Key Distribution and Management

Key Distribution

- Both symmetric key schemes and public key schemes require both parties to acquire valid keys.
- In symmetric key schemes, the shared key should be securely distributed between the source and destination, while protecting it from others.
- Frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key.
- A new session key should be used for each new connection-oriented session. For a connectionless protocol, a new session key is used for a certain fixed period only or for a certain number of transactions.
- On many occasions systems have been broken, not because of a poor encryption algorithm, but because of poor key selection or management.

Key Distribution

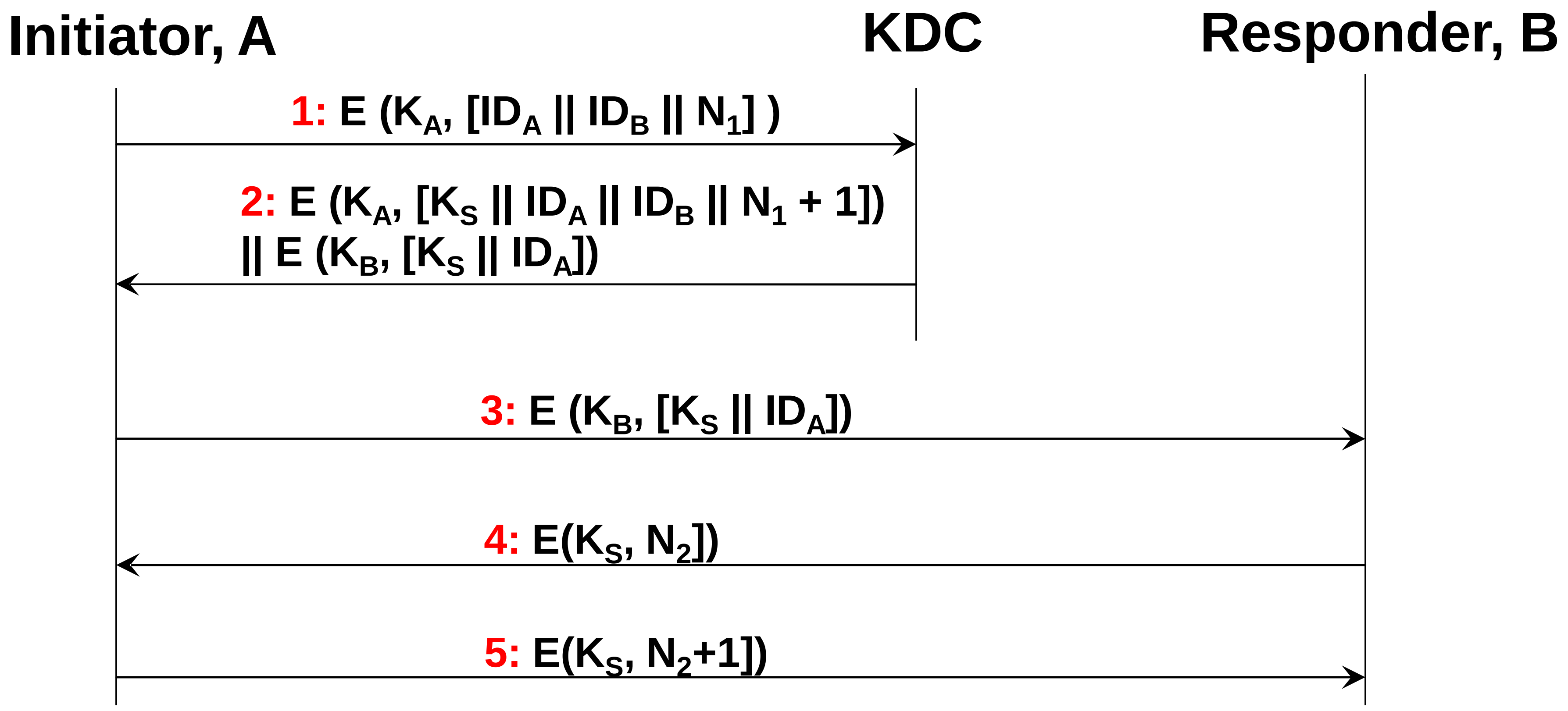
- given parties A and B have various **key distribution** alternatives:
    - A can select key and physically deliver to B
    - third party can select & deliver key to A & B
    - if A & B have communicated previously can use previous key to encrypt a new key
    - if A & B have secure communications with a third party C, C can relay key between A & B
- Preferred Approach, especially for scalability - A third party, whom all parties trust, can be used as a trusted intermediary to mediate the establishment of secure communications between users.

Key Distribution Techniques

- Centralized
  - Needham-Schroeder Protocol
- Distributed
  - Diffie-Hellman Key Exchange
- Use of Public Key Certificates for Validating Authentication during Key Distribution

Needham-Schroeder Protocol for Secure Key Distribution and Authentication

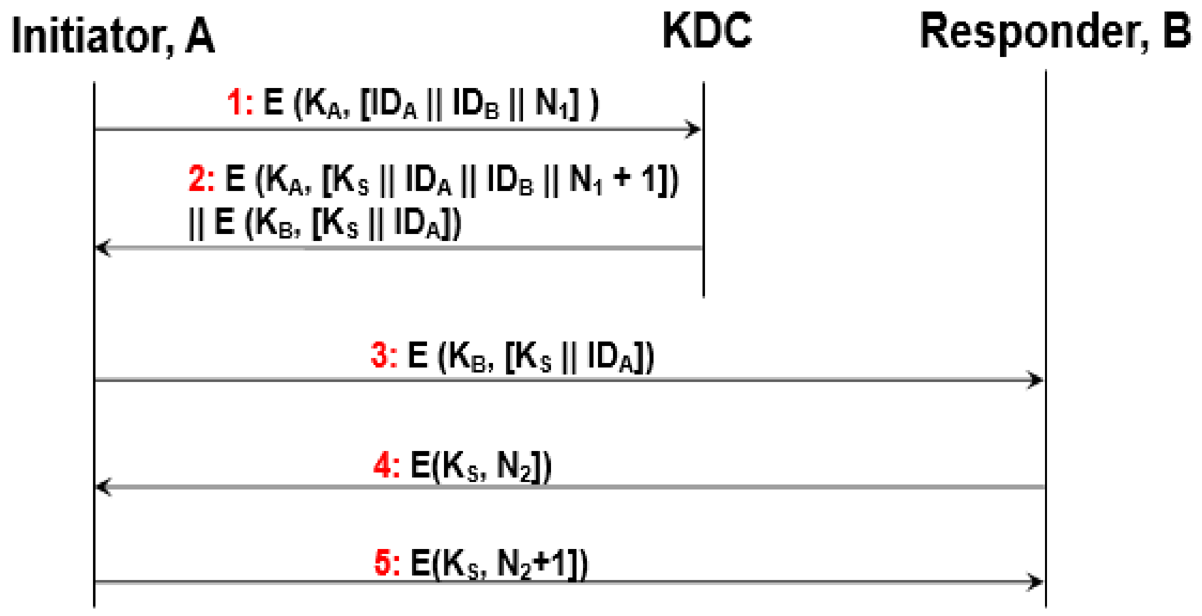
- We assume a Key Distribution Center (KDC) shares a unique key with each party/ user.


Initiator, A
KDC
Responder, B
1: E $(K_A, [ID_A \| ID_B \| N_1])$
2: E $(K_A, [K_S \| ID_A \| ID_B \| N_1 + 1])$ $\| $ E $(K_B, [K_S \| ID_A])$
3: E $(K_B, [K_S \| ID_A])$
4: E$(K_S, N_2])$
5: E$(K_S, N_2+1])$

Note: Steps 1, 2 and 3 are related to "Key Distribution," while steps 3, 4 and 5 are related to providing "Authentication" for the initiator A at the responder B

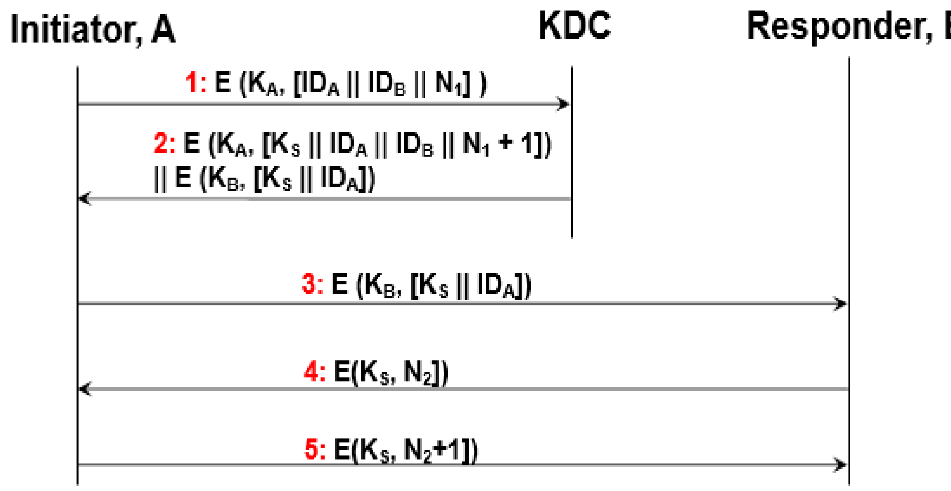
Needham-Schroeder Protocol for Secure Key Distribution and Authentication

- A issues a request to the KDC for a session key to protect a logical connection to B. The message includes the identity of A and B and a unique identifier, N1, for this transaction, which we refer to as a nonce. The nonce may be **a timestamp, a counter, or a random number;** the minimum requirement is that it differs with each request. Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.



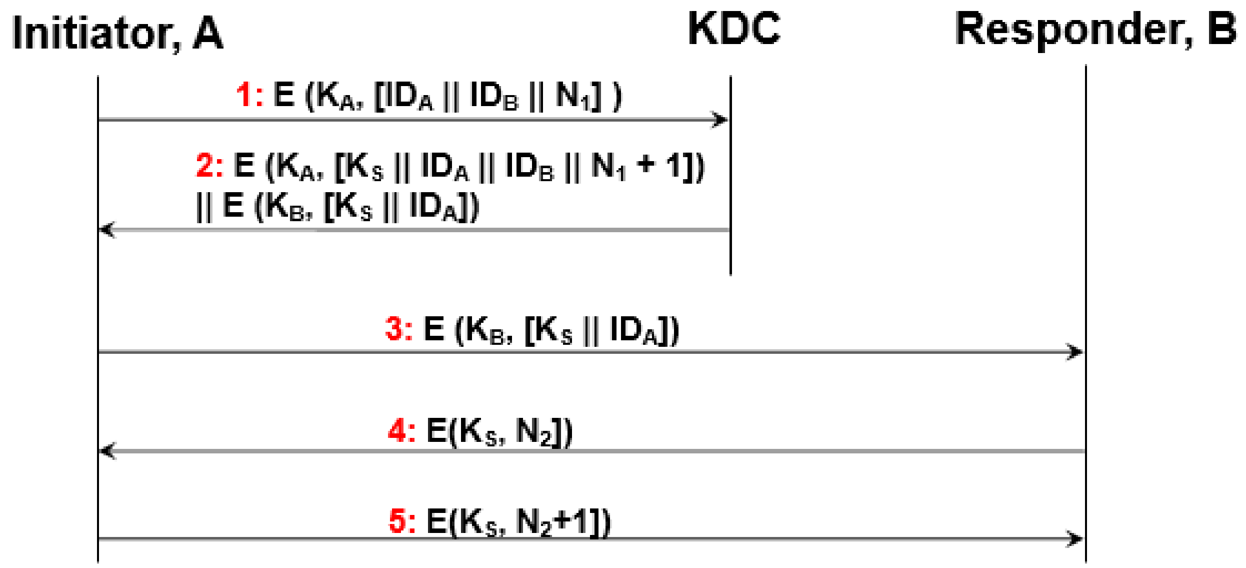
Initiator, A
KDC
Responder, B
1: $E(K_A, [ID_A \| ID_B \| N_1])$
2: $E(K_A, [K_S \| ID_A \| ID_B \| N_1 + 1]) \| E(K_B, [K_S \| ID_A])$
3: $E(K_B, [K_S \| ID_A])$
4: $E(K_S, N_2])$
5: $E(K_S, N_2+1])$

- The KDC responds with a message encrypted using Ka. Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC. The message includes two items intended for A:
  - The one-time session key, Ks, to be used for the session
  - The original request message, including the nonce, to enable A to match this response with the appropriate request
- Thus, A can verify that its original request was not altered before reception by the KDC and, because of the nonce, that this is not a replay of some previous request. In addition, the message includes two items intended for B:
  - The one-time session key, Ks, to be used for the session
  - An identifier of A (e.g., its network address), IDA These last two items are encrypted with Kb (the master key that the KDC shares with B). They are to be sent to B to establish the connection and prove A's identity



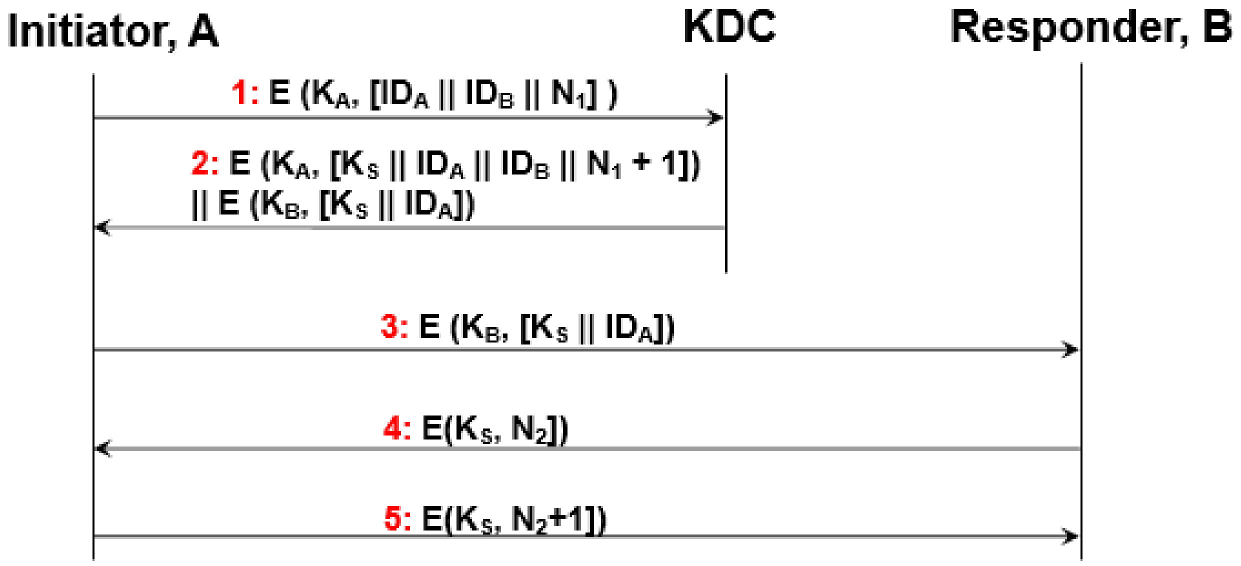
Initiator, A
KDC
Responder,
1: $E(K_A, [ID_A || ID_B || N_1])$
2: $E(K_A, [K_S || ID_A || ID_B || N_1 + 1])$ $|| E(K_B, [K_S || ID_A])$
3: $E(K_B, [K_S || ID_A])$
4: $E(K_S, N_2])$
5: $E(K_S, N_2+1])$

- A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC for B, namely, E(Kb, [Ks || IDA]). Because this information is encrypted with Kb, it is protected from eavesdropping. B now knows the session key (Ks), knows that the other party is A (from IDA), and knows that the information originated at the KDC (because it is encrypted using Kb).



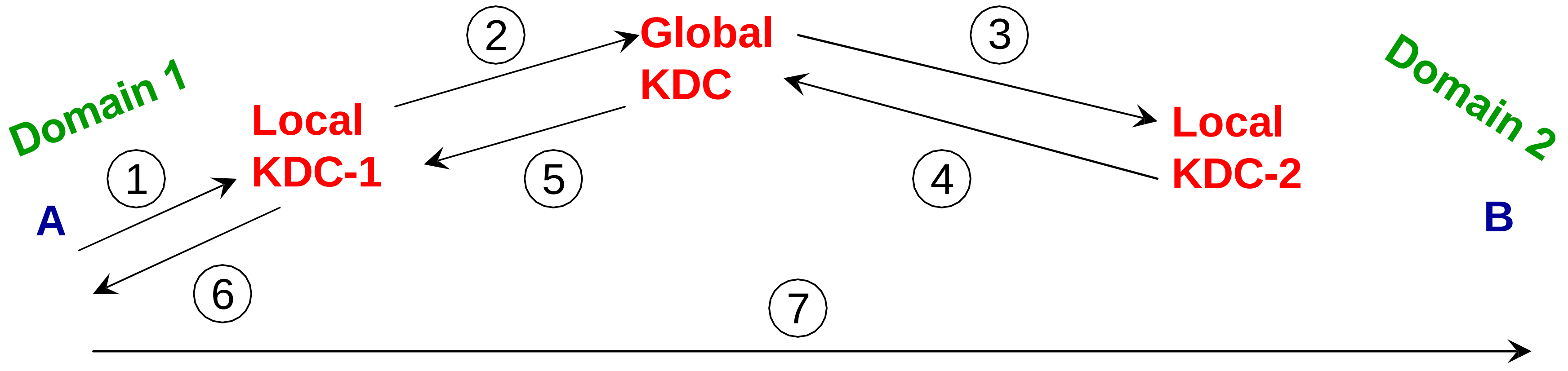
Initiator, A
KDC
Responder, B
1: $E(K_A, [ID_A || ID_B || N_1])$
2: $E(K_A, [K_S || ID_A || ID_B || N_1 + 1])$ $|| E(K_B, [K_S || ID_A])$
3: $E(K_B, [K_S || ID_A])$
4: $E(K_S, N_2])$
5: $E(K_S, N_2+1])$

- Using the newly minted session key for encryption, B sends a nonce, N2, to A. 5.
- Also, using Ks, A responds with f(N2), where f is a function that performs some transformation on N2 (e.g., adding one).
    - the actual key distribution involves only steps 1 through 3, but that steps 4 and 5, as well as step 3, perform an authentication function


Initiator, A
KDC
Responder, B
1: E (K_A, [ID_A || ID_B || N_1])
2: E (K_A, [K_S || ID_A || ID_B || N_1 + 1]) || E (K_B, [K_S || ID_A])
3: E (K_B, [K_S || ID_A])
4: E(K_S, N_2])
5: E(K_S, N_2+1])

Needham-Schroeder Protocol across Domains

- For very large networks, a hierarchy of KDCs can be established.
- For communication among entities within the same local domain, the local KDC is responsible for key distribution. If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a (hierarchy of) global KDC(s)
- The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized (not easy though).



Diffie-Hellman Key Exhange

Primitive Root

- Let *n* be a prime integer.
- Then, *g* is a primitive root for *n* if the first n-1 powers of g: g^1, g^2, g^3, …, g^{n-1} include all of the distinct n-1 integers (except 0) in the class modulo n.
- For example: 3 is a primitive root of 7.

$$\begin{array}{lclclclclcl}
3^1 & = & 3 & = & 3^0 \times 3 & \equiv & 1 \times 3 & = & 3 & \equiv & 3 \pmod{7} \\
3^2 & = & 9 & = & 3^1 \times 3 & \equiv & 3 \times 3 & = & 9 & \equiv & 2 \pmod{7} \\
3^3 & = & 27 & = & 3^2 \times 3 & \equiv & 2 \times 3 & = & 6 & \equiv & 6 \pmod{7} \\
3^4 & = & 81 & = & 3^3 \times 3 & \equiv & 6 \times 3 & = & 18 & \equiv & 4 \pmod{7} \\
3^5 & = & 243 & = & 3^4 \times 3 & \equiv & 4 \times 3 & = & 12 & \equiv & 5 \pmod{7} \\
3^6 & = & 729 & = & 3^5 \times 3 & \equiv & 5 \times 3 & = & 15 & \equiv & 1 \pmod{7}
\end{array}$$

Note: A prime integer could have one or more primitive roots.

How to test for Primitive Root?

- Let n be a prime integer.
- We need to find the prime factors of n-1.
  - Note that any integer can be written as the product of prime integers and/or their exponents
  - For example: $760 = 2^3 * 5 * 19$
- To check whether g is a primitive root of n, do the following:
  - Find the prime factors of n-1
  - For every prime factor q,
    - **Check if $g^{(n-1)/q} \bmod n \neq 1$**
  - If the above test satisfies for all prime factors of n-1, then g is a primitive root of n; otherwise, not
- Repeat the above test for all possible values of g: 2, 3, 4, …, n-1.

Example for Primitive Root (1)

- Let n = 13
- n-1 = 12 = $2^2 * 3$
- Prime factors q_s = {2, 3}
- One can try for all possible values of g = 2, 3, ...,12 with the test for primitive root.
- Try g = 2

q = 2: g^(n-1)/q mod n = 2^(12/2) mod 13 = 12 ≠ 1

q = 3: g^(n-1)/q mod n = 2^(12/3) mod 13 = 3 ≠ 1.

Hence, g = 2 is a primitive root of 13.

- Try g = 5

q = 2: g^(n-1)/q mod n = 5^(12/2) mod 13 = 12 ≠ 1

q = 3: g^(n-1)/q mod n = 5^(12/3) mod 13 = 1.

Hence, g = 5 is NOT a primitive root of 13.

Example for Primitive Root (2)

- Let n = 23
- n-1 = 22 = 2 * 11
- Prime factors q_s = {2, 11}
- One can try for all possible values of g = 2, 3, ..., 22 with the test for prime factors.
- Try g = 3

q = 2: g^(n-1)/q mod n = 3^(22/2) mod 23 = 1

Hence, g = 3 is NOT a primitive root of 23.

- Try g = 4

q = 2: g^(n-1)/q mod n = 4^(22/2) mod 23 = 1

Hence, g = 4 is NOT a primitive root of 13.

- Try g = 5

q = 2: g^(n-1)/q mod n = 5^(22/2) mod 23 = 22 $\neq$ 1

q = 11: g^(n-1)/q mod n = 5^(22/11) mod 23 = 2 $\neq$ 1

Hence, g = 5 is a primitive root of 23.

Diffie-Hellman Key Exchange

- Used to establish a secret key over an insecure communication channel
- The field size *n* (a prime integer) and a starting number *g* (g < *n* and is a primitive root of *n*) are publicly known.
- Alice locally generates an integer 'a' (where a < n) and sends (g^a) mod n as an intermediate key to Bob.
- Bob locally generates an integer 'b' (where b < n) and sends (g^b) mod n as an intermediate key to Alice.
- Bob receives (g^a) mod n from Alice and computes $\{(g^a) \bmod n\}^b \bmod n$
- Similarly, Alice receives (g^b) mod n from Bob and computes $\{(g^b) \bmod n\}^a \bmod n$
- Due to the commutative property:

$$\{(g^a) \bmod n\}^b = \{(g^{a*b}) \bmod n\} = \{(g^b) \bmod n\}^a$$

  - Hence, Alice and Bob agree on the same secret key.

Diffie-Hellman Key Exchange

Alice

Local key: ‘a’

Intermediate key:

$I_{AB} = g^a \bmod n$

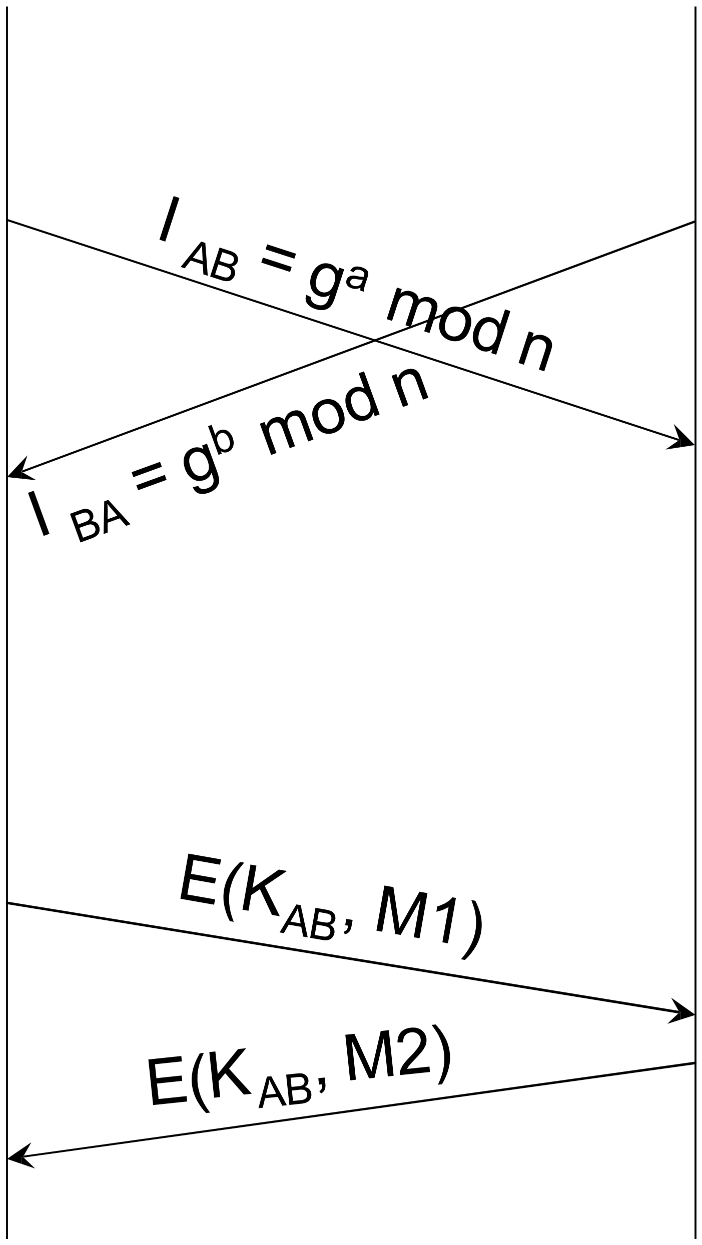
Alice computes

$I_{BA}{}^a \bmod n$

$= (g^b \bmod n)^a \bmod n$

$= (g^{b*a} \bmod n)$

$=$ Secret Key, K_{AB}



Bob

Local key: ‘b’

Intermediate key:

$I_{BA} = g^b \bmod n$

Bob computes

$I_{AB}{}^b \bmod n$

$= (g^a \bmod n)^b \bmod n$

$= (g^{a*b} \bmod n)$

$=$ Secret Key, K_{AB}

Diffie-Hellman Key Exch. Example 1

- Let $n = 17$.
- Find a suitable value for the parameter ‘g’ (a primitive root of n).
- Generate two integers a and b ($a < n$ and $b < n$) for Alice and Bob as their private keys.
- Compute the intermediate keys sent from Alice to Bob and from Bob to Alice
- Compute the final secret key agreed upon by both Alice and Bob.

D-H Ex-1: To find a primitive root for n = 17

- $n = 17$; $n-1 = 16 = 2^4$.
- Prime factors of n-1: qs = {2}.
- Lets try g = 3

  $g^{(n-1)/q} \bmod n = 3^{(16/2)} \bmod 17 = 6561 \bmod 17 = 16 \neq 1$
- Hence, g = 3 is a primitive root of n = 17.
- Let Alice choose ‘a’ = 5 and Bob choose ‘b’ = 7.
- Intermediate key computed by Alice and sent to Bob: $g^a \bmod n = 3^5 \bmod 17$
- Intermediate key computed by Bob and sent to Alice: $g^b \bmod n = 3^7 \bmod 17$.
- Secret key greed upon by both Alice and Bob

  $g^{a*b} \bmod n = 3^{5*7} \bmod 17 = 3^{35} \bmod 17$.

D-H Ex-1: Interm. Key Computed by Alice

- $g^a \bmod n = 3^5 \bmod 17$

4	2	1
1	0	1

Exponent 5

$3^1 \bmod 17 = 3$

$3^2 \bmod 17 = (3^1 * 3^1) \bmod 17 = 9$

$3^4 \bmod 17 = (3^2 * 3^2) \bmod 17 = (9 * 9) \bmod 17 = 13$

$3^5 \bmod 17 = (3^4 * 3^1) \bmod 17 = (13 * 3) \bmod 17 =$ **5** **(Sent to Bob)**

D-H Ex-1: Interm. Key Computed by Bob

- $g^b \bmod n = 3^7 \bmod 17$

4	2	1
1	1	1

Exponent 7

$3^1 \bmod 17 = 3$

$3^2 \bmod 17 = (3^1 * 3^1) \bmod 17 = 9$

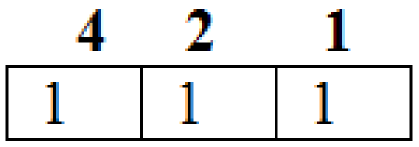
$3^4 \bmod 17 = (3^2 * 3^2) \bmod 17 = (9 * 9) \bmod 17 = 13$

$3^7 \bmod 17 = (3^4 * 3^2 * 3^1) \bmod 17 = (13 * 9 * 3) \bmod 17 =$ **11**

(Sent to Alice)

D-H Ex-1: Secret Key Computed by Bob

- $(g^a \bmod n)^b \bmod n = 5^7 \bmod 17$



4	2	1
1	1	1

Exponent 7

$5^1 \bmod 17 = 5$

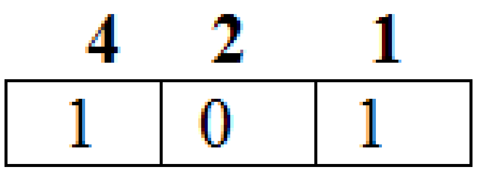
$5^2 \bmod 17 = (5^1 * 5^1) \bmod 17 = 8$

$5^4 \bmod 17 = (5^2 * 5^2) \bmod 17 = (8 * 8) \bmod 17 = 13$

$5^7 \bmod 17 = (5^4 * 5^2 * 5^1) \bmod 17 = (13 * 8 * 5) \bmod 17 =$ **10**

D-H Ex-1: Secret Key Computed by Alice

- $(g^b \bmod n)^a \bmod n = 11^5 \bmod 17$



4	2	1
1	0	1

Exponent 5

$11^1 \bmod 17 = 11$

$11^2 \bmod 17 = (11^1 * 11^1) \bmod 17 = 2$

$11^4 \bmod 17 = (11^2 * 11^2) \bmod 17 = (2 * 2) \bmod 17 = 4$

$11^5 \bmod 17 = (11^4 * 11^1) \bmod 17 = (4 * 11) \bmod 17 =$ **10**

D-H Ex-1: Secret Key Computed by both Alice and Bob (Global view)

- $g^{a*b} \bmod n = 3^{35} \bmod 17$

32	16	8	4	2	1
1	0	0	0	1	1

Exponent 35

$3^1 \bmod 17 = 3$

$3^2 \bmod 17 = (3^1 * 3^1) \bmod 17 = 9$

$3^4 \bmod 17 = (3^2 * 3^2) \bmod 17 = (9 * 9) \bmod 17 = 13$

$3^8 \bmod 17 = (3^4 * 3^4) \bmod 17 = (13*13) \bmod 17 = 16$

$3^{16} \bmod 17 = (3^8 * 3^8) \bmod 17 = (16*16) \bmod 17 = 1$

$3^{32} \bmod 17 = (3^{16} * 3^{16}) \bmod 17 = (1*1) \bmod 17 = 1$

$3^{35} \bmod 17 = (3^{32} * 3^2 * 3^1) \bmod 17 = (1 * 9 * 3) \bmod 17 = \mathbf{10}$

Diffie-Hellman Key Exch. Example 2

- Let n = 19.
- Find a suitable value for the parameter ‘g’ (a primitive root of n) such that g > 4.
- Let Alice choose a = 6 and Bob choose b = 9.
- Compute the intermediate keys sent from Alice to Bob and from Bob to Alice
- Compute the final secret key agreed upon by both Alice and Bob.

D-H Ex-2: To find a primitive root for n = 19

- n = 19; n-1 = 18 = 2 * 3^2.
- Prime factors of n-1: qs = {2, 3}.
- Lets try g = 5

  q = 2: $g^{(n-1)/q}$ mod n = $5^{(18/2)}$ mod 19 = 1; Hence, 5 is not a possible value for g.
- Lets try g = 6

  q = 2: $g^{(n-1)/q}$ mod n = $6^{(18/2)}$ mod 19 = 1; Hence, 6 is not a possible value for g.
- Lets try g = 7

  q= 2: $g^{(n-1)/q}$ mod n = $7^{(18/2)}$ mod 19 = 1; Hence, 7 is not a possible value for g.
- Lets try g = 8

  q = 2: $g^{(n-1)/q}$ mod n = $8^{(18/2)}$ mod 19 = 18

  q = 3: $g^{(n-1)/q}$ mod n = $8^{(18/3)}$ mod 19 = 1. Hence, 8 is not a possible value for g.
- Lets try g = 9

  q = 2: $g^{(n-1)/q}$ mod n = $9^{(18/2)}$ mod 19 = 1. Hence, 9 is not a possible value for g.
- Lets try g = 10

  q = 2: $g^{(n-1)/q}$ mod n = $10^{(18/2)}$ mod 19 = 18.

  q = 3: $g^{(n-1)/q}$ mod n = $10^{(18/3)}$ mod 19 = 11. Hence, **g = 10 is a possible value!**

D-H Ex-2: Intermediate Keys & Secret Key

- $n = 19$; $g = 10$.
- Let Alice choose ‘a’ = 6 and Bob choose ‘b’ = 9.
- Intermediate key computed by Alice and sent to Bob: $g^a \bmod n = 10^6 \bmod 19$
- Intermediate key computed by Bob and sent to Alice: $g^b \bmod n = 10^9 \bmod 19$.
- Secret key greed upon by both Alice and Bob (global view)

  $g^{a*b} \bmod n = 10^{6*9} \bmod 19 = 10^{54} \bmod 19$.

D-H Ex-2: Secret Key Computed by Alice

- g^a mod n = 10^6 mod 19

4	2	1
1	1	0

Exponent 6

10^1 mod 19 = 10

10^2 mod 19 = (10^1 * 10^1) mod 19 = (10 * 10) mod 19 = 5

10^4 mod 19 = (10^2 * 10^2) mod 19 = (5 * 5) mod 19 = 6

10^6 mod 19 = (10^4 * 10^2) mod 19 = (5 * 6) mod 19 = 11 (sent to Bob)

D-H Ex-2: Interm. Key Computed by Bob

- g^b mod n = 10^9 mod 19

8	4	2	1
1	0	0	1

Exponent 9

10^8 mod 19 = (10^4 * 10^4) mod 19 = (6 * 6) mod 19 = 17

10^9 mod 19 = (10^8 * 10^1) mod 19 = (17 * 10) mod 19 = 18

(sent to Alice)

D-H Ex-2: Interm. Key Computed by Bob

- $(g^a \bmod n)^b \bmod n = 11^9 \bmod 19$

$11^1 \bmod 19 = 11$

$11^2 \bmod 19 = (11^1 * 11^1) \bmod 19 = (11 * 11) \bmod 19 = 7$

$11^4 \bmod 19 = (11^2 * 11^2) \bmod 19 = (7 * 7) \bmod 19 = 11$

$11^8 \bmod 19 = (11^4 * 11^4) \bmod 19 = (11 * 11) \bmod 19 = 7$

$11^9 \bmod 19 = (11^8 * 11^1) \bmod 19 = (7 * 11) \bmod 19 = 1$

D-H Ex-2: Interm. Key Computed by Alice

- $(g^b \bmod n)^a \bmod n = 18^6 \bmod 19$

$18^1 \bmod 19 = 18$

$18^2 \bmod 19 = (18^1 * 18^1) \bmod 19 = (18 * 18) \bmod 19 = 1$

$18^4 \bmod 19 = (18^2 * 18^2) \bmod 19 = (1 * 1) \bmod 19 = 1$

$18^6 \bmod 19 = (18^4 * 18^2) \bmod 19 = (1 * 1) \bmod 19 = 1$

D-H Ex-2: Secret Key Computed by both Alice and Bob

- g^{a*b} mod n = 10^{54} mod 19

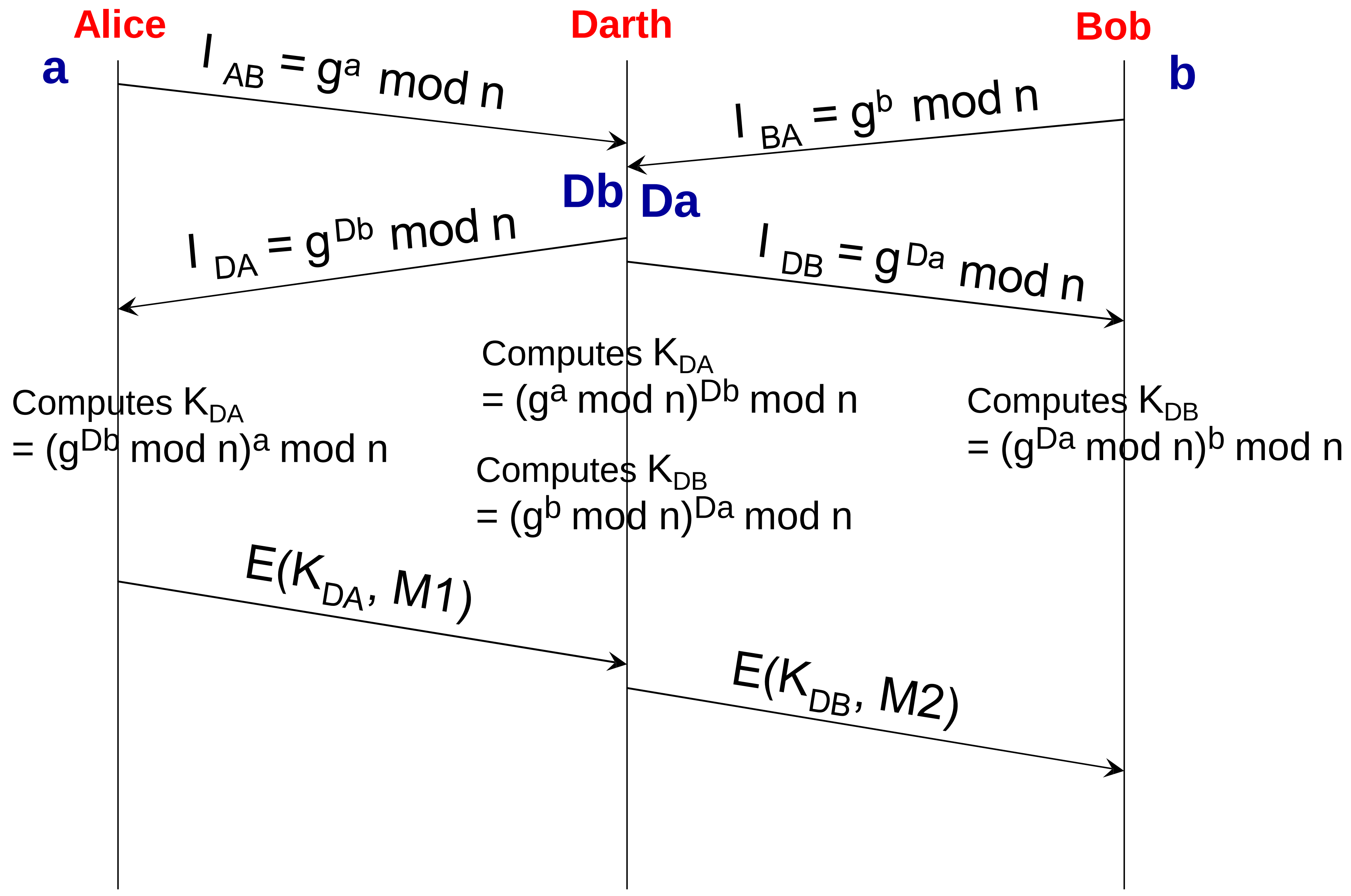
32	16	8	4	2	1
1	1	0	1	1	0

Exponent 54

10^1 mod 19 = 10
10^2 mod 19 = (10^1 * 10^1) mod 19 = (10 * 10) mod 19 = 5
10^4 mod 19 = (10^2 * 10^2) mod 19 = (5 * 5) mod 19 = 6
10^8 mod 19 = (10^4 * 10^4) mod 19 = (6 * 6) mod 19 = 17
10^{16} mod 19 = (10^8 * 10^8) mod 19 = (17 * 17) mod 19 = 4
10^{32} mod 19 = (10^{16} * 10^{16}) mod 19 = (4 * 4) mod 19 = 16

10^{54} mod 19
= (10^{32} * 10^{16} * 10^4 * 10^2) mod 19 = (16 * 4 * 6 * 5) mod 19
= (64 * 30) mod 19 = (7 * 11) mod 19 = **1**

Man-in-the-Middle Attack (D-H Key Exch.)



Alice
Darth
Bob
a
b
$I_{AB} = g^a \bmod n$
$I_{BA} = g^b \bmod n$
Db
Da
$I_{DA} = g^{Db} \bmod n$
$I_{DB} = g^{Da} \bmod n$
Computes K_{DA} $= (g^{Db} \bmod n)^a \bmod n$
Computes K_{DA} $= (g^a \bmod n)^{Db} \bmod n$
Computes K_{DB} $= (g^b \bmod n)^{Da} \bmod n$
Computes K_{DB} $= (g^{Da} \bmod n)^b \bmod n$
$E(K_{DA}, M1)$
$E(K_{DB}, M2)$

Man-in-the-Middle Attack (D-H Key Exchg.)

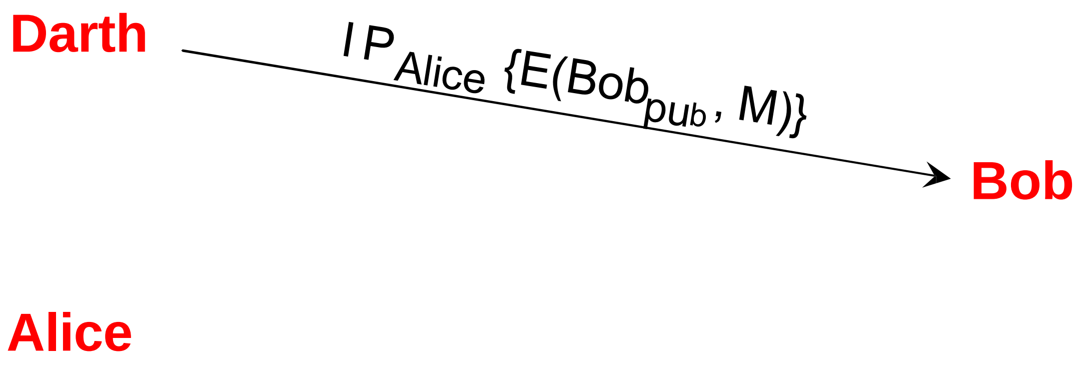
- The vulnerability lies in Alice accepting the incoming message has come from Bob (without actually verifying or validating it) and vice-versa (i.e., Bob accepting the incoming message to have come from Alice without validating it).

- The attack basically involves Darth taking the role of Bob as far as Alice is concerned (generating a local key Db that is used to set up the secret key K_{DA} with Alice) and similarly taking the role of Alice as far as Bob is concerned (generating a local key Da that is used to set up the secret key K_{DB} with Bob).

- Alice and Bob use the respective secret keys K_{DA} and K_{DB} to communicate messages thinking they are communicating directly with each other; whereas, what happens is Darth intercepting all the communication and changing them if needed before forwarding to the other end.

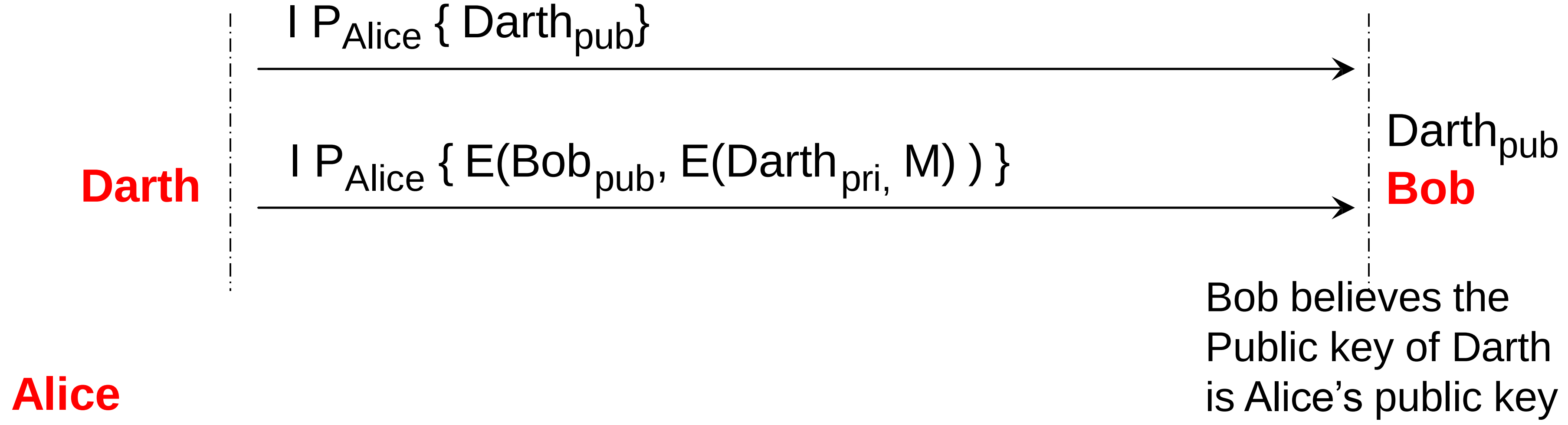
Man-in-the-Middle Attack (1) (Public Key Encryption)

- Using a technique called “IP Spoofing”, a malicious user (say Darth) could send a message to Bob (encrypted with the public key of Bob) and make the message to appear to come from Alice’s IP address.
- Bob will be the only one who can decrypt the message (using his private key). If Bob trusts Alice, Bob could process whatever is in the message as Bob thinks the message came from Alice.



Man-in-the-Middle Attack (2) (Public Key Encryption)

- More diligently, Darth could send his own public key to Bob and using IP spoofing make the public key to have come from Alice, making Bob to believe what he receives is Alice's public key.
- Darth could then send Bob a message encrypted with his own private key and using IP spoofing make the message to have come from Alice.
- Bob decrypts the message using the public key that he perceives to be of Alice's (actually, it is Darth's public key) as the message appears to have come from Alice.
- So, the issue is how to make the receiver verify that the message indeed came from the sender who appears to have sent it.



$IP_{Alice}\{Darth_{pub}\}$
$IP_{Alice}\{E(Bob_{pub}, E(Darth_{pri}, M))\}$
Darth
$Darth_{pub}$
Bob
Alice
Bob believes the Public key of Darth is Alice's public key

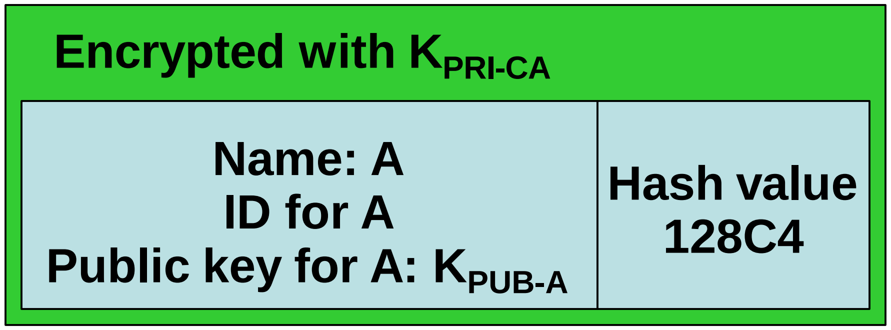
Solution: Public Key Certificate

- The public key certificate of a user is the public key of the user along with the user's identifying information as well as the hash value of all the above – encrypted with the private key of a certifying authority (CA) that is being trusted by all the users in the system.
  - The CA is like a digital notary.

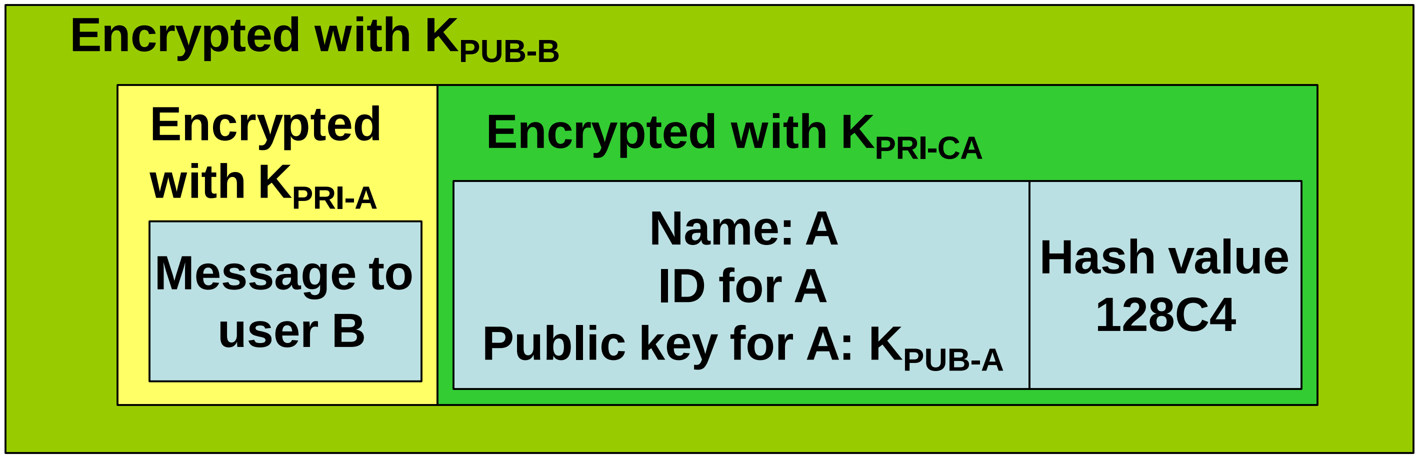
- The sender could encrypt the message with its private key and attach its public key certificate along with the encrypted message – both of which are further encrypted with the receiver's private key.
  - As a result, only the receiver (who also trusts the CA) could decrypt what is received from the sender and validate that the message indeed came from the sender by extracting the public key of the

Public Key Certificate

Digital Certificate for the Public Key of A


Encrypted with $K_{PRI\text{-}CA}$
Name: A
ID for A
Public key for A: $K_{PUB\text{-}A}$
Hash value
128C4

User A sending to user B


Encrypted with $K_{PUB\text{-}B}$
Encrypted with $K_{PRI\text{-}A}$
Message to user B
Encrypted with $K_{PRI\text{-}CA}$
Name: A
ID for A
Public key for A: $K_{PUB\text{-}A}$
Hash value
128C4

Note: The certificates are created and formatted based on the X.509 standard, which outlines the necessary fields of a certificate and the possible values that can be inserted into these fields. The latest X.509 version is v.3.

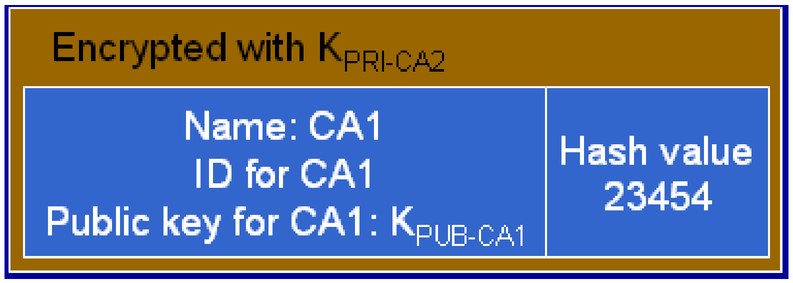
Certificates

- What if user B is in another network and cannot directly accept the attestation done by the CA of A, and needs another CA to attest the public key of the CA of A?

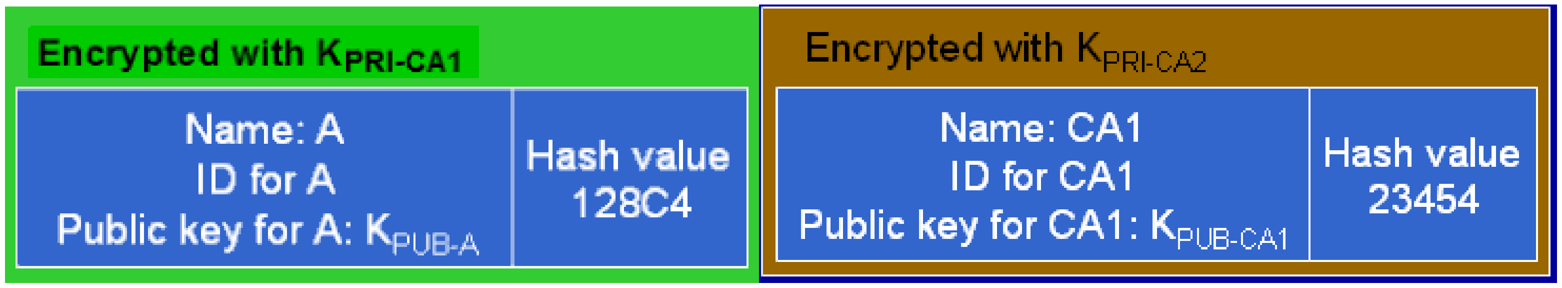
- Let CA1 be the CA that could attest A.
- Let CA2 be the CA that needs to attest CA1 and this attestation would be believed by B.
- Along with the digital certificate issued by CA1 for A, CA1 needs to append the digital certificate it received from CA2 for the public key of CA1
- User A need to send both these digital certificates to B.
- User B will first extract the public key of CA1 from the digital certificate issued by CA2, using the public key of CA2.
- User B will then extract the public key of user A from the digital certificate issued by CA1, using the extracted public key of CA1.
- User B will then use this certified public key of user A to extract the message.

Certificates

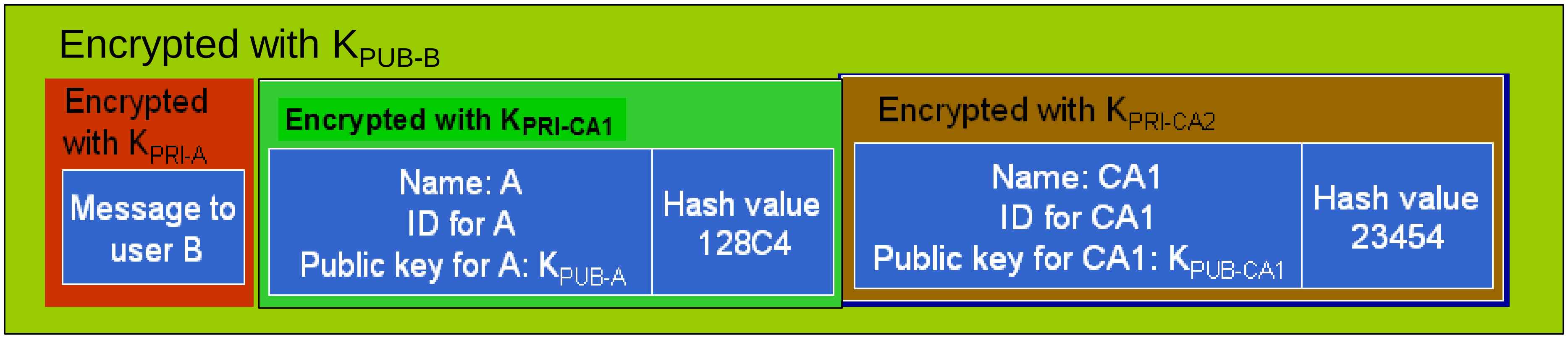
Digital Certificate for the Public Key of CA1


Encrypted with $K_{PRI\text{-}CA2}$
Name: CA1
ID for CA1
Public key for CA1: $K_{PUB\text{-}CA1}$
Hash value 23454

Digital Certificate for the Public Key of A

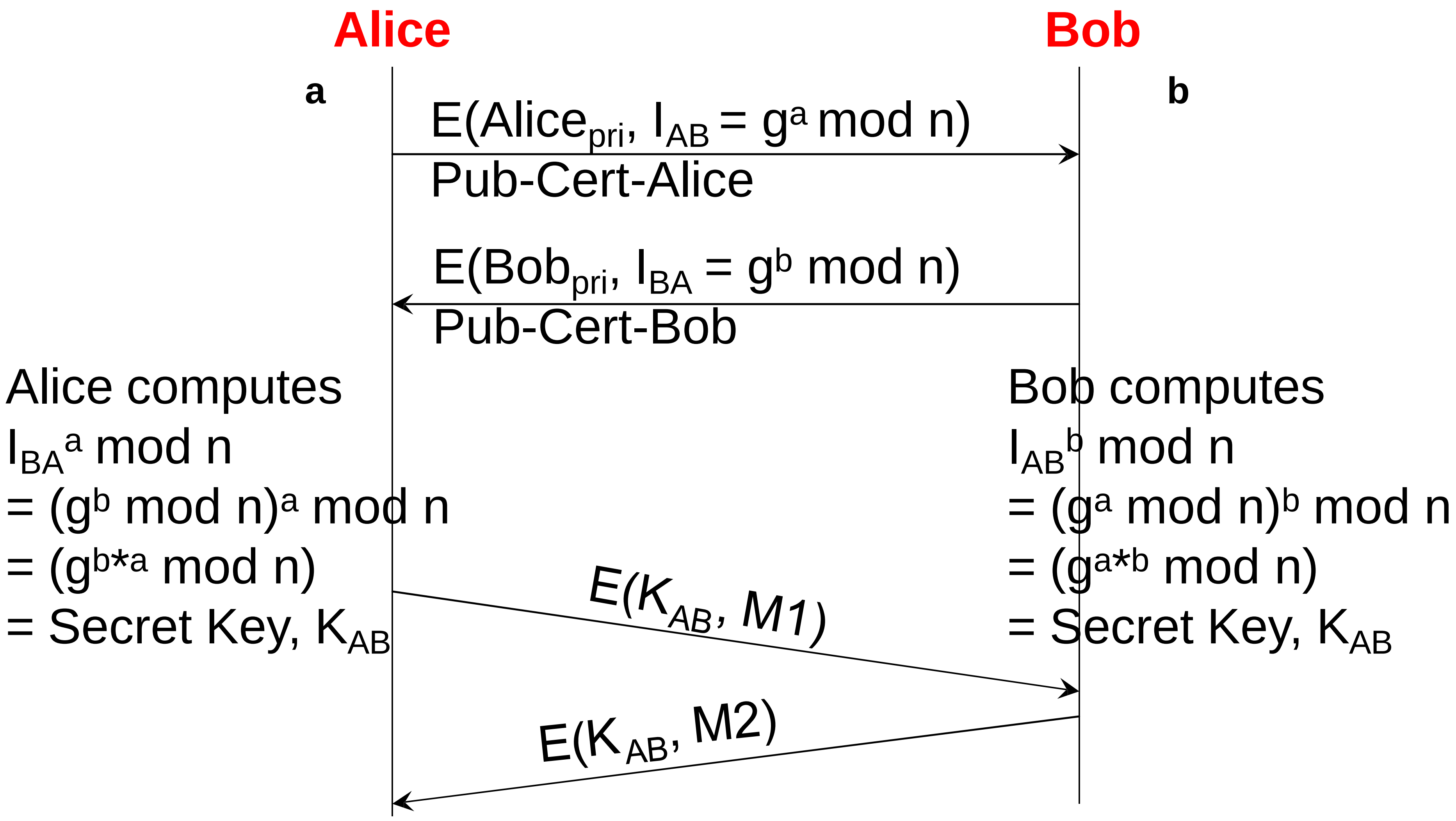

Encrypted with $K_{PRI\text{-}CA1}$
Name: A
ID for A
Public key for A: $K_{PUB\text{-}A}$
Hash value 128C4
Encrypted with $K_{PRI\text{-}CA2}$
Name: CA1
ID for CA1
Public key for CA1: $K_{PUB\text{-}CA1}$
Hash value 23454

User A sending to user B


Encrypted with $K_{PUB\text{-}B}$
Encrypted with $K_{PRI\text{-}A}$
Message to user B
Encrypted with $K_{PRI\text{-}CA1}$
Name: A
ID for A
Public key for A: $K_{PUB\text{-}A}$
Hash value 128C4
Encrypted with $K_{PRI\text{-}CA2}$
Name: CA1
ID for CA1
Public key for CA1: $K_{PUB\text{-}CA1}$
Hash value 23454

Station-to-Station Protocol

- Authenticated version of Diffie-Hellman Key Exchange


Alice
Bob
a
b
$E(Alice_{pri}, I_{AB} = g^a \bmod n)$
Pub-Cert-Alice
$E(Bob_{pri}, I_{BA} = g^b \bmod n)$
Pub-Cert-Bob
$E(K_{AB}, M1)$
$E(K_{AB}, M2)$

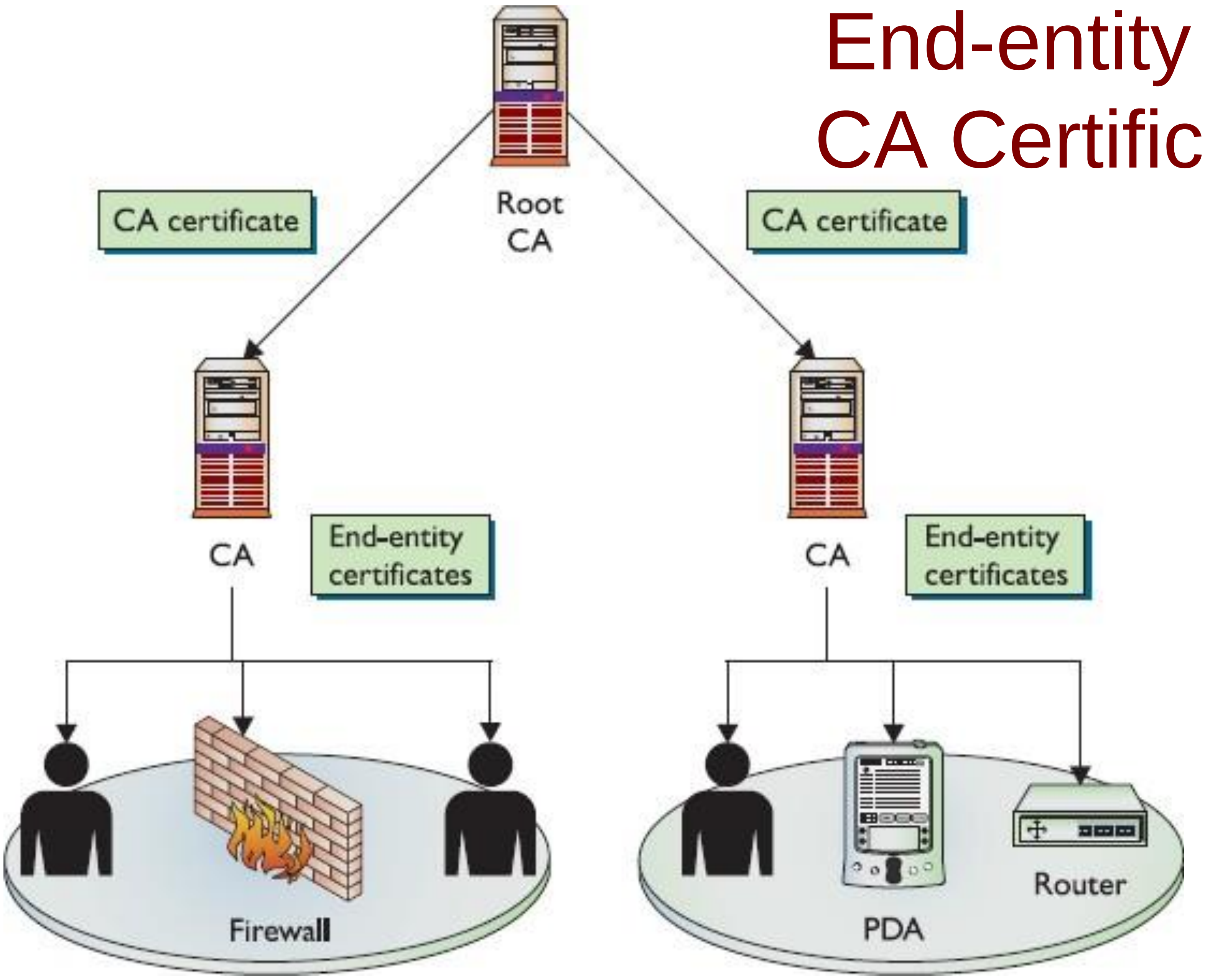
Alice computes
$I_{BA}{}^a \bmod n$
$= (g^b \bmod n)^a \bmod n$
$= (g^{b*a} \bmod n)$
$=$ Secret Key, K_{AB}

Bob computes
$I_{AB}{}^b \bmod n$
$= (g^a \bmod n)^b \bmod n$
$= (g^{a*b} \bmod n)$
$=$ Secret Key, K_{AB}

Classes of Digital Certificates

- The types of certificates available can vary between CAs; but, all CAs should at least support the following three classes of certificates:
- Class 1 – A Class 1 certificate is usually used to verify an individual's identity through e-mail. A person who receives a Class 1 certificate can use his public/ private key pair to digitally sign e-mail and encrypt message contents.
- Class 2 – A Class 2 certificate can be used for software signing. A software vendor would register for this type of certificate so that it could digitally sign the software. This provides integrity for the software after it is developed and released, and it allows the receiver software to verify where the software actually came from before installation.
- Class 3 – A Class 3 certificate can be used by a company to set up its own CA which will allow it to carry out its own identification verification and internally generate certificates.
- End-entity certificates (Class 1 and 2 are referred to as End-entity certificates)
- CA certificate – Class 3 can also be referred to as CA certificates.
- Note: An entity can have multiple public/ private key pairs and corresponding digital certificates, used for different purposes.

End-entity and CA Certificates



Source: Figure 6.8 from Conklin and White – Principles of Computer Security, 2nd Edition

Distribution of Public Keys

- can be considered as using one of:
  - public announcement
  - publicly available directory
  - public-key authority
  - public-key certificates

Public Announcement

- users distribute public keys to recipients or broadcast to community at large
  - eg. append PGP keys to email messages or post to news groups or email list
- major weakness is forgery
  - anyone can create a key claiming to be someone else and broadcast it
  - until forgery is discovered can masquerade as claimed user

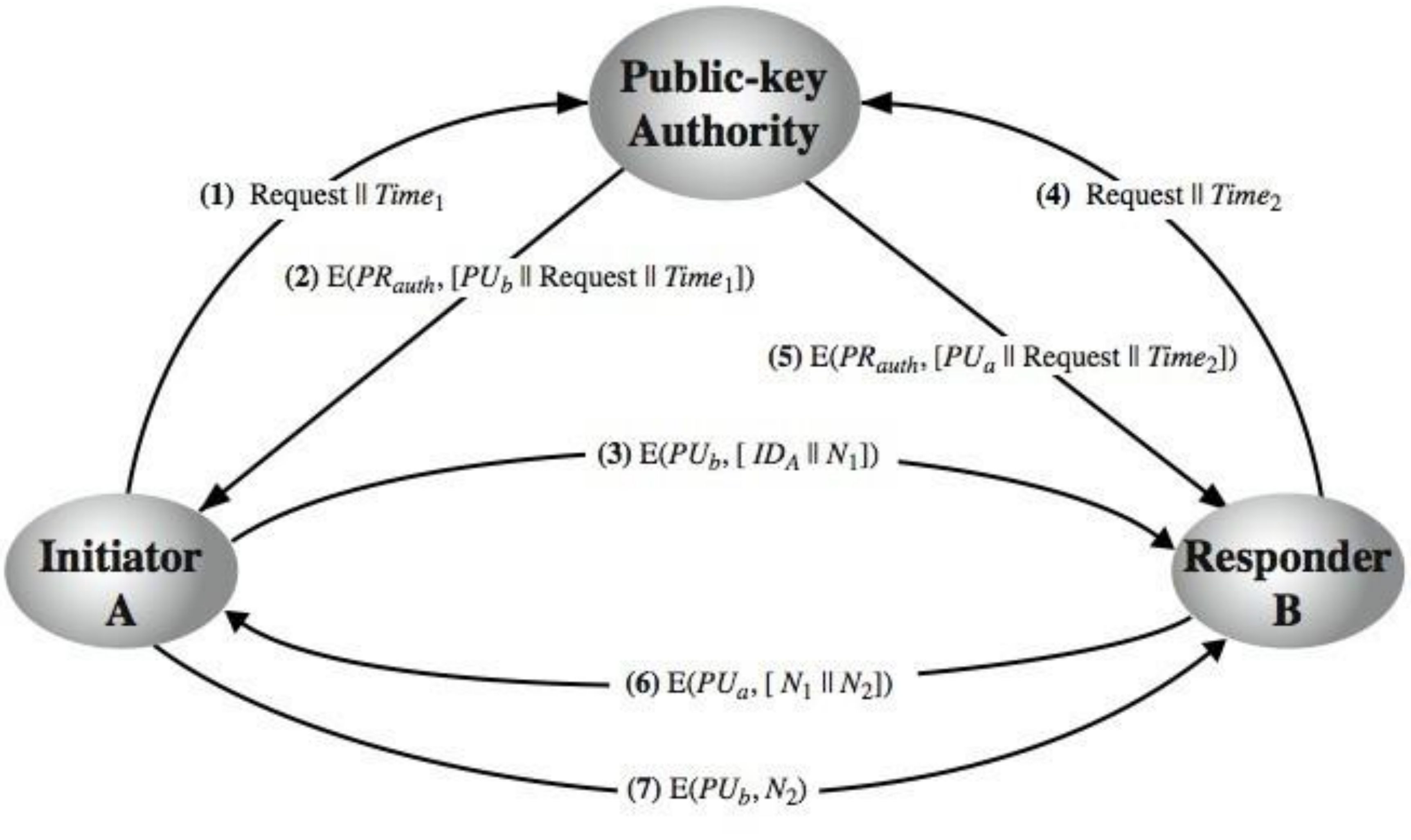
Publicly Available Directory

- can obtain greater security by registering keys with a public directory
- directory must be trusted with properties:
  - contains {name,public-key} entries
  - participants register securely with directory
  - participants can replace key at any time
  - directory is periodically published
  - directory can be accessed electronically
- still vulnerable to tampering or forgery

Public-Key Authority

- improve security by tightening control over distribution of keys from directory
- has properties of directory
- and requires users to know public key for the directory
- then users interact with directory to obtain any desired public key securely
  - does require real-time access to directory when keys are needed
  - may be vulnerable to tampering

Public-Key Authority


Public-key Authority
(1) Request ‖ $Time_1$
(2) E(PR_{auth}, [PU_b ‖ Request ‖ $Time_1$])
(4) Request ‖ $Time_2$
(5) E(PR_{auth}, [PU_a ‖ Request ‖ $Time_2$])
(3) E(PU_b, [ID_A ‖ N_1])
Initiator A
Responder B
(6) E(PU_a, [N_1 ‖ N_2])
(7) E(PU_b, N_2)