Key Management and the Public-Key Revolution (密钥管理与公钥加密变革)

Sheng Zhong Yuan Zhang

Computer Science and Technology Department
Nanjing University

1. Key Management in Private-key Cryptography
2. A Partial Solution: Key-Distribution Centers
3. Achieving Private Communications without Pre-existence of Secure Channel
4. The Public-Key Revolution

A question that we have deferred

- We have seen private-key cryptography allows two parties who share a secret key to conduct secret, authenticated communications.
- However, there is one question that we have not discussed:

"How can the parties share a secret key in the first place?"

Assuming a secure channel is there

Our problem is easy when there is a secure channel between the two parties, e.g.

- At some time, Alice and Bob are co-located, at which point they agree on a key secretly.
- Or they can use a trusted courier service.

Q: Does the assumption make private-key cryptography useless?

"We design priv-key cryptosystem to implement secure communication, but here we assume the existence of secure communication channels to implement priv-key cryptosystem ..."

Private-key cryptography is meaningful

Private-key cryptography is meaningful, EVEN when a secure channel exists or existed:

- The parties may have a secure channel at one point in time, but not indefinitely.
- Utilizing the secure channel may be slower and more costly than communicating over an insecure channel.

When the number of parties is big

Assuming N parties need to establish pairwise secure communications. Relying on pairwise secure channels to share secret keys may not be a good idea especially when N is large:

- (*Key distribution* issues): $\frac{N(N-1)}{2}$ secure channels are required.
- (*Key storage and management* issues): Each party needs to store $N-1$ secret keys. Storing a large number of keys, and keeping them safe are troublesome.

A partial solution: key-distribution centers

One way to address the issues is to use a key-distribution center (KDC):

- KDC generates shared keys for all parties.
- KDC sends keys to each party via a secure channel between KDC and this party.

Using a KDC, reduces the total number of secure channels, but still has the same key storage and management issue.

Utilizing the KDC in an online fashion

A better approach that avoids requiring parties to store and manage multiple keys, is to utilize the KDC in an online fashion:

- Shared keys are generated on demand.
- A new, random, short-term key, called "session key" is generated every time two parties want to communicate.
- For security, when communication is over, both parties need to erase the session key.
- Each party needs to store/manage one long-term shared key between itself and the KDC.

A secure key-distribution protocol using a KDC

To reduce the load on the KDC, the Needham-Schroeder key-distribution protocol is often used in practice.

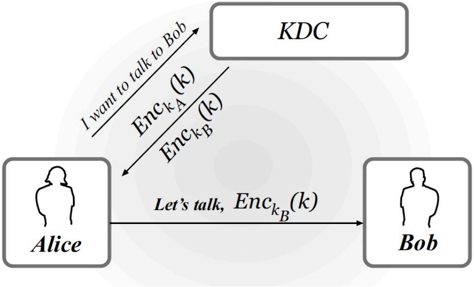

图 1: A general template for Needham-Schroeder key-distribution

Drawbacks to relying on KDCs

KDC-based solutions are commonly used in practice, but they have several drawbacks:

- A successful attack on the KDC will result in a complete break of the system.
- The KDC is a single point of failure.
- Pre-existence of private, authenticated channels are required.
- A trusted entity (KDC) is required.

The Diffie-Hellman key-exchange protocol

Q: Can we design secure key-exchange protocol without relying on KDCs?
YES, we can use Diffie-Hellman key-exchange protocol:

- The protocol was proposed by Whitfield Diffie and Martin Hellman in 1976.
- It allows two parties to share a secret key on their own via a public channel.
- Diffie and Hellman's work is considered to be the first steps toward shifting private-key cryptography to public-key cryptography.

The detailed construction of D-H key-exchange protocol

Let $\mathcal{G}$ be a PPT algorithm that, on input 1^n, output a cyclic group $\mathbb{G}$, its order q (with $|q| = n$), and a generator $g \in \mathbb{G}$, the D-H key-exchange protocol is constructed as follows.

CONSTRUCTION 10.2

- **Common input:** The security parameter 1^n
- **The protocol:**
  1. Alice runs $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$.
  2. Alice chooses a uniform $x \in \mathbb{Z}_q$, and computes $h_A := g^x$.
  3. Alice sends $(\mathbb{G}, q, g, h_A)$ to Bob.
  4. Bob receives $(\mathbb{G}, q, g, h_A)$. He chooses a uniform $y \in \mathbb{Z}_q$, and computes $h_B := g^y$. Bob sends h_B to Alice and outputs the key $k_B := h_A^y$.
  5. Alice receives h_B and outputs the key $k_A := h_B^x$.

The Diffie–Hellman key-exchange protocol.

We illustrate the D-H key-exchange protocol in Figure 2.

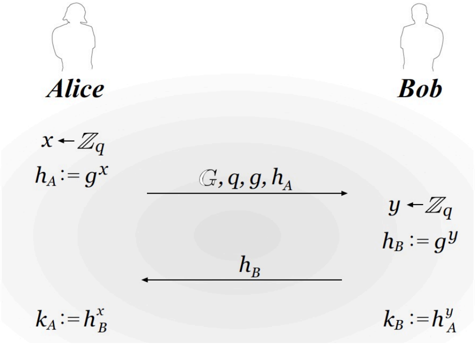

图 2: An illustratiion of D-H key-exchange protocol

Correctness and security analyses

- **Correctness**: It is easy to see

$$k_A = h_B^x = (g^y)^x = (g^x)^y = h_A^y = k_B.$$

- **Security** (against an eavesdropper *Eve*):
  Eve sees

$$\mathbb{G}, q, g, h_A = g^x, h_B = g^y,$$

can she learn any knowledge about $k_A = k_B = g^{xy}$?

Formalizing the security definition of key-exchange protocol

To prove the security, we first specify our security requirements by defining an experiment:

The key-exchange experiment $\mathsf{KE}^{eav}_{\mathcal{A},\Pi}(n)$:

1. Two parties holding 1^n execute the key-exchange protocol Π. This results in a transcript **trans** containing all the messages sent by the parties, and a key k output by each of the parties.
2. A uniform bit $b \in \{0,1\}$ is chosen. If $b = 0$ set $\hat{k} := k$, and if $b = 1$ then choose $\hat{k} \in \{0,1\}^n$ uniformly at random.
3. $\mathcal{A}$ is given **trans** and $\hat{k}$, and outputs a bit b'.
4. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise. (In case $\mathsf{KE}^{eav}_{\mathcal{A},\Pi}(n) = 1$, we say that $\mathcal{A}$ succeeds.)

Formalizing the security definition of key-exchange protocol

Based on the $\mathsf{KE}^{eav}_{\mathcal{A},\Pi}(n)$, we define the security of a key-exchange protocol:

DEFINITION 10.1

A key-exchange protocol Π is **secure in the presence of an eavesdropper** if for all PPT adversaries $\mathcal{A}$ there is a negligible function *negl* such that

$$Pr[\mathsf{KE}^{eav}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + negl(n).$$

Proving D-H key-change protocol is secure

For simplicity, we prove the security using a modified version of experiment $\hat{\mathsf{KE}}^{eav}_{\mathcal{A},\Pi}(n)$ in which it is required that the shared key is indistinguishable from a uniform element of $\mathbb{G}$ rather than from a uniform n-bit string.

THEOREM 10.3

If the decisional Diffie-Hellamn problem is hard relative to $\mathcal{G}$, then the Diffie-Hellman key-exchange protocol Π is secure in the presence of an eavesdropper (with respect to the modified experiment $\hat{\mathsf{KE}}^{eav}_{\mathcal{A},\Pi}(n)$).

More about D-H key exchange

- We only prove the D-H key exchange protocol is secure against a static adversary (an eavesdropper).
- D-H key exchange is vulnerable to an active attack called man-in-the-middle attack. Due to this reason, D-H key exchange is typically not used ALONE in practice.

Public-key cryptography

In addition to key-exchange, Diffie and Hellman also introduced the notion of **public-key** or **asymmetric** cryptography.

	Private-Key Setting	Public-Key Setting
Secrecy	Private-key encryption	Public-key encryption
Integrity	Message authentication codes	Digital signature schemes

图 3: Cryptographic primitives in the private-key and the public-key settings.

The public-key cryptography vs the private-key cryptography

We summarize a few advantages of public-key cryptography compared with private-key cryptography:

- Public-key encryption allows key distribution to be done over public (but authenticated) channels, which simplifies the distribution and updating of key material.
- Public-key cryptography reduces the need for users to store many secret keys.
- Public-key cryptography is more suitable for open environment where parties who have never previously interacted want the ability to communicate securely.

An interesting reading related to D-H key-exchange protocol

"There have been rumors for years that the NSA can decrypt a significant fraction of encrypted Internet traffic. In 2012, James Bamford published an article quoting anonymous former NSA officials stating that the agency had achieved a "computing breakthroug" that gave them "the ability to crack current public encryption." The Snowden documents also hint at some extraordinary capabilities ...
The key is, somewhat ironically, Diffie-Hellman key exchange, an algorithm that we and many others have advocated as a defense against mass surveillance..."

———from "How is NSA breaking so much crypto?" (Oct. 14, 2015) BY A. HALDERMAN and N. HENINGER