# Lecture 6: Cryptographic Algorithms

Cryptography is the study of secure communications techniques that allow only the sender and intended recipient of a message to view its contents. The term is derived from the Greek Word kryptos, which means hidden. It is closely associated to encryption, which is the act of scrambling ordinary text into what's known as ciphertext and then back again upon arrival.

Let K be a key and ***enc*** and ***dec*** a encryption function such that+

$$dec = enc^{-1}$$

then

$$\textbf{plaintext} = dec(K, enc(K, \textbf{plaintext}))$$

Key space (set of all keys) has $2^{64}!$ elements
This implies a key length up to $\log_2(2^{64}!) \cong 10^{21}$ bit

## Symmetric Encryption

Secure communication has been an essential human need for a very long time. For example, the ancient Greeks used a device in 400 BC to hide and reveal military messages to protect them from enemy armies. The method of changing a message to hide or reveal its meaning is called a cipher. Ciphers are used to encrypt (hide) a message or decrypt (reveal) a message using an algorithm (a sequence of defined steps). It is often very hard to decrypt a message without knowing what cipher was used.

Symmetric-key algorithms are algorithms for cryptography that use the same cryptographic keys for both the encryption of plaintext and the decryption of ciphertext.

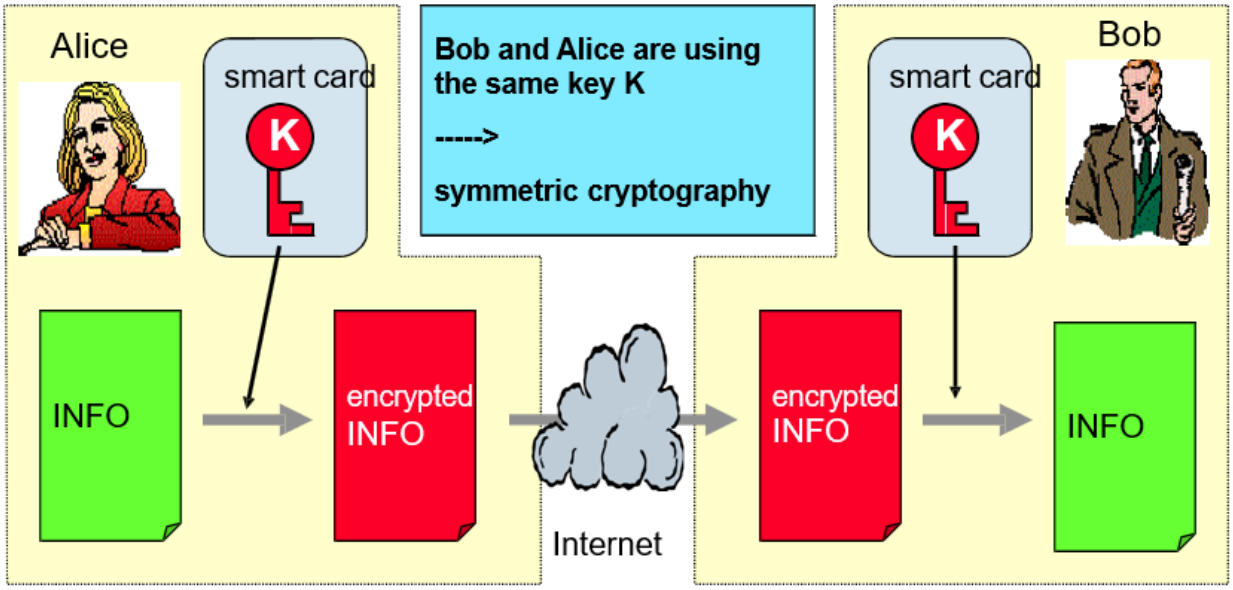


Figure 1. The concept of symmetric encryption

The two types of traditional symmetric ciphers are **transposition cipher** and **substitution cipher**.

A transposition cipher encrypts a word by changing the order of its letters. One order could be to remove the letters from even positions and add them to the end of the word. For example: PINEAPPLE is encrypted as PNAPEIEPL.

In a substitution cipher, any character of plain text from the given fixed set of characters is substituted by some other character from the same set depending on a key. For example, with a shift of 1, A would be replaced by B, B would become C, and so on. Caesar cipher and linear cipher are two popular substitution ciphers.

**Caesar cipher**

A Caesar cipher encrypts a word by replacing each letter with another letter in the alphabet, usually in a pattern. For example, letter A is replaced by B, letter B is replaced by C, etc. (letter Z is replaced by A). In other words, this Caesar cipher rotates each letter in the alphabet to the right by 1 position. For example: PINEAPPLE is encrypted as QJOFBQQMF.

The table below shows mapping of letters while using a Caesar cipher rotated by 1 position.

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A |

The rotation can, of course, be larger than 1. The most popular rotation value is 3. If the alphabet is rotated right by three positions, then letter A is replaced by D, letter B is replaced by E, etc. For example: PINEAPPLE is encrypted as SLQHDSSOH (with a rotation of 3).

The table below shows the mapping of letters when using a Caesar cipher with a rotation of 3.

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |

Remember! Sometimes multiple ciphers are used to encrypt a message.

**Linear Cipher**

In the encryption process, this technique uses the linear function y=ax+b, which is known as the equation of the line in geometry. Accordingly, x denotes the message to be encrypted (plain text), y denotes the encrypted message (cipher text), and the pair a and b form the key.

Example message: “baba dede”
Key: (3,2), i.e., given as a = 3, b = 2

Creating the encrypted message: As b is the 2nd letter in the alphabet, it is replaced by 3×2 +2 = 8, that is, the 8th letter of the alphabet. This letter corresponds to 'h'. It is calculated similarly for other letters. For example, the letter 'e' is replaced by 3×5 +2 = 17, that is, the letter 'q'.

Encrypted message: “hehe nqnq”

**Public Key Cryptography - The Concept of Confidentiality**

Asymmetric cryptography, also known as public-key cryptography, is a process that uses a pair of related keys -- one public key and one private key -- to encrypt and decrypt a message and protect it from unauthorized access or use.

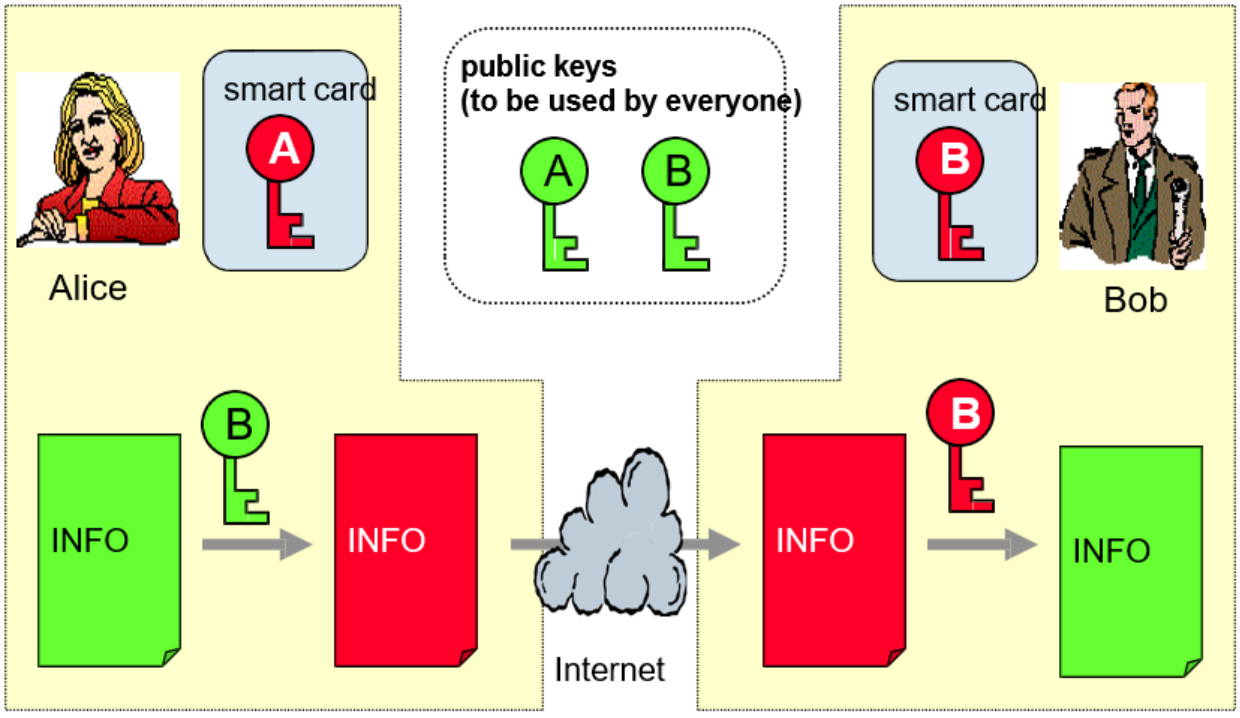


Figure 2. The concept of asymmetric encryption

**Public Key Algorithms:**

RSA (Rivest–Shamir–Adleman)

- based on prime numbers
- typical key length: 1024 bit

Elliptic functions

- based on elliptic functions and finite fields
- typical key length: 160 bit

Pro and cons of public key cryptography:

+ allows sophisticated key management

+ very high safety

- slow algorithms

**Asymmetric Encryption using RSA algorithm**

RSA is an asymmetric encryption method developed by R. **R**ivest, A. **S**hamir, and L. **A**dleman in 1977. The RSA method is based on a key pair consisting of a non-secret **public key for encryption** and a **private key for decryption**. The public key allows anyone to encrypt data for the owner of the private key. The private key enables its owner to decrypt data encrypted with the public key.

The public and private RSA keys are calculated as follows:

1. Compute n as the product of two very large, freely chosen prime numbers p and q:

   **n = p * q**

2. Compute **φ(n) = (p - 1) * (q - 1)**

3. Choose an encryption key **e** that satisfies the following two conditions:

   i. $1 < e < \varphi(n)$ and
   ii. e and φ (n) are coprime[1].

4. Compute a value for decryption key **d** such that (d * e) % φ(n) = 1. The % sign refers to the modulo operation[2].

5. Your public key is **(e, n)**

6. Your private key is **(d, n)**

[1] Co-prime means that e should not multiply by factors of φ(n) and also not divide by φ(n), e.g., e = 18 and φ(n) = 9 are not coprime as 18 is dividable by 9. In contrast, 18 and 21 are coprime as their greatest common divisor is 1.
[2] The modulo operation returns the remainder of a division, after one number is divided by another, e.g. 8 % 3 = 2

The data can be encrypted and decrypted as follows.

1. The encryption of cleartext *m* is

   ***c = m^e % n***

   where c represents the encrypted form of m.

2. The decryption of *c* is

   ***m = c^d % n***

   where m represents the clear text form of c.

**Example:** Let be p = 3 and q = 11.

1. Compute **n = p * q = 3 * 11 = 33**

2. Compute **φ(n) = (p - 1) * (q - 1) = 2 * 10 = 20**

3. Choose, e.g., the encryption key **e = 7** that satisfies the following two conditions:
   - 1 < e = 7 < φ(n) = 20 and
   - e = 7 and φ (n) = 20 are coprime.

4. Compute a value for the decryption key **d** that satisfies the following condition:
   - (d * e) % φ(n) = 1.
   - One solution is **d = 3** because (3 * 7) % 20 = 1

5. Public key is **(e, n) => (7, 33)**

6. Private key is **(d, n) => (3, 33)**

Let, e.g., m = 2 be the clear text to be encrypted with the public key **(e, n) => (7, 33)**.

7. *For encryption,* compute $c = m^e \% n$ => **$c = 2^7 \% 33 = 29$**

8. *For decryption,* compute $m = c^d \% n$ => **$m = 29^3 \% 33 = 2$**

**Hash Functions**

One of the most essential objectives of cryptography is to ensure the integrity of messages. Integrity protection is the ability to determine that a message in a communication has not been modified or tampered with between the creator and the viewer by an unauthorized user (i.e., attacker). The integrity of data can be protected using hash functions. Hash functions generate a fixed-length hash value from a data record of arbitrary length. A data record can be a word, a sentence, a long piece of text, or an entire file. The hash value generated is often referred to as a digital fingerprint or cryptographic checksum, which is characteristic for the input data. The hash value is generated by a formula in such a way that it is extremely unlikely that some other text will generate the same hash value.

Four heuristic hash functions are **division hashing, mid-square hashing, digit folding hashing and multiplicative hashing**, which operate on numeric or alphanumeric keys/messages.

**Division Hashing**

This is the most simple and easiest method to generate a hash value. The hash function divides the value k by M and then uses the remainder obtained.

***h(K) = k mod M***

*Here,*

***k** is the key value, and*

***M** is the size of the hash table.*

It is best suited that **M** is a prime number as that can make sure the keys are more uniformly distributed. The hash function is dependent upon the remainder of a division.

**Example #1:**

*k = 12345*

*M = 95*

*h(12345) = 12345 mod 95 = 90*

**Example #2:**

*k = 1276*

*M = 11*

*h(1276) = 1276 mod 11 = 0*

**Mid-Square Hashing**

The mid-square hashing involves two steps to compute the hash value:

1. Square the value of the key k, i.e., $k^2$
2. Extract the middle **r** digits as the hash value.

**Formula:**

***h(k) = h(k * k)***

*Here,*

***k** is the key value.*

The value of **r** can be decided based on the size of the table.

**Example:**

Suppose the hash table has 100 memory locations. So, r = 2 because two digits are required to map the key to the memory location.

*k = 60*

*k * k = 60 * 60 = 3600*

*h(60) = 60*

*The hash value obtained is 60*

**Pros:**

1. The performance of this method is good as most or all digits of the key value contribute to the result. This is because all digits in the key contribute to generating the middle digits of the squared result.
2. The result is not dominated by the distribution of the top digit or bottom digit of the original key value.

**Cons:**

1. The size of the key is one of the limitations of this method, as the key is of big size then its square will double the number of digits.
2. Another disadvantage is that there will be collisions but we can try to reduce collisions.

**Digit Folding Hashing**

This technique involves two steps:

1. Divide the key-value **k** into a number of parts, i.e., $\mathbf{k_1, k_2, k_3,\ldots,k_n}$, where each part has the same number of digits except for the last part that can have lesser digits than the other parts.

2. Add the individual parts. The hash value is obtained by ignoring the last carry if any.

**Formula:**

$k = k_1, k_2, k_3, k_4, \ldots\ldots, k_n$

$s = k_1 + k_2 + k_3 + k_4 + \ldots + k_n$

$h(k) = s$

*Here, **s** is obtained by adding the parts of the key **k***

**Example:**

*k = 12345*

*k1 = 12, k2 = 34, k3 = 5*

*s = k1 + k2 + k3*

*= 12 + 34 + 5*

*= 51*

*h(k) = 51*

**Note:**

The number of digits in each part varies depending upon the size of the hash table. Suppose for example the size of the hash table is 100, then each part must have two digits except for the last part which can have a lesser number of digits.

**Multiplicative Hashing**

This method involves the following steps:

1. Choose a constant value c such that 0 < c < 1.
2. Multiply the key value with c.
3. Extract the fractional part of k * c.
4. Multiply the result of the above step by the size of the hash table, i.e. M.
5. The resulting hash value is obtained by taking the floor of the result obtained in step

**Formula:**

***h(k) = floor (M * (k * c mod 1))***

*Here,*

***M** is the size of the hash table.*

***k** is the key value.*

***c** is a constant value.*

**Example:**

*k = 12345*

*c = 0.357840*

*M = 100*

*h(12345) = floor[ 100 * (12345*0.357840 mod 1)]*
*= floor[ 100 * (4417.5348 mod 1) ]*
*= floor[ 100 * (0.5348) ]*
*= floor[ 53.48 ]*
*= 53*

**Pros:**

The advantage of the multiplication method is that it can work with any value between 0 and 1, although there are some values that tend to give better results than the rest.

**Cons:**

The multiplication method is generally suitable when the table size is the power of two, then the whole process of computing the index by the key using multiplication hashing is very fast.

Popular cryptographic hash functions: MD5, SHA family

**Public Key Cryptography - The Concept of Digital Signatures**

The digital or electronic signature is a checksum that is generated from a message in combination with a key. Digital signatures are used for digital messages to make their authenticity credible and verifiable. The digital signature is attached to the message, and the authenticity of the signature can be checked electronically.

Digital signatures are necessary for data transmission because the sender of messages can be forged. For example, it is very easy to fake the sender of an email. That is, it is possible for someone to impersonate someone else. Also, in real life, you can write any sender address on a letter. To underline the credibility of the letter, we put our signature at the end of the letter. The same is done with the digital signature.

Earlier, we learned that a hashing function is a technique that verifies the integrity of a message (i.e., hashing code calculated by a hashing function reveals whether the originally sent message has been manipulated by an attacker). Furthermore, we also learned the concept of asymmetric encryption, where we use two different keys, namely a public key to encrypt a message and a private key to decrypt the encrypted message. For signing a message, we will need to combine the hashing function and asymmetric encryption concept.

**In summary, creating and verifying digital signatures work as follows:**

To sign a message, the sender, Alice, first creates the hash code of the message (e.g., using the division hashing). Then, Alice creates the signature by encrypting the hash code with her private key and sends this signature to Bob. Finally, Bob verifies the signature by using the public key of Alice. Note that, differently from the classical asymmetric encryption, we reverse the use of the public and private keys. We're going to encrypt with the private key and see if we can decrypt with the public key.

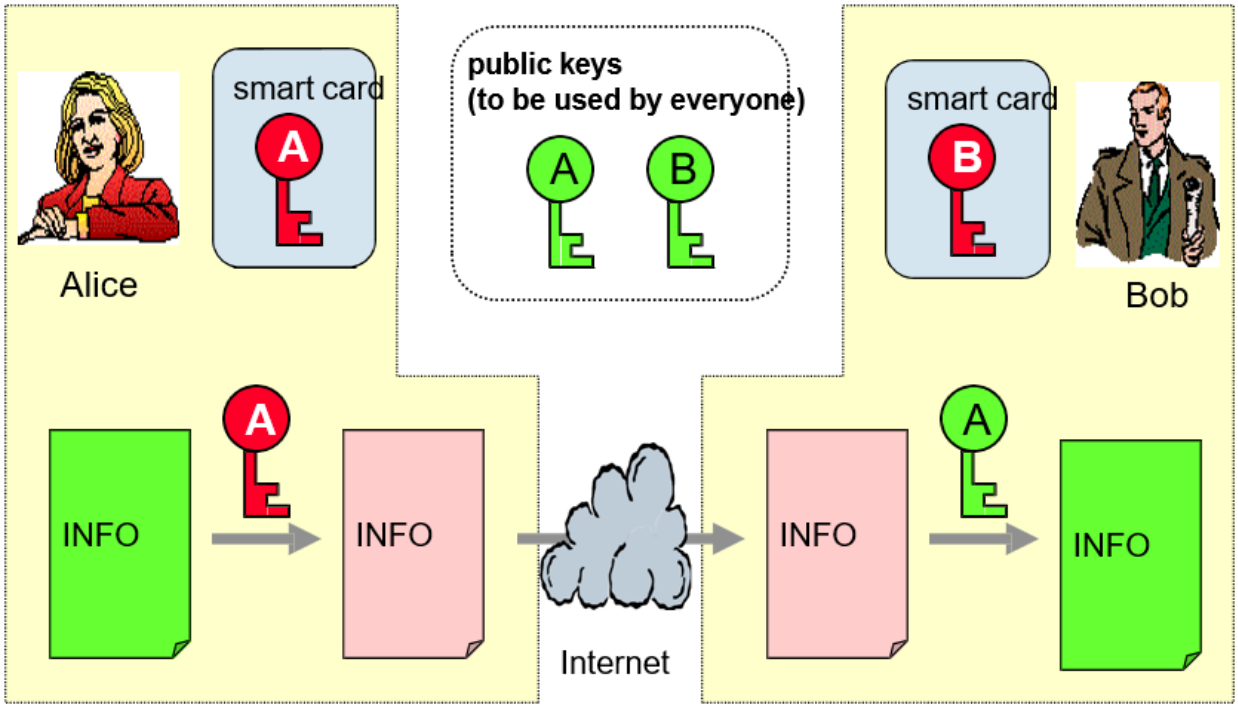


Figure 3. The concept of digital signature

Let be (e, n) and (d,n) the public and private keys, respectively. The hash code c of the message m with h(m) = c can be signed and verified as follows.

1. The signature of the hash code c is

   $\underline{\boldsymbol{s = c^d \% n}}$

   where s represents the signed hash code c.

2. The verification of the signature *s* is

   $\underline{\boldsymbol{c = s^e \% n}}$

   where c represents the verified hash code c.

**Example:** Let, e.g., the hash code of a message be c=2; (e, n) => (7, 33) the public key; and (d, n) => (3, 33) the private key.

1. *For signing*, compute $s = c^d \% n$ => $\underline{\mathbf{c = 2^3 \% 33 = 8}}$

2. *For verification*, compute $c = s^e \% n$ => $\underline{\mathbf{c = 8^7 \% 33 = 2}}$

**The Problem of Key Management**

Let Alice and Bob be two participants in communication. If Alice and Bob use the same secret session key for encryption and decryption, it must be ensured that this key is only known to both. Thus, it must be securely exchanged or agreed upon prior to communication. Manual procedures for key exchange are complex and confusing. If a medium can be eavesdropped, the key used must under no circumstances be transmitted via this medium since attackers could intercept the key and use it for encryption and decryption.

Two possible solutions:

- Solution 1: By sending an encrypted symmetric key
- Solution 2: Using the Diffie-Hellman key exchange algorithm

**The Concept of Hybrid Encryption**

Symmetric encryption methods are usually faster than asymmetric ones and can achieve a high level of security with comparatively shorter key lengths. The disadvantage of symmetric encryption is that the symmetric key must be kept secret, i.e., distributing the key to several participants over a potentially insecure network is problematic.

The advantage of asymmetric encryption is that the key can be exchanged without any problems since the public key of a user is not secret and can be distributed to all participants without hesitation. The disadvantage of asymmetric encryption is that it is slower than symmetric in terms of performance.

Hybrid encryption was developed to combine the advantages of both encryption techniques: the speed of symmetric encryption and the simplicity of key exchange of asymmetric encryption. More specifically, a symmetric key is created for quick encryption of the data. Then, to securely exchange this symmetric key with the partner, the symmetric key is asymmetrically encrypted with

the receiver's public key and sent to the receiver. Upon arrival, the receiver can decrypt the symmetric key with his/her private key and then start to use the symmetric key.

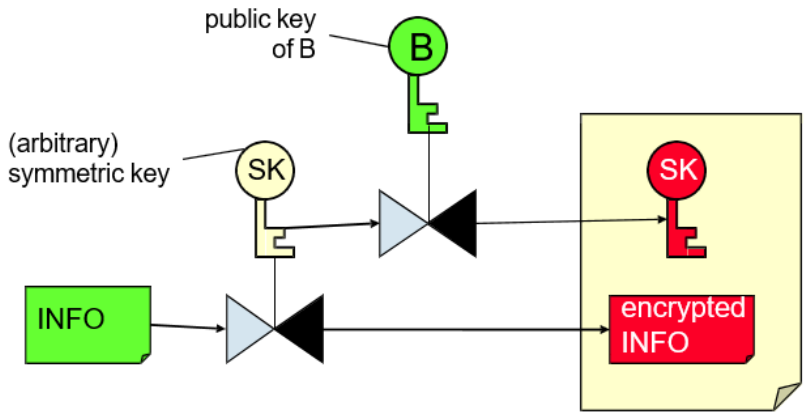


Figure 4. Alice wants to encrypt a document INFO for Bob

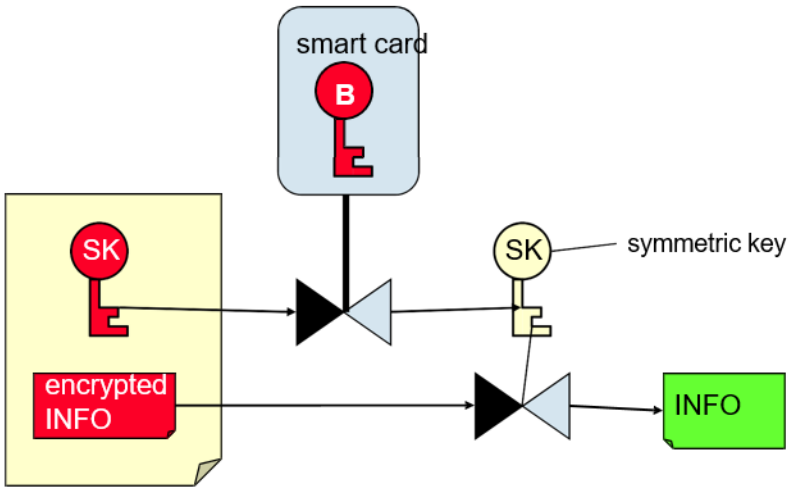


Figure 5. Bob wants to read the document sent by Alice

**Diffie-Hellman Key Exchange Algorithm**

The Diffie-Hellman key exchange is a procedure with which a shared session key can be securely agreed between two communication partners over a potentially insecure transmission medium. Strictly speaking, no key exchange occurs since the shared secret session key is never transmitted. Instead, the keys are calculated over other non-secret exchanged information. For attackers who eavesdrop on the medium, it is mathematically impossible to calculate the common session key with justifiable effort. Once Alice and Bob have agreed on a shared session key, they use this to encrypt and decrypt the data.

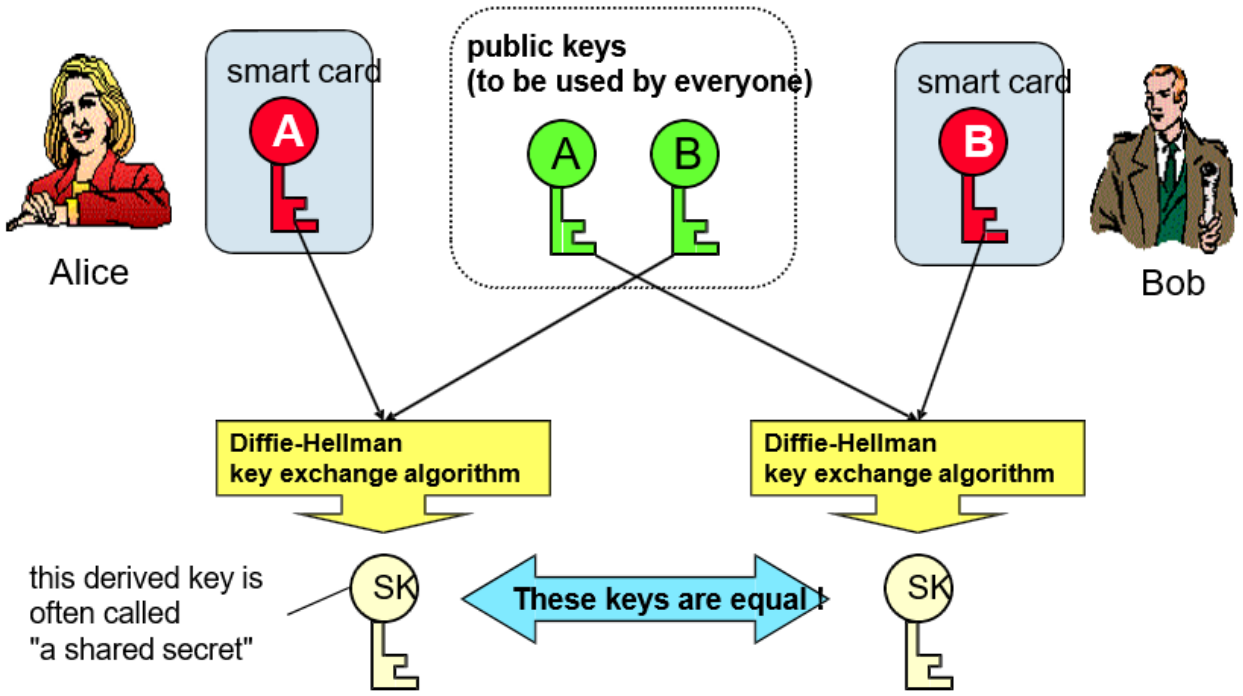


Figure 6. Overview of Diffie-Hellman key exchange algorithm

In the following, the steps of the Diffie-Hellman key exchange algorithm are listed:

1. Alice and Bob agree on a large public **prime number p** and **a smaller number g**

2. Simultaneously, Alice and Bob choose **the random numbers a and b**, respectively. These are secret keys that will never be exchanged between the communication partners, i.e., these random numbers do not leave the local machines of Alice and Bob.

3. Using the random numbers a and b as well as the two previously agreed numbers p and g, Alice and Bob calculate the new numbers A and B, respectively, as follows:

   Alice computes **$A = g^a \% p$**

   Bob computes **$B = g^b \% p$**

   Note: The % sign refers to the modulo operation[3].

4. Alice and Bob exchange their public numbers A and B mutually.

5. Alice and Bob calculate the common key k using their random numbers a and b as well as the mutually exchanged public numbers A and B as follows:

   Alice computes **$k = B^a \% p$**

   Bob computes **$k = A^b \% p$**

[3] The modulo operation returns the remainder of a division, after one number is divided by another, e.g. 8 % 3 = 2

6. Alice and Bob use the key **k** to encrypt and decrypt the data

Example:

1. Alice and Bob agree on **p = 5** and **g = 3**.

2. Alice selects **a = 7** and Bob selects **b = 9** as secret random numbers.

3. Using the random numbers a and b as well as the two previously agreed numbers p and g,
   a. Alice computes **$A = g^a \% p = 3^7 \% 5 = 2$**
   b. Bob computes **$B = g^b \% p = 3^9 \% 5 = 3$**

4. Alice and Bob exchange the public numbers **A = 2** and **B = 3** mutually.

5. Alice and Bob calculate the common key k as follows:

   Alice computes **$k = B^a \% p = 3^7 \% 5 = 2$**

   Bob computes **$k = A^b \% p = 2^9 \% 5 = 2$**

6. Alice and Bob use the key **k = 2** to encrypt and decrypt the data

**Homework:**

You are given three single-word passwords encrypted using the ciphers presented in this lecture. To be more secure, some passwords are encrypted using more than one cipher. Your task is to determine which ciphers were used and decrypt the passwords.

Encrypted password1 = CMUIAINOMNCTO
Encrypted password2 = VHFUHW
Encrypted password3 = WGIIVX

What was password1 before encryption?
Answer:

What was password2 before encryption?
Answer:

What was password3 before encryption?
Answer:

RSA and Digital Signature: Let be p = 17 and q = 11. Also, assume that RSA chooses e = 7 and d = 23 as encryption and decryption keys, respectively. Given these values of p, q, e, and d,

a) calculate your public and private keys using RSA.
b) Then, assume you want to exchange your clear text message m = 5 securely. Encrypt m with your public key and decrypt the encrypted text again with your private key.
c) Sign the message m=5 with the private key and verify it with the corresponding public key.

**Diffie Hellman Algorithm:** Let be p = 11 and g = 5. Also, assume that Alice and Bob choose a = 7 and b = 3, respectively. Given these values of p, g, a, and b, calculate the shared session secret for Alice and Bob.

**Hash Functions:** In order to calculate the numerical representation of m, we can use the table below, which shows the mapping between letters and their numerical codes. We can just sum up the codes of each character for a given input string to obtain its numerical representation. For example: HELLO = 8 + 5 + 12 + 12 + 15 = 52

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |

As shown in the table below, you are given four single-word messages and the corresponding (sent) hash values (p and c values are set to 31 and 0.56, respectively). Some messages were manipulated by an unauthorized user. Your task is to verify the sent hash values of each message to determine which messages are manipulated.

| **Hash function** | **Message** | **Sent Hash Value** | **Calculated Hash Value** | **Message Verified / Manipulated** |
|---|---|---|---|---|
| Division Hashing | MORNING | 28 | | |
| Division Hashing | REGARDS | 30 | | |
| Multiplicative Hashing | SECURITY | 13 | | |
| Multiplicative Hashing | HASHING | 29 | | |