Hybrid Encryption: A Brief Introduction

Ashish Panigrahi[1], Saswat Das[2], and Sai Sriharsha Indukuri[3]

[1]School of Physical Sciences, NISER, Bhubaneswar
[2,3]School of Mathematical Sciences, NISER, Bhubaneswar

***Abstract*—The dichotomy between symmetric key encryption and public key encryption is an oft discussed and studied topic in cryptography; and both come with their own sets of advantages and drawbacks, where symmetric key encryption is susceptible to being compromised by the repeated use of a symmetric key and public key encryption, for all its security benefits, is often too computationally intensive to encrypt plaintexts of a large volume. In our term paper, we introduce and discuss hybrid encryption, where these two forms of encryption are melded together in a way that helps us take advantage of the security benefits of public key encryption and the ease of use (computationally speaking) of symmetric key encryption. We subsequently discuss two concrete examples of hybrid cryptosystems, viz. one based on ElGamal encryption (public key cryptosystem) and AES (a block cipher, and ergo symmetric key), and the popular SSL Handshake which is used to facilitate secure communication online. We also delve briefly into some aspects of the security of hybrid cryptosystems.**

I. Introduction and Motivation

A. The Need for Hybrid Encryption

For quite some time leading up to the present, symmetric and asymmetric key cryptosystems have been used extensively for securing communications in networks and among various parties; and both have their own merits and drawbacks. We will not go deep down into the details of these types of cryptosystems, but instead briefly discuss those to set the stage for the object of our attention - hybrid encryption.

In the case of symmetric key encryption, the sender (Alice) and receiver (Bob) share the same key (and therefore must share the key somehow to enable secure connection and decryption of the ciphertext by the receiver) that only they are privy to, and the sender uses said key to encrypt the message and sends the ciphertext to the receiver, who decrypts it with the aid of his knowledge of the secret key. Any third party/adversary would not be able to obtain any information about the message being sent by looking at the ciphertext, if this is implemented properly [1].

Asymmetric key encryption, also known as public key encryption or two-key encryption [2], requires the use of a pair of keys, one public key and the other one being a private key. The public key can be made visible to anyone without compromising the security of the communication and is used to encrypt messages, but the private key is to be kept secret at all times as it is used to decrypt any messages encrypted using the public key [3].

Symmetric key encryption is often faster and more efficient than asymmetric key encryption, and therefore it is preferable to use the former for encrypting large volumes of data, though it has severe drawbacks, viz. the reuse of the secret key in a symmetric cryptosystem leaks some information about the key and renders its use precarious with repetitive use and degrades the security of the communication, as the adversary can use the information she (Eve) gathers by looking at ciphertexts generated by using the same key repetitively to learn some information about the key in question [1].

On the other hand, asymmetric key encryption is more secure in this regard as knowledge of the public key does not enable the adversary to figure out the private key at all, even if those two keys are mathematically related in the construction of the cryptosystem. However, this amazing property of asymmetric key cryptosystems comes at a cost - they take are less efficient than symmetric key cryptosystems computationally, i.e. they are computationally intense, and this trade-off means that asymmetric key cryptosystems are rendered impractical for the purpose of encrypting large volumes of data, as mentioned earlier. This definitely hampered the usability of asymmetric key cryptosystems and their adoption.

So the natural question is whether we can combine the efficiency of symmetric key encryption along with the strong security of asymmetric key encryption to produce a cryptosystem that gives us the best of both worlds? The answer is yes, and is found in the form of hybrid encryption.

All authors have contributed equally.

B. Basic Introduction to Hybrid Encryption

So what we do with hybrid encryption is this - we use a symmetric key encryption scheme to encrypt the message itself, and then we use an asymmetric key encryption scheme to encrypt the key used in the symmetric key encryption part earlier. The former component of hybrid encryption with symmetric key encryption is called the *data encapsulation mechanism*, or DEM for short, and the latter with asymmetric encryption is called the *key encapsulation mechanism*, or KEM for short [4].

This is implemented in the following way, given that the sender is named Alice and the receiver is named Bob. (This has been described in great and more precise detail by Cramer and Shoup (2003) [5])

1) Alice obtains Bob's public key;
2) DEM: She then produces a fresh symmetric key k_{symm} and then encrypts the message m she wants to send using k_{symm} to obtain a ciphertext χ;
3) KEM: Note that only Alice possesses k_{symm} at this point in time, so she encrypts k_{symm} using Bob's public key (which is convenient, because symmetric keys in practice are usually and ideally of small and manageable sizes, and thus appropriate for public key encryption) to get ψ;
4) Alice sends $\bar{\psi} = (\psi, \chi)$ to Bob, and he decrypts it using his private key - first he verifies whether $\bar{\psi}$ properly encodes the tuple of ciphertexts (ψ, χ);
   a) If not, then $\bar{\psi}$ is rejected and the process halts;
   b) If yes, then ψ is decrypted using Bob's private key; if this produces a reject, then the process is halted; if not, then he obtains the symmetric key k_{symm}, using which he can decrypt χ, the ciphertext pertaining to the message m, to get m itself, if there's no issue that can possibly produce a reject.

What this essentially does is make the hybrid key encryption scheme essentially a public key (i.e. asymmetric key) encryption scheme and share in the security definitions of the same.

II. Common Notions of Security

The first thing we must do before testing the security of any encryption scheme, is to define what security is. The most widely accepted definition of security for an encryption scheme that involves Public Key Encryption is security against *adaptive* chosen cipher-text attacks [6] (introduced by Rackoff and Simon [7]).

Chosen cipher-text attack (CCA) is a scenario in which the attacker has the ability to choose cipher-texts and to view their corresponding decryptions. Schemes that are secure against chosen cipher text attacks are called CCA secure [8].

An *adaptive* chosen cipher-text attack is a chosen cipher-text attack scenario in which the attacker has the ability to make his or her choice of the inputs to the decryption function based on the previous chosen cipher-text queries.

It is clear from the two definitions that an adaptive chosen cipher text attack is stronger, and schemes that are secure against such attacks are called CCA2-secure.

In the context of hybrid encryption, CCA security of the hybrid system as a whole boils down to the security of the Key Encapsulation Mechanism (KEM) (the scheme that encrypts the symmetric key with the receiver's public key) and the data encapsulation mechanism (DEM) (scheme that encrypts the message with the symmetric key). It has been proven that, if both the KEM and DEM are CCA secure, then the hybrid system as a whole is CCA secure [5].

(A few years later, it was also shown that CCA security of KEM is in fact not a necessary condition for the CCA security for the hybrid system [9]).

A weaker notion of security is security against chosen plain text attacks (*a.k.a.* CPA security). A chosen plaintext attack is when the attacker has the ability to choose messages and view their corresponding encryptions(the reverse of what happens in CCA).

The main advantage of systems that are CPA secure, when compared to CCA secure systems, is that CPA secure systems are very practical in terms of their usage as they are computationally viable to implement. (This is the main reason for the popularity of schemes like ElGamal, which has been proven to be CPA secure but not CCA secure.)

III. A Hybrid Cryptosystem using ElGamal Encryption and AES

In this section, we will proceed to describe a clever but efficient public key encryption scheme, most prominently used in hybrid encryption, called *ElGamal encryption* [10]. However, before diving into the details of the scheme, we would like to point out that some basic prerequisites in group theory and other concepts that are necessary for the reader to understand how ElGamal works are listed in the **Appendix**.

A. Overview of Diffie-Hellman Key Agreement

The *Diffie-Hellman Key Agreement* (DHKA) provides a very practical solution to the key distribution problem, i.e. it enables two parties, Alice and Bob to agree on a common key over an insecure public channel prior to the actual communication.

The key idea behind the scheme is that in the group $\mathbb{Z}_p^*$ where p is a prime, exponentiation is a one-way function and is commutative [11], i.e.

$$k = (g^x)^y \equiv (g^y)^x \mod p \tag{1}$$

The value $(g^y)^x \mod p$ is the joint secret key that the parties, Alice and Bob can use to securely communicate in a session.

Let us consider two parties, Alice and Bob who wish to communicate securely and would like to establish a secret key for the same. There are certain parameters laid out in the public, either by a trusted third party or by Alice and Bob themselves. The DHKA consists of basically two steps, setup and exchange, where a common secret key is agreed upon.

The setup protocol is as follows:

Setup

1) Choose a large prime number p.
2) Choose an integer $g \in \{2, 3, \ldots, p-2\}$. (Technically, this is a generator of group $\mathbb{Z}_p^*$)
3) Release p and g publicly.

Alice and Bob agree publicly on prime p and generator $g \in \mathbb{Z}_p^*$. Let $n = |\mathbb{Z}_p^*|$.

The exchange protocol then follows as:

Exchange

1) Alice chooses $a \leftarrow \mathbb{Z}_n$.
2) Bob chooses $b \leftarrow \mathbb{Z}_n$.
3) Alice computes $\alpha = g^a \mod p$ and sends it to Bob.
4) Bob computes $\beta = g^b \mod p$ and sends it to Alice.
5) Bob privately computes $k_1 := \alpha^b \mod p$. Similarly, Alice privately computes $k_2 := \beta^a \mod p$.

We see that $k_1 := \alpha^b \mod p = (g^a)^b \mod p$ and $k_2 := \beta^a \mod p = (g^b)^a \mod p$. Hence we see that both parties come to agree on a common secret key, namely

$$k_1 = k_2 = g^{ab} \mod p \tag{2}$$

It is to be noted that in the public domain, we have p, g, g^a, g^b. From these parameters, it is "hard" for an attacker to decipher $k := g^{ab} \mod p$, which is the secret key Alice and Bob have agreed upon. However, we shall not show this explicitly for the sake of brevity.

B. Description of the ElGamal Encryption Protocol

We proceed to describe the ElGamal encryption scheme which is based on the DHKA scheme [10]. As usual, let's consider two parties, Alice and Bob. As seen in sec. III-A, Alice and Bob agree on a common secret key k_M via DHKA. In the setup phase of the protocol, we require no trusted third party to generate the prime p and generator g in group $\mathbb{Z}_p^*$, for they can be generated by the receiver, Bob who makes them public, either in a database or on his website [11].

The ElGamal protocol consists of 3 steps, i.e key generation $\textsc{Gen}(1^l)$, encryption), encryption $\mathcal{E}_{pk}(m)$ and decryption $\mathcal{D}_{sk}(\beta, B)$ as described below:

Key generation $\textsc{Gen}(1^l)$:
$(\mathcal{G}, p, g) \leftarrow \textsc{GroupGen}(1^l)$
Choose $a \leftarrow \mathbb{Z}_p^*$; set $\alpha = g^a \mod p$
Output $pk = (\mathcal{G}, p, g, \alpha)$ and $sk = a$

Encryption $\mathcal{E}_{pk}(m)$ (where $m \in \mathcal{G}$)
Pick $b \leftarrow \mathbb{Z}_p^*$; set $\beta = g^b \mod p$
Output $(\beta, m \cdot \alpha^b \mod p)$

Decryption $\mathcal{D}_{sk}(\beta, y \equiv m \cdot \alpha^b \mod p)$:
Compute $m = y/\beta^a$

The correctness of the decryption process follows from,

$$\frac{m \cdot \alpha^b}{(g^b)^a} = \frac{m \cdot \alpha^b}{(g^a)^b} = \frac{m \cdot \alpha^b}{\alpha^b} = m$$

To reiterate the above algorithm, Bob computes his private key $b \leftarrow \mathbb{Z}_p^*$ and public key $\beta \equiv g^b \mod p$. Note that this key pair remains constant throughout the communication session. However, Alice will have to generate a new public-private key pair everytime she wishes to send a message. $a \leftarrow \mathbb{Z}_p^*$ and $\alpha = g^a \mod p$ are her private and public keys respectively. Note that α is a temporary key. The joint shared secret key is then computed by both parties as k_M used to mask the plaintext message [11].

During the actual encryption process, Alice multiplies the plaintext message m with the shared secret key (*a.k.a.* masking key). On the decryption side, Bob simply reverses the encryption by multiplying the ciphertext with the inverse of the masking key.

C. Decisional Diffie-Hellman (DDH) Assumption

The DHKE protocol is dependent on the cyclic group $\mathcal{G}$ chosen (along with generator g). As seen in sec. III-A, Alice and Bob choose a and b respectively, uniform randomly. After which, g^a and g^b is computed, and the secret key corresponding to this is g^{ab}.

Let us define the two schemes,

- Define scheme $L_1^{\mathcal{G}}$[1] as follows:
  1) $a, b \leftarrow \mathbb{Z}_n$.
  2) Return tuple (g^a, g^b, g^{ab}).
- Define scheme $L_2^{\mathcal{G}}$ as follows:
  1) $a, b, c \leftarrow \mathbb{Z}_n$.
  2) Return tuple g^a, g^b, g^c.

As seen, scheme $L_1^{\mathcal{G}}$ generates a secret key g^{ab} and scheme $L_2^{\mathcal{G}}$ generates an independent random key g^c.

The *decisional Diffie-Hellman (DDH) assumption* states that

$$L_1^{\mathcal{G}} \approx L_2^{\mathcal{G}} \tag{3}$$

In layman terms, eq. 3 means that choosing c is "as random as" ab where a and b have been chosen uniform randomly.

D. Security of Elgamal

When dealing with the security of the scheme, we need to distinguish between two types of attacks, i.e. passive (listen-only) and active attacks (where the adversary can generate and alter messages between the two main parties of communication).

1) Passive attacks

In the public channel, we have p, g, $k_a \equiv g^a$, $k_b \equiv g^b$ and ciphertext y . A passive attack would mean recovering message x from the encrypted ciphertext $y = x \cdot g^{ab}$ [2] via eavesdropping. Security against this form of attack depends on the hardness of DHKE. This has no known[3] methods other than computing the discrete logarithm problem. Let assume that our adversary Charlie is powerful enough to compute these logarithms, then there are two ways he could go about it:

- By finding Bob's secret key b and recovering message x:
$$b = \log_g k_b \mod p \tag{4}$$
All this is assuming that Charlie can compute discrete logarithms, which is practically not feasible. If the adversary succeeds in doing so, then he follows the same steps as Bob to recover x, i.e.
$$x \equiv y \cdot (k_a^b)^{-1} \mod p \tag{5}$$
- Another way would be for Charlie to recover Alice's secret exponent a:
$$a = \log_g k_a \mod p \tag{6}$$
We make the same assumption as before i.e. Charlie has the power to compute this discrete logarithm problem. He then similarly, proceeds to compute the plaintext:
$$x \equiv y \cdot (k_b^a)^{-1} \mod p \tag{7}$$

2) Active attacks

The ElGamal encryption protocol is an asymmetric scheme and as such, it must be ensured that the public keys are indeed authentic i.e. Alice (the sender) indeed has the public key that belongs to Bob. If an adversary Charlie convinces Alice otherwise that his key is indeed Bob's, then the communication channel is compromised and Charlie can then attack the scheme. This is a *man-in-the-middle* attack. Preventing this type of attack requires authentication of the key via certificates[4].

Another breach of the protocol can happen when the private key a of Alice is used more than once. As mentioned earlier[5], Alice generates her public-private key pair for every message. Let us assume a case scenario where Alice uses the same private key a to encrypt two subsequent messages, m_1 and m_2. In both the cases, the shared masking key $k_M \equiv \beta^a \mod p$ will be the same. Also note that for both the messages, Alice's public key α would be identical. She then sends the corresponding ciphertexts for both the messages as $y_1 = m_1 \cdot k_M \mod p$ and $y_2 = m_2 \cdot k_M \mod p$ respectively, along with her public key in the form of tuples i.e.

$$(y_1, \alpha) \quad \& \quad (y_2, \alpha)$$

If somehow the adversary, Charlie can guess one of the messages, say m_1, he can then compute the shared secret key via the operation:

$$k_M \equiv y_1 m_1^{-1} \mod p$$

[1]The superscript denotes that the scheme operates in the cyclic group $\mathcal{G}$.

[2]Technically, it is $y = x \cdot g^{ab} \mod p$ but we omit $\mod p$ for the sake of brevity.

[3]This is subject to change.

[4]We shall not discuss this for the sake of brevity.

[5]See section III-B

And can subsequently decipher the second message m_2 via

$$m_2 \equiv y_2 k_M^{-1} \mod p$$

Note that for any message encrypted using the same private key a, Charlie can recover the message, provided he can guess one of the many messages encrypted with this private key a.

Also note that Charlie can detect the reuse of the private key simply because it leads to the same temporary public key α [11].

E. AES - A Symmetric Key Cryptosystem

Originally named the Rijndael algorithm after its creators, Vincent Rijmen and Joan Daemen, AES is a widely used block cipher and ergo uses a symmetric key, with a block size of 128 bits but with varying key sizes of 128, 192, and 256 bits.

We shall not be going into a lot of detail about AES but we shall provide a very brief introduction to the same before we club it with ElGamal to produce a hybrid encryption scheme.

AES is based on a design principle called a substitution-permutation network, and is implemented in several rounds/iterations, which in turn depend on the key size as follows,

$$\text{Number of rounds} = \begin{cases} 10 \text{ if key size} = 128 \text{ bits}, \\ 12 \text{ if key size} = 192 \text{ bits}, \\ 14 \text{ if key size} = 256 \text{ bits}. \end{cases}$$

It is worth noting that AES takes each the data in each block and treats it bytewise, not bitwise, to put it loosely, so, say, a 128 bit block of plaintext can be seen as being 16 bytes long, and it can be represented as a two dimensional array, called a state, of dimensions 4×4.

There exists a specified and fixed lookup table called the S-box given in the design which is used to substitute bytes of the plaintext block, which are represented by a two dimensional array as mentioned earlier, in each round.

Now we shall give a very brief description of what goes down in each round of an implementation of AES, without getting too much into the mathematics or details of it as it will involve quite a bit of field theory and the like, and interested readers may take a look at [12] for a detailed and well guided explanation of what AES does.

Key Expansion: AES uses something called a key schedule (whose details have been omitted for the sake of brevity) to expand the given cipher key into a desired number of round keys, with the round key length being equal to the block length, for every round of transformation.

AES Round

1) *Initial Round Key Addition*: The round key is added to each byte of the initial state by bitwise XOR;

2) *ByteSub*: The bytes in the input block are substituted by the means of the S-box (each byte is replaced by a byte obtained by the means of an 8-bit substitution box) given in the design to produce a 4×4 two (major order) dimensional array/state, the mathematics of which we will not go into, and this substitution introduces non-linearity into the cipher and this step is invertible for the sake of decryption;

3) *ShiftRow*: Then the rows of the state are shifted as follows
   a) The first row remains untouched;
   b) The i^{th} row is shifted to the left by $i - 1$ positions/bytes, and any bytes that spill over, so to speak, are circled back around and added to the right side of said row;

   This step is clearly and easily invertible;

4) *MixColumn*: In every round but the last round, every column is treated as a polynomial with coefficients in the finite field of size 2^8 and is multiplied modulo $x^4 + 1$ with a fixed polynomial (that is chosen in such a way that it is coprime to $x^4 + 1$ and therefore has an inverse which will be handy during decryption) to produce another polynomial which corresponds to another column which replaces the original column; in short the bytes of each column are run through an invertible linear transformation in this step to get a new column;

5) *Round Key Addition*: The next round key is added and this process is iterated, and in the last round, the *MixColumn* step shall be omitted.

In short, the ability to invert each step given the knowledge of the symmetric key involved will allow decryption of the ciphertext produced using AES.

Note that due to the size of the key in AES, whether 256 bits or 128 towards the lower end (therefore giving us either $2^{128}, 2^{192},$ or 2^{256} many possible keys), brute forcing AES is close to impossible using currently available computational power, although

other attacks have been discovered that compromised flawed implementations of AES and/or for lesser number of rounds of AES, so they are not considered to be of much concern in a practical sense at the moment.

Now that we have introduced a suitable pair of symmetric key and asymmetric key cryptosystems, we can go on and describe the construction of a hybrid encryption scheme using the same.

F. A Hybrid Encryption Scheme Based on ElGamal and AES

Now that we have sufficiently described the schemes required for the purposes of this paper, they can now be incorporated into a hybrid encryption scheme. This process is quite straightforward.

1) First, keys are generated for ElGamal and AES, by the recipient and the sender respectively;
2) *KEM*: Then the symmetric key for AES is encrypted using the ElGamal public key of the intended recipient to obtain the ciphertext $C = (A, B)$ [6];
3) *DEM*: Then the message plaintext is encrypted using AES with the key the sender generated earlier to get the ciphertext C';
4) The message gets sent to the recipient, who can then apply ElGamal and AES decryption algorithms, in that order, to first decrypt C using their secret ElGamal key to get the symmetric key for AES then use that to decrypt C' to get the plaintext of the message.

With this, we end up with a hybrid encryption scheme that combines ElGamal and AES and thus shares in both their security advantages.[7]

IV. Secure Sockets Layer (SSL)

One of the most common applications of hybrid encryption is the SSL protocol.

Secure Sockets Layer (SSL) is the communication protocol of choice for majority of the Internet community. The point of the protocol is to provide privacy and reliability between two communicating applications (user's browser and a website the user visits for example). The three main features of the SSL protocol are [14]:

1) Privacy of transferred information through encryption.
2) Identity authentication through certificates.
3) Reliability and stability of the established secure connection.

The most common applications of SSL are in email and e-commerce transactions. In fact, every website whose URL contains *HTTPS* (Secure HTTP) is secured by the SSL protocol.

Naturally, an encryption scheme to establish a link between a user's web browser and a client website's server must be as time efficient as possible. So asymmetric encryption doesn't seem to be a viable scheme to implement in such a setting, given its relatively high computation time. On the other hand, asymmetric encryption schemes are definitely more secure and this is highly desirable in an internet setting as users are often required to input sensitive information on various websites. This is precisely the reason hybrid encryption is used by SSL: we want a scheme that is both time efficient and highly secure.

A. SSL Certificate

Websites that wish to be secured by SSL must first acquire an "SSL Certificate". This digital certificate is a means of verifying the ownership of the website and also prevents attackers from creating a fake version of the site, and gain user trust. It is issued to a website by a Certificate Authority (CA). A CA is a trusted third party, that generates and gives out SSL certificates. The certificate contains [15]:

1) The domain name the certificate was issued for;
2) Organization or person it was issued to;
3) The issuing CA and their digital signature;
4) Issue date and expiration date of certificate;
5) A public key(unique to the website) that will be used by clients of the website;

B. The Handshake

When a browser attempts to access a website that is secured by SSL, the browser and the web server establish an SSL connection using a process called an "SSL Handshake" (the handshake, in fact, is invisible to the user and happens instantaneously) [16], [17].

The outline of the handshake:

1) **Client Hello**: The client's web browser initiates the handshake by sending a "hello" message to the server;

[6] Refer to the section on the description of the ElGamal encryption protocol for an explanation about the contents of the tuple C.

[7] One can refer to [13] for a look at a ElGamal+AES based encryption scheme described by Iavich et al.

2) **Server Hello**: In reply to the client hello message, the server sends a message containing the server's SSL certificate;

3) **Authentication**: The client's web browser verifies the server's SSL certificate with the certificate authority that issued it (every browser has a list of trusted CA's). This confirms that the server is who it says it is, and that the client is interacting with the actual owner of the domain.

4) **Secret Key Generation**: Client's web browser generates a random "session key", encrypts with public key of website (available on the SSL certificate) and sends it to the server;

5) The server decrypts the key with its private key, encrypts a "finished message" using the session key and sends it to the client;

6) Handshake is over and communication proceeds with use of the session key.

V. Conclusion

Hybrid encryption is a very clever and sophisticated idea that grants the advantages of both symmetric encryption and asymmetric encryption to any system implementing it. we started off by briefly discussing the key differences between symmetric and asymmetric encryption and explained how hybrid cryptosystems are built upon both of them.

Some common definitions of security that are frequently used when designing hybrid cryptosystems are also introduced, so that the reader gets a general idea of the level of security of a hybrid cryptosystem and how its security compares with the level of security of symmetric and asymmetric encryption schemes.

The rest of the paper was divided into two sections. The first section was about one of the most popular hybrid cryptosystems: the hybrid of ElGamal and AES algorithms.

In the second section we detailed the SSL handshake, a security protocol that utilises hybrid encryption and one that we interact with almost everyday. Some information about SSL certificates was provided to enable the reader to understand exactly what transpires in the SSL handshake.

Appendix

A. An overview of Group theory

[18] defines an Abelian group $\mathcal{G}$ as

Definition 1. An Abelian group $\mathcal{G}$ is a finite set of elements along with an operation $*$ such that:

- **Closure** For all $a, b \in \mathcal{G}$ we have $a * b \in \mathcal{G}$. We use $a * b$ as ab henceforth for brevity.
- **Associativity** $\forall a, b, c \in \mathcal{G}$, $(ab)c = a(bc)$.
- **Commutativity** $\forall a, b \in \mathcal{G}$, $ab = ba$.
- **Existence of identity** $\exists$ an element $1 \in \mathcal{G}$ such that $1 * a = a \ \forall a \in \mathcal{G}$. This element is the identity of $\mathcal{G}$.
- **Inverse** $\forall a \in \mathcal{G}$, $\exists$ an element $a^{-1} \in \mathcal{G}$ such that $aa^{-1} = 1$.

Theorem 1. *Let $\mathcal{G}$ be a finite Abelian group of order q. Then $a^q = 1$ for all $a \in \mathcal{G}$.*

Proof. Taken from [18].

Let us take $a_1, \dots a_q$ to be the elements of $\mathcal{G}$ and let $a \in \mathcal{G}$ be some arbitrary element. We note that the sequence of elements $aa_1, aa_2, \dots, aa_q$ also contains exactly the elements of group $\mathcal{G}$. Therefore:

$$\begin{aligned} a_1 \cdot a_2 \cdots a_q &= (aa_1) \cdot (aa_2) \cdots (aa_q) \\ &= a^q (a_1 \cdot a_2 \cdots a_q) \end{aligned}$$

On multiplying both sides by $(a_1 \cdots a_q)^{-1}$, we get

$$a^q = 1$$

■

Corollary 1.1. *Let $\mathcal{G}$ be a finite Abelian group of order q, and let n be a positive integer. Then $g^n = g^{n \mod q}$.*

Proof. Taken from [18].

Let us write $n = n_q \mod q$ so that n can be further written as $n = aq + n_q$ for some $a \in \mathbb{Z}$. Then, we have

$$g^n = g^{aq + n_q} = (g^a)^q g^{n_q} = g^{n_q}$$

■

Lemma 2. *If $\mathcal{G}$ is an Abelian group with prime order q, then*

1) *$\mathcal{G}$ is cyclic; furthermore,*
2) *every element of $\mathcal{G}$ (except the identity) is a generator.*

B. Discrete Logarithm Problem

The problem can be described as follows: given a generator g of group $\mathcal{G}$ and a random element $h \in \mathcal{G}$, compute $\log_g h$.

A common reference is made to the *discrete logarithm assumption*, which says that for most groups, the discrete logarithm problem is "hard".

References

[1] P. Smirnoff and D. M. Turner, *Symmetric key encryption - why, where and how it's used in banking*, Jan. 2019. [Online]. Available: https://www.cryptomathic.com/news-events/blog/symmetric-key-encryption-why-where-and-how-its-used-in-banking.

[2] B. Kaliski, "Assymmetric cryptosystem," in *Encyclopedia of Cryptography and Security*, H. C. van Tilbourg and S. Jajodia, Eds., 2011, pp. 49–50.

[3] *What is asymmetric encryption? understand with simple examples*, Jan. 2021. [Online]. Available: https://cheapsslsecurity.com/blog/what-is-asymmetric-encryption-understand-with-simple-examples/.

[4] K. Kurosawa, "Hybrid encryption," in *ECS*, H. C. van Tilbourg and S. Jajodia, Eds., 2011, pp. 570–572.

[5] R. Cramer and V. Shoup, *Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack*, Cryptology ePrint Archive, Report 2001/108, https://eprint.iacr.org/2001/108, 2001.

[6] ——, "Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public key encryption," 2001. [Online]. Available: https://link.springer.com/chapter/10.1007%2F3-540-46035-7_4.

[7] C. Rackoff and D. R. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack," 2001. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-46766-1_35.

[8] A. Biryukov. "Adaptive chosen ciphertext attack." (), [Online]. Available: https://link.springer.com/referenceworkentry/10.1007%2F978-1-4419-5906-5_543.

[9] K. Kurosawa and Y. Desmedt, "A new paradigm of hybrid encryption scheme," [Online]. Available: https://iacr.org/archive/crypto2004/31520425/crypto-hybrid-04.pdf.

[10] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985. DOI: 10.1109/tit.1985.1057074. [Online]. Available: https://doi.org/10.1109%2Ftit.1985.1057074.

[11] C. Paar and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media, 2009.

[12] J. Daemen and V. Rijmen, "Aes proposal: Rijndael," 1999.

[13] M. Iavich, S. Gnatyuk, E. Jintcharadze, Y. Polishchuk, and R. Odarchenko, "Hybrid encryption model of aes and elgamal cryptosystems for flight control systems," in *2018 IEEE 5th International Conference on Methods and Systems of Navigation and Motion Control (MSNMC)*, IEEE, Oct. 2018. DOI: 10.1109/msnmc.2018.8576289. [Online]. Available: http://dx.doi.org/10.1109/MSNMC.2018.8576289.

[14] H. McKinley, "Ssl and tls: A beginner's guide," 2003. [Online]. Available: https://www.sans.org/reading-room/whitepapers/protocols/ssl-tls-beginners-guide-1029.

[15] cloudflare. "What is an ssl certificate?" (2021), [Online]. Available: https://www.cloudflare.com/en-gb/learning/ssl/what-is-an-ssl-certificate/.

[16] digicert. "What is an ssl certificate?" (2021), [Online]. Available: https://www.digicert.com/what-is-an-ssl-certificate (visited on 05/14/2021).

[17] cloudflare. "What happens in a tls handshake?" (2021), [Online]. Available: https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/.

[18] C. Y. Koo, N. Yakovenko, and J. Blank, "Cmsc 858k—advanced topics in cryptography february 5, 2004,"