Public Key Cryptography

Key Exchange

All the ciphers mentioned previously require keys known **a-priori** to all the users, before they can encrypt and decrypt.

To communicate securely, they need to have a **common key, known only to them**.

Possible key exchanges:

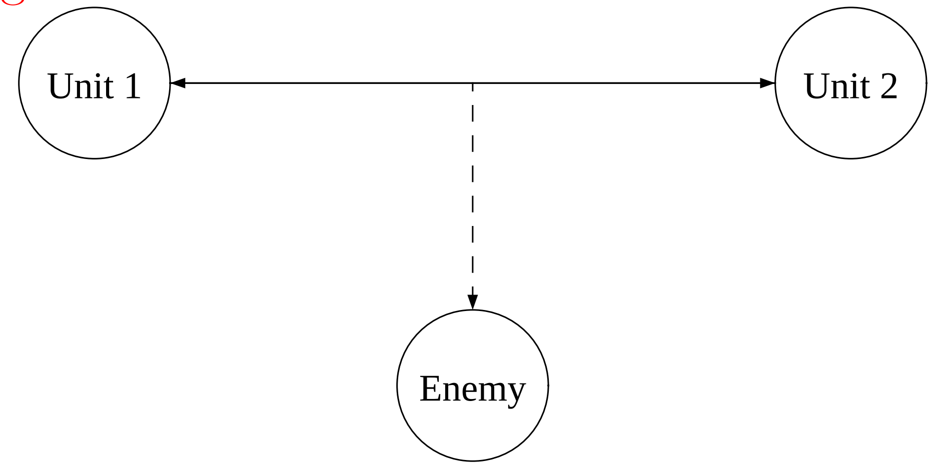
1. An a-priori meeting to exchange keys.
2. Sending the keys by a special courier.
3. Using an already existing common key, to encrypt additional keys.

Key Exchange (cont.)

The problem: A meeting is required between the users (or their couriers), in order to be able to communicate securely.
Conclusion: **Two users who have never met cannot communicate securely!**

Example

Two armies during a war.



The enemy wins, since it found all the keys that our army use for communication.
Our army observes that the enemy found the keys. Our army must choose new keys, immediately, but it is impossible to send couriers through the enemies lines.
If our army does not agree on new common secret keys, we will certainly loose.

Public Key Cryptography - Background

The model: A network of N users.
In order to let any user to communicate securely to any other user, $\binom{N}{2} = O(N^2)$ keys should be distributed **in advance** in a **secure way**.
Possible solution: Trusted center.
Each user sets in advance a secret key with the trusted center. When user A wishes to communicate with user B, he chooses a new common key and sends it (encrypted under A's key) to the center, who forwards it (encrypted under B's key) to B.

Public Key Cryptography - Background (cont.)

Drawbacks:

1. There should be somebody that all the users trust.
2. All the users should have a common key with the trusted center **in advance**.
3. The center can always understand all the users' messages (and can fake messages).

Requirements for Key Exchange

We prefer a solution that allows two users who

- have **never** met, and
- do not have any a-priori common information,

to decide on a common key

- known only to them, and
- unknown to anybody else, even to passive eavesdroppers who listen to their communication (but cannot modify it).

Public Key Cryptography 1

See:
Diffie and Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, Vol. IT-22, No. 6, Nov. 1976.

Trapdoor Problems

Basing the solution on the complexity of problems, which are easy to solve for the legal users, but are very difficult to the eavesdroppers.

Such problems are called **trapdoor problems**.

They allow to exchange secure common keys using insecure channels!

Diffie-Hellman Key Exchange Protocol

Based on number theory assumptions.

The basic idea:

1. It is easy to calculate
$$a^x \bmod q$$
for any a, x and q.
2. There is no efficient algorithm which computes x given a, q, and
$$a^x \bmod q.$$
This is the **discrete logarithm (DLOG)** problem.

Diffie-Hellman Key Exchange Protocol (cont.)

Notations:

- Denote x in binary representation as $x = x_{n-1}x_{n-2}\ldots x_1x_0$, where $x = \sum_{i=0}^{n-1} x_i 2^i$.
- Let q be a large prime number.
- All the multiplications from now on are modulo q.

Diffie-Hellman Key Exchange Protocol (cont.)

Preparations:
System parameters common to all users:

- Let q be a large prime number ($q > 2^{400}$).
- Let a an integer $1 < a < q$.

Public and private keys: Each user U:

- chooses a random value X_U ($1 < X_U < q$) and keeps it secret.
- publishes $Y_U = a^{X_U} \bmod q$.

Diffie-Hellman Key Exchange Protocol (cont.)

The key exchange:

Two users A,B who wish to have a common key, known only to them:

- A calculates $K = (Y_B)^{X_A} \bmod q$.
- B calculates $K = (Y_A)^{X_B} \bmod q$.

A and B result with the same common key K:

$$\begin{aligned}(Y_B)^{X_A} &\equiv (a^{X_B})^{X_A} \equiv a^{X_B X_A} \equiv \\ &\equiv a^{X_A X_B} \equiv (a^{X_A})^{X_B} \equiv (Y_A)^{X_B} \pmod{q}.\end{aligned}$$

Diffie-Hellman Key Exchange Protocol (cont.)

Security:

1. The secret keys are secure: if one can compute the secret key X_A of A from $Y_A = a^{X_A} \bmod q$, he solved the DLOG problem, and we assume it is difficult.

2. Can somebody compute the common key of A and B from their published keys (without computing the secret keys)? The problem of computing $a^{X_A X_B} \bmod q$ from a, $a^{X_A} \bmod q$ and $a^{X_B} \bmod q$ is assumed to be as difficult as DLOG.

Public Key Cryptography

Solution: Each user chooses two keys:

- A **public key** K_E which he publishes. This key is publicly known. The public key is used for encryption.
- A **secret key** K_D which he keeps secret (also called **private key**). The secret key is used for decryption.

Public Key Cryptography (cont.)

Everybody (B) who knows A's public key can encrypt messages to A by

$$C = E_{K_E}(M)$$

but only A can decrypt it by

$$M = D_{K_D}(C).$$

Even B cannot decrypt messages he encrypted under A's public key (unless he keeps records of the messages he encrypted).

Public Key Cryptography (cont.)

Required properties:

1. the encryption and decryption functions E, D are publicly known and easy to compute.
2. It is possible to generate pairs of keys K_E and K_D which satisfy $\forall M : D_{K_D}(E_{K_E}(M)) = M$.
3. Without the knowledge of K_D, it is difficult to decrypt C, given only the public key K_E (even though encryption is easy).

Result: It is difficult compute K_D from K_E (even if the attacker have also many encrypted messages).

The Key Generation

It is difficult to calculate K_D from K_E. In many cases it is also difficult to calculate K_E from K_D.

We need a trapdoor function E_{K_E}: easy to calculate, but difficult to invert.

We should use an **efficient function** $G(X)$ which takes a random X and generates both keys **simultaneously**.

The Key Generation (cont.)

Usage: Each user U generates a pair of random keys

$$(K_E, K_D) = G(\text{random } X),$$

and publishes K_E (in a public file). K_D is kept secret.
When another user A wishes to send a message M to U, he requests U's public key K_E (from the public file), computes

$$C = E_{K_E}(M),$$

and sends C to U. U decrypts by

$$M = D_{K_D}(C).$$

The Key Generation (cont.)

Properties:

1. Everybody can send messages to U, without the need to distribute a common secret key in advance.
2. Only U can decrypt.
3. The center (maintaining the public file) cannot decrypt (if he is only trusted to send U's real key to A).
4. There is no need to set common secret keys in advance. A and B can communicate securely after they request each others key from the center. The communication with the center does not have to be encrypted.

The Key Generation (cont.)

5. Two users who have never met, can communicate securely even without a trusted center. However, they **cannot authenticate each other without a trusted center**.

6. The center can generate **certificates** for the users: he signs the users identity together with their public key. The users can then receive the certificates directly from the receivers, rather than asking the center for the public keys of the receivers. Then, they verify with the center's well-known public key.

Shortened Notation

After the user U chooses his pair of keys K_E and K_D, and publishes his public key K_E, we denote his encryption function (known to everybody) by

$$E_U(\cdot) = E_{K_E}(\cdot)$$

and his decryption function (whose key is secret) by

$$D_U(\cdot) = D_{K_D}(\cdot).$$

For every user U, E_U can be computed by all the users, but D_U can be computed only by U.

Remark

Remark: Diffie and Hellman did not suggest a good implementation of a public key cryptosystem. Only after they published their paper, several public key cryptosystems were suggested, such as Merkle-Hellman's knapsack cryptosystem (broken later) and RSA.
They predicted that public key cryptosystems will be based on the following problems (as was the case later in the listed systems)

1. Knapsack (an NP-complete problem): such as Merkle-Hellman.
2. Factoring: RSA, etc.
3. Discrete logarithm: ElGamal, DSS, etc.

Public Key Signatures

The encryption function E_U is **1-1**. If it is also **onto**, $E_U : \mathcal{M} \rightarrow \mathcal{M}$, it can be used for signatures as well.

U signs a message M by

$$S = D_U(H(M)),$$

where H is a collision free hash function.

Everybody can verify the originality of the signature S by checking whether

$$E_U(S) \stackrel{?}{=} H(M).$$

Public Key Signatures (cont.)

Claim: $E_U(D_U(X)) = X$ for every X.
Proof: Let X be some value. From the definition, $D_U(E_U(Y)) = Y$ for every Y, and in particular for $Y = D_U(X)$.
Therefore, $D_U(E_U(D_U(X))) = D_U(E_U(Y)) = Y = D_U(X)$.
Since E_U is 1-1, then D_U is 1-1, and $E_U(D_U(X)) = X$. QED
Secret signatures: If U wishes to keep the signature (sent to B) secret, he sends $E_B(S) = E_B(D_U(H(M)))$. B will decrypt and get S, and then will be able to verify it as before.