Public and Private Key Protocols

We describe the standard symmetric and public key cryptography protocols, then describe a hybrid protocol, which could be used to exchange messages efficiently using public key cryptography for key exchange and symmetric key encryption for the message content. Each of these protocols describes the sequence of steps by which Alice can send a secure message to Bob. The word *public* or *sends* refer to events or information to which an adversary can have access. The word *private* refers to information or an exchange which is protected from evesdropping.

Symmetric Cryptography Protocol

Initial setup:
1. Alice and Bob publicly agree on a cryptosystem.

For each message Alice → Bob:
1. Alice and Bob privately agree on a key.
2. Alice enciphers her plaintext using the agreed key.
3. Alice sends the ciphertext message to Bob.
4. Bob deciphers the ciphertext message to obtain the plaintext.

The need to exchange keys for every exchange was a critical problem prior to the theoretical solution with public key cryptography by Diffie and Hellman. The public key protocol proceeds as follows, requiring only one initial public key exchange.

Public Key Cryptography Protocol

Initial setup:
1. Alice and Bob publicly agree on a cryptosystem.
2. Bob sends Alice his public key.

For each message Alice → Bob:
1. Alice enciphers her plaintext using Bob's public key.
2. Alice sends the ciphertext message to Bob.
3. Bob deciphers the ciphertext message using his private key.

As noted in the preceeding discussion, public key cryptosystems are several orders of magnitude slower than comparable symmetric key cryptosystems. Moreover, a public key cryptosystem is susceptible to chosen plaintext attacks: an adversary can create a lookup table of pairs $(E_K(M), M)$ for the known public key K. Therefore the amount of public key ciphertext which is transmitted should be strictly limited.

These considerations lead to the following hybrid protocol, in which the public key cryptosystem is used for key exchange and a fast symmetric key cryptosystem is used for message encryption.

Hybrid Cryptographic Protocol

Initial setup:

1. Alice and Bob publicly agree on a public key cryptosystem and a symmetric key cryptosystem.
2. Bob sends Alice his public key.

For each message Alice → Bob:

1. Alice generates a random session key K.
2. Alice enciphers K using Bob's public key.
3. Alice enciphers the plaintext message using K.
4. Alice sends Bob the enciphered session key and the ciphertext message.
5. Bob deciphers the enciphered session key using his private key.
6. Bob deciphers the ciphertext message using the session key.

An additional layer can be added to the hybrid exchange, involving a trusted authority, which carries out the function of certifying the identity of individuals and their public keys, managing a database of public keys, and handling expiration of public keys. The public key exchange step, in which Bob sends Alice his public key, in the hybrid protocol can therefore be replaced with any of the following:

- Alice gets Bob's key from Bob.
- Alice gets Bob's key from a trusted authority's database.
- Alice gets Bob's key from her private database.

Trapdoor one-way functions

Several years elapsed between when Diffie and Hellman presented their *New directions in cryptography* with the theory of public key cryptography based on trapdoor one-way functions, and the discovery of a practical example of trapdoor one-way functions for cryptographic use. The principal public key cryptosystems in use today, based on trapdoor one-way functions, are the RSA cryptosystem and the ElGamal cryptosystem. We describe the underlying mathematical functions used in the RSA and ElGamal constructions.

RSA.

The trapdoor one-way function used in RSA is a fixed exponentiation in $\mathbb{Z}/n\mathbb{Z}$ for a composite integer n.

$$\begin{aligned} \mathbb{Z}/n\mathbb{Z} &\longrightarrow \mathbb{Z}/n\mathbb{Z} \\ m &\longmapsto m^e \end{aligned}$$

The public data is the exponent e and the modulus n, which is presumed to be of the form pq for distinct primes p and q. The presumed difficulty of inverting this function is based

on the difficulty of factoring n, or on the difficulty of a weaker problem version called the RSA problem. The trapdoor for this function is the knowledge of the factorization of n.

ElGamal.

The ElGamal function is based on the exponentiation of a fixed multiplicative generator a for $\mathbb{F}_p^*$, the group of nonzero elements of the finite field of p elements. The map takes m to the power a^m mod p:

$$\begin{aligned} \mathbb{Z}/(p-1)\mathbb{Z} &\longrightarrow \mathbb{F}_p^* \\ m &\longmapsto a^m \end{aligned}$$

The public data is the generator a and the prime p, and inverting the cryptographic function is solved by the discrete logarithm problem – finding m given a and a^m, or on a weaker problem called the Diffie Hellman problem.

Elliptic Curves.

Alternatively, the ElGamal construction can be solved using the discrete logarithm problem on an elliptic curve E over a finite field $\mathbb{F}_q$. An elliptic curve is defined by an equation of the form

$$y^2 + (a_1x + a_3)y = x^3 + a_2x^2 + a_4x + a_6$$

with fixed coefficients a_1, a_2, a_3, a_4, a_6 in $\mathbb{F}_q$. The set $E(\mathbb{F}_q)$ of points (x, y) in $\mathbb{F}_q^2$ solving this equation plus a distiguished point O at infinity forms a group law under an addition, which we denote $+$. For groups $E(\mathbb{F}_q)$ and $\mathbb{F}_p^*$ of comparable size, the discrete logarithm problem on the elliptic curve is believed to be a harder problem than the classical one in $\mathbb{F}_p^*$. Elliptic curves will not be a topic of discussion in this course, but are becoming commonplace in implementations of smart cards and cryptography for low-power devices.