Security Protocols

Comments

- All the login protocol we saw are subject to man-in-the-middle attacks
- All protocols were entities do not share some sort of secret beforehand, they are subject to this type of attack

Authenticated D-H

A and B shared a pre-arranged secret S

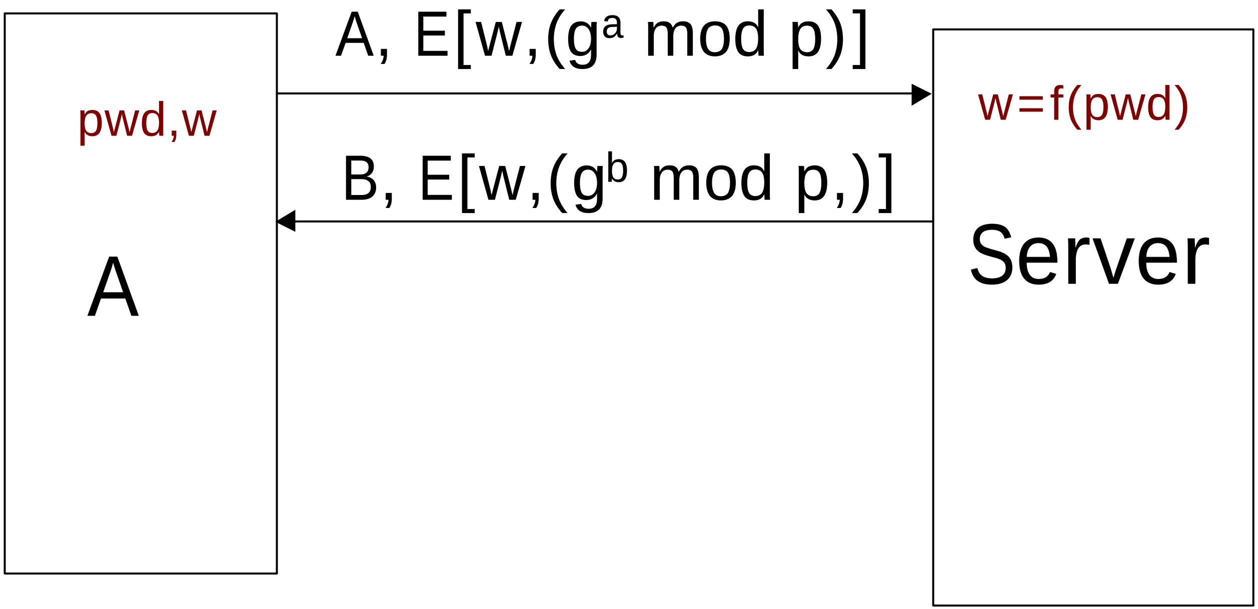
$A \rightarrow B$: $g^x \bmod n$ g^{xy}

$B \rightarrow A$: $g^y \bmod n$ g^{xy}

$A \rightarrow B$: $E[S,(A,g^{xy})]$

$B \rightarrow A$: $E[S,(B,g^{xy})]$

EKE=Encrypted Key Exchange



A, E[w,(g^a mod p)]

pwd,w

w=f(pwd)

B, E[w,(g^b mod p,)]

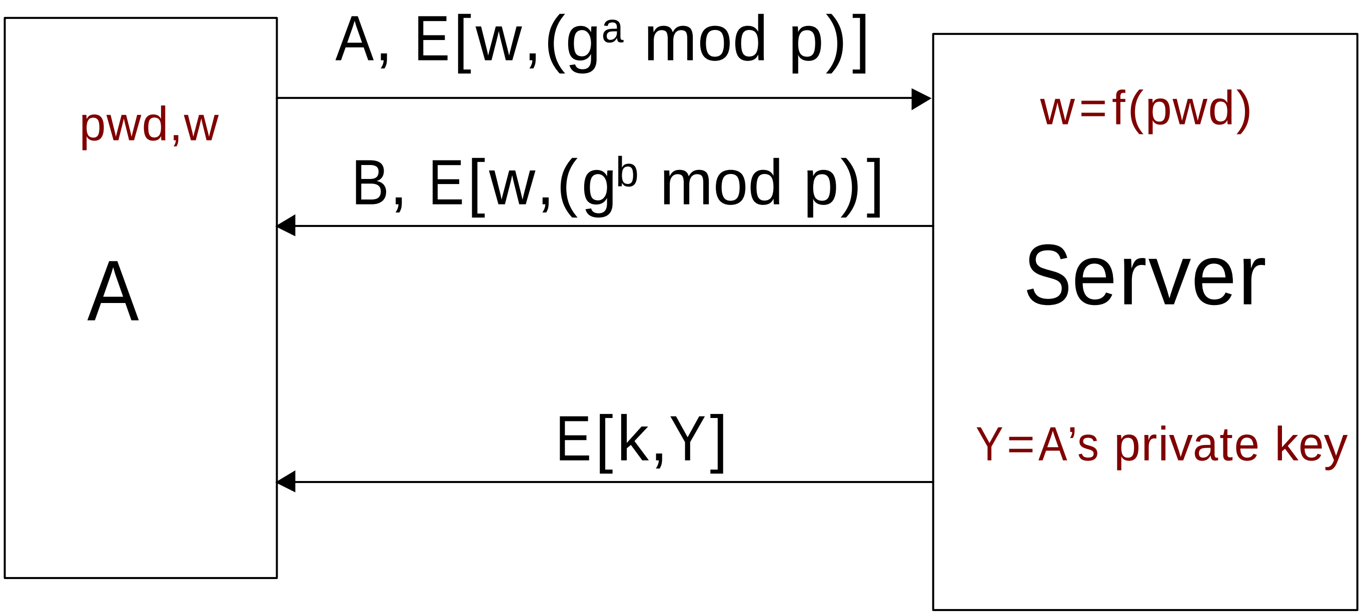
A

Server

k=g^{ab} mod p

Encryption protects from dictionary attacks
Mutual authentication

Secure Credentials Download Protocol



A, E[w,(g^a mod p)]

pwd,w

A

B, E[w,(g^b mod p)]

w=f(pwd)

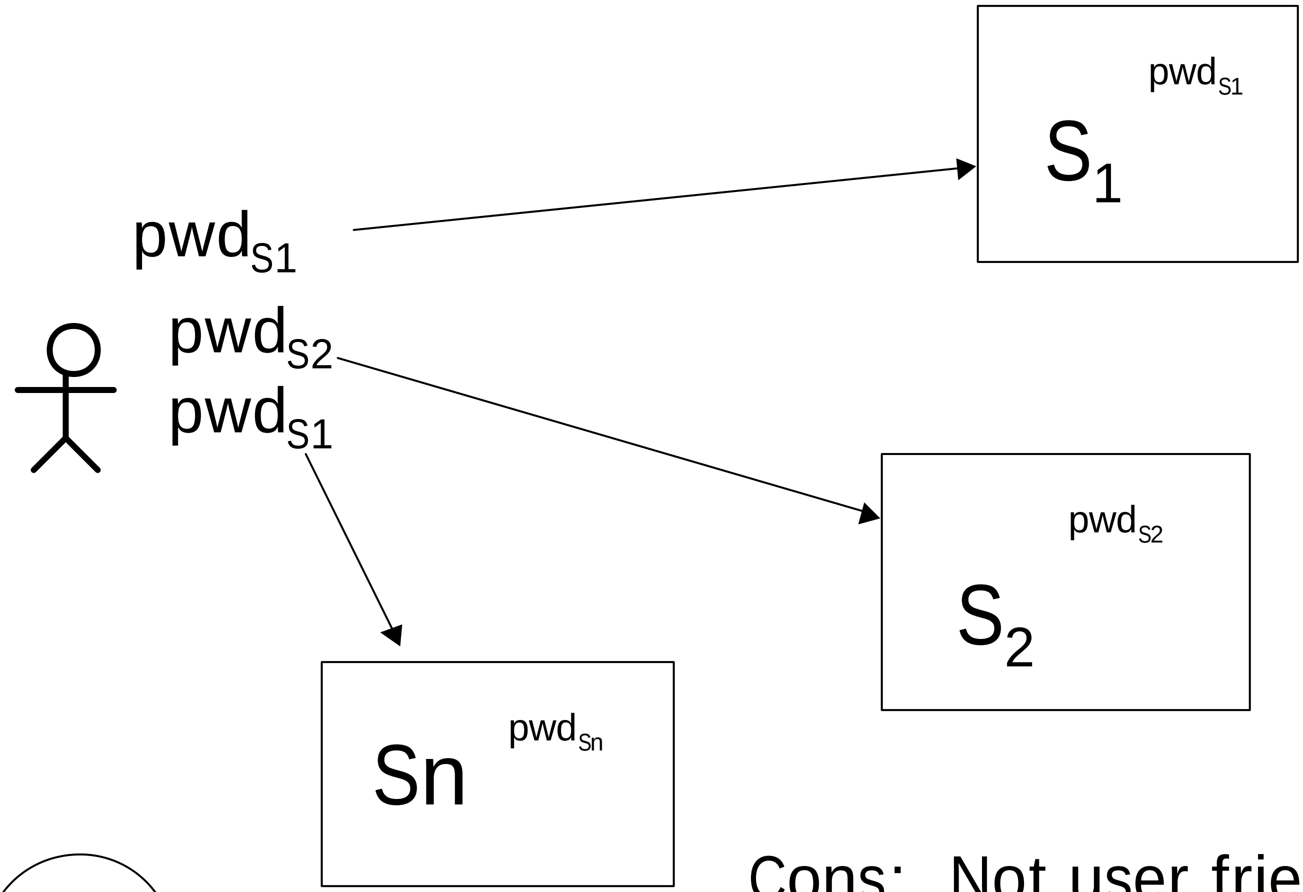
Server

E[k,Y]

Y=A’s private key

$k = g^{ab} \bmod p$

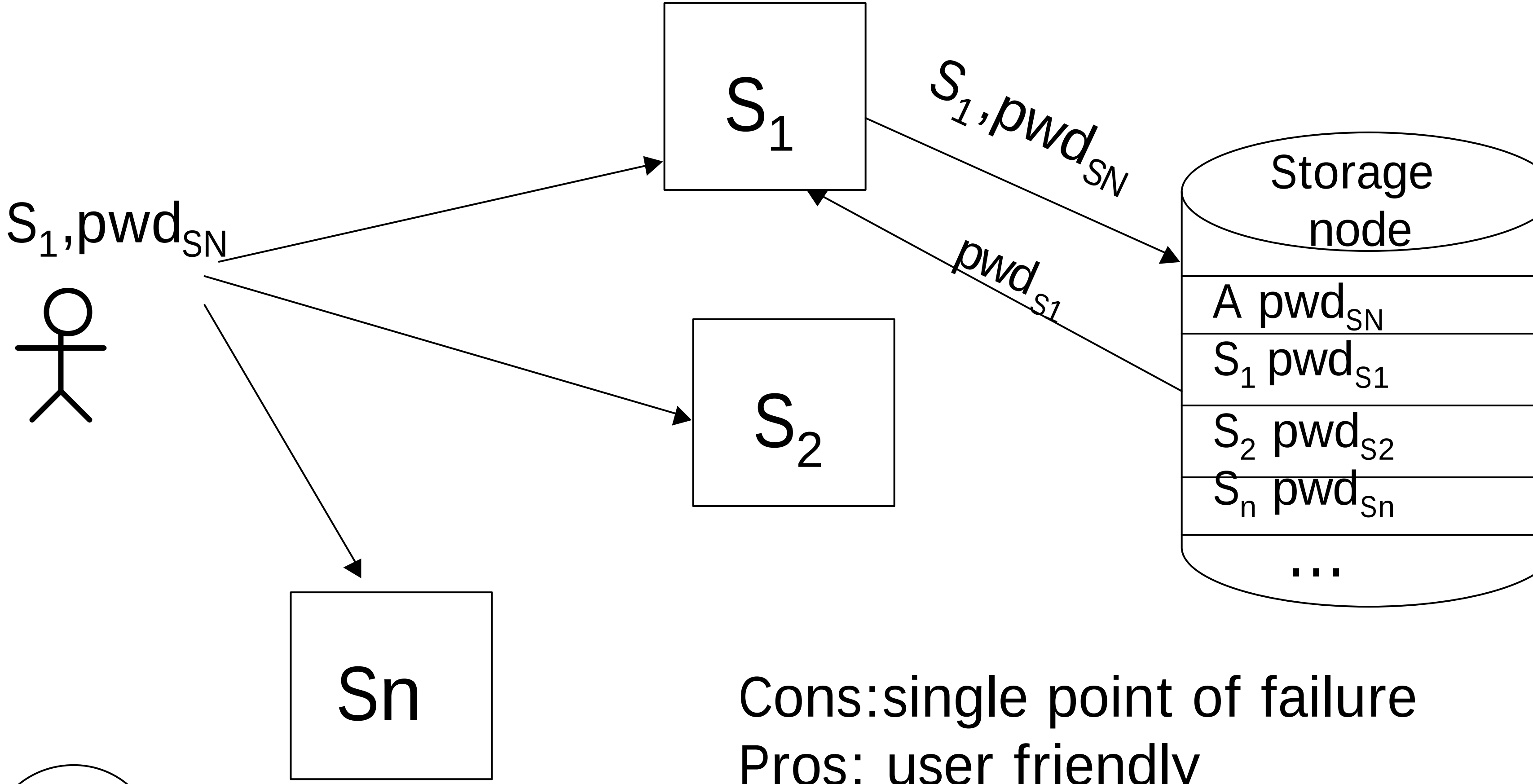
Server specific



Cons: Not user friendly

Pros: Confinement of damage

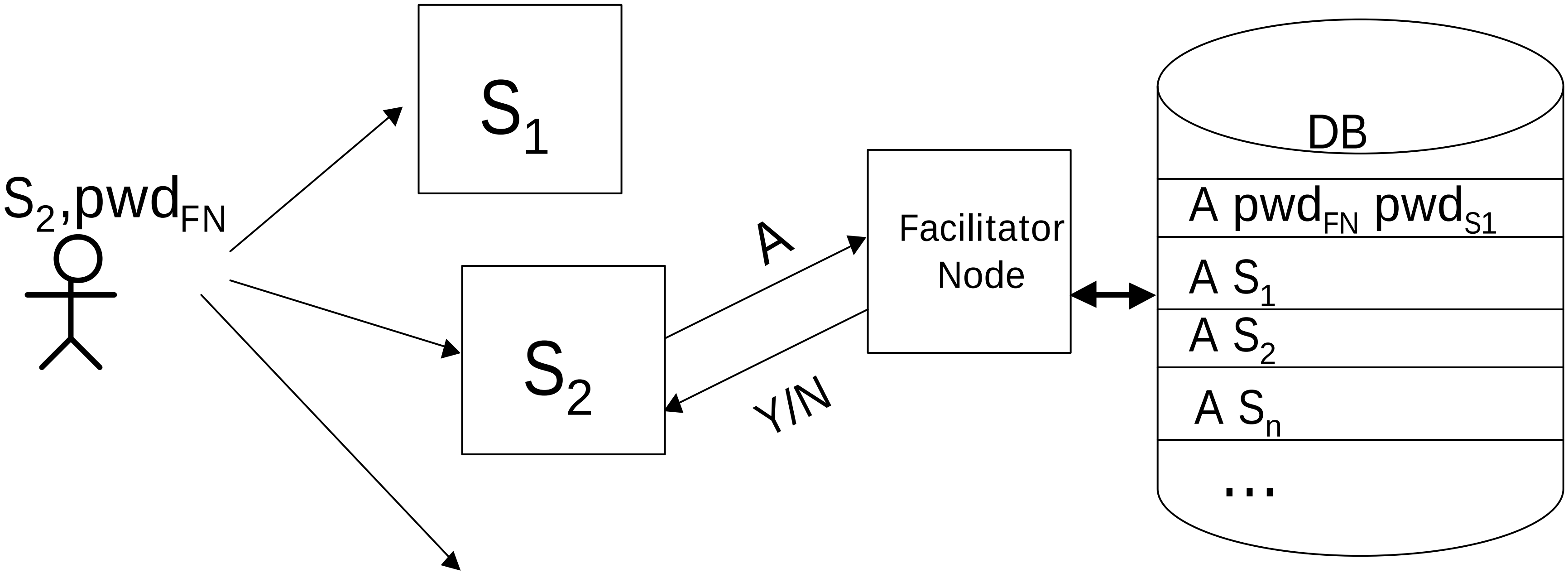
Storage Node



Cons:single point of failure
Pros: user friendly
centralized updates

Storage node has to authenticate to servers

Facilitator Node



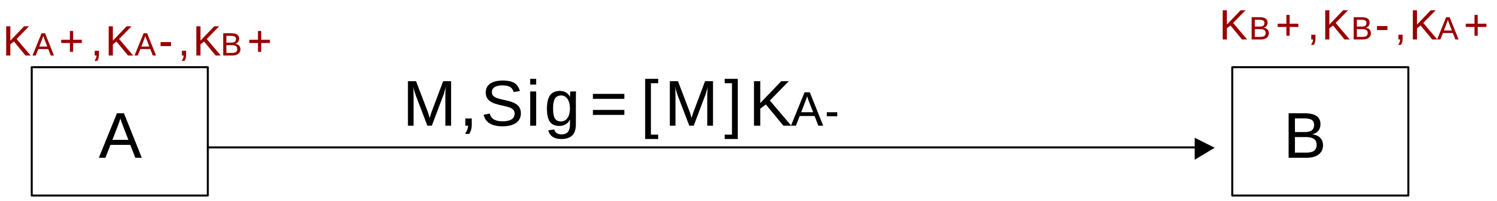
Cons: single point of failure and
bottleneck
Pros: user friendly
centralized update

Same requirements as storage node

Digital Signature

Assumptions:

K_X- :private key generated and owned only by X
K_X+ :public key public known by everybody



K_A+, K_A-, K_B+

A

$M, Sig = [M]K_{A-}$

B

K_B+, K_B-, K_A+

signing

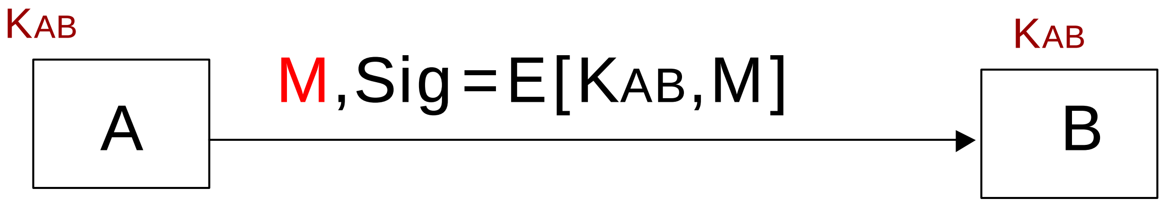
$$\{M' = [[M]K_{A-}]K_{A+} \overset{?}{=} M\} = T/F$$

K_X- :signing key

K_X+ :verification key

verification

Signature with SK?



K_{AB}

A

$M, Sig = E[K_{AB}, M]$

K_{AB}

B

$$Sig \overset{?}{=} E[K_{AB}, M]$$

– Since both A and B share K_{AB} it is impossible for a third party to univocally determine which party generated *Sig*. Both of them could have done it!

– With AK private keys do not need to be transmitted over the network

Challenge/response protocol

Host → A: random string=r

A → Host: Sig[r]K_{A-}

Host computes Ver(Sig[r]K_{A-},K_{A+}) = True/False

– Subject to some attack (context-switching)

Blind Signatures

e=Bob’s pub key d=Bob’s priv key public modulus n

- Alice chooses random $1<k<n$, then blinds m by computing $t=mk^e$ modn, then send it to Bob
- Bob signs t, $t^d=(mk^e)^d$ modn = $(m^d)k$ modn and send it to Alice
- Alice unblind t^d by computing

  $s = t^d/k$ modn $\rightarrow$ $s=m^d$ modn

$t^d \equiv (mk^e)^d \equiv m^d k (\text{mod} n) \rightarrow t^d/k = m^d k/k \equiv m^d (\text{mod} n)$

$\text{Sig}[mk^e]K_B\text{-} \rightarrow \text{Sig}[m]K_B\text{-}$

Authentication protocols

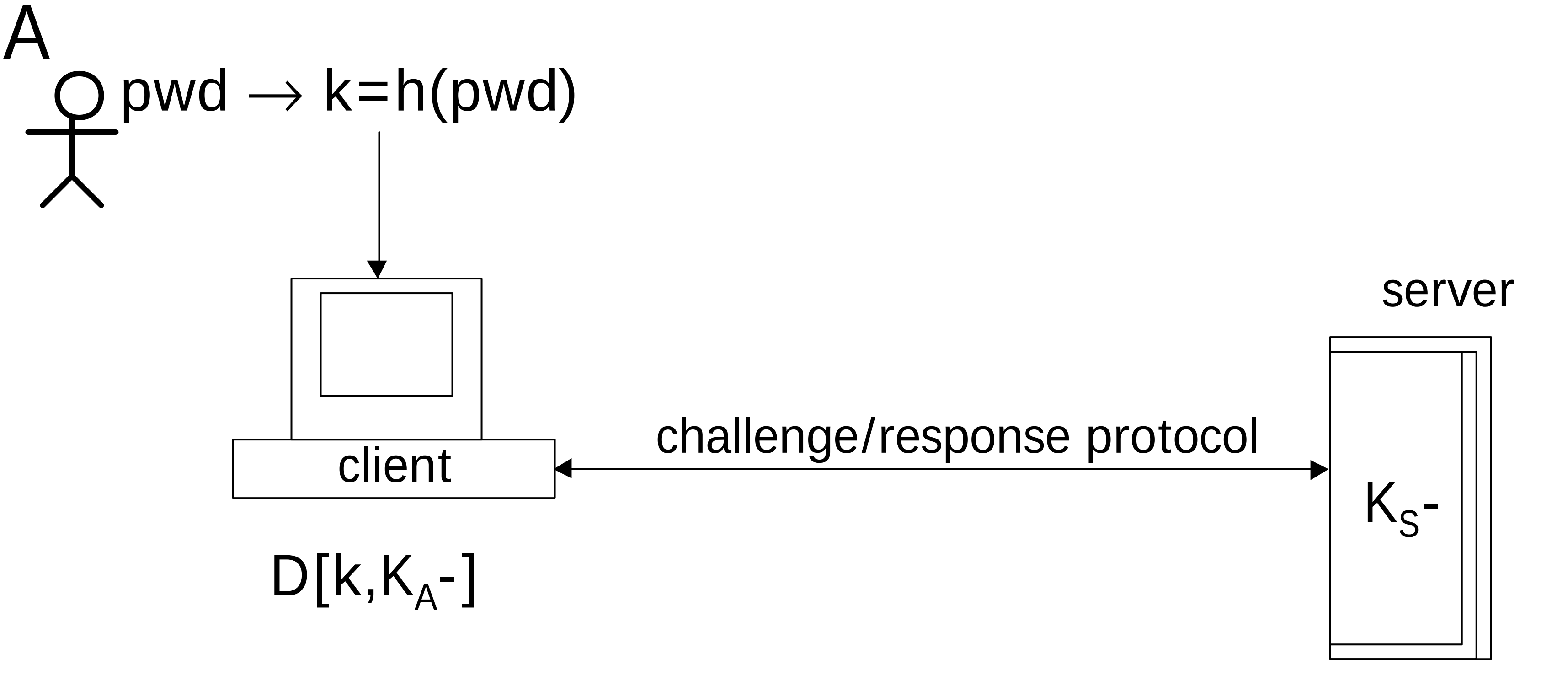
Lesson learned:

Never sign unknown strings

(i.e. Encrypted messages)

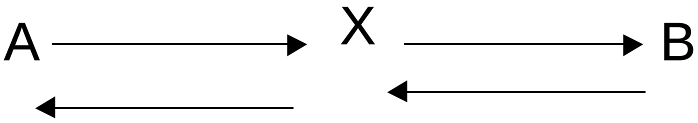
Always, first sign then encrypt!

Login authentication chain

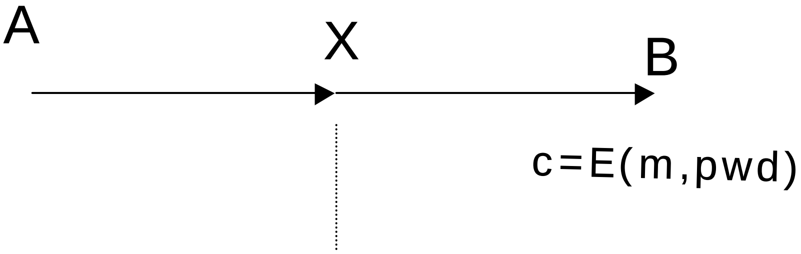


Protocols attacks

Man-in-the-middle



Dictionary

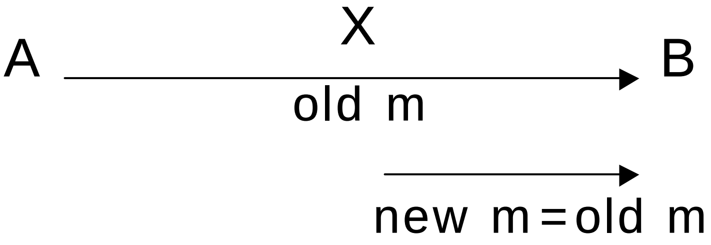


$c=E(m,pwd)$

$c \overset{?}{=} \theta[pwd_i]:E(m,pwd_i)$

Protocols attacks

Replay



A

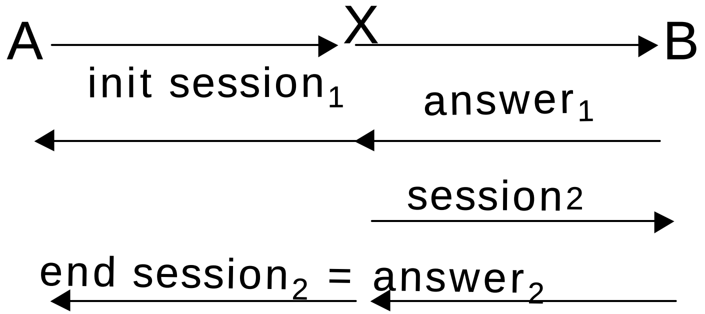
X

B

old m

new m=old m

Reflection



A

X

B

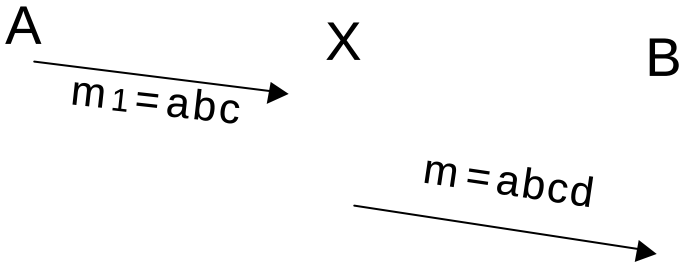
init session$_1$

answer$_1$

session2

end session$_2$ = answer$_2$

Padding



A

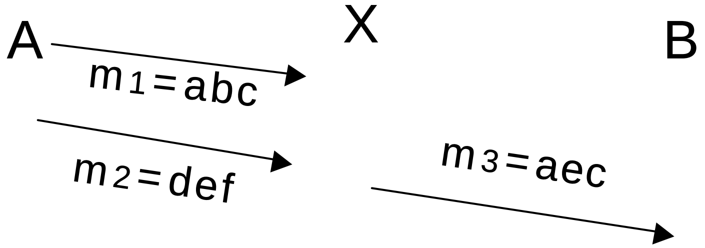
X

B

m_1=abc

m=abcd

Cut&Paste



A

X

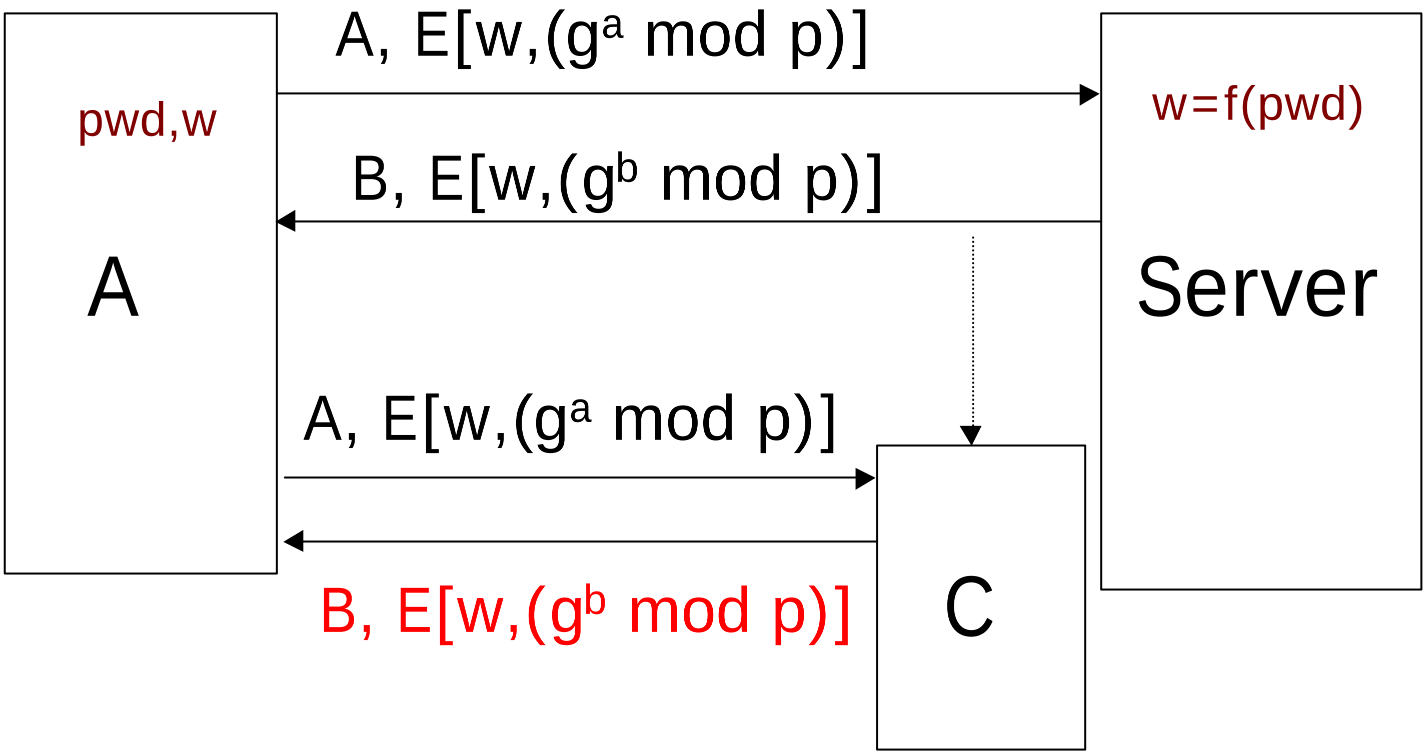
B

m_1=abc

m_2=def

m_3=aec

Replay attack



A, E[w,(g^a mod p)]

pwd,w

w=f(pwd)

B, E[w,(g^b mod p)]

A

Server

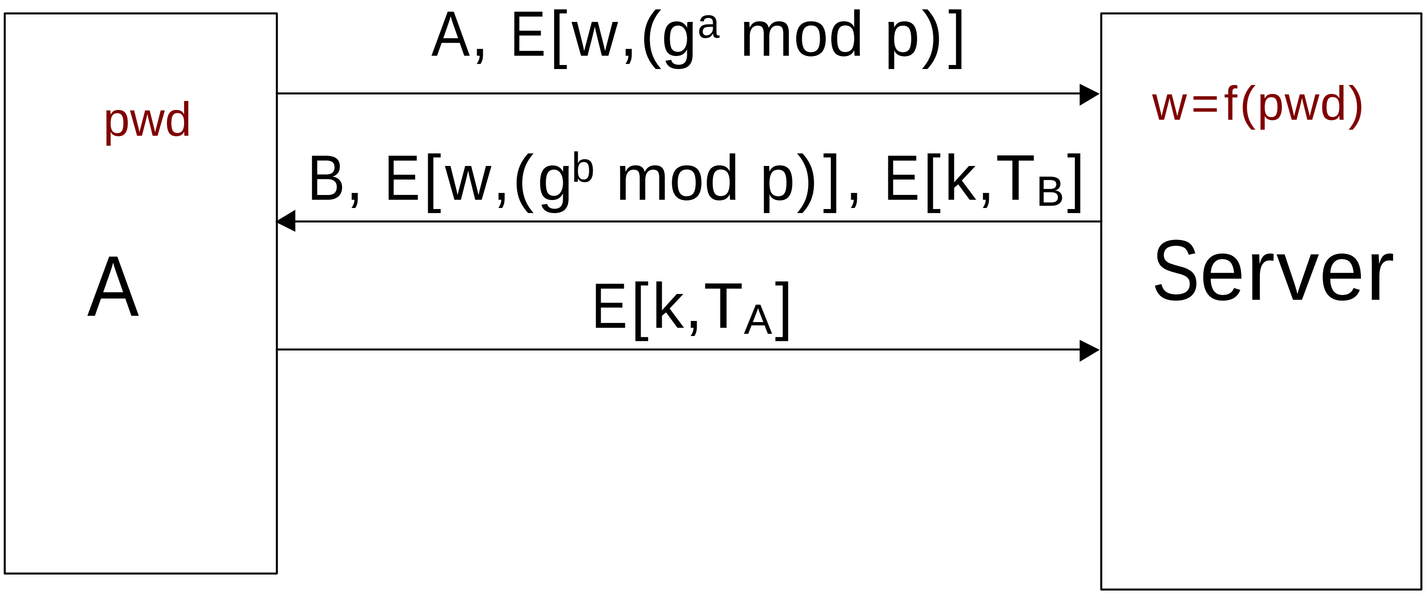
A, E[w,(g^a mod p)]

B, E[w,(g^b mod p)]

C

C can complete a fake run of the protocol

Timestamps



A, $E[w,(g^a \bmod p)]$

pwd

$w=f(pwd)$

B, $E[w,(g^b \bmod p)]$, $E[k,T_B]$

A

Server

$E[k,T_A]$

$k=g^{ab} \bmod p$

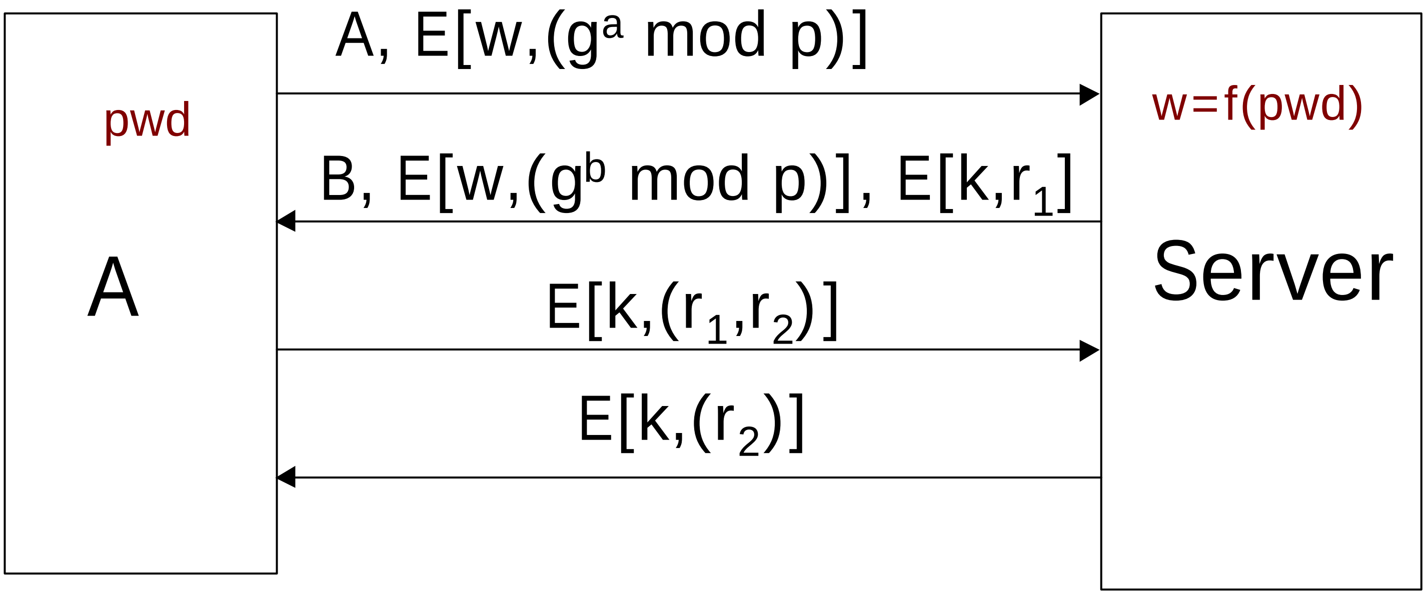
Solutions based on timestamps

Pros: no need of additional msg

Cons:

- Clock syncronization not trivial in large distributed system
- If B’s clock behind attack is still possible
- If A’s clock ahead attack is still possible

Nonces



pwd

A

A, $E[w,(g^a \bmod p)]$

B, $E[w,(g^b \bmod p)]$, $E[k,r_1]$

$E[k,(r_1,r_2)]$

$E[k,(r_2)]$

$w=f(pwd)$

Server

$k=g^{ab} \bmod p$

Solutions based on nonces

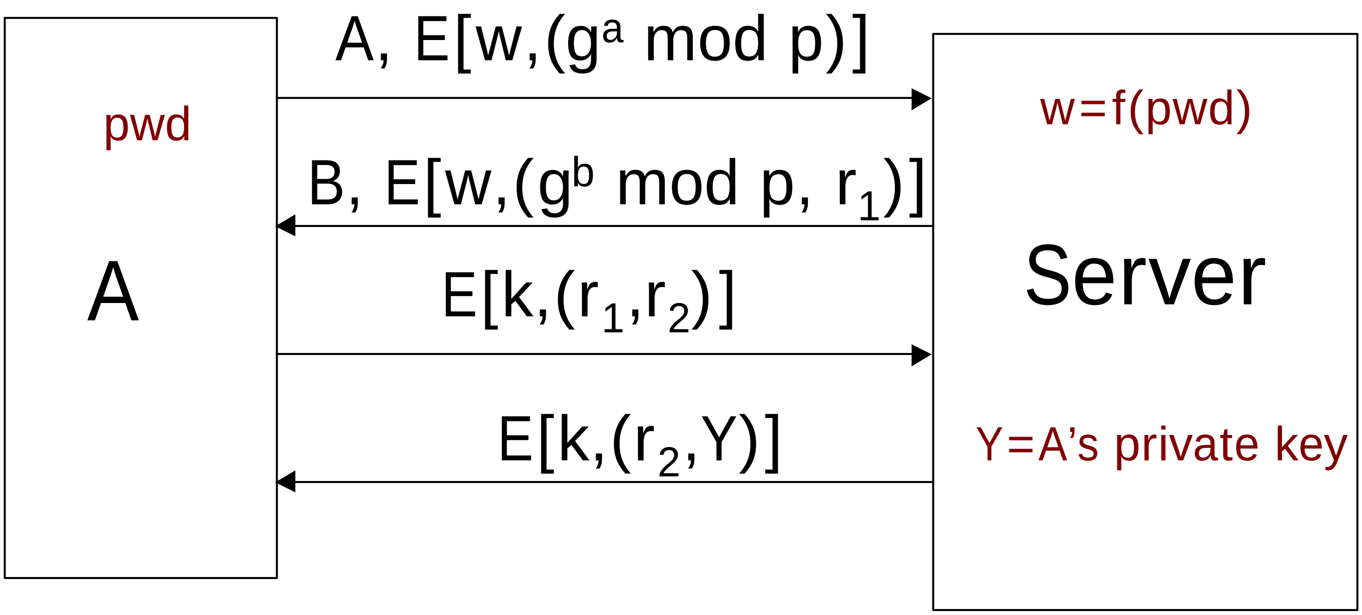
Nonce: an unpredictable and unique number

Pros: fix the problem

Cons:

- – Need to keep state to store all the used nonces. Senquence numbers reduce the state but their initial value need to be kept secret

Secure Credentials Download Protocol



pwd

A

A, E[w,(g^a mod p)]

B, E[w,(g^b mod p, r_1)]

E[k,(r_1,r_2)]

E[k,(r_2,Y)]

w=f(pwd)

Server

Y=A’s private key

k=g^{ab} mod p

Reflection attack

$A \rightarrow B:\ A, R_2$

$B \rightarrow A:\ R_1, [R_2]K_{AB}$

$A \rightarrow B:\ [R_1]\ K_{AB}$

$C(A) \rightarrow B:\ A, R_2$

$B \rightarrow C(A):\ R_1, [R_2]K_{AB}$

$C(A) \rightarrow B:\ A, R_1$

$B \rightarrow C(A):\ R_3, [R_1]K_{AB}$

$C(A) \rightarrow B:\ [R_1]K_{AB}$

Reflection attack

Solution

$A \rightarrow B$: A, R_2

$B \rightarrow A$: $R_1, [B, R_2]K_{AB}$

$A \rightarrow B$: $[A, R_1]\ K_{AB}$

$C(A) \rightarrow B$: A, R_2

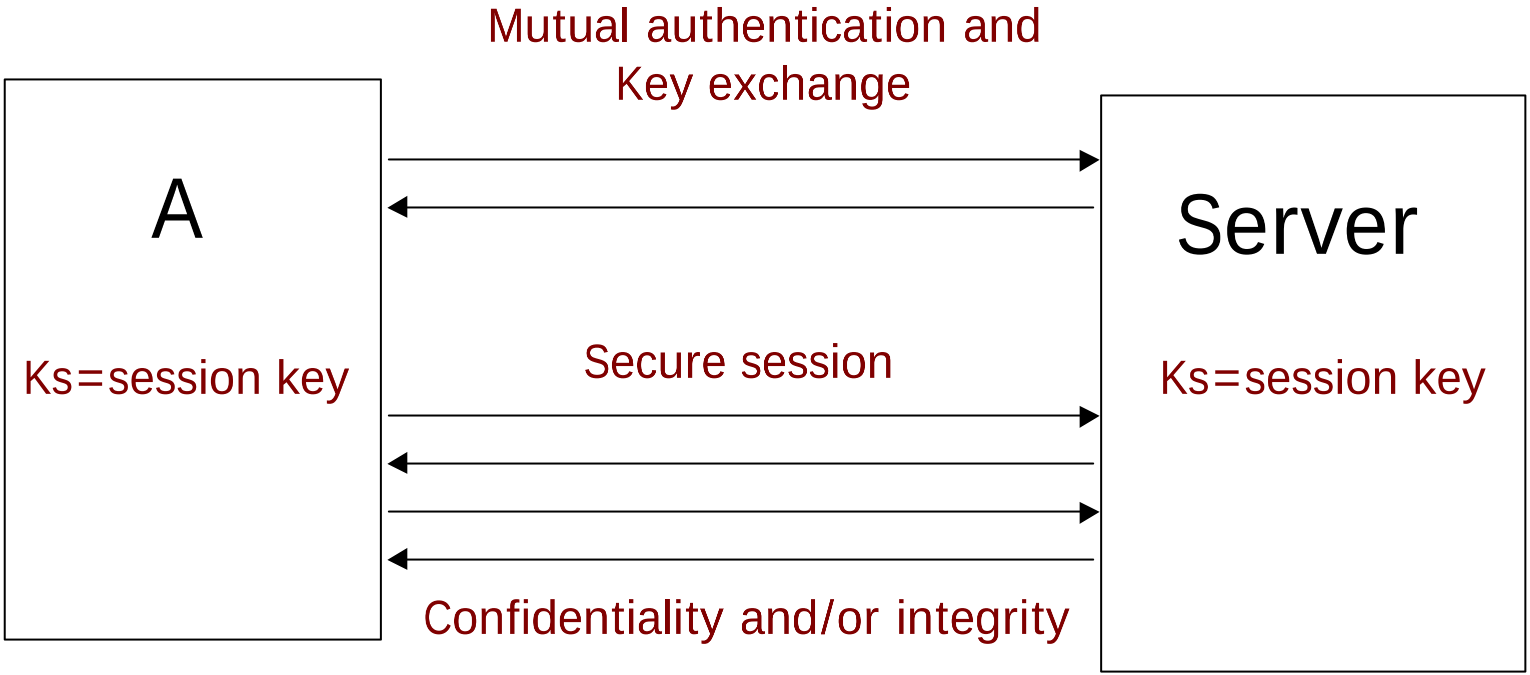
$B \rightarrow C(A)$: $R_1, [B, R_2]K_{AB}$

$C(A) \rightarrow B$: A, R_1

$B \rightarrow C(A)$: $R_3, [B, R_1]K_{AB}$

and not $[A, R_1]K_{AB}$!!

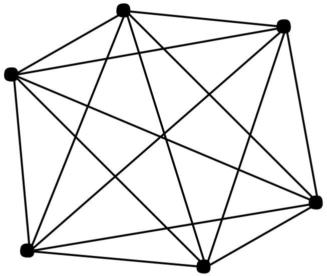
Mutual authentication and secure sessions

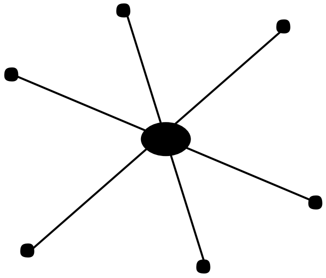


Key Distribution

- **N** users
- Without any server
  - With SK user needs to share N-1 symmetric keys
  - With AK user needs to know N-1 public keys

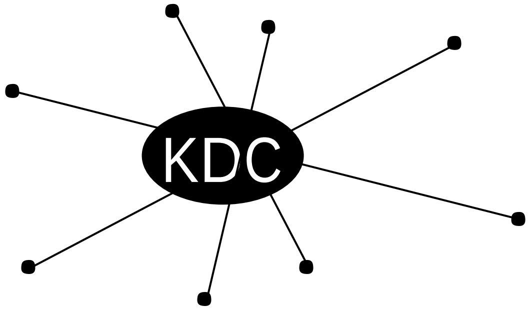
- With an intermediary only 1 key ® scalability



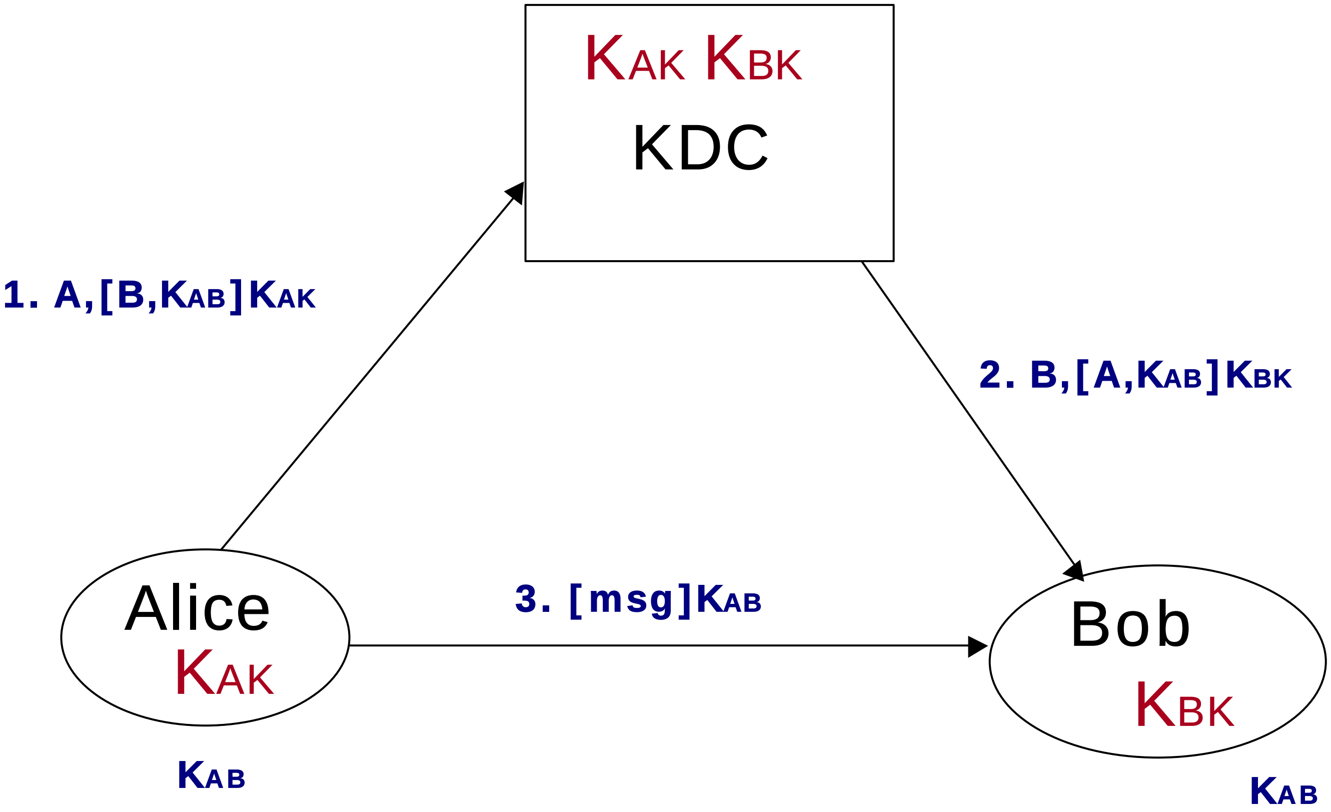


Key Distribution Center

- All users register with the KDC and they share a symmetric key with KDC or register their public key
- KDC centralized server that distributes keys and capable of establishing a secure channel with any registered user



Wide-Mouth Frog



K_{AK} K_{BK}

KDC

1. A,[B,K_{AB}]K_{AK}

2. B,[A,K_{AB}]K_{BK}

3. [msg]K_{AB}

Alice

K_{AK}

K_{AB}

Bob

K_{BK}

K_{AB}

Wide-Mouth Frog

$A \rightarrow KDC$: $A,[B,K_{AB}]K_{AK}$

$KDC \rightarrow B$: $B,[A,K_{AB}]K_{BK}$

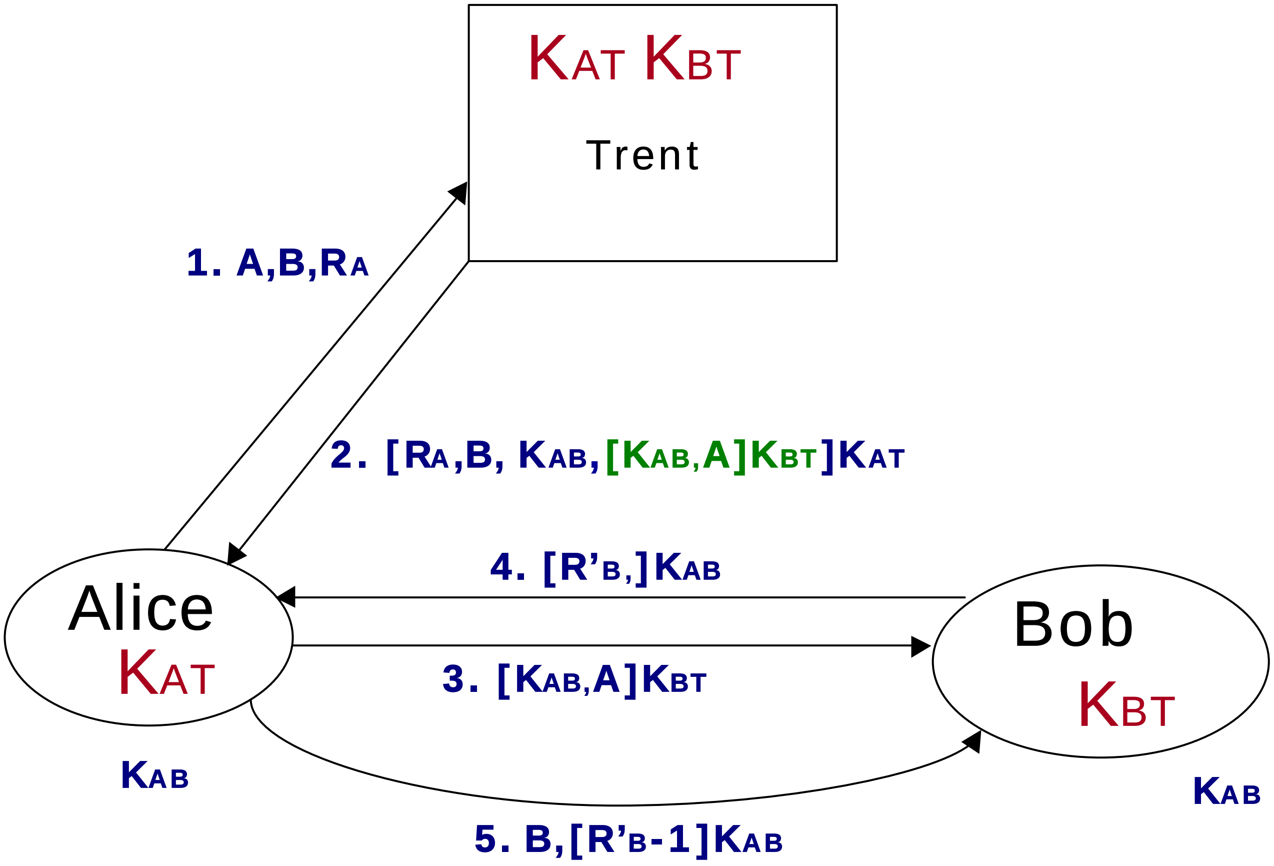
Replay attack

$A \rightarrow KDC$: $A,[B,T_{A},K_{AB}]K_{AK}$

$KDC \rightarrow B$: $B,[A,T_{KDC},K_{AB}]K_{BK}$

Needham-Schroeder

Needham, R. M., and Schroeder, M. D. "*Using encryption for authentication in large networks of computers*". Commun. ACM 21, 12 (Dec. 1978)



K_{AT} K_{BT}

Trent

1. A,B,R_A

2. $[R_A,B, K_{AB},[K_{AB},A]K_{BT}]K_{AT}$

4. $[R'_B,]K_{AB}$

Alice K_{AT}

Bob K_{BT}

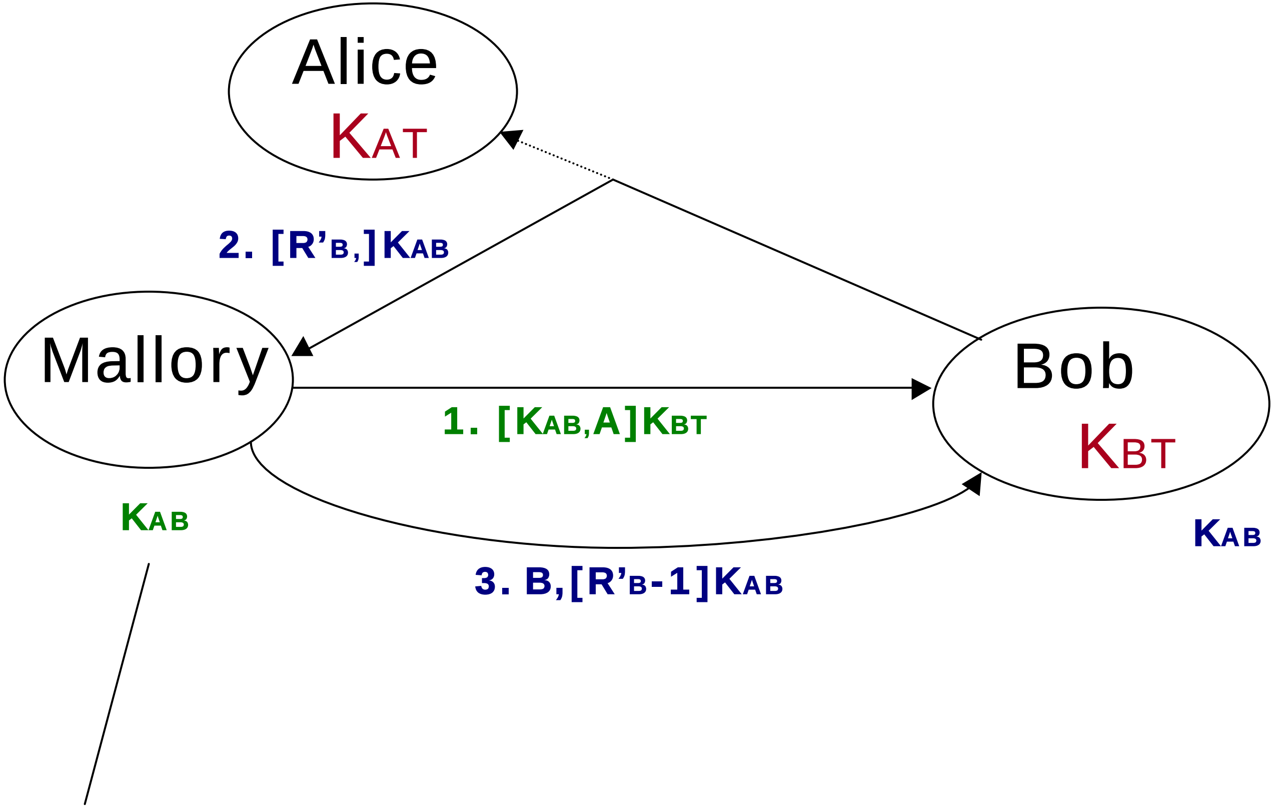
3. $[K_{AB},A]K_{BT}$

K_{AB}

K_{AB}

5. $B,[R'_B\text{-}1]K_{AB}$

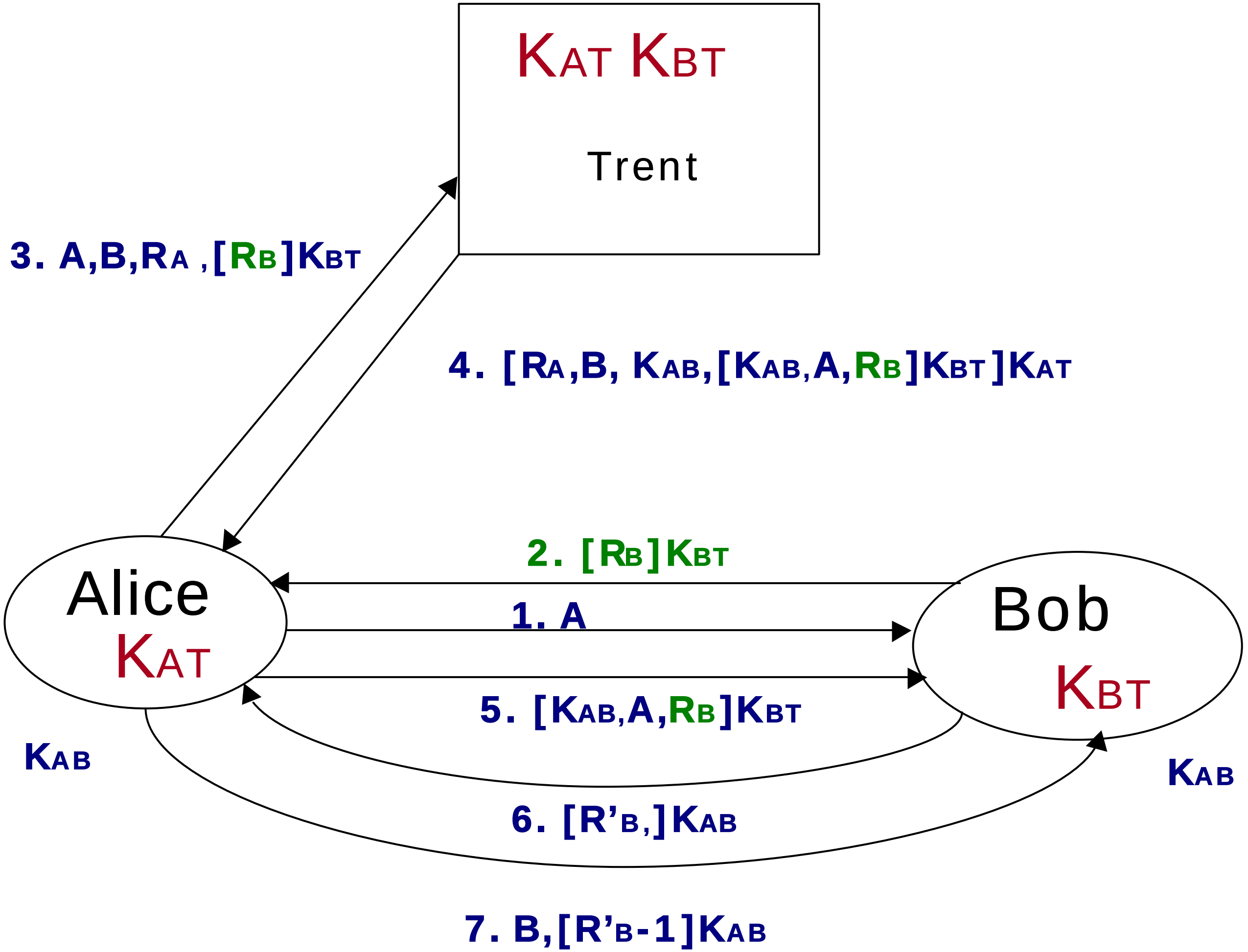
Needham-Schroeder



Old “*stolen*” key

Needham, R. M., Schroeder, M. D.:Authentication Revisited. *ACM Operating Systems Review*, Vol. 21, No. 1, 1987, pp. 993-999.

Needham-Schroeder



K_{AT} K_{BT}

Trent

3. A,B,R_A ,[R_B]K_{BT}

4. [R_A,B, K_{AB},[K_{AB},A,R_B]K_{BT}]K_{AT}

2. [R_B]K_{BT}

1. A

Alice

K_{AT}

Bob

K_{BT}

5. [K_{AB},A,R_B]K_{BT}

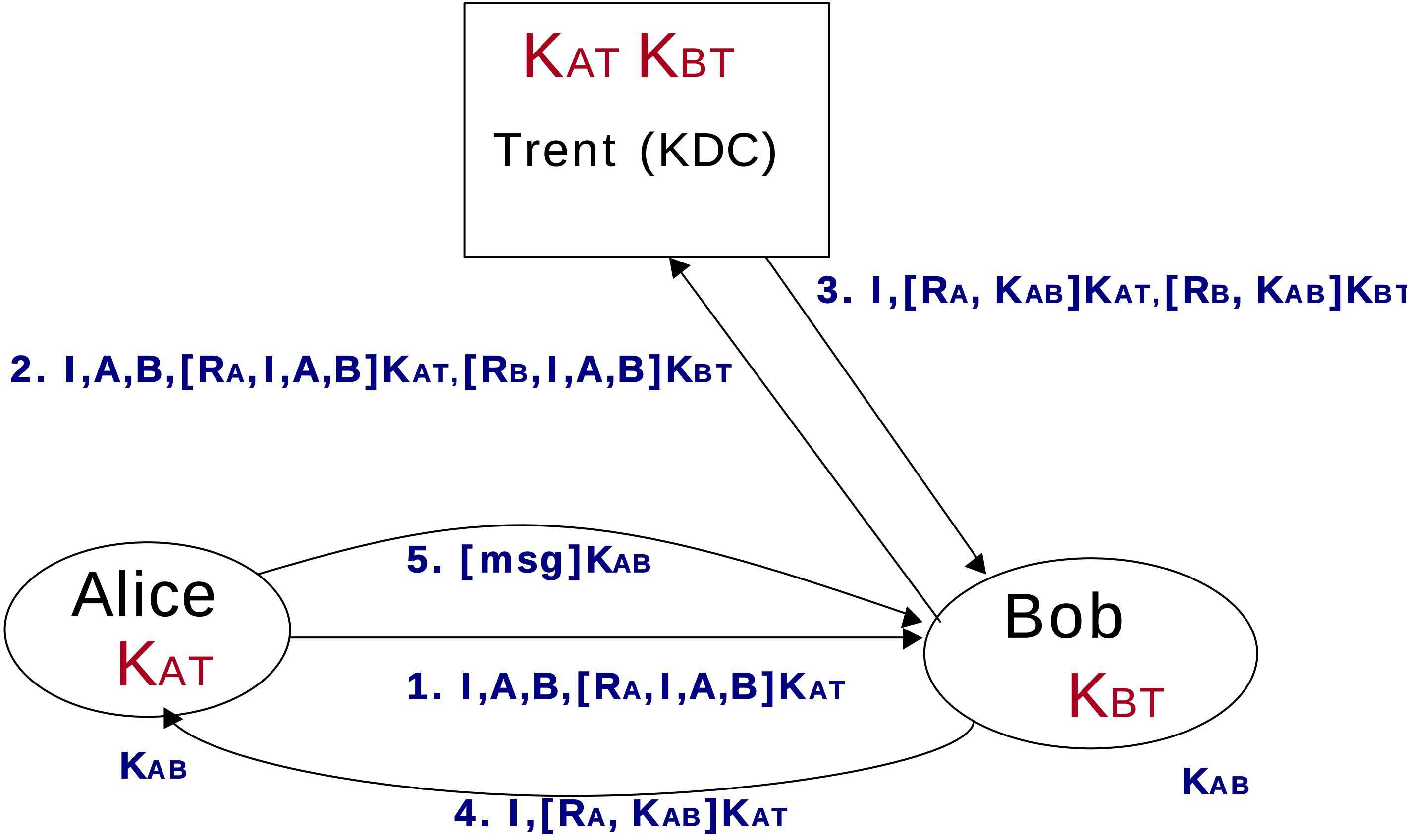
K_{AB}

K_{AB}

6. [$R'_{B,}$]K_{AB}

7. B,[R'_B-1]K_{AB}

Otway-Rees



K_{AT} K_{BT}

Trent (KDC)

3. $I, [R_A, K_{AB}]K_{AT}, [R_B, K_{AB}]K_{BT}$

2. $I, A, B, [R_A, I, A, B]K_{AT}, [R_B, I, A, B]K_{BT}$

Alice

K_{AT}

5. $[msg]K_{AB}$

Bob

K_{BT}

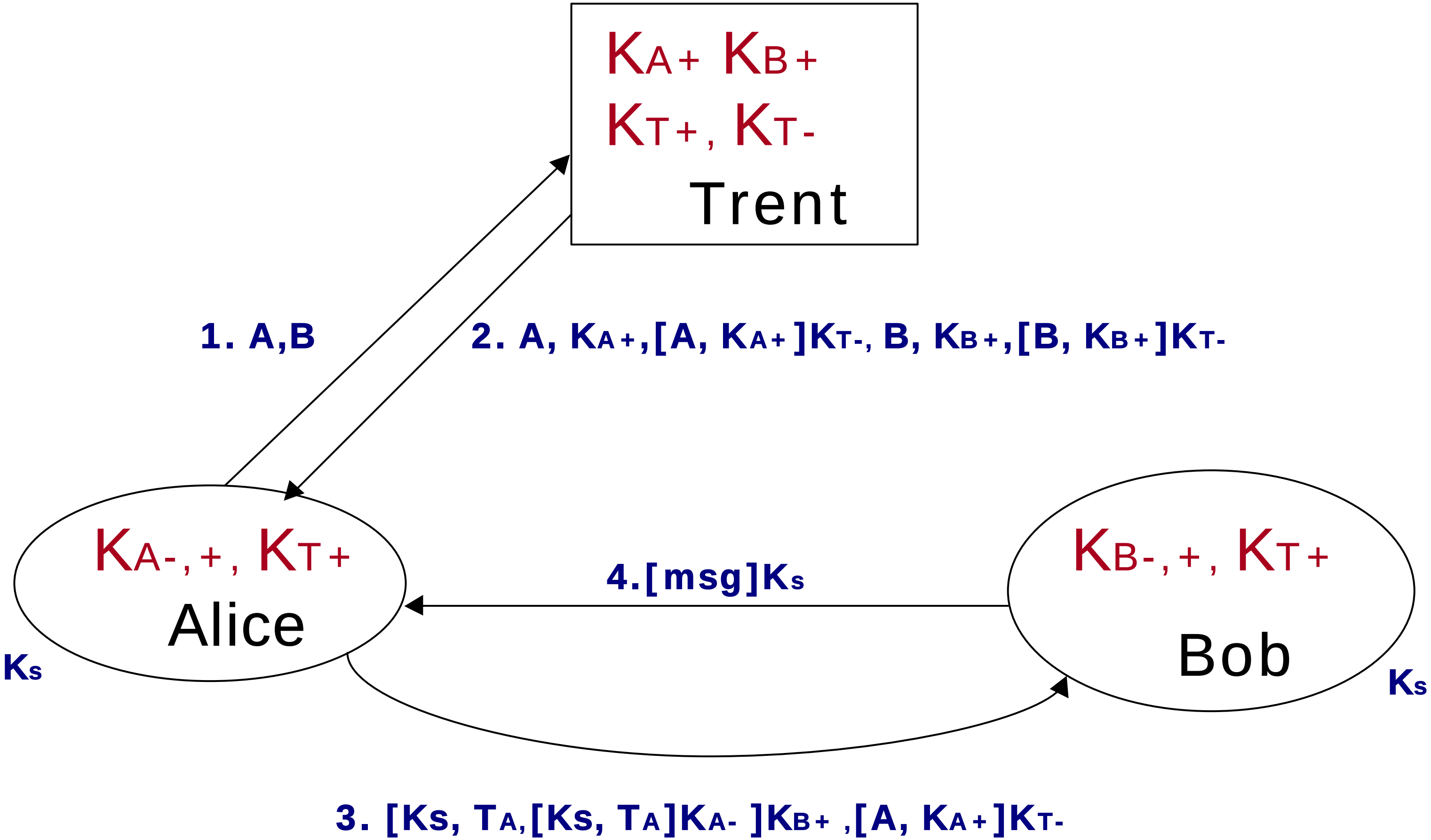
1. $I, A, B, [R_A, I, A, B]K_{AT}$

K_{AB}

K_{AB}

4. $I, [R_A, K_{AB}]K_{AT}$

Dennis-Sacco



K_{A+} K_{B+}

K_{T+}, K_{T-}

Trent

1. A,B

2. A, K_{A+},[A, K_{A+}]K_{T-}, B, K_{B+},[B, K_{B+}]K_{T-}

$K_{A-,+}$, K_{T+}

Alice

K_s

4.[msg]K_s

$K_{B-,+}$, K_{T+}

Bob

K_s

3. [K_s, T_A,[K_s, T_A]K_{A-}]K_{B+} ,[A, K_{A+}]K_{T-}

vulnerable to B masquerading as A....

Dennis-Sacco attack

$A \rightarrow T\text{: } A,B$

$T \rightarrow A\text{: } (A,K_{A+},[A,K_{A+}]K_{T-}),\ (B,K_{B+},[B,K_{B+}]K_{T-})$

$A \rightarrow B\text{: } [Ks,T_{A,}[Ks,T_A]\ K_{A-}]K_{B+,}\ [A,K_{A+}]K_{T-}$

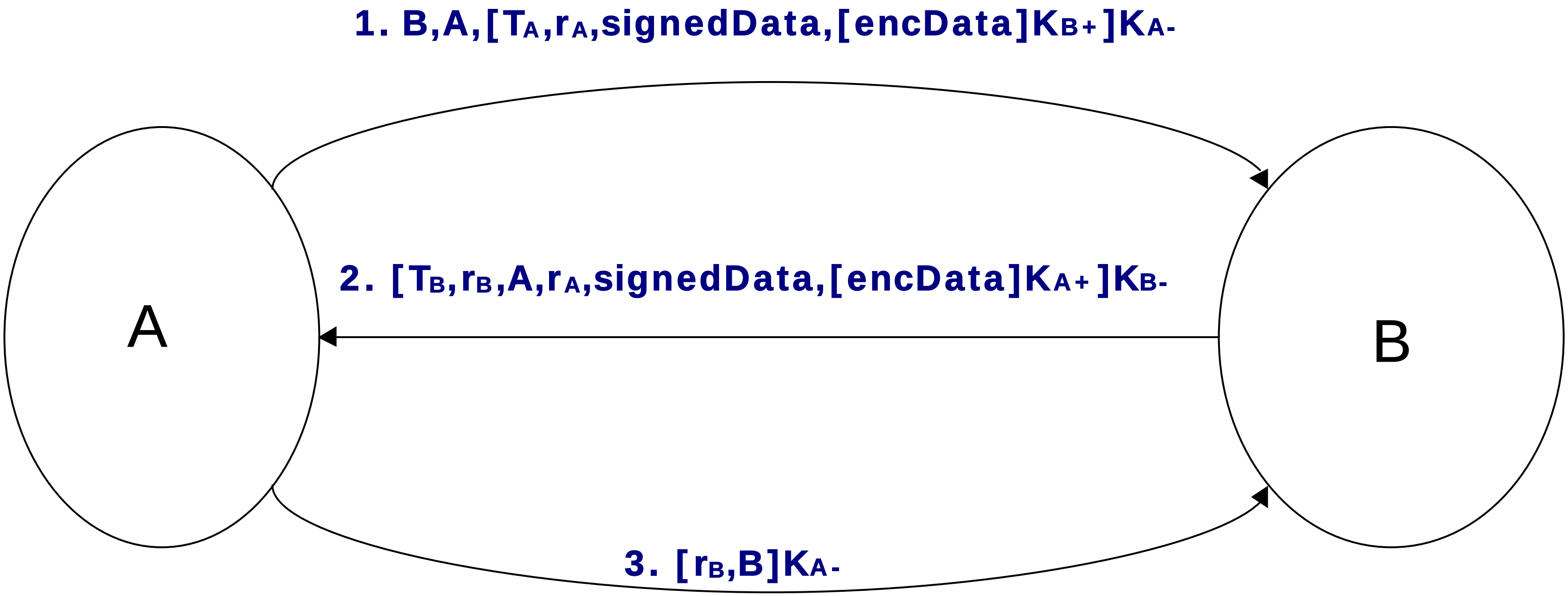
$B \rightarrow T\text{: } B,C$

$T \rightarrow B\text{: } (B,K_{B+},[B,K_{B+}]K_{T-}),\ (C,K_{C+},[C,K_{C+}]K_{T-})$

$B \rightarrow C\text{: } [Ks,T_{A,}[Ks,T_A]\ K_{A-}]K_{C+,}\ [A,K_{A+}]K_{T-}$

solution: include the principals’ names in the session key distribution message

X.509



1. B,A,[T_A,r_A,signedData,[encData]K_{B+}]K_{A-}

2. [T_B,r_B,A,r_A,signedData,[encData]K_{A+}]K_{B-}

3. [r_B,B]K_{A-}

T_x andr_x to prevent replay attacks
Step 3 avoid use of timestamps