2021-03-16 theory

March 2021

1 Key exchange protocol

Key exchange (also key establishment) is a method in cryptography by which cryptographic keys are exchanged between two parties, allowing use of a cryptographic algorithm. The key exchange problem describes ways to exchange whatever keys or other information are needed for establishing a secure communication channel so that no one else can obtain a copy. Historically, before the invention of public-key cryptography (asymmetrical cryptography), symmetric-key cryptography utilized a single key to encrypt and decrypt messages. For two parties to communicate confidentially, they must first exchange the secret key so that each party is able to encrypt messages before sending, and decrypt received ones. This process is known as the key exchange.

1.1 Diffie–Hellman key exchange

In 1976, Whitfield Diffie and Martin Hellman published a cryptographic protocol called the Diffie–Hellman key exchange (D–H) based on concepts developed by Hellman's PhD student Ralph Merkle. The protocol enables users to securely exchange secret keys even if an opponent is monitoring that communication channel. The D–H key exchange protocol, however, does not by itself address authentication (i.e. the problem of being sure of the actual identity of the person or 'entity' at the other end of the communication channel).

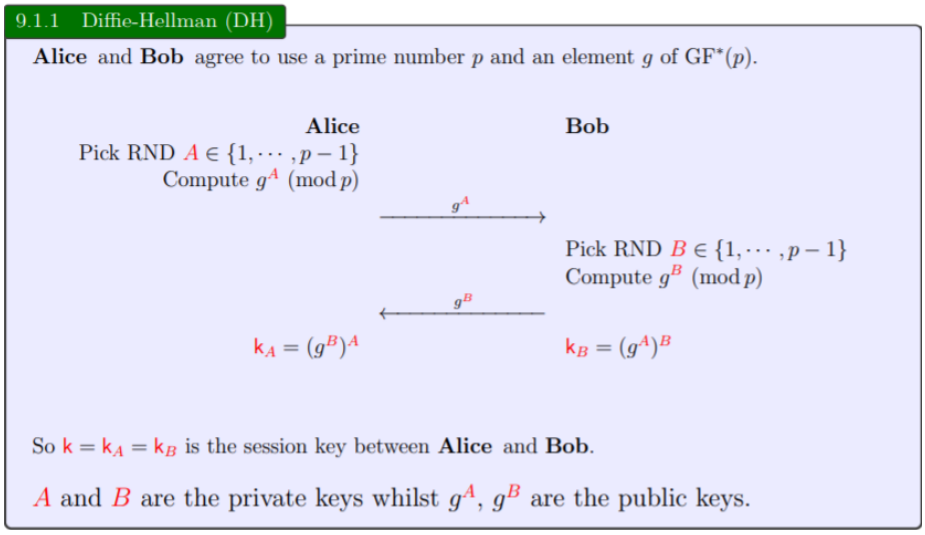

Figure 1: In the Diffie–Hellman key exchange scheme, each party generates a public/private key pair and distributes the public key. After obtaining an authentic copy of each other's public keys, Alice and Bob can compute a shared secret offline. The shared secret can be used, for instance, as the key for a symmetric cipher.

1.1.1 Behaviour

The simplest and the original implementation of the protocol uses the multiplicative group of integers modulo p, where p is prime, and g is a primitive root modulo p. These two values are chosen in this way to ensure that the resulting shared **secret** can take on any value from 1 to p–1. Here is an example of the protocol, with non-secret values in *italics*, and secret values in **bold**.

1.1.2 Exercise 9.1.1

1. Alice and Bob publicly agree to use a modulus $p = 23$ and base $g = 5$ (which is a primitive root modulo 23).
2. Alice chooses a secret integer $\mathbf{a} = 4$, then sends Bob A = $g^{\mathbf{a}} \mod p$
   - $A = 5^{\mathbf{4}} \mod 23 = 4$
3. Bob chooses a secret integer $\mathbf{b} = 3$, then sends Alice B = $g^{\mathbf{b}} \mod p$
   - $B = 5^{\mathbf{3}} \mod 23 = 10$
4. Alice computes $\mathbf{s} = B^{\mathbf{a}} \mod p$

- $\mathbf{s} = 10^{\mathbf{4}} \mod 23 = \mathbf{18}$

5. Bob computes $\mathbf{s} = A^{\mathbf{b}} \mod p$
   - $\mathbf{s} = 4^{\mathbf{3}} \mod 23 = \mathbf{18}$
6. Alice and Bob now share a secret (the number 18).

In applications p should have at least 2048 bits to avoid attacks as index calculus. More info here: https://www.keylength.com/en/compare/

Both Alice and Bob have arrived at the same values because under $\mod p$:$A^b \mod p = g^{ab} \mod p = g^{ba} \mod p = B^a \mod p$
More specifically : $(g^a \mod p)^b \mod p = (g^b \mod p)^a \mod p$

1.1.3 Man in the Middle (MITM)

In the original description, the Diffie–Hellman exchange by itself does not provide authentication of the communicating parties and is thus vulnerable to a man-in-the-middle attack. Oscar (an active attacker executing the man-in-the-middle attack) may establish two distinct key exchanges, one with Alice and the other with Bob, effectively masquerading as Alice to Bob, and vice versa, allowing her to decrypt, then re-encrypt, the messages passed between them. Note that Oscar must continue to be in the middle, actively decrypting and re-encrypting messages every time Alice and Bob communicate. If she is ever absent, her previous presence is then revealed to Alice and Bob. They will know that all of their private conversations had been intercepted and decoded by someone in the channel. In most cases it will not help them get Oscar's private key, even if she used the same key for both exchanges.

A method to authenticate the communicating parties to each other is generally needed to prevent this type of attack. Variants of Diffie–Hellman, such as STS protocol, may be used instead to avoid these types of attacks.

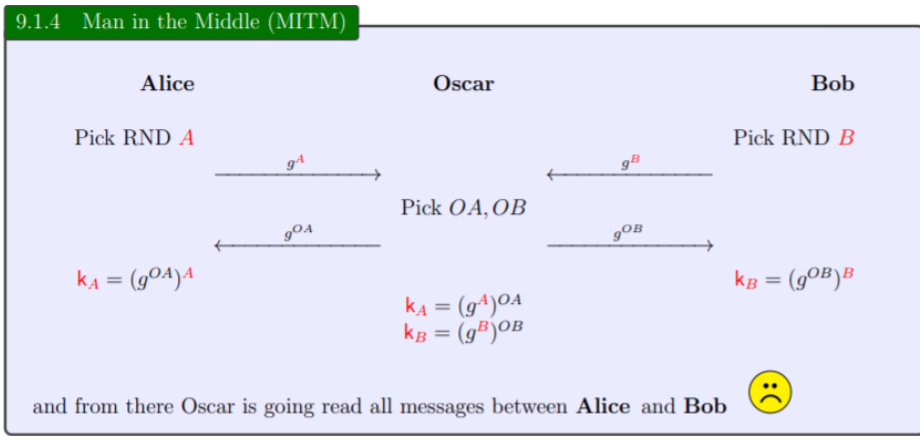

Figure 2:

2 Public Key Encryption

Public-key cryptography, or asymmetric cryptography, is a cryptographic system that uses pairs of keys: public keys (which may be known to others), and private keys (which may never be known by any except the owner). The generation of such key pairs depends on cryptographic algorithms which are based on mathematical problems termed one-way functions. Effective security requires keeping the private key private; the public key can be openly distributed without compromising security.

Public Key Cryptosystem (PKC)

Such cryptosystem consists of three algorithms

(Gen , Enc , Dec)

Gen generate a pair (sk,pk) of secret key sk and public key pk.

1) The algorithms Gen, Enc e Dec must be computationally feasible,
2) The secret key sk should be computationally infeasible to compute from the public key pk.

Figure 3: An unpredictable (typically large and random) number is used to begin generation of an acceptable pair of keys suitable for use by an asymmetric key algorithm.

2.1 ElGamal

In cryptography, the ElGamal encryption system is an asymmetric key encryption algorithm for public-key cryptography which is based on the Diffie–Hellman key exchange. It was described by Taher Elgamal in 1985. ElGamal encryption is used in the free GNU Privacy Guard software, recent versions of PGP, and other cryptosystems. ElGamal system is based on the difficulty of discrete logarithm problem where it is directly forward to raise numbers to large powers but it is much harder to do the inverse computation of the discrete logarithm.
The discrete logarithm: When dealing with real numbers, then the solution of the equation y=b^x can be found such that x= $log_b y$. Given the integers b and n such that b<n, the discrete logarithm of integer y to the base b is an integer x, such that $b^x = y \mod n$. Hence, unlike logarithms, the discrete logarithm problem is defined in a discrete domain where a solution must be exact.

2.1.1 Algorithm

The ElGamal encryption system parameters consists of a prime number (p) and an integer number (g) which is a root of (p-1) and whose power modulo (p) generate a large number of elements.

To encrypt plaintext, a private key (a) is used to generate the public key (y) as given in such that (a) is an integer between 1 and p-2.

$$y = g^a \mod p;$$

The public key for ElGamal encryption algorithm consists of the triple (p,g,y).

To encrypt a plaintext message (m), a random integer (k) is chosen such that it is between 1 and p-2. The message is converted to numbers before producing the ciphertext (c) which consists of the pair (y1,y2) calculated as given by and b;

$$y1 = g^k \mod p;$$
$$y2 = my^k \mod p;$$

The decryption process is applied in a reverse order such that the encrypted message (y1,y2) is used with the private key (a) and the prime number (p) to retrieve the original message (m) as given;

$$m = \frac{y2}{y1^a} \mod p;$$

The division by $y1^a$ in the last equation should be interpreted in the context of modular arithmetic, that is y2 is multiplied by the inverse of $y1^a$.

In a practical setting, ElGamal encryption does not really give a benefit when using it as an encryption scheme as it is, since it does only support message of size of elements of the group being used and parameters must be chosen carefully in order to obtain IND-CPA security (holds always in elliptic curves but you have to choose a suitable subgroup of $\mathbb{Z}_p^*$ to obtain it). ElGamal encryption, when the paramters are chosen in the right way achieves the weaker notion of indistinguishability under chosen plaintext attacks (IND-CPA).

Security of the ElGamal algorithm depends on the (presumed) difficulty of computing discrete logs in a large prime modulus.

- ElGamal has the disadvantage that the ciphertext is twice as long as the plaintext.
- It has the advantage the same plaintext gives a different ciphertext (with near certainty) each time it is encrypted

2.2 Rabin cryptosystem

Rabin Cryptosystem is an public-key cryptosystem invented by Michael Rabin. It uses asymmetric key encryption for communicating between two parties and encrypting the message. The security of Rabin cryptosystem is related to the difficulty of factorization. It has the advantage over the others that the problem on which it banks has proved to be hard as integer factorization. It has the disadvantage also, that each output of the Rabin function can be generated by any of four possible inputs. if each output is a ciphertext, extra complexity is required on decryption to identify which of the four possible inputs was the true plaintext.

2.2.1 Encryption Algorithm

Like all asymmetric cryptosystems, the Rabin system uses a key pair: a public key for encryption and a private key for decryption. The public key is published for anyone to use, while the private key remains known only to the recipient of the message.

Key generation
The keys for the Rabin cryptosystem are generated as follows:

1. Choose two large distinct prime numbers **p** and **q** such that $\mathbf{p} \equiv 3 \mod 4 and \mathbf{q} \equiv 3 \mod 4$
2. Compute **n**=pq.

Then **n** is the public key and the pair (p,q) is the private key.

Encryption
A message M can be encrypted by first converting it to a number $m < n$ using

a reversible mapping, then computing $c = m^2 \mod n$. The ciphertext is c.

Decryption
The message m can be recovered from the ciphertext c by taking its square root modulo n as follows.

1. Compute the square root of c modulo p and q using these formulas:

$$m_p = c^{\frac{1(p+1)}{4}} \mod p$$
$$m_q = c^{\frac{1(q+1)}{4}} \mod q$$

2. Use the extended Euclidean algorithm to find y_p and y_q such that $y_p \cdot p + y_q \cdot q = 1 y_p \cdot p + y_q \cdot q = 1$.

3. Use the Chinese remainder theorem to find the four square roots of c modulo n:

$$r_1 = (y_p \cdot p \cdot m_q + y_q \cdot q \cdot m_p) \bmod n$$
$$r_2 = n - r_1$$
$$r_3 = (y_p \cdot p \cdot m_q - y_q \cdot q \cdot m_p) \bmod n$$
$$r_4 = n - r_3$$

One of these four values is the original plaintext m, although which of the four is the correct one cannot be determined without additional information.

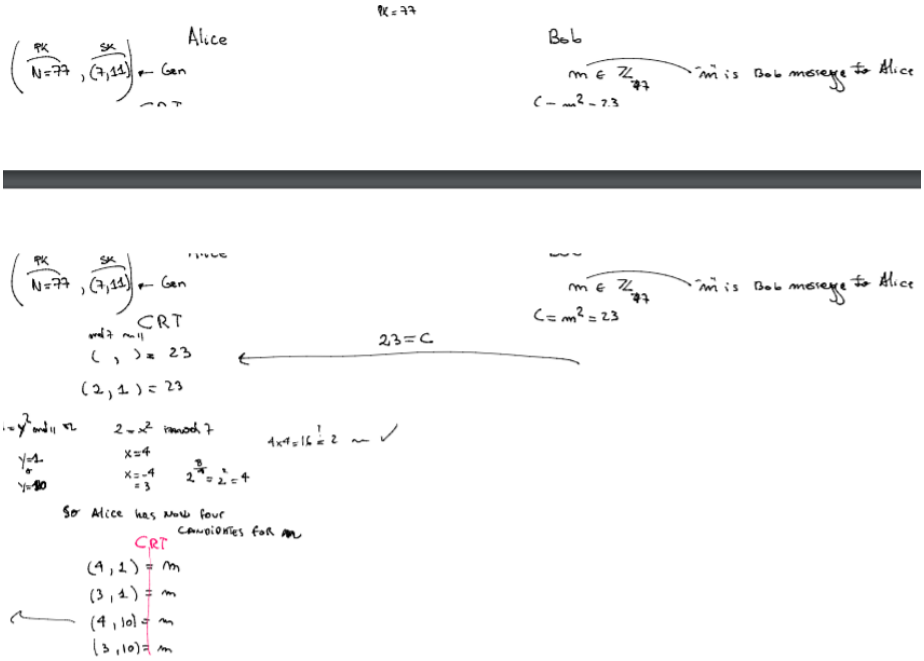

Figure 4: Exercise 9.2.7