Network Security - an overview

- Network Security has a number of elements; we are going to discuss here the following four:
  - protection against eavesdropping – confidentiality and privacy (note the difference between these two)
  - protection against user impersonation - authentication
  - protection against message alteration - message integrity
  - protection against denial of service
  - protection against un-authorized access
- Examples of possible security attacks:
  - <u>passive intruder:</u> eavesdropping
  - <u>active intruder:</u> message alteration, message injection (impersonation), reply attack, message deletion (denial of service)

There is no "absolute security"

- (Security) risk of a system (or information/data, or an algorithm, or a protocol, or a process, or a procedure, etc) depends on the cyber resources and the time available to the attacker.
- Given enough time, every system can be compromised.
- Given sufficient resources (which depends on the state of the technology), every system can be compromised.
- However, (nearly always) time is of essence; i.e., (most often) information is of value for a limited time duration; e.g.,
  - tactical information (hours), strategic information (weeks, months, years), national security (decades), personal information (lifetime), etc

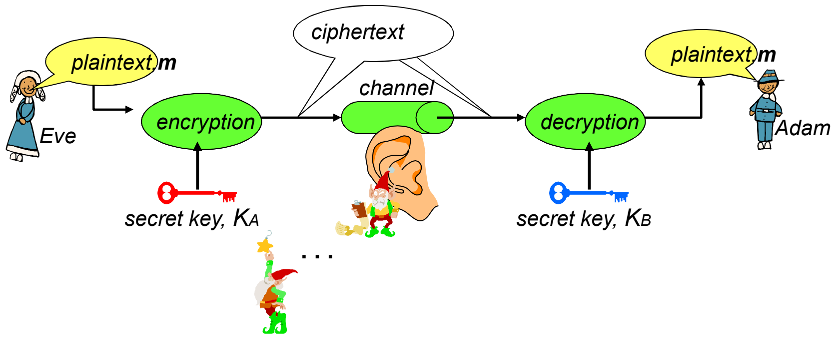
Network Security - an overview

- We start by discussing three basic schemes used in network security:
  - the private-key cryptography (the *Data Encryption Standard - DES*)
  - the public-key cryptography (the *Rivest, Shamir, and Adleman - RSA)*
  - anonymous key distribution (the *Diffie-Hellman Key Exchange)*
- These schemes are representative examples of the corresponding cryptographic tools.
- Some of those schemes rely on existence of "one-way function." An example of which is exponentiation over a finite field. I.e., it's easy to find $m = a^b \bmod n$. But given *m*, *a,* and *n,* it's "extremely hard" (what does this mean?) to find *b*.

Wireless Networks Laboratory Copyright ©by Zygmunt J. Haas, 2017

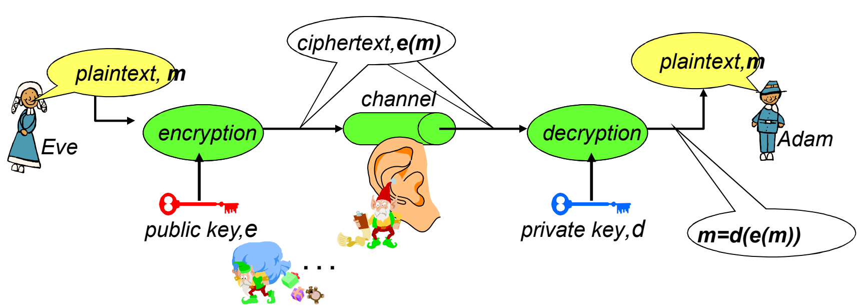
Network Security - basic schemes

The Private-Key Cryptography



Network Security - basic schemes

The Public Key Cryptography



Network Security - public key cryptography

- The RSA (Rivest, Shamir, Adelman) algorithm requires two elements:
  (1) a pair of keys - a public and a private key
  (2) an encryption/decryption algorithm
- The RSA algorithm relies of the fact that there are no known algorithms that can reasonably fast factor a number into its prime components; i.e., *n* into *p* and *q*.

(1) Choosing the key pair:
- select two large prime numbers, *p* and *q*. (The recommended size of *p* and *q* is 768[bits] for personal applications and 1024 [bits] for cooperate use.)
- $n=pq; z=(p-1)(q-1)$
- select a number $e<z$ which is a prime relative to z (e and z have no common factors, except 1)
- select a number *d,* such that $ed-1$ is divisible by z; ($ed \bmod z = 1$)
- the public key is (n,e) and the private key is (n,d)

Network Security - public key cryptography

(2) The encryption/decryption algorithm will then be:
- a message *m* ($m<n$) is encrypted as ciphertext *c:*

$$c = m^e \bmod n$$

- a ciphertext is decrypted as follows:

$$m' = c^d \bmod n$$

We will show that

$$m' = m$$

Proof:

$$m' = \left(m^e\right)^d \bmod n = m^1 \bmod n = m$$

In the above, we have used the fact that, if *p* and *q* are prime and $n=pq$, then (Fermat's Little Theorem):

$$a^{(b \bmod ((p-1)(q-1))} \bmod n = a^b \qquad \text{and } m<n.$$

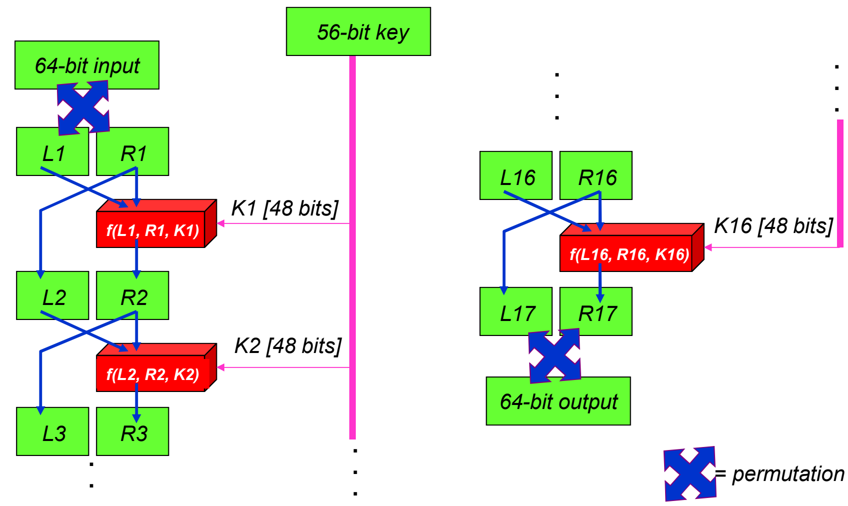
Network Security - the RSA (Rivest, Shamir, Adleman algorithm - an example

- Assume that we chose p=5 and q=11 (both are relatively prime).
- Thus, n=pq=55 and z=(p-1)(q-1)=40.
- Now we need to select a number that is relatively prime to 40, say 13 (13<n). So, e=13 and the public key is (13,55)
- We select d=37 (as 13*37 mod 40 = 1). Thus the private key is (37,55).
- Now assume that our message m=7. Thus, the ciphertext is: $7^{13} \bmod 55 = 2$.
- To decode the message, we do $2^{37} \bmod 55 = 7$!

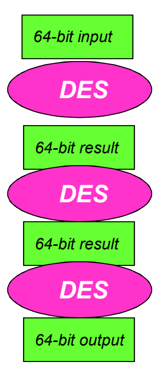
Network Security - The Data Encryption Standard (DES) (now obsolete)

- Based on IBM's LUCIFER algorithm.
- Adopted by National Bureau of Standards (NBS).
- The banking industry adopted DES as a wholesale banking standard. (Standards for the wholesale banking industry are set by the American National Standards Institute (ANSI)).
- ANSI X3.92, adopted in 1980, specified the use of the DES algorithm.

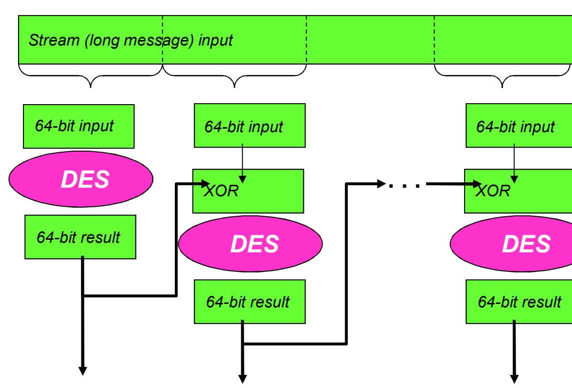
Network Security - The Data Encryption Standard (DES) (now obsolete)



Network Security - The Data Encryption Standard (DES) (now obsolete)



To improve the security of DES, *triple-DES (3DES)* could be used.



Also, stream plaintext is encoded using *cipher-block chaining.*

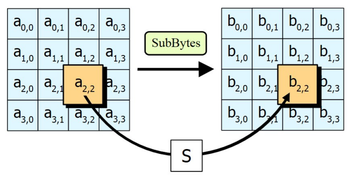
The Advanced Encryption Standard (AES) [NIST 2001] (replaced DES)

- Based on *Rijndael cipher* (after the inventors Joan Daemen and Vincent Rijmen)
- The key size determines the number of repetitions of transformation rounds that convert the plaintext into ciphertext.
- The number of cycles of repetition are:
- 128-bit keys: 10 cycles of repetition
- 192-bit keys: 12 cycles of repetition
- 256-bit keys: 14 cycles of repetition
- Each round consists of four step (except for the initial and final rounds); the fourth step depends on the encryption key itself.
- For decryption, a set of reverse rounds are applied that transform the ciphertext back into the original plaintext. It uses the same encryption key (i.e., it's a symmetric-key crypto-system).

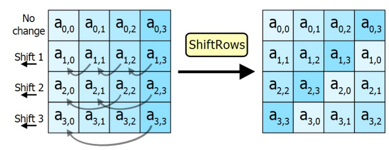
The Advanced Encryption Standard (AES) [NIST 2001] (replaced DES)

- <u>Key Expansion</u>: *round keys* are derived from the cipher key. (AES requires a separate 128-bit *round key* for each round + one more *round key)*.
- Initial Round consists of the following step:
  - <u>AddRoundKey</u>: each byte of the state is combined with a block of the *round key* using bitwise XOR operation.
- Each subsequent round consists of the following steps:
  - <u>SubBytes:</u> a non-linear substitution step where each byte is replaced with another according to a lookup table.
  - <u>ShiftRows:</u> a transposition step where rows of the state are shifted (cyclically) a certain number of steps.
  - <u>MixColumns:</u> a mixing operation, which combines the four bytes in each column.
  - <u>AddRoundKey:</u> round key is bitwise XORed with the state
- Final Round includes the following steps (no MixColumns step): SubBytes, ShiftRows, and AddRoundKey

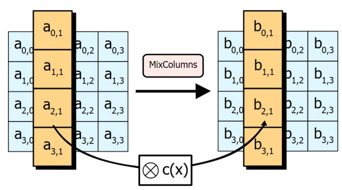
The Advanced Encryption Standard (AES) [NIST 2001] (replaced DES)



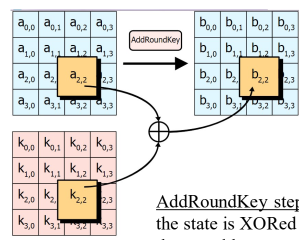
SubBytes step: each byte in the state is replaced with its entry in a fixed 8-bit lookup table, S; $b_{ij} = S(a_{ij})$.



ShiftRows step: bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs for each row



MixColumns step: each column of the state is multiplied with a fixed polynomial $c(x)$.



AddRoundKey step: each byte of the state is XORed with a byte of the round key

Network Security - key distribution problem - Diffie-Hellman Key Exchange

- The problem with the above schemes is that to create a secure channel, it is required to (securely) establish session keys. But in order to communicate such keys securely, we need a secure channel.
- Is it possible to establish a secure channel without prior secure communication over the channel?
- Yes - a simple approach is to distribute the keys in a different way, say by storing the keys at the manufacturing time. But this may not be a feasible solution. Why? …
- But there is another way …

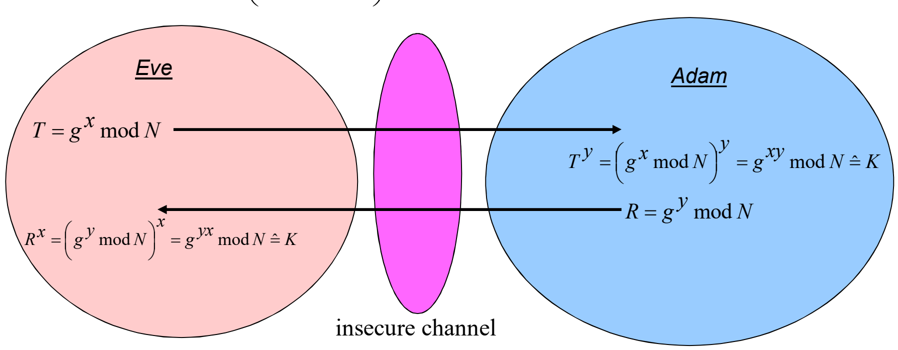
Network Security - key distribution problem - Diffie-Hellman Key Exchange

- *The Diffie-Hellman Key Exchange* scheme relies on the fact that exponentiation over a finite field is a "one way function," i.e., it's easy to find $m = a^b \bmod n$. However, given *m*, *a,* and *n,* it's "extremely hard" to find *b.*
- *Diffie-Hellman Key Exchange:*
  - both Eve and Adam agree on a large prime number, *N,* and a generator, *g*
  - Eve picks a random number, *x,* and computes: $T = g^x \bmod N$
  - Adam picks a random number, *y,* and computes: $R = g^y \bmod N$
  - Eve sends *T* to Adam, and Adam send *R* to Eve
  - Adam computes the share secret, *K,* as: $T^y = \left(g^x \bmod N\right)^y = g^{xy} \bmod N \triangleq K$

Network Security - key distribution problem - Diffie-Hellman Key Exchange

- Eve computes the share secret, *K,* as

$$R^x = \left(g^y \bmod N\right)^x = g^{yx} \bmod N \triangleq K$$



Network Security - key distribution problem - Diffie-Hellman Key Exchange

- Example: $N=29$ and $g=7$

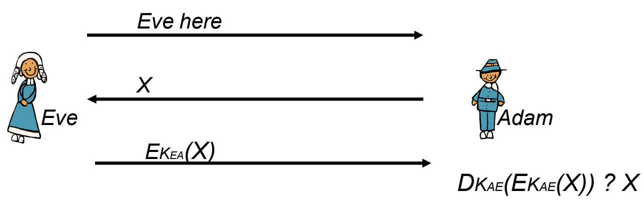
Eve		*Adam*
$x=4$		$y=3$
$g^x \bmod N = 7^4 \bmod 29 = 23$	**23** →	$g^y \bmod N = 7^3 \bmod 29 = 24$
	← **24**	
$K = 24^4 \bmod 29 = 16$		$K = 23^3 \bmod 29 = 16$

- Even knowing $N=29$ and $g=7$, and the values of *23* and *24*, it is impossible to get *4* and *3,* respectively; i.e., $\log_g T$ and $\log_g R$ are very difficult to do over finite field.

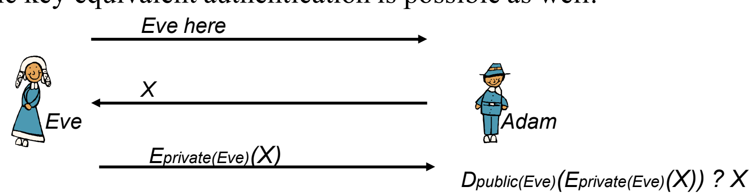
Network Security - Authentication

- An example of an authentication protocol is;
  - Eve sends a message to Adam.
  - Adam chooses a *nonce, X*, and sends it to Eve (in the clear).
  - Eve encrypts X using a shared (symmetric) key, K_{AE}, and sends the encrypted value $EK_{AE}(X)$ to Adam.
  - Adam decrypts the $EK_{AE}(X)$ using the secret key and if $DK_{AE}(EK_{AE}(X)) = X$, then Eve is authenticated.
- In the above scheme authentication is performed by verifying that Eve is in the possession of the shared key, K_{AE}.
- Use of nonce ensures freshness of the authentication; i.e., prevents reply attacks.

Network Security - Authentication (con't)



- A public key equivalent authentication is possible as well:



- This assumes that public keys can be securely distributed.

