CAPITAL UNIVERSITY OF SCIENCE AND TECHNOLOGY, ISLAMABAD

Cost Effective Generalized Proxy Signcryption Scheme For Internet of Things (IoT)

by

Sana Jahangir

A thesis submitted in partial fulfillment for the degree of Master of Philosophy

in the

Faculty of Computing
Department of Mathematics

2024

To begin, I offer this research project as a dedication to the All-Merciful and Benevolent Creator, Allah Almighty, who sustains and shapes the Earth.
Furthermore, I dedicate it to my parents, whose prayers and unwavering support consistently lead me towards success.
Lastly, I extend this dedication to my teachers, who serve as a constant wellspring of inspiration and motivation in my academic journey.

CERTIFICATE OF APPROVAL

Cost Effective Generalized Proxy Signcryption Scheme For Internet of Things (IoT)

by

Sana Jahangir

(MMT213029)

THESIS EXAMINING COMMITTEE

S. No.	Examiner	Name	Organization
(a)	External Examiner	Dr. Nasir Siddiqui	UET, Taxila
(b)	Internal Examiner	Dr Dure Shehwar	Cust, Islamabad
(c)	Supervisor	Dr Rashid Ali	Cust, Islamabad

Dr Rashid Ali
Thesis Supervisor
May, 2024

Dr. Muhammad Sagheer
Head
Dept. of Mathematics
May, 2024

Dr. M. Abdul Qadir
Dean
Faculty of Computing
May, 2024

Author's Declaration

I, **Sana Jahangir** affirm that in my MPhil thesis titled "**Cost Effective Generalized Proxy Signcryption Scheme For Interet of Things (IoT)**" the work presented is original and has not been previously submitted by me for any degree, either at Capital University of Science and Technology, Islamabad, or any other educational institution, whether within the country or abroad.

I acknowledge that if, at any point in the future, it is discovered that my statement is inaccurate, even after the completion of my graduation, the University reserves the right to revoke my MPhil Degree.

(Sana Jahangir)

Registration No: MMT213029

Plagiarism Undertaking

I solemnly declare that research work presented in this thesis titled "**Cost Effective Generalized Proxy Signcryption Scheme for Internet of Things (IoT)**" is solely my research work with no significant contribution from any other person. Small contribution/help wherever taken has been duly acknowledged and that complete thesis has been written by me. I understand the zero tolerance policy of the HEC and Capital University of Science and Technology towards plagiarism. Therefore, I as an author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred/cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of MPhil Degree, the University reserves the right to withdraw/revoke my MPhil degree and that HEC and the University have the right to publish my name on the HEC/University website on which names of students are placed who submitted plagiarized work.

(Sana Jahangir)

Registration No: MMT213029

Acknowledgement

I am deeply grateful to Allah for blessing me with the intellect to undertake this journey. The guidance of Prophet Mohammad (peace be upon him) has been a beacon of light throughout. I extend my sincere appreciation to Dr. Rashid Ali for his unwavering support, which has been instrumental in the completion of my MPhil thesis. My heartfelt thanks also go to my parents, whose love and encouragement have been my constant motivation. I pray for continued blessings and prosperity for all those who have contributed to this endeavor.

(Sana Jahangir)

Registration No: MMT213029

Abstract

The concept of signcryption, a cryptographic technique, unites the functionalities of digital signatures and encryption into a single, cohesive operation. Signcryption offers the dual assurance of message confidentiality and authenticity, merging them into a single step. This work is an extension of the Insaf Ullah's signcryption scheme [23] to introduce a more comprehensive, generalized signcryption scheme. This extended scheme offers a range of operational modes, adapting to specific security requirements. It seamlessly operates in signcryption mode when both confidentiality and authenticity are necessary, as well as in signature-only and encryption-only modes when only one of these attributes is required. Notably, our proposed scheme includes the introduction of a secret key, an enhancement not present in the original scheme [23]. The proposed scheme boasts multiple security features, including non-repudiation, unforgeability, message confidentiality, forward secrecy, integrity, authentication, and non-repudiation. Furthermore, our thesis includes a detailed examination of correctness and cost, affirming the security and efficiency of the proposed scheme.

Contents

List of Figures

List of Tables

Abbreviations

AES	Advanced Encryption Standard
CEGPS	Cost Effective Generalized Proxy Signcryption
DES	Data Encryption Standard
DH	Diffie-Hellman Key Exchange Protocol
DLP	Discrete Logarithm Problem
DSA	Digital Signature Algorithm
ECC	Elliptic Curve Cryptography
GCD	Greatest Common Divisor
GSC	Generalized Signcryption
HCC	Hyper Elliptic Curve Cryptography
HCDHP	Hyper Elliptic Curve Diffie-Hellman Problem
HCDLP	Hyper Elliptic Curve Discrete Logarithm Problem
HEDHP	Hidden Extended Diffie-Hellman Problem
IoT	Internet of Things
PKC	Public Key Cryptography
RSA	Rivest-Shamir-Adleman

Symbols

a_d	Delegator's private key
a_s	Proxy signcrypter's private key
a_u	Proxy unsigncrypter's private key
D	Divisor of HC
f_d	Delegator's public key
f_s	Proxy signcrypter's public key
f_u	Proxy unsigncrypter's public key
H_0, H_1, H_2, H_3	hash functions
N_d	Nonce for delegator
N_s	Nonce for proxy signcrypter
PS	Proxy signcrypter
PU	Proxy unsigncrypter
Uo	Delegator
α, β	Private and public keys for signature generation
E_K	Symmetric key encryption algorithm with key K
D_K	Symmetric key decryption algorithm with key K
T	Time stamp

Chapter 1

Introduction

1.1 Background

The cryptography [2] is an art and science of encrypting the message in such a manner that its decryption by unauthorized third person becomes impossible. The use of mathematics protects the data from possible threats. The plaintext message is encrypted using some encryption algorithms and then the resulting ciphertext is sent through a public channel. Consequently, a decryption algorithm that can restore plaintext from ciphertext can be used by the intended recipient or an authorized person. For this purpose, each of the parties has their own key that they call a secret, which allows them to carry out all mentioned processes. The entire framework is known as a cryptosystem, and the security of cryptosystems relies on the confidentiality of the secret key. Cryptographic schemes can be broadly categorized into two main types: symmetric key cryptography and asymmetric key cryptography. In symmetric key encryption, a single secret key, known exclusively to the sender and recipient, is utilized. Two prominent examples of symmetric key encryption algorithms are DES [3] and AES [4]. The principal challenge in this approach lies in securely delivering the key between the sender and the recipient. This becomes particularly complex when dealing with a large number of participants. To address this issue, Diffie and Helman [5] introduced the concept

of asymmetric key cryptography, usually referred to as public key cryptography (PKC). In the context of PKC, users have two encryption keys: a private key that is kept secret and one that is public and accessible to everyone. Two well-known asymmetric key cryptography algorithms are RSA [6] and ElGamal [7].

In cryptography, there are several fundamental security properties that are essential for ensuring the confidentiality, integrity, and authenticity of data.

These properties include:

1. **Confidentiality:**
   Unless the private key is known by the sender or the designated recipient, an adaptive attacker should be unable to extract any confidential information from the encrypted text.

2. **Unforgeability:**
   Making sure it is computationally impossible for a skilled attacker to pose as a trustworthy sender and generate a precise encrypted message that the decryption algorithm can decipher.

3. **Non-repudiation:**
   The recipient must possess the means to demonstrate to a third party the origin of signcrypted text sent by the genuine sender. This ensures that previously encrypted messages will not be disavowed by the sender.

4. **Integrity:**
   Verification capability should exist for the recipient to confirm that the received message aligns with the one originally transmitted by the sender.

5. **Public Verifiability:**
   The validity of signcrypted text should be ascertainable by any third party

without necessitating access to the recipient's private key.

6. **Forward Secrecy:**

   If someone gets hold of the sender's secret key, they shouldn't be able to decode messages sent before. Without forward secrecy, if the secret key is taken, all old messages won't be safe anymore. This is especially risky when encryption is done on devices like cell phones, which aren't very secure. So, having forward confidentiality is important in such cases.

1.2 Literature Review

In 1997, Zheng [8] presented Signcryption, a unique cryptographic technology reduces computational cost compared to the sequential signature-then-encryption strategy by integrating digital signature and encryption into a single phase. When message secrecy and authenticity are required, signcryption works best; in other cases, a signature-then-encryption strategy works better. Three distinct cryptographic algorithms encryption, signature, and signcryption must be used when only one of these elements is necessary. This technique is known as Generalized Signcryption. It is often necessary to use encryption and signatures separately and simultaneously in IoT devices. Han [9] developed Generalized Signcryption, a notion that allows an algorithm to function in three different modes: signcryption, encryption, and signature. The mode used is determined by the unique security requirements of the IoT devices, offering a flexible and adaptable solution to satisfy their various security requirements. When the CEO of the company travels on regular basis for business, it is important to assign signing authority to reliable subordinates who serve as proxies and handle official communication processes on the CEO's behalf, as mentioned in reference [10].

This way, effective and timely communication is guaranteed. High-ranking executives often provide their assistants instructions on how to handle communication formalities in the context of the (IoT), using their services for legal verification and authentication.

A common component of this communication strategy is the transfer of signing authority. In order to solve this, Mambo et al [11] present the idea of a proxy signature in their work. Mambo's proxy signature scheme doesn't have important security features like public verifiability and forward secrecy. Because of this, it's not a good choice for discussions involving confidential company information. Hence, in order to fix this issue, Gamage et al. [12] introduces new scheme called proxy signcryption, where the execution of both proxy signature and encryption is seamlessly combined into a unified logical step. Nonetheless, adding the discrete logarithm problem to the proxy signcryption technique developed by Gamage et al. [12] results in increased processing and communication costs. Furthermore, the suggested method is devoid of important security elements like public verifiability and forward secrecy. Li and Chen [13] employed pairing-based techniques in their identity-based proxy signcryption (IDBPYS) scheme, which, as a prerequisite, demands a secure means of transmitting the user's secret key. Wang et al. [14] introduced an identity-based proxy signcryption system designed to meet security requirements, encompassing features such as forward secrecy and public verifiability. However, their suggested strategy runs into the crucial escrow issue. Duan et al. [15] presented a secure Identity-Based Proxy Signcryption (IDBPYS) system for safe delegation-by-warrant, demonstrating security under the random oracle model (ROM). This method relies on bilinear pairing for both security robustness and efficiency. It results in high computing costs and demands greater transmission bandwidth.

A novel publicly verifiable proxy signcryption strategy based on the discrete logarithm problem (DLP) was developed by Elkamshouchy et al.[16], alongside enhancements to existing techniques. However, the authors identified a vulnerability in their security mechanism, which relies on solving the computationally demanding DLP. Elkamchouchi et al. [17] came up with the idea of proxy signcryption, but efforts to improve security added more challenges, like the integer factorization problem (IF), DiffieHellman problem (DHP), and DSA problem. These make the method use up a lot of computer resources and need more data transfer.

To deal with this, Elkamchouchi and Abouslseoud [18] used a clever method using elliptic curves (EC) to give some rights to others. However, this also means needing more computer power and time, especially for solving IF, DHP, and DLP. Nevertheless, in partial delegation, the absence of constraints on the proxy signcrypter results in an inadvertent misuse of the signcrypting right in the proposed approach. Bilinear pairing was used in the creation of a novel proven secure proxy signcryption method by Lin et al.[19]. Unfortunately, the proposed approach by the authors does not meet the necessary security requirement for ensuring unforgeability. Elkamchouchi et al. [20] introduced the concept of warrants-based proxy signcryption as a potential enhancement, particularly suitable for resource-constrained devices. However, the effectiveness and security of this scheme hinge entirely on elliptic curve cryptography, leading to heightened power consumption in the device. A verifiable secured proxy signcryption on the standard model was proposed by Ming and Wang [21]. The suggested method may still be impacted by increased machine control usage and information transmission due to the computational burden of bilinear pairing (BP). In response to these challenges, Hussain et al. (2021) introduced a lightweight proxy signcryption method based on Hyperelliptic Curve Cryptography (HC). They assert that their newly developed scheme offers all security services at minimal computational and communication expenses. However, the utilization of larger operations over the hyperelliptic curve noticeably influences the overall effectiveness of the strategy.

1.3 Thesis Contribution:

Motivated by the considerations discussed above, we introduce a scheme referred to as Cost effective generalized proxy signcryption scheme ($CEGPS$) for internet of things (IOT), an extension of Insafullah's scheme [23] that provide generalized form. The primary aim is to address the limitations inherent in existing schemes, focusing on improvements in both security and efficiency. Our scheme leverages the utilization of Hyperelliptic Curve (HC), contributing to its lightweight nature by requiring smaller key sizes compared to traditional cryptographic approaches such

as RSA, ECC, and BP. This choice not only enhances computational efficiency but also minimizes communication overhead. In addition to adopting HC, our proposed (CEGPS) scheme addresses several overlooked aspects and introduces significant enhancements. Notably, we extend the scheme to incorporate a shared secret key, a feature absent in the original Insafullah's scheme [23].

Furthermore, the proposed ($CEGPS$) scheme rectifies certain oversights present in the earlier Insafullah's scheme [23].

One of the key features of the proposed scheme is the Syntax Definition, which offers a clear and concise outline of the components and their interactions in the ($CEGPS$) scheme. A detailed algorithm is presented, elucidating the step-by-step execution of the (CEGPS) scheme.
Security is a paramount concern, and our proposed scheme is designed to fulfill the following security requirements:

1. **Confidentiality:** Our proposed scheme (CEGPS) meets the security requirements of indistinguishable under adaptive chosen ciphertext attacks.

2. **Unforgeability:** The scheme demonstrates resistance against existential forgery under adaptive chosen-message attacks, establishing its reliability in preventing unauthorized signature generation.

3. **Forward Secrecy:** The proposed scheme incorporates forward secrecy, ensuring that the compromise of a current key does not jeopardize the security of previously exchanged messages.

To validate the effectiveness of the proposed scheme, comprehensive comparisons were conducted with existing schemes. The evaluation focused on computational cost and communication overhead.
Our proposed scheme introduces a generalized proxy feature. While this inclusion results in a slight increase in costs as compared

to Insafullah's scheme [23], it remains considerably lower than those associated with other existing generalized proxy signcryption schemes.

1.4 Thesis Structure

The thesis is structured as follows:

Chapter 1:

This chapter provides background of cryptography, Security properties and essential terms relevant to the thesis.

Chapter 2:

This chapter covers the fundamental concepts and materials related to basic algebra and hyper elliptic curve cryptography. Its primary objective is to facilitate the reader's comprehension of crucial terms.

Chapter 3:

This chapter covers the basic terminology and concepts associated with proxy signcryption and also detailed overview of Insafullah's schem. [1]

Chapter 4:

This chapter covers the proposed generalized signcryption scheme, which is based on hyper elliptic curve cryptography. The details and workings of this scheme are discussed thoroughly.

Chapter 5:

This chapter presents an in-depth security analysis of the proposed scheme (CEGPS) which includes formal security analysis, computational cost comparison and communication overhead.

Chapter 2

Preliminaries

This chapter introduces fundamental concepts, terminology, and cryptographic background relevant to Hyperelliptic curves, providing the necessary groundwork for a thorough understanding of the thesis. It presents formal definitions for the Hyperelliptic Curve Discrete Logarithm Problem ($HDLP$) and the Hyperelliptic Curve DiffieHellman Problem ($HDHP$). Furthermore, the chapter elucidates the associated threat model, offering insights into potential risks and vulnerabilities.

2.1 Mathematical Background

Definition 2.1.1. Group

" A group G, sometimes denoted by $(G, \cdot)$, is a set of elements with a binary operation, denoted by $\cdot$, that associates to each ordered pair (a, b) of elements in G an element $(a \cdot b)$ in G, such that the following axioms are obeyed:

1. **Closure:** If a and b belong to G, then $a \cdot b$ is also in G.
2. **Associative:** $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all a, b, c in G.
3. **Identity element:** There is an element e in G such that $a \cdot e = e \cdot a = a$ for all a in G.

4. **Inverse element:** For each a in G, there is an element a^{-1} in G such that $a \cdot a^{-1} = a^{-1} \cdot a = e$.

A group G is said to be abelian if it satisfies the following additional condition: $a \cdot b = b \cdot a$ for all a, b in G." [2]

Example 2.1.2.

Consider the set of rational numbers $\mathbb{Q}$ under addition. We aim to show that $\mathbb{Q}$ forms a group with respect to this operation.

1. **Closure:**

   Take any two rational numbers $a = \frac{p}{q}$ and $b = \frac{r}{s}$ from $\mathbb{Q}$. Their sum, $a + b = \frac{ps+rq}{qs}$, is also a rational number. This demonstrates closure under addition.

2. **Associativity:**

   For all $a, b, c \in \mathbb{Q}$, the equation $(a + b) + c = a + (b + c)$ holds, confirming the associativity of addition.

3. **Identity Element:**

   The additive identity in $\mathbb{Q}$ is 0. For any $a \in \mathbb{Q}$, we have $a + 0 = 0 + a = a$, verifying the existence of an identity element.

4. **Inverse Element:**

   Each $a \in \mathbb{Q}$ has an additive inverse, denoted as $-a$, such that $a + (-a) = (-a) + a = 0$.

5. **Commutativity (Abelian Property):**

   The addition of rational numbers is commutative; that is, for any $a, b \in \mathbb{Q}$, $a + b = b + a$.

Hence, $\mathbb{Q}$, equipped with addition, satisfies all the necessary group properties, establishing it as a group. Additionally, the commutative property renders it an abelian group.

Definition 2.1.3. Cyclic Group

"A group G is said to be cyclic if, for every element g in G, there exists an integer k such that $g = a^k$, where a is a fixed element in G. The element a is referred to as a generator of G "[2]

Example 2.1.4.

1. The set of integers $\mathbb{Z}$ operating under multiplication, does not form a cyclic group, because there is no single integer 'a' within $\mathbb{Z}$ for which all elements of $\mathbb{Z}$ can be expressed as powers of 'a' i.e., a^i for all integers 'i'

2. The set of integers Z operating under addition, form a cyclic group,because every element in Z can be generated by using the elements 1 and -1 through repeated addition. The elements 1 and -1 are considered generators of this cyclic group.

Definition 2.1.5. Field

"A field is a mathematical structure represented by a nonempty set $\mathbb{F}$, equipped with two binary operations, denoted as $+$ and $\cdot$. These operations satisfy the following properties: $\mathbb{F}$, if the following properties hold:

1. $\mathbb{F}$ is abelian group under addition.

2. forms an abelian group under multiplication (only nonzero elements).

3. Multiplication is distributed over addition in $\mathbb{F}$." [24]

Definition 2.1.6. Galois Field:

"A Galois field, also known as a finite field, is a field that contains a finite number of elements. The order of a finite field is always a prime or a power of a prime. For each prime power, there exists exactly one finite field, up to an isomorphism.The Galois field is denoted by GF(p^n) and has an order given by p^n." [25]

Example 2.1.7.

GF(5) consists of 5 elements, each of which is a polynomial of degree 0, and these elements are $\{0, 1, 2, 3, 4\}$.

Definition 2.1.8. Characteristics of a finite field:

The term characteristics of a finite field refers to a distinctive property exhibited by such fields. In particular, it pertains to the scenario where iteratively adding the multiplicative identity (1) to itself within the field eventually yields zero. Formally, if there exists a prime number p such that the sum $1 + 1 + 1 + \ldots + 1$ (repeated p times) equals zero, then p is denoted as the characteristic of the field, symbolized as $\mathbb{F}_q$. If q takes the form p^n, where p is a prime number and n is a positive integer, then n is defined as the dimension of the field.

Definition 2.1.9. Projective space:

Projective space is a mathematical framework used to study the transformation of geometric figures between different planes. In this context, we have two types of planes: the xy-plane, known as the affine plane, and the projective plane, which consists of points represented as (X, Y, Z), with the condition that $x = \frac{X}{Y}$. When Z equals zero, it signifies that a point is positioned at infinity.

This concept can be illustrated by considering a practical example, such as capturing a 3D scene through a camera and converting it into a 2D image. Imagine a scenario where you have parallel railway tracks in the 3D scene that never meet. However, when you project this scene onto a 2D image, it creates the illusion that the tracks converge at a single point, which we refer to as the point at infinity. This phenomenon demonstrates the principles of projective geometry and how perspective can alter our perception of spatial relationships.

2.1.1 Hyperelliptic Curve:

Definition 2.1.10. Hyperelliptic Curve

"Let K be a field, and let $\overline{K}$ be the algebraic closure of K. A hyperelliptic curve C of genus g over K ($g \geq 1$) in $K[u, v]$ is given by an equation of the form:

$$C : v^2 + h(u)v = f(u)$$

Where $h(u) \in K[u]$ is a polynomial of degree at most g, $f(u) \in K[u]$ is a monic polynomial of degree $2g + 1$, and there are no solutions $(u, v) \in K \times K$ that simultaneously satisfy the equation $v^2 + h(u)v = f(u)$ and the partial derivative equations $2v + h(u) = 0$ and $h'(u)v - f'(u) = 0$.

A singular point on C is a solution $(u, v) \in K \times K$ that simultaneously satisfies the equation $v^2 + h(u)v = f(u)$ and the partial derivative equations $2v + h(u) = 0$ and $h'(u)v - f'(u) = 0$. This definition says that a hyperelliptic curve does not have any singular points." [27]

Example 2.1.11.

Consider the curve C given by the equation

$$v^2 + uv = u^5 + 5u^4 + 6u^2 + u + 3$$

over the finite field $\mathbb{Z}_7$. This means we are working with numbers from 0 to 6. The functions are $h(u) = u$, $f(u) = u^5 + 5u^4 + 6u^2 + u + 3$, and $g = 2$. The curve C has no singular points except at infinity, making it a hyperelliptic curve.

The points on C that have coordinates in $\mathbb{Z}_7$ are:

$$C(\mathbb{Z}_7) = \{\infty, (1, 1), (1, 5), (2, 2), (2, 3), (5, 3), (5, 6), (6, 4)\}.$$

Definition 2.1.12. Genus Of curves

The genus of a curve is a topological and algebraic invariant that quantifies the number of handles or holes in the curve's surface, influencing the security and efficiency of cryptographic schemes based on the curve.

Here are examples of curves with varying levels of complexity, each characterized by a different genus:

An example of an elliptic curve with $g = 1$ is the well-known curve:

$$y^2 = x^3 - 3x + 4.$$

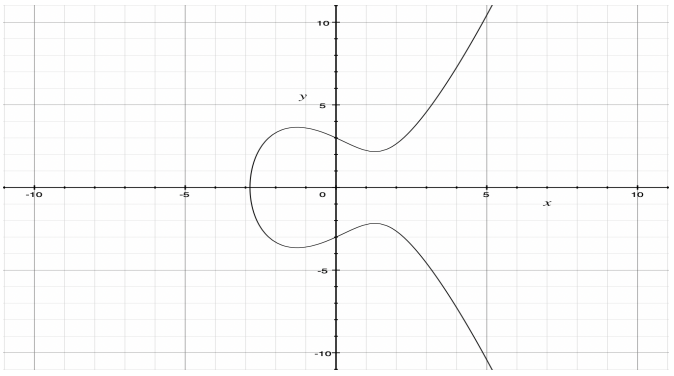

FIGURE 2.1: Graph of the hyperelliptic curve $y^2 = x^5 - 4x^3 + 3x$ with $g = 2$

A specific example of a genus-2 curve is a hyperelliptic curve defined as: $f(x) = x^5 - 2x^4 - 7x^3 + 8x^2 + 12x$

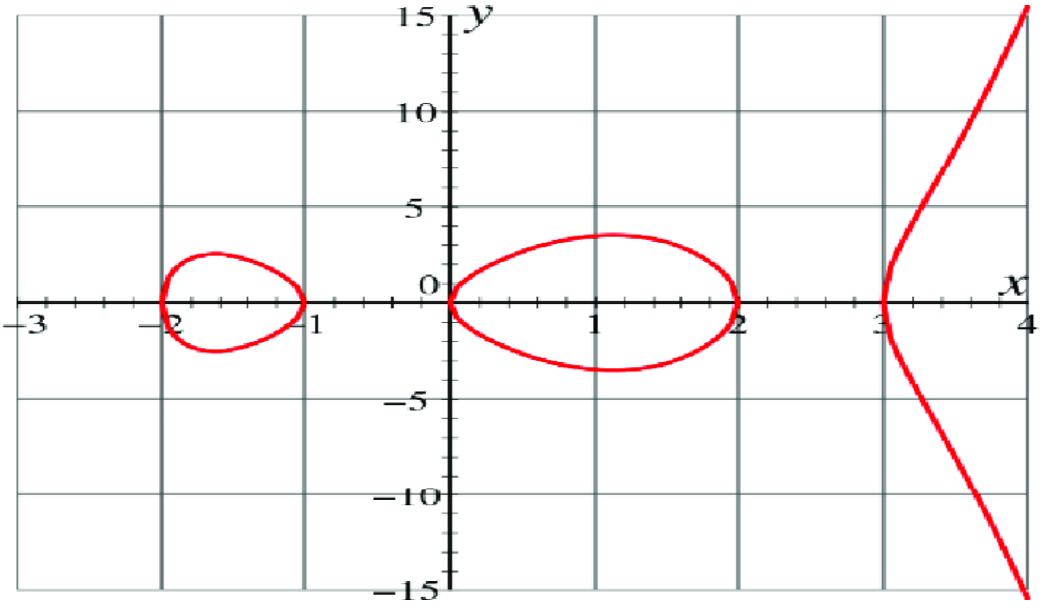

FIGURE 2.2: The graph of hyperelliptic curve $C : y^2 = f(x)$ where $f(x) = x^5 - 2x^4 - 7x^3 + 8x^2 + 12x$

Definition 2.1.13. (Point at Infinity, Special Point and Ordinary point)

If $p = (x, y)$ is a point on the HC, then its opposite point, denoted as $\overline{p} = (x, -y - h(x))$, also lies on the curve.

1. **Point at Infinity:** The point with projective coordinates $(0 : 1 : 0)$ is referred to as the point at infinity.

2. **Special Point:** A point p is considered a special point if it is equal to its opposite point $\overline{p}$, i.e., $p = p^{-}$. Special points may have unique properties or considerations.

3. **Ordinary Points:** Points on the HC that are not equal to their opposite points are classified as ordinary points. Ordinary points do not share the special properties associated with special points.

Example 2.1.14.

We have the hyperelliptic curve C over the finite field F_7 defined by the equation:

$$y^2 = x^5 - 10x^4 + 35x^3 - 50x^2 + 24x$$

To find the rational points, special points, and ordinary points on this curve, let's first calculate $h(x)$, where:

$$h(x) = x^5 - 10x^4 + 35x^3 - 50x^2 + 24x$$

$(1,0), (2,0), (3,0), (4,0), (5,1), (5,6)$ are the rational points on the curve C over F_7.
Now, let's check for special and ordinary points based on the equation $(x, y) = (x, -y - h(x))$:

$$\text{For } (0,0) : (0,0) = (0, -0 - h(0)) = (0,0) \text{ (Special Point)}$$
$$\text{For } (1,0) : (1,0) = (1, -0 - h(1)) = (1,0) \text{ (Special Point)}$$
$$\text{For } (2,0) : (2,0) = (2, -0 - h(2)) = (2,0) \text{ (Special Point)}$$

$$\text{For } (3,0) : (3,0) = (3, -0 - h(3)) = (3,0) \text{ (Special Point)}$$
$$\text{For } (3,0) : (3,0) = (3, -0 - h(3)) = (3,0) \text{ (Special Point)}$$
$$\text{For } (4,0) : (4,0) = (4, -0 - h(4)) = (4,0) \text{ (Special Point)}$$

$$\text{For } (5,1) : (5,1) = (5, -1 - h(5)) = (5,5) \text{ (Ordinary Point)}$$
$$\text{For } (5,6) : (5,1) = (5, -1 - h(5)) = (5,0) \text{ (Ordinary Point)}$$

Definition 2.1.15. (Coordinate Ring, Potential Functions, Rational Functions)

" The coordinate ring of a hyperelliptic curve C over a finite field $\mathbb{F}_q$, denoted as $\mathbb{F}_q[H]$, is defined as the quotient ring obtained by taking the polynomial ring $\mathbb{F}_q[x_1, y_1]$ and factoring out the ideal generated by the hyperelliptic curve's defining polynomial:

$$\mathbb{F}_q[C] = \frac{\mathbb{F}_q[x_1, y_1]}{\langle y_1^2 + h(x)y_1 - f(x_1)\rangle}$$

- $\mathbb{F}_q[x_1, y_1]$ is the polynomial ring in two variables over the finite field $\mathbb{F}_q$,
- $\langle y_1^2 + h(x_1)y_1 - f(x_1)\rangle$ represents the ideal generated by the hyperelliptic curve's defining polynomial $y_1^2 + h(x_1)y_1 - f(x_1)$.

Similarly, The coordinate ring of C over $\overline{\mathbb{F}}_q[H]$ is:

$$(\overline{\mathbb{F}}_q[C]) = \frac{\overline{\mathbb{F}_q}[x_1, y_1]}{\langle y_1^2 + h(x_1)y_1 - f(x_1)\rangle}$$

This notation, $\langle y_1^2 + h(x_1)y_1 + f(x_1)\rangle$, indicates the ideal generated by the given polynomial within the polynomial ring. The coordinate ring $\mathbb{F}_q[C]$ consists of equivalence classes of polynomials in $\mathbb{F}_q[x_1, y_1]$, where any two polynomials that differ by a multiple of $y_1^2 + h(x_1)y_1 + f(x_1)$ are considered equivalent. This construction allows us to study polynomial functions on the hyperelliptic curve C over the finite field $\mathbb{F}_q$.

An element of $\overline{\mathbb{F}}_q[C]$ is called a polynomial function on C. The function field $\overline{\mathbb{F}}_q(H)$ of C over $\overline{\mathbb{F}}_q$ is the field of fractions of $\overline{\mathbb{F}}_q[C]$. The elements of $\overline{\mathbb{F}}_q(C)$ are called rational functions on C." [30]

Definition 2.1.16. Divisors of Hypereliptic Curve

"Let C be a hyperelliptic curve of genus 2. A divisor D is a formal sum of points in C:

$$D = \sum_{P \in C} m_P P, \quad m_P \in \mathbb{Z},$$

where only a finite number of the m_P are non-zero. The degree of D, denoted by $\deg D$, is the integer

$$\deg D = \sum_{P \in C} m_P.$$

The order of D at P is the integer m_P; we write $\text{ord}_P(D) = m_P$." [27]

Geometric Interpretation:

Geometrically, for any point $P = (x_i, y_i)$ on the curve, the divisor of P is defined differently for ordinary and special points. On a hyperelliptic curve, the divisor of an ordinary point $P = (x, y)$ is represented by $\text{div}(P - \tilde{P} + 2\infty)$, where $\tilde{P}$ denotes the point $(x, -y - h(x))$ and 2∞ counts the point at infinity twice. This construction considers a line passing through $(x, 0)$, where y corresponds to the y-value of the point. For ordinary points, there are two possible values for y, leading to two potential intersection points with the curve. Thus, the divisor accounts for these two intersections, along with the point at infinity.

In contrast, the divisor of a special point is given by $\text{div}(2P - 2\infty)$. Special points are characterized by having only one possible value for y. Consequently, the divisor for special points accounts for this single intersection point with the curve and the point at infinity.

In simpler terms, the distinction between ordinary and special points lies in how their divisors are constructed. For ordinary points, the divisor considers the possibility of two y values, reflecting the potential for two intersections with the curve, while for special points, there is only one y value and hence only one intersection.

Definition 2.1.17. Divisor Group

The Divisor Group, denoted as $D = D(C)$, is a set of divisors forming an additive group under addition. Specifically, the sum is expressed as

$$\sum_{P \in C} m_P P + \sum_{P \in C} n_P P = \sum_{P \in C} (m_P + n_P) P.$$

Additionally, within the divisor group, there is a subgroup consisting of divisors of degree zero, denoted as D^0. This subgroup, D^0, is characterized by divisors whose

algebraic sum of coefficients is zero. Mathematically, it is expressed as

$$D^0 = \{D \in D(C) \mid \deg(D) = 0\}$$

Definition 2.1.18. (Principal Divisor, Jacobian)

" A divisor $D \in D^0$ is called a principal divisor if $D = \text{div}(R)$ for some rational function $R \in K(C)^*$. The set of all principal divisors, denoted P, is a subgroup of D^0. The quotient group

$$J = \frac{D^0}{P}$$

is called the Jacobian of the curve C. If $D_1, D_2 \in D^0$, then we write $D_1 \sim D_2$ if $D_1 - D_2 \in P$, D_1 and D_2 are said to be equivalent divisors." [27]

Definition 2.1.19. Support of a Divisor:

" Let $D = \sum_{P \in C} m_P P$ be a divisor. The support of D is the set

$$\text{supp}(D) = \{P \in C \mid m_P \neq 0\}$$

". [27]

Definition 2.1.20. Semi-Reduced Divisor

"A semi-reduced divisor is a divisor of the form

$$D = \sum_i m_i P_i - \left(\sum_i m_i\right) \infty,$$

where each $m_i \geq 0$, and the P_i's are finite points such that when $P_i \in \text{supp}(D)$, then $\overline{P_i} \notin \text{supp}(D)$, unless $P_i = P_j$, in which case $m_i = 1$". [27]

Definition 2.1.21. Reduced Divisor:

" Let $D = \sum_i m_i P_i - (\sum_i m_i)\,\infty$ be a semi-reduced divisor. If $m_i \leq g$ (where g is the genus of HC), then D is called a reduced divisor." [27]

2.2 Hyperelliptic Curve Cryptography

Koblitz [28] suggested that hyperelliptic curves, a particular type of mathematical curve, could be used to develop secure cryptographic systems. When we design cryptographic systems based on mathematical groups, their security often depends on how hard it is to solve a specific mathematical problem within that group. However, for these systems to be practically useful, operations like adding elements in the group need to be fast.

In Section 2.4, Cantor's algorithm is discussed to perform computations efficiently within the mathematical structures associated with hyperelliptic curves. These structures are important for cryptographic applications, and Cantor's algorithm helps speed up the necessary calculations.

The genus is a measure of the complexity of the curve. The higher the genus, the more vulnerable the system is to this specific type of attack. So, to balance security and efficiency, it is recommended to use hyperelliptic curves with a genus of 2 in cryptographic applications.

The following result demonstrates that the reduced divisors effectively represent all divisor classes of degree 0.

Proposition 2.2.1.

"Let D be a divisor of degree 0 on C. There exists a unique reduced divisor D_1 such that $D - D_1$ is a principal divisor." [30]

The following result is crucial for our study of divisors in hyperelliptic curves. It enables us to represent divisors in a clear and practical way, using pairs of polynomials. This representation allows us to work with divisors more concretely, making it easier to understand and apply in the context of hyperelliptic curves.

Theorem 2.2.2.

" There exists a one-to-one correspondence between semi-reduced divisors $D = \prod_j [P_j] - [\infty]$ and pairs of polynomials $(U(x), V(x))$ satisfying

1. $U(x)$ is monic,
2. $\deg U(x) = \prod_j \deg[P_j]$ and $\deg V(x) < \deg U(x)$,
3. $V(x)^2 - f(x)$ is a multiple of $U(x)$.

Under this correspondence, $D = \gcd(\text{div}(U(x)), \text{div}(y - V(x)))$." [30]

Theorem 2.2.3.

" Divisor classes of degree 0 on C can be uniquely represented by pairs of polynomials $(U(x), V(x))$ with the following properties:

1. $U(x)$ is a simple polynomial with a leading coefficient of 1.
2. The degree of $V(x)$ is less than the degree of $U(x)$, and both are at most g.
3. The polynomial $V(x)^2 - f(x)$ is a multiple of $U(x)$." [30]

Proof:

Theorem 2.3.1 tells us that each degree 0 divisor class is uniquely represented by a reduced divisor. According to Theorem 2.3.2, these divisors have a direct correspondence with pairs (U, V) as described in this theorem.

Remark 2.2.4.

In the following section, we outline an algorithm that generates the Mumford representation for the combination of two divisor classes having a degree of 0. This method offers a more practical approach compared to directly manipulating the divisor classes themselves.

When we start with a degree-0 divisor where $[\infty]$ is the only point with a negative coefficient, it is quite easy to find a semi-reduced divisor in the same class. We can do this by getting rid of certain polynomials in x. But, even though the proof of Proposition 13.6 doesn't directly tell us how to move from the semi-reduced divisor to the fully reduced one in the same class, using the pair of associated polynomials (U, V) gives us a simple way to do this. This process helps us to consistently find the pair that corresponds to the reduced divisor.

Theorem 2.2.5 (Reduction Procedure)**.**

" Let (U, V) be a pair representing a semi-reduced divisor D of degree 0. Perform the following steps:

1. Let $\widetilde{U} = (f - V^2)/U$.
2. Let $\widetilde{V} \equiv -V \pmod{\widetilde{U}}$ with $\deg(\widetilde{V}) < \deg \widetilde{U}$.
3. Let $U = \widetilde{U}$ and $V = \widetilde{V}$.
4. Multiply U by a constant to make U monic.
5. If $\deg(U) > g$, go back to step 1. Otherwise, continue.
6. Output (U, V).

The reduction procedure terminates, and the output is the pair representing the reduced divisor in the divisor class of D." [30]

2.2.1 Cantor's Group Operation

While the theoretical explanation of Jacobian points using divisor classes of degree 0 is insightful, it is not very practical for computational purposes. On the contrary, the Mumford representation provides a clear and concrete way to express points on the Jacobian. In this section, we introduce an algorithm by David Cantor [29] for combining divisor classes specified by their Mumford representations. This algorithm draws inspiration from Gauss's theory of composing quadratic forms, making it more applicable and manageable in computational contexts. The algorithm proposed by David Cantor [1] computes the group operation on a Jacobian using the Mumford representation. While a detailed proof is omitted, let's explore the intuition behind how the algorithm works.

Suppose we begin with Mumford pairs (U_1, V_1) and (U_2, V_2), corresponding to congruence classes of divisors in $Div^0(C)$ that contain reduced divisors D_1 and D_2; here, $D_i = \gcd(\mathrm{div}(U_i), \mathrm{div}(y - V_i))$. Cantor's Algorithm aims to combine

the polynomials U_i and V_i to obtain a new pair (U, V) representing a Mumford representation of a semi-reduced divisor within the same congruence class as $D_1 + D_2$.

To achieve this, the algorithm employs the Euclidean Algorithm to find the greatest common divisor d of U_1, U_2, and $V_1 + V_2$. It then determines polynomials h_1, h_2, and h_3 such that $d = h_1U_1 + h_2U_2 + h_3(V_1 + V_2)$. Subsequently, the algorithm computes $U = \frac{U_1U_2}{d^2}$, $V_0 = \frac{h_1U_1V_2+h_2U_2V_1+h_3(V_1V_2+f)}{d}$, and $V \equiv V_0 \pmod{U}$, where $\deg(V) < \deg(U)$. This newly obtained pair (U, V) corresponds to a semi-reduced divisor in the same congruence class as $D_1 + D_2$.

The second phase of the algorithm involves a reduction procedure. It adjusts U and V iteratively until (U, V) corresponds to the reduced divisor that is congruent to $D_1 + D_2$. If $\deg(U) > \deg(f)$, the algorithm modifies U to be the quotient of $\frac{f-V^2}{U}$, and computes a new value for V such that $\deg(V) < \deg(U)$. This recursive process continues until $\deg(U) \leq \deg(f)$.

Theorem 2.2.6 (Cantor's Algorithm)**.**
"Let D_1 and D_2 be divisors of degree 0, whose classes correspond to pairs (U_1, V_1) and (U_2, V_2), as in Theorem 2.3.3.

1. Let $d = \gcd(U_1, U_2, V_1 + V_2)$. Find polynomials h_1, h_2, h_3 such that $d = U_1h_1 + U_2h_2 + (V_1 + V_2)h_3$.

2. Let $V_0 = (U_1V_2h_1 + U_2V_1h_2 + (V_1V_2 + f)h_3)/d$.

3. Let $U = U_1U_2/d^2$ and $V \equiv V_0 \pmod{U}$ with $\deg V < \deg U$.

4. Let $\widetilde{U} = (f - V^2)/U$ and $\widetilde{V} \equiv -V \pmod{\widetilde{U}}$ with $\deg(\widetilde{V}) < \deg \widetilde{U}$.

5. Let $U = \widetilde{U}$ and $V = \widetilde{V}$.

6. Multiply U by a constant to make U monic.

7. If $\deg(U) > g$, go back to step 4. Otherwise, continue." [30]

Example 2.2.7.

Consider the divisors D_1 and D_2 defined by the pairs (U_1, V_1) and (U_2, V_2) for the curve $C : y^2 = x^5 - 1$ over $\mathbb{F}_3$. We'll follow the steps:

1. Compute d:

$$d = \gcd(U_1, U_2, V_1 + V_2) = \gcd(x^2 - x + 1, x - 1, -x + 1)$$

   Calculating this, we get $d = 1$.

2. Find h_1, h_2, h_3:

$$1 = (x^2 - x + 1)h_1 + (x - 1)h_2 + (-x + 1)h_3$$

   Let's say we find $h_1 = 1, h_2 = x, h_3 = 0$.

3. Compute U and V:

$$U = \frac{(x^2 - x + 1)(x - 1)}{1^2} = x^3 + x^2 - x - 1$$

$$V \equiv 0 + (x - 1)(-x + 1)(-x) + 0 \equiv x^3 + x^2 + x \equiv -x + 1 \pmod{U}$$

4. Compute $\widetilde{U}$ and $\widetilde{V}$:

$$\widetilde{U} = \frac{x^5 - 1 - V^2}{U} = x^2 - x - 1$$

$$\widetilde{V} \equiv -V \pmod{\widetilde{U}} = -(x - 1) = -x + 1$$

5. Update U and V:

$$U = \widetilde{U}, \quad V = \widetilde{V}$$

   So, $U = x^2 - x - 1$ and $V = -x + 1$.

6. Make U monic: Multiply U by a constant to make it monic. Here, it's already monic.

7. Check degree of U: If $\deg(U) > g$, go back to step 4; otherwise, continue.

In this case, $\deg(U) = 2$ which is not greater than $g = 5$, so we continue.

The final values are $U = x^2 - x - 1$ and $V = -x + 1$.

Definition 2.2.8. Hyperelliptic Curve Discrete Logarithm Problem (HCDLP):

The (HDL) problem entails the task of deducing a specific value δ from an instance ξ. This instance ξ is created by multiplying a given divisor D by δ, and D possesses a predefined order n.

Definition 2.2.9. Hyperelliptic Curve DiffieHellman Problem (HDHP)

Suppose a divisor D of order n and an instance $\xi = \gamma \cdot \epsilon$ are given. Extracting both ϵ and γ from ξ is referred to as the Hyperelliptic Curve DiffieHellman Problem $(HDHP)$.

2.3 Cryptographic Background:

Human nature exhibits two innate tendencies: firstly, the inclination to communicate and share information; and secondly, the inclination to communicate selectively. These intrinsic traits have given rise to the practice of encoding messages in such a way that only authorized individuals can access the information, preventing unauthorized access even if the encoded message falls into the wrong hands. This practice is known as cryptography. The history of cryptography can be divided into two eras: one predating public key cryptography and the other following its advent. Prior to the development of public key cryptography, symmetric key encryption was employed for both encrypting and decrypting data. Cryptography is the science and practice of securing information by transforming it into an unreadable format, often referred to as ciphertext, to protect it from unauthorized access or disclosure. This process involves use of mathematical algorithms and keys to encode data in such a way that it can only be deciphered by individuals who possess the corresponding decryption key or possess the necessary credentials.

2.3.1 Symmetric Key Cryptography

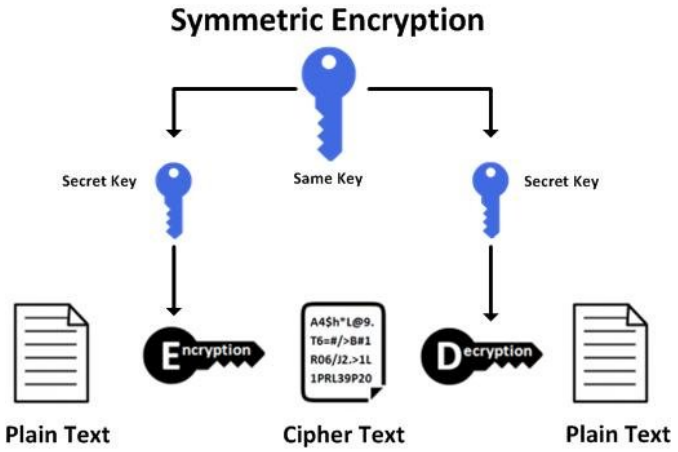

FIGURE 2.3: Symmetric Key Cryptography

Symmetric key encryption is an older cryptographic technique where both communicating parties employ a shared single key for both encrypting and decrypting their data. To enable secure communication, these parties must initially exchange the shared key through a trusted and secure channel. Symmetric key encryption relies on a high degree of trust between the communicating parties and is most effective when used for local communication. It allows for rapid data encryption and decryption but is less secure when used over networks. Examples of symmetric key encryption include block ciphers and stream ciphers.

2.3.2 Public Key Cryptography

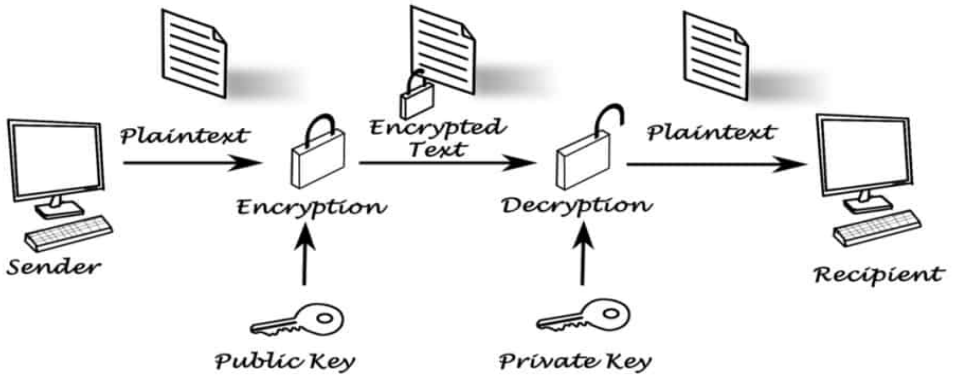

FIGURE 2.4: Public Key Cryptography

Public key cryptography is a cryptographic approach that employs two distinct keys: a public key, which is typically made openly available and used for encryption, and a private key, which is known exclusively to the intended recipient and employed for decryption or data signing. These two keys are mathematically linked, but deriving the private key from the public key is infeasible.

The inception of public key cryptography in 1976, credited to Diffie and Hellman, marked a significant advancement in the field. It gained prominence as the internet became more prevalent, eventually surpassing symmetric key cryptography in many applications. However, it is important to note that public key cryptography does have some drawbacks, notably its comparatively slower processing speed in contrast to symmetric key cryptography. Additionally, it relies on a trusted third party, known as a certification authority, which validates that a specific public key is associated with a particular individual or entity.

Examples of public key cryptography systems include RSA [31] and the Diffie-Hellman key exchange protocol [32].

2.3.3 Hash Functions

A hash function is a mathematical operation that transforms a variable-length string of characters (referred to as a message) into a fixed-length string of characters, known as a hash value or hash.

An ideal cryptographic hash function is expected to possess the following characteristics:

1. **Efficiency:**
   Calculating the hash value for any input should be a computationally efficient process.

2. **Preimage Resistance:**
   The process of retrieving the original message from a hash value should be

practically impossible without employing a brute-force search. Any slight modification to the input should lead to a significantly different hash value.

3. **Collision Resistance:**
   It should be computationally challenging to discover two different messages that yield an identical hash value.

A secure hash algorithm, commonly referred to as SHA, encompasses a series of hash algorithms, which includes SHA-1, SHA-2, SHA-3, and SHA-5. [2]

SHA, initially denoted as SHA-0, was the original cryptographic hash function. However, it was later succeeded by SHA-1. SHA-1, a cryptographic hash function developed by the United States National Security Agency, is a mathematical algorithm that transforms data of any length, up to 2^{64} bits, into a fixed 160-bit hash value, often referred to as a message digest.
It is classified as a one-way hash function, meaning that it is computationally infeasible to reverse the process and retrieve the original data. SHA-1 finds extensive applications in information secu-rity, particularly in Message Authentication Codes (MACs) and digital signatures.

SHA-2, the subsequent evolution of the SHA family, includes two notable members: SHA-256 and SHA-512. SHA-256 produces a 256-bit hash value, while SHA-512 generates a 512-bit hash value. These variants offer enhanced security through larger hash sizes.

SHA-3 and SHA-5, while supporting the same hash lengths as SHA-2, employ distinct internal structures and algorithms.
These differences contribute to their resilience against various cryptographic attacks.

For more comprehensive information on these topics, readers are encouraged to consult references such as “Applied Cryptography” by Bruce Schneier [33] and “Cryptography and Network Security” by William Stallings [34].

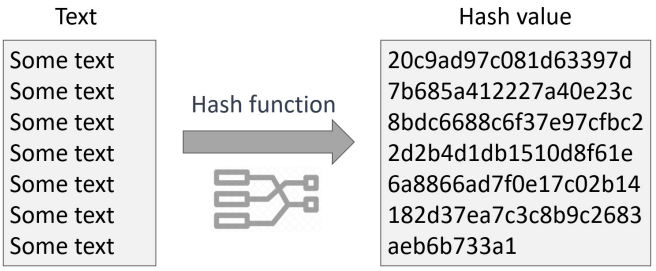

FIGURE 2.5: Hash Function

2.3.3.1 Message Authentication Code (MAC)

A cryptographic hash function is employed to affix an authentication tag to a message, serving as a means to verify data integrity and safeguard against inadvertent or malicious alterations. To accomplish this, session keys are employed. The key attributes of Message Authentication Codes (MAC) are summarized as follows:

1. It produces a cryptographically secure checksum for a given dataset, thereby guaranteeing the data's integrity.
2. MAC schemes rely on symmetric keys, necessitating that both the sender and verifier possess a shared secret key.
3. These schemes can process messages of variable lengths without issue.
4. MAC generates authentication tags of fixed lengths.
5. Any tampering or manipulation of the data will be readily detected by the recipient.

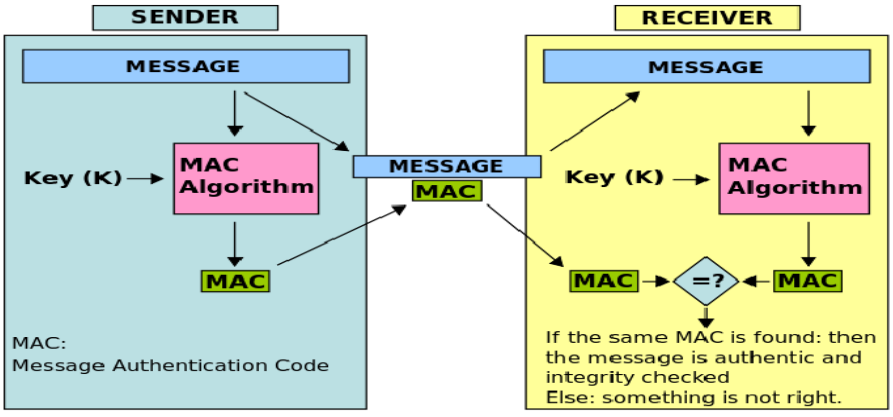

FIGURE 2.6: Message Authentication Code (MAC)

2.4 Signcryption

Public key cryptography, a breakthrough from nearly two decades ago Diffie and Hellman [32], has transformed the landscape of secure and authenticated communication. It enables individuals who have never met to securely exchange information over open and insecure networks like the Internet. The process involves a two-step approach: before sending a message, the sender signs it using a digital signature scheme. Following that, the message (and signature) is encrypted using a private key encryption algorithm, employing a randomly chosen message encryption key. The encrypted message encryption key is then further secured by encrypting it with the recipient's public key. This approach is known as signature-then-encryption.

However, both signature generation and encryption consume computational resources and introduce additional bits to the original message. Therefore, the cost of a cryptographic operation is typically measured in terms of the message expansion rate and the computational time required by both the sender and the recipient. The current standard, signature-then-encryption, incurs a cost equal to the sum of the digital signature and encryption costs. Over the past two decades since the inception of public key cryptography, the standard method for securely delivering messages of any length has been signature-then-encryption. During this time, it has been widely accepted without questioning whether it's absolutely necessary to combine the costs of both signature and encryption to ensure both content confidentiality and origin authenticity.

In addressing this fundamental question, which holds significance from both theoretical and practical perspectives, we aim to provide an answer. In 1996, Zheng [8] introduced a cryptographic innovation known as signcryption. Signcryption combines the functionalities of digital signature and encryption into a single logical step, significantly reducing computational costs and communication overhead. This can be succinctly expressed as:

Cost of Signcryption $<$ Cost of Signature + Cost of Encryption.

2.5 Features of Signcryption Scheme

A Signcryption Scheme for Hyperelliptic Curve, like any signcryption system, typically consists of three key algorithms: Key Generation (`Gen`), Signcryption (`SC`), and Unsigncryption (`USC`). Here are the essential features of such a scheme:

1. **Correctness:**
   The scheme ensures that the signcryption process is accurately performed, and the intended recipients can correctly retrieve the original message.
2. **Efficiency:**
   The computational cost and communication overhead in the signcryption scheme are designed to be lower when compared to separate signature-then-encryption schemes with similar functionalities.

2.5.1 Security of Signcryption:

The security of signcryption revolves around two main concerns, what needs protection and against whom. In the first aspect, we aim to keep the contents of a signcrypted message confidential, ensuring that only the sender (Alice) and the intended recipient (Bob) can access it. Simultaneously, we want to prevent others from impersonating Alice and sending messages on her behalf.

Considering potential attackers, we look at adaptive attackers who have access to both Alice's signcryption algorithm and Bob's unsigncryption algorithm, representing some of the most powerful adversaries in practical scenarios.

For a signcryption scheme to be deemed secure, it must meet the following conditions:

1. **Unforgeability:**
   It should be computationally challenging for an adaptive attacker, possibly a dishonest recipient (Bob), to impersonate Alice and create a forged signcrypted message.

2. **Non-repudiation:**

   In case of a dispute between Alice and Bob, a third party should be able to computationally resolve it. This is important when Alice denies being the originator of a signcrypted message sent to Bob.

3. **Confidentiality:**

   It should be computationally difficult for an adaptive attacker (any party other than Alice and Bob) to obtain partial information about the contents of a signcrypted message. This ensures the privacy of the communication.

Additionally, some signcryption schemes may offer:

- **Public Authentication:**

  Verifying the authenticity of the involved parties through a public process.

- **Forward Secrecy of Message Confidentiality:**

  Ensuring that even if a party's secret key is compromised in the future, previous communications remain secure.

The security of these features is typically dependent on solving underlying hard problems, such as the elliptic curve discrete logarithm problem in schemes based on elliptic curve cryptography (ECC) and hyper elliptic curve discrete logarithm problem in schemes based on hyper elliptic curve cryptography(HCC).

For more comprehensive information on signcryption, please refer to [35]

2.6 Generalized Signcryption:

The aim of achieving secure message delivery doesn't always demand both confidentiality and authenticity for every message. Some messages may only need a signature, while others may only require encryption. In 2006, Han and Yang [9] introduced a scheme offering flexibility in providing either signcryption, encryption, or signature functionalities, based on specific needs. This versatile scheme is known as generalized signcryption.

When there is a simultaneous need for both confidentiality and authenticity, the generalized signcryption scheme seamlessly performs dual functions. Conversely, when only confidentiality or authenticity is required, it smoothly executes a single encryption or signature function without the need for modifications or additional computations. In certain situations, the generalized signcryption scheme may function equivalently to a signature or encryption scheme.

There are three possible scenarios: (i) signcryption, (ii) signature-only, and (iii) encryption-only. Distinguishing between these situations poses a significant challenge. Conducting the authentication process in a public key environment requires knowledge of the sender's public and private keys. Meanwhile, the encryption process requires knowledge of a specific recipient's public and private keys.

2.7 Internet of Things (IoT)

The concept of the "Internet of Things" (IoT) or "Internet of Objects" encompasses a diverse range of electrical and electronic devices that are interconnected through the Internet. As our world becomes increasingly saturated with connected devices, the importance of addressing security and privacy concerns becomes more pronounced.

The majority of IoT gadgets are comprised of constrained devices, also known as sensors, smart objects, or smart devices. These devices typically have limited CPU, memory, and battery capacities, rendering them incapable of handling additional features and protocols.

Furthermore, their resource-constrained nature makes it challenging for them to perform cryptographic computations, posing a significant hurdle in ensuring the security of IoT systems.

Recognizing this challenge, there is a need of a cryptographic system that is not only lightweight and efficient but also secure enough to meet the demands of IoT

devices with constrained resources. In this study, we propose the implementation of a lightweight proxy system, allowing individuals or parties to delegate the authority to sign on their behalf.

Existing security methods for proxy signcryption have drawbacks, particularly in terms of computational expense and reliance on complex cryptographic techniques such as elliptic curve cryptography (ECC), RSA, and bilinear pairing. To address these concerns, we advocate for the use of the Hyperelliptic Curve Cryptosystem (HECC). This cryptographic approach offers a reduced key size without compromising on security, making it well-suited for the limitations of IoT devices.

In summary, our proposed lightweight proxy system, utilizing the Hyperelliptic Curve Cryptosystem, presents a viable solution to the security challenges posed by resource-constrained IoT devices. This approach ensures that cryptographic operations remain efficient while maintaining the necessary level of security for the diverse array of devices comprising the Internet of Things.

Chapter 3

Proxy Signcryption

The primary goal of this chapter is to provide a detailed explanation of proxy signcryption. we will delve into the discussion of a cost-effective proxy signcryption scheme presented by Insafullah et al. [1].

In IoT scenarios, devices often have limited computational power, memory, and energy resources. This makes it challenging to directly implement complex cryptographic operations like signcryption on these devices. Proxy signcryption addresses this challenge by introducing an intermediary entity, called a proxy, which assists in performing signcryption operations on behalf of IoT devices. Proxy signcryption is a cryptographic primitive that combines the functionalities of proxy signatures and encryption to achieve privacy, authenticity, and efficiency in secure communication. It allows a proxy entity to perform signcryption operations on behalf of a sender, enabling the proxy to encrypt a message using the recipient's public key and include a signature using the sender's private key. This integrated operation provides both confidentiality and authentication in a single step. By utilizing proxy signcryption, the sender can delegate the resource-intensive signcryption operations to a trusted proxy, relieving the sender's computational burden, especially in resource-constrained environments such as the Internet of Things (IoT). The proxy entity plays a crucial role in securely handling signcryption tasks, ensuring the privacy and authenticity of the communication while minimizing computational and communication costs.

Proxy signcryption has applications in various secure communication scenarios, including spanning e-business, mobile agent interactions, online voting processes, contract signing engagements, and online auctions.It offers a practical and efficient solution for achieving privacy and authenticity in these contexts.

3.1 Insafullah's Proxy Signcryption Scheme

In this section, we will review the paper titled Cost-Effective Proxy Signcryption Scheme for Internet of Things by Insafullah et al. [23].Before delving into the scheme itself, we will first introduce setup and some notations that are crucial for understanding the proposed scheme. These notations will provide a foundation for comprehending the technical details and concepts presented in the paper.

3.1.1 Setup

The Certificate Authority (CA) selects a Hyper Elliptic Curve (HC) with an 80-bit parameter size in this subsection. The CA then establishes a system parameter set as $\{HC, D, H_0, H_1, z_i\}$, where z_i is generated by randomly selecting y_i, and subsequently calculates $z_i = y_i \cdot D$. Finally, the CA ensures that z_i is made accessible publicly within a network.

3.1.2 Notations

Table 3.1: Notations.

Symbols	Descriptions
D	Divisor of HC
U_0, P_1, P_2	Addresses the job of delegator,proxy signcrypter, and unsigncrypter
y_d, y_p, y_u	Pivate key's of the participants U_0, P_1 and P_2
z_d, z_p, z_u	Public key's of participants U_0, P_1 and P_2
N_d, N_p	Nonce for participants U_0, P_1
H_0, H_1, H_2	Hash functions
x_1, x_2	Secret and public key for signature generation

3.1.3 Key Generation

In key generation phase, users initially generate their public and private keys as follows. Each participant selects an arbitrary $a_i \in \{1, 2, \ldots, q-1\}$ and computes $f_i = a_i \cdot D$. Therefore, a_i and f_i correspond to the private and public keys of the participants, respectively.

3.1.4 Proxy Delegation

In the proxy delegation phase, the proxy delegator, denoted as U_0, confers the signing authority to the proxy signcrypter P_1.

1. The proxy delegator chooses an arbitrary number $\zeta = \{1, 2, ...q-1\}$
2. Calculate $\vartheta = \zeta.D$
3. Calculate $\tau = H_0(W_m, \vartheta)$
4. Calculate $V = (\zeta\text{-}a_d.\tau) mod q$

Proxy delegator sends $(\tau, V, W_m)$to proxy signcrypter for validation.

$$\begin{aligned}\vartheta &= V.D + H_1(W_m, N_d, \zeta).f_d \\ &= VD + H_1(W_m, N_d, \tau).f_d \\ &= (\tau - a_d.H_1(W_m, N_d, \tau)).D + H_1(W_m, N_d, T).a_d.D \\ &= D.(\tau - a_d.H_1(W_m, N_d, \tau) + a_d.H_1(W_m, N_d, \tau)) \\ &= D.(\tau) = \vartheta\end{aligned}$$

Upon validation, the proxy signcrypter P_1 generates the secret key $x_1 = (V + a_p) \bmod q$ and subsequently computes and makes the public key $x_2 = x_1 \cdot D$ publicly available.

3.1.5 Proxy Signcryption:

In the proxy signcryption phase, P_1 executes the following actions to create a signcryption on m:

1. Select an arbitrary number $J \in \{1, 2, \ldots, q-1\}$.
2. Calculate $\psi = J \cdot D$
3. Calculate $K = H_1(K_{sr} + \psi)$, where K_{sr} is the shared secret key.
4. Calculate the ciphertext $C = \epsilon_K(m)$.
5. Calculate the hash function $\Omega = H_2(C, \psi)$.
6. Calculate the signature $S = (J/x_1 - \Omega) \bmod q$
7. Proxy Signcrypter send (C, Ω, S) to proxy unsigncrypter P_2.

3.1.6 Verification and Unsigncryption:

In this subsection, receiving the tuple (C, Ω, S) proxy unsigncrypter carry out the following steps for verication and decryption of the signcrypted message.
First recover $\psi = x_2.(S + \Omega)$ and$\psi = x_1.D.((J/x_1) - \Omega + \Omega) = J.D$
Following this, Verify signature $\omega^* = H_2(C, \omega)$. Accept the signature if $\Omega = \Omega^*$
Compute $K = H_1(K_{sr} + \psi)$
Decrypt $m = D_K(C)$

3.2 Security Analysis of Insafullah's Scheme

Insafullah's scheme [1] meets the security characteristics of being unforgeable under adaptive choosen message attacks (UU-ACMA) and indistinguishable under

adaptive chosen ciphertext attacks (IND- CCA) by adversary A. Theorems 1 and 2 can provide a more in-depth understanding of the potential security risks and vulnerabilities associated with Insafullah's proposed scheme [**?**]. This analysis will shed light on the threats and challenges this scheme may face.

Theorem 1:

Insafullah's scheme [1] ensures IND-CCA security when adversary A, assisted by a challenger Φ, cannot achieve a significant advantage in the following phases:

Phase 1:

In phase 1, the security is evaluated by examining the hash function queries submitted by adversary A and then we will check proxy delegation query, proxy Signcryption query and proxy unsigncryption query submitted by A. In the challenge phase, adversary A engages by submitting two plaintexts $\mathbf{m_a}$ and $\mathbf{m_b}$.

1. H_0 **Query:**
   If A sends the query (W_m, J) and the corresponding value in the list is LH_0, then Φ provides the values value of τ to A. On the contrary, Φ randomly selects τ and returns them to A.

2. H_1 **Query:**
   When this query is submitted by A, Φ checks the corresponding value in the list.

   If the value is LH_1, Φ provides the values of K_i to A. Alternatively, if LH_1 is not found, Φ randomly selects K_i and returns them to A.

3. H_2 **Query:**
   In the scenario of hash H_2 queries, when individual A submits a query, and LH_2 is found, Φ provides ω_i. Otherwise, Φ randomly selects ω_i.

4. **Private Key Generation Query:**

   If A requests the signer's private and public key, Φ randomly selects s_i and sends $(s_i, s_i \cdot D)$ to A.

   Proxy Delegation Query:

   Upon A's submission, Φ randomly selects ζ and τ to compute $\vartheta = \zeta \cdot D$. Then, Φ computes $V = (\zeta - ad \cdot \tau)$ and responds (V, τ, W_m).

5. **Proxy Signcryption Query:**

   When A submits a query with message m, proxy private key a_p, and delegation (V, τ, W_m), Φ responds with a valid proxy signcryption. Φ chooses random $J, K, \Omega, x_1 \in \{1, 2, 3, ..., q-1\}$, computes $\mathsf{C} = \mathsf{E}_{\mathsf{K}}(m)$, and the signature $\mathcal{S} = (J/x_1 - \Omega)$, responding $(\mathsf{C}, \Omega, \mathcal{S})$.

6. **Proxy Unsigncryption Query:**

   When A submits this query, if the query is not for the target participant, Φ returns valid plaintext from proxy unsigncryption. Otherwise, an incorrect proxy signcryption tuple is returned.

7. **Challenge:**

   A transmits two plaintexts $\mathbf{m_a}$ and $\mathbf{m_b}$. Φ randomly selects $\lfloor \in \{0, 1\}$ and generates an unintelligible text. Φ selects x_1,ψ, K_{sr} randomly, computes

   $\mathcal{K} = \mathcal{H}_1(\mathcal{K}_{\mathrm{sr}} + \psi)$, $\mathsf{C}^* = \mathsf{E}_{\mathsf{K}}(\mathrm{m})$, and $\Omega^* = \mathcal{H}_2(\mathsf{C}^*, \psi)$. Computes $S^* = (J/x_1 - \Omega)$ and responds $(\mathsf{C}^*, \Omega^*, \mathcal{S}^*)$.

Phase 2:

A can submit the same queries as in Phase 1, excluding the receiver's private key and a message corresponding to $(\mathsf{C}^*, \Omega^*, S^*)$. A returns $\lfloor \in \{0, 1\}$, and if $\lfloor^* = \lfloor$, returns 1; otherwise, returns 0.

1. If $\mathfrak{Q} = \mathcal{O} \cdot \mathbf{V} \cdot \mathcal{D}$ is an authentic signcrypted message, extracting the bit b using advantage π is $Pr[\oplus \rightarrow 1|Q = \mathcal{O} \cdot \mathbf{V} \cdot \mathcal{D}] = Pr[b \neq b|Q = \mathcal{O} \cdot \mathbf{V} \cdot \mathcal{D}] = \frac{1}{2} + \pi$.

2. If $Q \neq \mathcal{O} \cdot \mathbf{V} \cdot \mathcal{D}$, A may not extract b with advantages, resulting in $Pr[\oplus \rightarrow 1|Q \neq \mathcal{O} \cdot \mathbf{V} \cdot \mathcal{D}] = Pr[^{*b} \neq b|Q \neq \mathcal{O} \cdot \mathbf{V} \cdot \mathcal{D}] = \frac{1}{2}$.

Theorem 2:

This theorem affirms the security of the proposed novel scheme, asserting its ability to achieve IND-CEGPS-ACMA security. The condition for this security is that adversary A, working alongside challenger Φ, cannot gain a substantial advantage leading to victory in subsequent steps.

Proof:

Insafullah's scheme introduces a curve HC instance $(\mathfrak{Q}, \mathbf{V}, \mathcal{D})$ to Φ. The goal for Φ is to compute $\mathfrak{Q} = \mathbf{V} \cdot \mathcal{D} = x_2$. During the setup phase, Φ performs the necessary steps to generate the global parameter J and communicates it to A.

In Phase 1, the querying process is similar to that described in Theorem 1.

Regarding forgery, utilizing the forking lemma [24], Φ can acquire two valid proxy signcryption texts: $\psi = (\mathsf{C}, \mathcal{S}, \Omega)$ and $\psi^* = (\mathsf{C}, \mathcal{S}, \Omega^*)$. This results in the verification equations $\mu = x_2 \cdot (\mathcal{S} + \Omega)$ and $\mu^* = x_2 \cdot (\mathcal{S} + \Omega^*)$. Upon subtraction, we obtain:

$$\mu - \mu^* = x_2 \cdot (S + \Omega) - (x_2 \cdot S + \Omega^*) \tag{3.1}$$

$$= x_2 \cdot S + x_2 \cdot \Omega - x_2 \cdot S + x_2 \cdot \Omega^* \tag{3.2}$$

$$= x_2 \cdot \Omega - x_2 \cdot \Omega^* \& = \mu - \mu^* = \zeta \cdot \mathcal{D} - \zeta^* \cdot \mathcal{D} \tag{3.3}$$

$$= \mathbf{V} \cdot \mathcal{D} \cdot \Omega - \mathbf{V} \cdot D \cdot \Omega^* \tag{3.4}$$

$$= \zeta - \zeta^* \cdot \mathcal{D} \cdot \mathbf{V} \cdot \mathcal{D} \cdot (\Omega - \Omega^*) \tag{3.5}$$

$$= (\zeta - \zeta\,) \cdot \mathcal{D} = \cdot \mathbf{V} \cdot \mathcal{D}(\Omega - \Omega^*) \tag{3.6}$$

$$= (\zeta - \zeta^*) \cdot \mathbf{V} \cdot (\Omega - \Omega^*) \tag{3.7}$$

Probability Analysis:

Suppose there are three types of queries: q_{pk} for private key queries, q_{pd} for proxy delegation queries, and q_{psn} for proxy signcryption queries. We define the following measurements (M):

1. M1: Φ outputs positively in private key queries with a probability of $1 - \frac{p_{kq}}{2k}$.

2. M2: Φ outputs positively in proxy unsigncryption queries with a probability of $1 - \frac{1}{2k}$.

3. M3: Φ outputs positively in the challenge part with a probability of $\frac{1}{p_{kq}-2k}$.

The total probability is given by:

$$P_r[\Phi \to \pi] = P_r[M1 \wedge M2 \wedge M3] = \left(1 - \frac{p_{kq}}{2k}\right)\left(1 - \frac{1}{2k}\right)\left(\frac{1}{p_{kq} - 2k}\right) \cdot \pi.$$

Chapter 4

Cost Effective Generalized Proxy Signcryption Scheme for Internet of things (IoT)

In this chapter, the signcryption scheme presented in Chapter 3 is extended to a new generalized proxy signcryption scheme for internet of things (IoT). It is based on hyper elliptic curve and its security is based on difficulty of Hyperelliptic Curve Discrete Logarithm Problem (HCDLP).

4.1 The Proposed Generalized Proxy Signcryption Scheme

In this section, the newly propsed generalized cost effective proxy signcryption scheme is explained. It comprises of setup phase, the setting of global parameters followed by various algorithm for proxy signcryption and unsigncryption. When only confidentiality or authenticity is required, it smoothly executes a single encryption or signature mode without the need for modifications or additional computations.

4.1.1 Setup

In the setup phase, the certificate authority CA selects a hypereliptic curve (HC), creates a system parameter set as $\zeta = \{HC, D, H_0, H_1, H_2, H_3, \epsilon\}$ with 80-bit parameter size,where HC is hyper eliptic curve, D is divisor of HC, (H_0, H_1, H_2, H_3) are hash functions and ϵ is the public key. CA created this public key by picking a random number π, and then computing ϵ as point on HC by uing $\epsilon = \pi \cdot D$. Finally, CA ensures that ζ is publicly available in a network.
The table below lists all the global parameters used in the scheme.

TABLE 4.1: Global parameters.

Symbols	Descriptions
D	Divisor of HC
Uo	Delegator
PS	Proxy signcrypter
PU	Proxy unsigncrypter
a_d	Delegator's private key
a_s	Proxy signcrypter's private key
a_u	Proxy unsigncrypter's private key
f_d	Delegator's public key
f_s	Proxy signcrypter's public key
f_u	Proxy unsigncrypter's public key
N_d	Nonce for delegator
N_s	Nonce for proxy signcrypter
H_0, H_1, H_2, H_3	hash functions
α, β	Private and public keys for signature generation
E_K	Symmetric key encryption algorithm with key K
D_K	Symmetric key decryption algorithm with key K
T	Time stamp

4.1.2 Key Generation

The three participants U_0, PS, PU calculates their keys by using the formula. Participant x chooses a number at random $\{a_x = 1, 2, \ldots, q-1\}$ and calculate $f_x = a_x \cdot D$. The participant x private and public keys are represented by a_x and f_x. Where $x = d$ is for proxy delegator, $x = s$ is for proxy signcrypter and $x = u$ is for proxy unsigncrypter.

4.1.3 Proxy Delegation

In this particular section, the initial signcrypter/proxy delegator, denoted as U_0, grants the signing authority to a proxy signcrypter, referred to as PS.

Algorithm 4.1.1.

Input: W_m (The warrant message), D (Divisor of HC), a_d

Output: (τ, V, W_m)

1. Proxy Delegator selects $\kappa \in \{1, 2, \ldots, q-1\}$
2. Calculate $w = \kappa \cdot D$
3. Calculate $\tau = H_0(W_m, w)$
4. Compute $V = (\kappa - a_d \cdot \tau) \bmod q$

Proxy delegator sends (τ, V, W_m) to proxy signcrypter for validation.

4.1.4 Generalized Proxy Signcryption

Before executing the signcryption algorithm, PS ensures the validity of the process by undergoing the following steps for validation.

$$
\begin{aligned}
w &= V \cdot D + H_1(W_m, N_d, \tau) \cdot f_d \\
&= V \cdot D + H_1(W_m, N_d, \tau) \cdot f_d \\
&= (\kappa - ad \cdot H_1(W_m, N_d, \tau)) \cdot D + H_1(W_m, N_d, \tau) \cdot ad \cdot D \\
&= (\kappa - a_d \cdot H_1(W_m, N_d, T) + a_d \cdot H_1(W_m, N_d, \tau)) \cdot D \\
&= (\kappa) \cdot D = w
\end{aligned}
$$

After validation, PS creates the secret private key $\alpha = (V + a_s) \bmod q$, calculates the public key $\beta = \alpha \cdot D$. These keys will be used for signature generation and validation of signature in the scheme. α should be invertible mod$q$

Algorithm 4.1.2.

Input: D, m, T, H_1

Output: (C, Ω, S, μ)

1. Choose an arbitrary $r \in \{1, 2, \ldots, q-1\}$
2. Calculate $\mu = r \cdot D$, where D is a divisor over HC.
3. Compute $K = H_1(r \cdot fu)$
4. Compute the cipher text $C = E_K(m, T)$
5. Compute the hash function $\Omega = H_1(c, \mu)$
6. Compute the signature $S = (\frac{r}{\alpha} - \Omega) \bmod q$

Proxy Signcrypter sends (C, Ω, S, μ) to proxy unsigncrypter.

4.1.5 Generalized Proxy Unsigncryption

After receiving (C, Ω, S) from proxy signcrypter PS, Proxy unsigncrypter PU performs the following steps for unsigncryption.

Algorithm 4.1.3.

Input: (β, Ω, S, au)

Output: (m, T)

1. Compute $K^* = H_1(\mu \cdot a_u)$
2. Recover $\mu^* = \beta \cdot (S + \Omega) = \mu^* = \alpha \cdot D\left(\frac{r}{\alpha} - \Omega + \Omega\right)$
3. Decrypt $(m, T) = D_{K^*}(C)$
4. Compute $\Omega^* = H_1(m, T, \mu^*)$

If $\Omega^* = \Omega$ then accept the cipher text otherwise return the error message.

TABLE 4.2: Generalized Proxy Signcryption

Proxy Signcryption	**Proxy Unsigncryption**
Choose an arbitrary $r \in \{1, 2, \ldots, q-1\}$	Calculate $K^* = H_1(\mu \cdot a_u)$
Calculate $\mu = r \cdot D$	Recover $\mu^* = \beta \cdot (S + \Omega)$
Compute $K = H_1(r \cdot f_u)$	Decrypt $(m, T) = D_{K^*}(C)$
Compute the cipher text $C = E_K(m, T)$	Compute $\Omega^* = H_1(m, T, \mu^*)$
Compute the hash function $\Omega = H_1(C, \mu)$	Accept the cipher text if $\Omega^* = \Omega$
Compute the signature $S = (\frac{r}{\alpha} - \Omega) \bmod q$	
Send (C, Ω, S, μ) to PU	

4.1.6 Signature Only Mode

The following actions are taken by proxy signcrypter PS in this subsection in order to sign the message (m).

Algorithm 4.1.4.

Input:H_1, m, T

Output:(μ, S)

1. Choose an arbitrary $r \in \{1, 2, \ldots, q-1\}$
2. Compute $\mu = r \cdot D.$ where D is the divisor over HC.
3. Get $C = (m, T)$
4. Compute the hash function $\Omega = H_1(C, \mu)$
5. Compute the signature $S = \left(\frac{r}{\alpha} - \Omega\right) \bmod q$

Proxy signcrypter send (μ, S) to proxy unsighncrypter.

4.1.6.1 Signature Verification

Proxy unsigncrypter PS verifies the content of message (μ, S) and then perform following steps for signature.

Algorithm 4.1.5.

1. Recover $\mu^* = \beta \cdot (S + \Omega)$
2. Compute the hash function $\Omega^* = H_1(c, \mu^*)$
3. If $\Omega^* = \Omega$, Ciphertext is accepted as valid.

TABLE 4.3: Signature and Verification

Signature	**Verification**
Choose an arbitrary $r \in \{1, 2, \ldots, q-1\}$	Recover $\mu^* = \beta \cdot (S + \Omega)$
Compute $\mu = r \cdot D$	Compute $\Omega = H_1(C, \mu)$
Get $C = (m, T)$	If $\Omega^* = \Omega$, Ciphertext is accepted
Compute the hash function $\Omega^* = H_1(c, \mu^*)$	
Compute the signature $S = \left(\frac{r}{\alpha} - \Omega\right) \bmod q$	
Send (S, μ) to PU	

4.1.7 Encryption Only mode

The following actions are taken by proxy signcrypter PS in this subsection in order to generate encryption on the message (m).

Algorithm 4.1.6.

1. Choose an arbitrary $r \in \{1, 2, ..., q-1\}$
2. Compute $\mu = r \cdot D$ where D is the divisor over HC.
3. Compute $K = H_1(r \cdot fu)$
4. Compute the cipher text $C = E_K(m, T)$
5. Compute the hash function $\Omega = H_1(C, \mu)$

Send (C, μ, Ω) to the proxy unsigncrypter.

4.1.7.1 Decryption

To obtain the original plaintext message, the recipient executes the decryption algorithm.

Algorithm 4.1.7.

1. Compute $K^* = H_1(\mu \cdot au)$

2. Check $K^* = H_1(\mu \cdot a_u) = H_1(r \cdot D \cdot a_u) = H_1(r \cdot f \cdot u) = K$

3. Decrypt $(m, T) = D_{k^*}(C)$

TABLE 4.4: Encryption and Decryption

Encryption	**Decryption**
Choose an arbitrary $r \in \{1, 2, ..., q-1\}$	Calculate $K^* = H_1$
Compute $\mu = r \cdot D$	Decrypt $(m, T) = D_{k^*}(C)$
Compute $K = H_1(r \cdot fu)$	
Compute the cipher text $C = E_K(m, T)$	
Compute the hash function $\Omega = H_1(C, \mu)$	
Send (C, μ, Ω) to PU	

4.1.8 Correctness

For the correctness of the above algorithms, we need to establish the validity of the signature and the decryption.

Theorem 4.1.8. (The Validity of Signature)
In the settings of the above algorithms, If the receiver validates the equation then the signature is considered as valid.

$$\mu^* = \mu$$

Proof.
From step 2 of Algorithm 4.1.4, We have

$$\begin{aligned} \mu^* &= \beta \cdot (S + \Omega) \\ \mu^* &= \alpha \cdot D \left(\frac{r}{\alpha} - \Omega + \Omega \right) \\ &= r \cdot D = \mu \end{aligned}$$

□

Theorem 4.1.9. (Encryption only mode)
In the setting of the above algorithm, the decryption correctly recovers the original

message.

$$K^* = K$$

Proof.

The proof follows from the fact that

$$\begin{aligned} K^* &= H_1(\mu \cdot a_u) \\ &= H_1(r \cdot D \cdot a_u) \\ &= H_1(r \cdot f_u) = K \end{aligned}$$

□

4.2 Security Analysis of proposed Generalized Proxy Signcryption Scheme (CEGPS)

In this section we will address the security properties confidentiality, authenticity, integrity, non- repudiation, unforgeability, and forward secrecy.

4.2.1 Unforgeability of warrant

Unforgeability ensure that, the attacker can not generate a valid signature for warrant.

In this scheme, attacker will not be able to construct original signature without knowing private key of original signcrypter U_0. In the step 4 of Algorithm 4.1.3, we have

$$V = (\kappa - a_d.\tau) \bmod q$$

Note that from Step 2 of Algorithm 4.1.3, we have

$$\omega = \kappa \cdot D$$

To find random number κ and Delegator's private key a_d, attacker has to solve two HECDLPs, which would be computationally impossible.

$$f_d = a_d.D$$

4.2.2 Unforgeability of message

In the above proposed scheme, only proxy Signcrypter generate valid signature for message. Note that in step 6 of Algorithm 4.1.2 and in in the proxy unsigncryption's verfication step 2 of Algorithm 4.1.4, we have

$$S = (r/\alpha - \Omega) \bmod q$$

$$\mu^* = \beta \cdot (S + \Omega)$$

The signature computation involves the proxy signcrypter's secret key α. Therefore, an adversary without the correct secret key α cannot forge a valid signature. The proxy unsigncrypter's public key is required to compute μ^* correctly. For the knowledge of β from $\beta = \alpha \cdot D$, an adversary has to solve HECDLP which would be computationally impossible.

4.2.3 Confidentiality

In our approach, the attacker has to obtain the secret key K in order to determine the original message. The attacker has a variety of options when attempting to obtain the secret key K.

Case 1:

From Step 3 of Algorithm 4.1.2, we have

$$K = H_1(r \cdot fu)$$

The attacker calculates K from Step 3 of Algorithm 4.1.2. But he needs secret parameter r. The attacker just knows the public key f_u of proxy unsighncrypter then he solve HECDLP first,it is computationally infeasible for the attacker to get K.

Case 2:

From Step 6 of Algorithm 4.1.2, we have

$$S = \left(\frac{r}{\alpha} - \Omega\right)$$

In this case the attacker needs the secret private key of proxy signcrpter α. To calculate α, one has to solve the HECDHP as follows:

$$\alpha = (V + as)$$

4.2.4 Warrant Integrity

If an attacker modifies W_m to $W_{m'}$, then $\tau = H_0(\alpha, W_{m'})$ is constructed. Consequently, the attacker determines the private key a_d of the delegator from Step 4 of Algorithm 4.1.3 and the random number κ from Step 2 of Algorithm 4.1.3 as follows

$$\kappa - a_d \cdot T$$

$$f_d = a_d.D$$

$$\omega = \kappa \cdot D$$

This process is equivalent to solving two HECDLP problems, which are computationally infeasible for the attacker.

To ensure warrant integrity, the proxy signcrypter verifies the equivalence of given equation in proxy signcrypter's validation setps, if satisfied, it indicates that the

warrant has not been altered.

$$V \cdot D + H_1(W_m, Nd, \tau) \cdot f_d$$

4.2.5 Message Integrity

The message associated with the original ciphertext C is denoted as m. In the event that an attacker modifies the ciphertext from C to $C^{'}$, the corresponding message is altered to $m^{'}$. Let $\Omega^* = H_1(C^{'})$, The one-way property of the hash function ensures that it is computationally infeasible for the attacker to change C to $C^{'}$. To maintain message integrity, the proxy unsigncrypter compares the computed hash Ω^* with the original hash Ω. If Ω^* matches Ω, it indicates that the message is unchanged from its original form when transmitted by the proxy signcrypter.

4.2.6 NonRepudiation

If there is a disagreement between the delegator and the proxy signcrypter, a dependable third party will verify the authenticity of the warrant and make the appropriate ruling. The third party receives (τ, V, W_m) from the proxy. The third party takes the following actions:

- Verifies Delegator's public key f_d.
- Uses the one-way hash function to generate $H_0(W_m, \omega)$. The sender has sent (τ, V, W_m) to the proxy, otherwise not.

4.2.7 Forward Secrecy

Our security system, keeps messages safe by using a clever method called forward secrecy. Even if someone gets hold of the secret key used for signing and encrypting

messages, they still can not read the messages. This is because we use a different key, called the session secret key, for actually hiding and revealing the messages.

To make things even more secure, we make it really hard for someone trying to break into our system. They would need a special secret number (called r) that only the person sending the message knows. Solving the complicated math problem involved in getting our main secret key (K) from this number is practically impossible for attackers. This makes it extremely tough for them to decrypt messages even if they manage to get a part of the system's secret.

4.3 Performance Efficiency

This section will compare the computation and communication costs of proposed scheme with those of Guo and Deng 2020 [36], Abdelfatah [37], Saddam Hussain et al.[22], Hundera [10], Insafullah et al. [23] respectively, to assess its performance efficiency.

4.3.1 Computational Cost

In this section, we assess the compatibility of our proposed scheme with existing schemes. We utilized the Multiprecision Integer and Rational Arithmetic C Library (MIRACL) [25] to conduct the evaluation. The runtime of fundamental cryptographic operations was tested on a Raspberry Pi 3 B+ Rev 1.3, running Ubuntu 20.04 LTS (64-bit) with a 64-bit CPU and a 1.4 GHz Quad-Core processor. The device was equipped with 1 GB of memory. Our proposed scheme prove to be quicker than the previous schemes. Table 4.2 presents the running time for basic cryptographic operations, including $\mathscr{EM}$, $\mathscr{HEM}$, and C.

TABLE 4.5: Global parameters.

$\mathscr{EM}$	$\mathscr{HEM}$	$\mathscr{PR}$
2.28 ms	1.14 ms	32.08 ms

TABLE 4.6: Major Operation Comparison

Schemes	[36]	[37]	[22]	[10]	[23]	**proposed**
Proxy Delegation	$4\mathscr{EM}$	$3\mathscr{EM}$	$4\mathscr{HEM}$	$2\mathscr{PR}$	$3\mathscr{HEM}$	$3\mathscr{HEM}$
Proxy Signcryption	$5\mathscr{EM}$	$1\mathscr{EM}$	$5\mathscr{HEM}$	-	$1\mathscr{HEM}$	$2\mathscr{HEM}$
Proxy Verification & unsigncryption	$5\mathscr{EM}$	$1\mathscr{EM}$	$6\mathscr{HEM}$	$4\mathscr{PR}$	$1\mathscr{HEM}$	$2\mathscr{HEM}$
Total	$14\mathscr{HEM}$	$5\mathscr{EM}$	$15\mathscr{HEM}$	$6\mathscr{PR}$	$5\mathscr{HEM}$	$7\mathscr{HEM}$

TABLE 4.7: Computational Cost Comparison

Schemes	[36]	[37]	[22]	[10]	[23]	**proposed**
Proxy Delegation	9.12	6.84	4.56	64.16	3.42	3.42
Proxy Signcryption	11.40	2.28	5.7	0	1.14	2.28
Proxy Verification & unsigncryption	11.40	2.28	6.84	128.32	1.14	2.28
Total	31.92	11.40	17.1	192.48	5.70	7.98

4.3.2 Reduction in Computation Costs (RCC)

The Reduction in Computation Costs (RCC) can be determined with the following formula:

$$RCC = \frac{1}{\text{Previous scheme computation cost}} \Big(\text{Previous scheme computation cost} - \text{Proposed Scheme computation cost} \Big) \times 100$$

- RCC of the CEGPS from Guo and Deng. [36] is:

$$\left(\frac{14\mathscr{EM} - 7\mathscr{HEM}}{14\mathscr{EM}}\right) \times 100 = \left(\frac{31.92 - 7.98}{31.92}\right) \times 100 = 75\%$$

- RCC of the CEGPS from Abdelfatah. [37] is:

$$\left(\frac{5\mathscr{EM} - 7\mathscr{HEM}}{5\mathscr{EM}}\right) \times 100 = \left(\frac{11.4 - 7.98}{11.4}\right) \times 100 = 30\%$$

- RCC of the CEGPS from Hussain et al. [22] is:

$$\left(\frac{15\mathscr{HEM} - 7\mathscr{HEM}}{15\mathscr{HEM}}\right) \times 100 = \left(\frac{17.1 - 7.98}{17.1}\right) \times 100 = 53\%$$

- RCC of the CEGPS from N. W. Hundera et al. [10] is:

$$\left(\frac{6\mathscr{PR} - 7\mathscr{HEM}}{\mathscr{PR}}\right) \times 100 = \left(\frac{192.48 - 7.98}{192.48}\right) \times 100 = 95.85\%$$

- RCC of the CEGPS from Insafullah et al. [23] is:

$$\left(\frac{5\mathscr{HEM} - 7\mathscr{HEM}}{5\mathscr{HEM}}\right) \times 100 = \left(\frac{5.7 - 7.98}{5.7}\right) \times 100 = 40\%$$

4.4 Communication Overhead

Communication overhead is a crucial aspect to consider when comparing cryptographic schemes. It refers to the additional data that needs to be transmitted along with the original message.

A scheme with high communication overhead can lead to increased latency, bandwidth consumption, and potential security risks. In evaluating cryptographic schemes, it is important to assess their impact on commu-nication overhead, as it directly affects the efficiency and scalability of the system. Lower communication overhead is generally desirable, as it allows for faster and more streamlined data transmission, reducing the burden on network resources and enhancing overall performance. This makes the system more efficient and

scalable, improving the user experience. Choosing a cryptographic scheme with low overhead helps in maintaining high-speed communication and reducing costs.

TABLE 4.8: Communication overhead comparisons in terms of extra parameters.

Schemes	**Proxy delegation**	**Proxy signcryption**	**Total**
[36]	$2\|p\| + \|W_m\|$	$5\|p\| + \|C\| + \|W_m\|$	$7\|p\| + \|C\| + \|W_m\|$
[37]	$2\|p\| + \|W_m\|$	$1\|p\| + \|C\| + \|H\|$	$3\|p\| + \|W_m\| + \|C\| + \|H_i\|$
[22]	$\|n\| + \|mw\|$	$5\|n\| + \|C\| + \|W_m\|$	$7\|n\| + \|C\| + 2\|W_m\|$
[10]	$2\|G\| + \|W_m\|$	$1\|G\| + \|C\| + \|H\| + \|W_m\|$	$3\|G\| + \|C\| + \|H\| + 2\|W_m\|$
[23]	$2\|q\| + \|W_m\|$	$1\|q\| + \|C\| + \|H\|$	$3\|q\| + \|C\| + \|H\| + \|W_m\|$
Proposed	$2\|q\| + \|W_m\|$	$1\|q\| + \|C\| + \|H\|$	$3\|q\| + \|C\| + \|H\| + \|W_m\|$

TABLE 4.9: Communication overhead comparisons in terms of extra parameters.

Schemes	**Proxy delegation**	**Proxy signcryption**	**Total**
[36]	$2\|160\| + \|1024\|$	$5\|160\| + \|1024\| + \|1024\|$	$7\|160\| + \|1024\| + \|1024\|$
[37]	$2\|160\| + \|1024\|$	$1\|160\| + \|1024\| + \|512\|$	$3\|160\| + \|1024\| + \|1024\| + \|512\|$
[22]	$\|80\| + \|1024\|$	$5\|80\| + \|512\| + \|1024\|$	$7\|80\| + \|512\| + 2\|1024\|$
[10]	$2\|512\| + \|1024\|$	$1\|512\| + \|1024\| + \|512\| + \|1024\|$	$3\|512\| + \|1024\| + \|512\| + 2\|1024\|$
[23]	$2\|80\| + \|1024\|$	$1\|80\| + \|1024\| + \|512\|$	$3\|80\| + \|1024\| + \|512\| + \|1024\|$
Proposed	$2\|80\| + \|1024\|$	$1\|80\| + \|1024\| + \|512\|$	$3\|80\| + \|1024\| + \|512\| + \|1024\|$

TABLE 4.10: Communication overhead comparisons in terms of extra parameters.

Schemes	Proxy delegation	Proxy signcryption	Total
[36]	1344	2848	4192
[37]	1344	1696	3040
[22]	1184	1936	3120
[10]	2048	3072	5120
[23]	1184	1616	2800
Proposed	1184	1616	2800

4.4.1 Reduction in Communication Cost (RCMC)

The Reduction in Communicatin Cost (RCMC) can be determined with the following formula:

$$RCC = \frac{1}{\text{Previous scheme cost}} \Big(\text{Previous scheme cost} - \text{Proposed Scheme cost} \Big) \times 100$$

- RCMC of the CEGPS from Guo and Deng. [36] is:

$$\left(\frac{7|p| + |C| + |W_m| - 3|q| + |C| + |H| + |W_m|}{7|p| + |C| + |W_m|} \right) \times 100$$

$$= \left(\frac{4192 - 2800}{4192} \right) \times 100 = 33.2\%$$

- RCMC of the CEGPS from Abdelfatah. [37] is:

$$\left(\frac{3|p| + |W_m| + |C| + |H_i| - 3|q| + |C| + |H| + |W_m|}{3|p| + |W_m| + |C| + |H_i|}\right) \times 100$$

$$= \left(\frac{3040 - 2800}{3040}\right) \times 100 = 7.89\%$$

- RCMC of the CEGPS from Hussain et al. [22] is:

$$\left(\frac{7|n| + |C| + 2|W_m| - 3|q| + |C| + |H| + |W_m|}{7|n| + |C| + 2|W_m|}\right) \times 100$$

$$= \left(\frac{3120 - 2800}{3120}\right) \times 100 = 10.2\%$$

- RCMC of the CEGPS from N. W. Hundera et al. [10] is:

$$\left(\frac{3|G| + |C| + |H| + 2|W_m| - 3|q| + |C| + |H| + |W_m|}{3|G| + |C| + |H| + 2|W_m|}\right) \times 100$$

$$= \left(\frac{5120 - 2800}{5120}\right) \times 100 = 45.3\%$$

- RCMC of the CEGPS from Insafullah et al. [23] is:

$$\left(\frac{3|q| + |C| + |H| + |W_m| - 3|q| + |C| + |H| + |W_m|}{3|q| + |C| + |H| + |W_m|}\right) \times 100$$

$$= \left(\frac{2800 - 2800}{2800}\right) \times 100 = 0\%$$

4.5 Formal Security Analysis

Theorem 4.5.1. If adversary A cannot solve the Hidden Extended Diffie-Hellman Problem (HEDHP), our Cost-Effective Generalized Proxy Signcryption

(CEGPS) scheme can provide protection against an Indistinguishable Chosen Ciphertext Attack (IND- CEGPS-CCA).

(IND- CEGPS-CCA) refers to the property that an attacker cannot distinguish between two ciphertexts generated from different plaintexts, even if the attacker has access to the decryption oracle and can choose the ciphertexts to be decrypted

4.5.1 Initialization:

In this phase, the entity denoted as C_{HECGHP} is setting up the groundwork for its cryptographic system. First, it selects a security parameter, represented as l, to determine the strength of the underlying hyperelliptic curve (HC). The goal is to ensure a reasonable level of security, and the choice of l plays a role in achieving this. The system parameters are then formed into a set called ζ, which includes the hyperelliptic curve (HC), a divisor of HC named D, and four hash functions denoted as (H_0, H_1, H_2, H_3). The size of these parameters is set to 80 bits, indicating a balanced level of security and efficiency.

Furthermore, a crucial element is introduced to the system called ϵ, which serves as the public key. This public key is generated by randomly selecting a number (π) and then computing ϵ as a point on the hyperelliptic curve (HC) using the formula $\epsilon = \pi \cdot D$. In simpler terms, this means taking the randomly chosen number π and performing a mathematical operation involving the base point D on the hyperelliptic curve to obtain the public key ϵ.

4.5.2 Phase I

After the successful completion of the Initialization phase, A poses the following queries, and C_{HECGHP} is required to respond.

1. H_0 **queries**

   C_{HECGHP} maintains a list L_1, consisting of (τ, W_m). Upon receiving an H_0

query, C_{HECGHP} checks if the corresponding value is in the list L_1. If the value is in L_1, C_{HECGHP} provides the associated values τ to A. On the other hand, if the value in the list is not in the list L_1 , C_{HECGHP} randomly selects τ and send it to A.

2. H_1 **queries**

   In the context of hash H_1 queries, when individual A submits a query, C_{HECGHP} examines the corresponding value in the list. If the identified value is in the list L_2 , C_{HECGHP} promptly provides the associated values K to A. However, if the value in the list does not match, C_{HECGHP} takes the initiative to randomly select K and send it to A.

3. H_2 **queries:**

   In the scenario of hash H_2 queries, when individual A submits a query, C_{HECGHP} examines the corresponding value in the list L_3. If the identified value is in L_3, C_{HECGHP} promptly provides the associated values Ω to A. Conversely, if the value in the list does not match, C_{HECGHP} takes the initiative to randomly select values Ω and sends them to A.

4. **key generation query** If A sends a request for the signer's private key and public key, and C_{HECGHP} chooses at random $a_x \in \{1, 2, \ldots, q-1\}$, then compute $f_x = a_x \cdot D$ and send (a_x, f_x) to A.

5. **Proxy Delegation query**

   Upon A's submission of this query, C_{HECGHP} randomly select κ and τ to compute $w = \kappa \cdot D$.then C_{HECGHP} will compute $V = (\kappa\text{-}ad \cdot \tau)$ and respond (V, τ, W_m) to A.

6. **Generalized Proxy Signcryption query**

   Upon submission of this query by A along with message (m), proxy private

key (a_p), and delegation (V, τ, W_m), C_{HECGHP} responds to A as a valid proxy signcryption in the manner shown below.

C_{HECGHP} chooses random number r, $\Omega, \alpha \in \{1, 2, 3, \ldots, q-1\}$. Compute the cipher text $\mathsf{C} = \mathsf{E_K}(\mathrm{m}, T)$ also compute the signature $\mathcal{S} = (\frac{r}{\alpha} - \Omega)$ and respond $(\mathsf{C}, \Omega, \mathcal{S})$ as proxy signcryption to A.

7. **Generalized Proxy Unsigncryption query**

   When this query is submitted to C_{HECGHP} by A, if it is for the target participant, C_{HECGHP} returns valid plaintext created by proxy unsigncryption to A. Otherwise, An incorrect proxy signcryption tuple is returned.

8. **Challenge**

   A transmits two plaintexts of equal length, denoted as $\mathbf{m_a}$ and $\mathbf{m_b}$. C_{HECGHP} randomly selects a bit $\lfloor\rangle \in \{0, 1\}$ and generates an unintelligible text as described below.

   C_{HECGHP} computes an intermediate value K using the equation $K = H_1(r \cdot fu)$.

   This value K is derived from the hash function H_1 applied to the prod-uct of r and fu. Following this, C_{HECGHP} encrypts the plaintext message m using the key K, resulting in the ciphertext C ($C = E_K(m, T)$).
   Additionally, C_{HECGHP}computes Ω by applying the hash function H_1 to the combination of the ciphertext C and the random number μ ($\Omega = H_1(c, \mu)$).

   Simultaneously, C_{HECGHP} generates a signature S using the equation $S = (\frac{r}{\alpha} - \Omega) \bmod q$. Here, r and α are parameters involved in the signature generation process.

   Finally, C_{HECGHP} forms the un-understandable text by combining the ciphertext C, the signature S, and the hash value Ω (C, S, Ω). This resulting (C, S, Ω) is then sent back to party A as the proxy signcryption for the plaintext $\mathbf{m_b}$.

4.5.3 Phase 2

A can submit the same queries as in phase 1, but it does not make a query for the receiver's private key and a message corresponding to the (C, Ω, S).
Following that, A returns $b_i^* \in \{0, 1\}$, and if $b_i = b_i^*$, returns 1. Otherwise, the outcome is 0.

Chapter 5

Conclusion

This research work introduces a new and cost-effective security system for the Internet of Things (IoT), called Cost-Effective Generalized Proxy Signcryption (CEGPS). To enhance the level of security of the presented scheme, the scheme uses hyperelliptic curves. Hyperelliptic curves can provide a higher level of security with smaller key sizes compared to traditional elliptic curves. The proposed scheme is the extension of Insafullah's signcryption scheme [23] which is presented in Chapter 3, we extended it to ensure strong security features like unforgeability and confidentiality, tested through the Random Oracle Model (ROM).

Importantly, our CEGPS scheme performs better than previous methods in both speed and data use. It combines proxy signature and signcryption features, specifically designed for the demands of the Internet of Things (IoT), while keeping computational and communication costs low.

Our results demonstrate the advantages of CEGPS, with a fast computational speed of 7.98 milliseconds and minimal communication overhead of 2800 bits. In summary, our proposed scheme enhances security using hyperelliptic curves, offering improved safety and efficiency.
This advancement holds significant potential for IoT devices, contributing to the progress of emerging technologies.

Bibliography

[1] I. Ullah, I. U. Haq, N. U. Amin, A. I. Umar, and H. Khattak, "Proxy signcryption scheme based on hyper elliptic curves," *Int. J. Comput.*, vol. 20, no. 1, pp. 157–166, 2016.

[2] W. Stallings, *Cryptography and network security, 4/E.* Pearson Education India, 2006.

[3] W. G. Barker, *Introduction to the analysis of the Data Encryption Standard (DES).* Aegean Park Press, 1991.

[4] M. A. Musa, E. F. Schaefer, and S. Wedig, "A simplified aes algorithm and its linear and differential cryptanalyses," *Cryptologia*, vol. 27, no. 2, pp. 148–177, 2003.

[5] M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[6] E. Milanov, "The rsa algorithm," *RSA laboratories*, pp. 1–11, 2009.

[7] R. Singh and S. Kumar, "Elgamals algorithm in cryptography," *International Journal of Scientific & Engineering Research*, vol. 3, no. 12, pp. 1–4, 2012.

[8] Y. Zheng, "Digital signcryption or how to achieve cost (signature & encryption) cost (signature)+ cost (encryption)," in *Advances in Cryptology-CRYPTO'97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17*, pp. 165–179, Springer, 1997.

[9] Y. Han, X. Yang, P. Wei, Y. Wang, and Y. Hu, "Ecgsc: elliptic curve based generalized signcryption," in *International conference on ubiquitous intelligence and computing*, pp. 956–965, Springer, 2006.

[10] N. W. Hundera, Q. Mei, H. Xiong, and D. M. Geressu, "A secure and efficient identity-based proxy signcryption in cloud data sharing.," *KSII Transactions on Internet & Information Systems*, vol. 14, no. 1, 2020.

[11] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures: Delegation of the power to sign messages," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 79, no. 9, pp. 1338–1354, 1996.

[12] C. Gamage, J. Leiwo, and Y. Zheng, "An efficient scheme for secure message transmission using proxy-signcryption," in *Proceedings of the Twenty Second Australasian Computer Science Conference*, pp. 18–21, 1999.

[13] X. Li and K. Chen, "Identity based proxy-signcryption scheme from pairings," in *IEEE International Conference onServices Computing, 2004.(SCC 2004). Proceedings. 2004*, pp. 494–497, IEEE, 2004.

[14] M. Wang, H. Li, and Z. Liu, "Efficient identity based proxy-signcryption schemes with forward security and public verifiability," in *International Conference on Networking and Mobile Computing*, pp. 982–991, Springer, 2005.

[15] S. Duan, Z. Cao, and Y. Zhou, "Secure delegation-by-warrant id-based proxy signcryption scheme," in *International Conference on Computational and Information Science*, pp. 445–450, Springer, 2005.

[16] D. H. Elkamshoushy, A. AbouAlsoud, and M. Madkour, "New proxy signcryption scheme with dsa verifier," in *Proceedings of the Twenty Third National Radio Science Conference (NRSC'2006)*, pp. 1–8, IEEE, 2006.

[17] H. Elkamchouchi, M. Nasr, and R. Ismail, "A new efficient strong proxy signcryption scheme based on a combination of hard problems," in *2009 IEEE International Conference on Systems, Man and Cybernetics*, pp. 5123–5127, IEEE, 2009.

[18] H. Elkamchouchi and Y. Abouelseoud, "A new proxy identity-based signcryption scheme for partial delegation of signing rights," *Cryptology ePrint Archive*, 2008.

[19] H.-Y. Lin, T.-S. Wu, S.-K. Huang, and Y.-S. Yeh, "Efficient proxy signcryption scheme with provable cca and cma security," *Computers & Mathematics with Applications*, vol. 60, no. 7, pp. 1850–1858, 2010.

[20] H. M. Elkamchouchi, Y. Abouelseoud, and W. S. Shouaib, "A new proxy signcryption scheme using warrants," *International Journal of Intelligent Engineering Informatics*, vol. 1, no. 3-4, pp. 309–327, 2011.

[21] Y. Ming and Y. Wang, "Proxy signcryption scheme in the standard model," *Security and Communication Networks*, vol. 8, no. 8, pp. 1431–1446, 2015.

[22] S. Hussain, I. Ullah, H. Khattak, M. A. Khan, C.-M. Chen, and S. Kumari, "A lightweight and provable secure identity-based generalized proxy signcryption (ibgps) scheme for industrial internet of things (iiot)," *Journal of Information Security and Applications*, vol. 58, p. 102625, 2021.

[23] I. Ullah, A. Alkhalifah, M. A. Khan, and S. M. Mostafa, "Cost-effective proxy signcryption scheme for internet of things," *Mobile Information Systems*, vol. 2021, pp. 1–10, 2021.

[24] J. J. Rotman, *Journey into mathematics: An introduction to proofs*. Courier Corporation, 2013.

[25] E. W. Weisstein, "Finite field," *MathWorldA Wolfram Web Resource*, vol. 2, pp. 1999–2009, 1999.

[26] C. J. Benvenuto, "Galois field in cryptography," *University of Washington*, vol. 1, no. 1, pp. 1–11, 2012.

[27] A. Menezes, "An elementary introduction to hyperelliptic curves [ ]/menezes a., wu y., zuccherato r," tech. rep., Published as Technical Report CORR 96-19 Department of C&O University of , 1996.

[28] T. Wollinger, J. Pelzl, and C. Paar, "Cantor versus harley: optimization and analysis of explicit formulae for hyperelliptic curve cryptosystems," *IEEE Transactions on Computers*, vol. 54, no. 7, pp. 861–872, 2005.

[29] D. G. Cantor, "Computing in the jacobian of a hyperelliptic curve," *Mathematics of computation*, vol. 48, no. 177, pp. 95–101, 1987.

[30] L. C. Washington, *Elliptic curves: number theory and cryptography*. CRC press, 2008.

[31] S. Andalib and S. Azad, "The rsa algorithm," *Practical Cryptography: Algorithms and Implementations Using C+*, 2014.

[32] W. Diffie and M. E. Hellman, "New directions in cryptography," in *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, pp. 365–390, 2022.

[33] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. New York, NY: John Wiley & Sons, 2nd ed., 1996.

[34] W. Stallings, *Cryptography and Network Security: Principles and Practice*. Upper Saddle River, NJ: Pearson, 5th ed., 2010.

[35] Y. Zheng, "Digital signcryption or how to achieve cost(signature & encryption) cost(signature) + cost(encryption)," in *Advances in Cryptology — CRYPTO '97* (B. S. Kaliski, ed.), (Berlin, Heidelberg), pp. 165–179, Springer Berlin Heidelberg, 1997.

[36] H. Guo and L. Deng, "An identity based proxy signcryption scheme without pairings.," *Int. J. Netw. Secur.*, vol. 22, no. 4, pp. 561–568, 2020.

[37] R. I. Abdelfatah, "A novel proxy signcryption scheme and its elliptic curve variant," *International Journal of Computer Applications*, vol. 165, no. 2, pp. 36–43, 2017.