T-79.4501 Cryptography and Data Security

Lecture 10:
10.1 Random number generation
10.2 Key management
- Distribution of symmetric keys
- Management of public keys

Stallings: Ch 7.4; 7.3; 10.1

The Use of Random Numbers

- Random numbers are an essential ingredient in most (if not all) cryptographic protocols: there is no security without *apparent randomness* and *unpredictability*; things must look random to an external observer.
- Cryptographic keys
  - symmetric keys
  - (private) keys for asymmetric cryptosystems
- Cryptographic nonces (= **n**umbers used **once**) to guarantee freshness
  - random numbers with some additional properties

Random and pseudorandom numbers

Random numbers are characterized using the following statistical properties:

- Uniformity: Random numbers are uniformly distributed
- Independence: generated random numbers cannot be derived from other generated random numbers
- Generated using physical devices, e.g, quantum random number generator

Pseudorandom numbers are non-random numbers that cannot be distinguished from random numbers:

- Statistical distribution of a sample of certain (large) size cannot be distinguished from the uniform distribution
- Independent-looking: pseudorandom numbers should be unpredictable: given a sequence of previously generated pseudorandom numbers nothing cannot be said about the future terms of the sequence
- Generated using deterministic algorithms from a short truly random or pseudorandom seed.

Linear Congruential Generator (Lehmer 1951)

m	the modulus, $m > 0$
a	the multiplier, $0 < a < m$
c	the increment, $0 \le c < m$
x_0	the starting value, or seed

The sequence of pseudorandom numbers is computed as

$$x_{n+1} = (ax_n + c) \bmod m, \quad n = 0,1,2,\ldots.$$

Example: $m = 32$; $a = 7$; $c = 0$, $x_0 = 7$; then $x_0 = 7$, $x_1 = 17$, $x_2 = 23$, $x_3 = 1$, $x_4 = 7, \ldots$ The period of the sequence is 4. This is due to the fact that the order of 7 modulo 32 is equal to 4.

The period should be large. This can be achieved by suitable choice of the numbers: IBM360 family of computers used LCG with $a = 16807 = 7^5$; $m = 2^{31} - 1$; $c = 0$.

Weaknesses of LCG

- Given the parameters a, c and m, and just one term of the generated sequence, one can compute any term after and before this term.
- Assume a, c and m are unknown. Then given just four known terms x_0, x_1, x_2, x_3 of the generated sequence, one gets a system of equations:

$$x_1 = (ax_0 + c) \bmod m$$
$$x_2 = (ax_1 + c) \bmod m$$
$$x_3 = (ax_2 + c) \bmod m$$

  from where one has good chances to solve for a, c and m.
- Linear Feedback Shift Registers (LFSR) are very similar to LCG: good statistical properties, but no cryptographic security in itself. Given an output sequence of length that is 2 times the length of the LFSR, one can solve for the feedback coefficients. Therefore, LFSRs are used only as a part of a construction for a cryptographically secure key stream or pseudorandom number generator.

Cryptographic PRNGs

The security requirements for a cryptographically secure pseudorandom number generator are similar than those for a keystream generator. In practice, the difference lies in the fact that keystream generators are used for encryption and must be fast, and consequently, security is traded off to achieve the required speed. Pseudorandom number generators are used generate short strings, cryptographic keys and nonces, and therefore security is more important than speed.

Some standard PRNGs:

- Counter mode keystream generator is a cryptographically strong PRNG
- ANSI X9.17 PRNG based on Triple DES with two keys in encryption-decryption-encryption mode.
- FIPS 186-2 specifies a random number generator based on SHA-1 for generation of the private keys and per-message nonces.
- Blum-Blum-Shub generator is provably secure under the assumption that factoring is hard.

Counter Mode PRNG

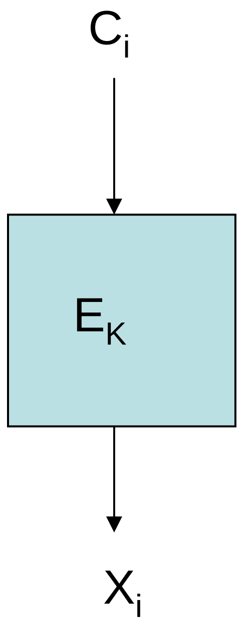
Also known as Cyclic Encryption (Meyers 1982). It consists of a counter with period N and an encryption algorithm with a secret key.

IV	Initial value of the counter C
K	Key of the block cipher encryption function E_K
X_i	i-th pseudorandom number output

$C_0 = IV;$

$C_i = C_{i-1}+1;$

$X_i = E_K(C_i)$, $i = 1,2,\ldots$



The period is N. If the length of the counter is less than the block size of E_K then all generated numbers within one period are different.

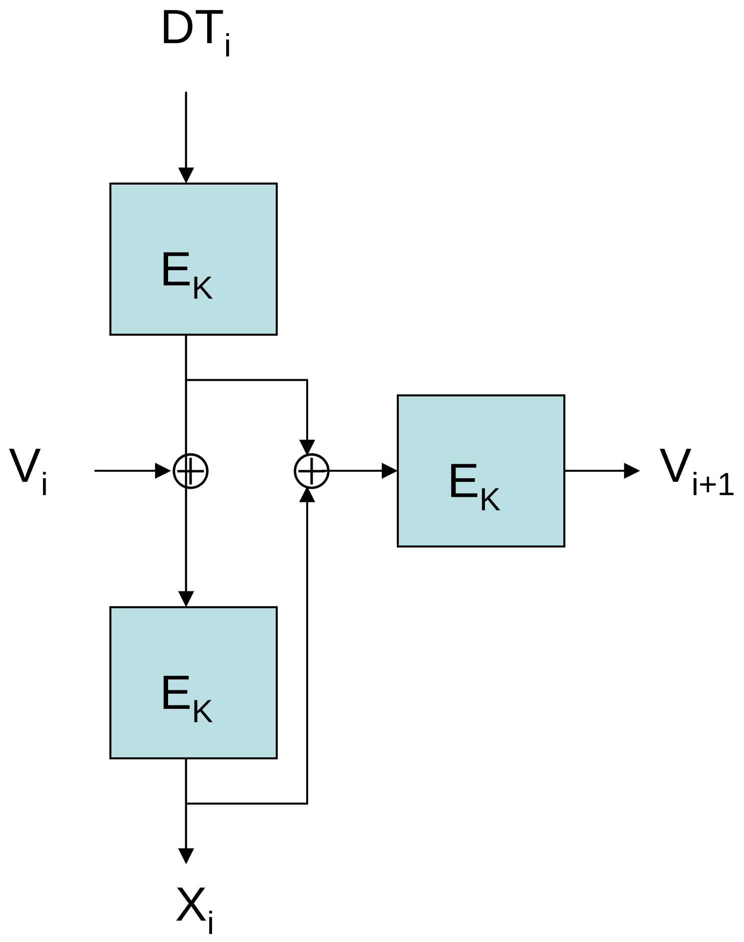
ANSI X9.17 PRNG

DT_i	64-bit time variant parameter, date and time
V_i	seed variable
E_K	3-DES encryption with two 56-bit keys K_1 and K_2, $K = (K_1,K_2)$
X_i	i-th pseudorandom number output

$X_i = E_K(V_i \oplus E_K(DT_i))$,

$V_{i+1} = E_K(X_i \oplus E_K(DT_i))$,

$i = 1,2,\ldots$



FIPS 186-2 PRNG for generation of per-message random numbers k_j for DSA

m number of messages to be signed

q the 160-bit prime in the definition of DSA

$KKEY_0$ initial 512-bit seed

$KKEY_j$ 512-bit seed variable

t the fixed initial value (a cyclic shift of the initial value CV_0 of SHA-1, see Lecture 5)

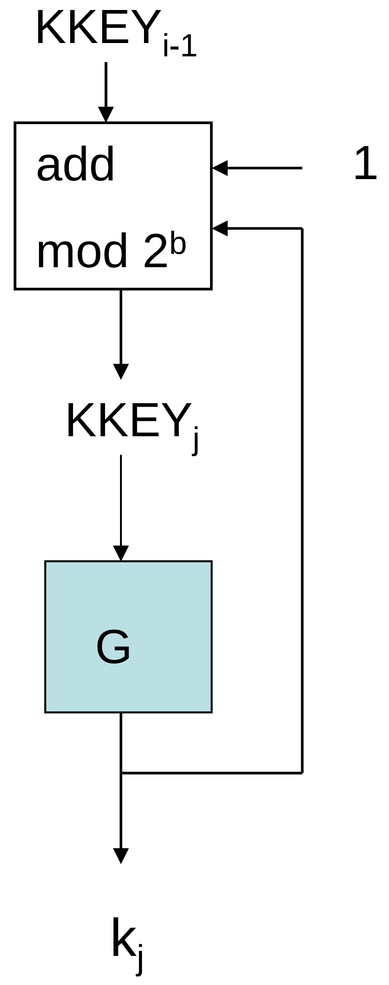
G(t,c) operation of SHA-1 on one 512-bit message block M (without length appending)

M = formed by appending seros to the end of c

k_j j-th per-message pseudorandom number

$k_j = G(t, KKEY_j) \bmod q$

$KKEY_{j+1} = (1 + KKEY_j + k_j ,) \bmod 2^{512}$, j = 0,1,…,m-1



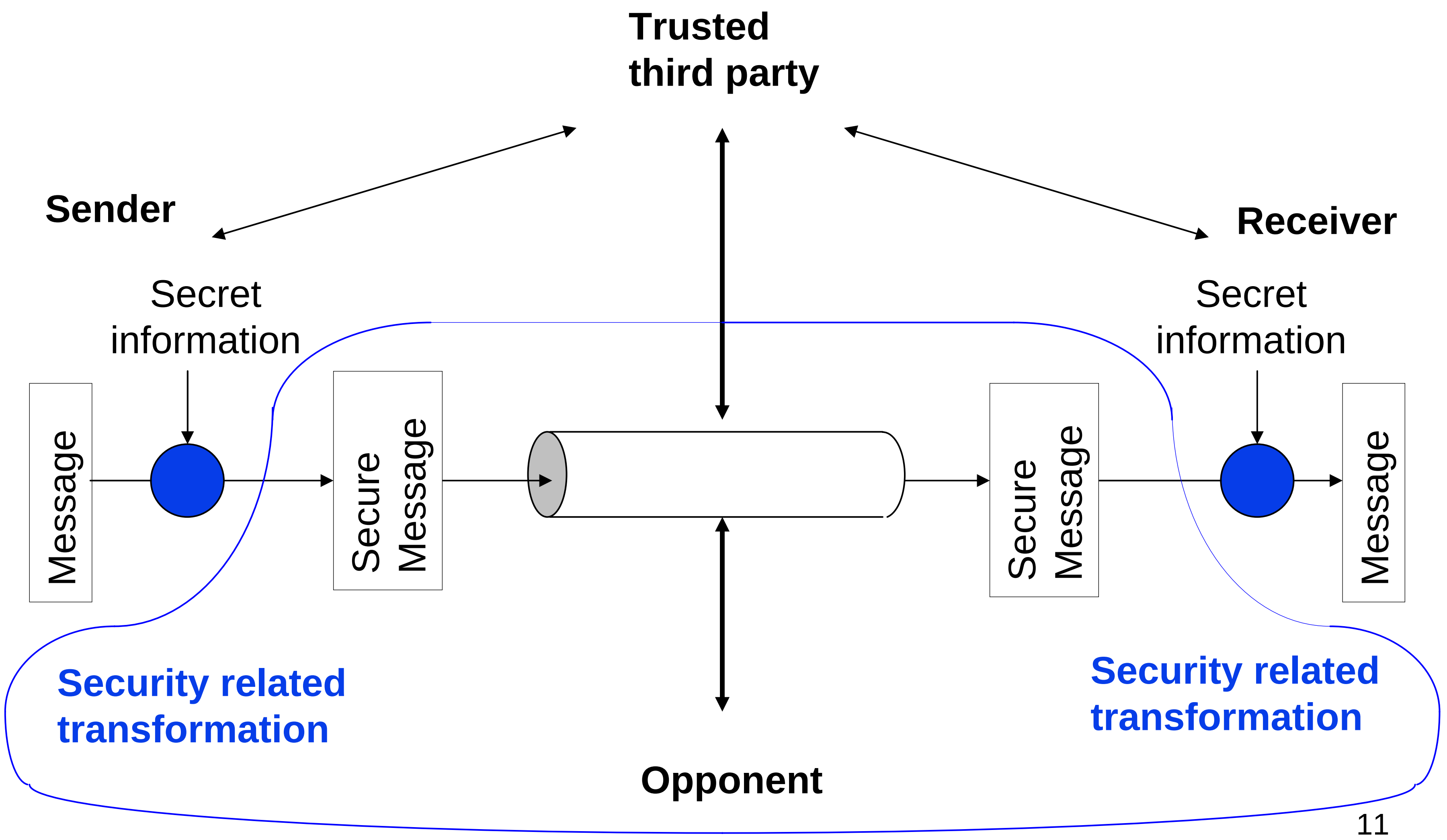
Blum-Blum-Shub

- Cryptographically provably secure PRNG
- Very slow, output 1 pseudorandom bit per one modular squaring modulo a large integer

p, q	two different large primes; $p = q = 3 \pmod 4$
n	modulus, $n = pq$
s	secret seed; set $x_0 = s^2 \bmod n$
x_i	i-th intermediate number
B_i	i-th output bit

```
For i = 1,2,…
    $x_i = (x_{i-1})^2 \bmod n$
    $B_i = x_i \bmod 2$
```

Model for network security



Distribution of symmetric keys

Distribution of shared symmetric keys for A and B; using one of the following options:

1. Physically secured

- A selects or generates a key and delivers it to B using some physically secure means
- A third party C selects a key and delivers it to A and B using some physically secure means

2. Key distribution using symmetric techniques

- If A and B have a shared secret key, A can generate a new key and send it to B encrypted using the old key
- If C is already using a shared secret key K_1 with A and a second key K_2 with B, then C can generate a key and send it encrypted to A and B.

3. Key management using asymmetric techniques

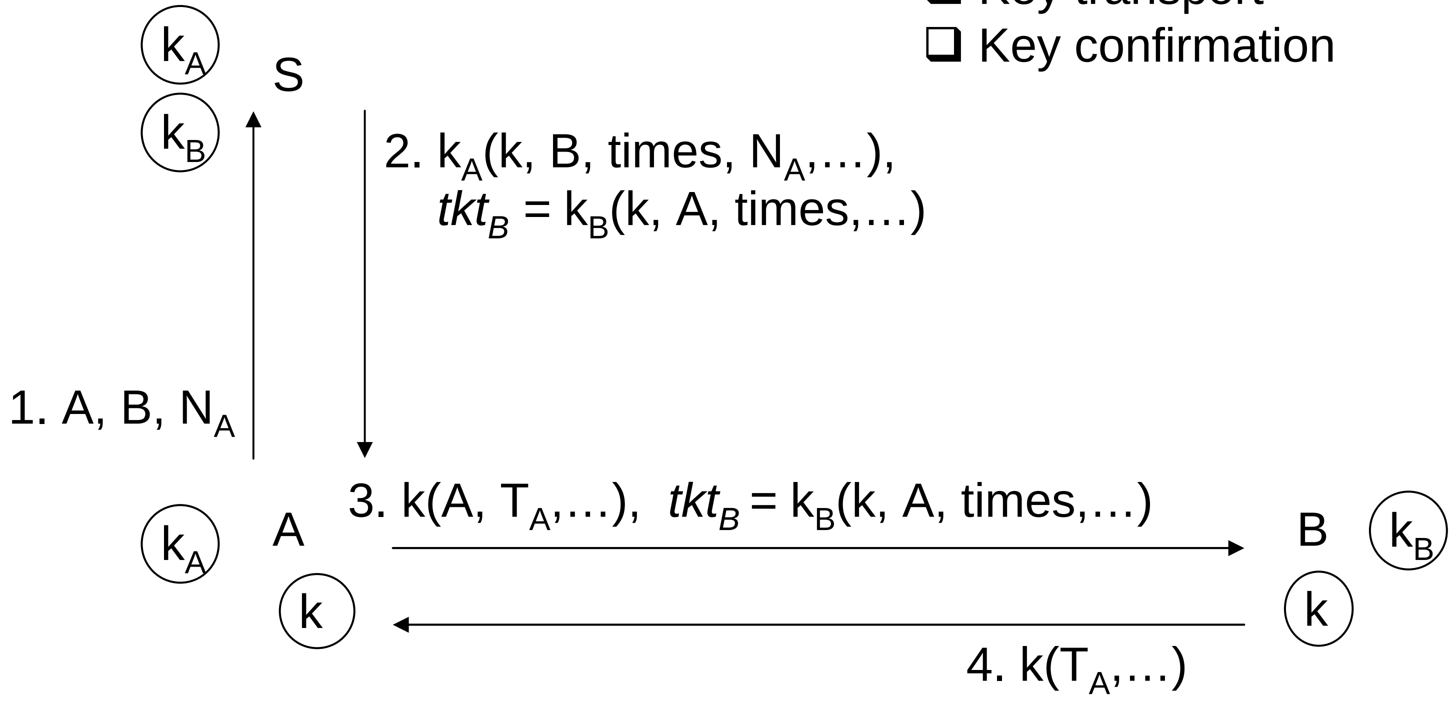
- If Party A has a public key of B, then A can generate a key and send it to B encrypted using a public key
- If party C has the public key of A and the public key of B, it can generate a key and send it to A and B encrypted using their public keys.

Key Hierarchy

1. Master Keys
   - long term secret keys
   - used for authentication and session key set up
   - Distributed using physical security or public key infrastructure
2. Session Keys
   - short term secret keys
   - used for protection of the session data
   - distributed under protection of master keys
3. Separated session keys
   - short term secrets
   - to achieve cryptographic separation: Different cryptographic algorithms should use different keys. Weaknesses in one algorithm should not endanger protection achieved by other algorithms.
   - derived from the main session key

Example: Kerberos

- Prior enrollment with server
- Timestamps to ensure freshness
- Key transport
- Key confirmation

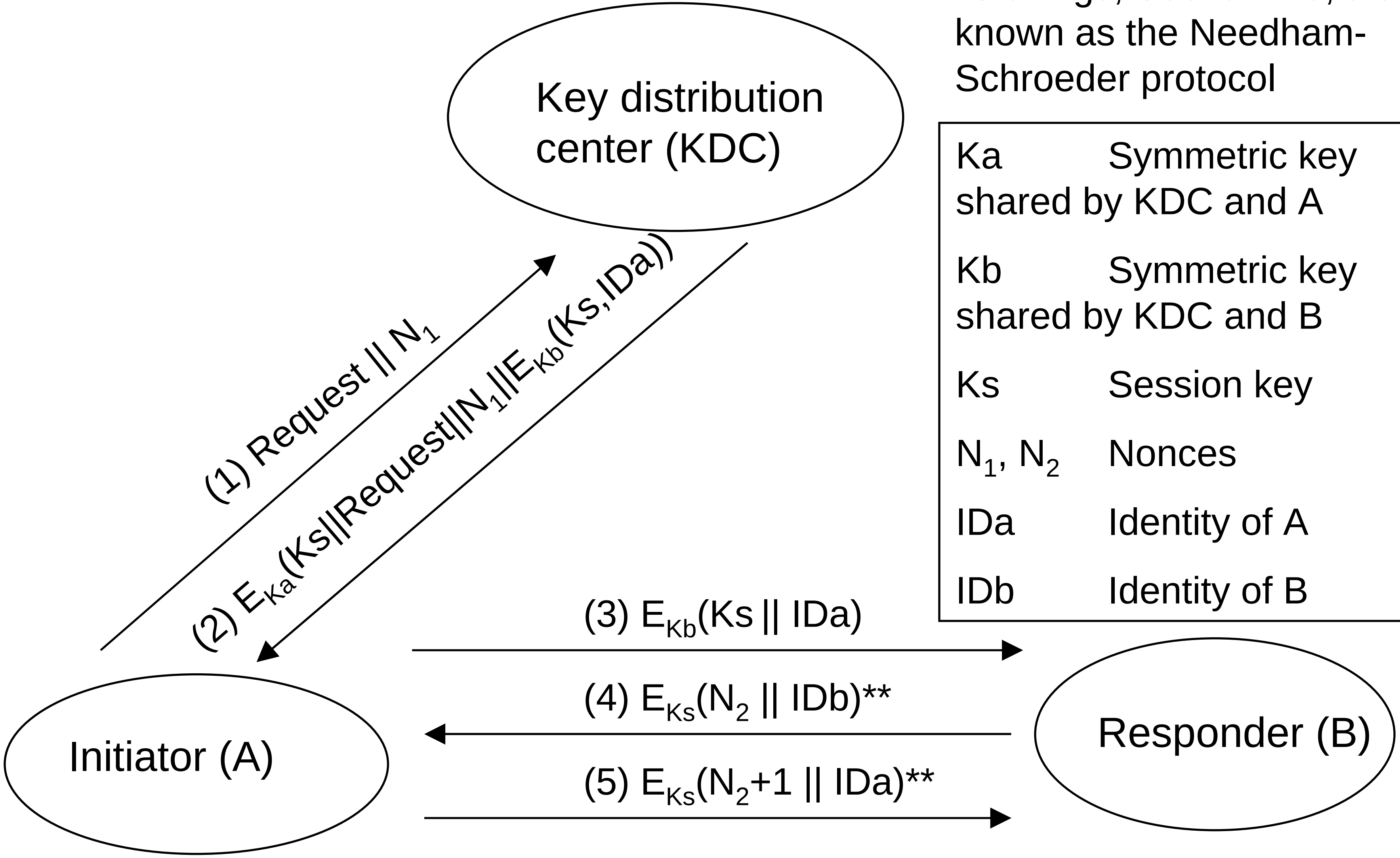

k_A
k_B
S
1. A, B, N_A
2. k_A(k, B, times, N_A,…),
tkt_B = k_B(k, A, times,…)
3. k(A, T_A,…), tkt_B = k_B(k, A, times,…)
k_A
A
k
B
k_B
k
4. k(T_A,…)

Needham-Schroeder protocol (1978)

- An earlier version of the Kerberos protocol (without time-stamps)
  - B had no guarantee of the freshness of the ticket tkt_B. If Malice knows some previous key used by A and B it can force B to use the key again by replaying the corresponding ticket.
- Depicted on the next slide

A Key Management Scenario*

*Stallings, Section 7.3; also known as the Needham-Schroeder protocol



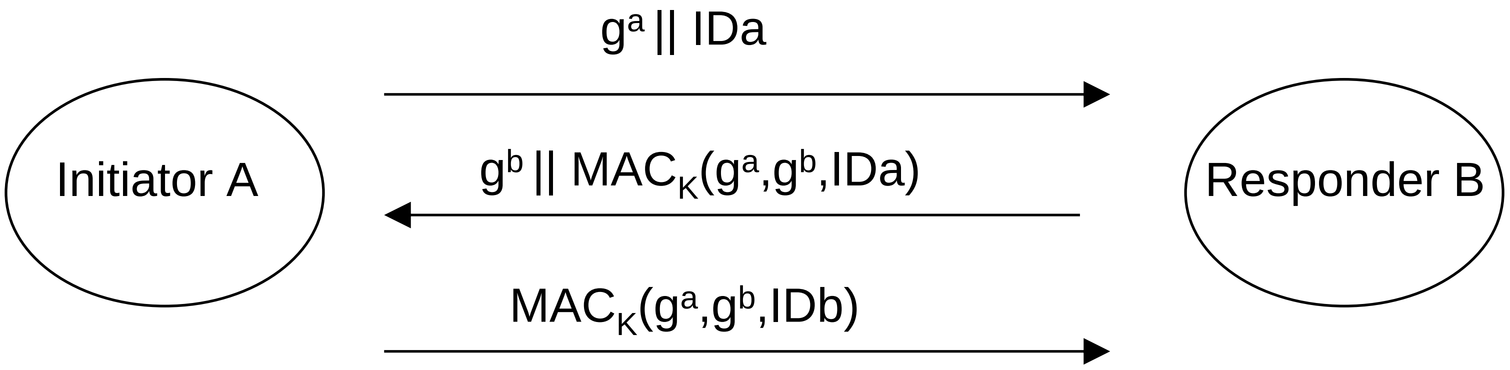
Key distribution center (KDC)
(1) Request || N_1
(2) $E_{Ka}(Ks||Request||N_1||E_{Kb}(Ks,IDa))$
(3) $E_{Kb}(Ks\ ||\ IDa)$
(4) $E_{Ks}(N_2\ ||\ IDb)$**
(5) $E_{Ks}(N_2+1\ ||\ IDa)$**
Initiator (A)
Responder (B)

Ka	Symmetric key shared by KDC and A
Kb	Symmetric key shared by KDC and B
Ks	Session key
N_1, N_2	Nonces
IDa	Identity of A
IDb	Identity of B

** slightly modified from Stallings' protocol

Authenticated Diffie-Hellman Key Exchange

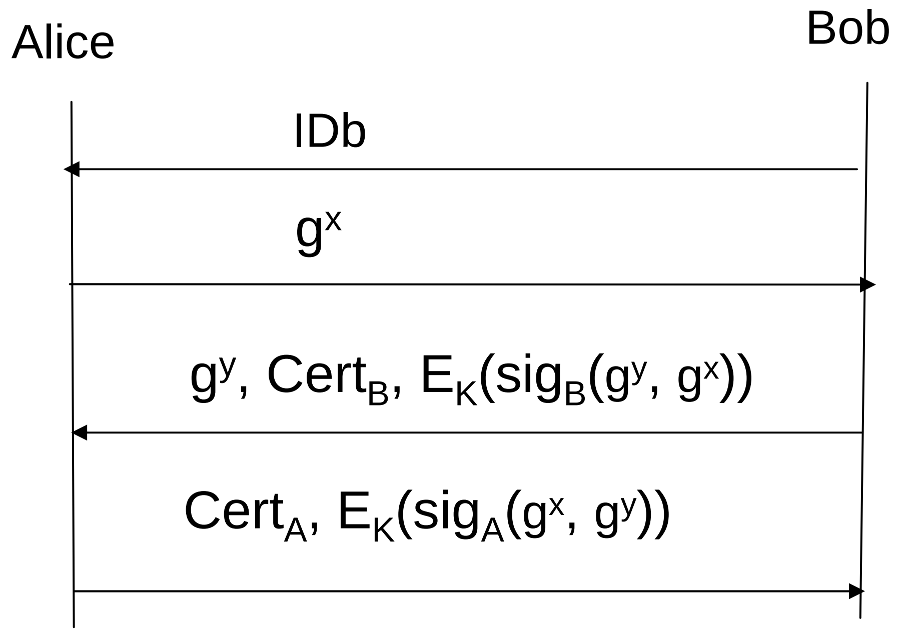
Recall: Diffie-Hellman Key Exchange provides confidentiality against passive wiretapper. Active man-in-the-middle attack can be prevented using authentication, e.g. as follows:



K	Authentication key shared by A and B
a	private exponent of A
IDa	Identity of A
IDb	Identity of B

Station-to-Station (STS) Protocol: Authenticated Diffie-Hellman

- Provides *perfect forward secrecy (PFS)*: compromise of long term private keys does not compromise past session keys
- *PFS* requires the use of public key cryptography


Alice
Bob
IDb
g^x
g^y, $Cert_B$, $E_K(sig_B(g^y, g^x))$
$Cert_A$, $E_K(sig_A(g^x, g^y))$

Desirable AKE Attributes Law, Menezes, Qu, Solinas, Vanstone (1998)

known-key security. Each run of a key agreement protocol between two identities A and B should produce a unique secret key; such keys are *session keys.* A protocol should still achieve its goal in the face of an adversary who has learned some other session keys.

(perfect) forward secrecy. If long-term private keys of one or more entities are compromised, the secrecy of previous session keys established by honest entities is not affected.

key-compromise impersonation. Suppose A's long-term private key is disclosed. Clearly an adversary that knows this value can now impersonate A, since it is precisely this value that identifies A. However, it may be desirable that this loss does not enable an adversary to impersonate other entities to A.

unknown key-share. Entity A cannot be coerced into sharing a key with entity B without A's knowledge, i.e., when A believes the key is shared with some entity C $\neq$ B, and B (correctly) believes that the key is shared with A.

key control. Neither entity should be able to force the session key to a pre-selected value.

key confirmation. Both entities get an explicit proof that the other entity has established the same key.

Distribution of Public Keys

- Public announcement
  - Just appending one’s public key, or the fingerprint (hash) of the public key in one’s signed email message is not secure
  - PGP public key fingerprints need to be truly authenticated based on face-to-face or voice contact
- Publicly available directory
  - An authorized directory, similar to phone directory that is published in print
- Public-key Authority
  - Public keys obtained from an online service. Communication needs to be secured.
- Public-key Certificates
  - Public keys bound to user’s identities using a certificate signed by a Certification Authority (CA)

X509 Public Key Certificates

Mandatory fields

- The version number of the X.509 standard
- The certificate serial number
- The CA’s Signing Algorithm Identifier
- The name of the issuing CA
- The validity period (not before date, not after date)
- The subject’s name, i.e. whose public key is being signed
- The subject’s public key value, including the algorithm and associated domain parameters
- The issuer’s signature on the public key and all other data that is to be bound to the subject’s public key such as the subject’s name, the validity period and other terms of usage of the subject’s public key.

CA and Registration Authority

Certification Authority

- E.g. in Finland: Population Register Center
- The certificate is stored in the subject’s Electronic Identity Card

Registration Authority

- Identifies the user based on user’s true identity and establishes a binding between the public key and the subject’s identity

Management of private keys

- Private keys generated by the user
- Private key generated by a trusted authority
- Private key generated inside a smart card from where it is never taken out. The public key is taken out.

Certificate Revocation List

- Black list for lost or stolen private keys
- CRL must be available online for certificates with long validity period