Cryptography

Corso di Laurea Magistrale in Informatica

The Public-Key Revolution

Ugo Dal Lago

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA
informatiques mathématiques Inria

Academic Year 2021-2022

Limitations of Private-Key Cryptography

- Although symmetric (or private-key) cryptography can solve the authentication and confidentiality problems, it has some fundamental limitations.
  1. **Key Distribution**
     - In private-key cryptography, a key must be shared between the sender and the receiver.
     - Certainly, a non secure channel cannot be used, and exchanging the key "manually" is only realistic for certain mission-cricial applications
  2. **Key Storage**
     - If the number of involved users is equal to U, the number of needed keys is $\Theta(U^2)$ and the entry of a new user requires $U - 1$ new keys.
     - This is a problem when U increases.
  3. **Managing Open Systems**
     - In open systems, where new users can connect to the system dynamically, symmetric cryptography offers no solution.

A Partial Solution: KDC

- A quite simple way to solve the first two problems above is to provide the (multi-party) system with a **key distribution centre** (KDC), where:
  - All users share a secret key with the KDC.
  - Whenever a user A wants to communicate with a user B, A sends a special message to the KDC requiring a special session-key.
  - Then, the KDC sends both A and B a new key, which is deleted at the end of the communication session.

A Partial Solution: KDC

- A quite simple way to solve the first two problems above is to provide the (multi-party) system with a **key distribution centre** (KDC), where:
  - All users share a secret key with the KDC.
  - Whenever a user A wants to communicate with a user B, A sends a special message to the KDC requiring a special session-key.
  - Then, the KDC sends both A and B a new key, which is deleted at the end of the communication session.
- The *advantages* of this approach are:
  - Each user needs to store only one key (i.e. the one shared with the KDC), and the session keys.
  - When a new user joins, it is enough to bother the KDC, and not all the other users.

A Partial Solution: KDC

- A quite simple way to solve the first two problems above is to provide the (multi-party) system with a **key distribution centre** (KDC), where:
  - All users share a secret key with the KDC.
  - Whenever a user A wants to communicate with a user B, A sends a special message to the KDC requiring a special session-key.
  - Then, the KDC sends both A and B a new key, which is deleted at the end of the communication session.
- The *advantages* of this approach are:
  - Each user needs to store only one key (i.e. the one shared with the KDC), and the session keys.
  - When a new user joins, it is enough to bother the KDC, and not all the other users.
- However, there are still some *disadvantages*:
  - Forcing the KDC means forcing the whole system.
  - If the KDC is a **point-of-failure**.

A Partial Solution: KDC

- How are session keys distributed?
- There are many protocols for key distribution through KDC.
- One of these is a protocol due to Needham and Schroeder (also known as the NS protocol):

$$\begin{array}{cl} A & \texttt{Send } (A, B) \texttt{ to } KDC \\ KDC & \begin{array}{l} k \leftarrow Gen \\ \texttt{Send } (Enc(k_A, k), Enc(k_B, k)) \texttt{ to } A \end{array} \\ A & \texttt{Send } Enc(k_B, k) \texttt{ to } B \end{array}$$

- The NS protocol is, however, vulnerable to attacks on the authentication side, if a CPA-secure encryption scheme is used.
- NS is the basis of the Kerberos protocol.

The Public-Key Revolution

- The public key revolution began in 1976, when Whitfield Diffie and Martin Hellman published an article, “New Directions in Cryptography”.
  - Before 1976, it was universally accepted that a shared key was a necessary condition for cryptography.

The Public-Key Revolution

- The public key revolution began in 1976, when Whitfield Diffie and Martin Hellman published an article, “New Directions in Cryptography”.
  - Before 1976, it was universally accepted that a shared key was a necessary condition for cryptography.
- Diffie and Hellman observed that there can be an asymmetric setting, where there are not *one* but *two* keys:
  - An **encryption key**, used by the sender.
  - A **decryption key**, used by the receiver.

The Public-Key Revolution

- The public key revolution began in 1976, when Whitfield Diffie and Martin Hellman published an article, "New Directions in Cryptography".
  - Before 1976, it was universally accepted that a shared key was a necessary condition for cryptography.
- Diffie and Hellman observed that there can be an asymmetric setting, where there are not *one* but *two* keys:
  - An **encryption key**, used by the sender.
  - A **decryption key**, used by the receiver.
- Surprisingly, the security of the encryption scheme must also hold against adversaries who know the encryption key, which is also called **public key**.
  - In this way, a secure channel for keys distribution is no longer required: the sender makes the encryption key *public*.

The Public-Key Revolution

- The three limitations of symmetric encryption are solved as follows:
  1. Key distribution can be *on public channels*, even if authenticated.
  2. Each of the parties involved in the communication must keep secret *a single* key.
  3. Open systems are handled in a satisfactory way: each new user *creates a public key* and makes it available to other users.

The Public-Key Revolution

- The three limitations of symmetric encryption are solved as follows:
  1. Key distribution can be *on public channels*, even if authenticated.
  2. Each of the parties involved in the communication must keep secret *a single* key.
  3. Open systems are handled in a satisfactory way: each new user *creates a public key* and makes it available to other users.
- Diffie and Hellman proposed three public key primitives:
  - **Public-Key Cryptography**, which we discussed.
  - **The digital signature**, or the asymmetric equivalent of MACs.
  - **An interactive key-exchange protocol**, through which two parties that do not share any secret information can generate a key securely.

Key-Exchange Protocols

- A *key-exchange protocol* is a set of rules Π that specifies how two parties A and B should exchange a certain number of messages.
  - At the end of the protocol A constructed a key k_A, while B has constructed k_B.
  - The protocol is **correct** if $k_A = k_B$.
- The security of a protocol for key exchange is usually formulated, as usual, through an experiment:

$\mathsf{KE}^{eav}_{A,\Pi}(n)$:
$(transcript, k) \leftarrow \Pi(1^n)$;
$b \leftarrow \{0,1\}$;
if $b = 0$ **then**
$\quad k^* \leftarrow \{0,1\}^n$
else
$\quad k^* \leftarrow k$
$b^* \leftarrow A(transcript, k^*)$;
Result: $\neg(b \oplus b^*)$

Π is *secure* iff for every PPT A exists $\varepsilon \in \mathcal{NGL}$ s.t. $Pr(\mathsf{KE}^{eav}_{A,\Pi}(n) = 1) = \frac{1}{2} + \varepsilon(n)$.

The Diffie-Hellman Protocol

- Diffie-Hellman protocol is built on principles similar to those that led the two researchers to formulate the assumptions that are named after them.
- The protocol is defined as follows:

$$
\begin{array}{ll}
 & (\mathbb{G}, q, g) \leftarrow \mathsf{GenCG}(1^n) \\
A & x \leftarrow \mathbb{Z}_q \\
 & h_1 \leftarrow g^x \\
 & \texttt{Send } (\mathbb{G}, q, g, h_1) \texttt{ to } B \\
 & \\
 & y \leftarrow \mathbb{Z}_q \\
 & h_2 \leftarrow g^y \\
B & \texttt{Send } h_2 \texttt{ to } B \\
 & \texttt{Output } h_1^y \\
 & \\
A & \texttt{Output } h_2^x
\end{array}
$$

The Diffie-Hellman Protocol

- Diffie-Hellman protocol is built on principles similar to those that led the two researchers to formulate the assumptions that are named after them.
- The protocol is defined as follows:

$$
\begin{array}{ll}
A & \begin{array}{l} (\mathbb{G}, q, g) \leftarrow \mathsf{GenCG}(1^n) \\ x \leftarrow \mathbb{Z}_q \\ h_1 \leftarrow g^x \\ \texttt{Send } (\mathbb{G}, q, g, h_1) \texttt{ to } B \end{array} \\
 & \\
B & \begin{array}{l} y \leftarrow \mathbb{Z}_q \\ h_2 \leftarrow g^y \\ \texttt{Send } h_2 \texttt{ to } B \\ \texttt{Output } h_1^y \end{array} \\
 & \\
A & \texttt{Output } h_2^x
\end{array}
$$

$$
\begin{aligned}
k_B &= h_1^y = (g^x)^y = g^{xy} \\
k_A &= h_2^x = (g^y)^x = g^{xy}
\end{aligned}
$$

On Diffie-Hellman Protocol's Security

Theorem

If DDH is difficult relative to GenCG, *then* Π *is secure (with respect to* KE^{eav}*).*

On Diffie-Hellman Protocol's Security

Theorem

If DDH is difficult relative to GenCG*, then* Π *is secure (with respect to* KE^{eav}*).*

- We note that the DDH Assumption was formulated **after** the introduction of the aforementioned protocol and **exactly** in order to understand which property was sufficient for the relative security.
  - What do we obtain? We have identified a property, that can be then related to the discrete logarithm.

On Diffie-Hellman Protocol's Security

Theorem

If DDH is difficult relative to GenCG, *then* Π *is secure (with respect to* KE^{eav}*).*

- We note that the DDH Assumption was formulated **after** the introduction of the aforementioned protocol and **exactly** in order to understand which property was sufficient for the relative security.
  - What do we obtain? We have identified a property, that can be then related to the discrete logarithm.
- In addition to the "passive" attacks we have considered so far, there are others, which the experiment KE^{eav} does not catch:
  - **Impersonation Attacks**
  - **Man-in-the-Middle Attacks**, against which Diffie-Hellman is known to be not secure.