Cryptography

Key Establishment

Prof. Dr. Heiko Knospe

TH Köln – University of Applied Sciences

June 19, 2021

Key Exchange

A key exchange (or key agreement) protocol is a distributed algorithm between two (or more) parties, who exchange messages and finally compute a shared secret key. Key establishment can be based on a pre-shared secret key and may involve a *trusted third party*, as for example the *Kerberos* protocol.

However, in the following we do not assume a pre-distribution of secret keys or a secure channel between the parties.

Nevertheless, the protocol should be secure against *eavesdropping* attacks. The security is defined by a *key distinguishability experiment*, where an adversary eavesdrops the protocol messages and tries to get information about the secret key.

Security Definition

Alice		*Bob*
	$\longrightarrow$	
Derive k	$\xleftarrow{\quad m \quad}$	Derive $k \in \{0,1\}^n$
Adversary		*Challenger*
	$\xleftarrow{\quad 1^n,\, m \quad}$	$b \xleftarrow{\$} \{0,1\}$, $r \xleftarrow{\$} \{0,1\}^n$
	$\xleftarrow{\quad k' \quad}$	$k' = \begin{cases} k & \text{if } b = 1 \\ r & \text{if } b = 0 \end{cases}$
Distinguish	$\xrightarrow{\quad b' \quad}$	Compare b and b', output 1 or 0

Key distinguishability experiment. The adversary gets a copy of the messages m.

EAV Security

The key exchange (KE) advantage of an adversary A is defined as

$$\mathrm{Adv}^{\text{KE-eav}}(A) = \left|\, Pr[b' = b] - Pr[b' \neq b] \,\right| .$$

Definition

A key exchange protocol is *secure in the presence of an eavesdropper* (EAV-secure), if for every probabilistic polynomial time adversary A, the advantage $\mathrm{Adv}^{\text{KE-eav}}(A)$ is negligible in n.

Diffie-Hellman Protocol

The Diffie-Hellman (DH) protocol was a breakthrough in cryptography, because it solved the problem of a secure key exchange over an insecure channel without a pre-distribution of secret keys. The protocol uses a cyclic group G, and the security depends on properties of G (see below).

We explain the protocol for an arbitrary cyclic group G. A standard choice of G are (subgroups of) the multiplicative group $\mathbb{Z}_p^* = GF(p)^*$ for a large prime number p. Other options are $GF(2^m)^*$, and the group of points $E(GF(p))$ on an elliptic curve E over a finite field $GF(p)$ (see chapter on elliptic curves) .

Parameters, Keys and Messages

The Diffie-Hellman key exchange protocol requires a cyclic group G of order q and a generator $g \in G$. The parameters (G, q, g) are public and have to be exchanged in advance.

Alice chooses a private uniform random key $a \in \mathbb{Z}_q$, i.e., a positive integer less than q, and sends the public key $A = g^a \in G$ to Bob. Bob also chooses a private uniform key $b \in \mathbb{Z}_q$, and sends $B = g^b \in G$ to Alice. The communication channel between Alice and Bob can be public.

Alice derives the shared secret key by computing $k = B^a \in G$, and Bob computes $k = A^b \in G$. The scheme is correct since

$$B^a = A^b = g^{ab}.$$

Diffie-Hellman Protocol

$$\begin{array}{ccc}
\textit{Alice} & \xrightarrow{\quad G, q, g \quad} & \textit{Bob} \\
a \xleftarrow{\$} \mathbb{Z}_q,\ A = g^a & \xrightarrow{\quad A \quad} & b \xleftarrow{\$} \mathbb{Z}_q,\ B = g^b \\
k = B^a & \xleftarrow{\quad B \quad} & k = A^b
\end{array}$$

Diffie-Hellman key exchange between Alice and Bob.

Discrete Logarithm Problem

The security of the Diffie-Hellman key exchange is closely related to the *discrete logarithm* (DL) problem. If g is a generator of the cyclic group G and $ord(G) = ord(g) = q$, then

$$G = \{e, g^1, \ldots, g^{q-1}\}.$$

There is a bijection between the elements of G and the exponents $0, 1, \ldots, q-1$. For each $h \in G$, we call the corresponding exponent the *discrete logarithm* of h to the base g and write $\log_g(h)$. One has

$$g^{\log_g(h)} = h.$$

In the Diffie-Hellman protocol, the exponents $a = \log_g(A)$ and $b = \log_g(B)$ are private. An eavesdropper should not be able to compute the discrete logarithm of A or B in an efficient way.

Security of Diffie-Hellman

If the discrete logarithm can be efficiently computed, then DH is broken.

However, the security of DH is actually based on the *Diffie-Hellman problem*. The decisional DH problem (DDH) is to distinguish between the shared secret k and a uniform random element in G, when g^a and g^b are given to an adversary.

Theorem

If the DDH problem is hard relative to the generation of group parameters, then the Diffie-Hellman key exchange protocol is secure in the presence of an eavesdropper (EAV-secure).

Active Attacks against Diffie-Hellman

It is important to observe that the plain Diffie-Hellman protocol does not protect against *active* adversaries. If an attacker is able to replace A and B with their own parameters, then they can perform a *Man-in-the-Middle attack*.

A problem of the plain Diffie-Hellman protocol is the lack of *authenticity*. In practice, public keys are often signed in order to prove their authenticity.

However, some issues remain because at some point a trusted public key (a trust anchor) is needed.

Diffie-Hellman with Subgroups of $\mathbb{Z}_p^*$

The multiplicative group of integers modulo p is the classical choice for Diffie-Hellman. If p is a prime, then $\mathbb{Z}_p^* = GF(p)^*$ is a cyclic group of order $p-1$. Any $g \in \mathbb{Z}_p^*$ generates a cyclic subgroup G of order $q = ord(g) \mid p-1$ and could be used in the protocol, but since the discrete logarithm problem should be hard, q must be large. Furthermore, q should contain a large prime factor. For the hardness of the discrete logarithm and the DDH problem, it is recommended to choose a large prime order q. It is currently recommended that the length of p is at least 2048 bits, and the length of q at least 256 bits.

Suppose that h is a generator of $\mathbb{Z}_p^*$, i.e., $ord(h) = p-1$, and $p-1 = r \cdot q$, where q is a prime. Then $ord(h^r) = \frac{p-1}{r} = q$ and $g = h^r$ generates a cyclic group G of order q.

Diffie-Hellman Groups

In practice, standardized groups and generators are used. A set of pre-defined parameters is called a *Diffie-Hellman group*.

Example: RFC 7919 defines a 2048-bit Diffie-Hellman group. The group order is $ord(g) = q = \frac{p-1}{2}$, a prime number.

```
p = FFFFFFFF FFFFFFFF ADF85458 A2BB4A9A AFDC5620 273D3CF1
    D8B9C583 CE2D3695 A9E13641 146433FB CC939DCE 249B3EF9
    7D2FE363 630C75D8 F681B202 AEC4617A D3DF1ED5 D5FD6561
    2433F51F 5F066ED0 85636555 3DED1AF3 B557135E 7F57C935
    984F0C70 E0E68B77 E2A689DA F3EFE872 1DF158A1 36ADE735
    30ACCA4F 483A797A BC0AB182 B324FB61 D108A94B B2C8E3FB
    B96ADAB7 60D7F468 1D4F42A3 DE394DF4 AE56EDE7 6372BB19
    0B07A7C8 EE0A6D70 9E02FCE1 CDF7E2EC C03404CD 28342F61
    9172FE9C E98583FF 8E4F1232 EEF28183 C3FE3B1B 4C6FAD73
    3BB5FCBC 2EC22005 C58EF183 7D1683B2 C6F34A26 C1B2EFFA
    886B4238 61285C97 FFFFFFFF FFFFFFFF
g = 2
```

Discrete Logarithm Algorithms

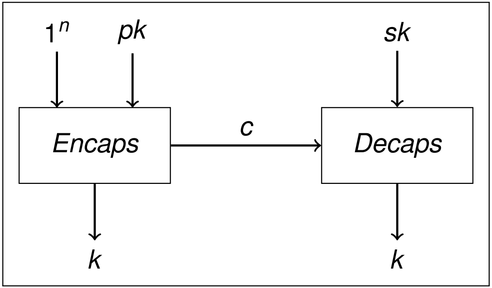
- Compute g^a for $a = 0, 1, \ldots, q-1$ and compare the result with A. The complexity is $O(2^n)$, where $n = \text{size}\,(q)$.
- Babystep-Giantstep: set $m = \lfloor \sqrt{q} \rfloor$ and find $s,\ r \leq m$ such that $a = ms + r$. Compute Ag^{-r} (babysteps) and $(g^m)^s$ (giantsteps) until they match. The complexity is $O(2^{n/2})$.
- Pollard's ρ algorithm also has complexity $O(2^{n/2})$, but requires less storage than the Babystep-Giantstep algorithm.
- Index-Calculus algorithm (for the multiplicative group $\mathbb{Z}_p^*$) with sub-exponential complexity.
- The Number Field Sieve for Discrete Logarithms is currently the best available algorithm for the multiplicative group $\mathbb{Z}_p^*$. It has sub-exponential complexity $O(e^{(c+o(1))\ln(p)^{\frac{1}{3}}\ln(\ln(p))^{\frac{2}{3}}})$.

Key Encapsulation Mechanisms

Key encapsulation is a mechanism, where a *public-key scheme* is leveraged to establish a secret key over an insecure channel. The sender *encapsulates* a randomly chosen secret key using the public key of the receiver. The receiver *decapsulates* the symmetric key using their private key.

Example: Suppose a public-key encryption scheme is given. Bob possesses a key pair (pk, sk) and Alice has a copy of his public key pk. Now Alice generates a uniform random symmetric key k and encrypts k using pk. The ciphertext $c = \mathcal{E}_{pk}(k)$ is sent to Bob, who decrypts c and recovers k using his private key sk, i.e., $k = \mathcal{D}_{sk}(c)$.

Encapsulation and Decapsulation



Key encapsulation mechanism (KEM): Alice takes Bob's public key pk, runs the encapsulation algorithm and keeps k. She sends the ciphertext c to Bob, who obtains k by running the decapsulation algorithm using his private key sk.

Security Definition

A key encapsulation mechanism (KEM) is secure under chosen plaintexts attacks (CPA-secure), if an adversary, who has access to *pk* and *c*, cannot distinguish between the encapsulated key *k* and a uniform random string of the same length. CPA security means that an adversary does not learn a single bit of *k* from the ciphertext *c*.

A stronger notion is security against *adaptive chosen ciphertext attacks* (CCA2 security). The corresponding experiment gives the adversary additional access to a *decapsulation oracle* (before and after obtaining the challenge). However, the adversary must not request the decapsulation of the challenge ciphertext *c*.

RSA Key Encapsulation

Definition

The *RSA key encapsulation mechanism* is defined as follows:

- The key generation algorithm $Gen(1^n)$ outputs the RSA key pair $pk = (e, N)$, $sk = (d, N)$. Fix a hash function $H : \mathbb{Z}_N^* \to \{0, 1\}^n$.
- The encapsulation algorithm *Encaps* takes the public key *pk* as input, chooses a uniform random element $s \in \mathbb{Z}_N^*$, and outputs

$$c = s^e \mod N$$

  as well as the key $k = H(s)$.
- *Decaps* takes c and the private key *sk* as input, computes

$$s = c^d \mod N$$

  and outputs $k = H(s)$.

Security of RSA Key Encapsulation

We infer from the RSA construction that the above encapsulation mechanism is correct. If the RSA assumption holds and the hash function behaves like a random oracle, then CPA security follows from the fact that an adversary is unable to obtain information on s from c. Now s is unknown, and so $H(s)$ is uniform random for an adversary.

Note that padding schemes like OAEP are not required here, since s *is uniform* in $\mathbb{Z}_N^*$.

Furthermore, the RSA key encapsulation mechanism turns out to be CCA2-secure, if the hash function has no weaknesses.

Theorem

If the RSA assumption holds and H is modeled as a random oracle, then the RSA key encapsulation mechanism is CCA2-secure.

Diffie-Hellman Key Encapsulation

Definition

The *Diffie-Hellman KEM* is defined as follows:

- The key generation algorithm *Gen* takes 1^n as input and outputs a cyclic group G of order q with $n = \text{size}\,(q)$, a generator $g \in G$, a uniform random element $b \in \mathbb{Z}_q$ and $B = g^b$. The public key is $pk = (G, q, g, B)$ and the private key is $sk = (G, q, g, b)$. Also fix a hash function $H : G \to \{0,1\}^n$.
- The encapsulation algorithm takes pk as input, chooses a uniform random element $a \in \mathbb{Z}_q$, and outputs the ciphertext $c = A = g^a$ as well as the key $k = H(B^a)$.
- The decapsulation algorithm *Decaps* takes sk and c as input, and outputs the key $k = H(c^b) = H(A^b)$.

Security of Diffie-Hellman Key Encapsulation

It is not surprising that the security of DH key encapsulation depends on the Diffie-Hellman assumption and properties of the hash function.

Theorem

If the DDH assumption (or the weaker computational Diffie-Hellman assumption) holds and H is modeled as a random oracle, then the Diffie-Hellman key encapsulation mechanism is CPA-secure.

Hybrid Encryption Schemes

Definition

Suppose a key encapsulation mechanism (KEM) and a symmetric-key encryption scheme are given. Then a *hybrid encryption scheme* can be defined as follows:

- Run the key generation algorithm of the KEM on input 1^n and output the keys *pk* and *sk*.
- The hybrid encryption algorithm takes the public key *pk* and a message $m \in \{0,1\}^*$ as input. *Encaps* computes

  $$(c,k) \leftarrow Encaps_{pk}(1^n).$$

  Then the symmetric encryption algorithm $\mathcal{E}$ takes k and the plaintext m as input and computes $c' = \mathcal{E}_k(m)$. Finally, output the ciphertext (c, c').

Decryption using a Hybrid Scheme

Definition

- The hybrid decryption algorithm takes the private key *sk* and the ciphertext (c, c') as input. First, the symmetric key is retrieved by computing
$$k = Decaps_{sk}(c).$$
Then decrypt c' and output the plaintext $m = \mathcal{D}_k(c')$. If c or c' are invalid then output $\perp$.

So hybrid encryption schemes combine a key encapsulation mechanism (KEM) and a symmetric encryption scheme. Hybrid schemes are public-key schemes, which leverage symmetric schemes and can efficiently encrypt mass data.

Security of Hybrid Encryption

Theorem

Consider a hybrid encryption scheme as defined above.

1. *If the KEM is CPA-secure and the symmetric scheme is EAV-secure, then the corresponding hybrid scheme is CPA-secure.*
2. *If the KEM and the symmetric scheme are both CCA2-secure, then the corresponding hybrid scheme is CCA2-secure.*

Example: The hybrid encryption scheme that combines RSA key encapsulation and an authenticated encryption scheme (AES/GCM is a candidate) is CCA2-secure, if the RSA assumption holds and the hash function is modeled as a random oracle.