based Medical Sensor Data. *IEEE Access* **2020**, *8*, 52018–52027. [CrossRef]
29. Muhammad, U.; Ahmed, I.; Imran, M.A.; Shujaat, K.; Usman, A.S. SIT: A lightweight encryption algorithm for secure internet of things. *Int. J. Adv. Comput. Sci. Appl.* **2017**, *8*, 402–411.

30. Shah, R.H.; Salapurkar, D.P. A multifactor authentication system using secret splitting in the perspective of Cloud of Things. In Proceedings of the International Conference on Emerging Trends & Innovation in ICT (ICEI), Pune, India, 3–5 February 2017; pp. 1–4.
31. Hammi, B.; Fayad, A.; Khatoun, R.; Zeadally, S.; Begriche, Y. A Lightweight ECC-Based Authentication Scheme for Internet of Things (IoT). *IEEE Syst. J.* **2020**, *14*, 3440–3450. [CrossRef]
32. Rangwani, D.; Sadhukhan, D.; Ray, S.; Khan, M.K.; Dasgupta, M. A robust provable-secure privacy-preserving authentication protocol for Industrial Internet of Things. *Peer-to-Peer Netw. Appl.* **2021**, *14*, 1548–1571. [CrossRef]
33. NIST. *Fips Publication 180-2: Secure Hash Standard*; Technical Report; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2003.
34. Lochter, M.; Merkle, J. *RFC 5639: Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation*; IETF: Fremont, CA, USA, 2010.
35. Li, X.; Niu, J.W.; Ma, J.; Wang, W.D.; Liu, C.L. Cryptanalysis and improvement of a biometrics-based remote user authentication scheme using smart cards. *J. Netw. Comput. Appl.* **2011**, *34*, 73–79. [CrossRef]
36. Wang, J.; Han, K.; Alexandridis, A.; Zilic, Z.; Pang, Y.; Wu, W.; Jeon, G. A novel security scheme for Body Area Networks compatible with smart vehicles. *Comput. Netw.* **2018**, *143*, 74–81. [CrossRef]
37. Wang, Y.; Yang, G.; Li, T.; Li, F.; Tian, Y.; Yu, X. Belief and fairness: A secure two-party protocol toward the view of entropy for IoT devices. *J. Netw. Comput. Appl.* **2020**, *161*, 102641. [CrossRef]
38. Biryukov, A. Adaptive Chosen Plaintext Attack. In *Encyclopedia of Cryptography and Security*; Van Tilborg, H.C.A., Jajodia, S., Eds.; Springer: Boston, MA, USA, 2011.
39. Biryukov, A. Related Key Attack. In *Encyclopedia of Cryptography and Security*; Van Tilborg, H.C.A., Jajodia, S., Eds.; Springer: Boston, MA., USA, 2011.
40. Silverma, J.H. *An Introduction to the Theory of Elliptic Curves, Summer School on Computational Number Theory and Applications to Cryptography*; Brown University: Providence, RI, USA, 2006.
41. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. Available online: http://www.ietf.org/rfc/rfc4919.txt (accessed on 27 November 2022).
42. Mininet-IoT Emulator of Internet of Things. Available online: https://github.com/ramonfontes/mininet-iot (accessed on 27 November 2022).
43. A Security Site. Available online: https://asecuritysite.com/encryption (accessed on 27 November 2022).
44. Ahmed, A.A. An optimal complexity H. 264/AVC encoding for video streaming over next generation of wireless multimedia sensor networks. *Signal Image Video Process.* **2016**, *10*, 1143–1150. [CrossRef]