The Diffie-Hellman Key Exchange

Raskopoulou Vasiliki

Algorithms and Complexity

February 17, 2022

Contents

Contents

A cryptographic system is a single parameter family $\{S_K\}_{K\in\{K\}}$ of invertible transformations

$$S_K : \{P\} \to \{C\}$$

for a space $\{P\}$ of plaintext messages to a space $\{C\}$ of ciphertext messages. The parameter K is called the key and is selected from a finite set $\{K\}$ called the keyspace.

For example, let us encode the english alphabet $\{A, B, \ldots, Z\}$ to the alphabet $\Sigma = \{1, 2, \ldots, 26\}$.

For example, let us encode the english alphabet $\{A, B, \ldots, Z\}$ to the alphabet $\Sigma = \{1, 2, \ldots, 26\}$.
The Caeser cipher with $\{P\} = \{C\} = \Sigma^*$ and $\{K\} = \Sigma$ uses the invertible transformations

$$S_K(a_1 a_2 \ldots a_n) = (a_1 + K \mod 26)(a_2 + K \mod 26) \ldots (a_n + K \mod 26).$$

Similarly, the One-Time-Pad with $\{P\} = \{C\} = \{K\} = \Sigma^*$ uses keys as long as the plaintext:

$$S_K(a_1 a_2 \ldots a_n) = (a_1 + k_1 \mod 26)(a_2 + k_2 \mod 26) \ldots (a_n + k_n \mod 26),$$

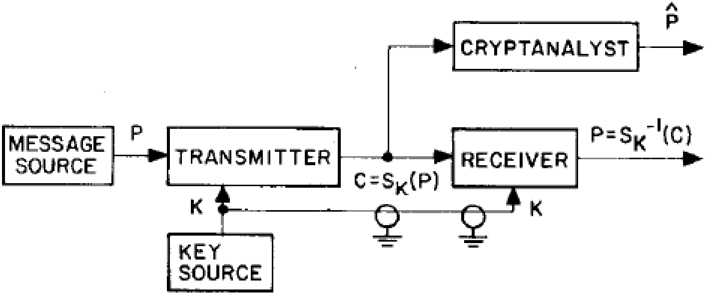
where $K = k_1 k_2 \ldots k_n$.

Classifying the threats:

- Ciphertext-only attack: totally insecure systems
- Known plaintext attack: not secure in case of later public disclosure
- Chosen plaintext attack: allows to opponents to plant messages

Private Key Exchange

A basic problem in conventional cryptography is to ensure a secure communication via a private channel



Private Key Exchange

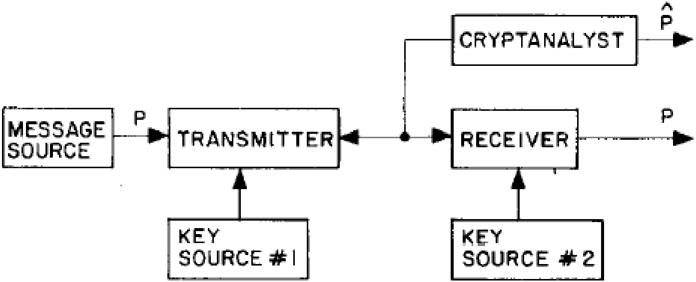
Due to physical constraints, this is infeasible when developing large, secure, telecommunication systems.

Private Key Exchange

Due to physical constraints, this is infeasible when developing large, secure, telecommunication systems.

- immediate, fast communication
- communication of unknown parties
- n different users who wish to communicate privately from others

Contents


CRYPTANALYST
$\hat{P}$
MESSAGE SOURCE
P
TRANSMITTER
RECEIVER
P
KEY SOURCE #1
KEY SOURCE #2

A public key cryptosystem is a pair of families $\{E_K\}_{K\in\{K\}}, \{D_K\}_{K\in\{K\}}$ of algorithms representing invertible transformations:

$$E_K : \{M\} \to \{M\}$$

$$D_K : \{M\} \to \{M\}$$

where $\{M\} = \{P\} = \{C\}$ is a finite message space, such that

A public key cryptosystem is a pair of families $\{E_K\}_{K\in\{K\}}, \{D_K\}_{K\in\{K\}}$ of algorithms representing invertible transformations:

$$E_K : \{M\} \to \{M\}$$

$$D_K : \{M\} \to \{M\}$$

where $\{M\} = \{P\} = \{C\}$ is a finite message space, such that

- for every $K \in \{K\}$, E_K is the inverse of D_K,

A public key cryptosystem is a pair of families $\{E_K\}_{K\in\{K\}}, \{D_K\}_{K\in\{K\}}$ of algorithms representing invertible transformations:

$$E_K : \{M\} \to \{M\}$$

$$D_K : \{M\} \to \{M\}$$

where $\{M\} = \{P\} = \{C\}$ is a finite message space, such that

- for every $K \in \{K\}$, E_K is the inverse of D_K,
- for every $K \in \{K\}$ and $M \in \{M\}$, the algorithms E_K and D_K are easy to compute,

A public key cryptosystem is a pair of families $\{E_K\}_{K\in\{K\}}, \{D_K\}_{K\in\{K\}}$ of algorithms representing invertible transformations:

$$E_K : \{M\} \to \{M\}$$

$$D_K : \{M\} \to \{M\}$$

where $\{M\} = \{P\} = \{C\}$ is a finite message space, such that

- for every $K \in \{K\}$, E_K is the inverse of D_K,
- for every $K \in \{K\}$ and $M \in \{M\}$, the algorithms E_K and D_K are easy to compute,
- for almost every $K \in \{K\}$, each easily computed algorithm equivalent to D_K is computationally infeasible to derive from E_K,

A public key cryptosystem is a pair of families $\{E_K\}_{K\in\{K\}}, \{D_K\}_{K\in\{K\}}$ of algorithms representing invertible transformations:

$$E_K : \{M\} \to \{M\}$$

$$D_K : \{M\} \to \{M\}$$

where $\{M\} = \{P\} = \{C\}$ is a finite message space, such that

- for every $K \in \{K\}$, E_K is the inverse of D_K,
- for every $K \in \{K\}$ and $M \in \{M\}$, the algorithms E_K and D_K are easy to compute,
- for almost every $K \in \{K\}$, each easily computed algorithm equivalent to D_K is computationally infeasible to derive from E_K,
- for every $K \in \{K\}$ it is feasible to compute inverse pairs E_K and D_K from K.

Each user generates a pair of inverse transformations, E and D. The enciphering algorithm E can de made public, while the deciphering transformation D must be kept secret.

A naive example

Let the message to be enciphered be a binary n-vector $\mathbf{m}$, and E be an invertible $n \times n$ matrix, i.e., $\mathbf{c} = E\mathbf{m}$. Then $D = E^{-1}$ and $\mathbf{m} = D\mathbf{c}$.

A naive example

Let the message to be enciphered be a binary n-vector $\mathbf{m}$, and E be an invertible $n \times n$ matrix, i.e., $\mathbf{c} = E\mathbf{m}$. Then $D = E^{-1}$ and $\mathbf{m} = D\mathbf{c}$.

- E is the inverse of D

A naive example

Let the message to be enciphered be a binary n-vector $\mathbf{m}$, and E be an invertible $n \times n$ matrix, i.e., $\mathbf{c} = E\mathbf{m}$. Then $D = E^{-1}$ and $\mathbf{m} = D\mathbf{c}$.

- E is the inverse of D
- $E\mathbf{m}$ and $D\mathbf{c}$ are easy to compute (about n^2 operations)

A naive example

Let the message to be enciphered be a binary n-vector $\mathbf{m}$, and E be an invertible $n \times n$ matrix, i.e., $\mathbf{c} = E\mathbf{m}$. Then $D = E^{-1}$ and $\mathbf{m} = D\mathbf{c}$.

- E is the inverse of D
- $E\mathbf{m}$ and $D\mathbf{c}$ are easy to compute (about n^2 operations)
- Calculating D from E (matrix inversion) is a harder problem (but not hard enough!$\rightarrow$ n^3 operations)

A naive example

Let the message to be enciphered be a binary n-vector $\mathbf{m}$, and E be an invertible $n \times n$ matrix, i.e., $\mathbf{c} = E\mathbf{m}$. Then $D = E^{-1}$ and $\mathbf{m} = D\mathbf{c}$.

- E is the inverse of D
- $E\mathbf{m}$ and $D\mathbf{c}$ are easy to compute (about n^2 operations)
- Calculating D from E (matrix inversion) is a harder problem (but not hard enough!$\rightarrow$ n^3 operations)
- It is simpler to obtain a pair of inverse matrices than it is to invert a given matrix

Example: RSA

Let $K = (p, q)$, where p, q are prime numbers (secret), $N = pq$ (public) and $\{M\} = \mathbb{Z}_N^*$.
Then

$$E_K : \mathbb{Z}_N^* \to \mathbb{Z}_N^*, m \mapsto m^e \mod N,$$

where $e < N$ and

$$D_K : \mathbb{Z}_N^* \to \mathbb{Z}_N^*, c \mapsto c^d \mod N,$$

for some d such that $ed = 1 \mod \phi(N)$.

Example: RSA

- E_K is the inverse of D_K:

$$D_K(E_K(m)) = E_K(m)^d \mathrm{mod} N = (m^e)^d \mathrm{mod} N = m^{ed} \mathrm{mod} N = m$$

Example: RSA

- E_K is the inverse of D_K:

$$D_K(E_K(m)) = E_K(m)^d \mathrm{mod} N = (m^e)^d \mathrm{mod} N = m^{ed} \mathrm{mod} N = m$$

- $E_K(m)$ and $D_K(c)$ are easy to compute

Example: RSA

- E_K is the inverse of D_K:

$$D_K(E_K(m)) = E_K(m)^d \mathrm{mod} N = (m^e)^d \mathrm{mod} N = m^{ed} \mathrm{mod} N = m$$

- $E_K(m)$ and $D_K(c)$ are easy to compute
- Calculating D_K from E_K (given e, find d) is an **NP** problem

Example: RSA

- E_K is the inverse of D_K:

$$D_K(E_K(m)) = E_K(m)^d \bmod N = (m^e)^d \bmod N = m^{ed} \bmod N = m$$

- $E_K(m)$ and $D_K(c)$ are easy to compute
- Calculating D_K from E_K (given e, find d) is an **NP** problem
- It is feasible to find e, d such that $ed = 1 \mod \phi(pq)$

"$E_K(m)$ and $D_K(c)$ are easy to compute":
Modular Exponentiation $a^m \mod n$ can be done in $O(\log k)$, where k is the exponent's length in binary.

"$E_K(m)$ and $D_K(c)$ are easy to compute":
Modular Exponentiation $a^m \mod n$ can be done in $O(\log k)$, where k is the exponent's length in binary.

Require: $a, n, b_0, b_1, \ldots, b_{k-1}$ such that $m = (b_{k-1} \ldots b_1 b_0)_2$

$x := a$
$y := 1$
for $i = 0, \ldots k-1$ **do**
    if $b_i = 1$ **then**
        $y := y \cdot x \mod n$
        $x := x^2 \mod n$
    end if
end for
return y

Contents

The Diffie-Hellman Protocol

A and B wish to exchange a private key via a public, insecure channel, in order to use a symmetric cryptosystem. Let q be a prime number and $a \in \{2, \ldots, q-1\}$. q and a are agreed upon by A and B, and they can be made public.

The Diffie-Hellman Protocol

A and B wish to exchange a private key via a public, insecure channel, in order to use a symmetric cryptosystem. Let q be a prime number and $a \in \{2, \ldots, q-1\}$. q and a are agreed upon by A and B, and they can be made public.

- User A generates a random number $X \in \{1, 2, \ldots, q-1\}$ and keeps it secret.

The Diffie-Hellman Protocol

A and B wish to exchange a private key via a public, insecure channel, in order to use a symmetric cryptosystem. Let q be a prime number and $a \in \{2, \ldots, q-1\}$. q and a are agreed upon by A and B, and they can be made public.

- User A generates a random number $X \in \{1, 2, \ldots, q-1\}$ and keeps it secret.
- Makes public $Y = a^X \mod q$.

The Diffie-Hellman Protocol

A and B wish to exchange a private key via a public, insecure channel, in order to use a symmetric cryptosystem. Let q be a prime number and $a \in \{2, \ldots, q-1\}$. q and a are agreed upon by A and B, and they can be made public.

- User A generates a random number $X \in \{1, 2, \ldots, q-1\}$ and keeps it secret.
- Makes public $Y = a^X \mod q$.
- User B also picks $X' \in \{1, 2, \ldots, q-1\}$ (secret) and publicizes $Y' = a^{X'} \mod q$.

The Diffie-Hellman Protocol

A and B wish to exchange a private key via a public, insecure channel, in order to use a symmetric cryptosystem. Let q be a prime number and $a \in \{2, \dots, q-1\}$. q and a are agreed upon by A and B, and they can be made public.

- User A generates a random number $X \in \{1, 2, \dots, q-1\}$ and keeps it secret.
- Makes public $Y = a^X \mod q$.
- User B also picks $X' \in \{1, 2, \dots, q-1\}$ (secret) and publicizes $Y' = a^{X'} \mod q$.
- A computes $K = (Y')^X \mod q = a^{X'X} \mod q$
- B computes $Y^{X'} \mod q = a^{XX'} \mod q = K$

Both A and B have the same key K.

The Diffie-Hellman Protocol

Since any other user does not possess neither X nor X', they should be able to compute either $\log_a Y \mod q = X$ or $\log_a Y' \mod q = X'$, which is computationally infeasible:

The Discrete Logarithm Problem is in **NP**.

The Diffie-Hellman Protocol

Since any other user does not possess neither X nor X', they should be able to compute either $\log_a Y \mod q = X$ or $\log_a Y' \mod q = X'$, which is computationally infeasible:

The Discrete Logarithm Problem is in **NP**.

Proof.

$$q \text{ is a prime and } a \text{ is a primitive root} \mod q$$

$$\text{iff}$$

$$\mathrm{ord}_q(a) = q - 1.$$

We can verify these conditions in non-deterministic polynomial time:

- guess all prime factors $p_1, \ldots, p_k$ of $q - 1$,
- use primality test for all of them,
- compute $(q-1)/p_i$ and verify that $a^{(q-1)/p_i} \not\equiv 1 \mod q$ for all p_i.

So, given q, a, n, m we can verify whether q is a prime and a a primitive root and if $a^m \equiv n \mod q$. This means that DLP is in **NP**.
On the other hand, if $a^m \not\equiv n \mod q$, then

- there are i, j with $0 < i, j < q$ such that $a^i \equiv a^j \mod q$ or
- there is an $\ell \leq m$ such that $a^\ell \equiv n \mod q$ or
- $n \geq q$

All of the above can be checked in polynomial time, so DLP is in **coNP**.
We have that DLP is in $\mathbf{NP} \cap \mathbf{coNP}$. □

Contents

Conventional Cryptography

- Written signature $\rightarrow$ digital signature

Conventional Cryptography

- Written signature $\rightarrow$ digital signature
- easy to recognize the signature as authentic

Conventional Cryptography

- Written signature $\rightarrow$ digital signature
- easy to recognize the signature as authentic
- impossible to produce it

Conventional Cryptography

- Written signature $\rightarrow$ digital signature
- easy to recognize the signature as authentic
- impossible to produce it

Digital signature must be recognizable without being known

The "login" problem

- User enters password PW
- Computer computes and stores a function $f(PW)$
- Each time a login with X is attempted, computer calculates $f(X)$ and compares with stored value $f(PW)$

Computation time of f must be small, computation of f^{-1} must be practically infeasible.

The "login" problem

- User enters password PW
- Computer computes and stores a function $f(PW)$
- Each time a login with X is attempted, computer calculates $f(X)$ and compares with stored value $f(PW)$

Computation time of f must be small, computation of f^{-1} must be practically infeasible.

$$f \rightarrow \text{ one-way function}$$

The "login" problem

- User enters password PW
- Computer computes and stores a function $f(PW)$
- Each time a login with X is attempted, computer calculates $f(X)$ and compares with stored value $f(PW)$

Computation time of f must be small, computation of f^{-1} must be practically infeasible.

$$f \rightarrow \text{ one-way function}$$

Still not entirely secure

Public Key Cryptography

- A wants to send a message M to B
- A deciphers the message $D_K(M)$ using his (secret) algorithm D_K and sends it to B, along with M.
- B enciphers the message $E_K(D_K(M))$ using A's (public) algorithm E_K.
- Obviously, B obtains M and can therefore be assured of the authenticity of its sender.

Example: RSA digital signature

Let $E(x) = x^e \mod N$ be A's public encryption algorithm, and $D(y) = y^d \mod N$ their secret decryption algorithm.

- A sends (m, m') to B, where $m' = m^d \mod N$.
- B computes $E(m') = (m')^e \mod N = (m^d)^e \mod N = m$, and therefore recognizes the authenticity of the sender.

Conversely, suppose that a third user C claims to be A. Since C has no knowledge of d, he sends to user B (m, m'), where $m' = m^{d'} \mod N$ for some $d' \neq d$. B computes $E(m') = (m')^e \mod N = (m^{d'})^e \mod N \neq m$ and sees through C's forgery.

Bibliography

- W. Diffie, M. E. Hellman. New Directions in Cryptography. *IEEE Transactions of Information Theory*, Vol. IT-22, No.6 (1976): 644-654
- J. Katz, Y. Lindell. *Introduction to Modern Cryptography*. CRC Press, 2nd edition, 2015.
- G. Brassard, (1979). A note on the complexity of cryptography (Corresp.). *IEEE Transactions on Information Theory*, 25(2), 232–233. doi:10.1109/tit.1979.1056010
- B. Schneier. 1995. *Applied Cryptography: Protocols, Algorithms, and Source Code in C* (2nd. ed.). John Wiley & Sons, Inc., USA.

Thank You!