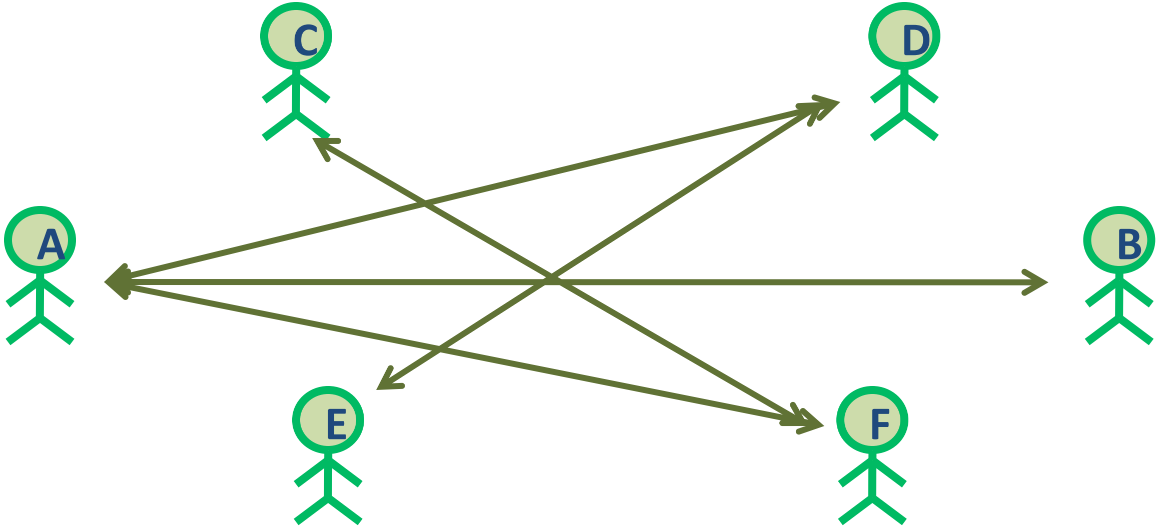
Key Establishment

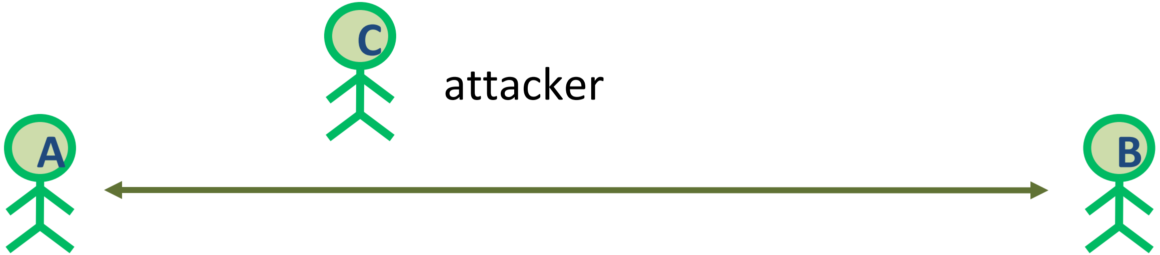
Chester Rebeiro
IIT Madras

Multi Party secure communication



- N parties want to communicate securely with each other (N=6 in this figure)
- If U sends a message to V (U ≠V and U,V Ɛ {a,b,c,d,e,f})
  - Only V should be able to read the message
  - No other parties (even if they cooperate) should be able to read the message

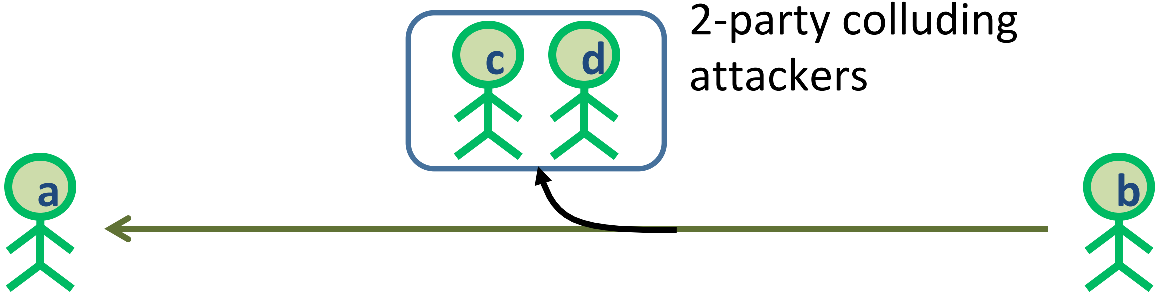
Adversary Assumptions



- **Passive Attacker** (evesdropper)
- **Active Attacker**
  - Aim :
    fool A and B into accepting an invalid key
    (invalid key : expired key, a key chosen by the attacker)
    fool A / B into believing that they have exchanged a key with the other
    get partial information about the key exchanged between A and B

  - Modus-Operandi :
    - alter messages
    - save messages and replay later
    - masquerade

Adversary Assumptions



2-party colluding attackers

- Attackers can collude to get the secrets
- k-party colluding attacks
  - K attackers collude

Types of Keys

- **Long lived keys**
  - Generally used for authentication, setting up session keys
    - Could be either a key corresponding to a symmetric cipher
    - Or a private key corresponding to a public key cipher

- **Session keys**
  - Used for a brief period of time such as a single session.
    - Typically session key corresponds to a symmetric key cipher
  - and requires to be changed periodically
  - Derived from LL keys

Example (the keys in GSM)

- **Long lived (LL) keys**
  - SIM contains a individual subscriber authentication key (k_i)
    - It is never transmitted or the network.
  - A copy of k_i is also stored in databases in the base station
  - k_i is used to authenticate the SIM using an algorithm called A3

- **Session keys (k_c)**
  - Created at the time of a call changed periodically during the call
  - It is created using k_i and an algorithm A8
  - Voice and Signals are encrypted using the session key ki using a cipher A5

Why use Session Keys?

- Limit the amount of ciphertext an attacker sees.
- Limit exposure when device is compromised.
- Limits the amount of long term information that needs to be stored on device.

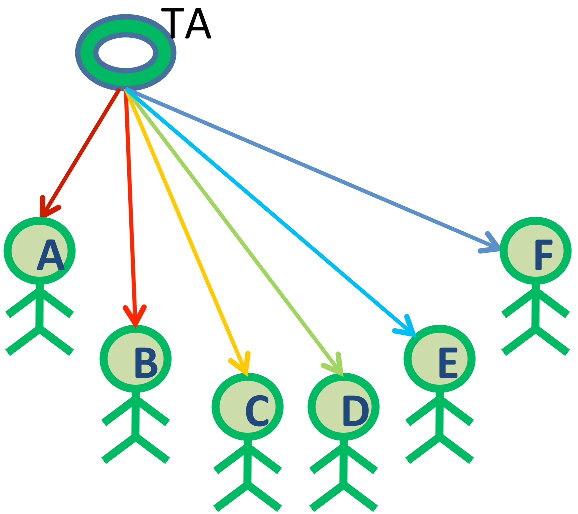
Distributing LL Keys

Non-interactively

- LL keys are stored in the device (such as TPMs)
  - Or computed from stored secrets (such as PUFs)

Interactively

- Could also be sent to the device by a trusted authority (TA)
  - Trusted Authority
    - Verifies identities of users
    - Issues certificates
    - Has a secure link with each user
- Distribution schemes from TA
  - Using public key constructs
    - User’s store private keys
    - User certificates stored by TA contains the public keys
  - Using symmetric key constructs
    - TA has a secure channel to distribute secret keys to pairs of users

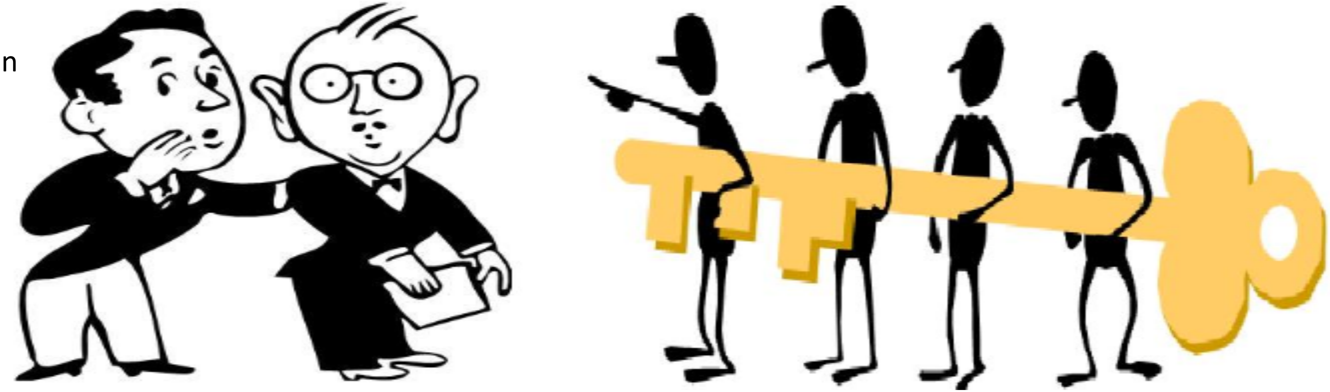


Key Predistribution

Definition

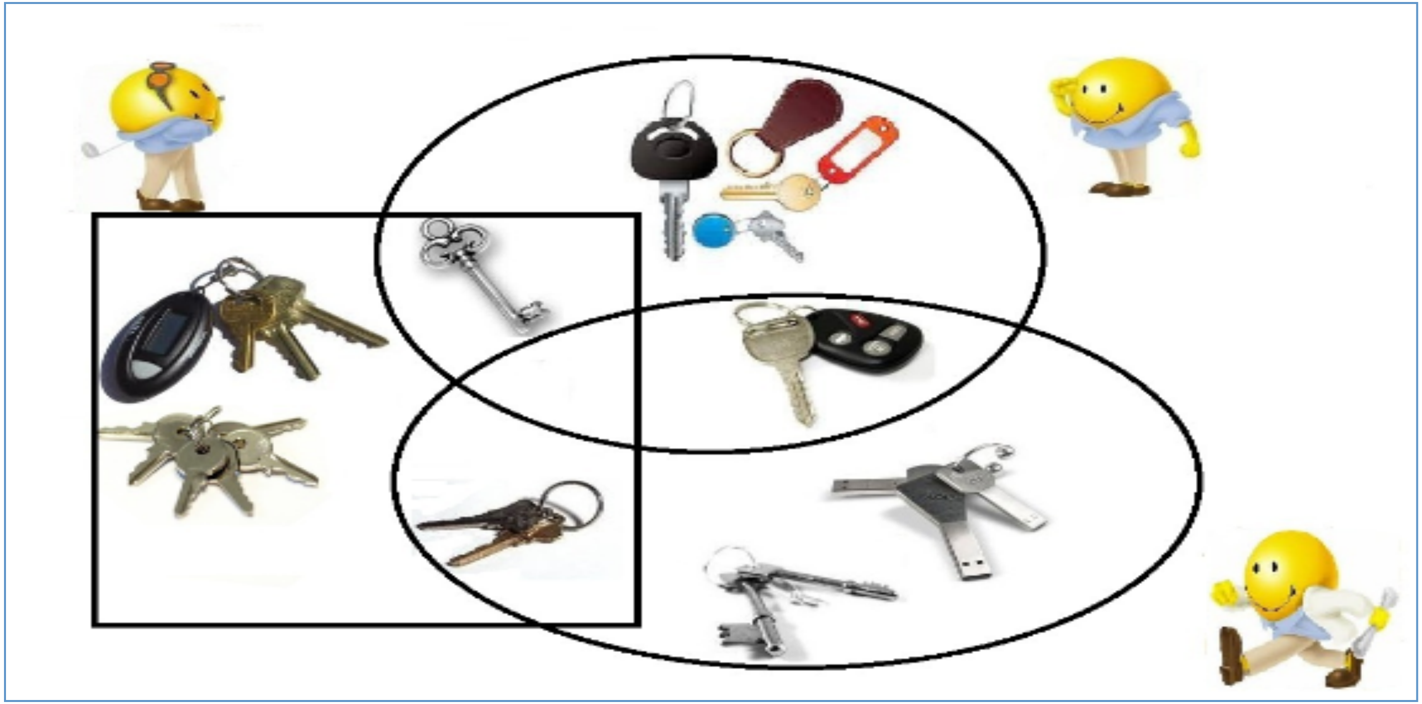
A Key Predistribution Scheme is a mechanism of distributing information among a set of users in such away that every user in a group in some specified family is able to compute individually a common key associated with that group.

Defining Feature:
Key Pre-distribution
affects all users



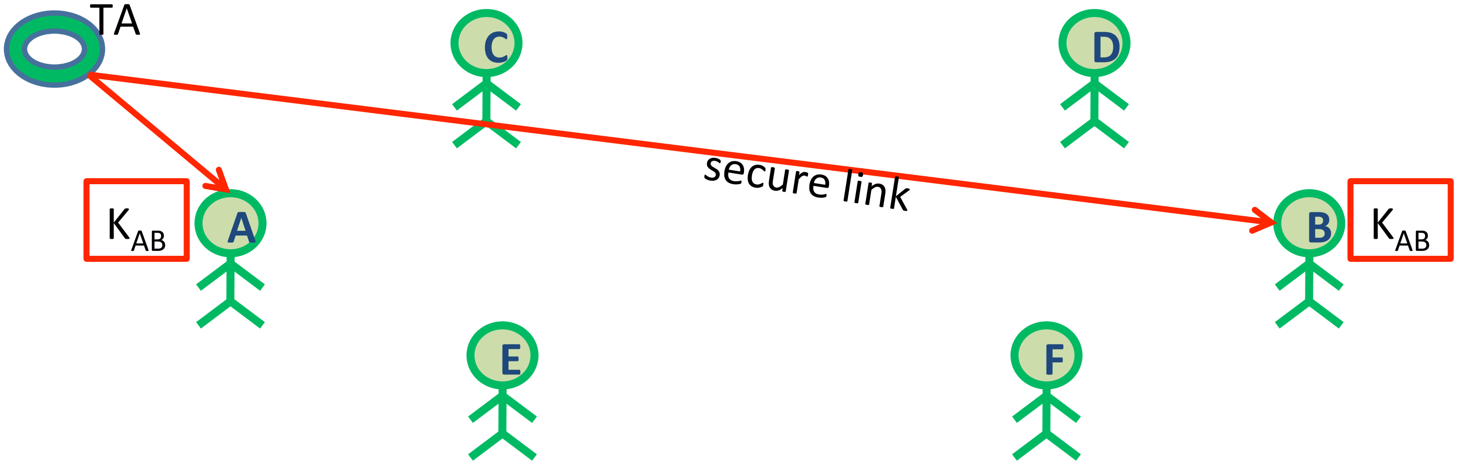
slide borrowed from Hossein Hajiabolhassan(SBU)

Key Predistribution Scheme



Slide borrowed from Hossein Hajiabolhassan(SBU)

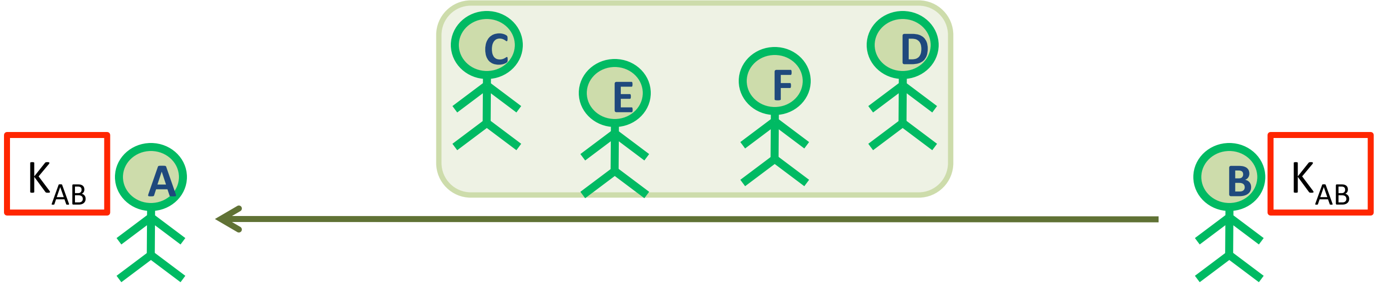
Solution using symmetric key cryptography (Naïve Scheme)



- TA generates a key and sends it securely to A and B.
- Storage in each user : N – 1
- Maximum secure links : N
- Network Overheads : $\binom{N}{2}$ transfers

can we reduce the overheads?

Trading Security for reduced Overheads



- The naïve scheme protects against N-2 colluding users
- What if we reduce this assumption to say k (< N-2) colluding users?
  - Security reduces
  - But overheads may also reduce.

Blom’s Key PreDistribution Scheme

Aim : each pair of users requires a unique key

- Unconditionally secure key distribution in a k-party colluding network (k < N – 2)
  - At-most k parties can collude
    (k parties acting together will not be able to determine the key for anyone else)
- Maximum secure links N (no change here)
- Network Transfers : N(k+1)
  (reduced from $\binom{N}{2}$)
- Storage : Each user stores (k+1) elements
  (reduced from N-1)

Blom's Key Distribution Scheme (for k=1)

- Public parameters:
  (1) prime p $(> N)$ and (2) for each user a distinct value (public) $r_u \, \varepsilon \, Z_p$

- Trusted Authority

1. Choose **secret** $a, b, c \, \varepsilon \, Z_p$ and forms the polynomial

$$f(x,y) = (a + b(x + y) + cxy) \bmod p$$
$$= (a + by) + (b + cy)x \bmod p$$

2. For each user u, the TA transmits two elements (2=k+1) to user U over a secure channel

$$a_U = (a + br_U) \bmod p \text{ and } b_U = (b + cr_U) \bmod p$$

- Usage : if 'U' and 'V' want to communicate
  - **U :** has $f(x, r_U)$, computes $K_{VU} = f(r_V, r_U)$
  - **V :** has $f(x, r_V)$, computes $K_{UV} = f(r_U, r_V) = f(r_V, r_U) = K_{VU}$

Blom’s Key Distribution Scheme (for k=1, U, V, W)

- Public parameters:
  (1) p = 17 (2) $r_u = 12;\ r_v = 7;\ r_w = 1$

- Trusted Authority

1. Choose **secret** $a=8, b=7, c=2$ and forms the polynomial

$$f(x,y) = (a + b(x + y) + cxy) \bmod p$$
$$= (a + by) + (b + cy)x \bmod p$$

2. $a_U = (8 + 7*12) \bmod 17 = 7$ and $b_U = (7 + 2*12) \bmod 17 = 14$

   $a_v = 6$ and $b_v = 4$

   $a_v = 15$ and $b_v = 9$

- Usage : if ‘U’ and ‘V’ want to communicate
  - $K_{VU} = f(r_V, r_U) = 7 + 14 * 7 \bmod 17 = 3$
  - $K_{Uv} = f(r_u, r_v) = 6 + 4 * 12 \bmod 17 = 3$

Blom’s Key Distribution Scheme (for k=1)

- Public parameters:

  (1) prime p $(> N)$ and (2) for each user a distinct value (public) $r_u \, \varepsilon \, Z_p$

a,b, c are the only secrets. If an attacker can compute these, then the system is broken!

f(x,y) is symmetric. Interchanging x and y values will not alter results.

1. Choose **secret** $a, b, c \, \varepsilon \, Z_p$ and forms the polynomial

$$f(x,y) = (a + b(x + y) + cxy) \bmod p$$
$$= (a + by) + (b + cy)x \bmod p$$

2. For each user u, the TA computes $f(x, r_u)$ and transmits two elements (k+1) to user U over a secure channel

$$a_U = (a + br_U) \bmod p \text{ and } b_U = (b + cr_U) \bmod p$$

This is an Affine transformation. There are three unknowns (a, b, c). Therefore requires 3 equations to solve. However, each user has only a_U and b_U.

Needs more information!!

- Usage : if ‘U’ and ‘V’ want to communicate
  - U : has $f(x, r_U)$, computes $K_{VU} = f(r_V, r_U)$
  - V : has $f(x, r_V)$, computes $K_{UV} = f(r_U, r_V) = f(r_V, r_U) = K_{VU}$

Blom’s scheme is unconditionally secure

- What does this means? Any other user W (not U or V) cannot get any information about K_{UV}

apriori probability of K_{UV} = aposteriori probability of K_{UV}

$=1/|Z_p|$

Given all of Blom’s public parameters and $f(x, r_W)$

What ‘W’ has?

$\mathbf{a_W = a + br_W}$

$\mathbf{b_W = b + cr_W}$

Two equations; three unknowns (a, b, c)
This is an underdetermined system therefore number of solutions possible is |Zp|.

Aposteriori probability of $K_{UV} = 1/|Z_p|$

2-party Colluding Attackers

- If two attackers (say W and X) collude, then
  4 equations present and 3 unknowns
  This will result in a unique solution for a,b,c … system broken!!!

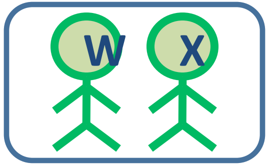
What 'W' and 'X' have?

$\mathbf{a_W = a + br_W}$

$\mathbf{b_W = b + cr_W}$

$\mathbf{a_X = a + br_X}$

$\mathbf{b_X = b + cr_X}$



2-party coalition attackers

Thus, the scheme is not secure against 2 (or more) party colluding attacks

Generalizing Blom’s Scheme

- More complex polynomial so that secret coefficients cannot be retrieved
- For a k-party colluding network

$$f(x,y)=\sum_{i=0}^{k}\sum_{j=0}^{k} a_{i,j}x^{i}y^{j} \bmod p$$

$$\text{where}\, a_{i,j}\in Z_{p}\ \ (0\le i,j\le k)\ \ and\ \ a_{i,j}=a_{j,i}\ \ for\ all\ i,j$$

Limits of Blom’s Scheme

Pairwise keys cannot be changed

- i.e. U and V cannot change their keys

To change keys, all users need to be reconfigured

Thus, it is difficult to implement this scheme for session keys

Key Distribution Patterns

- suppose we have a TA and a network of n users, $\mathcal{U} = \{U_1, \ldots, U_n\}$
- the TA chooses v random keys, say $k_1, \ldots, k_v \in \mathcal{K}$, where $(\mathcal{K}, +)$ is an additive abelian group, and gives a (different) subset of keys to each user (This is a secret operation).
- a *key distribution pattern* is a public v by n incidence matrix, denoted M, which has entries in $\{0, 1\}$
- M specifies which users are to receive which keys: user U_j is given the key k_i if and only if $M[i, j] = 1$

Key Distribution Patterns (Trivial Example)

Suppose

- There are n users (n = 4)
- and v keys (v = 6)

$$U_1 \ \ has \ \ keys \ \ k_1, k_2, k_3$$
$$U_2 \ \ has \ \ keys \ \ k_1, k_4, k_5$$
$$U_3 \ \ has \ \ keys \ \ k_2, k_4, k_6$$
$$U_4 \ \ has \ \ keys \ \ k_3, k_5, k_6$$

$$M = \begin{array}{cc} \begin{array}{cccc} U_1 & U_2 & U_3 & U_4 \end{array} & \\ \underbrace{\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}}_{\text{users}} & \left.\begin{array}{c} k_1 \\ k_2 \\ k_3 \\ k_4 \\ k_5 \\ k_6 \end{array}\right\} \text{keys} \end{array}$$

Group Keys

$$P \subseteq \mathcal{U}$$

- Consider that a subset of users P (|P| ≥ 2) want to communicate together
- Define, $keys(P) = \bigcap_{U_j \in P} keys(U_j)$

$$keys(U_1) = \{ k_1, k_2, k_3 \}$$
$$keys(U_2) = \{ k_1, k_4, k_5 \}$$

$$keys(P) = keys(U_1) \cap keys(U_2) = k_1$$

In this case, k_P = keys(P) = k_1 can be used as the key

- Each user in P can compute keys(P) independently because M is public

$$If \ |keys(P)| > 2, \ then \ define \quad k_P = \sum_{i \in keys(P)} k_i \bmod K$$

Security of Group Keys

- Consider another subset of users F, who want to collaborate to determine the group key k_P

① $If \quad F \cap P \neq \phi, \; then \; there \; exists \; some \; U_j \in F \; who \; can \; compute \; k_P$

② $Assume \; F \cap P = \phi$

$$If \left(keys(P) \subseteq \bigcup_{U_j \in F} keys(U_j) \right)$$

$then \; there \; exists \; a \; subset \; in \; F \; who \; can \; cooperate \; to \; compute \; k_P$

If such a subset does not exist, then the system in unconditionally secure

Another Example

- M: n=7, v=7
- Storage in each user is 4

$$M = \begin{array}{cccccccc} U_1 & U_2 & U_3 & U_4 & & & & \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & k_1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & k_7 \end{array}$$

$keys(U_1) = \{1, 4, 6, 7\}$, $keys(U_2) = \{1, 2, 5, 7\}$, and $keys(U_1, U_2) = \{1, 7\}$, so $k_{\{U_1, U_2\}} = k_1 + k_7$.

No other user has both k_1 and k_7.
U_3 has k_1 but not k_7
U_4 has k_7 but not k_1
Therefore the scheme is secure against single party attackers

The scheme is not secure against two (or more) party attackers

If U_3 and U_4 collaborate, they can compute $k_1 + k_7$

Key Distribution Pattern (Trivial Example)

- If there are n users,
- For each pair to communicate securely, the matrix size is $\binom{n}{2} \times n$
- Each user must store n – 1 keys

- Security Guarantee:

  The system is secure against a coalition of size n – 2.

  i.e. to get the key between Alice and Bob, everyone remaining must cooperate

Maximum security guarantees, but huge of storage requirements.

Can we trade security for lower storage?

Fiat-Naor Key Distribution Patterns

- Consider n users : U = {U_1, U_2, ….,U_n}.
- How do we construct a key pattern matrix M which can resist attacks from w collating users (1 ≤ w ≤n)

(w is called the security parameter)

1. Compute : $$v = \sum_{i=0}^{w} \binom{n}{i}$$
2. Compute the matrix M (v x n)
   - The columns are the users (U_1, U_2, ….., U_n)
   - Each row corresponds incidence vector of a subset of users with cardinality at-least n-w

Example

- Number of users is 6
- Security Parameter w = 1
- v = 7

Subsets of U having at-least n-w elements

$\{U_1, U_2, U_3, U_4, U_5, U_6\}$

$\{U_1, U_2, U_3, U_4, U_5\}$

$\{U_1, U_2, U_3, U_5, U_6\}$

$\{U_1, U_2, U_4, U_5, U_6\}$

$\{U_1, U_3, U_4, U_5, U_6\}$

$\{U_2, U_3, U_4, U_5, U_6\}$

$$M = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Example

- Number of users is 6
- Security Parameter w = 1
- v = 7

Consider $P = \{U_1, U_3, U_4\}$

$$k_{\{U_1,U_2,U_4\}} = k_1 + k_2 + k_3 + k_6$$

$$M = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{array}{c} \leftarrow \\ \leftarrow \\ \leftarrow \\ \\ \\ \leftarrow \\ \\ \end{array}$$

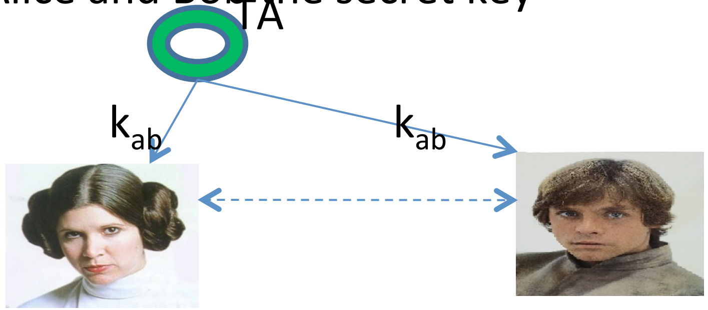
Note that no other user (individually) has access to all keys k_1, k_2, k_3, and k_6
Thus the system is secure against non-cooperating attackers

Session Keys

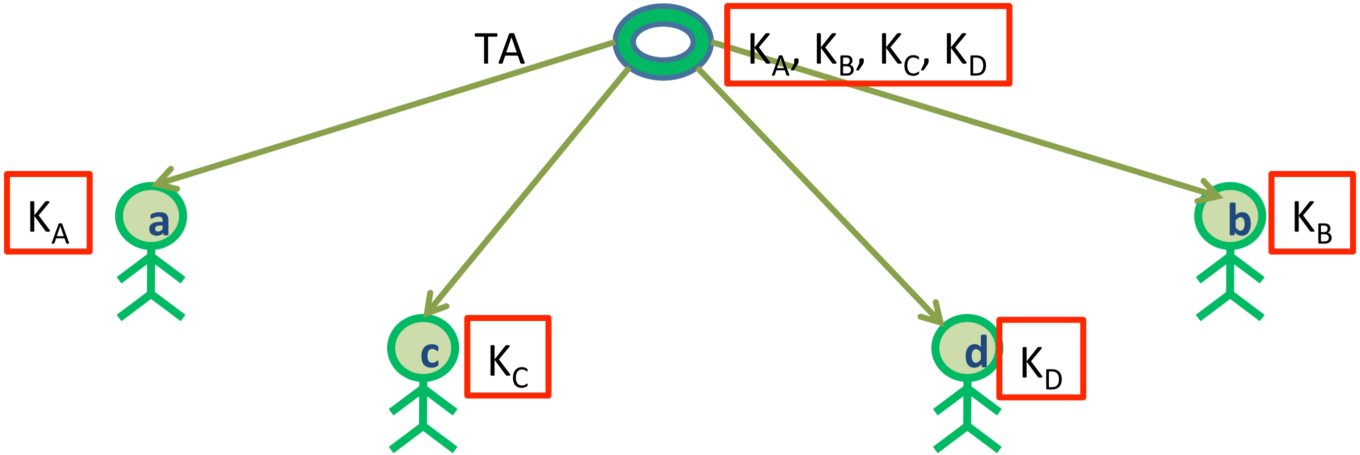
Are between pairs of users (e.g. Alice and Bob)

Distribution of Session Keys

- Makes use of the TA
  - TA tells Alice and Bob the secret key

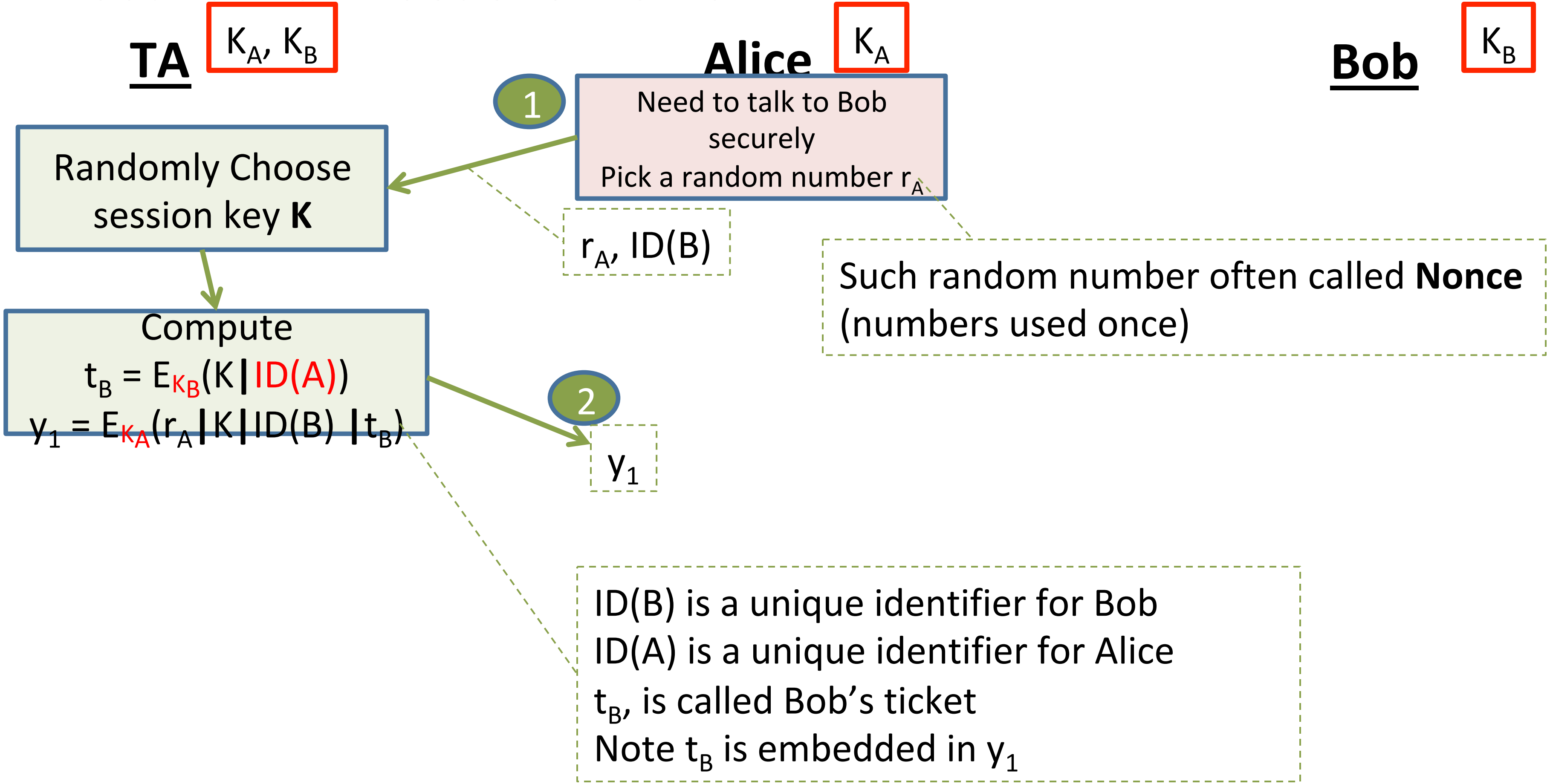


Setting : (shared keys with TA)



- TA shares a secret key with each user.
- This key is used to securely communicate between TA and a user.

Needham Schroeder Scheme



TA K_A, K_B

Alice K_A

Bob K_B

1

Need to talk to Bob securely

Pick a random number r_A

Randomly Choose session key **K**

r_A, ID(B)

Such random number often called **Nonce** (numbers used once)

Compute

$t_B = E_{K_B}(K \mid ID(A))$

$y_1 = E_{K_A}(r_A \mid K \mid ID(B) \mid t_B)$

2

y_1

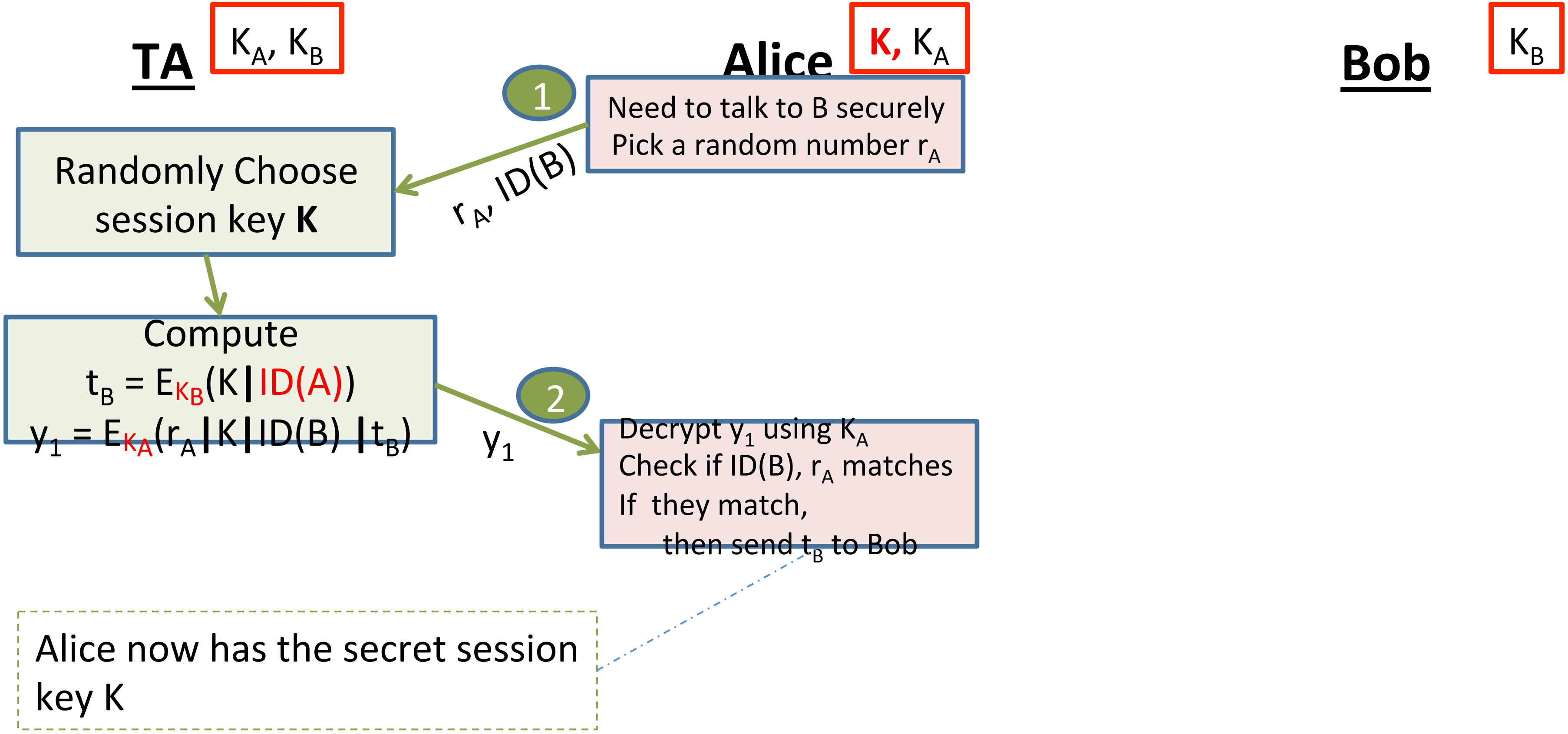
ID(B) is a unique identifier for Bob

ID(A) is a unique identifier for Alice

t_B, is called Bob's ticket

Note t_B is embedded in y_1

Needham Schroeder Scheme



TA K_A, K_B

Alice K, K_A

Bob K_B

① Need to talk to B securely
Pick a random number r_A

r_A, ID(B)

Randomly Choose session key **K**

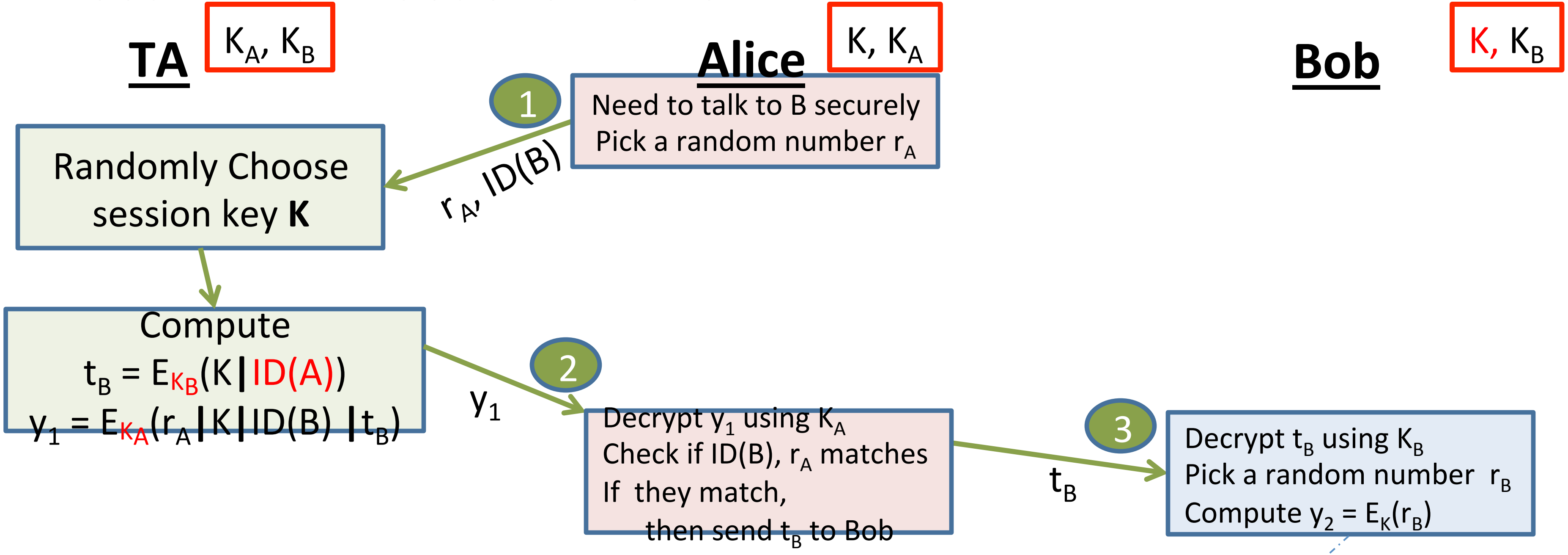
Compute
$t_B = E_{K_B}(K | ID(A))$
$y_1 = E_{K_A}(r_A | K | ID(B) | t_B)$

② y_1

Decrypt y_1 using K_A
Check if ID(B), r_A matches
If they match,
then send t_B to Bob

Alice now has the secret session key K

Needham Schroeder Scheme



TA K_A, K_B

Alice K, K_A

Bob K, K_B

①

Need to talk to B securely
Pick a random number r_A

r_A, ID(B)

Randomly Choose session key **K**

Compute
$t_B = E_{K_B}(K|ID(A))$
$y_1 = E_{K_A}(r_A|K|ID(B)\ |t_B)$

②

y_1

Decrypt y_1 using K_A
Check if ID(B), r_A matches
If they match,
then send t_B to Bob

③

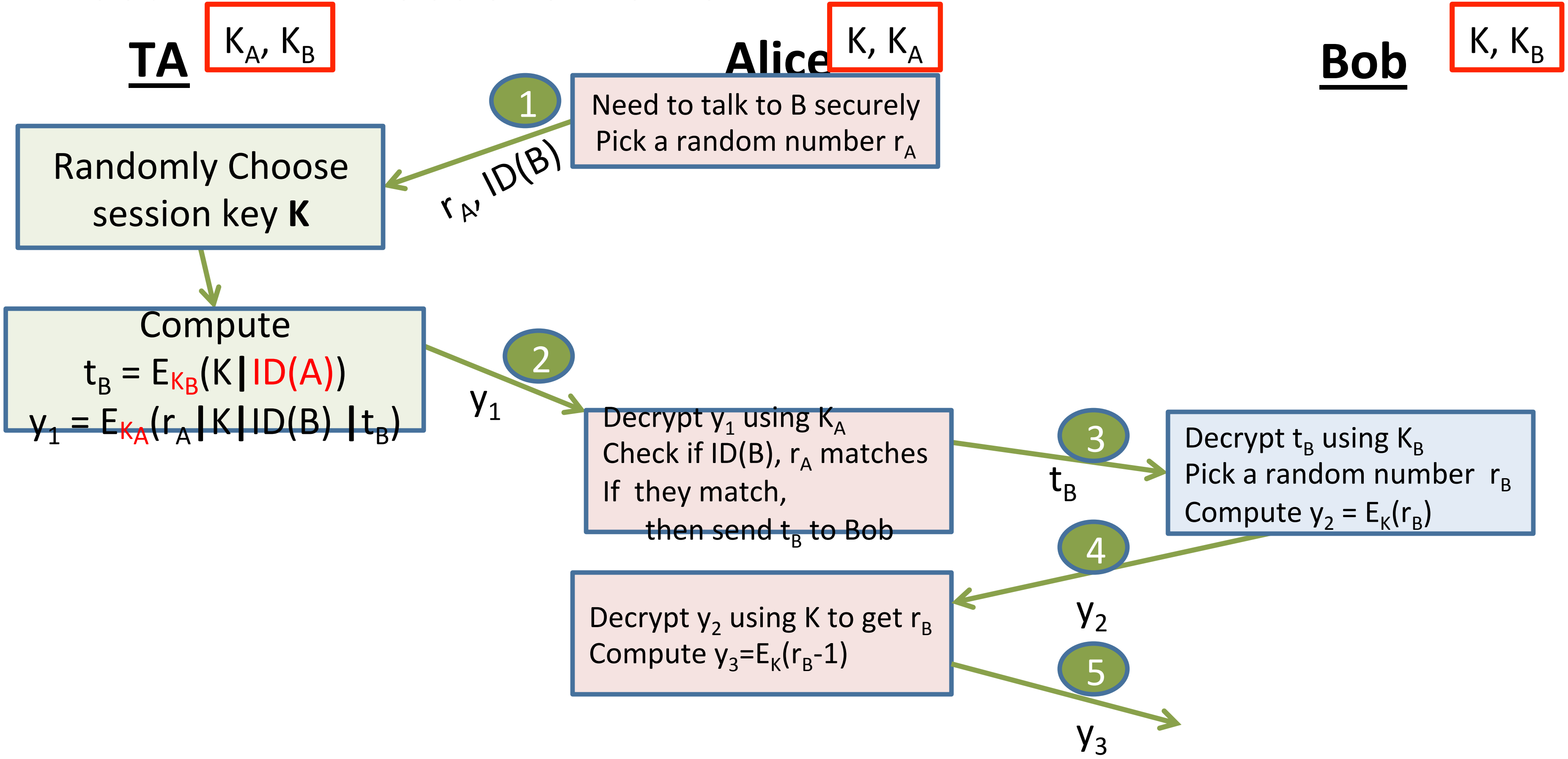
t_B

Decrypt t_B using K_B
Pick a random number r_B
Compute $y_2 = E_K(r_B)$

Bob too now has the secret K,
He also has ID(A), so he knows it's a session key with Alice

K is used for encrypting r_B

Needham Schroeder Scheme



TA K_A, K_B

Alice K, K_A

Bob K, K_B

① Need to talk to B securely
Pick a random number r_A

r_A, ID(B)

Randomly Choose session key **K**

Compute
$t_B = E_{K_B}(K|ID(A))$
$y_1 = E_{K_A}(r_A|K|ID(B)\ |t_B)$

② y_1

Decrypt y_1 using K_A
Check if ID(B), r_A matches
If they match,
then send t_B to Bob

③ t_B

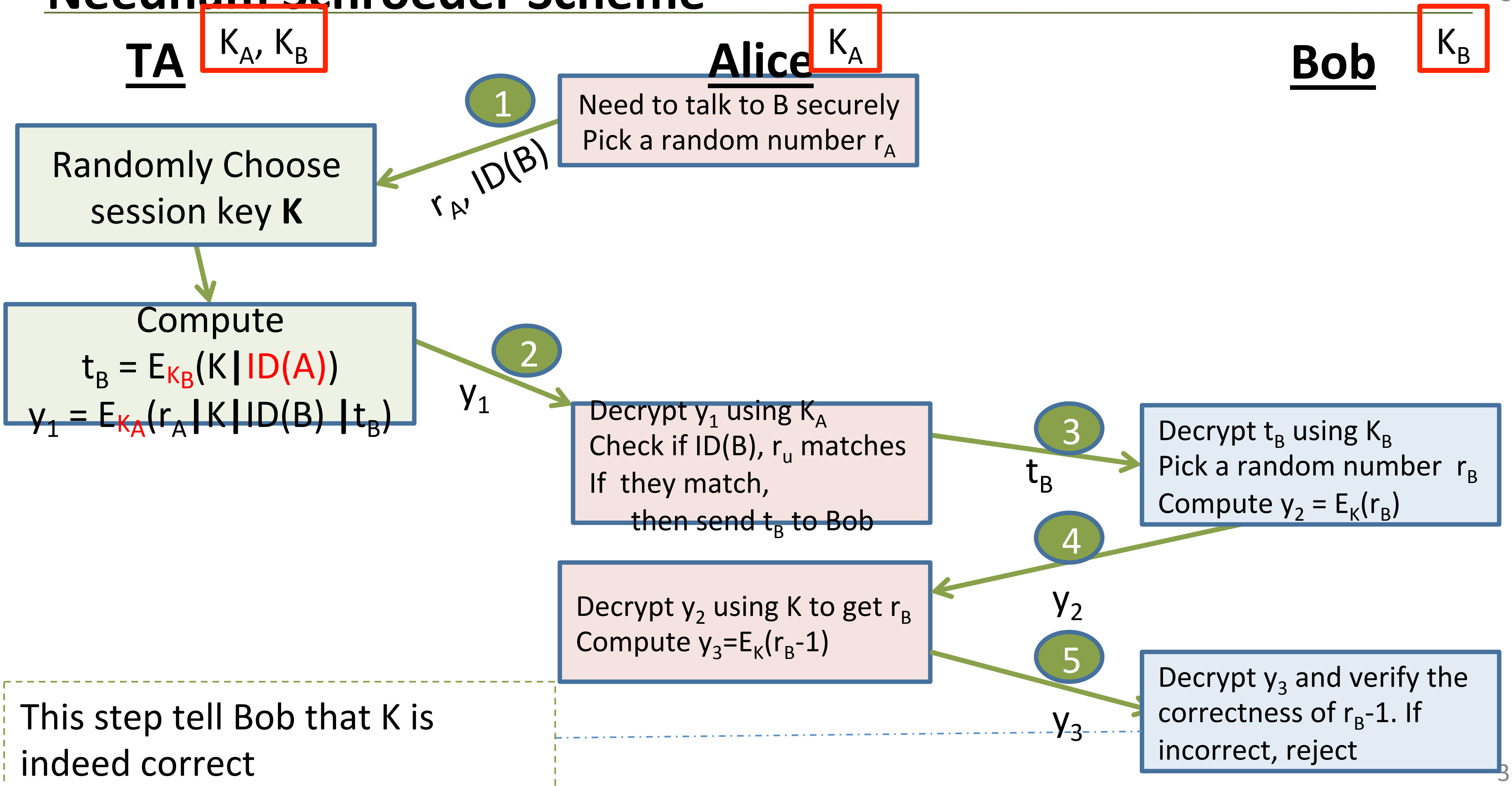
Decrypt t_B using K_B
Pick a random number r_B
Compute $y_2 = E_K(r_B)$

④ y_2

Decrypt y_2 using K to get r_B
Compute $y_3=E_K(r_B-1)$

⑤ y_3

Needham Schroeder Scheme



TA K_A, K_B

Alice K_A

Bob K_B

① r_A, ID(B)

Need to talk to B securely
Pick a random number r_A

Randomly Choose session key **K**

Compute
$t_B = E_{K_B}(K|ID(A))$
$y_1 = E_{K_A}(r_A|K|ID(B)|t_B)$

② y_1

Decrypt y_1 using K_A
Check if ID(B), r_u matches
If they match,
then send t_B to Bob

③ t_B

Decrypt t_B using K_B
Pick a random number r_B
Compute $y_2 = E_K(r_B)$

④ y_2

Decrypt y_2 using K to get r_B
Compute $y_3=E_K(r_B-1)$

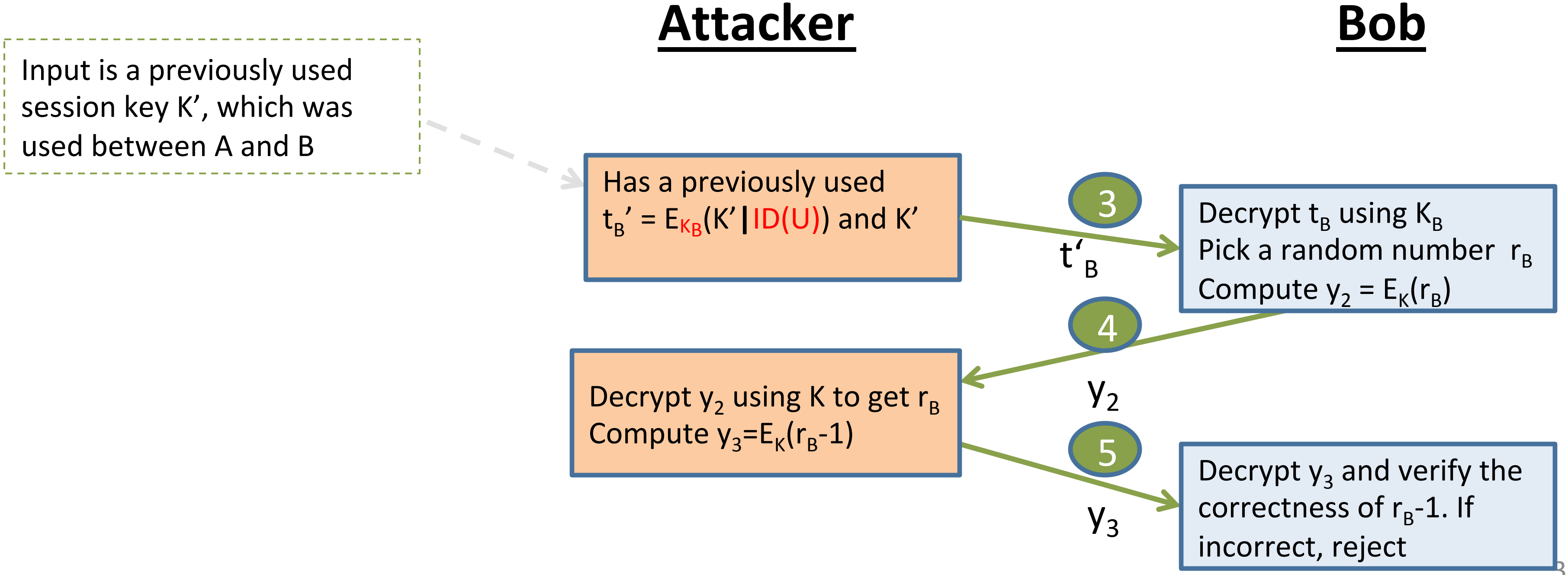
⑤ y_3

Decrypt y_3 and verify the correctness of r_B-1. If incorrect, reject

This step tell Bob that K is indeed correct

Denning-Sacco Attack on the NS Scheme

This is a **known session key attack / replay attack**, where the attacker has a previously used session key between U and V, and can convinces V to use this old session key



Attacker

Bob

Input is a previously used session key K’, which was used between A and B

Has a previously used $t_B' = E_{K_B}(K' | ID(U))$ and K’

③ t'_B

Decrypt t_B using K_B
Pick a random number r_B
Compute $y_2 = E_K(r_B)$

④ y_2

Decrypt y_2 using K to get r_B
Compute $y_3=E_K(r_B-1)$

⑤ y_3

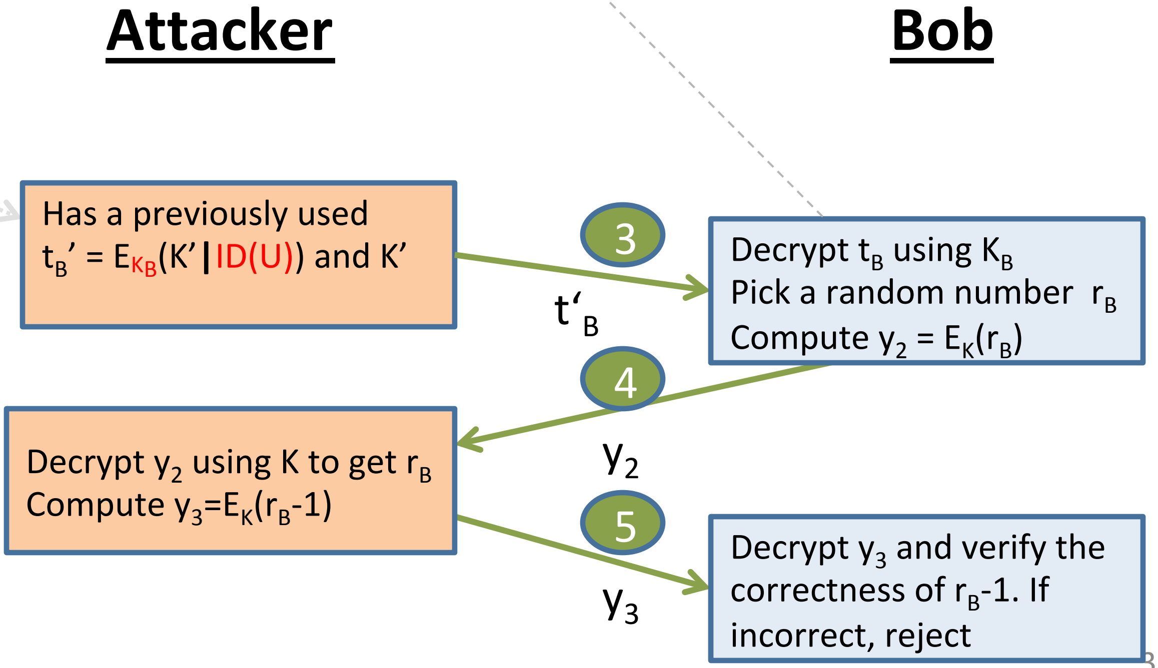
Decrypt y_3 and verify the correctness of r_B-1. If incorrect, reject

Denning-Sacco Attack on the NS Scheme

What is the flaw in the NS scheme?

Bob has no way to know if t_B has been used previously.

Input is a previously used session key K’, which was used between A and B



Attacker

Bob

Has a previously used $t_B' = E_{K_B}(K' | ID(U))$ and K’

3

t'_B

Decrypt t_B using K_B
Pick a random number r_B
Compute $y_2 = E_K(r_B)$

4

y_2

Decrypt y_2 using K to get r_B
Compute $y_3 = E_K(r_B - 1)$

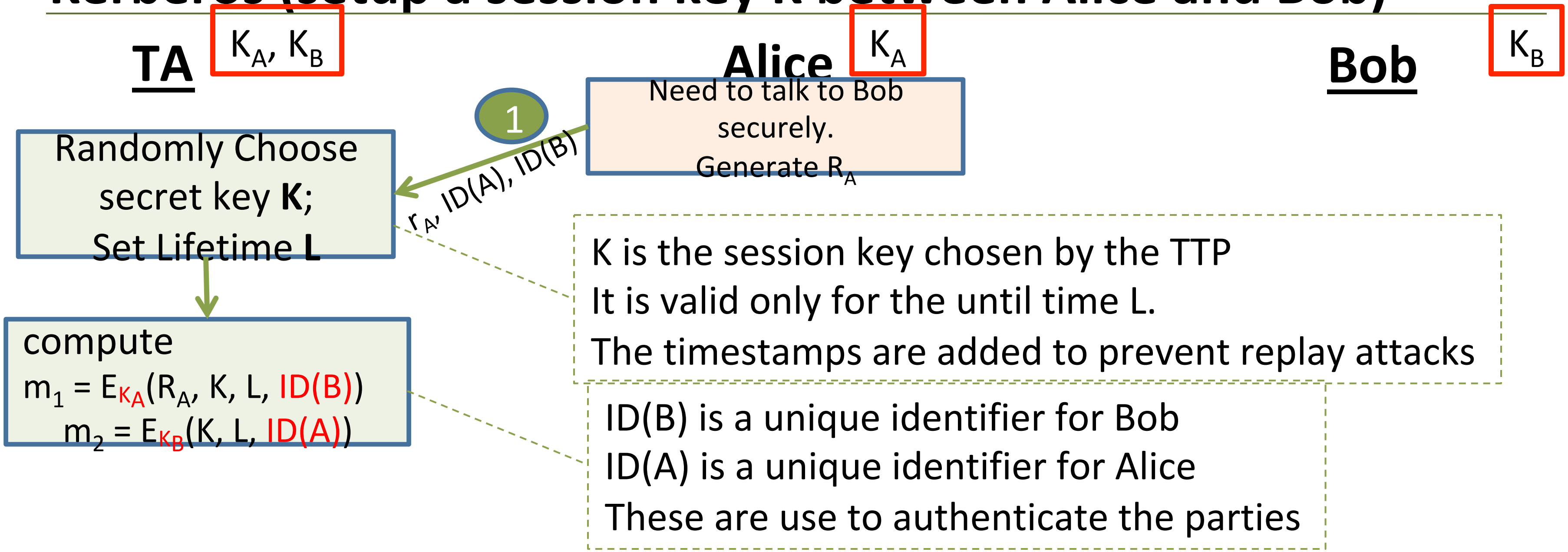
5

y_3

Decrypt y_3 and verify the correctness of r_B-1. If incorrect, reject

Fixed in Kerberos by adding a timestamp

Kerberos (setup a session key K between Alice and Bob)



TA K_A, K_B

Alice K_A

Bob K_B

1

Need to talk to Bob securely.
Generate R_A

r_A, ID(A), ID(B)

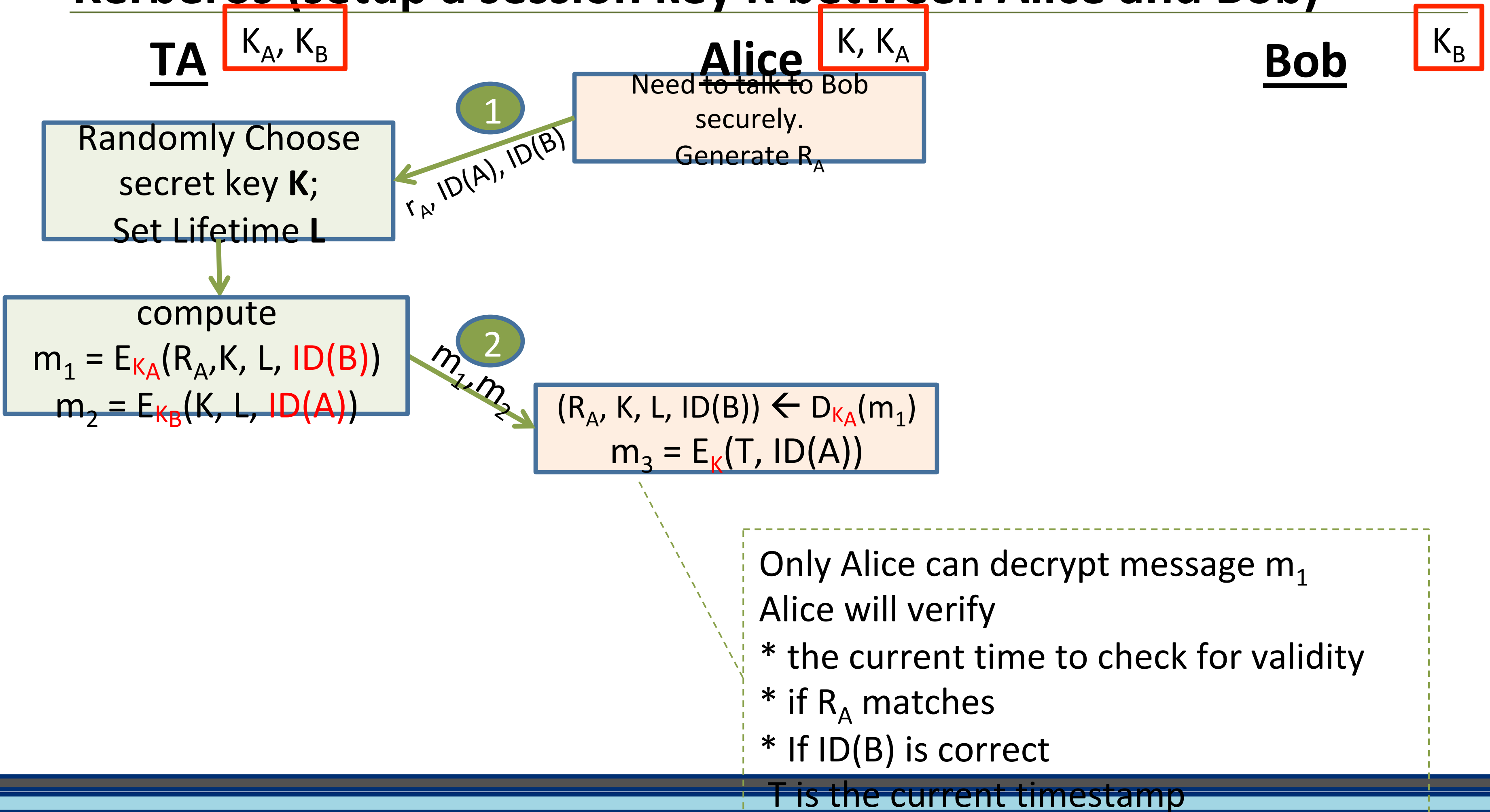
Randomly Choose secret key **K**;
Set Lifetime **L**

compute
$m_1 = E_{K_A}(R_A, K, L, ID(B))$
$m_2 = E_{K_B}(K, L, ID(A))$

K is the session key chosen by the TTP
It is valid only for the until time L.
The timestamps are added to prevent replay attacks

ID(B) is a unique identifier for Bob
ID(A) is a unique identifier for Alice
These are use to authenticate the parties

Kerberos (setup a session key K between Alice and Bob)



TA K_A, K_B

Alice K, K_A

Bob K_B

Need ~~to talk to~~ Bob securely.
Generate R_A

1

r_A, ID(A), ID(B)

Randomly Choose secret key **K**;
Set Lifetime **L**

compute
$m_1 = E_{K_A}(R_A, K, L, ID(B))$
$m_2 = E_{K_B}(K, L, ID(A))$

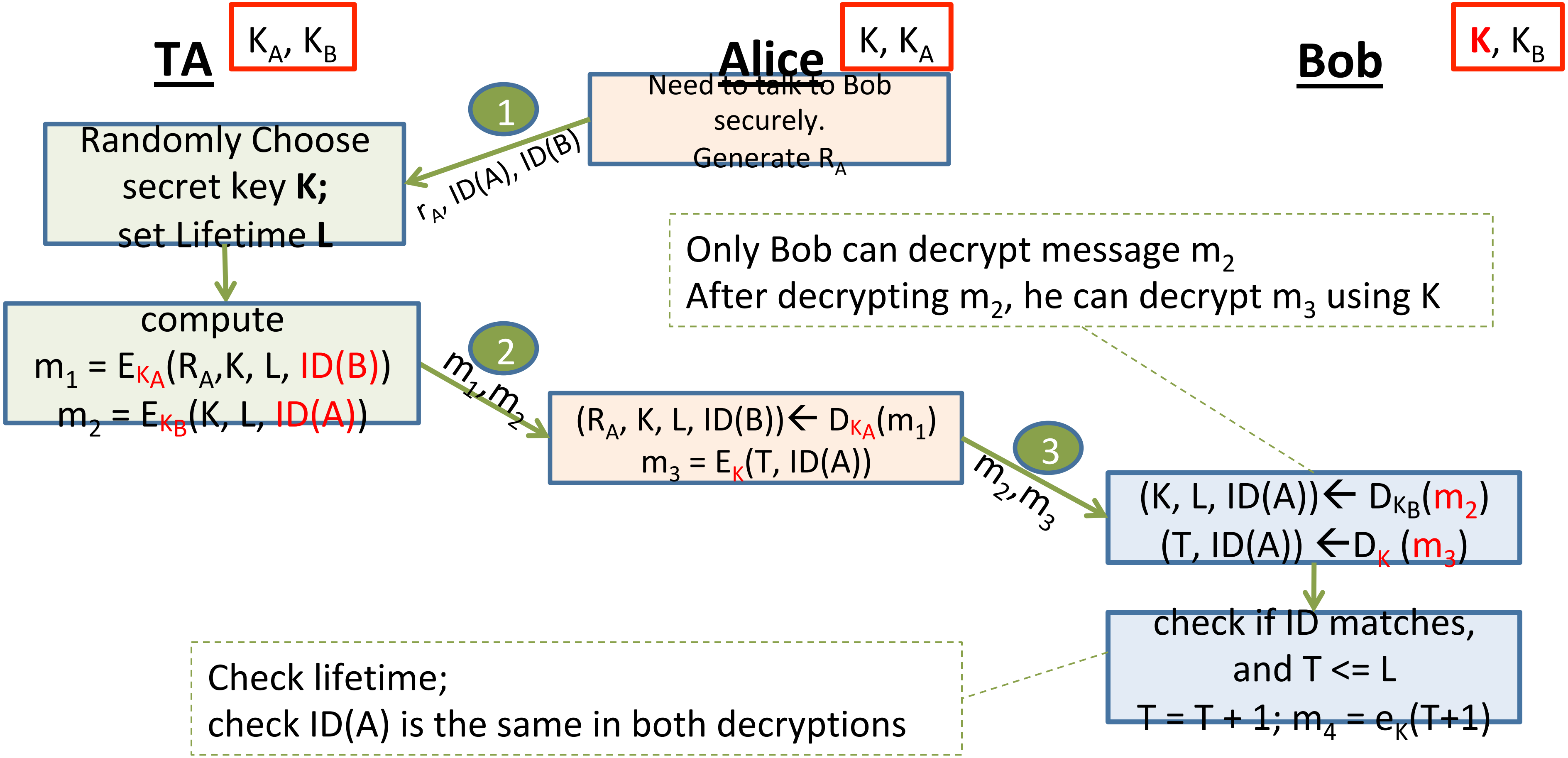
2

m_1, m_2

$(R_A, K, L, ID(B)) \leftarrow D_{K_A}(m_1)$
$m_3 = E_K(T, ID(A))$

Only Alice can decrypt message m_1
Alice will verify
* the current time to check for validity
* if R_A matches
* If ID(B) is correct
T is the current timestamp

Kerberos (setup a session key K between Alice and Bob)



TA K_A, K_B

Alice K, K_A

Bob K, K_B

Need ~~to talk to~~ Bob securely.
Generate R_A

①

r_A, ID(A), ID(B)

Randomly Choose secret key **K;**
set Lifetime **L**

compute
$m_1 = E_{K_A}(R_A, K, L, ID(B))$
$m_2 = E_{K_B}(K, L, ID(A))$

②

m_1, m_2

$(R_A, K, L, ID(B)) \leftarrow D_{K_A}(m_1)$
$m_3 = E_K(T, ID(A))$

③

m_2, m_3

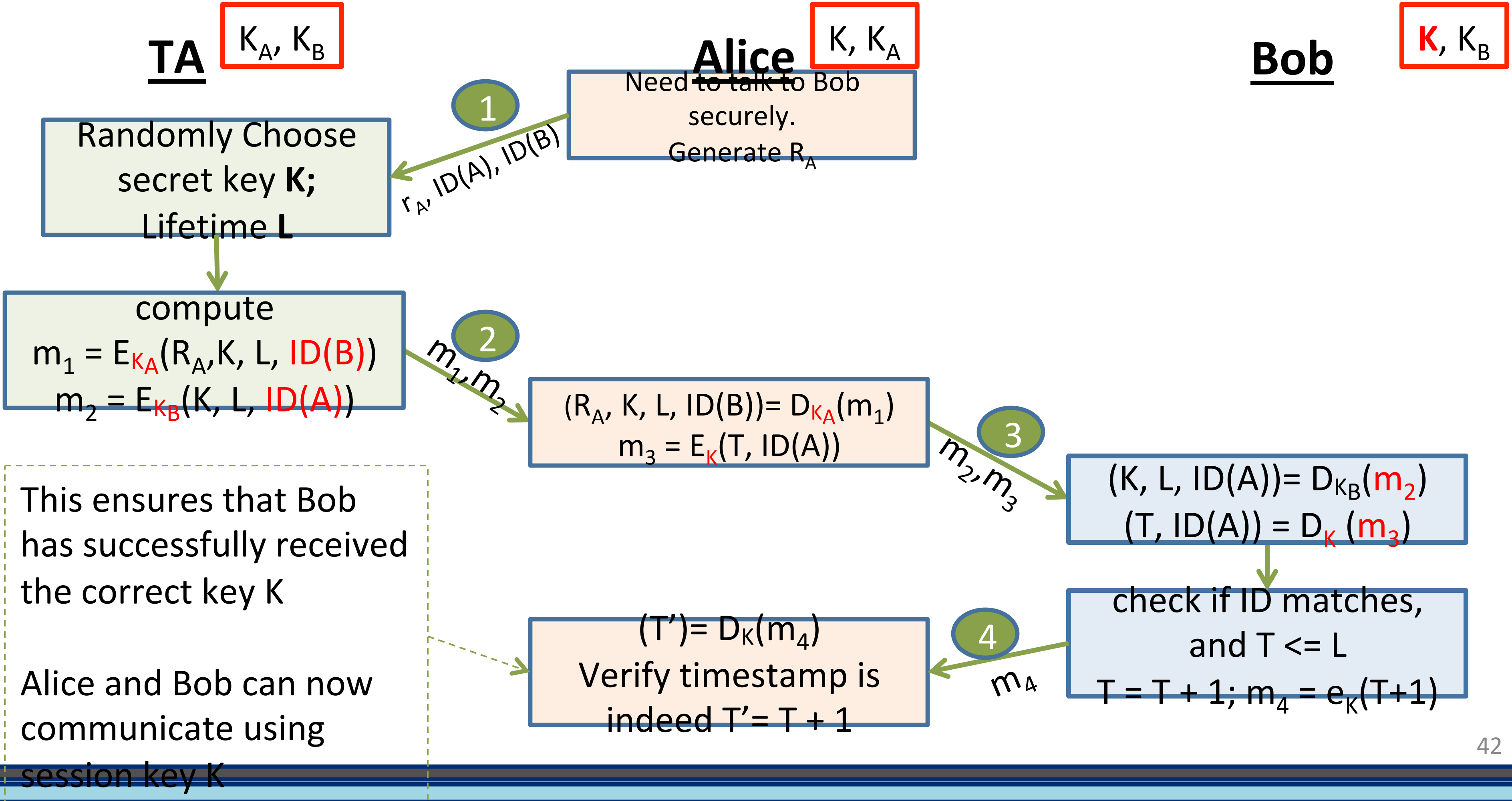
Only Bob can decrypt message m_2
After decrypting m_2, he can decrypt m_3 using K

$(K, L, ID(A)) \leftarrow D_{K_B}(m_2)$
$(T, ID(A)) \leftarrow D_K(m_3)$

check if ID matches, and T <= L
$T = T + 1; m_4 = e_K(T+1)$

Check lifetime;
check ID(A) is the same in both decryptions

Kerberos (setup a session key K between Alice and Bob)



TA K_A, K_B

Alice K, K_A

Bob K, K_B

Need ~~to talk to~~ Bob securely.
Generate R_A

① r_A, ID(A), ID(B)

Randomly Choose secret key **K;**
Lifetime **L**

compute
$m_1 = E_{K_A}(R_A, K, L, ID(B))$
$m_2 = E_{K_B}(K, L, ID(A))$

② m_1, m_2

$(R_A, K, L, ID(B)) = D_{K_A}(m_1)$
$m_3 = E_K(T, ID(A))$

③ m_2, m_3

$(K, L, ID(A)) = D_{K_B}(m_2)$
$(T, ID(A)) = D_K(m_3)$

check if ID matches,
and T <= L
$T = T + 1$; $m_4 = e_K(T+1)$

④ m_4

$(T') = D_K(m_4)$
Verify timestamp is indeed $T' = T + 1$

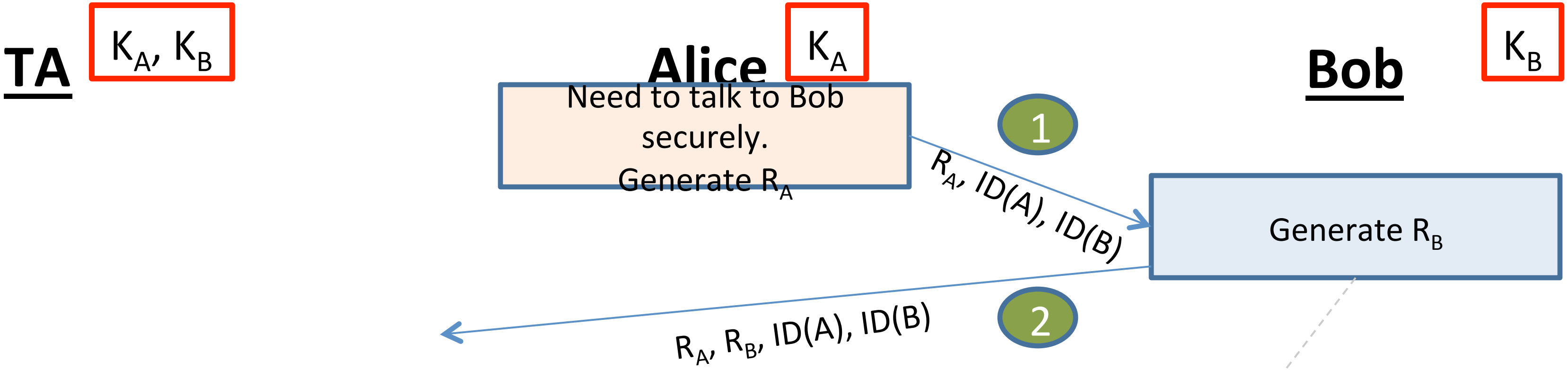
This ensures that Bob has successfully received the correct key K

Alice and Bob can now communicate using session key K

Limitations of Kerberos

- Requires all users and the TA to be synchronized due to the timestamp requirements.
  - Not easily done
- Does not completely prevent replay attacks
  - Replay attacks can still occur within the lifetime (L) of a key
- Is key confirmation (step 4) actually needed?
  - Nobody else can decrypted the encrypted message anyways.

Bellare-Rogaway Scheme



TA K_A, K_B

Alice K_A

Bob K_B

Need to talk to Bob securely.
Generate R_A

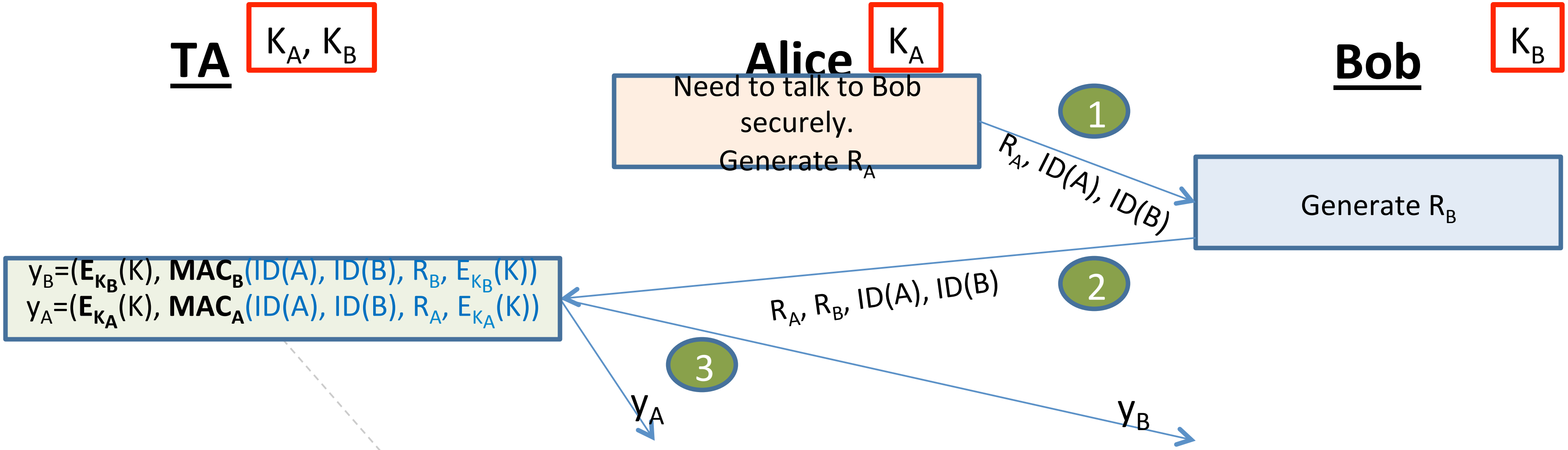
① R_A, ID(A), ID(B)

Generate R_B

② R_A, R_B, ID(A), ID(B)

Notice that Alice contacts Bob first.
This is crucial to eliminate replay attacks

Bellare-Rogaway Scheme



TA K_A, K_B

Alice K_A

Bob K_B

Need to talk to Bob securely.
Generate R_A

① R_A, ID(A), ID(B)

Generate R_B

② R_A, R_B, ID(A), ID(B)

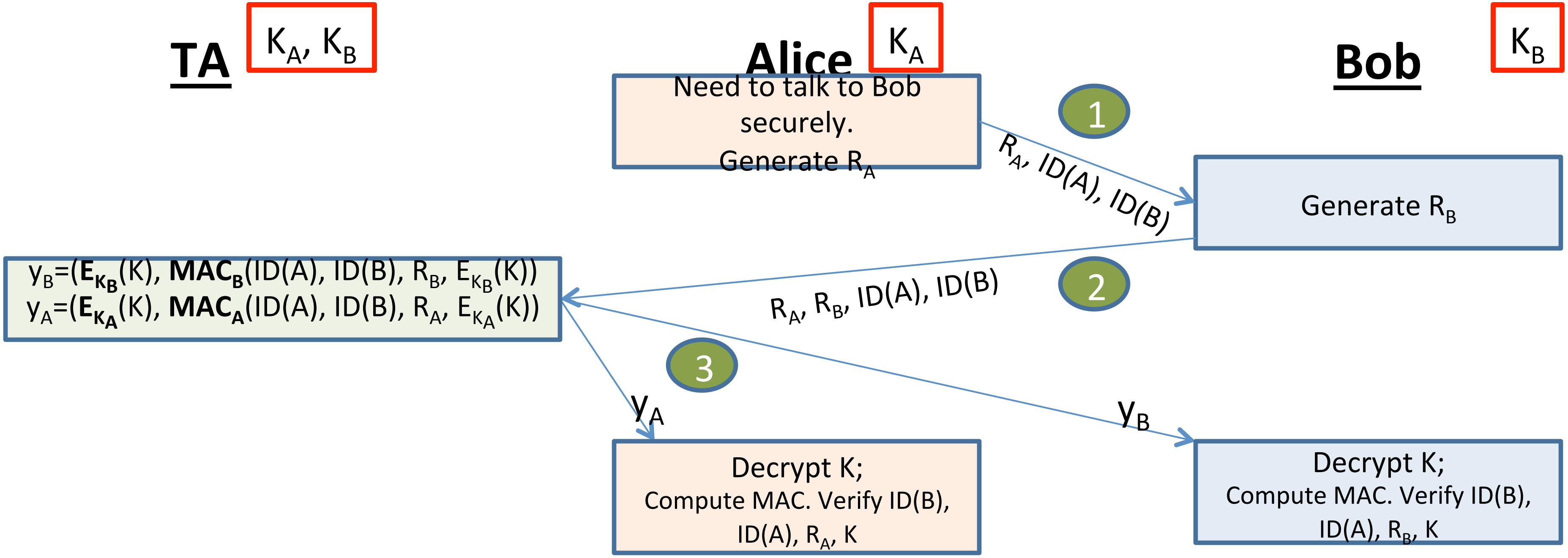
$y_B=(\mathbf{E}_{\mathbf{K_B}}(K), \mathbf{MAC}_{\mathbf{B}}(ID(A), ID(B), R_B, E_{K_B}(K))$

$y_A=(\mathbf{E}_{\mathbf{K_A}}(K), \mathbf{MAC}_{\mathbf{A}}(ID(A), ID(B), R_A, E_{K_A}(K))$

③ y_A y_B

Uses MAC, prevents double encryption.
No timestamps present

Bellare-Rogaway Scheme



TA K_A, K_B

Alice K_A

Bob K_B

Need to talk to Bob securely. Generate R_A

1

R_A, ID(A), ID(B)

Generate R_B

2

R_A, R_B, ID(A), ID(B)

y_B=($\mathbf{E}_{\mathbf{K_B}}$(K), $\mathbf{MAC}_{\mathbf{B}}$(ID(A), ID(B), R_B, E_{K_B}(K))

y_A=($\mathbf{E}_{\mathbf{K_A}}$(K), $\mathbf{MAC}_{\mathbf{A}}$(ID(A), ID(B), R_A, E_{K_A}(K))

3

y_A

y_B

Decrypt K;
Compute MAC. Verify ID(B), ID(A), R_A, K

Decrypt K;
Compute MAC. Verify ID(B), ID(A), R_B, K

Replay attacks prevented. As Alice and Bob expect a key K corresponding to R_A and R_B

No key confirmation phase. Alice / Bob does not know if the other person has received the key.

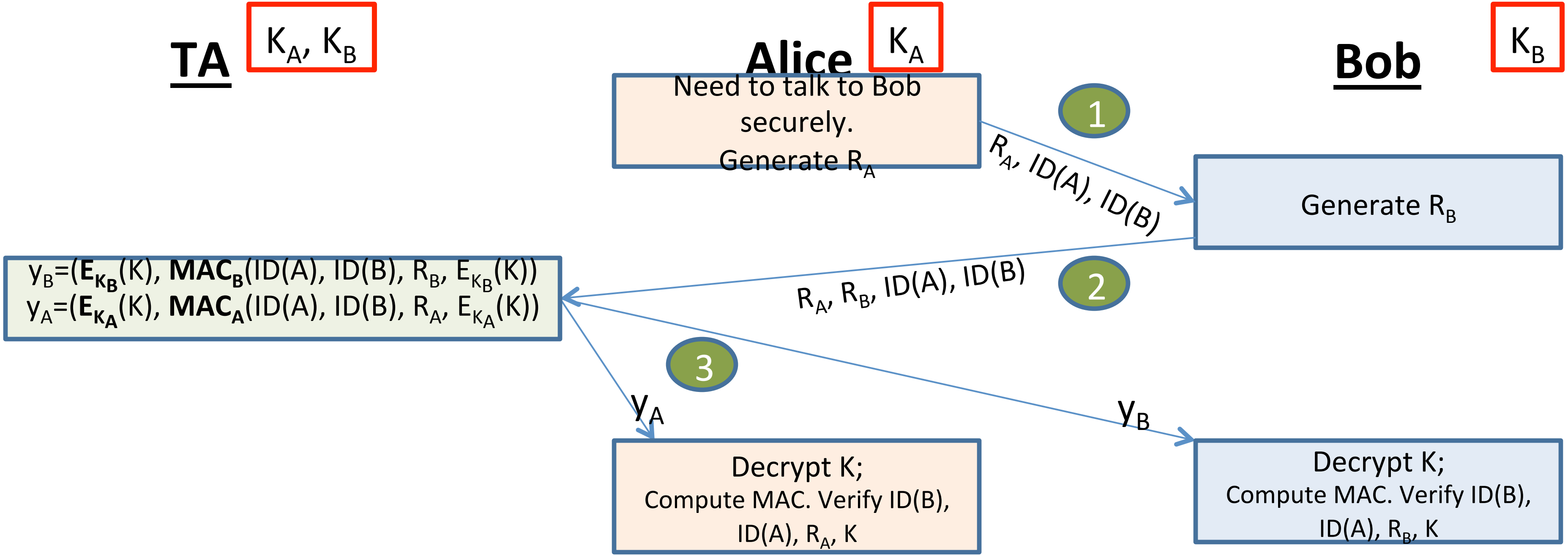
Security of Bellare-Rogaway Session Key Distribution Scheme

- The Bellare-Rogaway scheme is secure under the assumptions
  - A, B, and TA are honest
  - MACs generated are secure
  - Secret keys are not known to anyone other than the required parties
  - Random numbers are generated perfectly

BR Scheme Analysis : When Attacker is Passive

Attacker Knows r_A, r_B, ID(A), ID(B), y_A, y_B

Attacker cannot get the K because she doesn't have K_A or K_B that decrypts Y_A, Y_B respectively



TA K_A, K_B

Alice K_A

Bob K_B

Need to talk to Bob securely. Generate R_A

① R_A, ID(A), ID(B)

Generate R_B

② R_A, R_B, ID(A), ID(B)

y_B=($\mathbf{E}_{K_B}$(K), $\mathbf{MAC}_B$(ID(A), ID(B), R_B, E_{K_B}(K))

y_A=($\mathbf{E}_{K_A}$(K), $\mathbf{MAC}_A$(ID(A), ID(B), R_A, E_{K_A}(K))

③ y_A y_B

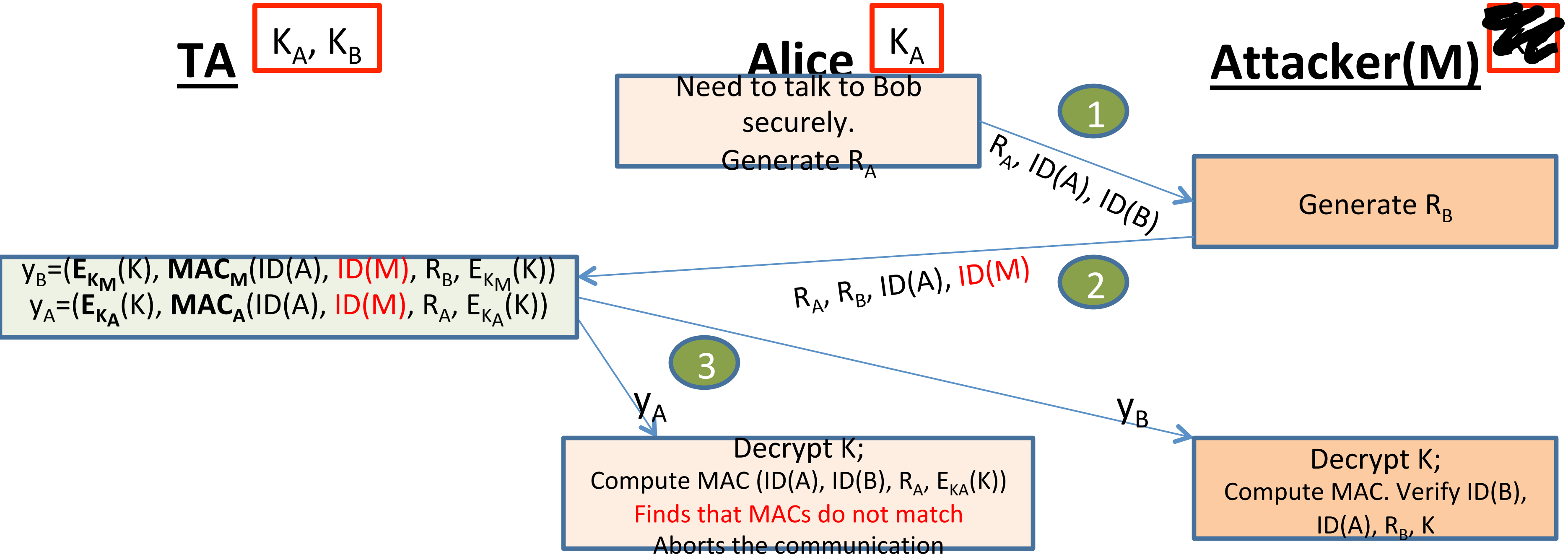
Decrypt K; Compute MAC. Verify ID(B), ID(A), R_A, K

Decrypt K; Compute MAC. Verify ID(B), ID(A), R_B, K

BR Scheme Analysis : When Attacker is Active and Impersonates Bob

Attacker Sends ID(M) instead of ID(B) to TA

Alice finds that the MAC she computes does not match the MAC sent by the TA



TA K_A, K_B

Alice K_A

Attacker(M) [redacted]

Need to talk to Bob securely.
Generate R_A

① R_A, ID(A), ID(B)

Generate R_B

② R_A, R_B, ID(A), ID(M)

y_B=($\mathbf{E}_{\mathbf{K_M}}$(K), $\mathbf{MAC_M}$(ID(A), ID(M), R_B, E_{K_M}(K))
y_A=($\mathbf{E}_{\mathbf{K_A}}$(K), $\mathbf{MAC_A}$(ID(A), ID(M), R_A, E_{K_A}(K))

③ y_A y_B

Decrypt K;
Compute MAC (ID(A), ID(B), R_A, E_{KA}(K))
Finds that MACs do not match
Aborts the communication

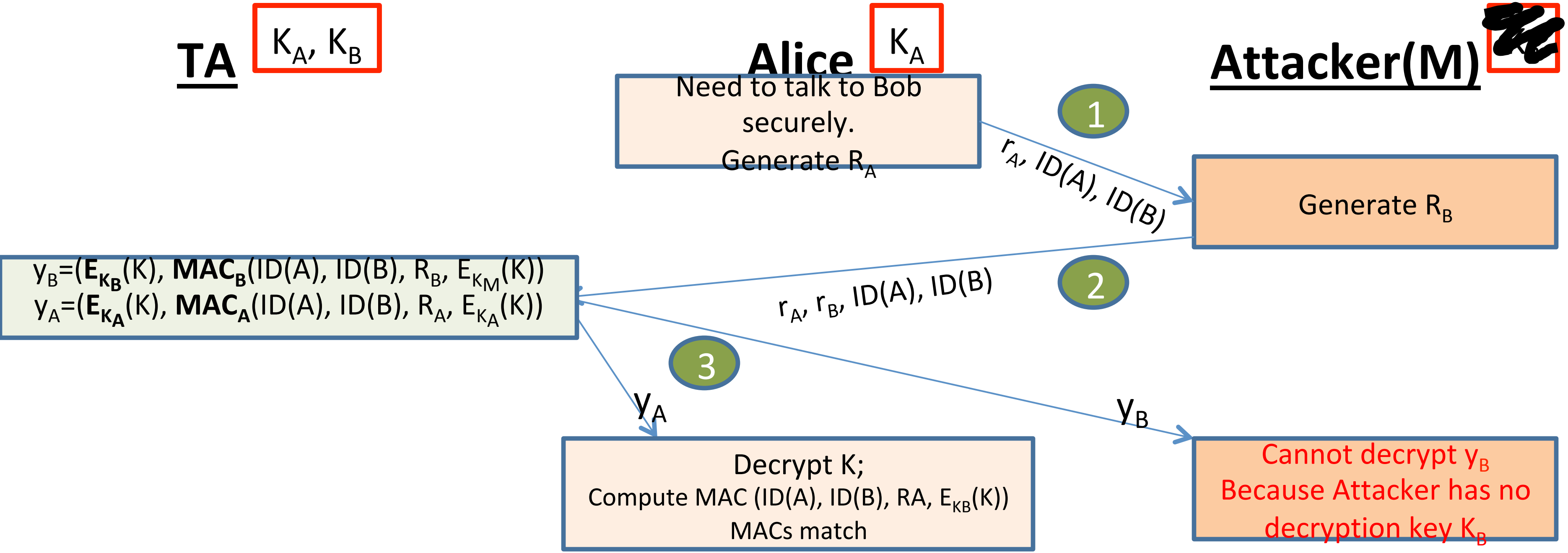
Decrypt K;
Compute MAC. Verify ID(B), ID(A), R_B, K

BR Scheme Analysis : When Attacker is Active and Impersonates Bob

Attacker Sends ID(B) as usual

Attacker cannot decrypt y_B because she does not have the decryption key KB

Messages sent from Alice encrypted with K, cannot be decrypted by the attacker



TA K_A, K_B

Alice K_A

Attacker(M) [redacted]

Need to talk to Bob securely.
Generate R_A

① r_A, ID(A), ID(B)

Generate R_B

② r_A, r_B, ID(A), ID(B)

y_B=(**E_{K_B}**(K), **MAC_B**(ID(A), ID(B), R_B, E_{K_M}(K))
y_A=(**E_{K_A}**(K), **MAC_A**(ID(A), ID(B), R_A, E_{K_A}(K))

③ y_A y_B

Decrypt K;
Compute MAC (ID(A), ID(B), RA, E_{KB}(K))
MACs match

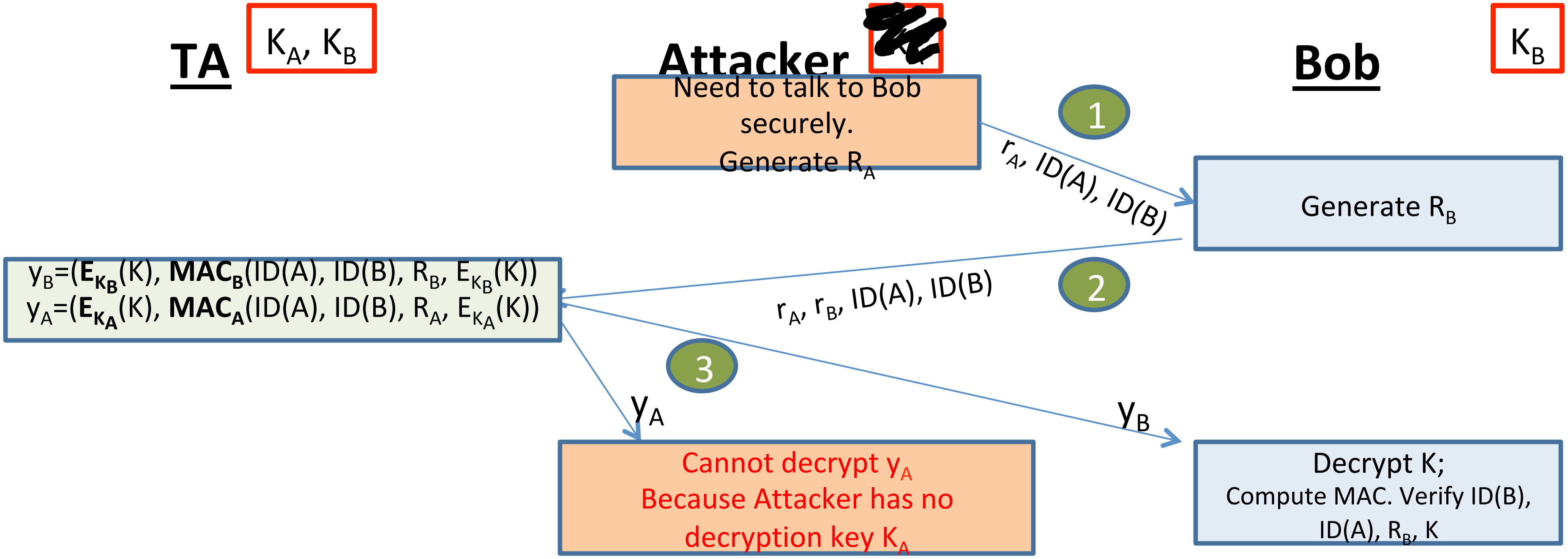
Cannot decrypt y_B
Because Attacker has no decryption key K_B

BR Scheme Analysis : When Attacker is Active and Impersonates Alice

Attacker sends ID(A), r_A to Bob

Attacker cannot decrypt y_A because she does not have the decryption key K_A

Messages sent from Bob encrypted with K, cannot be decrypted by the attacker



TA K_A, K_B

Attacker [redacted]

Bob K_B

Need to talk to Bob securely.
Generate R_A

① r_A, ID(A), ID(B)

Generate R_B

② r_A, r_B, ID(A), ID(B)

y_B=($\mathbf{E}_{\mathbf{K_B}}$(K), $\mathbf{MAC_B}$(ID(A), ID(B), R_B, E_{K_B}(K))

y_A=($\mathbf{E}_{\mathbf{K_A}}$(K), $\mathbf{MAC_A}$(ID(A), ID(B), R_A, E_{K_A}(K))

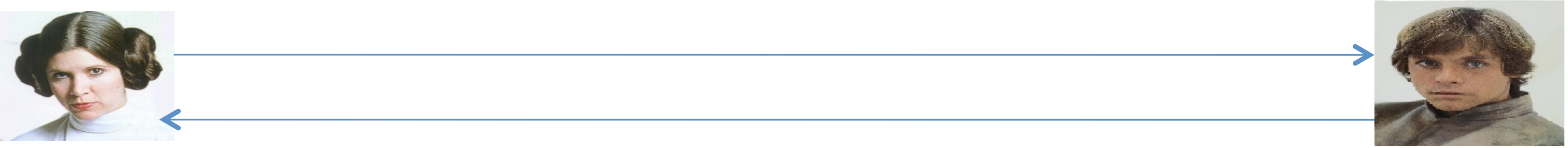
③ y_A y_B

Cannot decrypt y_A
Because Attacker has no decryption key K_A

Decrypt K;
Compute MAC. Verify ID(B), ID(A), R_B, K

Key Agreement Schemes

How does Alice and Bob agree upon a secret key without active use of a TA?



- Users use a public key algorithm
  - The secret key agreed on is a function of
    - Alices' public and private keys
    - Bob's public and private keys

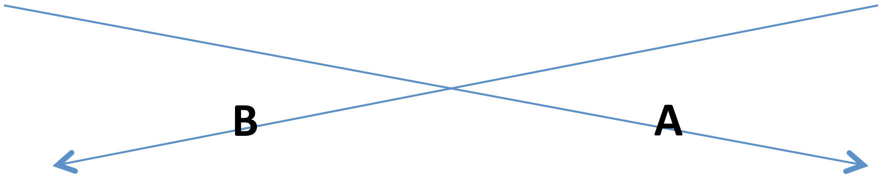
Recall…

Diffie Hellman Key Exchange

Alice and Bob agree upon a prime **p** and a generator **g**.
This is public information

choose a secret **a**
compute $\mathbf{A = g^a \bmod p}$

choose a secret **b**
compute $\mathbf{B = g^b \bmod p}$

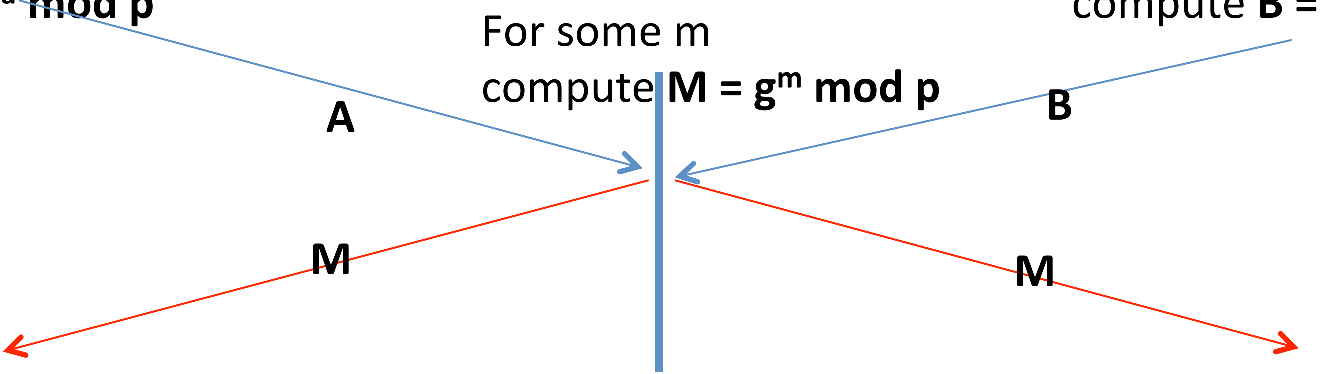


Compute K = $\mathbf{B^a \bmod p}$

Compute K = $\mathbf{A^b \bmod p}$

$$\mathbf{A^b \bmod p = (g^a)^b \bmod p = (g^b)^a \bmod p = B^a \bmod p}$$

Diffie Hellman (Man in the Middle Attack)



choose a secret **a**
compute **A = g^a mod p**

For some m
compute **M = g^m mod p**

choose a secret **b**
compute **B = g^b mod p**

Compute
K_a = **M^a mod p**

Compute
K_a = **A^m mod p**
K_b = **B^m mod p**

Compute
K_b = **M^b mod p**

Diffie Hellman (Man in the Middle Attack)

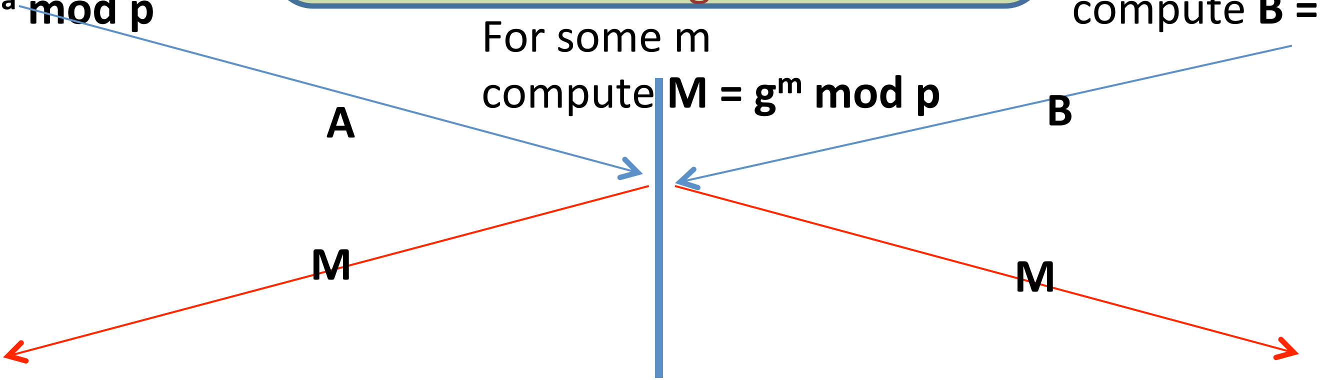
What's missing is Authentication!

Alice and Bob need to authenticate each other before exchanging messages

choose a secret **a**
compute **$A = g^a \bmod p$**

choose a secret **b**
compute **$B = g^b \bmod p$**

For some m
compute **$M = g^m \bmod p$**



Compute
K_a = **$M^a \bmod p$**

Compute
K_a = **$A^m \bmod p$**
K_b = **$B^m \bmod p$**

Compute
K_b = **$M^b \bmod p$**