Date: Monday, October 21, 2002
Prof.: Dr Jean-Yves Chouinard

Design of Secure Computer Systems CSI4138/CEG4394

Notes on Public Key Cryptography

Public-key encipherment concept

Each user in a secure communication network has a private key, which he keeps secret, and a public key which is stored in a directory. The network users use the public keys to communicate with each other.

Secrecy:

Network user A encrypts message M to user B, by applying B's public key transformation $E_{K_B}(\bullet)$ to the plaintext and sends it to recipient B.

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

however, only user B can decrypt the ciphertext C, since only him knows the private key transformation $D_{k_B}(\bullet)$:

$$D_{k_B}(C) = D_{k_B}[E_{K_B}(M)] \Longrightarrow M$$

Authentication:

If user A wants to authentify a message M, he uses his own private key transformation, i.e. $D_{k_A}(\bullet)$ and then sends the authentified message S.

$$S = D_{k_A}(M)$$

Everyone in the network can verify the authenticity of the message by performing the inverse transformation with public key $E_{K_A}(\bullet)$:

$$E_{K_A}(S) = E_{K_A}[D_{k_A}(M)] \Longrightarrow M$$

Note that the message is not protected, since every user in the network can recover the message M.

Secrecy and authentication:

The message M can also be first authentified by A with his private key, and then encrypted using the recipient own public key transformation:

$$C = E_{K_B}(S) = E_{K_B}[D_{k_A}(M)]$$

At this point, only recipient B can retrieve the authentified message since only him can perform the decryption transformation $D_{k_B}(\bullet)$. He then uses A's public key to verify the message authenticity.

$$E_{K_A}[D_{k_B}(C)] = E_{K_A}[D_{k_B}[E_{K_B}[D_{k_A}(M)]]] = E_{K_A}[D_{k_A}(M)] \Longrightarrow M$$

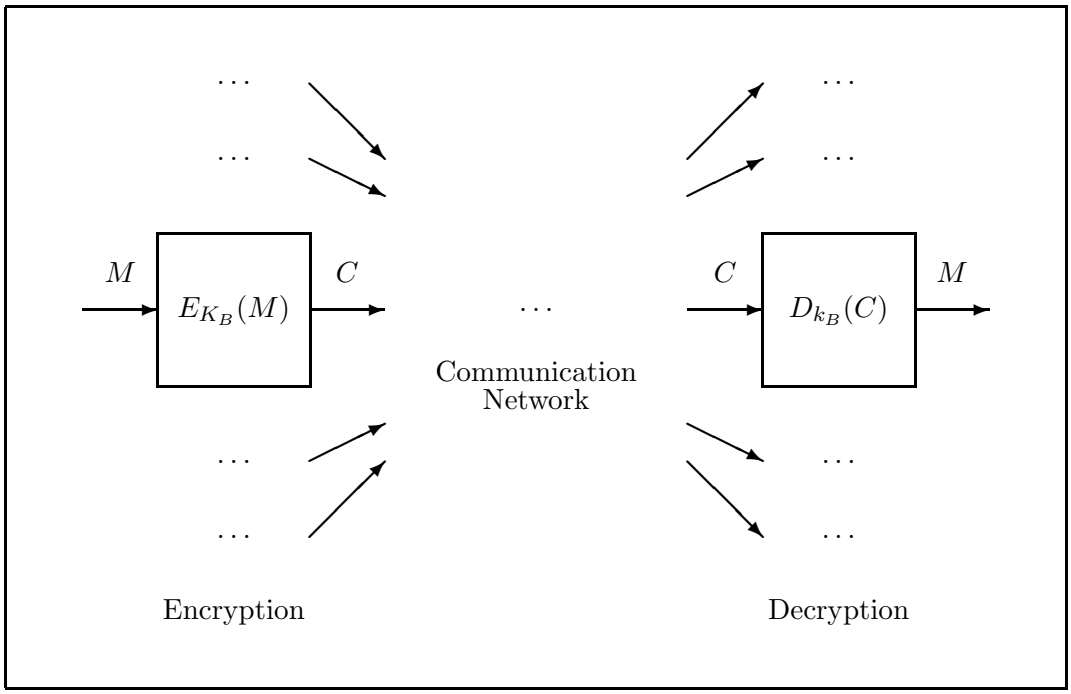

Figure 1: Communication link with public key encryption

Exponentiation ciphers

The encryption as well as the decryption transformation consists in a modular exponentiation:

$$\begin{aligned} C &= E_K(M) = M^e \bmod n \qquad \text{and} \\ M &= D_K(C) = C^d \bmod n \end{aligned}$$

For instance, the fast exponentiation program can be used for encryption (i.e. $fastexp(M, e, n)$) and decryption ($fastexp(C, d, n)$). The message stream must be broken into message blocks $M_1, M_2, \ldots$. If M is relatively prime to n then:

$$\gcd(M, n) = 1 \qquad \text{and then}$$
$$M^{\phi(n)} \bmod n = 1$$

Now, if the exponents e and d are such that

$$(e\ d) \bmod \phi(n) = 1$$

then $M = C^d \bmod n$ is the inverse transformation of $C = M^e \bmod n$ or:

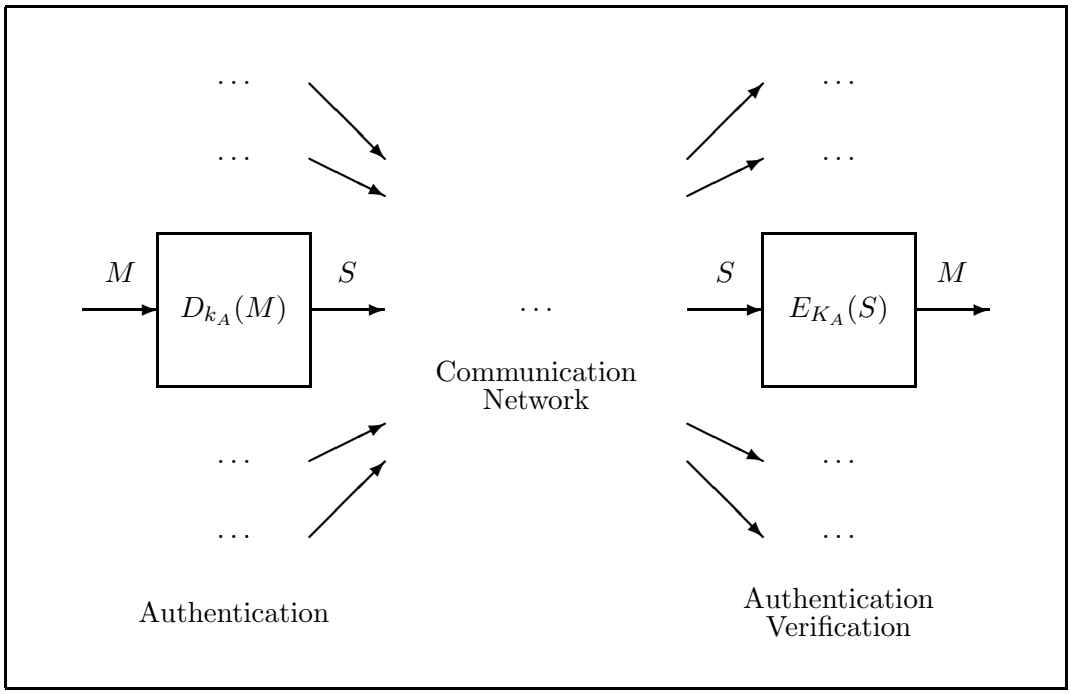

Figure 2: Communication link with public key authentication

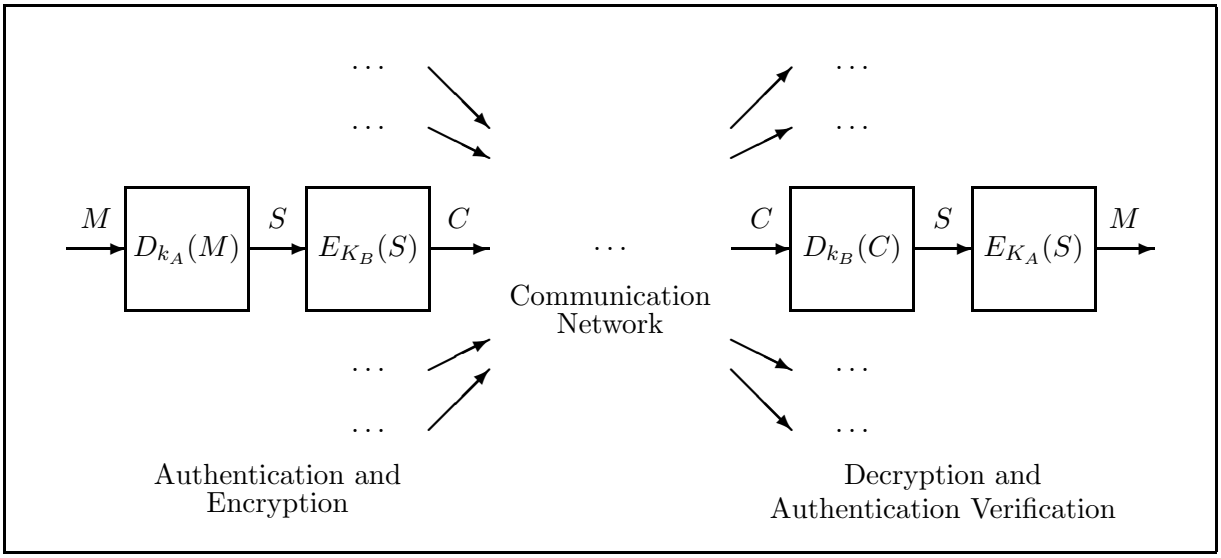

Figure 3: Communication link with public key authentication and encryption

$$M = D_K[E_K(M)]$$

$$M = [M^e \bmod n]^d \bmod n \qquad \text{and}$$

$$\begin{aligned} M &= E_K[D_K(M)] \\ M &= [M^d \bmod n]^e \bmod n \end{aligned}$$

Consider the expression $D_K[E_K(M)]$. By the principle of modular arithmetics:

$$\begin{aligned} D_K[E_K(M)] &= [M^e \bmod n]^d \bmod n \\ &= \{\underbrace{[M^e \bmod n]\ [M^e \bmod n]\ \ldots\ [M^e \bmod n]}_{d \text{ times}}\} \bmod n \\ &= [M^e\ M^e\ \ldots\ M^e] \bmod n \\ D_K[E_K(M)] &= M^{ed} \bmod n \end{aligned}$$

but since $(ed) \bmod \phi(n) = 1$, we can write that $(ed) = k\phi(n) + 1$, where k is an integer and 1 is the residue modulo $\phi(n)$. Then:

$$\begin{aligned} D_K[E_K(M)] &= M^{k\phi(n)+1} \bmod n \\ &= [MM^{k\phi(n)}] \bmod n \\ &= [(M \bmod n)\ (M^{k\phi(n)} \bmod n)] \bmod n \\ &= [M\ (M^{k\phi(n)} \bmod n)] \bmod n \end{aligned}$$

since $M \in [0, n-1]$

$$\begin{aligned} D_K[E_K(M)] &= [M\ (M^{k\phi(n)} \bmod n)] \bmod n \\ &= [M\ (M^{\phi(n)})^k \bmod n] \bmod n \\ &= [M\ \underbrace{M^{\phi(n)}\ M^{\phi(n)}\ \ldots\ M^{\phi(n)}}_{k \text{ times}}] \bmod n \\ &= [M\ M^{\phi(n)} \bmod n\ M^{\phi(n)} \bmod n\ \ldots\ M^{\phi(n)} \bmod n] \bmod n \\ &= [M\ 1\ 1\ \ldots\ 1] \bmod n \\ D_K[E_K(M)] &= M \end{aligned}$$

Rivest, Shamir and Adleman (RSA) cipher

One the most important public key encipherment algorithm is the RSA cipher [RSA78] which has been proposed by Rivest, Shamir and Adleman in 1978. It also makes use of modular exponentiation.

Secrecy using RSA

Let p and q be two distinct prime numbers and let n be the product of these numbers: $n = p \times q$. Choose an integer e relatively prime to $\phi(n) = (p-1)(q-1)$. The encryption transformation is then computed as:

$$C = E_K(M) = M^e \bmod n$$

while the decryption transformation is done as:

$$M = D_K(C) = C^d \bmod n$$

In the RSA cryptosystem, the private key consists in the triplet (p, q, e) and the public key is simply the pair (e, n). This public pair (e, n) is stored in a public directory which can be read by all the users in a communication network.

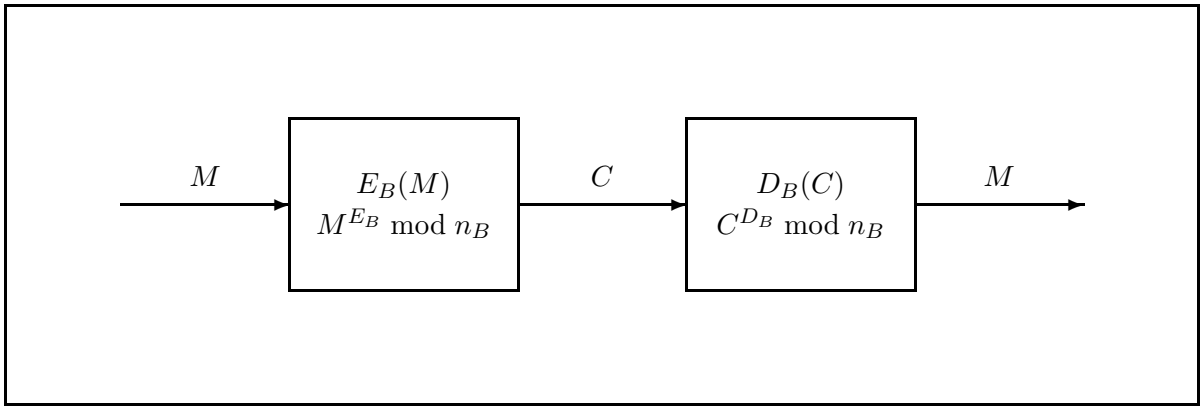

Figure 4: Block diagram of RSA secure communication link

When user A wants to send a secret message to user B, he uses B's public key (e_B, n_B) to encrypt his message M prior transmission over the communication network.

$$(e_B, n_B) \Longrightarrow C = M^{e_B} \bmod n_B$$

Even if user A has the public key (e_B, n_B), the message M as well as the corresponding ciphertext $C = E_K(M)$, it is extremely difficult to compute the private key (p_B, q_B, e_B). This involves the factorization of the number n_B in two large prime numbers, namely p_B and q_B.

The legitimate recipient of the message, user B, is the only one in the network to possess the private key information. Upon receiving the ciphertext C, he can decrypt the original plaintext using the multiplicative inverse d_B of the exponent e_B. From the two large prime numbers p_B and q_B it is easy to compute this inverse d_B; and thus to perform the decryption transformation D_K of the ciphertext:

$$(e_B \times d_B) \bmod \phi(n_B) = 1$$

where $\phi(n_B) = (p_B - 1)(q_B - 1)$. By Euler's theorem:

$$M^{e_B d_B} \bmod n_B = M \qquad \text{with } 1 \leq M \leq n_B$$

Example*(RSA cipher):*

Suppose that the recipient, user B, chooses $p_B = 19$ and $q_B = 23$ as his private key prime numbers and computes n_B:

$$n_B = 19 \times 23 = 437$$

Since p_B and q_B are primes, the Euler function $\phi(n_B)$ is given by:

$$\begin{aligned} \phi(n_B) &= (p_B - 1)(q_B - 1) \\ \phi(n_B) &= 18 \times 22 \\ \phi(n_B) &= 396 \end{aligned}$$

He then arbitrarily chooses an exponent e_B relatively prime with $\phi(n_B) = 396$. For instance, he may take $e_B = 13$ (note: $\gcd(13, 396) = 1$). The multiplicative inverse of e_B modulo $\phi(n_B)$ is determined as:

$$d_B = e_B^{\phi[\phi(n_B)]-1} \bmod \phi(n_B)$$

where $\phi[\phi(n_B)] = \phi(396) = 120$. Since

$$\phi(n_B) = \prod_i p_i^{e_i} = 2^2 \times 3^2 \times 11^1 = 396$$

then

$$\begin{aligned} \phi[\phi(n_B)] &= \prod_i p_i^{(e_i-1)}(p_i - 1) \\ \phi[\phi(n_B)] &= 2^1(2-1) \times 3^1(3-1) \times 11^0(11-1) = 2 \times 6 \times 10 \\ \phi[\phi(n_B)] &= 120 \end{aligned}$$

And

$$\begin{aligned} d_B &= e_B^{\phi[\phi(n_B)]-1} \bmod \phi(n_B) \\ d_B &= 13^{119} \bmod 396 \\ d_B &= 61 \end{aligned}$$

User B then stores the pair $(e_B, n_B) = (13, 437)$ in the public network key directory. If another network user, say user A, wants to encrypt a message M for user B, e.g. $M = 123$, he then does so by using the public key $(13, 437)$ of the recipient, user A.

$$\begin{aligned} C &= M^{e_B} \bmod n_B \\ C &= 123^{13} \bmod 437 \\ C &= 386 \end{aligned}$$

The recipient, using his secret key information (p_B, q_B, e_B) and thus d_B, is the only network user knowing that $d_B = 61$ and is thus the only one able to perform the decryption transformation properly on the received ciphertext C:

$$\begin{aligned} M &= C^{d_B} \bmod n_B \\ M &= 386^{61} \bmod 437 \\ M &= 123 \end{aligned}$$

Schroeppel [SS79] has estimated that the number of steps required to factorize an RSA cipher with the product n to be:

$$T(n) = \exp[\sqrt{\ln(n) \ln \ln(n)}] \qquad \text{steps.}$$

Rivest, Shamir and Adleman suggested using 100 decimal digits numbers for both p and q, resulting in a 200 decimal digit number for the product n.

Authentication using RSA

The RSA algorithm can also be used for authentication purposes. User A may prove to user B that the message M that he is sending is a genuine message by applying his own private key information (p_A, q_A, e_A) to the message M.

$$S = M^{d_A} \bmod n_A$$

Every user in the network can decrypt A's message by using his public key information (e_A, n_A), but only user A can produce this authenticated message S.

$$\begin{aligned} M &= S^{e_A} \bmod n_A \\ M &= (M^{d_A} \bmod n_A)^{e_A} \bmod n_A \\ M &= M^{d_A \; e_A} \bmod n_A \end{aligned}$$

Secrecy and authentication using RSA

Finally, both secrecy and authentication can be achieved with RSA by using the public and private keys of both users. The sender, user A for instance, authenticates his message by using his private key (p_A, q_A, e_A) and then encrypts it using B's public key (e_B, n_B):

$$\begin{aligned} C &= E_B[D_A(M)] \\ C &= \left[M^{d_A} \bmod n_A\right]^{e_B} \bmod n_B \end{aligned}$$

Only user B, having the private key triplet (p_B, q_B, e_B), can decrypt the original message and assess its authenticity with A's public key (e_A, n_A):

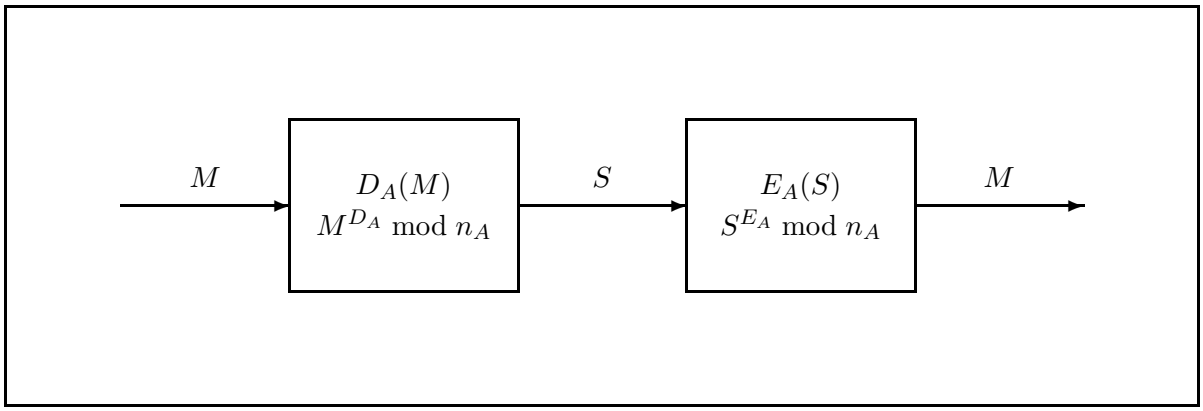

Figure 5: Message authentication using RSA algorithm

$$
\begin{aligned}
M &= E_A[D_B(C)] \\
M &= E_A[D_B[E_B[D_A(M)]]] \\
M &= \Big[\Big[\Big[M^{d_A} \bmod n_A\Big]^{e_B} \bmod n_B\Big]^{d_B} \bmod n_B\Big]^{e_A} \bmod n_A \\
&= \Big[C^{d_B} \bmod n_B\Big]^{e_A} \bmod n_A
\end{aligned}
$$

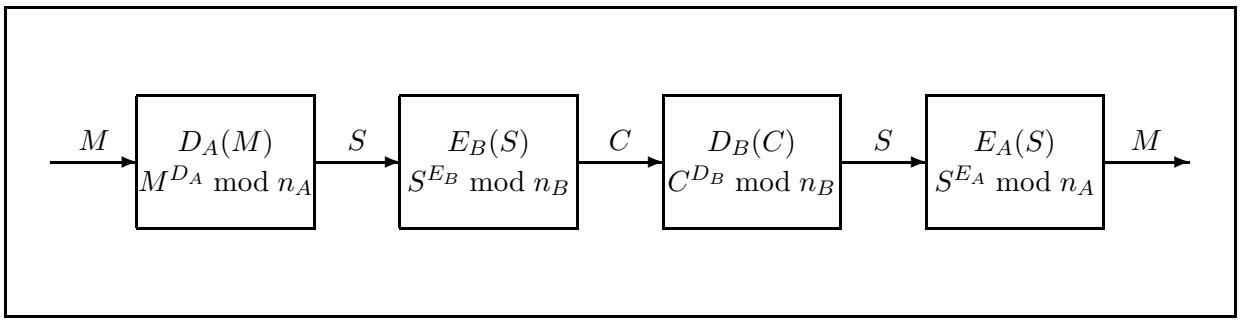

Figure 6: Message secrecy and authentication using RSA

Security of the RSA cryptosystem

1. Computation of e and d: polynomial time:

$$\mathbf{O}(n^t)$$

2. Encryption: polynomial time:

$$C = M^e \bmod n$$

3. Decryption: polynomial time:
$$M = C^d \bmod n$$

4. Computation of the secret decryption key d from the public key e and the modulus value n: non-polynomial time (computationally infeasible)

5. Computation of the plaintext message M from the ciphertext C, the public key e, and n: non-polynomial time (computationally infeasible)

Hence the name *asymmetric algorithm* cryptosystems.

Storage and time requirements

Storage requirements:

As mentioned before, it is suggested to choose a 200 decimal digits number for n; this corresponds to a 664-bits binary word. Each user has to store a 664-bits binary vector for n as well as two more 664-bits vectors for e and d. Therefore, the amount of private key information that each user has to store is approximately 2 $kbits$. The network directory must contain the public key information of each network user, that is a 664-bits vector for n and another 664-bits vector for the public key e, i.e. $\approx 1,3$ $kbits$ per network user. By comparison, the storage requirement for a Data Encryption Standard key is only 56 $bits$.

Time requirements:

The computation time requirements for RSA are 1 to 2 multiplications per bit (modular arithmetics) or roughly 1000 operations for a 664-bit number n. With special purpose chips, a few thousands bits can be processed per second, while for DES processing rates of up to 100 $Mbits/s$ are achievable today.

Diffie-Hellman Key Exchange Algorithm

In 1976, Diffie and Hellman [DH76] published the first public key based algorithm which was designed to provided a means to exchange securely a key K over a public network. That key K can later be used as a session key. Note however that this algorithm applies only to the exchange of keys.

Before the key exchange actually begins, two global public values are generated and made public to everyone: a prime number q and α, a *primitive root* of q, where $\alpha < q$.

A primitive root a of a prime number p is an integer for which the *successive powers* modulo p:

$$a \bmod p, a^2 \bmod p, a^3 \bmod p, \ldots, a^i \bmod p, \ldots, a^{(p-1)} \bmod p$$

generates $(p-1)$ distinct integers, without repetition. The sequence $\{a^i \bmod p\}_{i=0,\ldots,(p-1)}$ is thus a permutation of the integers from $i = 1$ to $(p-1)$.

For any integer $b = a^i \bmod p$, the exponent i is refered to as the *discrete logarithm* (or index) of b in base a modulo p.

Suppose that Alice, A, and Bob, B, want to exchange a key K. Alice selects a secret random number $X_A < q$, compute the public value $Y_A = \alpha^{X_A} \bmod q$ and makes it public. Bob does the same, that is, he generates his private random number $X_B < q$, calculate $Y_B = \alpha^{X_B} \bmod q$ and publishes Y_B. Then Alice computes the key K as follows:

$$\begin{aligned} K &= (Y_B)^{X_A} \bmod q \\ K &= \left(\alpha^{X_B} \bmod q\right)^{X_A} \bmod q \\ K &= \left(\alpha^{X_B}\right)^{X_A} \bmod q \qquad \text{(principle of modular arithmetics)} \\ K &= \alpha^{X_B X_A} \bmod q \end{aligned}$$

Similarly, Bob computes K using the public information and his own private random number:

$$\begin{aligned} K &= (Y_A)^{X_B} \bmod q \\ K &= \left(\alpha^{X_A} \bmod q\right)^{X_B} \bmod q \\ K &= \left(\alpha^{X_A}\right)^{X_B} \bmod q \\ K &= \alpha^{X_A X_B} \bmod q = \alpha^{X_B X_A} \bmod q \qquad \text{(same key } K\text{)} \end{aligned}$$

The security of the algorithm is based on the difficulty to compute discrete logarithms. While it is easy to compute the modular exponentials $Y = \alpha^X \bmod q$ knowing the secret number X, it is difficult for an attacker to compute X from Y and the global public values α and q:

$$\begin{aligned} Y &= \alpha^X \bmod q \qquad \text{(modular exponentiation: easy)} \\ X &= \log_\alpha Y \bmod q \qquad \text{(discrete logarithm: difficult)} \end{aligned}$$

Hybrid cryptosystems (public key)

RSA: Slow to communicate information

DES: Impossible to use in a public key context or may not be sufficiently secure

In a hybrid cryptosystem, the public key cryptosystem (e.g. RSA) may be used to exchange a limited amount of secret information which, for instance, may be used then as a secret key for encryption and decryption for a symmetric algorithm (e.g. DES). Then the symmetric cryptosystem can encrypt the message data itself at a much higher rate.

Note 1: Keys can also be updated frequently using this type of hybrid cryptosystem approach. A cryptanalyst, succeeding in breaking the symmetric cryptosystem secret key will decrypt only the corresponding part of the plaintext message before the key is updated.

Note 2: The information transmission rate, in such an hybrid cryptosystem, won't be much slowed down by the use of the public key cryptosystem.

References

[DH76] W. Diffie and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22:644–654, November 1976.

[RSA78] R.L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communin. Ass. Comput. Mach. (ACM)*, 21(2):120–126, February 1978.

[SS79] R. Schroeppel and A. Shamir. A TS2 = O(2n) Time/Space Tradeoff for Certain NP-Complete Problems. In *Proceedings of the IEEE 20th Annual Symposium on the Foundations of Computer Science*, Washington, D.C, October 1979.