UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA
SCHOOL OF COMPUTER SCIENCE TECHNOLOGY

DESIGN AND ANALYSIS OF PUBLIC KEY CRYPTOGRAPHY (PKC)

FOUNDATION OF CRYPTOGRAPHY

SUBMITTED TO

PROF. LI FANGEN

BY

YEBOAH-DUAKO ELVIS

yeboahduako770@gmail.com

(ID: 202124080120)

JUNE, 2022

ABSTRACT

Public Key Cryptosystems (PKC) have been a great advancement to modern cryptography and security in general. With its key-pair feature, along with the defined hard problems, PKC not only solved security issues related to confidentially but also, it has helped to address issues such as authentication and non-repudiation. In this report, I design and discuss a couple of Public Key Cryptosystems that seeks to provide both authentication and confidentiality as opposed to the original encryption schemes which lack authentication.

Table of Contents

1.0 Background and Significance

1.1 Overview of Cryptography

Cryptography has been used for thousands of years to help to provide confidential communications between mutually trusted parties. Its fundamental objective is to enable two people, usually referred to as Alice and Bob, to communicate over an insecure channel in such a way that an opponent, Eve, cannot understand what is being said. This is achieved by the encryption of plaintexts with a predetermined secret key into ciphertexts which are then transmitted through the unsecure channel with the assurance that, Eve, upon seeing the ciphertext in the channel by eavesdropping, cannot determine what the plaintext was; but Bob, who knows the encryption key, can decrypt the ciphertext and reconstruct the plaintext.

The secret key(s) that are used to encrypt and decrypt a message segregates cryptosystems into two: Symmetric Key Cryptosystems and Asymmetric Key Cryptosystems. Symmetric cryptosystems use a single key for both encryption and decryption of messages. While this process is effective fast and more preferred, key agreement procedures are impractical over the insecure channel and hence poses a major setback. Often, parties could meet physically to agree on a key but where distance is a barrier, it is quite hard, if not impossible, to safely share the secret keys without Eve getting access to it. Also, this system provided not provide security against non-repudiation since it is a single general secret key being used.

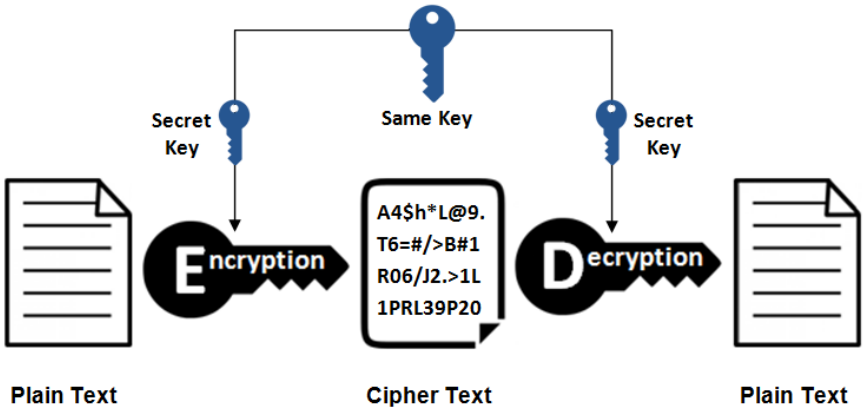

Fig 1.1: Symmetric Encryption

Asymmetric Cryptosystems, also known as Public Key Cryptosystems, on the other hand use key pairs: a public key, which may be known by anybody, is used for encryption (or signature verification), and a private key, known only to the recipient, is used for decryption (or signing of digital certificates). These keys are complimentary yet it is infeasible to determine the private key from the public key. Hence, those who encrypt messages or verify signatures cannot decrypt those messages or create the signatures.

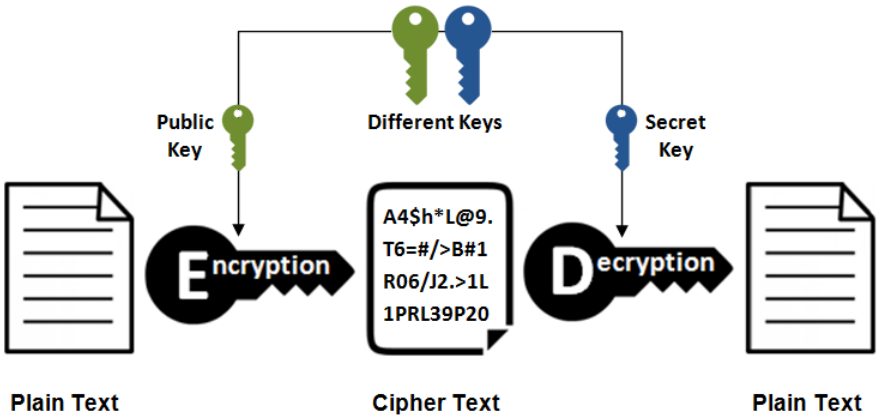

Fig. 1.2: Asymmetric Encryption/ Public Key Cryptography

1.2 Analogy: How Public key Cryptography work

The Mailbox analogy is the perfect concept that illustrates how Public Key Cryptosystems function. One party of the communication, say Bob, has a locked mailbox. Only Bob has the keys to this mailbox. Another party, say Alice can insert a letter into the box at any time but cannot unlock it to take the letter out after. Only Bob can do that since he owns the key.

1.3 Requirements of a Public Key Cryptosystem

Three main conditions must always be met by any public key cryptosystem:

1. It must be computationally easy to encrypt or decrypt the message given the appropriate key.
2. It must be computationally infeasible to derive the private key from the public key.
3. It must be computationally infeasible to determine the private keys from chosen plaintext attack

To meet these requirements, we need a trapdoor one way function such that:

$y = f_k(x)$ is computationally easy if k and x are known,

$x = f_k^{-1}(y)$ is computationally easy if k and y are know, And finally,

$x = f_k^{-1}(y)$ is computationally infeasible if y is known but k is unkown.

1.4 Advantages / Significance of Public Key Cryptography

The Public Key cryptography was designed to address two key issues:

1. Key Distribution: How to have secure communications in general without having to trust a Key Distribution Center (KDC) with your key.
2. Digital Signatures: How to verify a message comes intact from the claimed sender

1.5 Applications of Public Key Cryptography

Public key cryptosystems can be classified broadly into three categories:

1. Encryption / Decryption Cryptosystems which mainly provide confidentiality (privacy)
2. Digital Signature cryptosystems which provide authentication to messages
3. Key Exchange cryptosystems which are used in negotiating session keys

Some cryptosystems are suitable for all uses whereas others are strictly specific to one

Table 1.1: Summary of Public Key Cryptosystem

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
El Gamal	Yes	Yes	Yes
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

1.6 Security of Public Key Schemes

Like symmetric key cryptosystems, a brute force exhaustive search attack is always theoretically possible. Nevertheless, difficulty of such an attack increases as the key length increases. Hence, the goal is to use large key sizes such that the computational difficulty increases making it impossible to solve with polynomial time. Security therefore relies on a large enough difference in difficulty between easy (en/decrypt) and hard (cryptanalysis) problems. The hard problem, generally is known, but it is made hard enough to be impractical to break. These, together with the large numbers involved in the cryptosystems make the Public Key Cryptography relatively slower when compared with the symmetric or private key schemes.

1.6.1 PKC Hard problems

The Public Key Cryptography hard problems form the foundation of PKC's security. They are problems that are believed to be computationally very difficult given the only the ciphertext and the available public keys and yet very easy when given the necessary private key(s). I discuss briefly some of these PKC hard problems below:

1. Discrete Log Problem
   The discrete log problem simply requires one to compute the value of $\boldsymbol{b}$ from $y = a^b \ mod \ p$ when the value of y, a and p are publicly available. This is believed to be a hard problem that has no efficiently established algorithm to calculate b when the a carefully selected group G, is defined for the cryptosystem. Based on this problem, other hard problems have been defined. They include:
   a. Computational Diffie-Hellman (CDH), where an attacker is tasked to compute b
   b. Decisional Diffie-Hellman (DDH), where an attacker is tasked to tell if a given value, say k is equal to b.
   c. Elliptic Curve Discrete Log Problem where the discrete log problem is defined over finite fields based on the algebraic structure of elliptic curves.

Example of PKC's that use the Discrete Log problem include: RSA, Diffie-Hellman Key Exchange, El Gamal cryptosystem, Elliptic Curve Cryptography, and the Crammer Shoup Encryption.

2. Integer Factorization Problem
   This problem requires one to decompose a composite number into a product of small integers. In cryptography, mostly, these factors are restricted to prime numbers in which case it is referred to as Prime Factorization problem. For instance, given the value of N, find two small prime numbers p and q such that $N = p \times q$. When the numbers are sufficiently large, it is believed that there are no known efficient non-quantum integer factorization algorithm to decompose it.

   Example of PKC's that use the Integer factorization hard problem include RSA.

1.7 Historical Background

Public Key cryptography emerged in the 1970's. The first use of public key cryptography was the Diffie-Hellman key exchange protocol which was designed by Diffie and Hellman in 1976. This cryptosystem was influence by Ralph Merkle's work on public key distribution, the "public key-agreement technique" which later became known as Merkle's Puzzle. Though, it was invented in 1974, it was published in 1978 and given that it was the first of its kind, this makes asymmetric encryption a rather new field in cryptography although cryptography itself dates back more than 2,000 years.

Over the years, several other public key cryptosystems have been developed, each with its own security and features. These include the RSA cryptosystem which was developed in 1978 by Rivest, Shamir, and Adleman is currently the most widely used cryptosystems. Also, the El-Gamal Cryptosystem which is a probabilistic cryptosystem was designed by Taher El Gamal in 1985.

Though public key cryptography comes with great security, it's relatively slow nature in encryption has relegated it into a mere key establishment protocol where it is used to securely negotiate symmetric encryption keys to allow for faster encryption and decryption services. It also provides authentication services by means of a digital signature.

2.0 Design and Analysis of public Key Authenticated Encryption Schemes

In this section, I design and analyze the two public key encryptions schemes (The El-Gamal Encryption and the RSA Encryption). The original encryption schemes primarily provide confidentially and not authentication which is a great security challenge. To address this issue, I explore and integrate digital signatures to the original encryptions as presented in the subsequent sections.

2.1 Authenticated El Gamal Encryption

The El Gamal cryptosystem is a public-key probabilistic cryptosystem that is based on the discrete logarithm hard problem. It uses asymmetric key encryption for communicating between two parties and encrypting the message. This cryptosystem is based on the difficulty of finding discrete logarithm in a cyclic group such that given g^a and g^k, it is extremely difficult to compute g^{ak}. It was developed by Taher El Gamal in 1985.

2.1.1 The Design: Block Diagram

This design is implemented using both the El Gamal encryption scheme and the El Gamal digital signature. These two principles are integrated in a unique way as shown in figure … to ensure that the El Gamal cryptosystem provides both confidentiality and authentication.

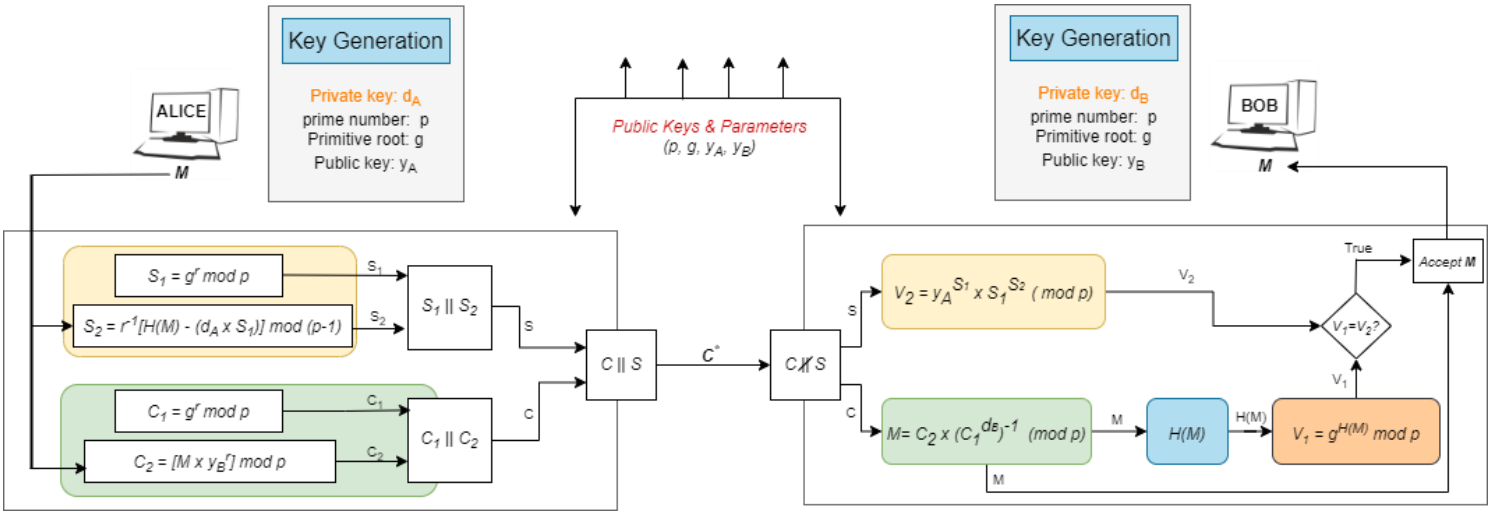

Fig. 2.1: Authenticated El Gamal Encryption

2.1.2 The procedure: Algorithm Explained

The Authenticated Encryption algorithm as shown in fig. 2.1 above is made up of 3 main stages: The Key Generation stage, the Encryption & Signature stage, and the Decryption & Signature verification stage. These

Key Generation

1. The communicating parties (Alice and Bob) publicly agree on the El Gamal cryptosystem parameters such as a cyclic group $\boldsymbol{G}$, a large prime number $\boldsymbol{p}$ and a generator (primitive root) $\boldsymbol{g}$.

2. From p and g, Alice and Bob each compute their public key, $\boldsymbol{y_x}$ using the formular $y_x = g^{d_x} \, mod \, p$, where $\boldsymbol{d_x}$ is a private key such that $\gcd(d_x, p) = 1$

3. They publish their public keys (G, p, g, y_x)

Table 2.1: Key Generation for the Authenticated El Gamal Encryption Scheme

	Alice	Bob
Prime Number	p	
Primitive Root	g	
Private key	d_A	d_B
Public key	$y_A = g^{d_A} \, mod \, p$	$y_B = g^{d_B} \, mod \, p$

Encryption and Signature

Once the both Alice and Bob have generated their keys, encryption and signing of the message can begin. To make this process effective. I make an important that the public keys of both Alice and Bob is being made available and distributed to them using a trusted public key infrastructure (PKI) system with valid certificates. By so doing, Alice and Bob have a certain level of confidence that they are communicating with the right person.

The steps to the encryption and signature are as follows:

A. *Signature*
Using the flow of message M from Alice to Bob, Alice signs the message with her secret key by:

1. Compute a random integer r (also known as the ephemeral key) in the interval $(1, p)$.

2. Compute:

$$S_1 = g^r \, mod \, p$$

$$S_2 = r^{-1} \, [H(M) - (d_A \times S_1)] \, mod \, p - 1)$$

Where $H(M)$ is the message digest of the message M

3. The Signature, S = (S_1, S_2) as shown in the block diagram (the pipe, '||' implies concatenation).

B. Encryption

1. Using the message M, the same random number $\boldsymbol{r}$ generated for the signature, and Bob's public key $\boldsymbol{y_B}$, Alice then computes the ciphertexts as follows:

$$C_1 = g^r \ mod\ p$$
$$C_2 = (M \times {y_B}^r)\ mod\ p$$

2. The final Ciphertext $C = (C_1, C_2)$

C. Transmission

Alice then sends the signed message, $C^* = (C, S)$ to Bob

Decryption and Verification

A. Decryption

Bob uses his private key $\boldsymbol{d_B}$ to decrypt the ciphertext, $C = (C_1, C_2)$ as follows:

$$M = \left[C_2 \times \left(C_1^d\right)^{-1}\right]\ mod\ p$$

B. Verifying Signature

1. Bob finds the message digest, $H(M)$ using decrypted message M

2. Using the Signature, $S = (S_1, S_2)$ and Alice's public key $\boldsymbol{y_A}$, Bob computes V_1 and V_2:

$$V_1 = \ g^{H(M)}\ mod\ p$$
$$V_2 = \left[{y_A}^{S_1} \times {S_1}^{S_2}\right]\ mod\ p$$

3. Bob compares $V_1$and V_2. If $V_1 \equiv V_2$: He accept Mas an authentic message from Alice. Else, he rejects M.

2.1.3 Proof of Correctness

The proof of correctness is shown for both encryption and authentication (digital signature).

a. Encryption Correctness

From:

$$C_1 = g^r\ mod\ p \qquad (1)$$
$$C_2 = (M \times {y_B}^r)\ mod\ p \qquad (2)$$
$$M = C_2 \times \left(C_1^{d_B}\right)^{-1}\ mod\ p \qquad (3)$$

Substitute $Eqn.(1)$ and $Eqn.(2)$ into $Eqn.(3)$

$$\Rightarrow M = (M \times {y_B}^r) \times \left(g^{rd_B}\right)^{-1} mod\ p$$

$but\ y_B = g^{d_B}\ mod\ p$

$$\Rightarrow\ M = \left(M \times \left(g^{d_B}\right)^r\right) \times\ \left(g^{rd_B}\right)^{-1}\ mod\ p$$

$$M = [M \times g^{rd_B} \times g^{-rd_B}]\ mod\ \ p = M$$

$$\therefore\ M = M$$

Hence, Encryption correctness proved!

b. Signature Correctness

Let:

$$V_1 = g^{H(M)} mod\ p \qquad (1)$$

And:

$$V_2 = {y_A}^{S_1} \times S_1^{S_2}\ \ modp \qquad (2)$$

but

$$y_A = g^{d_A}\ \ mod\ p\ ;\ \ S_1 = g^r\ mod\ p\ \ \ ;S_2 = r^{-1}[H(M) - d_A S_1] mod\ p$$

$$V_2 = \left[\left(g^{d_A}\right)^{S_1} \times (g^r)^{S_2}\right]\ mod\ p$$

$$V_2 = \left(g^{d_A S_1} \times g^{rS_2}\right) mod\ p$$

$$V_2 = g^{d_A S_1 + rS_2}\ \ mod\ p \qquad (3)$$

Focusing on the Exponent:

$$\Rightarrow d_A S_1 + rS_2\ \ (mod\ p) = d_A(g^r) + r(r^{-1}[H(M) - d_A S_1])\ mod\ p$$

$$= d_A(g^r) + r(r^{-1}[H(M) - d_A(g^r)])\ mod\ p$$

$$= d_A(g^r) + H(M) - d_A(g^r) mod\ p$$

$$\therefore d_A S_1 + rS_2\ = H(M) \qquad (4)$$

Substituting (4) into (3)

$$V_2 = g^{H(M)}\ mod\ p$$

By Comparing $Eqn.(1)$ and $Eqn.(5)$:

$$\Rightarrow V_1 = V_2$$

Hence, Signature correctness proved!

2.1.4 Performance Analysis

The designed authenticated El Gamal encryption scheme, like the original El Gamal encryption is probabilistic in nature and hence a single plaintext can be encrypted to many different ciphertexts due to the random number (ephemeral key). Similarly, the signature varies for a single plaintext message each time it is signed. Whereas this is good security-wise, performance-wise, the scheme results in a ciphertext which is twice the size of the plaintext. Also, the signature gives two outputs of same length and hence this scheme maps plaintext to ciphertext in a ratio of 1:3. However, by using the message digest for the signature instead of the message itself, this ratio was reduced to approximately 1:2.

In terms of computation costs, this authenticated El Gamal encryption scheme requires four (4) exponentiations: two each for both the encryption and the signature computations. However, these exponentiations are independent of the message and can be computed in advance if necessary to speed up the process. Decryption on the other hand, requires one exponentiation and one computation of a group inverse, both of which can be easily combined into a single exponentiation. And finally, the signature verification requires 3 exponentiations. In total, the computation cost for this designed authenticated El Gamal encryption scheme is about 8 exponentiations and one group inverse calculation.

2.1.5 Security Analysis

The key point for the security of this authenticated El Gamal encryption scheme resides in the group G and its order. As long as the generated pair of keys (public and private) effectively map the message into the chosen group G such that the Decisional Diffie-Hellman (DDH) assumption holds, then the cryptosystem achieves Semantic security.

Moreover, as long as the discrete logarithm problem holds, the cryptosystem is safe. That is, the authenticated El Gamal Cryptosystem will be secure if an attacker, Eve cannot compute the value $\boldsymbol{d}$ from $\boldsymbol{y}$ given that $y = g^d \ mod \ p$. If Eve can compute d, then she can decrypt ciphertexts exactly as the legitimate receiver, Bob, does. Hence, a necessary condition for this and all El Gamal Cryptosystem to be secure is that the Discrete Logarithm problem in Zp^* is infeasible. This is generally regarded as being the case if p is carefully chosen and g is a primitive element modulo p. In particular, there is no known polynomial-time algorithm for this version of the Discrete Logarithm problem. However, for a secure setting, it is recommended that p should have at least 2048 bits in its binary representation, and $p - 1$ should have at least one “large” prime factor.

In terms of weakness, the designed El Gamal encryption scheme is unconditionally malleable, and as a result, it is not secure against chosen ciphertext attacks. For instance, given an encryption (C_1, C_2) of a message m, one can easily construct a valid encryption $(C_1, 2C_2)$ of the message $2m$. To achieve security against the chosen-ciphertext attack, the scheme will require further modification as seen in the Cramer-Shoup cryptosystem.

Finally, one key weakness of this authenticated El-Gamal encryption scheme is seen when the random number r is reused. Reusing r exposes both the message and the private key of the signature signee to Eve. Assuming r has been reused for encrypting and signing two messages m and m^*, Let’s consider the two cases below:

Case 1: Messages are exposed

Let m be encrypted as $C_m = (C_1, C_2)$ such that:

$$C_1 = g^r \ mod\ p$$

$$C_2 = (m \times {y_B}^r)\ mod\ p$$

And let m^* be encrypted as $C_{m^*} = (C_1^*, C_2^*)$ such that:

$$C_1^* = g^r \ mod\ p$$

$$C_2^* = (m^* \times {y_B}^r)\ mod\ p$$

Now:

$$\Rightarrow \ C_2 \times ({C_2}^*)^{-1} = [(m \times {y_B}^r) \times (m^* \times {y_B}^r)^{-1}] mod\ p$$

$$C_m \times (C_{m^*})^{-1} = m \times (m^*)^{-1}$$

Assuming Eve, the attacker knows either m or m^*, she can find the other message as follows:

Say Eve knows m^*, She can obtain m as follows:

$$m^* \ [C_m \times (C_{m^*})^{-1}] \ = m^* \ [m \times (m^*)^{-1}] = m$$

Hence, when the secret key is reused, just by knowing the content of one message, it is easy to know the content of other messages!

Case 2: Private key is exposed

Let the signature of m and m^* be S and S^* respectively such that:

$$S = (s_1, s_2) \text{ and } S^* = (s_1^*, s_2^*)$$

$$r(s_2 - s_2^*) \equiv (m - m^*) mod (p-1)$$

$$r = (s_2 - {s_2}^*)^{-1} \times (m - m^*)\ mod (p-1)$$

Therefore, if Eve obtains m and m^* as described in case 1 or through any means, with access to the corresponding signatures such that $\gcd(s_2 - {s_2}^*, p-1) = 1$ and the $\gcd(s_1, p-1) = 1$, then r can be calculated from the above expression. By obtaining r, Eve can calculate the secret key of the Alice (the signee) d_A as shown below:

$from$

$$s_2 = r^{-1} \ [H(m) - (d_A \times s_1)]\ mod\ p-1)$$

$making\ d_A\ the\ subject$:

$$s_2 = [r^{-1}H(m) - r^{-1}d_A s_1] \; mod(p-1)$$

$$r^{-1}d_A s_1 = (r^{-1}H(m) - s_2) \; mod(p-1)$$

$$\Rightarrow d_A = (r^{-1}s_1)^{-1} \times (r^{-1}(H(m) - s_2) \; mod(p-1)$$

$$\therefore d_A = s_1^{-1} \times [r \times s_1 - H(m)] \; mod(p-1)$$

Hence, the attacker has obtained Alice's secret key!

2.2 Authenticated RSA Encryption Scheme

RSA is a number-theoretic-based public-key cryptosystem developed by Ronald **R**ivest, Adi **S**hamir, and Leonard **A**dleman in 1977. It is used to encrypt data for secure transmissions and to authenticate data. The basic operation of RSA is modular exponentiation. Decryption takes place by finding the modular inverse. RSA is an asymmetric cryptosystem hence its algorithm makes use of two complementary keys: The Private key and the Public Key. As the name suggests, the Private key is to be kept secret at all times, and it is used mainly for decryption of data and for signing digital signatures whereas the public key has little to no protection and is open to all parties, adversaries inclusive. It is used for data encryption and digital signature authentication. Though RSA is relatively strong security-wise compared to other symmetric cryptosystems, it requires longer computation time and as such, it is mostly not used to encrypt real data. Instead, it is used to encrypt and securely negotiate secret symmetric keys that would be used for bulk data encryption and decryption by the two parties involved.

By default, the RSA encryption process comes with no authentication and hence, forgery and man in the middle attacks are prevalent. In this report, I provide a design that incorporates authentication in the RS encryption process by complementing the original algorithm with the RSA digital signature.

2.2.1Mode of Operation

Consider the diagram below for the Illustration of how RSA works. Alice and Bob want to communicate securely without any third-party eavesdropping on their conversation. The entire algorithm can be broken down into 3 phases:

- Key Generation and Distribution
- Plaintext Encryption and Signature
- Ciphertext Decryption and Signature verification

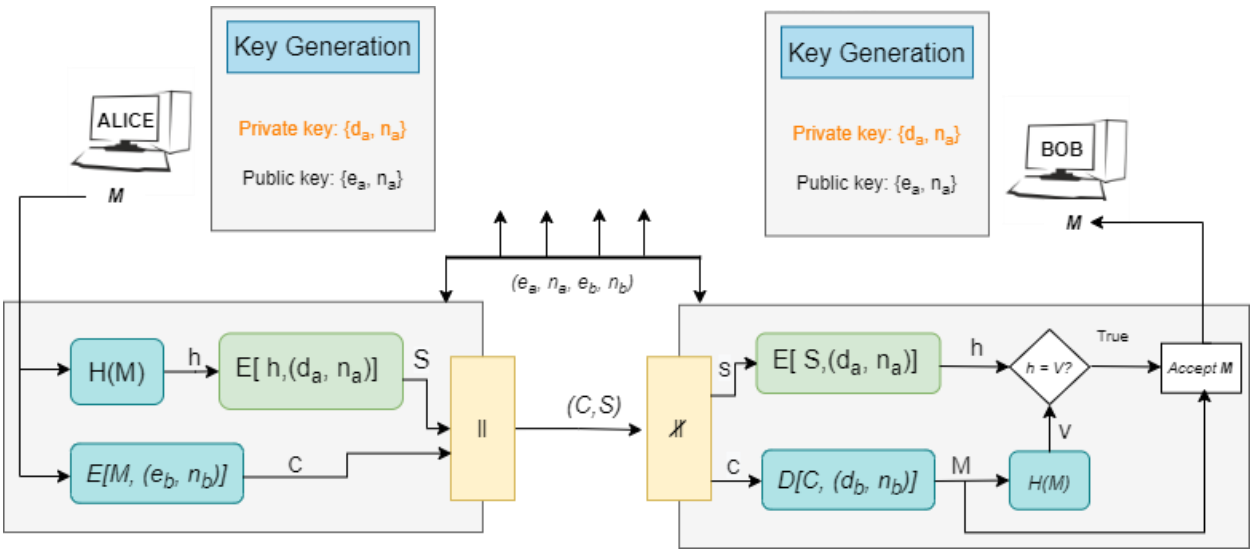

Fig 2.2: Authenticated RSA Encryption Scheme

1. ***Key Generation and Distribution***

Both Alice and Bob go through their unique processes of generating their key pairs as shown below:

Steps:

I. Bob Selects 2 large prime numbers (512 – 1024 bits), say $\boldsymbol{p_b}$ and $\boldsymbol{q_b}$.
The larger the prime numbers, the more secure the generated keys due to the seeming difficulty in factoring larger prime numbers

II. Bob computes $\boldsymbol{n_b}$, such that n_b is the product of the two chosen prime numbers p_b and q_b

$$\boldsymbol{n_b = p_b \times q_b}$$

III. Bob calculates the totient function, $\boldsymbol{\phi(n_b)}$ such that:

$$\phi(n_b) = (p_b - 1)\,(q_b - 1)$$

IV. Bob then selects an integer $\boldsymbol{d_b}$, such that d_b is co-prime to $\phi(n_b)$, $(ie.\, gcd\,(d_b, \phi(n_b) = 1)$ and also d_b must lie between 1 and $\phi(n_b)$. That is:

$$1 < d_b < \phi(n_b)$$

V. Finally, Bob computes the multiplicative inverse of d_b modulo $\phi(n_b)$ which is denoted by $\boldsymbol{e}_b$ such that $d_b e_b \equiv 1\,(mod \phi(n_b))$.

VI. Bob obtains a certificate from a trusted Certificate Authority (CA) for $(\boldsymbol{e_b, n_b})$ as his ***public key***. Alice and everyone have access to Bob's public key.

VII. Bob keeps $(d_b, p_b, q_b, \phi(n_b))$ as secret and uses $(\boldsymbol{d_b}, \boldsymbol{n_b})$ as his ***private key.***

Note: Alice goes through a similar process to obtain her parameters as follows:

$$Secret\ parameters: \{d_a, p_a, q_a, \phi(n_a)\}$$
$$Private\ key: \{d_a, n_a\}$$
$$Public\ key: \{e_a, n_a\}$$

Table 2.2: Summary of RSA Generated keys

	Alice	Bob
Secret Parameters	$\{d_a, p_a, q_a, \phi(n_a)\}$	$\{d_b, p_b, q_b, \phi(n_b)\}$
Private key	$\{d_a, n_a\}$	$\{d_b, n_b\}$
Public key	$\{e_a, n_a\}$	$\{e_b, n_b\}$

2. *Plaintext Encryption and Signature*

Encryption

a. Given a plaintext M, M is first encoded as a word over the alphabet $\{0,1, \dots, 9\}$, then divided into blocks of length $i-1$, where $10^{i-1} < n < 10^i$.

b. Alice computes the Ciphertext, C using modular exponentiation from the blocks of M and Bob's public key (e_b, n_b) as follows:

$$\boldsymbol{C = E[M, (e_b, n_b)]}$$
$$\therefore\ \boldsymbol{C = M^{e_b} \bmod n_b}$$

Signature

a. Alice finds the Hash of the message M, say using an agreed hash function such as SHA-256 to obtain $H(M)$

b. Alice then signs $H(M)$ by encrypting it with her private key as follows:

$$\boldsymbol{S = E[H(M), (e_a, n_a)]}$$
$$\therefore \boldsymbol{S = H(M)^{e_a} \bmod n_a}$$

Finally, Alice sends $(\boldsymbol{C}, \boldsymbol{S})$ over the unsecure channel to Bob.

3. Ciphertext Decryption and Signature Verification

Decryption:

1. Upon Reception of the ciphertext, Bob uses his private key (d_b, n_b) to generate the original plaintext using modular exponentiation as follows:

$$\boldsymbol{M} = \boldsymbol{D}[\boldsymbol{C}, (\boldsymbol{d_b}, \boldsymbol{n_b})]$$
$$\therefore \quad \boldsymbol{M} = \boldsymbol{C}^{\boldsymbol{d_b}} \boldsymbol{mod}\ \boldsymbol{n_b}$$

2. Bob finds the Hash of M using the agreed hashing algorithm, SHA-256 as follows:

$$\boldsymbol{V} = \boldsymbol{H}(\boldsymbol{M})$$

Verification of Signature:

1. Bob decrypts the signature S using Alice's public key (e_a, n_a) as follows:

$$\boldsymbol{h} = \boldsymbol{H}(\boldsymbol{M}) = \boldsymbol{S}^{\boldsymbol{e_a}} \boldsymbol{mod}\ \boldsymbol{n_a}$$

2. Bob compares $\boldsymbol{V}$ and $\boldsymbol{h}$

$$if\ V \equiv h$$

Bob accepts $\boldsymbol{M}$ as a genuine message from Alice, Else $\boldsymbol{M}$ is rejected!

2.2.2 Proof of Correctness

Both the RSA Signature and Encryption processes involve simple encryption with a key. The only difference is that encryption is done using the recipient's public key (Bob) whereas the sender's private key (Alice) is used for the signature.

Since both signature and encryption operate on the basis of RSA encryption, I show only the proof of correctness for the encryption part. The same can be done to prove the signature.

Given that:

$$C = M^{e_b} \bmod n_b \qquad ; \qquad n_b = p_b \times q_b \qquad ; \qquad M = C^{d_b} \bmod n_b$$

The private key, $\boldsymbol{d_b}$ and the public key, $\boldsymbol{e_b}$ are related as follows:

$$d_b e_b \equiv 1\ (mod \phi(n_b))$$

⇨ There exists an integer $\boldsymbol{j}$, $(j \in N_b)$ such that $d_b e_b = j\phi(n_b) + 1$

Case 1: *Neither p_b nor q_b divides M*

$$\Rightarrow \quad gcd(n_b, M) = 1$$

By Euler's Totient Theorem,

$$\Rightarrow \boldsymbol{C^{d_b}} = (M^{e_b})^{d_b} = M^{e_b d_b}$$

$$= M^{j\phi(n_b)} + 1 \equiv \boldsymbol{M}\ (mod\ n_b)$$

Case 2: *Exactly one of p_b, q_b divides M (say p)*

$$\Rightarrow M^{e_b d_b} = M\ (mod\ p_b)$$

By Fermat's Little theorem, $M^{q_b-1} \equiv 1\ (mod\ q_b)$

$$\Rightarrow M^{q_b-1} \equiv 1\ (mod\ q_b)$$

$$\Rightarrow M^{\phi(n_b)} \equiv 1\ (mod\ q_b)$$

$$\Rightarrow M^{j\phi(n_b)} \equiv 1\ (mod\ q_b)$$

$$\Rightarrow M^{e_b d_b} \equiv M\ (mod\ q_b)$$

$$\therefore \boldsymbol{M} \equiv M^{e_b d_b} \equiv \boldsymbol{C^{d_b}} (mod\ n_b)$$

Case 3: *Both p_b and q_b divide M*

Not Possible as long as $\boldsymbol{M} < \boldsymbol{n_b}$

2.2.3 Security and Performance Analysis

The security of the designed RSA cryptosystem strongly resides in both integer factorization problem and the RSA problem. On the assumption that both of these problems are hard, i.e., no efficient algorithm exists for solving them, full decryption of an RSA ciphertext is thought to be infeasible. To provide security against partial decryption, a secure padding scheme may be required.

The RSA problem is defined as the task of recovering a value m such that $c \equiv m^e\ (mod\ n)$, where (n, e) is an RSA public key and c is an RSA ciphertext. At the moment, factoring the modulus n is the most promising approach to solving the RSA problem. An attacker who can recover prime factors can compute the secret exponent d from a public key (n, e) and then decrypt c using the standard procedure. To accomplish this, an attacker divides n into p and q and computes $lcm(p-1, q-1)$, which allows d to be calculated from e. There has yet to be discovered a polynomial-time method for factoring large integers on a classical computer, but this has not been proven.

The largest publicly known factored RSA number had 829 bits as of 2020. (250 decimal digits, RSA-250). Its factorization took approximately 2700 CPU years using a cutting-edge distributed implementation. RSA keys are typically 1024 to 4096 bits long in practice. In 2003, RSA Security predicted that 1024-bit keys

would be crackable by 2010. It is unknown whether such keys can be cracked as of 2021, but minimum recommendations have increased to at least 2048 bits. Outside of quantum computing, it is generally assumed that RSA is secure if n is large enough.

If n is 300 bits or less, it can be factored in a few hours on a personal computer using freely available software. Keys of 512 bits were shown to be practically breakable in 1999, when RSA-155 was factored using hundreds of computers, and these are now factored in a matter of weeks using common hardware. In 2011, exploits involving 512-bit code-signing certificates that may have been factored were reported. TWIRL, a theoretical hardware device described by Shamir and Tromer in 2003, called the security of 1024-bit keys into question.

3.0 Design and Analysis of an Authenticated Three-Party Diffie Hellman Key Exchange Protocol

3.1 Overview of Diffie-Hellman Key Exchange Protocol

The Diffie Hellman Protocol, developed by Whitefield Diffie and Martin Hellman in 1976 is a public key cryptosystem that enables two or more parties to securely agree on a shared secret key via an unsecured channel. It is mostly used to assist in secret key generation for symmetric key encryption and decryption The Diffie-Helman algorithm is best illustrated by the analogy of mixing colors, where it is assumed that separating the mixed colours back to the individual colour components is not possible or extremely difficult. In practice, the Diffie Hellman algorithm makes use of primitive roots and modular exponentiation operations to define its trap door functions (f_0 and f_1) which mimics the color mixing scenario. There are two global parameters (a large prime number, p and a generator, g) known publicly to all parties involved in the communication (adversaries inclusive). The generator is a primitive root modulo p. The trap door functions (f_0 and f_1) are defined below where a and b are secret keys of say Alice and Bob respectively.

As secure as the Diffie Hellman Protocol is from eavesdroppers, it suffers greatly from the Man in the middle attack where Eve, an attacker, intercepts all communications to one party say, Bob and as a result, is able to negotiate different shared keys with Bob and the other parties (Alice and Chris). Thereafter, she masquerades Bob and is able to encrypt and decrypt messages to and from Alice & Chris and further relay such messages either as they are or modified versions to Bob. See figure 3.1 Below.

3.2 Authenticated Three Party Diffie-Hellman Protocol

In this design of an authenticated 3 – party Diffie Hellman Protocol, I used the RSA digital signature to achieve authentication in the key exchange process which alleviates the problem of the Man in the middle (MITM) attack .

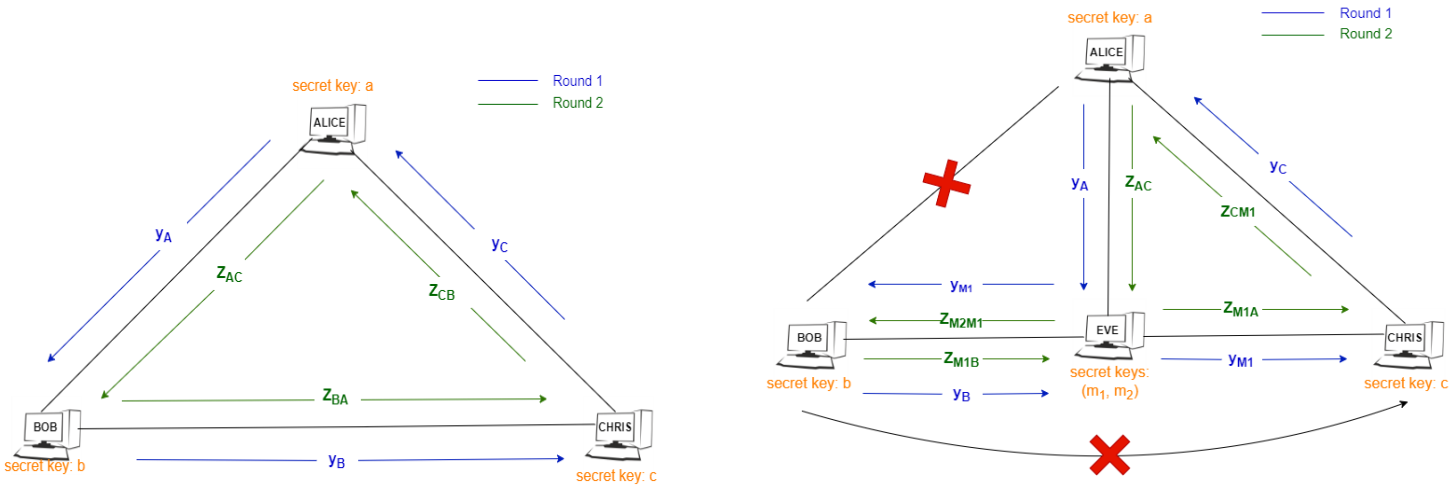

Fig. 3.1: 3-party Diffie-Hellman key exchange: (a) no MITM attack. (b) with MITM attack

3.2.1 Designed Algorithm Explained

The authenticated 3-party Diffie Hellman design is achieved in two main steps: the Key Generation for Authentication and the Key Exchange Process.

Key-Pair Generation for Authentication

In this design, the RSA digital signature was used to achieve authentication. A brief overview is presented below. Details of the RSA encryption and signature was discussed in the preceding question (Question 2).

Steps:

i. Bob Selects 2 large prime numbers (512 – 1024 bits), say $\boldsymbol{p_b}$ and $\boldsymbol{q_b}$.
The larger the prime numbers, the more secure the generated keys due to the seeming difficulty in factoring larger prime numbers

ii. Bob computes $\boldsymbol{N_b}$, such that N_b is the product of the two chosen prime numbers p_b and q_b

$$\boldsymbol{N_b = p_b \times q_b}$$

iii. Bob calculates the totient function, $\boldsymbol{\phi(N_b)}$ such that:

$$\phi(N_b) = (p_b - 1)\,(q_b - 1)$$

iv. Bob then selects an integer $\boldsymbol{d_b}$, such that d_b is co-prime to $\phi(Nn_b)$, $(ie.\, gcd\ (d_b, \phi(N_b) = 1)$ and also d_b must lie between 1 and $\phi(N_b)$. That is:
i. $1 < d_b < \phi(N_b)$

v. Finally, Bob computes the multiplicative inverse of d_b modulo $\phi(N_b)$ which is denoted by $\boldsymbol{e}_b$ such that $d_b e_b \equiv 1\ (mod \phi(N_b))$.

vi. Bob keeps $(d_b, p_b, q_b, \phi(N_b))$ as secret and uses $(\boldsymbol{d_b}, \boldsymbol{N_b})$ as his ***private key.***

vii. Bob obtains a certificate from a trusted Certificate Authority (CA) for $(\boldsymbol{e_b}, \boldsymbol{N_b})$ as his ***public key***. *Alice and everyone can now have access to Bob's public key.*

Note: Alice and Chris go through a similar process to obtain her parameters as shown in table 3.1:

A key assumption in this design is that: Each user's public key is well certified by a trusted Certificate Authority (CA) in a Public Key Infrastructure (PKI) system; hence the public keys of the parties are ***trusted*** to be authentic.

Table 3.1: RSA Generated keys for Authentication

	Alice	Bob	Chris
Public Keys	$\{e_a, N_a\}$	$\{e_b, N_b\}$	$\{e_c, N_c\}$
Private keys	$\{d_a, N_a\}$	$\{d_b, N_b\}$	$\{d_c, N_c\}$

The Key Exchange Process

Once the parties each have their pair of keys to be used for both signing and verifying signatures, the key exchange process can begin as shown in figure 3.2 below and subsequently explained in the follow us steps.

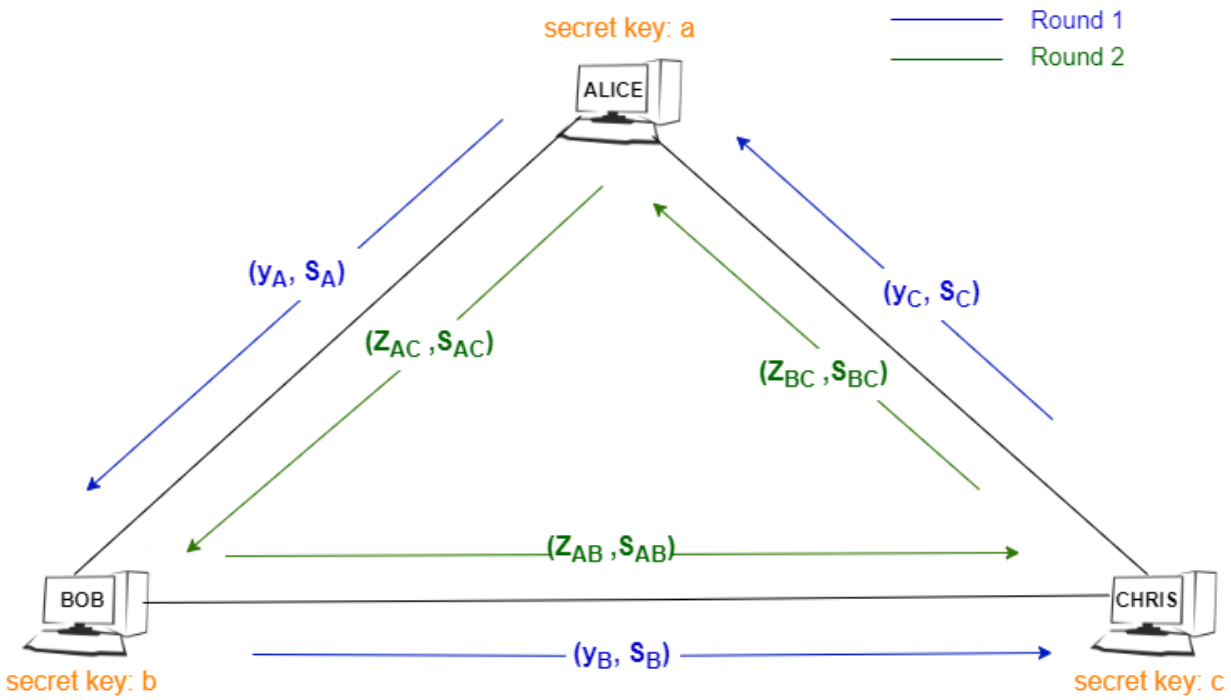

Fig. 3.2: Authenticated Diffie-Hellman Key Exchange Protocol

Steps:

1. $Alice\ (A)$, $Bob\ (B)$, and $Chris\ (C)$ agree on public parameter values for the generator (g) and the prime modulo (p).

2. Each party (A, B, C) selects their private key. $(i.e.\ A \rightarrow\ a,\ B \rightarrow b, and\ C \rightarrow c)$.

3. Each party (A, B, C) calculates a shared public variable y_X such that:

A	B	C
$\boldsymbol{y_A} = g^a\ mod\ p$	$\boldsymbol{y_B} = g^b\ mod\ p$	$\boldsymbol{y_C} = g^c\ mod\ p$

4. Each party signs the public variable, y_X using their private key to obtain S_X as follows:

A	B	C
$\boldsymbol{S_A} = {y_A}^{d_a} \ mod \ N_a$	$\boldsymbol{S_B} = {y_B}^{d_b} \ mod \ N_b$	$\boldsymbol{S_C} = {y_C}^{d_c} \ mod \ N_c$

5. First round key exchange: the communicating parties exchange S_X and y_X as follows:
   i. $Alice$ sends (S_A, y_A) to Bob,
   ii. Bob sends (S_B, y_B) to $Chris$, and
   iii. $Chris$ sends (S_C, y_C) to $Alice$.

6. Each party (A, B, C) verify the signed message using the corresponding party's public key as follows:

A	B	C
$\boldsymbol{y_C}' = {S_C}^{e_c} \ mod \ N_c$ Accept y_C : if $y_C = {y_C}'$	$\boldsymbol{y_A}' = {S_A}^{e_a} \ mod \ N_a$ Accept y_A : if $y_A = {y_A}'$	$\boldsymbol{y_B}' = {S_B}^{e_b} \ mod \ N_B$ Accept y_B : if $y_B = {y_B}'$

7. After verification, each party calculates the public variable z_{XY} such that:

A	B	C
$\boldsymbol{z_{AC}} = y_C^a \ mod \ p$ $i.e. \ z_{AC} = (g^c)^a \ mod \ p$	$\boldsymbol{z_{BA}} = y_A^b \ mod \ p$ $i.e. \ z_{BA} = (g^a)^b \ mod \ p$	$\boldsymbol{z_{CB}} = y_B^c \ mod \ p$ $i.e \ \ z_{CB} = (g^b)^c \ mod \ p$

8. Each party signs the public variable, Z_{XY} using their private key to obtain S_{XY} as follows:

A	B	C
$\boldsymbol{S_{AC}} = {z_{AC}}^{d_a} \ mod \ N_a$	$\boldsymbol{S_{BA}} = {z_{BA}}^{d_b} \ mod \ N_b$	$\boldsymbol{S_{CB}} = {z_{CB}}^{d_c} \ mod \ N_c$

9. Second Round key exchange: the communicating parties exchange S_{XY} and z_{XY} as follows:
   i. $Alice$ sends (S_{AC}, z_{AC}) to Bob,
   ii. Bob sends (S_{BA}, z_{BA}) to $Chris$, and
   iii. $Chris$ sends (S_{CB}, z_{CB}) to $Alice$.

10. Each party (A, B, C) verify the signed message using the corresponding party's public key as follows:

A	B	C
$\boldsymbol{z_{CB}}' = S_{CB}^{\ e_c} \ mod \ N_c$ Accept z_{CB} : if $z_{CB} = z_{CB}'$	$\boldsymbol{z_{AC}}' = S_{AC}^{\ e_a} \ mod \ N_a$ Accept z_{AC} : if $z_{AC} = z_{AC}'$	$\boldsymbol{z_{BA}}' = S_{BA}^{\ e_b} \ mod \ N_B$ Accept z_{BA} : if $z_{BA} = z_{BA}'$

11. Finally, they compute the secret key, k_{XYZ} as follows:

A	B	C
$\boldsymbol{k_{ACB}} = z_{CB}^{a} \ mod \ p$ $i.e. k_{ACB} = (((g^b)^c)^a) \ mod \ p$	$\boldsymbol{k_{BAC}} = z_{AC}^{b} \ mod \ p$ $i.e. \ k_{BAC} = (((g^c)^a)^b) \ mod \ p$	$\boldsymbol{k_{CBA}} = z_{BA}^{c} \ mod \ p$ $i.e. \ k_{CBA} = (((g^a)^b)^c) \ mod \ p$

By Comparing keys $\boldsymbol{k_{ACB}}$ from A, $\boldsymbol{k_{BAC}}$ from B, and $\boldsymbol{k_{CBA}}$ from C

$$\Longrightarrow ((g^b)^c)^a = ((g^c)^a)^b = ((g^a)^b)^c$$

$$\therefore \quad k_{ACB} = k_{BAC} = k_{CBA}$$

Hence

$$\boldsymbol{The\ shared\ secret\ key, k = k_{ACB} = k_{BAC} = k_{CBA}}$$

3.3 Security Analysis

When a large prime number $\boldsymbol{p}$, the group $\boldsymbol{G}$, and the generator $\boldsymbol{g}$ are well chosen, the protocol can be deemed secure against eavesdroppers. In such cases, the order of the group G, in particular, must be substantial, especially if the same group is used for a large quantity of traffic. To calculate the share secret key, $k = g^{abc}$, the eavesdropper must solve the Diffie–Hellman problem which is currently not feasible in polynomial time for groups with a sufficiently large enough order. An efficient discrete logarithm problem solution would make computing the secret keys $a, b \ or \ c$ and solving the Diffie–Hellman problem simple, thereby rendering this Diffie-Hellman and many other public key cryptosystems vulnerable. The Pohlig–Hellman algorithm is good algorithm that can be used to obtain $a, b \ or \ c$ when the order of G has a small prime factor. As a result, a Sophie Germain prime q, also known as a safe prime, is sometimes used to derive $p = 2q + 1$, since the order of G is therefore only divisible by 2 and q. Also g is then sometimes chosen to generate the order q subgroup of G, rather than G, so that the Legendre symbol of g^a never reveals the low order bit of a. A protocol using such a choice is for example IKEv2.

The generator, g is often a small integer, such as 2. Because of the discrete logarithm problem's random self-reducibility, a small g is as secure as any other generator in the same group. Also, it is much easier to eavesdrop if Alice and Bob use random number generators whose outputs are not completely random and can be predicted to some extent.

The Diffie–Hellman key exchange, as described in the original description, did not provide authentication of the communicating parties and is thus vulnerable to a man-in-the-middle attack as shown in figure 3.1 (b) where Eve, an active attacker carrying out the man-in-the-middle attack, may establish two distinct key exchanges, one with Alice and Chris, and another with Bob, effectively masquerading as Bob to Alice and Chris, and vice versa, allowing her to decrypt and then re-encrypt the messages passed between them. Every time Alice or Chris communicates with Bob, Eve must remain in the middle, actively decrypting and re-encrypting messages. If she is ever absent, Alice and Bob are made aware of her previous presence. They will be aware that someone in the channel has intercepted and decoded all of their private conversations. This improved algorithm solved this problem of Man in the Middle by authenticating the communication parties. With the standing assumption that the public keys of the communicating parties are certified by a trusted Public Key Infrastructure system. By so doing, the key establishment session is authenticated.

4.0 Design and Analysis of Identity-Based Encryption with Bilinear Pairing

Overview

Identity-based encryption (IBE) is a type of public-key encryption in which a user generates a public key from a known unique identifier (such as an email address or IP address), and a trusted third-party server calculates corresponding private key from the public key. There is therefore no need to distribute public keys before exchanging encrypted data. The sender can simply use the receiver's unique identifier to generate a public key and encrypt the data. With the assistance of a trusted third-party server – the private-key generator (PKG) – the receiver can generate the corresponding private key.

To use this encryption scheme, the PKG first publishes a master public key while keeping the corresponding master private key private. Any party can compute a public key corresponding to an identity given the master public key by combining the master public key with some known identity value (i.e., an email address). To obtain a corresponding private key, the owner of the identity used to generate the public key contacts the PKG, which generates the corresponding private key using its master private key.

ID-based encryption was proposed by Adi Shamir in 1984. He was however only able to give an instantiation of identity-based signatures. Identity-based encryption remained an open problem for many years. The pairing-based Boneh–Franklin scheme and Cocks's encryption scheme based on quadratic residues both solved the IBE problem in 2001.

In this report, I design an Identity-Based Encryption scheme using Bilinear pairing as inspired by the Boneh-Franklin scheme.

Bilinear Pairing

Let G_1 be a cyclic additive group generated by P, whose order is a prime q, and G_2 be a cyclic multiplicative group of the same order q. A bilinear pairing is a map $\hat{e}: G_1 \times G_1 \rightarrow G_2$ with the following properties:

i. Bilinearity: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in Z_q^*$.
ii. Non-degeneracy: There exists P and Q in G_1 such that $\hat{e}(P, Q) \neq 1$.
iii. Computability: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in G_1$

The IBE with Bilinear Pairing Algorithm

To realize the IBE with bilinear pairing, four major steps need to be followed: Setup, Extract, Encrypt and Decrypt.

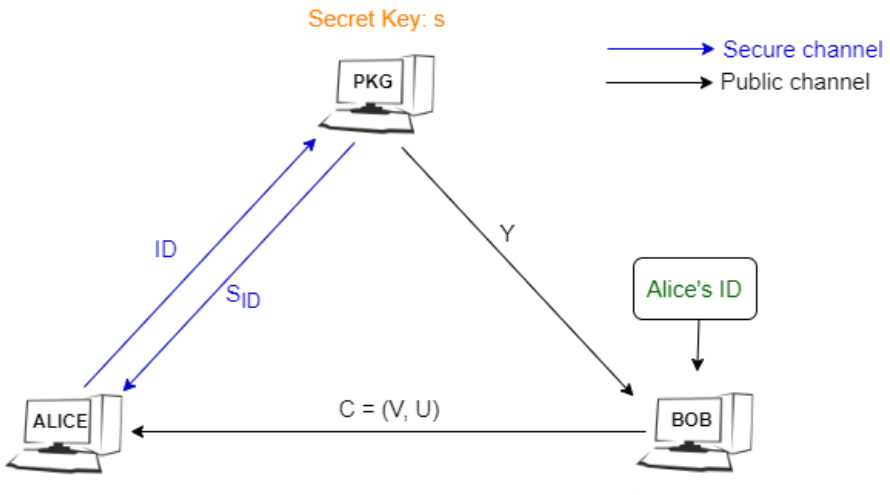

Fig. 4.1: IBE with Bilinear pairing

Setup

This algorithm is run by the PKG once to create the entire IBE environment. The master key is kept secret and used to derive user's private keys, while the system parameters are made public.

Steps:

1. Let G_1 be a cyclic additive group generated by P, whose order is a prime q, and G_2 be a cyclic multiplicative group of the same order q. ê: $G_1 \times G_1 \rightarrow G_2$ is a bilinear map.
2. Message space, M is $\{0,1\}^n$ and ciphertext space, C is $G_1^* \times \{0,1\}^n$
3. Define two hash functions $H_1: \{0,1\}^* \rightarrow G_1^*$ and $H_2: G_2 \rightarrow \{0,1\}^n$
4. The PKG chooses a master secret key $s \in Z_q^*$ randomly and computes the corresponding public key, $P_{pub} = sP$.
5. The PKG publishes parameters $Y = \{G_1, G_2, q, n, \hat{e}, P, P_{pub}, H_1, H_2\}$

Extract

Like the setup, this algorithm is run by the PKG. It is run each time a user requests for a private key. Given a user identity, $ID \in \{0,1\}^*$, the PKG computes the corresponding private key and sends it securely to the requesting user. Note that the verification of the authenticity of the requestor and the secure transport of the user private key are problems with which IBE protocols do not try to deal.

Steps:

1. PKG finds the digest of the user identity, ID and computes Q_{ID} such that $Q_{ID} = H_1(ID)$.
2. PKG computes the user private key, S_{ID} using his master key as:
$$S_{ID} = s \times Q_{ID}$$
3. PKG sends S_{ID} to the requesting user via a secured channel.

Encrypt:

This step is done by the communicating party say Alice or Bob. It takes a message $m \in M$ and an $ID \in \{0,1\}^*$ in addition with other key system parameters and outputs the encryption $c \in C$.

Steps:

1. Given message $m \in M$, choose $r \in Z_q^*$.
2. Compute the ciphertext as follows:

$$g_{ID} = \hat{e}(Q_{ID}, P_{pub}),$$
$$U = rP,$$
$$V = m \oplus H_2({g_{ID}}^r)$$

3. Send ciphertext, $c = (U, V)$ to Bob

Decrypt

This accepts a user secret key S_{ID}, the ciphertext, $c \in C$ as well as other system parameters, and returns $m \in M$ as follows:

$$m = V \oplus H_2(\hat{e}(S_{ID}, U))$$

Proof of Correctness

Given that:

$$c = (V, U) \;\; ; \;\; S_{ID} = s \times Q_{ID} \;\; ; \; and \;\; P_{pub} = sP:$$

From the Encryption:

$$V = m \oplus H_2({g_{ID}}^r)$$
$$\Rightarrow m = V \oplus H_2({g_{ID}}^r) \qquad (1)$$

From the Decryption:

$$m = V \oplus H_2(\hat{e}(S_{ID}, U)) \qquad (2)$$

By Comparison, $Eqn.(1) = Eqn.(2)$ if and only if ${g_{ID}}^r = \hat{e}(S_{ID}, U)$

Taking the R.H.S and by using consistency analysis:

$$\Rightarrow \hat{e}(S_{ID}, U) = \hat{e}(sQ_{ID}, rP)$$
$$= \hat{e}(Q_{ID}, srP)$$

$$= \hat{e}(Q_{ID}, P_{pub})^{r}$$

$$\therefore \hat{e}(S_{ID}, U) = {g_{ID}}^{r} \quad (3)$$

Hence, by substituting $Eqn.(3)$ into $Eqn.(2)$:

$$\Rightarrow m = V \oplus H_2({g_{ID}}^{r}) \quad (4)$$

Given that $Eqn.(4) = Eqn.(1)$: hence, proof of correctness is complete.

Security and Performance Analysis

While IBE relieves us of the need for certificates in the traditional Public Key Infrastructure system, its security cost can potentially be relatively higher given that its success depends heavily on a third-party IBE server that generates the private keys. And as stated earlier, this renders the entire cryptosystem liable because the Public Key Generator (PKG) becomes a single point of failure. The following are key security issues that threaten the safety of the designed IBE.

- When a Private Key Generator (PKG) is compromised, all messages protected by that server's public-private key pair are also compromised. As a result, adversaries regard the PKG as a high-value target. To mitigate the risk posed by a compromised server, the master private-public key pair could be replaced with a new independent key pair. This, however, creates a key-management issue because all users must have the most recent public key for the server.
- Because the Private Key Generator (PKG) generates private keys for users, it has the authority to decrypt and/or sign any message. As a result, IBS systems cannot be used for non-repudiation purposes. This may not be an issue for organizations that host their own PKG and are willing to trust their system administrators, as long as non-repudiation is not required.
- When a user joins the system, a secure channel must be established between the user and the Private Key Generator (PKG). IBE does not make provision for such authenticated secure channel and hence, An SSL-like connection is a common solution for a large-scale system in this case. It is critical to note that users with PKG accounts must be able to authenticate themselves. In theory, this can be accomplished through the use of a username and password, or through the use of public key pairs managed on smart cards.

Regardless of the above security concerns, IBE is great for many reasons including:

- It requires no certificates needed. A recipient's public key is derived from his identity.
- No pre-enrollment is required.
- Keys expire, so they don't need to be revoked. In a traditional public-key system, keys must be revoked if compromised.
- Less vulnerable to spam.
- Enables postdating of messages for future decryption.
- Enables automatic expiration, rendering messages unreadable after a certain date.

Conclusion

In this report, I designed and analyzed four public key cryptosystems: an Authenticated El Gamal Encryption Scheme, an Authenticated RSA Encryption Scheme, an Authenticated 3-party Diffie-Hellman key Exchange protocol and an Identity-based Encryption Scheme with bilinear pairing.

Both the authenticated El Gamal encryption and authenticated RSA encryption schemes were designed by integrating their respective digital signatures with the normal encryption scheme so as to ensure that both confidentiality and authentication requirements were satisfied.

Also, I extended the Diffie-Hellman key exchange protocol from 2 parties to 3 parties and further addressed the issue of Man-in-the-middle (MITM) attack by integrating the RSA digital signature to provide authentication services. This works perfectly on the assumption that a trusted public key infrastructure system is in place to ensure that genuine public keys are certified and made available to the public.

Finally, by using bilinear pairing, I designed an Identity based encryption scheme. It was found that, the Public Key Generator (PKG) needed to be trusted to not abuse its power since it generates all private keys and hence, a rogue PKG can encrypt and decrypt all messages that defined within the IBE's cryptosystem. Also, for this reason, the designed IBE scheme cannot be used to address non-repudiation issues. Finally, the PKG also served as a single point of failure in that should it be compromised, all past messages can be decrypted whereas the old signatures are made non-functional.