# New Directions in Cryptography (by W. Diffie and M. Hellman)

Vasilis Stamatis

MSc. Program in Algorithms, Logic and Discrete Mathematics UoA & NTUA

for the course requirements "Algorithms and Complexity"

November 2022

## Importance of Paper

1. Comes to solve a fundamental problem: Safe communication, without having to exchange a key in advance.
2. **Public key Cryptosystem**: Transmitting key information over insecure channels, without compromising the security of the system. Instead of **one secret key** we have **two distinct keys**.
3. Two Keys: **E**ncrypting and **D**ecrypting, computing $D$ from $E$ is computationally infeasible (eg. $10^{100}$ instructions) therefore, the Encrypting key $E$ can be made public (!)
4. If I know your public key (your Encrypting key) I can use that key to encrypt a message that I'll send to you, such that, only you, can decript that message with your decrypting key.
5. This obviously, solves the problem of having to share a secret key beforehand because anyone can look up anyone elses public key, and then establish a secure communication channel with that person.

## Importance of Paper

1. **Public Key Distribution Systems**: Two people who do not share a secret key, that they have exchanged in advance, want to arrive at a secret key. So, they will communicate back and forth, until they arrive at a key in common.
2. The property of the system is that, even someone who has observed all the communication between those two parties, cannot use that information to derive the secret key that those two parties share at the end of that process.
3. **Digital Signatures**: We want to produce Digital Signatures, that have the same properties as written signatures on paper documents. Only the owner of the signature, could have produced that signature and it is attached to a specific document, in other words, a digital signature proves the authenticity of a document, and, it can be checked by anyone to verify that authenticity.

## Security of a Cryptosystem

Types of attacks that can be mounted on one. A cryptosystem that requires an infeasibly large amount of storage, or computational resources in general for it to break is called **unconditionally secure**.

- The later, obviously, is impractical to achieve.
- The only uncondiotionally secure system in common use is the "One - Time Pad"

## One-Time Pad

**One - Time Pad**: Requires the use of a single-use pre-shared randomly chosen key that is not smaller than the message being sent. Then, each bit or character of the plaintext is encrypted by combining it with the corresponding bit or character from the pad using modular addition. The resulting ciphertext will be impossible to decrypt or break if the following four conditions are met:

- The key must be at least as long as the plaintext.
- The key must be random
- The key must never be reused in whole or in part.
- The key must be kept **completely secret** by the communicating parties.

## One-Time Pad

Hello

$\downarrow$

7, 4, 11, 11, 14

$+$ 23, 12, 2, 10, 11 (Key)

30, 16, 13, 21, 25 (Message + Key)

$\downarrow$ mod(26)

4, 16, 13, 21, 25 (Ciphertext)

$\downarrow$

EQNVZ

EQNVZ

$\downarrow$

4, 16, 13, 21, 25

$\downarrow$

$-$19, 4, 11, 11, 14 (Ciphertext $-$ Key)

$\equiv$

7, 4, 11, 11, 14 ( -//- mod(26))

$\downarrow$

Hello

## Types of Attacks

In increasing order of strenght:

- **Ciphertext only attack**: Where the attacker only knows the ciphertext. Occurs frequently in practice, the cryptanalyst uses only the knowledge of the statistical properties of the language in use and knowledge of certain "probable" words. **(Weak)**
- **Known plaintext attack**: An attacker possesses a decent number of ciphertext and plaintext pairs
- **Chosen plainetext attack**: An attacker can craft plaitext messages and then see what the corresponding cyphertext for them is. **(Strong)**

## Foundations of Public Key Cryptography

The major limitation of existing Cryptosystems at the time was that you had to share a private key, before communicating. This is not always a practical thing to achieve.
I.e. For a large group of people, let's say $n$, you'd have $\sum_{i=1}^{n} i = \frac{n(n-1)}{2} = O(n^2)$ potential pairs who may wish to communicate privately from all others. It is unrealistic to assume either that a pair of users with no prior acquintance will be able to wait for a key to be sent by some secure physical means, or that keys for all $\frac{n(n-1)}{2}$ pairs can be arranged in advance.

## Proposal of Public Key Cryptosystem

Given a key $K$ the system consists of an encryption algorithm

$$E_k : \{M\} \longrightarrow \{M\}$$

A decription algorithm

$$D_k : \{M\} \longrightarrow \{M\}$$

for every $K \in \{K\}$ on a finite message space $\{M\}$ such that:

1. For every $K$, $E_k = D_k^{-1}$
2. For every $K, M$ the algorithms $E_k$ and $D_k$ are easy to compute.
3. For (almost) every $K$, each easily computed algorithm equivalent to $D_k$ is computationally infeasible to derive from $E_k$. (i.e. computing the decryption key $D_k$ is computationally infeasible, knowing only the encryption key $E_k$)
4. For every $K$, it is easy to compute inverse pairs $E_k$ and $D_k$ from $K$.

## Proposal of Public Key Cryptosystem

Because it is computationally infeasible to derive the Decryption Key from the Encryption Key, the encryption key can be made public.
The authors in this paper propose the Public Key cryptosystem, but they do not provide an actual practical implementation of this, they do provide a practical implementation of Key Exchange, but not a public key Cryptosystem. This was done in the RSA Cryptosystem which relies on the practical difficulty of factoring the product of two large prime numbers, the "factoring problem".

## Implementation of a Key Exchange

If we had such a system, the users would generate the keys $E$ and $D$ and while keeping the decryption key $D$ secret, they would make the encryption key $E$ totally public, by placing it in a public directory along with the user's name and address.
Anyone, could then encrypt messages and send them to the user, but only the specific user could decypher those messages.

## Diffie - Hellman Key Exchange Protocol

Goal: Two parties to arrive at a shared secret key while only communicating back and forth on a public channel. So that, anyone, who has heard all the messages, going back and forth between them is still unable to feasibly compute the shared secret key that each of them posess at the end.
The proposed technique by Diffie and Hellman, makes use of the difficulty of computing logarithms over $\mathbb{F}_q$, where $q$ is prime.

## Feedback

- A Finite (or Galois) Field, is a field with an order $q$ or $q^e$ where $q$ is prime. In that case, $\mathbb{F}_q \cong \mathbb{Z}_q/\mathbb{Z} \cong \mathbb{Z}_q$.
- We can see every Finite Field with $q^n$ elements ($q$ is prime) as the roots of the polynomial $x^{q^n} - x$ in $\mathbb{Z}_q[x]$.
- **A primitive element** of a finite field is a generator of the multiplicative group $< \mathbb{F}_q^*, \cdot >$ (which is cyclic) of the field. Meaning, $a^n = 1$ for the minimum possible $n$. The number of primitive elements in $\mathbb{F}_q$ is $\phi(q-1)$, where $\phi$ is Euler's totient function.
- There are infinitely many finite fields with a given characteristic $q$.

## Feedback

- **Discrete log problem (DLP)**: in any group $G$, powers $b^k$ can be defined for all integers $k$, and the discrete logarithm $log_b(a)$ is an integer $k$ such that $b^k = a$. The more commonly used term is index: we can write $x = ind_r(a) \pmod{m}$ (read "the index of $a$ to the base $r \pmod{m}$") if if $r$ is a primitive root of $m$ and $gcd(a, m) = 1$.
- The discrete logarithm problem is considered to be computationally intractable. That is, no efficient classical algorithm is known for computing discrete logarithms in general. A general algorithm for computing $log_b(a)$ in finite groups $G$ is to raise $b$ to larger and larger powers $k$ until the desired $a$ is found. This algorithm is sometimes called trial multiplication.

## Feedback

- It requires running time linear in the size of the group $G$ and thus exponential in the number of digits in the size of the group. Therefore, it is an exponential-time algorithm, practical only for small groups $G$. More sophisticated algorithms exist, usually inspired by similar algorithms for integer factorization. These algorithms run faster than the naive algorithm, some of them proportional to the square root of the size of the group, and thus exponential in half the number of digits in the size of the group. However none of them runs in polynomial time (in the number of digits in the size of the group).
- While computing discrete logarithms and factoring integers are distinct problems, they share some properties:
  - Both are special cases of the hidden subgroup problem for finite abelian groups
  - Both problems seem to be difficult (no efficient algorithms are known for non-quantum computers)
  - Algorithms from one problem are often adapted to the other

## Diffie - Hellman Key Exchange Protocol

As we mentioned earlier, the proposed technique by Diffie and Hellman, makes use of the difficulty of computing logarithms over $\mathbb{F}_q$. Let

$$Y = a^X \pmod q \text{ for } 1 \leqslant X \leqslant q-1$$

Where, $a$ is a fixed primitive element of $\mathbb{F}_q$, then $X$ is referred to as the logarithm of $Y$ to the base $a \pmod q$:

$$X = log_a Y \pmod q \text{ for } 1 \leqslant Y \leqslant q-1$$

- Computing $Y$ from $X$ is easy. Exponentiation is fast (using for example the Doubling Method for Exponentiation) taking $2 \cdot log_2 q = O(log(q))$ operations.
- if $Y = a^{18} = (((a^2))^2)^2)^2 \cdot a^2$
- Computing $X$ from $Y$ is much harder (for carefully chosen $q$'s), as it requires $O(\sqrt{q})$ operations (using the best known algoiritm at that time).

## The Protocol

Each user $i$ chooses a number $X_i$, uniformly from the set $\{1, 2, ..., q-1\}$. And keeps, $X_i$ secret. However:

$$Y_i = a^{X_i} \pmod{q}$$

is made public.
When user $i$ wants to communicate with user $j$. He goes and finds $Y_j$ (which is public) and then computes $K_{ij}$, where:

$$K_{ij} = Y_j^{X_i} \pmod{q} \equiv (a^{X_j})^{X_i} \pmod{q} \equiv a^{X_j X_i} \pmod{q}$$

**But Wait!**:

$$a^{X_i X_j} \pmod{q} \equiv a^{X_j X_i} \pmod{q}$$

Therefore, user $j$ can do the exact same thing for $i$, as $i$ did for $j$(!), computing $K_{ji} = a^{X_i X_j} \pmod{q}$. Therefore, both arriving, in the same key!

## Security of the Protocol

Now, let's say, a third party wants to learn what $i$ and $j$ are saying. Then, that user must compute $K_{ij}$ from $Y_i$ and $Y_j$ alone. Therefore, computing:

$$K_{ij} = Y_i^{(log_a Y_j)} \pmod q$$

If $q$ is a prime number, represented in $b$-bits (i.e. slightly less than $2^b$). Then exponentiation takes at most $2b$ multiplications $\pmod q$, while taking logs, will require $q^{1/2} = 2^{b/2}$ operations.
Examples:

- If $q < 2^b = 2^{200}$, then at most 400 multiplications are required to compute $Y_i$ from $X_i$, or $K_{ij}$ from $Y_i$ and $X_j$, yet, taking logs $\pmod q$ requires $2^{200/2} = 2^{100}$ operations. (For a 9th Gen. $i9$ it will take $71,088,880$ years).
- If $q = M_{82,589,933} = 2^{82,589,933} - 1$ (GIMPS Sept. 2022), then at most $2 \cdot 82,589,933$ multiplications are required to compute $Y_i$ from $X_i$, or $K_{ij}$ from $Y_i$ and $X_j$, yet, taking logs $\pmod q$ requires $2^{82,589,933/2}$ operations.

## One - Way Authentication - Digital Signatures

- It must be easy for anyone to check that the signature is authentic. Meaning, that anyone other than that person, cannot produce a valid signature.
- If a party wants to digitally sign a message $M$, It creates the message $D_A(M)$ (which is the decryption of that message). If anyone else wants to know my message, using my public key $E_A$ can decrypt the message I just produced, they will be able to read my message but also know that only my private key could have produced that message.

## Historical Perspective

- Diffie and Hellman, view this paper as natural outgrowth of trends in cryptography, stretching back hundreds of years.
- It was very well accepted at the time that secrecy was the heart of cryptography. But didn't know what to hide. *(The Algorithm? The key?)*
- Kerchoff in 1881 wrote that the cryptographic system, should be made public, the keys on the other hand should not.
- In 1960, we wanted cryptosystems to be strong enough to resist known plaitext attacks (meaning all messages, no longer had to be secret).
- Overtime the portion of a cryptosystem, which had to be kept secret, kept decreasing.
- Public Key systems, were a natural continuation of this trend toward decreasing secrecy.