第6章 密码协议

南京大学计算机系黄皓教授

2011年11月7日

参考文献

[1] Wenbo Mao著, 现代密码学理论与实践，电子工业出版社，2004年7月。

[2] Bruce Chneier, 应用密码学，机械工业出版社，2000年1月。

[3] Douglas R. Stinson, Cryptography: Theory and Practice, CRC Press LLC, 2002. (冯登国译，密码学原理与实践， 电子工业出版社，2003年2月。)

内容

1. 密码协议的基本概念
2. 一个鉴别与密钥交换的一个研究过程
3. 认证协议
4. 秘密分割/秘密共享
5. 盲签名
6. 一次签名
7. 不可抵赖签名
8. 同时签约

1. 密码协议的基本概念

1. 密码协议的基本概念—协议

- 协议是一系列步骤
- 它包括两方或多方
- 设计它的目的是要完成一项任务

密码协议的基本概念—协议特点

- 协议中的每人都必须了解协议，并且预先知道所要完成的所有步骤。
- 协议中的每人都必须同意遵循它。
- 协议必须是不模糊的，每一步必须明确定意义，并且不会引起误解。
- 协议必须是完整的，对每种可能的情况必须规定具体的动作。

密码协议的基本概念—协议特点

- 密码协议是使用密码技术的协议。
- 参与该协议的伙伴
  - 可能是朋友和完全信任的人，
  - 可能是竞争对手和互相不完全信任的人。
- 密码协议包含某种密码算法。
- 在协议中使用密码的目的是防止或发现偷听者、欺骗、抵赖等等。
- 不可能知道得比协议中规定的更多。

密码协议概述—协议的目的

- 计算一个数值想共享它们的秘密部分
- 共同产生随机系列
- 确定互相的身份
- 同时签署合同
- 证明一个事实
- ……

密码协议概述—仲裁协议

- 仲裁者是在完成协议过程中值得信任的第三方；
- 仲裁者在协议中没有既得利益、与参与协议的任何人没有利害关系；
- 协议中的所有的人都接受这一事实
- 仲裁者能帮助互不信任的双方完成协议

密码协议概述—裁决协议

裁决协议可以分成两个子协议

- 非仲裁子协议：每次执行；
- 仲裁子协议：出现争议时执行；

非仲裁子协议

I. Alice和Bob谈判合同的条款。
II. Alice签署合同。
III. Bob签署合同。

裁决子协议

I. Alice和Bob出现在法官面前。
II. Alice提出她的证据。
III. Bob也提出他的证据。
IV. 法官根据证据裁决。

密码协议概述—对协议的攻击

- 被动攻击 (passive attack)
  - 窃听协议的部分和全部
- 主动攻击(active attack)
  - 引入新的消息、删除原有的消息、重放消息
- 被动骗子(passive cheater)
  - 遵守协议，但试图获得协议外的消息
- 主动骗子(active cheater)
  - 破坏协议

- 协议对被动欺骗来说应该是安全的
- 合法用户可以发觉是否有主动欺骗

2. 一个鉴别与密钥交换的一个研究过程

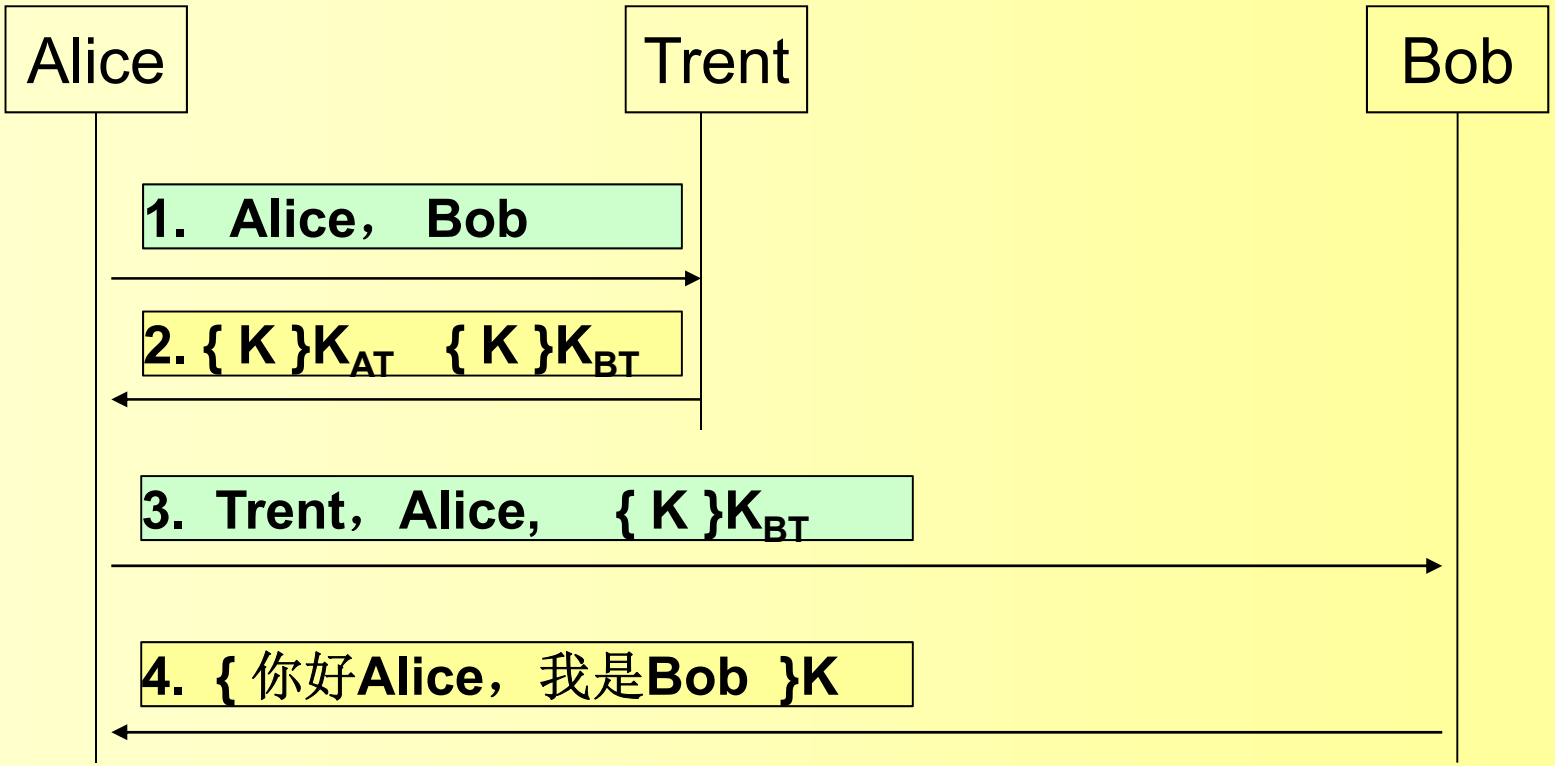
符号约定

- A：Alice，B：Bob，T：Trent
- K_{AT}：Alice与Trent的共享密钥；
- K_{BT}：Bob与Trent的共享密钥；
- { M } K：用密钥k加密消息M。
- $E_k(M)$，$D_k(M)$， E(k, M)，D(k, M)

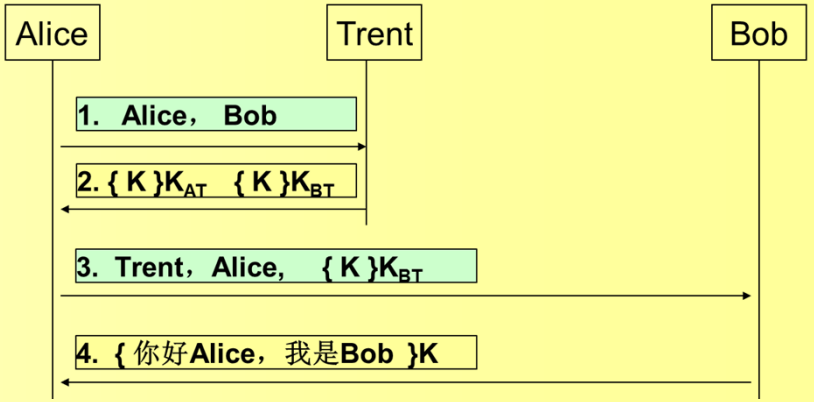
- K_U, K_R,
- $K_{U,Alice}$，$K_{U,Bob}$，$K_{R,Alice}$，$K_{R,Bob}$

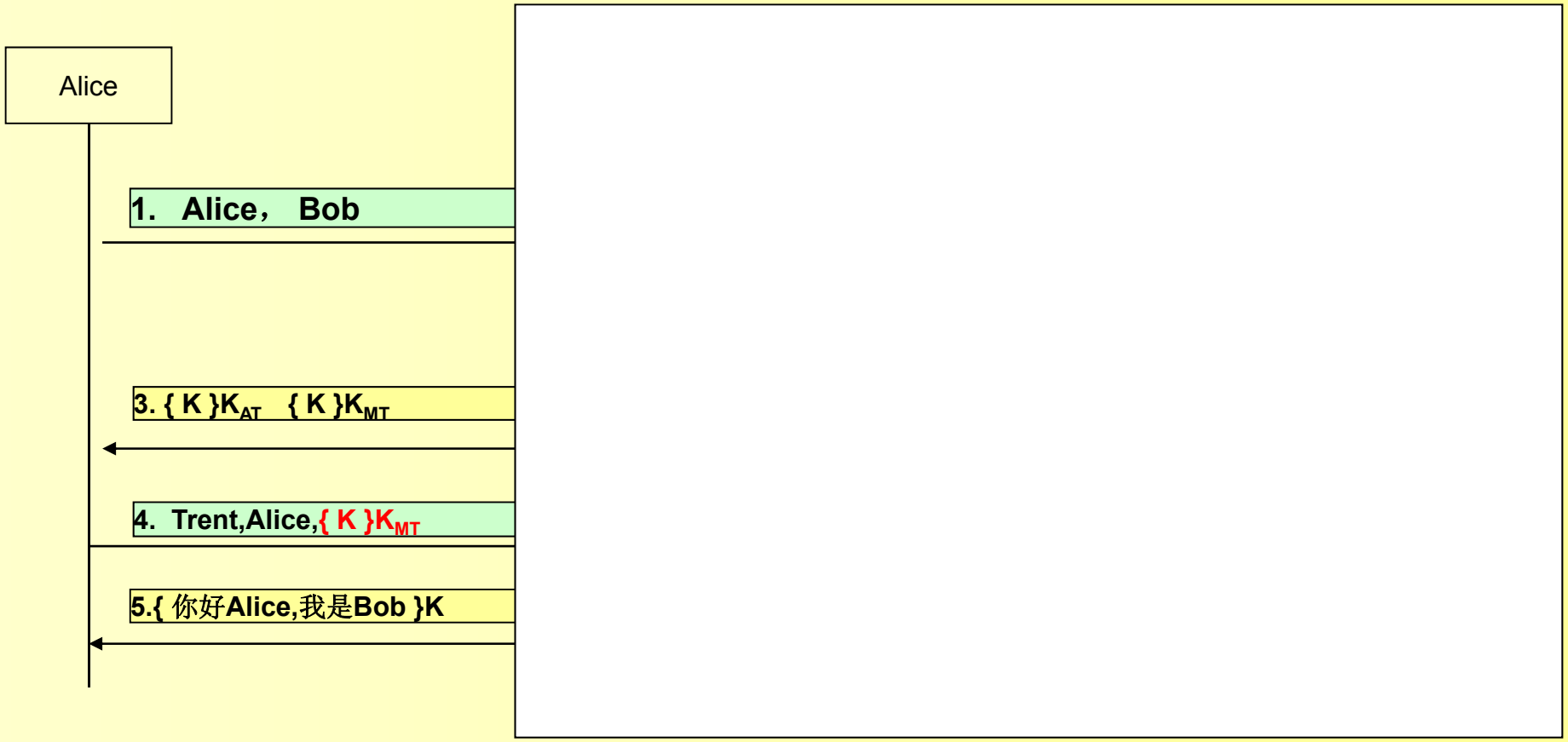
- [M] K = M || HMAC(K, M)

协议1

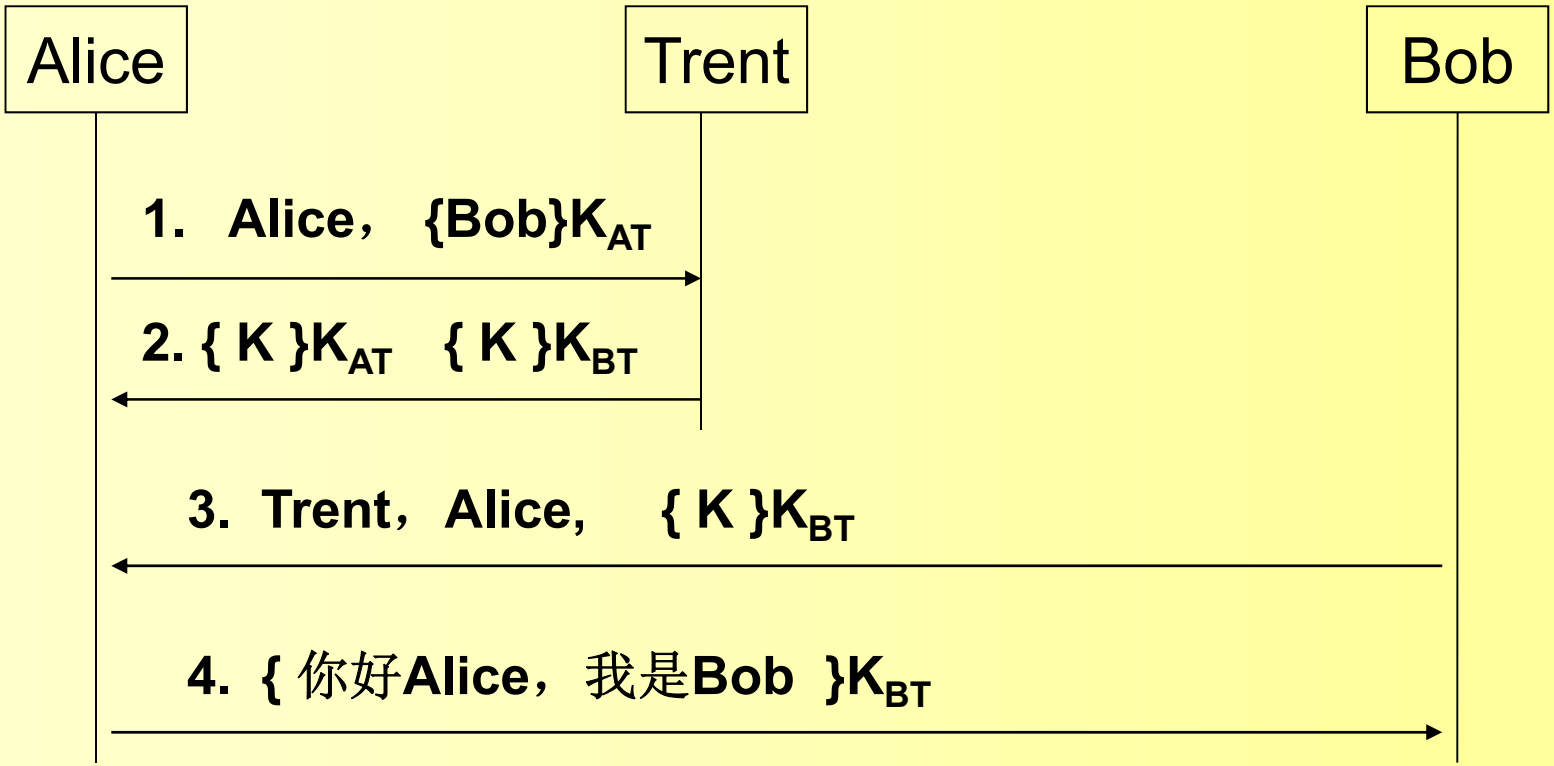

Alice
Trent
Bob
1. Alice， Bob
2. { K }K_AT { K }K_BT
3. Trent，Alice， { K }K_BT
4. { 你好Alice，我是Bob }K

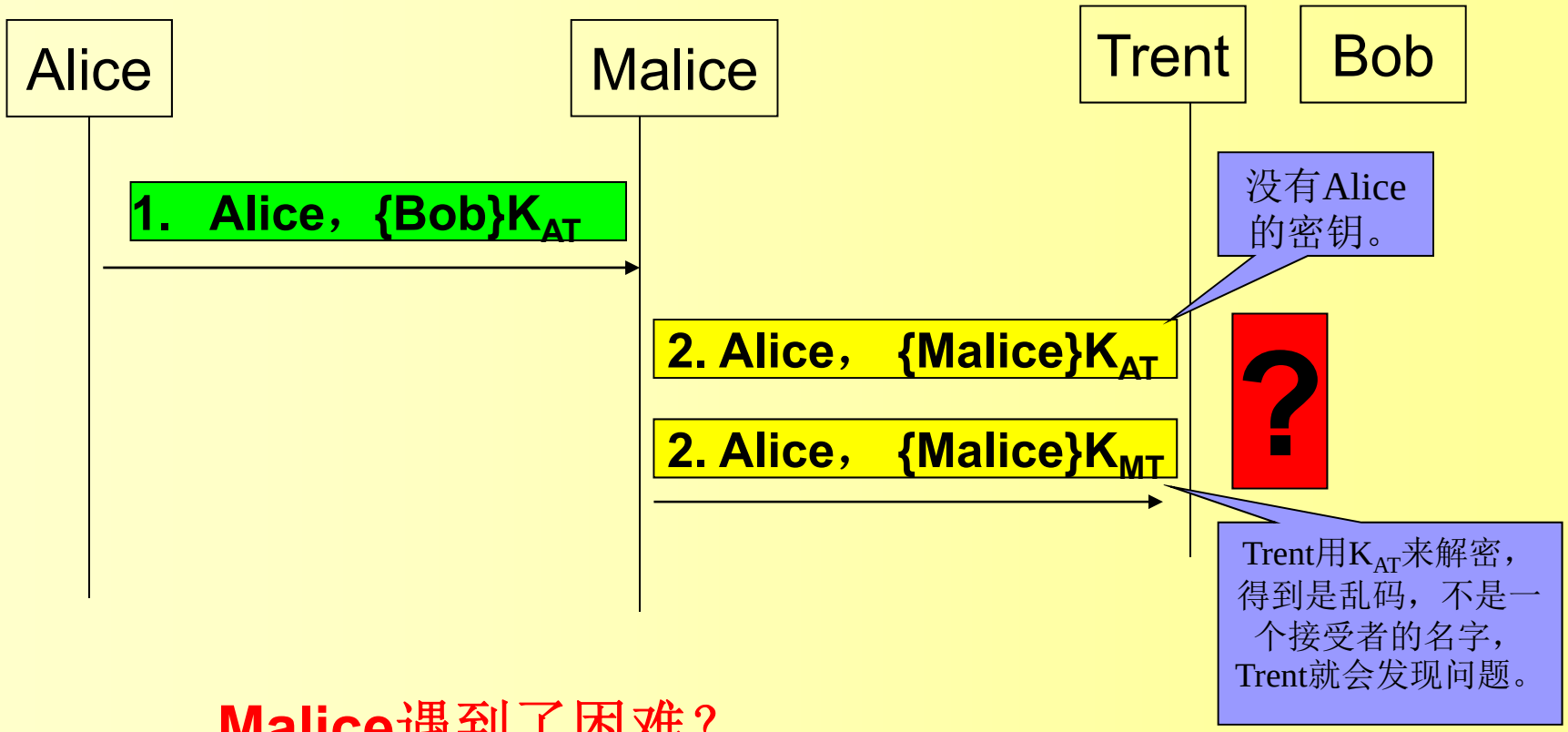
攻击1 — 对协议1的一个攻击





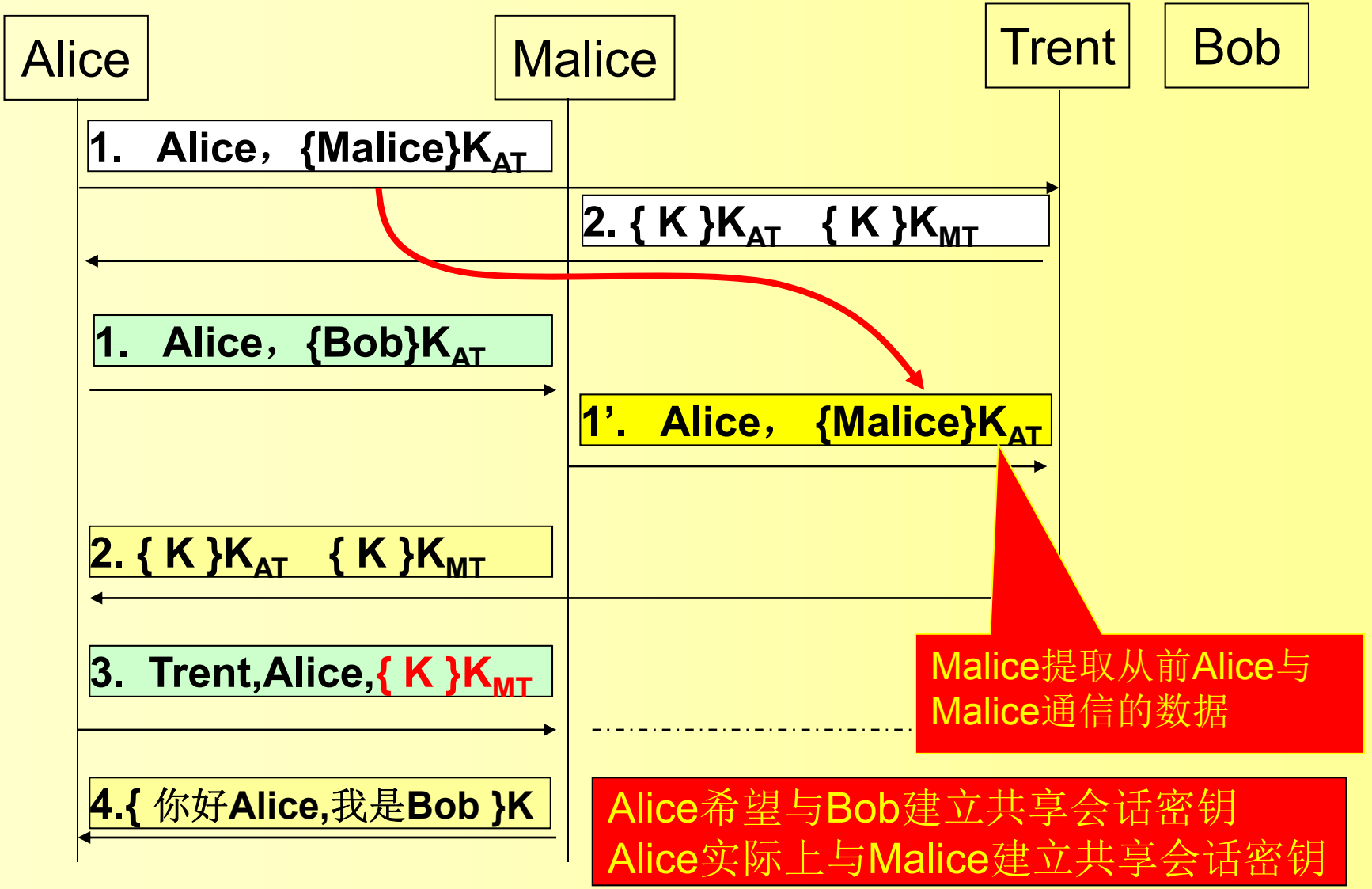
协议2 — 对协议1的一个修改



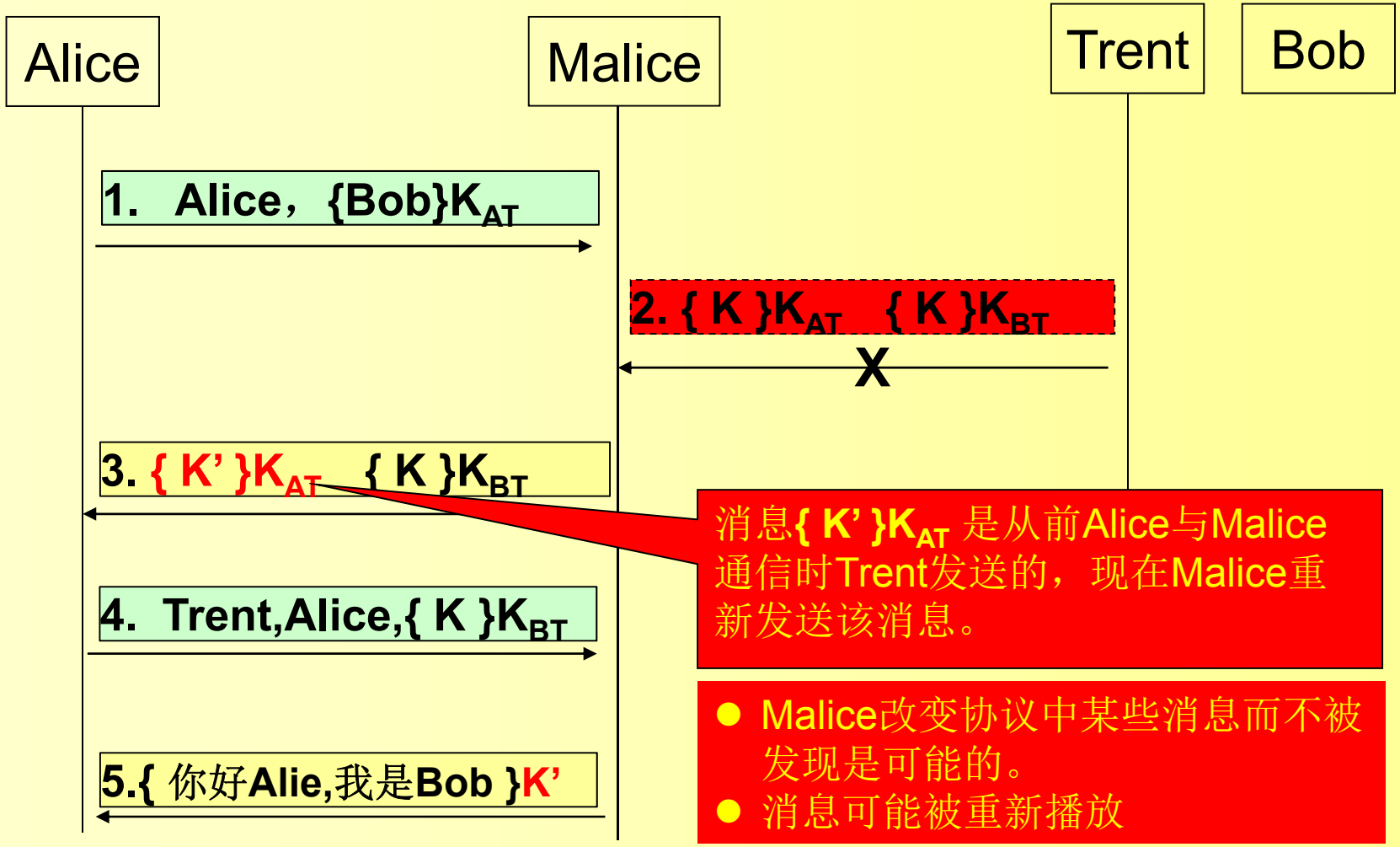


Malice遇到了困难？

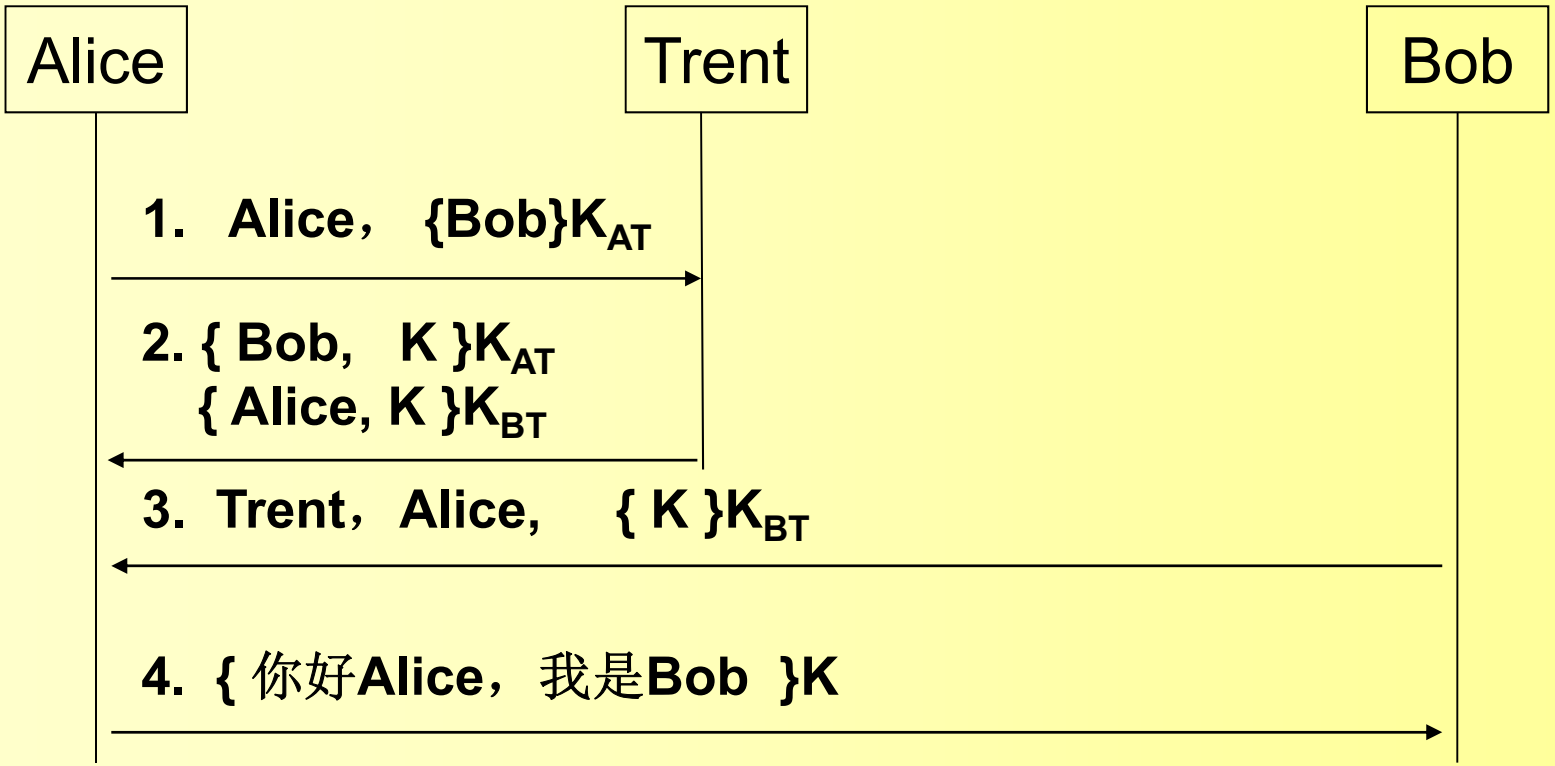
攻击2 — 对协议2 的一个攻击



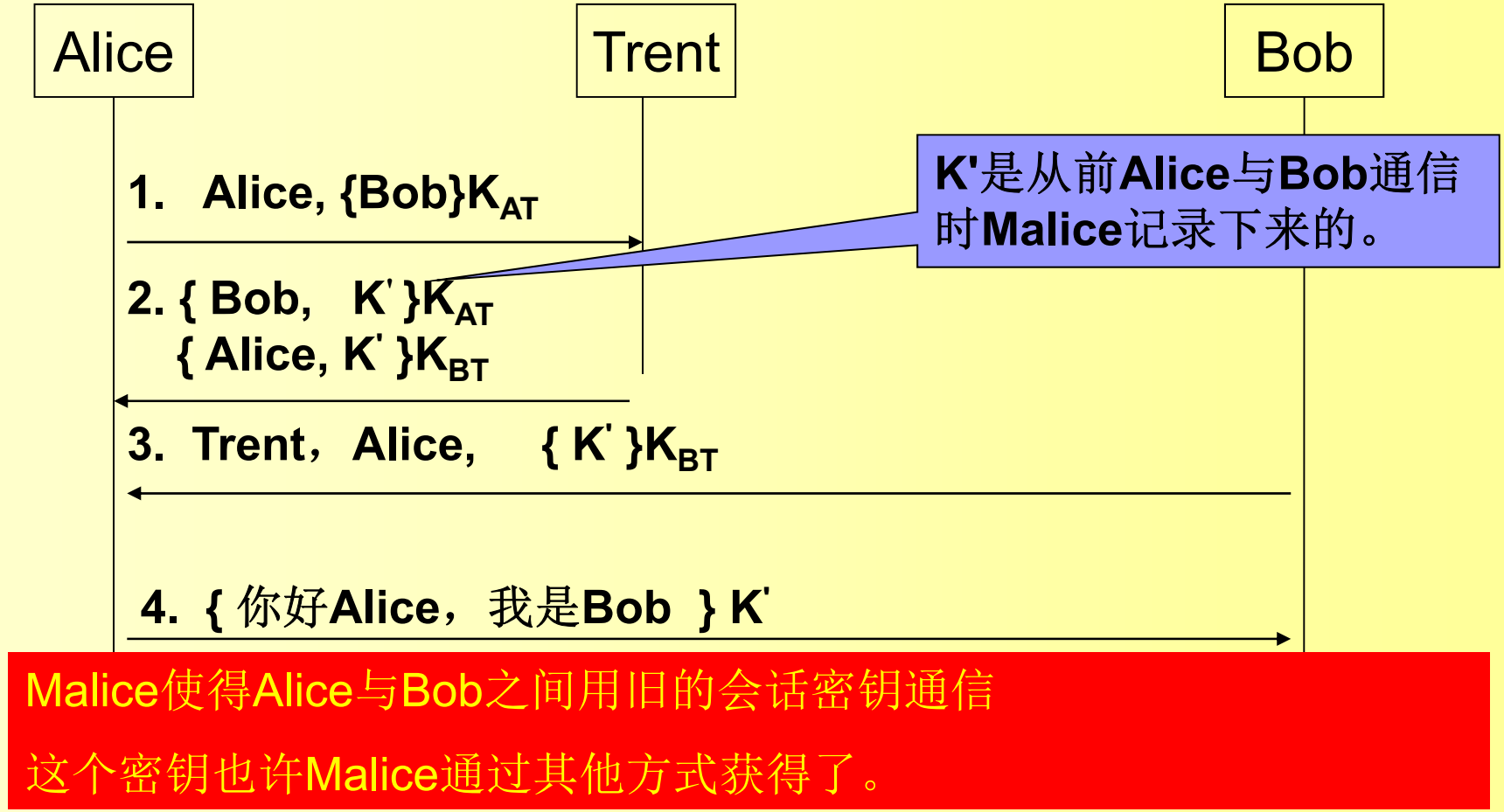
攻击3 — 对协议2的又一个攻击



协议3 —对协议2的一个修改



攻击4— 对协议3 的一个攻击



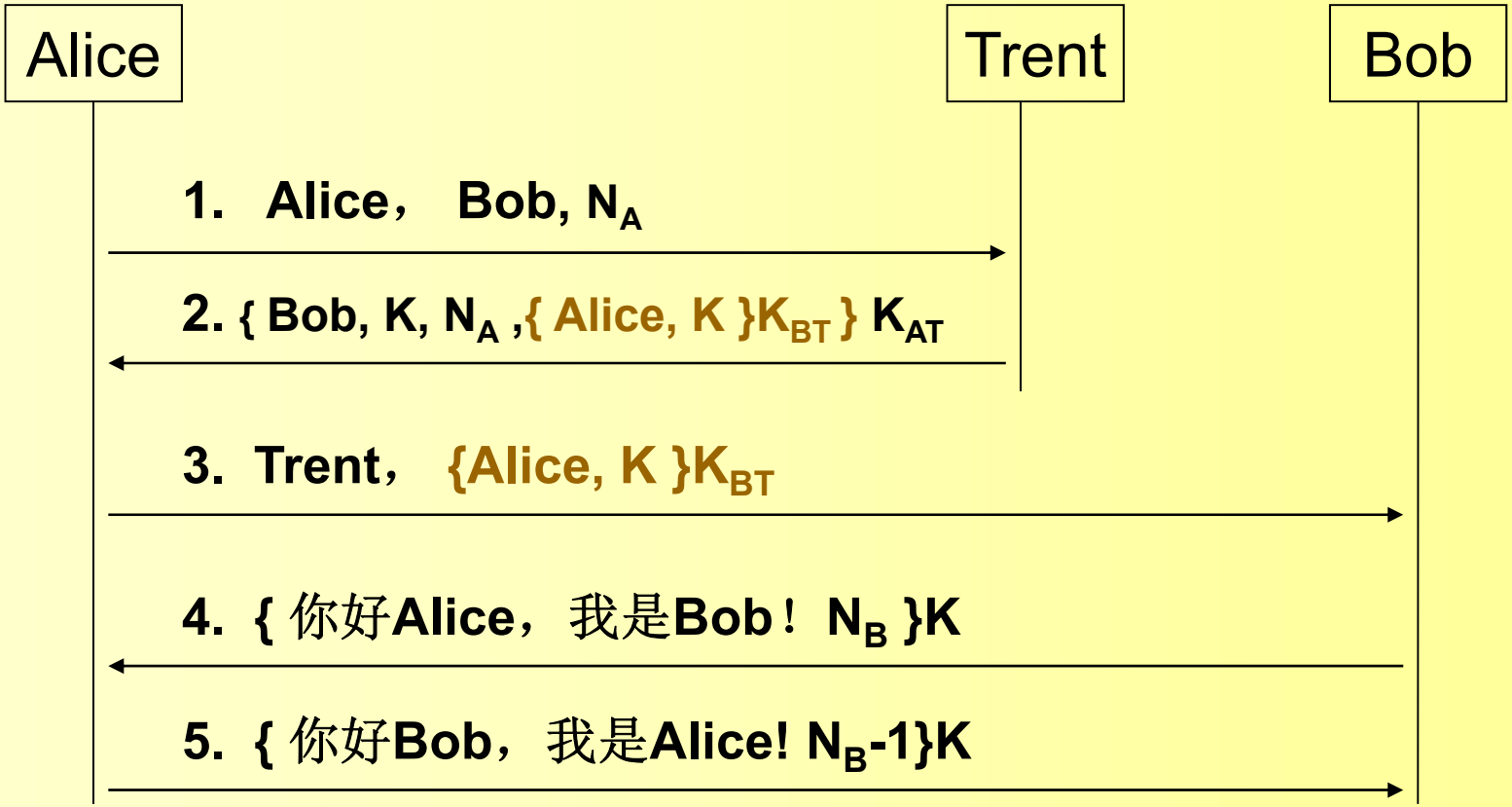
密码协议的关键问题

- 修改消息：消息没有鉴别，
  - 攻击1：修改消息的内容。
- 消息重放：消息不是新鲜的，
  - 攻击2，重放实体名称消息
  - 攻击3，重放密钥消息
  - 攻击4，重放破解的密钥消息

3. 认证协议

3.1 消息的新鲜性和实体的活现性

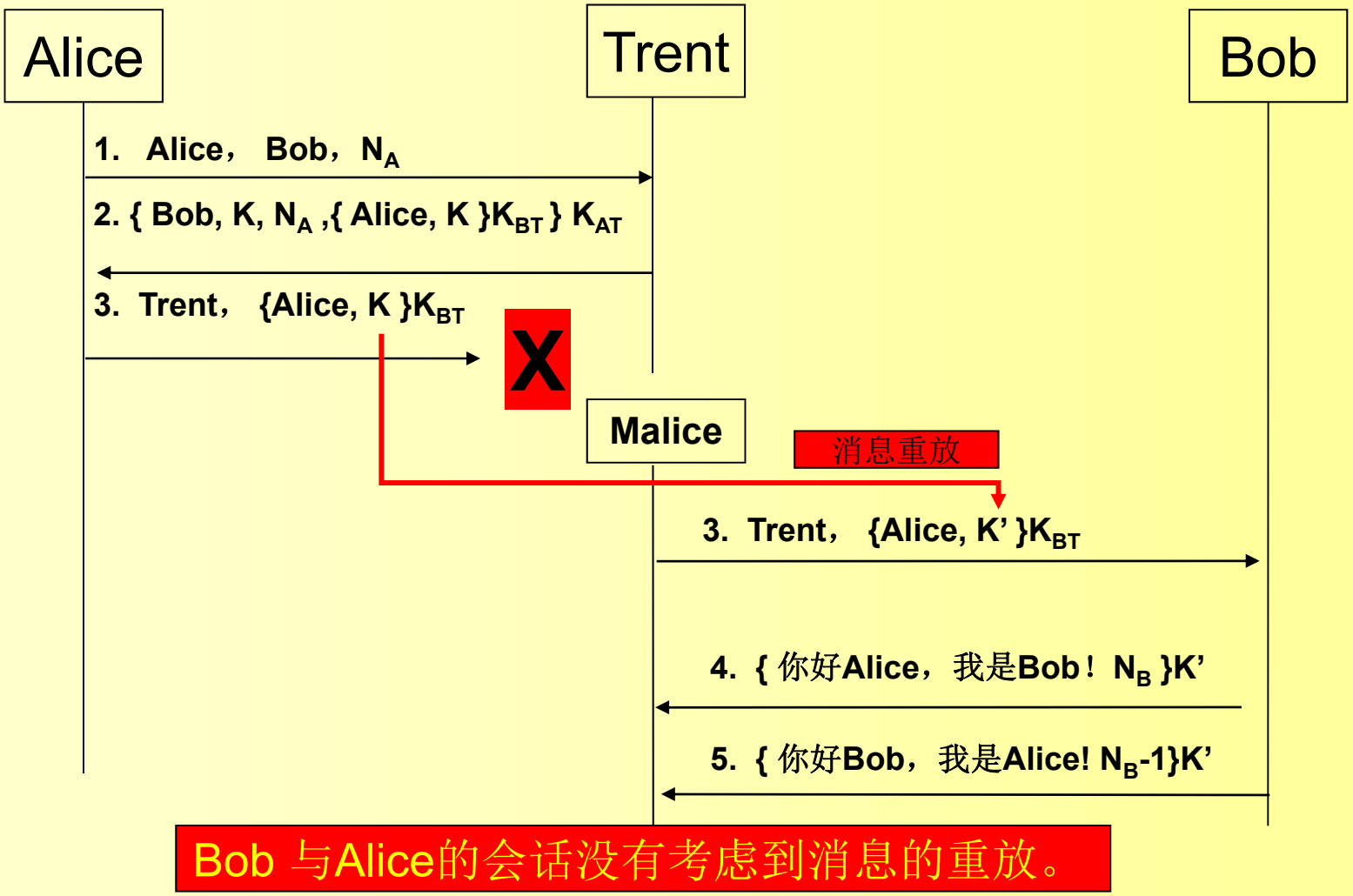
Needham-Schroeder 对称密钥认证协议



R.M. Needham and M.D. Schroeder, using encryption for authentication in large networks of computers, communications of the ACM, 21(12): 993-999, 1978.

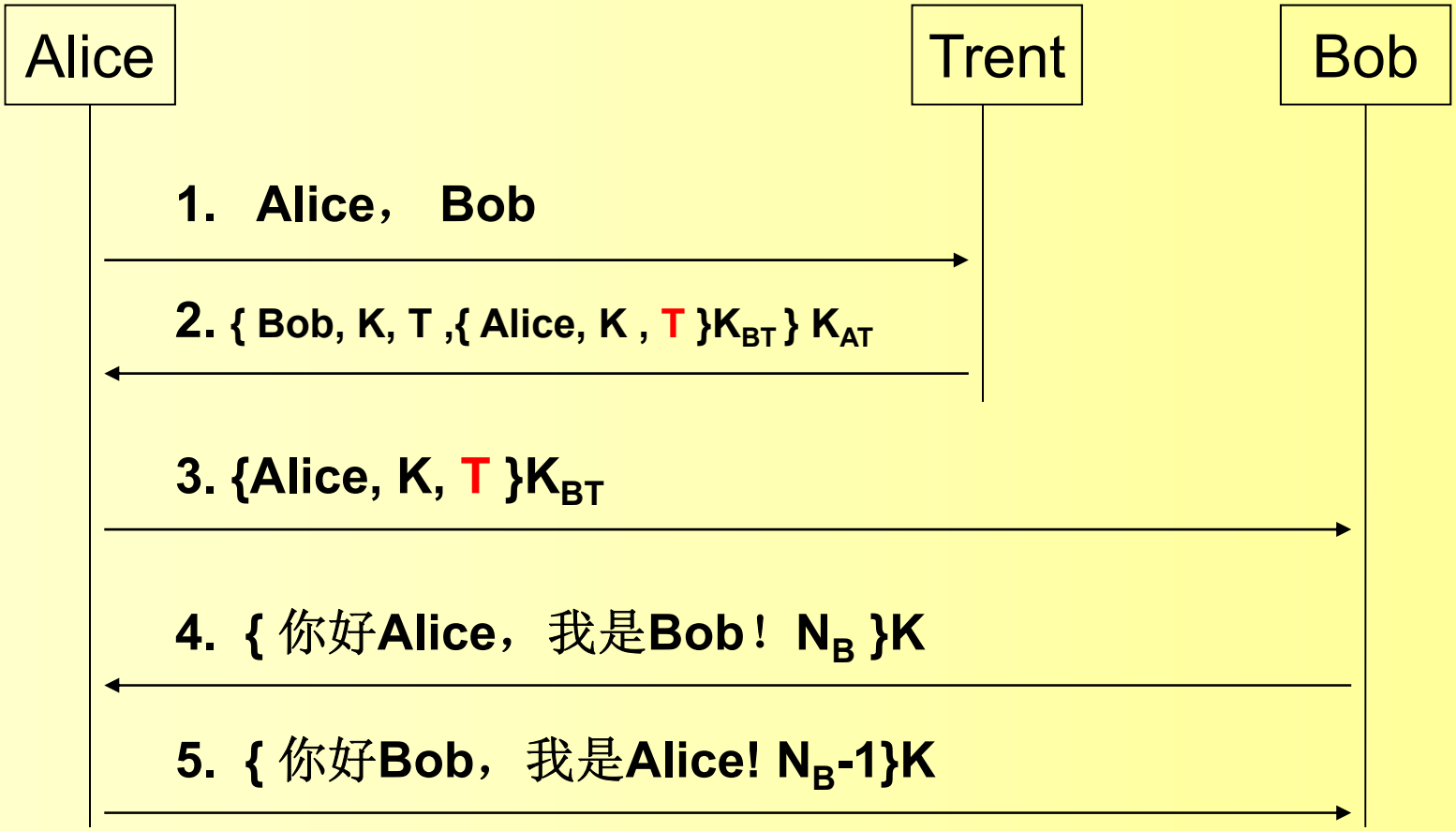
3.1 消息的新鲜性和实体的活现性

对Needham-Schroeder 对称密钥认证协议的攻击



Bob 与Alice的会话没有考虑到消息的重放。

Denning & Sacco的建议— 时间戳



验证是要求： $|Clock - T| < \Delta t_1 + \Delta t_2$

Denning & Sacco的建议— 时间戳

- Denning的协议与Needham / Schroeder的协议相比似乎增加了安全度。然而，又出现了新的担忧：这个新机制需要信任通过网络进行同步的时钟。
- 存在一种危险，分布时钟由于阴谋破坏或同步时钟、同步机制的故障变得不同步。
- 当发方的时钟快于预想的收方时钟时，这个问题就会发生。在这种情况下，对手可截获发自A的报文，当报文中的时间戳变成收方的当前时间时就重放该报文。这种重放可导致不可预料的结果。这样的攻击称为抑制—重放攻击。

3.1 消息的新鲜性和实体的活现性

询问－应答机制

I. Bob →Alice： N_B

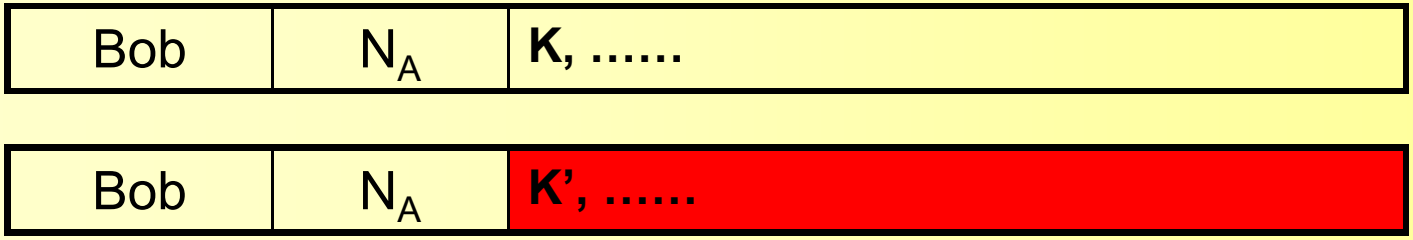
II. Alice →Bob：$E_{K_{AB}}(M, N_B)$

III. Bob 解密接收到的密文分组中看到N_B则接受Alice；否则拒绝接受Alice。

3.1 消息的新鲜性和实体的活现性

1. Alice， Bob, N_A
2. { Bob, N_A , K, { Alice, K }K_{BT} } K_{AT}

- 新鲜标记N_A可能在一个密文分组内，攻击这将会话密钥相关的分组换成过去的截取的分组，这样使得Alice使用旧的会话密钥。
- Trent发给Alice 的密文串： IV, C_1, C_2, … C_l, （1）
- Malice 截取的上次T给A的密文串： IV', C'_1, C'_2, … C'_l, （2）
- Malice截取(1)并将串(1) 改成： IV, C_1, C'_2, … C'_l, (3)
  N_A , K, …



- $K''=D_{K_{AT}}(C'_2) \oplus C_1 = K' \oplus C'_1 \oplus C_1$

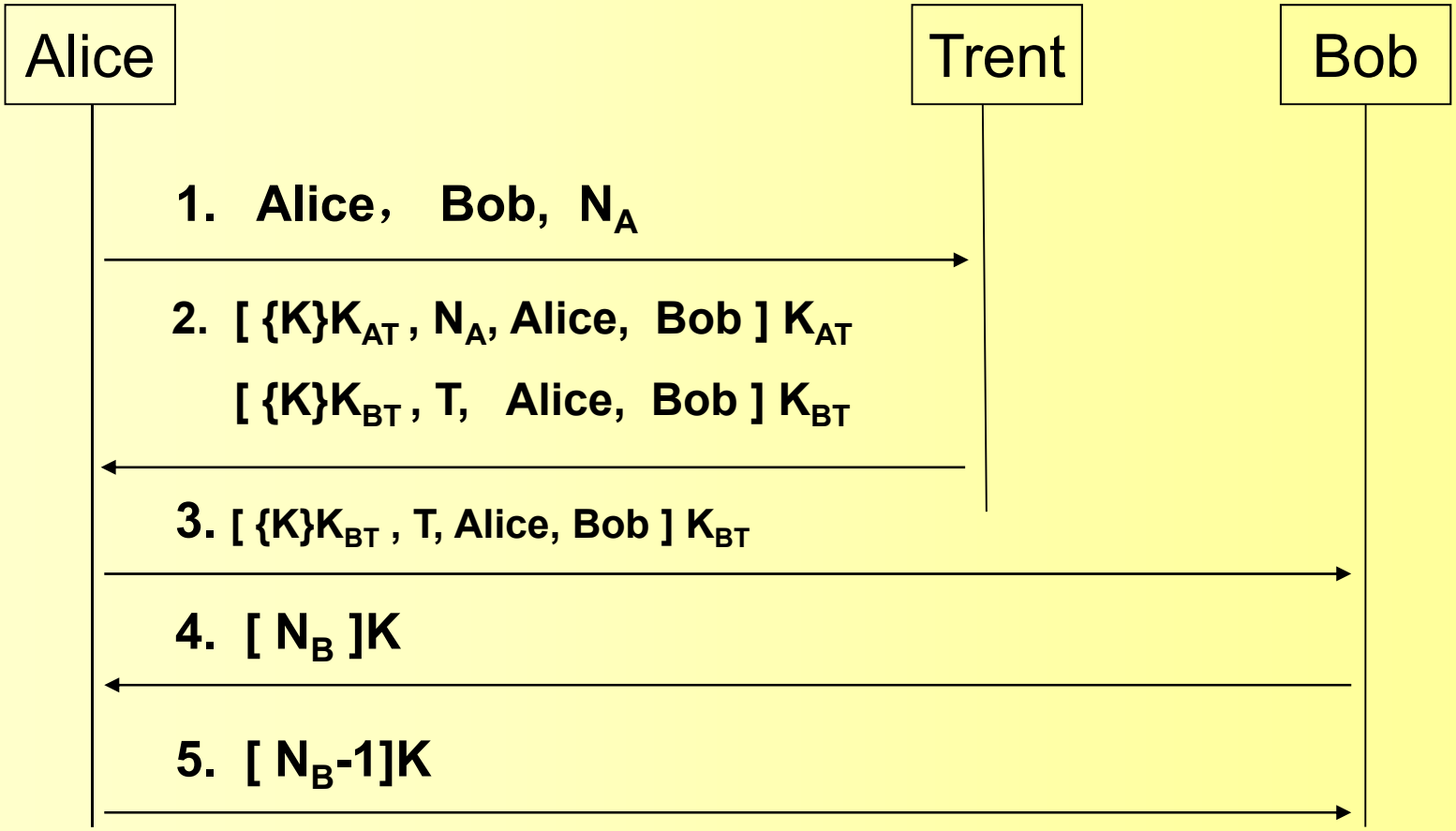
3.1 消息的新鲜性和实体的活现性

- 加密方法用来认证是不精确的
  - 加密不能提供数据完整性服务。
  - 我们还将看到：攻击者可能利用应答者的解密预言服务。

3.1 消息的新鲜性和实体的活现性

- 询问－应答机制 (利用消息鉴别码)
  - Bob →Alice： N_B
  - Alice →Bob：$MAC(K_{AB}, N_B)$
  - Bob 重构$MAC(K_{AB}, N_B)$，如果相等则接受Alice；如果不相等拒绝接受Alice。

3.1 消息的新鲜性和实体的活现性
Needham-Schroeder 对称密钥认证协议

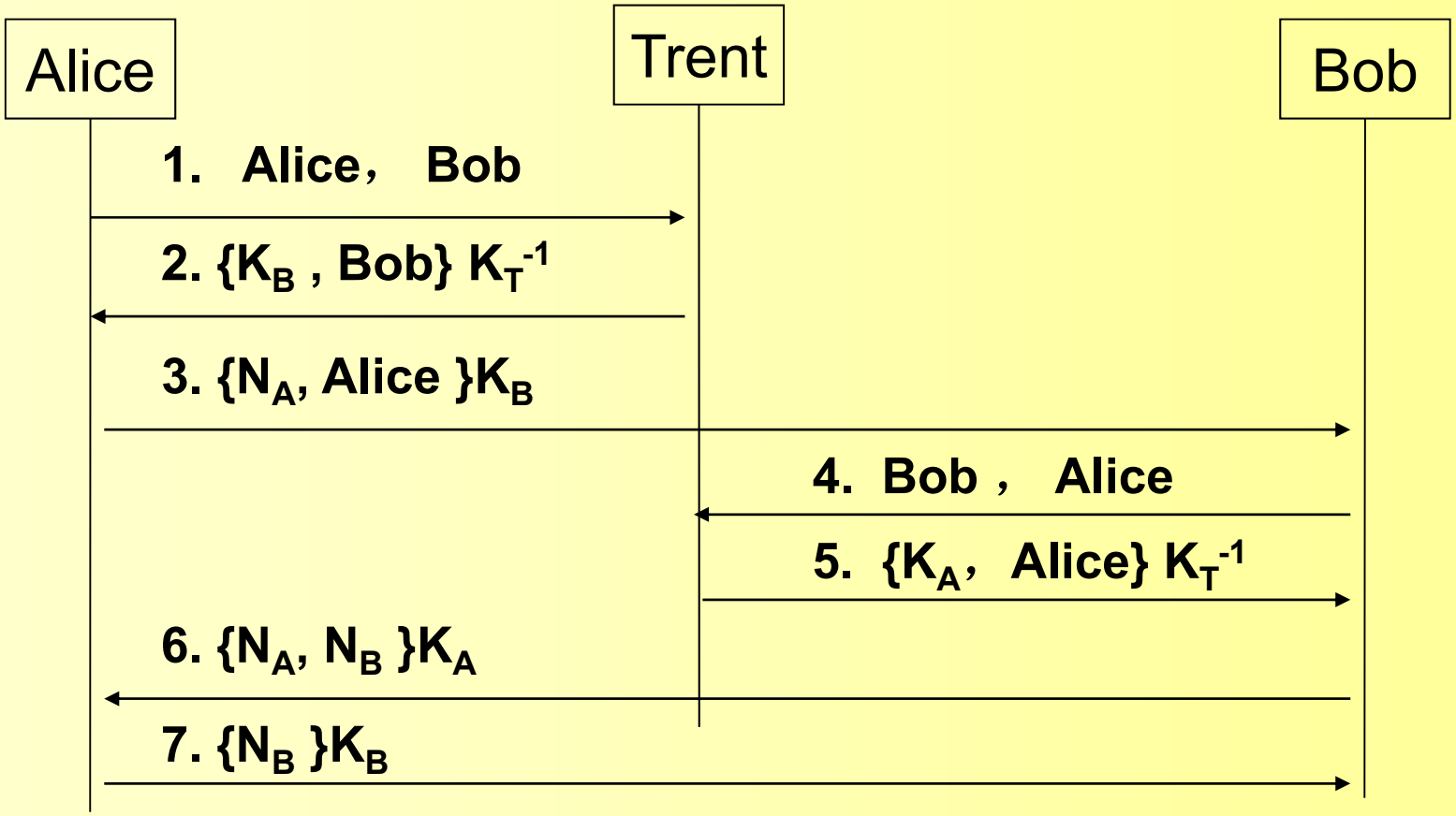


3.1 消息的新鲜性和实体的活现性

- 询问－应答机制(利用签名机制)
  - Bob →Alice： N_B
  - Alice →Bob：$sig_A(M, N_B)$
  - Bob 使用他的一次性随机数N_B验证签名，如果通过验证则接受Alice；否则拒绝接受Alice。

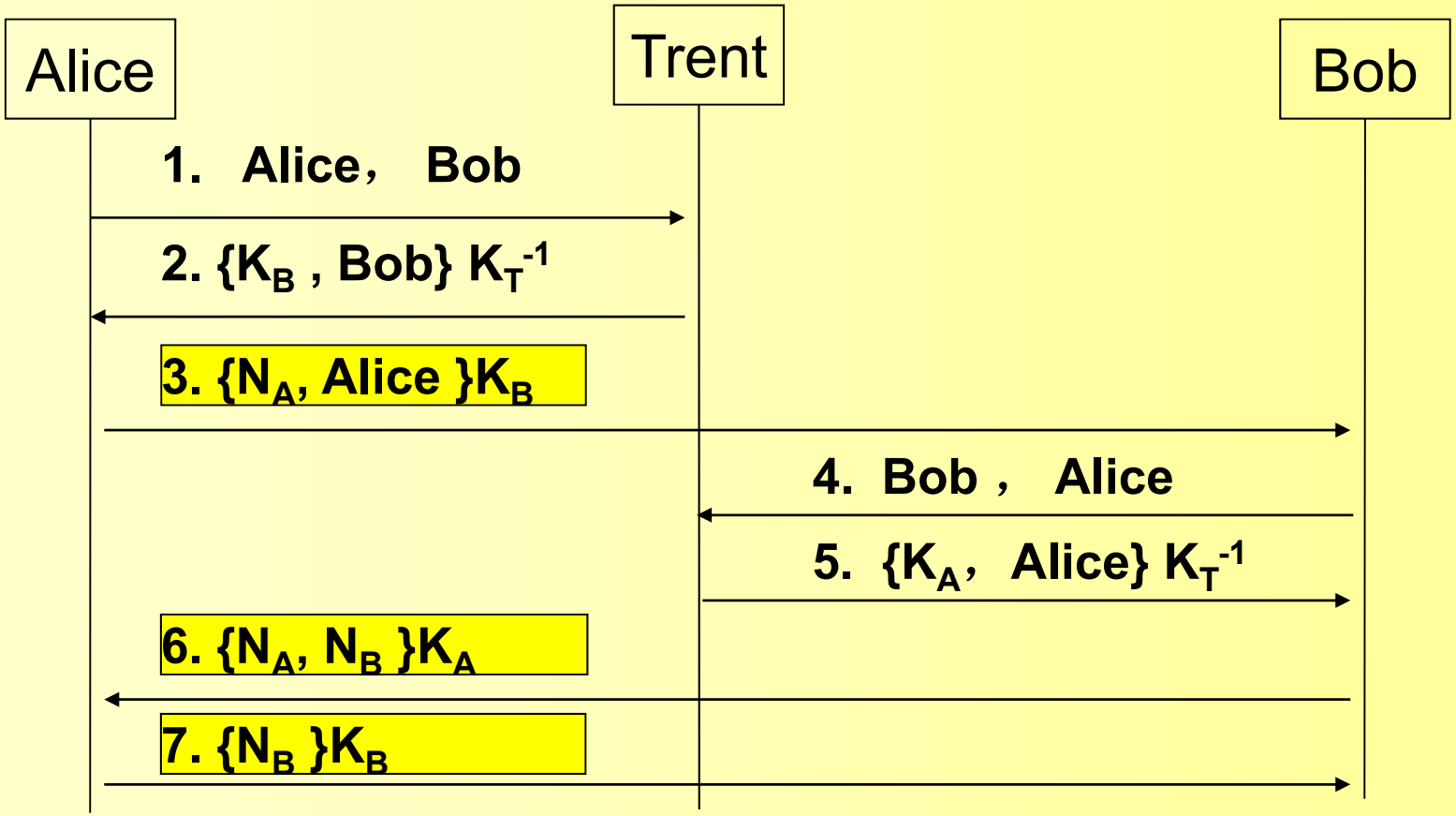
3.2 Needham-Schroeder公钥认证协议

- 假定: **Alice、Bob、Trent的密钥对分别为** $\langle K_A, K_A^{-1} \rangle$, $\langle K_B, K_B^{-1} \rangle$, $\langle K_T, K_T^{-1} \rangle$, K_A^{-1}**等为私钥。**

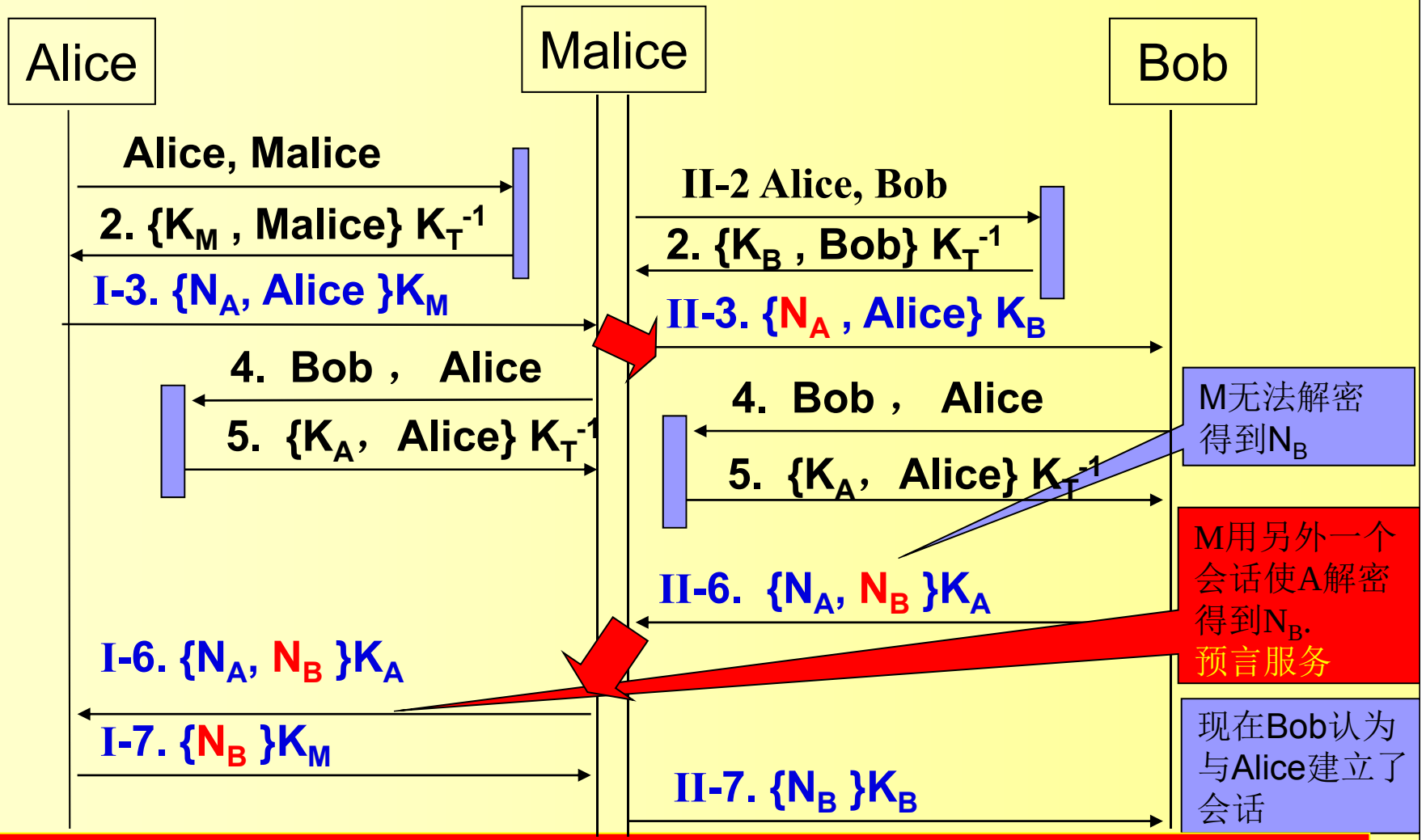


3.2 Needham-Schroeder公钥认证协议

- 假定: **Alice、Bob、Trent的密钥对分别为** $\langle K_A, K_A^{-1} \rangle$, $\langle K_B, K_B^{-1} \rangle$, $\langle K_T, K_T^{-1} \rangle$, K_A^{-1}**等为私钥。**

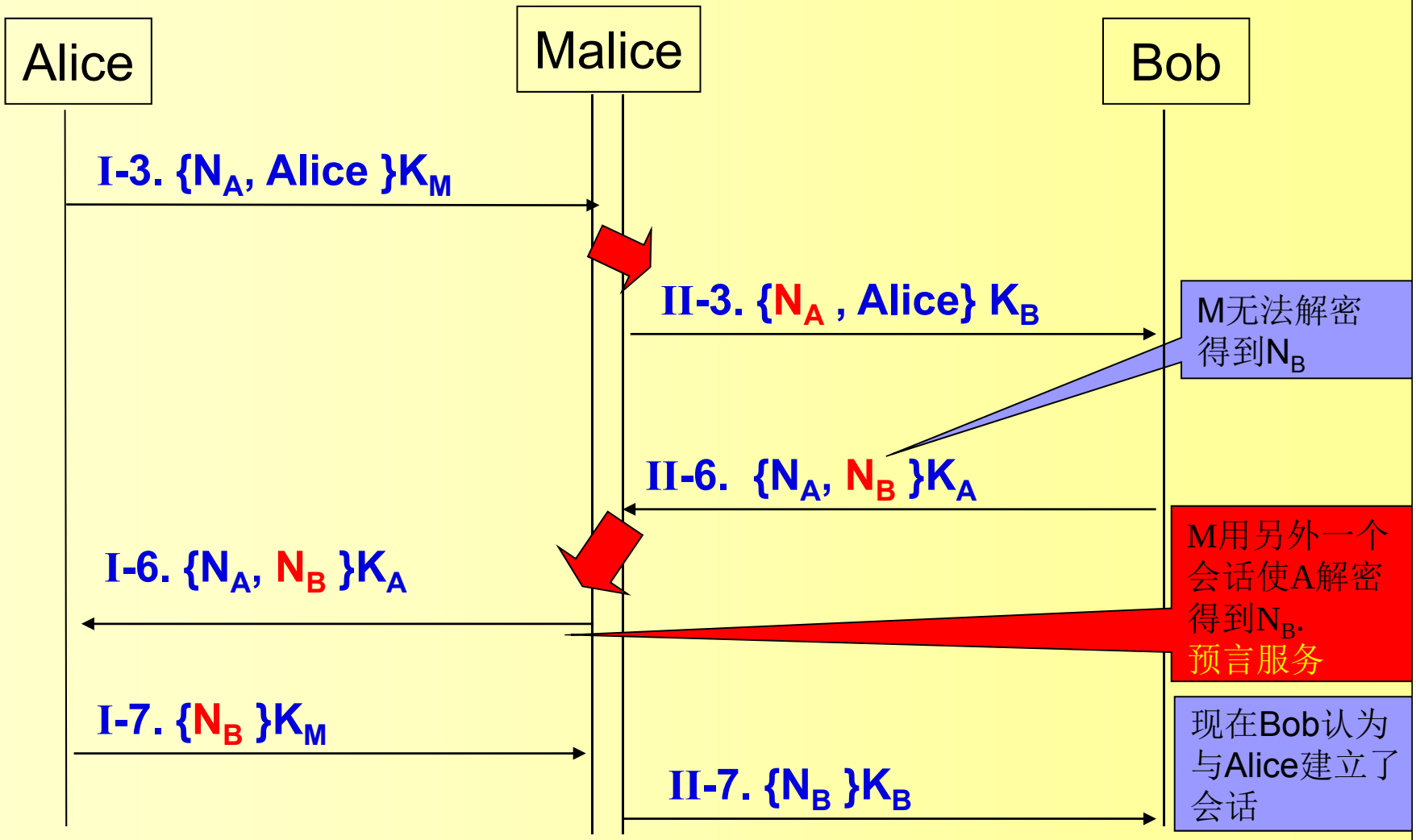


Lowe 的对Needham-Schroeder公钥认证协议攻击



密码协议应该设计成：即使用户为攻击者提供预言服务(oracle service)它也是安全的。

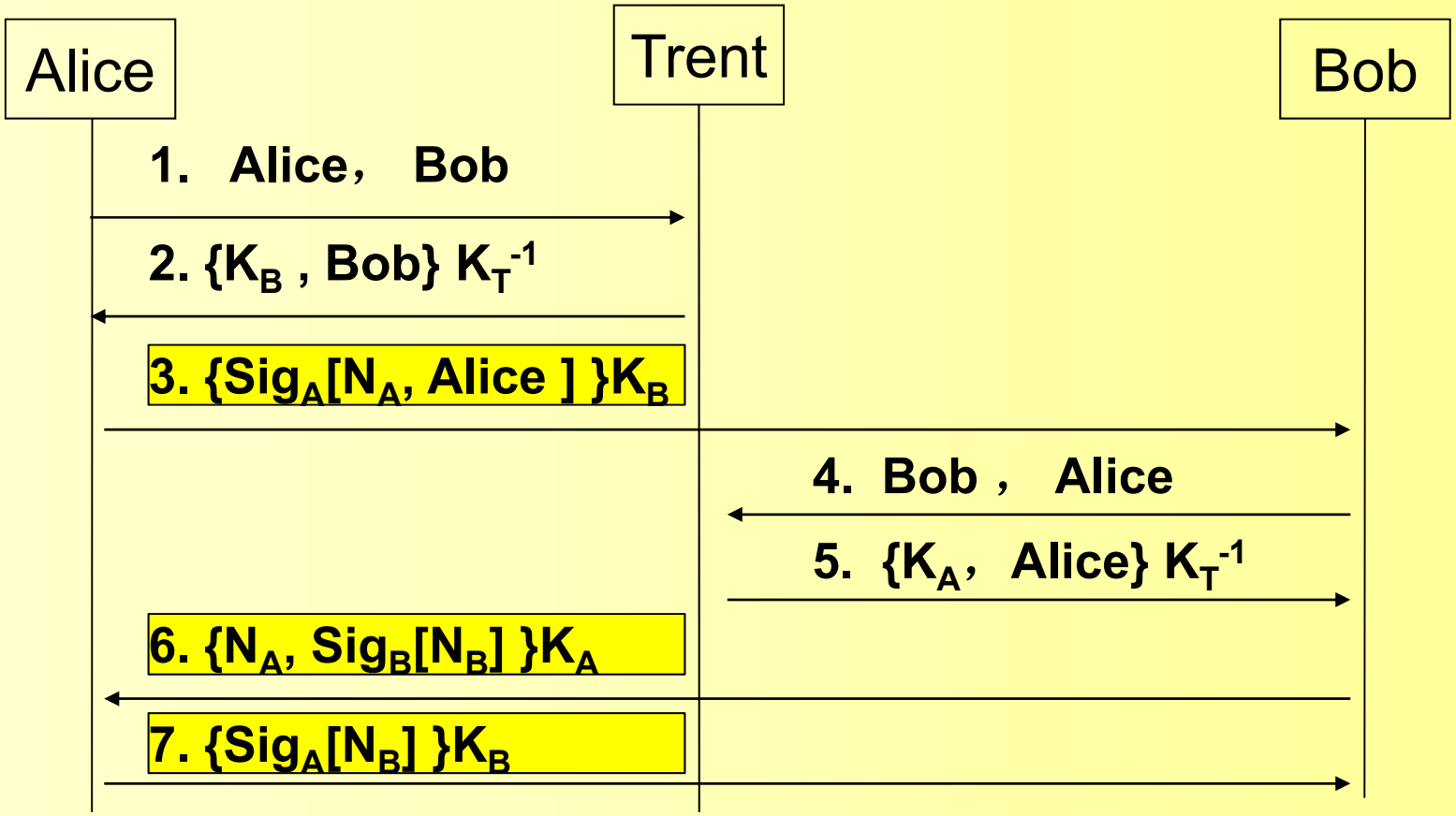
Lowe 的对Needham-Schroeder公钥认证协议攻击



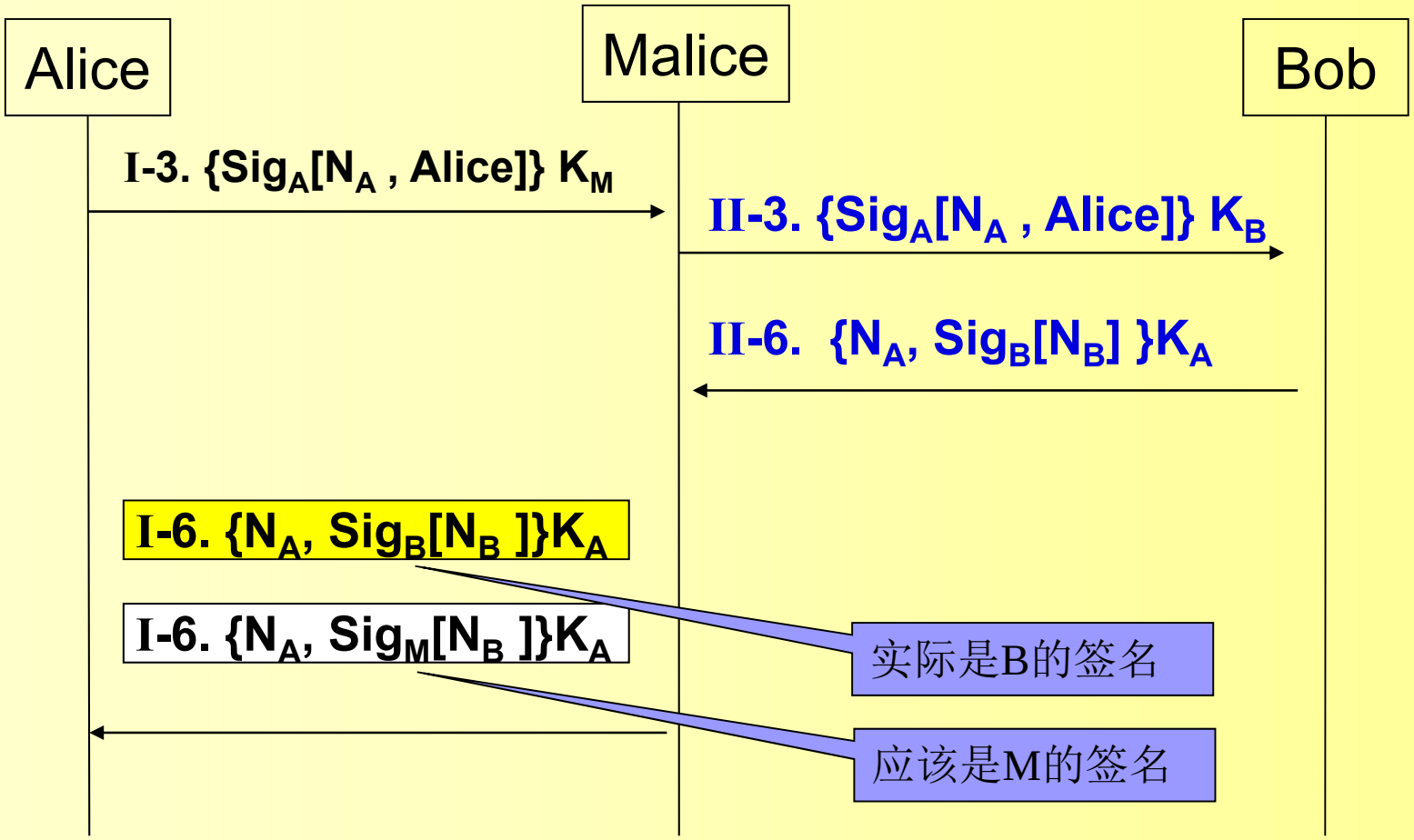
密码协议应该设计成：即使用户为攻击者提供预言服务它也是安全的。

Needham-Schroeder公钥认证协议的改进

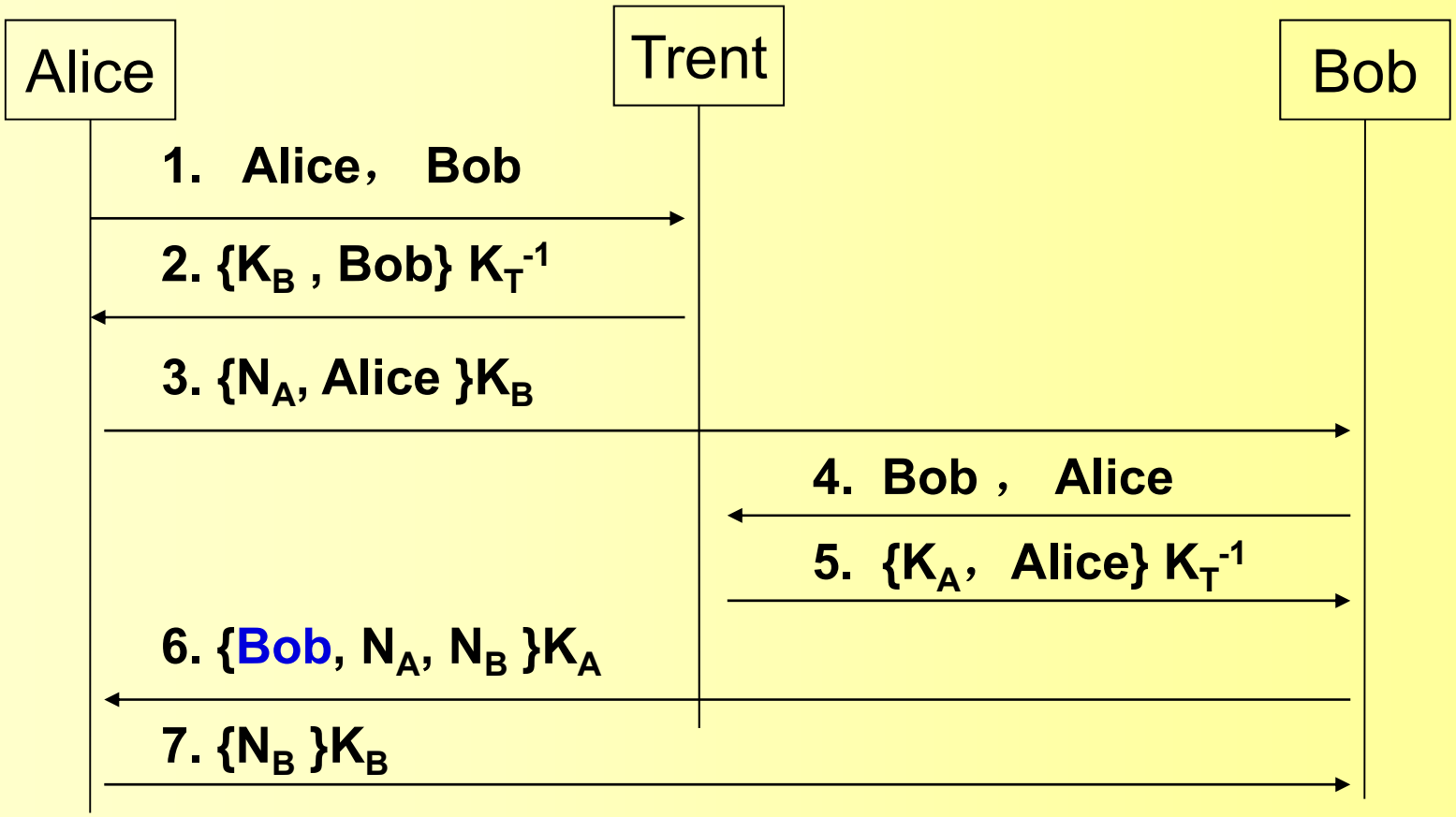
- 假定：Alice、Bob、Trent的密钥对分别为$\langle K_A, K_A^{-1}\rangle$，$\langle K_B, K_B^{-1}\rangle$，$\langle K_T, K_T^{-1}\rangle$，$K_A^{-1}$等为私钥。



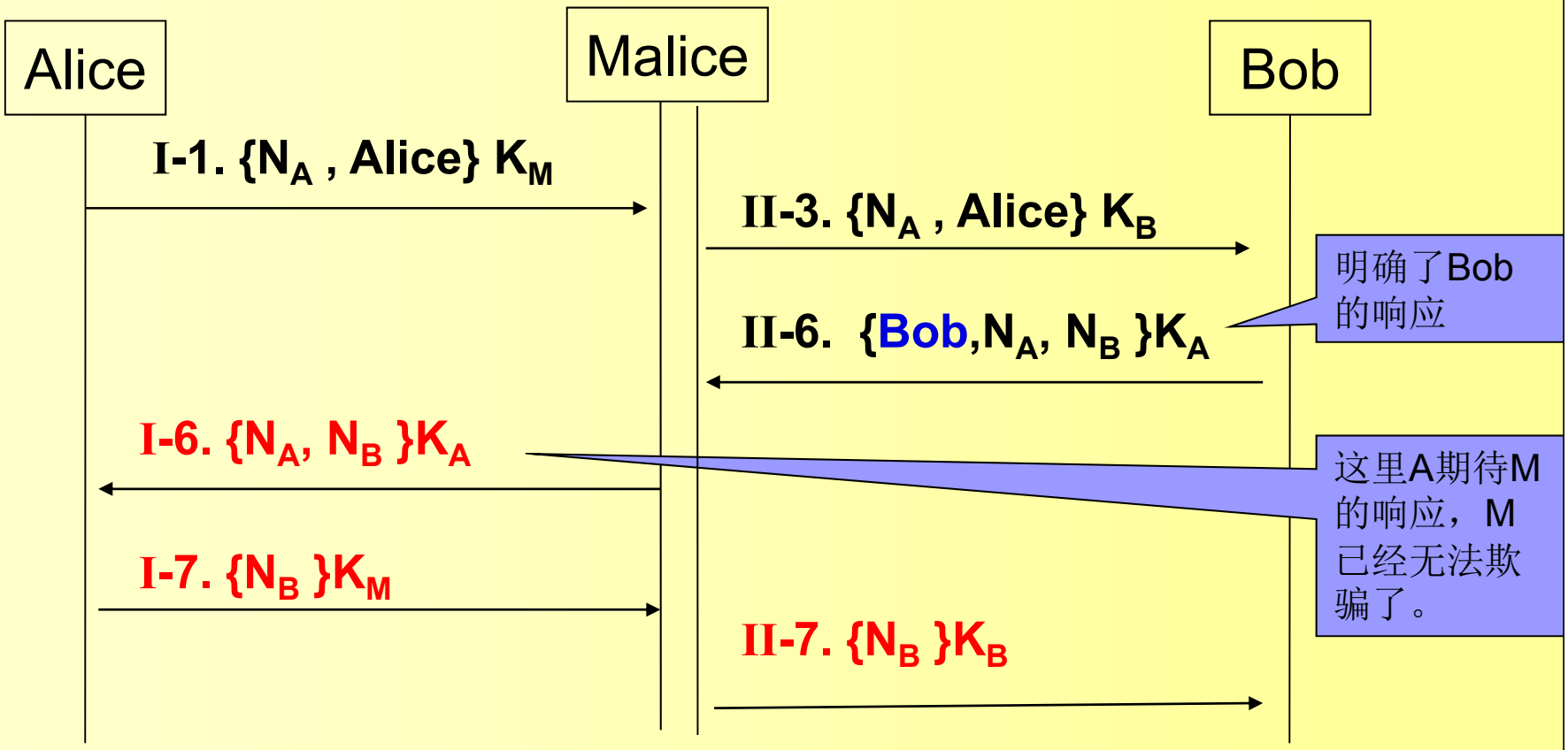
Needham-Schroeder公钥认证协议的改进



N-S公钥认证协议的一个修补



如果主体的身份对消息来说是必要的，则应当在消息中明确放入主体的名字。


Alice
Malice
Bob
I-1. {N_A , Alice} K_M
II-3. {N_A , Alice} K_B
II-6. {Bob,N_A, N_B }K_A
I-6. {N_A, N_B }K_A
I-7. {N_B }K_M
II-7. {N_B }K_B
明确了Bob的响应
这里A期待M的响应，M已经无法欺骗了。

问题

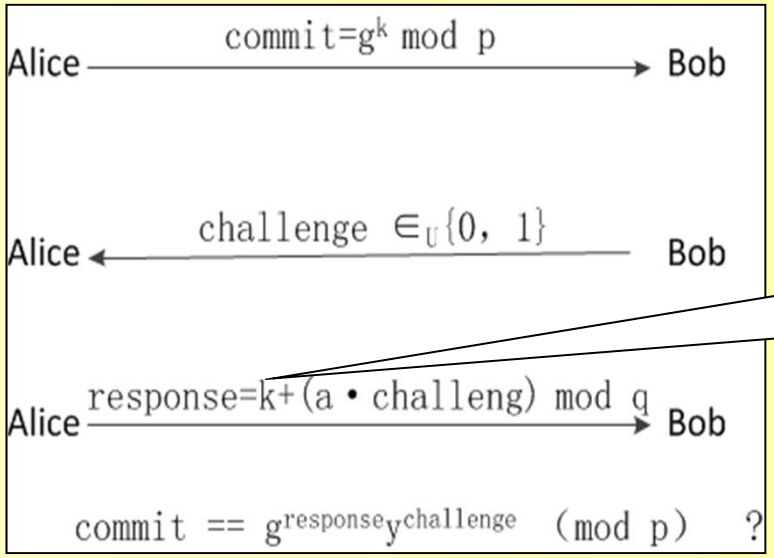
- 预言机：
  - 当一个主体无意地为攻击者执行了一个密码运算时，该主体就被用做预言机（oracle）或提供了预言服务（oracle service）。
  - 在验证一个用户的时候，通过解密一个密文来证实自己的用户的身份时，就可能被当做了预言机。（问题：冒充者可以设法让被冒充的用户来帮自己解密(提供预言服务)）。
- 加密方法用来认证是不精确的
  - 攻击者可能利用应答者的解密预言服务。
- 零知识性

3.3 零知识证明

- 国际象棋特级大师问题
  - 卡尔波夫执白与Alice对局；
  - Alice执白与卡斯帕罗夫对局
- 黑手党骗局
  - Alice 在黑手党Bob开的饭馆吃饭
  - 黑手党Carol在Dave的珠宝店买珠宝。
- 法拉第罩

Schnorr 身份认证协议

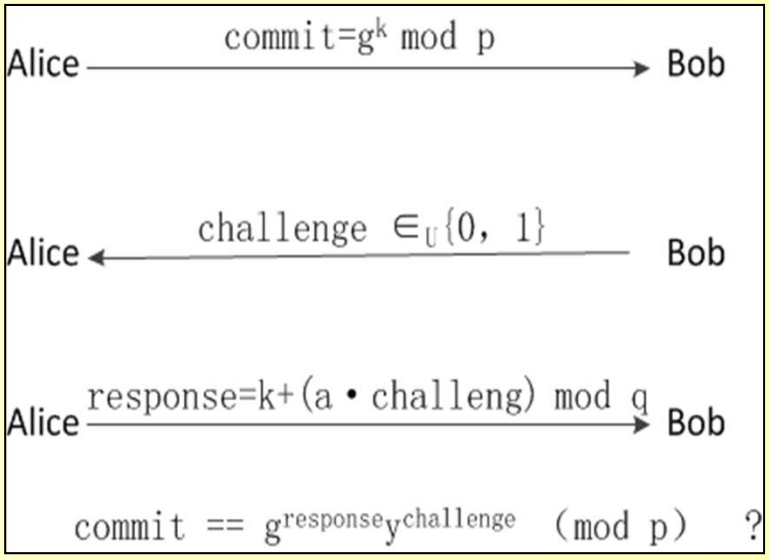
- 公共输入：
  - p，q:两个素数，满足 q | p-1；
  - g：$ord_p\ g = q$；
  - $y = g^{-a}$；
  - (p，q，g，y)是Alice的公钥，a 是Alice 的私钥。
- 认证



1. **Bob 挑战Alice 是否有a;**
2. **k+a是 避免直接暴露a。k是隐藏因子。**

Schnorr 身份认证协议的安全性

- 如果Malice想冒充Alice,但她没有Alice 的私钥a，她能成功地完成上述Schnorr 身份认证协议吗？



	challege = 0	challege = 1
	commit = g^k	commit = $g^k \cdot g^{-a}$
	Response = k	Response = k
	commit = g^k = $g^{Responce} \cdot y^{challenge}$	commit = $g^k \cdot g^{-a}$ = $g^{responce} \cdot y^{challenge}$

事先知道challenge=0所以在commit中就不乘y

事先知道challenge=1所以在commit中就乘y

Schnorr 身份认证协议的安全性

- 随机数会被利用吗？（认证者发出的数据会被验证利用来冒充认证者）

A	B	A’	C
→ g^{k_1} ← b_1 → $k_1+a\bullet b_1$	→ g^{k_1} ← b_1 → $k_1+a\bullet b_1$	→ g^{k_1} ← b'_1 → $k_1+a\bullet b_1$?	第一次验证者冒充认证者发出C的挑战不会刚好是b_1可能是b'_1 $k_1+a\bullet b_1$不再有用，而 $k_1+a\bullet b'_1$ 无法构造（因为没有a和k_1）

→ g^{k_2}
← b_2
→ $k_2+a\bullet b_2$

.....

→ g^{k_n}
← b_n
→ $k_n+a\bullet b_n$

- 认证者发送的是 $g^{k_1}, k_1+a\bullet b_1$ $g^{k_2}, k_2+a\bullet b_2$ … $g^{k_n}, k_n+a\bullet b_n$

  外人看来是一组随机数

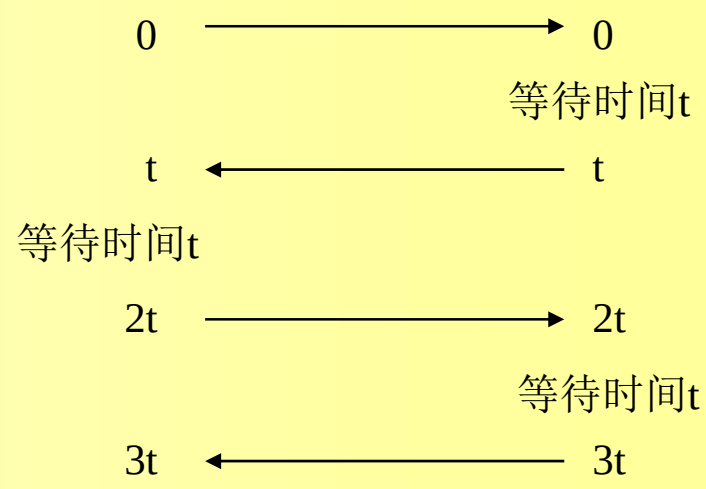
恐怖分子骗局

- Carol是一个恐怖分子。
- Alice帮助Carol进入这个国家，Alice与Carol用一条秘密的无线电信道联系。
- Dave是移民局官员。

- 当Dave问Carol零知识协议一部分问题时；
- Carol用无线电将它们发给Alice；
- Alice回答这些问题；
- Carol向Dave复述答案；
- 实际上是Alice向Dave证明她的身份，Carol只是作为一条通信路径。Dave 在实施恐怖行为时，Alice可以给出不在场证明。

能够抵御国际象棋大师攻击的交互式协议

1. 约定：协议的各方约定在固定的时间点发送数据：**0，t，2t，3t，4t，……。P**是认证者，**V**是验证者。
2. **P**在时刻**0**发送数据，设置检验时钟**z = 0**。
3. **V**收到数据，将时钟清**0**，在他自己的时刻**t**发送数据，并设验证时钟**y=t**。
4. **P**收到数据，观察时钟**e**，验证**e – z = t**。
   - 如果不成立则停止，认为协议被攻击。
   - 如果成立（还没有取胜），在时刻**e+t**发送数据，并设**z = e + t**。
5. **V**收到数据，观察时钟**f**，验证**f – y = t**，
   - 如果不成立则停止，认为协议被攻击。
   - 如果成立（还没有取胜），在时刻**f + t**发送数据，并设**y = f + t**。
6. 转**4**。



- **Y.Desmedt, C.Goutier, and Bengio, Special uses and abuses of the Fiat-Shamir passport protocol. CRYPTO’90, 1991, pp169-176**

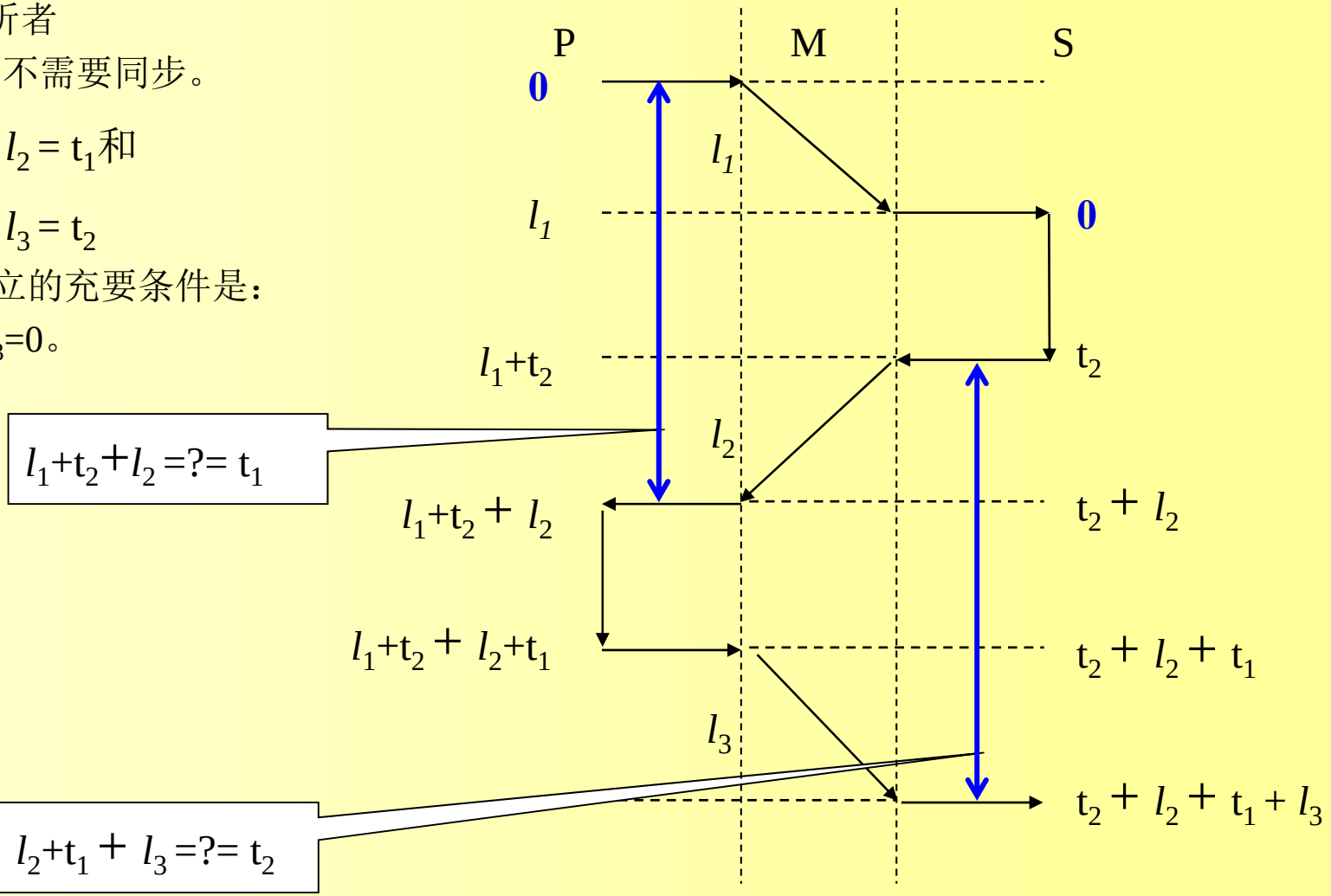
具有精确时钟的交互协议的正确性证明

- P：证明者，V：验证者，
  M：窃听者
- P和V的不需要同步。
- l_1+t_2 + l_2 = t_1和

  l_2+t_1 + l_3 = t_2

  同时成立的充要条件是：

  l_1= l_2 =l_3=0。



Basic Feige-Fiat-Shamir ID Scheme

- P证实自己身份的主体，V验证者；
- 取一个模数n，n为两个大素数的乘积。在验证者们中共享；
- 取一个数v，使得存在z,满足$z^2=v \pmod n$，$v^{-1} \bmod n$存在;v是P的公钥。
- P选择s, 使得$s=\text{sqrt}(v^{-1}) \pmod n$， 作为P的私钥。

二次剩余问题是一个公认的难题。Goldwasser-Micali密码体制是基于二次剩余问题的困难性。

(1) P 选择随机数r, r<n, 计算$x=r^2 \pmod n$, 将x发送给V;
(2) V 选择一个随机位b,将b发送给P;
(3) 如果b=0,则P发送y=r给V,如果b=1,则P发送$y=r \cdot s \pmod n$;
(4) V 验证 $x = y^2 v^b \bmod N$
(5) 连续进行t次

Feige-Fiat-Shamir ID Scheme

(1) P 选择随机数r, r<n, 计算$x=r^2 \pmod n$, 将x发送给V;
(2) V 选择一个随机位b,将b发送给P;
(3) 如果b=0,则P发送y=r给V,如果b=1,则P发送$y=r \cdot s \pmod n$;
(4) V 验证 $x = y^2 v^b \bmod N$
(5) 连续进行t次

- 如果P不知道s, 并且猜到V在第(3)步中
  - b=0: 选择r, 计算$x = r^2 \pmod n$; (1)发送x, (3)发送r;
  - b=1: 选择r,计算$x = r^2 \cdot v \pmod n$; (1)发送x, (3)发送r;
  - 但是，在第(1)步中发送x时，P并不知道b, 所以无法计算x, 所以如果P不知道s,P在第(1)步中只能猜。t次猜中的概率为$1/2^t$。

Feige-Fiat-Shamir ID Scheme

- 如果V试图冒充P
  - P向V证实自己时V记录P发送的数据<x, r | y>;
  - V向W证实自己是P时，重放<x, r| y>;
  - 但是每一个轮次W选择的b与V选择的b未必是一样的；因此V冒充P成功的概率为$1/2^t$。在每一轮中，P选择的r是不一样的。
  - P向V证实自己时是零知识的。

Feige-Fiat-Shamir ID Protocol

- P证实自己身份的主体，V验证者；
- 取一个模数n，n为两个大素数的乘积。在验证者们中共享；
- 取一个数v，使得存在x,满足$x^2=v_i \pmod n$, i=1,2, … , k;
- P选择s_i, 使得$s_i=sqrt(v_i^{-1}) \pmod n$, i=1,2, … , k;

(1) P 选择随机数r, r<n, 计算$x=r^2 \pmod n$, 将x发送给V;

(2) V 选择k个随机位b_i, i=1,2, … , k,将它们发送给P;

(3) P计算 $y= r \cdot s_1^{b1} \cdot s_2^{b2} \cdot \cdot \cdot s_k^{bk} \pmod n$, 将y发送给V

(4) V 验证 $x = y^2 \cdot v_1^{b1} \cdot v_2^{b2} \cdot \cdot \cdot v_k^{bk} \pmod n$

(5) 连续进行t次，欺骗成功的概率为$1/2^{kt}$

3.4 基于口令的认证

Lamport 一次性口令机制

L. Lamport. Password authentication with insecure communication. Communications of the ACM, 24(11)::770-772, 1981.

- 计算机计算$x_1=f(P_u)$, $x_2= f(f(P_u))=f^2(P_u)\ldots x_n= f^n(P_u)$等等n次。
- 计算机在登录数据库中用户U的名字后面存储(U, x_n)。
- 用户U保管P_u
- 第一次登录时，用户U输入他的名字和$f^{n-1}(P_u)$。计算机计算$f(f^{n-1}(P_u))$, 判断是否与保存的x_n相等，若相等则认证成功并用$f^{n-1}(P_u)$ a代替x_n。
- 第二次登陆时主机和用户分别用$f^{n-1}(P_u)$和 $f^{n-2}(P_u)$来认证。
- 以此类推。

主机$f^{k-1}(P_u)$和 用户$f^{k-2}(P_u)$的同步问题

S/KEY

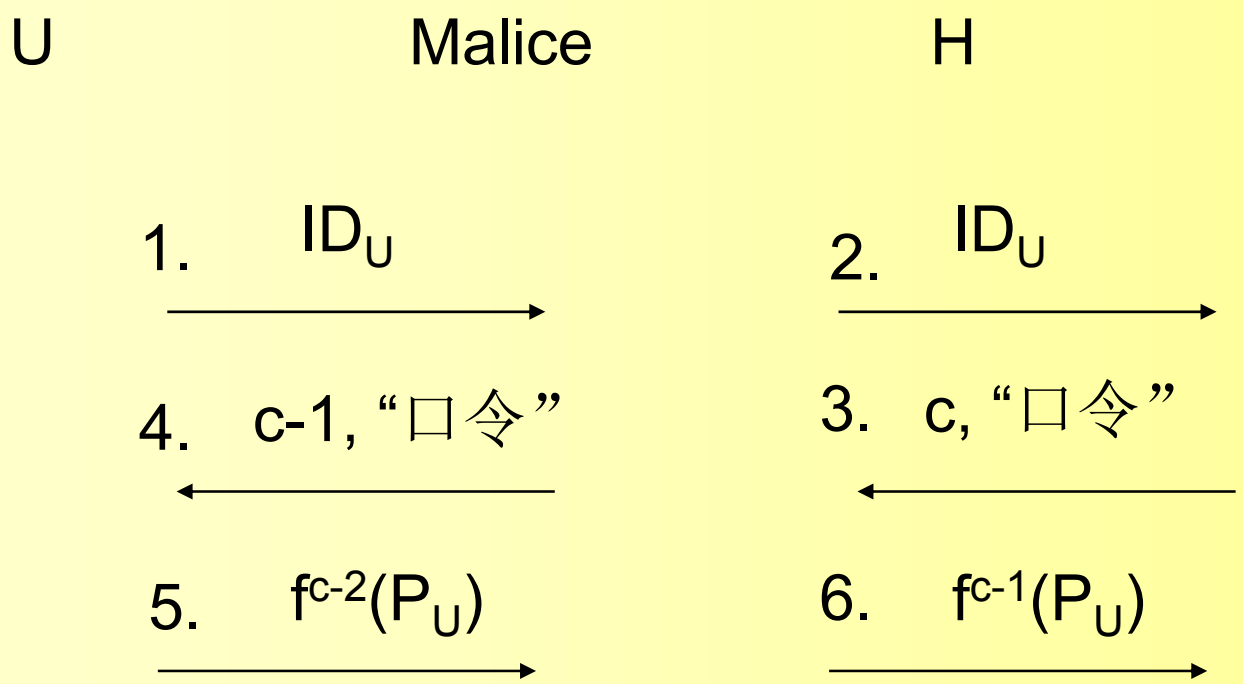
N. M. Haller. The S/KEY one-time password system . In Proceedings of the Synposium on Ntework and Distributed System Security, Pages 151-157, 1994.

- 用户U和主机H已经设定初始口令记录(ID_U, $f^n(P_u)$, n)

1. U→H: ID_U
2. H→U: c, “输入口令”
3. U→H：$f^{c-1}(P_u)$
4. H计算$f(f^{c-1}(P_u))$, 比较库中的数据(ID_U, $f^c(P_u)$, c)

对于S/KEY协议的攻击

中间人攻击



加密密钥交换EKE

S.M. Bellovin and M. Merritt. Encrypted key exchange； Password-based protocols secure against dictionary attacks. In Proceedings of the 1992 IEEE Symposium on Reseach in Security and Privacy, 1992.

(1) A 产生一对密钥<K_{UA} ,K_{RA} >，以P (Password)作为密钥用对称算法加密K_{UA}后发给B：A, $E_P(K_{UA})$

(2) B解密$E_P(K_{UA})$后得到K_{UA}；产生随机会话密钥K, 把下列消息发送给A: $E_P(E_{K_{UA}}(K))$; （E_{KUA}是非对称算法）

(3) A 产生随机串R_A,把下列消息发送给B: $E_K(R_A)$ A挑战B

(4) B 产生随机串R_B,把下列消息发送给A: $E_K(R_A, R_B)$ B证实自己，挑战A

(5) A 解密消息$E_K(R_A, R_B)$，得到R_A, R_B，将其中的R_A与(3)中产生的R_A比较，如果相等，则说明B是真实的，并把下列消息发送给B: $E_K(R_B)$。 A接受B，证实自己

(6) B解密消息$E_K(R_B)$，得到R_B，将其中的R_B与(4)中产生的R_B比较，如果相等，则说明A是真实的。 B接受A

(7) 双方用K加密通信数据。

A挑战B

(1) A →B：A, $\{K_{UA}\}P$

(2) B解密 $\{K_{UA}\}P$ 后得到K_{UA}；产生随机会话密钥K

B →A: $\{\{K\}\ K_{UA}\}P$

- 如果窃听，监听到了$\{K_{UA}\}P$，$\{\{K\}\ K_{UA}\}P$，$\{R_A\}K$
- E猜测的P解密$E_P(K_{UA})$，仍然要判断是否猜测对了，然而这时的判断需要猜测随机数K和R_A，而随机数K和R_A的空间是很大的，即使口令选择得很坏，E仍然很难完成猜测。
- EXE中的非对称算法可以用各种非对称算法实现：RSA, ElGamal, Diffie-Hellman等。

3.5 Diffie-Hellman密钥交换

Diffie-Hellman算法

- Alice选取一个大的随机数x，发送给Bob
  $X=g^x \bmod n$
- Bob选取一个大的随机数y，发送给Alice
  $Y=g^y \bmod n$
- Alice计算 $s=Y^x \bmod n$
- Bob计算 $t=X^y \bmod n$
- $s=t=g^{xy} \bmod n$

- 理论基础：离散对数计算的复杂性。 已知$g^x \bmod n$，求x是计算不可行的。

三方Diffie-Hellman算法

I. Alice选取一个大的随机数x，发送给Bob
$X=g^x \bmod n$

I. Bob选取一个大的随机数y，
$Y=g^y \bmod n$

I. Carol选取一个大的随机数z，发送给Alice
$Z=g^z \bmod n$

I. Alice计算 $Z'=Z^x \bmod n$，发送给Bob。

II. Bob计算 $X'=X^y \bmod n$，发送给Carol

III. Carol计算 $Y'=Y^Z \bmod n$，发送给Alice

IV. Alice计算 $k= Y'^x \bmod n$

V. Bob计算 $k= Z'^y \bmod n$

VI. Carol计算 $k= X'^z \bmod n$

其中 $k= g^{xyz} \bmod n$

Diffie-Hellman算法的中间人攻击

A		M		B
	g^x →		g^{Mx} →	
$g^{x \cdot My}$	← g^{My}	$g^{x \cdot My}$		
		$g^{y \cdot Mx}$	g^y ←	$g^{y \cdot Mx}$

工作站 – 工作站协议 (STS)

W.Diffie, P.C. van Oorschot, and M. Wiener. Authenticaiton and authenticated key Exchanges, Designs, Codes and Cryptography, 2: 107-125, 1992.

- 前提：
  - Alice和Bob各自拥有公钥证书：
    $Cert_A = sig_{CA}$ (Alice, P_A, desc <a>)
    $Cert_B = sig_{CA}$ (Alice, P_B, desc <a>)
    其中P_A, P_B, 分别表示Alice和Bob的公钥。
  - 同一系统中的用户共享一个高阶阿贝尔群desc <a>
  - 同一系统中的用户确定了一个对称密码算法
- 目标：
  - Alice 和Bob实现双方认证和双方认证的密钥交换

工作站 – 工作站协议 (STS)

- Alice 随机选择某个大整数x，并把以下消息发送给Bob

$$a^x$$

- Bob 随机选择某个大整数y，并把以下消息发送给Alice

$$a^y, Cert_B, E_K(sig_B(a^y, a^x))$$

- Alice 给Bob发送以下消息：

$$Cert_A, E_K(sig_A(a^x, a^y))$$

其中$K = a^{xy} = a^{yx}$

4　秘密分割与共享

秘密分割

（1）Trent产生一随机比特串R，和消息M一样长。

（2）Trent用R异或M得到S：

$$M \oplus R = S$$

（3）Trent把R给Alice，将S给Bob。

为了重构此消息，Alice和Bob只需一起做一步：

（4）Alice和Bob将他们的消息异或就可得到此消息：

$$R \oplus S = M.$$

秘密分割

1. Trent产生三个随机比特串R、S、T，每个随机串与消息M一样长。
2. Trent用这三个随机串和M异或得到U：
3. $M \oplus R \oplus S \oplus T = U$
4. Trent 将 U 给Alice，S给Bob，T给Carol，U给Dave。
5. Alice、Bob和Carol、Dave在一起可以重构此消息：

$$R \oplus S \oplus T \oplus U = M$$

秘密共享

- 怎样保存密钥？
  - 加密密钥
  - 放在保险柜中
  - 复制多份，放在不同的地方

秘密共享

- 秘密分割的要求
  - 将一个秘密数据分成n块：$D_1, D_2, \ldots, D_n$
  - 知道k个或更多的D_j就能有效地计算出D。
  - 知道k-1个或更少的D_j就不能有效地计算出D
- Shamir 称这种方法为(k,n)阈值方案

秘密共享

- 基于拉格郎日的插值多项式的方案：
  - 在二平面上给出k个点(x_i, y_i), i=1,2,…,k。
  - 其中x_i各不相同。
  - 则有一个且仅有一个k-1次多项式q(x)满足

$$q(x_i) = y_i, \quad i=1,2, \ldots, k$$

  - 我们选取一个随机k-1次多项式

$$q(x)=a_0+a_1x+\ldots+a_{k-1}x^{k-1}$$

  - 其中$a_0=D$, 并计算出

$$D_i=q(i), \quad i=1,\ldots, n$$

5 盲签名

盲签名问题

- 反间谍机构想为每一个特工签署一份文件
  - “这个签名文件的持有者____享有完全的外交豁免权，签名：外交部长。”
- 特工们不想把自己的化名告诉机构（减少危险）。
- 机构不想签署的文件是：
  - “特工___已经退休，并获得一年一百万美元的养老金，签名：外交部长。”

盲签名协议

- Bob 准备n份文件，每一个使用不同的化名，并给与那个特工外交豁免权。
- Bob 用不同的盲因子隐蔽每一个文件。
- Bob把隐蔽好的文件给Alice。
- Alice随即选择n-1份文件，并向Bob索要每份文件的盲因子。
- Bob向Alice发送适当的盲因子。
- Alice 去掉n-1份文件的盲因子，检验它们。
- Alice 在最后一份文件上签名并送给Bob。
- Bob在Alice签署的文件上去掉盲因子，得到签名文件。

盲签名算法

Bob有一个私钥为d，公钥为e，RSA模数为n。
Alice打算让Bob对消息m进行盲签名：

Bob不知道m的内容；
Alice可以向别人证明B签署了m。

(1) Alice在1至n之间选择一个随机值k， 计算：

$$t=mk^e \bmod n$$

(2) Bob 签署t： $t^d = (mk^e)^d \bmod n$

(3) Alice 通过下列计算揭开t^d ：

$$s = t^d/k \bmod n$$

得到： $s = m^d \bmod n$

$$t^d=(mk^e)^d = m^dk \bmod n$$
$$t^d/k = m^dk/k = m^d \bmod n$$

6. 不可抵赖的数字签名

6. 不可抵赖的数字签名

- 一般的数字签名可以被准确的复制
  - 宣传品的发布；
  - 承诺一个软件没有特罗伊木马；
- 问题
  - 商业信件：到处散播而且任何人都可以验证可能导致窘迫和勒索；
- 不可抵赖的数字签名
  - **没有签名者的合作，签名不能得到验证。**
- 应用
  - 软件公司可以在软件中附加签名证明软件中没有病毒，但非合法的买主无法得到验证。

不可抵赖的数字签名—特点

- 没有签名者的合作，签名就得不到验证；
  - 防止了由签名者签署的文档在没有经过他同意的情况而被复制和分发的可能性；
- 签名者可以执行一个否认协议，证明一个伪造的签名的确是伪造的：
  - 对于合法的用户可以要求签名者验证签名的有效的；如果签名是伪造的签名者可以裁判证明文件不是他签署的，所以可以不对文件负责；如果签名者拒绝执行否认协议，就说明签名者在抵赖。
  - 对于非法的用户，签名者有权拒绝验证；
- 不可抵赖的数字签名由签名算法、验证协议、否认协议组成。

不可抵赖的数字签名—签名、验证协议

- 公开一个大素数p, p=2q+1, q是一个素数， g是Z^*_p的一个阶为q的元素， G是以g为生成元的循环群(Z^*_p 的阶为q的乘法子群) 。
- Alice有一个私钥x， 1≤x≤q-1。
- 令β＝g^x(mod p)，Alice的公钥为：(p, g, β)
- Alice是签名者， Bob是签名验证者。

- **消息签名：** Alice计算 $z= m^x \pmod p$

- **消息验证：**

(1) Bob选择两个小于p的随机数a和b，发送给Alice：

$$c=z^a(g^x)^b \pmod p$$

(2) Alice计算$t=x^{-1} \bmod (q)$,并发送给Bob：$d=c^t \pmod p$

(3) Bob进一步确认：$d=m^a g^b \pmod p$，如果成立则接受。

$$c^t = (z^a(g^x)^b)^t= (m^{xa}(g^x)^b)^t= (m^a g^b)^{xt}= m^a g^b$$

不可抵赖的数字签名—可证明性

- 验证者必须亲自参加(2)的计算，证明者通过(1)、(3)的计算，构造(2)的值**$d=c^t$** 。
- 一个攻击者，随机选取的z，一般有 $z \neq m^x \pmod p$，
- 这个攻击者想欺骗验证者z是一个正确的签名，必须给出一个响应$d = c^t$,
- 攻击者在没有x 的情况下给出的响应d满足 $d = c^t$的概率为1/q。
- 即验证者将以1/q的概率错误地把z当作一个合法的签名接受。

- d 与 c 都是阶为q 的循环群中的元素，d恰好等于c^t的概率为1/q。

不可抵赖的数字签名—否认协议

1. Bob选择两个小于p的随机数a1和b1，发送给Alice： $c1=z^{a1}(g^x)^{b1} \pmod p$
2. Alice计算$t=x^{-1} \bmod (p-1)$,并发送给Bob：$d1=c1^t \pmod p$
3. Bob进一步确认是否成立：$d1 \neq m^{a1}g^{b1} \pmod p$，如果成立则转下一步。否则否认失败。
4. Bob选择两个小于p的随机数a2和b2，发送给Alice： $c2=z^{a2}(g^x)^{b2} \pmod p$
5. Alice计算$t=x^{-1} \bmod (p-1)$,并发送给Bob：$d2=c2^t \pmod p$
6. Bob进一步确认是否成立： $d2 \neq m^{a2}g^{b2} \pmod p$，如果成立则接受。
7. 当且仅当$(d1 \cdot g^{-b1})^{a1}=(d2g^{-b2})^{a1} \pmod p$时，验证者推断签名z是伪造的

不可抵赖的数字签名—否认协议

- 定理：如果$z \neq m^x \pmod p$，并且签名者和验证者都遵守协议，那么

$$(d1 \cdot g^{-b1})^{b1} = (d2 g^{-b2})^{a1} \pmod p$$

- 定理：如果$z = m^x \pmod p$，并且验证者遵守协议。如果**$d1 \neq m^{a1} g^{b1} \pmod p$**，并且**$d2 \neq m^{a2} g^{b2} \pmod p$**，那么**$(d1 \cdot g^{-b1})^{b1} \neq (d2 g^{-b2})^{a1} \pmod p$**的概率为**1-1/p**。

7. 同时签约

7.1 位承诺

- **Alice试图让Bob相信她有预测股赢利的能力**
  - **Alice说：“上月股票我选了这5只股票，都赢利了。我可以为你选择下个月的赢利的股票。”**
  - **Bob：“我怎样知道你在了解了上月股票的收益后没改变你上月选择的股票呢？如果你现在告诉我你选的股票，我就可以知道你不能改变他们。在我买你的方法以前我不在这些股票中投资。相信我。”**
  - **Alice：“我宁愿告诉你我上月选择的股票。我不会变，相信我。”**
- **Alice选择一个随机位，在揭示这个随机位之前Alice不能改变它，Bob无法知道它；**

位承诺—使用对称密码算法的比特承诺

1) Bob产生一个随机比特串R，并把R它发送给Alice。
2) Alice生成一个由她想承诺的比特b组成的消息（b实际上可能是几个比特），以及Bob的随机串。她用某个随机密钥K对它加密，并将结果送回给Bob：E_K（R，b）

这是这个协议的承诺部分，Bob不能解密消息，因而不知道比特为何。当到了Alice揭示她的比特的时候，协议继续：

1) Alice发送密钥给Bob。
2) Bob解密消息以揭示比特。他检测他的随机串R以证实比特b的有效性。

7.2 不经意传输

- Alice 发送2份消息。
- Bob只能收到其中一份，另外一份收不到。
- Alice不知道Bob是否收到了哪一份消息。

7.2 不经意传输

假设：非对称加密算法R， 对称加密算法E。

I. Alice产生两个公开密钥/私钥密钥对：p_1,s_1,p_2,s_2，总共四个密钥。她把两个公开密钥(p_1,p_2)发送给Bob。

II. Bob选择一个对称算法（例如DES）密钥k。他选择Alice的一个公开密钥并用它加密他的DES密钥

$$X=R(p, k), \quad p=p_1 \text{或} p_2$$

他把这个加了密的密钥X发送给Alice，且不告诉她DES密钥k是用她的哪一个公开密钥(p_1或p_2)加密的。

7.2 不经意传输

III. Alice解密Bob的密钥两次，每次用一个她的私钥来解密Bob的密钥。

$$Y_1=R(s_1, X)\ ,\quad Y_2=R(s_2, X)$$

Y_1和Y_2中有一个是k

- 在一种情况下，她使用了正确的密钥并成功地解密Bob的DES密钥。
- 在另一种情况下，她使用了错误的密钥，只是产生了一堆毫无意义，而看上去又象一个随机DES密钥的比特。由于她不知道正确明文，故她不知道哪个是正确的。

7.2 不经意传输

IV. Alice加密她的两份消息，每一份用一个不同的在上一步中产生的DES密钥（一个真的和一个毫无意义的），并把两份消息都发送给Bob。

$$Z_1=E(Y_1, M)\ ,\ Z_2=E(Y_2, M)$$

IV. Bob收到一份用正确DES密钥加密的消息及一份用无意义DES密钥加密的消息。当Bob用他的DES密钥解密每一份消息时，他能读其中之一，另一份在他看起来是毫无意义的。

$$N_1=E(Y_1, Z_1)\ N_2=E(Y_2, Z_2)$$

- N_1和N_2中有一个是M，另一个是无意义的(如果在II中$X=R(p_1, k)$，则N_1时真的消息，如果在II 中$X=R(p_2, k)$，则N_2时真的消息。Bob知道，Alice不知道。

7.3 无需仲裁者的同时签约

I. Alice和Bob将消息分成n对，L_i和R_i, i = 1, …,n。

II. Alice和Bob二者随机选择2n个DES密钥，分成一对对的。

III. Alice和Bob二者用每个DES密钥对加密他们的消息对，左半消息用密钥对中的左密钥，右半消息用密钥对中的右密钥。

IV. Alice和Bob相互发送给对方2n份加密消息。

V. Alice和Bob利用不经意传输协议相互送给对方n对密钥，对方都得到这n对密钥中各对中的一个。现在Alice和Bob都有每一对密钥中的一个密钥，但都不知道对方有哪一半。

VI. Alice和Bob用收到的密钥解密他们能解的那一半消息。他们确信解密消息是有效的。(各方都能验证是否收到了正确的一半消息);

无需仲裁者的同时签约

VI. Alice和Bob都把所有2n个DES密钥的按照次序第一个比特、第二个比特，…，第n个比特发送给对方。

VII. Alice和Bob解密剩余一半消息对，合约被签署。

VIII. Alice和Bob交换在第V步的不经意传输中使用的私钥，并各方验证对方没有欺骗。

第1轮： 11，21，31，41，51，61

第2轮： 12，22，32，42，52，62

.....

7.4 数字证明邮件

- 问题
  - Alice要把一条消息送给Bob，但如果没有签名的收条，Bob就读不出 。
  - Alice用密钥加密邮件。Alice向B要签名的收条，Bob向Alice要解密密钥。
  - 用作签约的同时不经意传输协议也可以用于计算机证明邮件，但要做一些修改。

分析

- 为什么这个协议不能工作？
  - 在第V步中的不经意传输保证双方是诚实的。他们两人都知道他们发送给另一方一个有效的半密钥，但都不知道是哪一半。他们在第VII步中没有进行欺骗是因为做了坏事而不被发觉的机会太小。
  - 如果Alice要发送给Bob的不是一份消息，而是一个DES密钥的一半，**在第VI步中Bob没有办法检查这个DES密钥的有效性。**

协议

I. Alice用一个随机的DES密钥k加密她的消息，并把它发送给Bob。

II. Alice产生n对DES密钥，lk_i, rk_i, i=1,…,n 。 lk_i是随机产生的； $rk_i = lk_i \oplus k$, i=1,…,n 。

III. Alice用她的2n个密钥的每一个加密一份假消息。

IV. Alice把所有加密消息都发送给Bob，保证他知道哪些消息是哪一对的哪一半。

协议(续1)

V. Bob产生n对随机DES密钥。

VI. Bob产生一对指明一个有效收条的消息。比较好的消息可以是“这是我收条的左半”和“这是我收条的右半”，再附加上某种类型的随机比特串。他做了n个收条对，每个都编上号。如同先前的协议一样，如果Alice能产生一个收条的两半（编号相同）和她的所有加密密钥，这个收条被认为是有效的。

VII. Bob用DES密钥对加密他的每一对消息，第i份消息用第i个密钥，左半消息用密钥对中的左密钥，右半消息用密钥对中的右密钥。

VIII. Bob把他的消息对发送给Alice，保证Alice知道哪些消息是哪一对的哪一半。

协议(续2)

IX. Alice和Bob利用不经意传输协议发送给对方每个密钥对。那就是说，对n对中的每一对而言，Alice或者送给Bob用来加密左半消息的密钥，或者送给Bob用来加密右半消息的密钥。Bob也同样这么做。他们可以或者交替传送这些一半，或者一方发送n个，然后另一方再发送n个这都没有关系。现在Alice和Bob都有了每个密钥对中的一个密钥，但是都不知道对方有哪些一半。

X. Alice和Bob都解密他们能解的那些一半，并保证解密消息是有效的。

协议(续3)

XI. Alice和Bob送给对方所有2n个DES密钥中的第一个比特（如果他们担心Eve可能会读到这个邮件消息，那么他们应当对相互的传输加密）。

XII. Alice和Bob对所有2n个DES密钥中的第二比特、第三比特都重复第（XI）步，如此继续下去，直到 所有DES密钥的所有比特都传送完。

XIII. Alice和Bob解密消息对中的余下一半。Alice有了一张来自Bob的有效收条，而Bob能异或任一密钥对以得到原始消息加密密钥。

XIV. Alice和Bob交换在不经意传输协议期间使用的私钥，同时每一方验证另一方没有进行欺骗。

其他密码协议

- 团体签名协议
- 会议密钥交换协议
- 电子投票协议
- 数字现金协议
- 电子支票协议
- ……

8. Kerberos

动机

- 分布式环境下提供安全鉴别的三种方法：
  - 依靠每个客户工作站来确保用户的身份，并依靠每个服务器通过基于用户身份标识来强化安全策略。
  - 需要客户系统向服务器证实它们自己的身份，但要信任客户系统的用户身份。
  - 调用每项服务时需要用户证明身份。也需要这些服务器向客户证明它们的身份。

Kerberos 需求

1. 安全 一个网络窃听者应该不能获得必要的信息来假装成另一个用户。更一般地，Kerberos应该足够强以防止潜在的对手发现脆弱的链路。

2. 可靠 对所有依赖Kerberos进行访问控制的服务来说，无法获得Kerberos服务就意味着无法获得所要求的服务。因此，Kerberos应该是高可靠性的，应该使用分布式的服务器结构，一个系统能够对另一个系统进行备份。

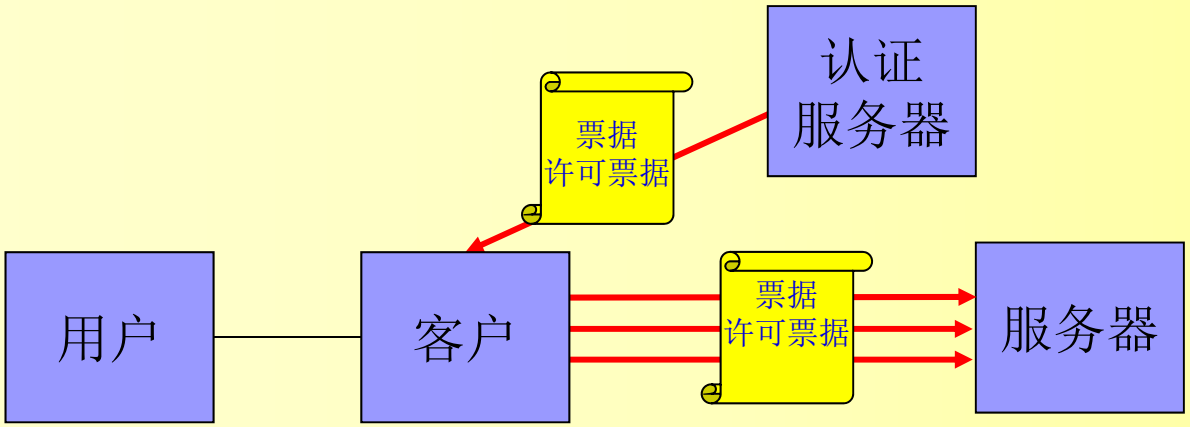
3. 透明 理想情况下，除了需要输入一个口令外，用户应该没有意识到鉴别服务的发生。

4. 可扩缩 系统应该拥有支持大量客户和服务器的能力。这意味着需要一个模块化的、分布式结构。

Kerberos协议票据许可票据

我们更希望让用户必须输入口令的次数最小。

- 假定每次鉴别只能请求一次服务，如果C打算在一天中多次检查邮件，每次尝试都必须进行鉴别。
- 对一次登录会话，用户向客户端证实自己。
- 客户可以向鉴别服务器证实自己，鉴别服务器向用户提供一张票据许可票据，客户可以保存收到的票据许可票据。
- 每次请求邮件服务时向邮件服务器出示这张票据许可票据，并用它代表用户来多次访问邮件服务器。

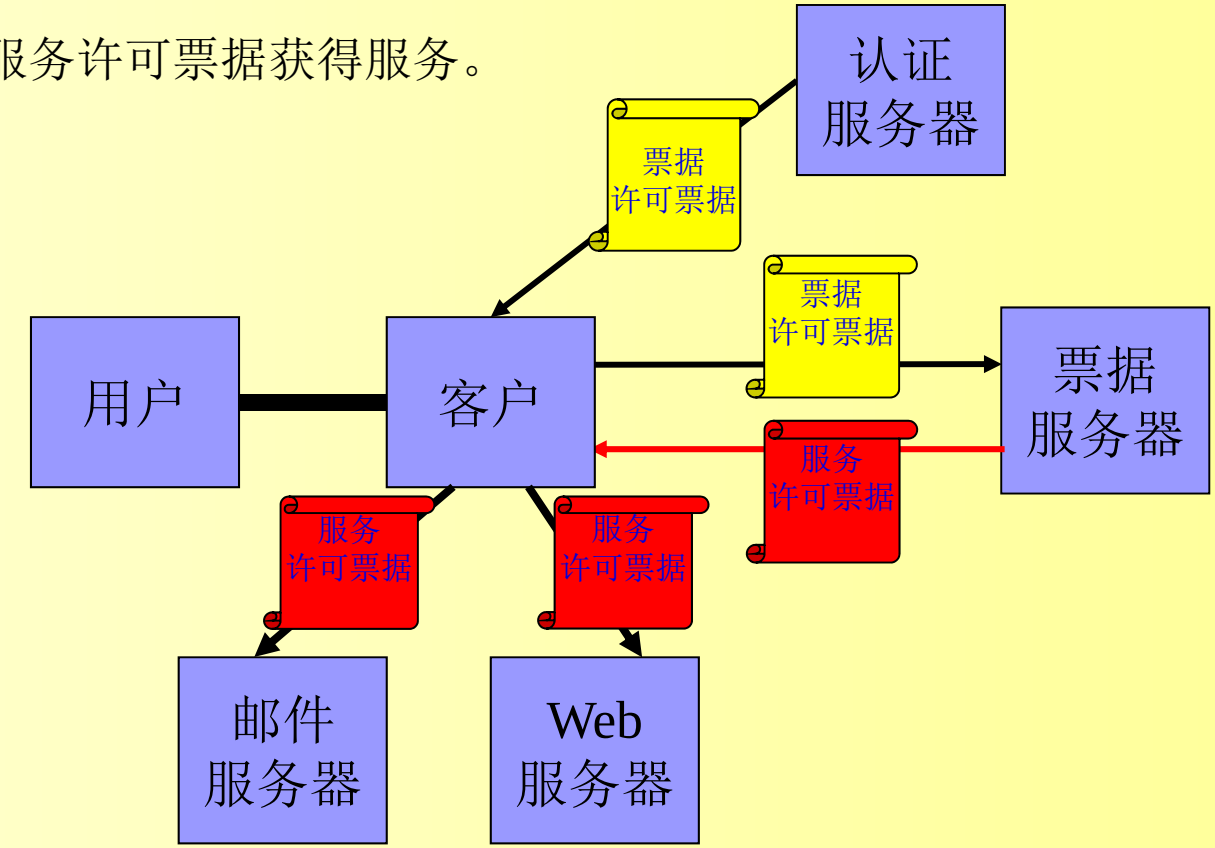


这个方案对不同的服务用户需要新的票据的问题。

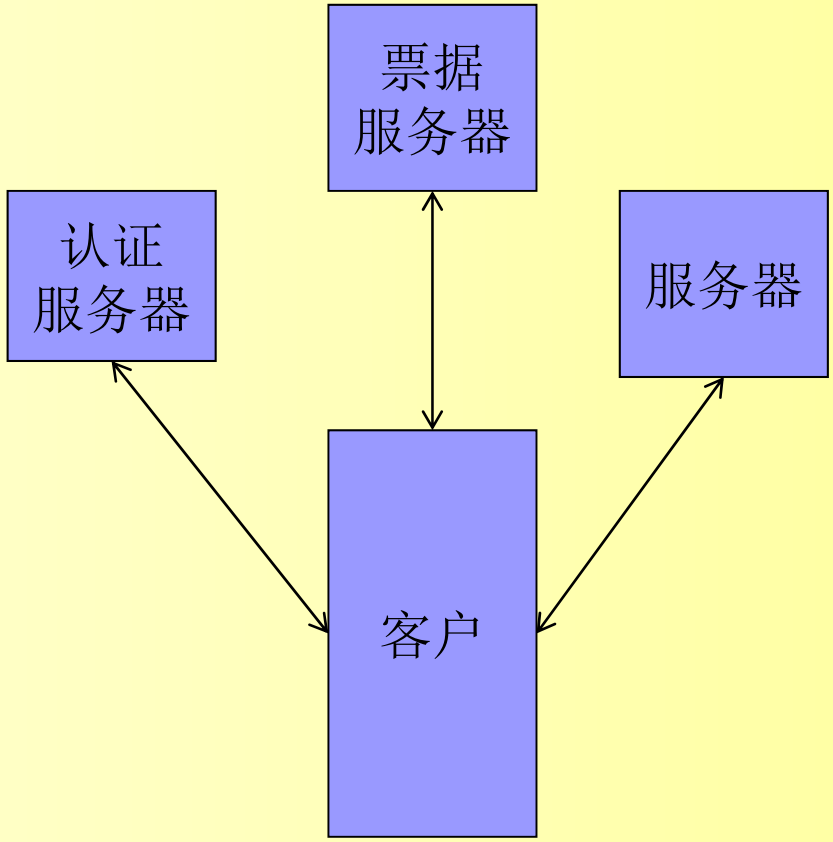
如果用户希望访问打印服务器、邮件服务器、文件服务器等等，每个服务的第一次访问都需要一个新的票据，因此需要用户输入口令。

Kerberos协议服务许可票据

- 客户向鉴别服务器鉴别自己后，获得一张票据许可票据。
- 客户利用票据许可票据获得服务许可票据。
- 客户利用服务许可票据获得服务。



Kerberos——地狱之门守护者：三个头的狗



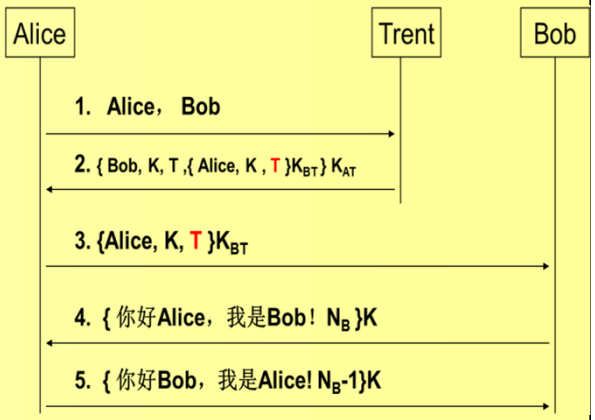
Kerberos这一名词来源于希腊神话
“地狱之门守护者：三个头的狗”

Kerberos 第4版

(a) 鉴别服务交换：获得票据许可票据				
(1)	**C**	→	$ID_C \| ID_{tgs} \| TS_1$	**AS**
(2)	**C**	←	$\{ K_{C,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs} \} K_c$ $Ticket_{tgs} = \{ K_{C,tgs} \| ID_c \| AD_c \| ID_{tgs} \| TS_2 \| Lifetime_2 \} K_{tgs}$	**AS**
(b) 票据许可交换服务：获得服务许可票据				
(3)	**C**	→	$IDv \| Ticket_{tgs} \| Authenticatorc$ $Authenticatorc = \{ IDc \| ADc \| TS3 \} K_{c,tgs}$	**TGS**
(4)	**C**	←	$\{ K_{e,v} \| ID_v \| TS_4 \| Ticket_v \} K_{c\ tgs}$ $Ticket_v = \{ K_{C,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime_4 \} K_v$	**TGS**
(c) 客户/服务器鉴别交换：获得服务				
(5)	**C**	→	$Ticket_v \| Authenticator_c$ $Authenticator_c = \{ ID_c \| AD_c \| TS_5 \} k_{c,v}$	**V**
(6)	**C**	←	$\{ TS_5+1 \} K_{c,v}$ （对于相互鉴别） $Ticket_v = \{ K_{C,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime_4 \} K_v$	**V**

Kerberos 第4版

(a) 鉴别服务交换：获得票据许可票据				
(1)	**C**	→	$ID_C \| ID_{tgs} \| TS_1$	**AS**
(2)	**C**	←	$\{ K_{C,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs} \} K_c$ $Ticket_{tgs} = \{ K_{C,tgs} \| ID_c \| AD_c \| ID_{tgs} \| TS_2 \| Lifetime_2 \} K_{tgs}$	**AS**
(b) 票据许可交换服务：获得服务许可票据				
(3)	**C**	→	$IDv \| Ticket_{tgs} \| Authenticatorc$ $Authenticatorc = \{ IDc \| ADc \| TS3 \} K_{c,tgs}$	**TGS**
(4)	**C**	←	$\{ K_{e,v} \| ID_v \| TS_4 \| Ticket_v \} K_{c\ tgs}$ $Ticket_v = \{ K_{C,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime_4 \} K_v$	**TGS**
(c) 客户/服务器鉴别交换：获得服务				
(5)	**C**	→	$Ticket_v \| Authenticator_c$ $Authenticator_c = \{ ID_c \| AD_c \| TS_5 \} k_{c,v}$	**V**
(6)	**C**	←	$\{ TS_5+1 \} K_{c,v}$ （对于相互鉴别） $Ticket_v = \{ K_{C,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime_4 \} K_v$	**V**



鉴别服务交换

(1) $ID_C \| ID_{tgs} \| TS_1$

(2) $\{ K_{C,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs} \} K_c$

$Ticket_{tgs} = \{ K_{C,tgs} \| ID_c \| AD_c \| ID_{tgs} \| TS_2 \| Lifetime_2 \} K_{tgs}$

报文(1)　客户请求票据许可票据

IDc：　从这个客户将用户的身份标识告诉AS

ID_{tgs}：　告诉AS用户请求TGS

TS_1：　允许AS验证客户的时钟是否与AS的时钟同步

采用了时间戳TS_1，以便让AS知道报文是及时的。

报文(2)　AS返回票据许可票据

K_c：　基于用户口令的加密，使AS和客户能验证口令，并保护报文(2)的内容。

$K_{c,tgs}$：　客户可理解的会话密钥的副本；由AS生成允许客户与TGS间无需共享一个永久密钥就能安全交换报文，使得C与TGS之间的交换可以防止重放攻击。

ID_{tgs}：　证实这张票据是给TGS的。

TS_2：　通知客户这张票据发出的时间

$Lifetime_2$：通知客户这张票据的有效期

$Ticket_{tgs}$：客户用来访问TGS的票据

票据许可服务交换

(3) $ID_v \| Ticket_{tgs} \| Authenticator_c$ $\quad Authenticator_c = \{ ID_c \| AD_c \| TS_3 \} K_{c,tgs}$

(4) $\{ K_{e,v} \| ID_v \| TS_4 \| Ticket_v \} K_{c\,tgs}$

$Ticket_v = \{ K_{C,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime_4 \} K_v$

- 报文(3) 客户请求服务许可票据
  - ID_v: 告诉TGS用户请求访问的服务器v
  - $Ticket_{tgs}$: 使TGS确信这个用户已经经过AS的鉴别
  - $Authenticator_c$:由客户产生用来证明票据的有效性，这可以用来防止者重放票据许可票据攻击
- 报文(4) TGS返回的服务许可票据
  - $K_{c,tgs}$： 只有C可理解的会话密钥的副本；用来保护报文(4)。由TGS生成， 允许 客户与服务器间无需共享一个永久密钥就能安全交换报文
  - ID_v： 证实这张票据是给服务器v的
  - TS_4： 通知客户这张票据发出的时间
  - $Ticket_v$：客户用来访问服务器V的票据

$Ticket_{tgs}$ 可重用的，以便用户不用重新输人口令

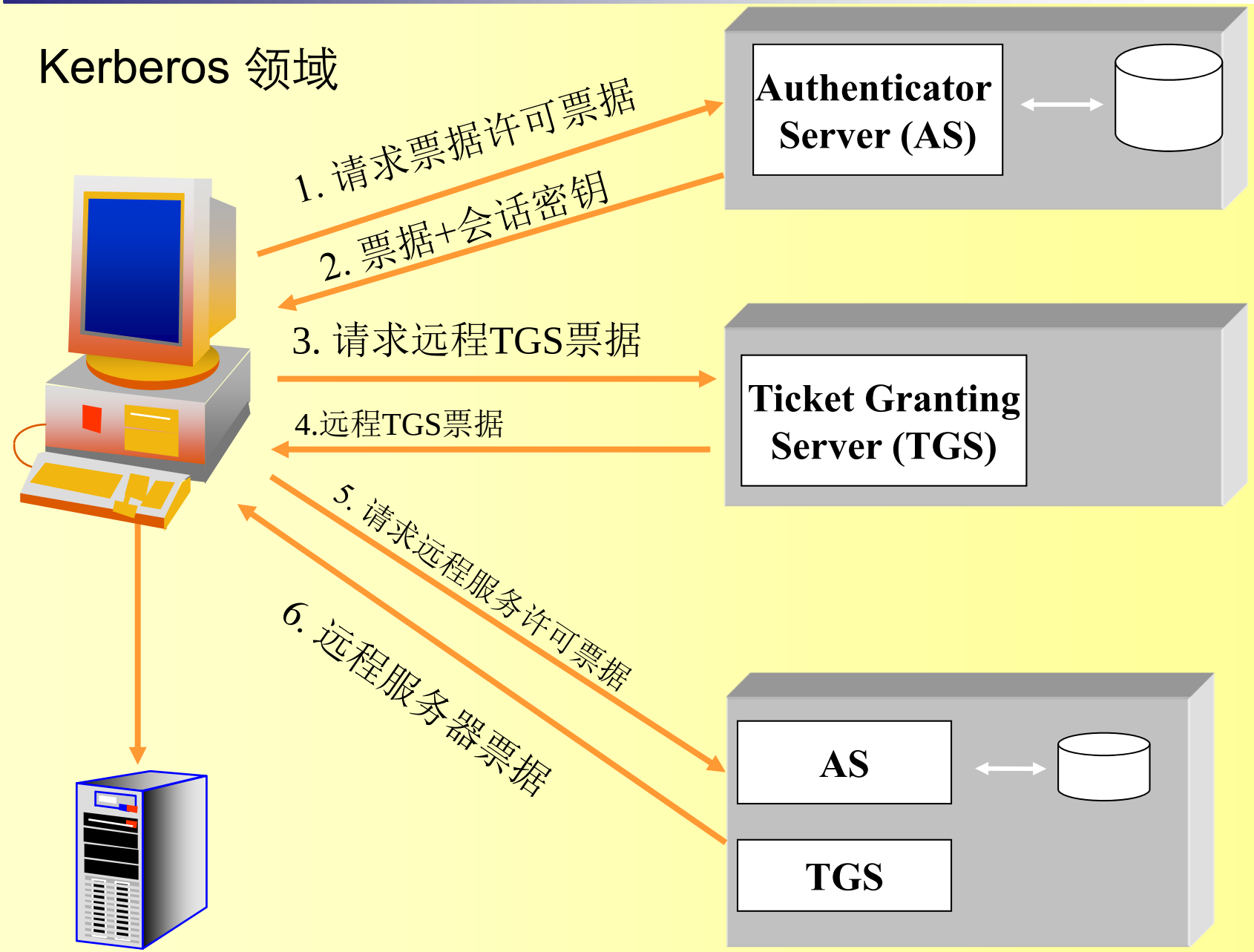
ID_c: 指明这张票据的合法拥有者

AD_c: 防止在另一台工作站上使用该票据的人不是票据的初始申请者

ID_{tgs} : 使服务器确信解密正确

TS_2: 通知TGS张票据发出的时间

ID_c: 必须与票据中的，D匹配来鉴别这张票据


Kerberos 领域
Authenticator Server (AS)
1. 请求票据许可票据
2. 票据+会话密钥
3. 请求远程TGS票据
Ticket Granting Server (TGS)
4.远程TGS票据
5. 请求远程服务许可票据
6. 远程服务器票据
AS
TGS

Kerberos领域和多个Kerberi

- 完整服务的Kerberos环境包含一个Kerberos服务器、许多客户和许多应用服务器。Kerberos环境有如下需求：
  - Kerberos服务器的数据库必须包含所有参与用户的ID和用户口令的散列码。所有用户都向Kerberos服务器注册。
  - Kerberos服务器必须与每个服务器共享一个密钥。所有服务器都向Kerberos服务器件注册。
  - 这样的环境被称为一个领域。不同行政组织下的由客户和服务器组成的网自成不同的领域。
  - 这通常是不切实际的，或因不能肯定管理的策略，让在一个管理域下的用户和服务器向别处的Kerberos服务器注册。
  - 然而，在一个领域中用户可能需要访问他领域的服务器，同时某些服务器可能也愿意向其他领域的用户提供服务，如果这些用户月过鉴别的。

Kerberos 领域

- Kerberos提供了一种支持不同领域间鉴别的机制。
  - 每个互操作领域中的Kerberos服务器要与另一个领域中的Kerberos服务器共享一个密钥。两个Kerberos服务器都必须相互注册。

  这个方案需要一个领域中的Kerberos服务器信任另一个领域中Kerberos服务器鉴别的用户，并且第二个领域的参与服务器也愿意信任第一个领域中的Kerberos服务器。

第4版与第5版的差异

- 第5版打算两个方面改进第4版的不足：
  - 环境上的缺点
  - 技术上的不足

第四版的缺陷

1. 加密系统的依赖性

第4版需要使用DES。DES的出口限制以及对DES强度的怀疑都是备受关注的。在第5版中，密文被附上加密类型标识符以便可以使用任何加密技术。加密密钥也加上类型和长度标记，允许不同的算法使用同样的密钥，允许说明给定算法的不同版本。

2. 对Internet协议的依赖性

第4版需要使用网际协议(IP)地址。不提供其他地址类型，如ISO网络地址。第5版中的网络地址加上了类型和长度标记，允许使用任何类型的网络地址。

第四版的缺陷(续1)

3. 报文字节序

在第4版中，发方的报文使用的字节序由它自己选择，可在报文中说明是低位字在低地址或者高位字在低地址。这种技术能发挥作用，但不遵守已建立的约定。在第5版中，所有报文结构采用抽象语法记法(ASN．1)和基本编码规则(BER)，这将提供一个明确的字节序。

4. 票据有效期

有效期的值在第4版中是用8bit量编码的，每个单位是5分钟。这样最长的有效期可表示为28x 5；1280分钟，或者21小时多一点。这对某些应用来说可能是不够的(如运行时间很长的仿真在整个执行期间都需要Kerberos的鉴别证书)。在第5版中，票据包含显式的开始时间和结束时间，允许票据有任意大小的有效期。

第四版的缺陷(续2)

5. 鉴别的转发

第4版不允许发给一个用户的鉴别证书被转发到其他主机或被其他客户使用。这种能力可以使一个客户访问某服务器时，该服务器能以这个客户的名义访问另一个服务器。例如，一个客户向一个打印服务器发出请求，然后使用客户的访问鉴别证书从一个文件服务器访问该客户的文件。第5版提供了这种能力。

6. 领域间鉴别

在第4版中，N个领域间的互操作需要如前面介绍的N^2数量级的Kerberos到Kerboeros的联系。第5版支持一种需要更少关系的方法。

第四版的缺陷 — 技术上的不足

1. 双重加密

Kerberos 4，给客户的票据被加密了两次，一次采用目标服务器的密钥，然后再采用客户知道的密钥。第二次加密是不必要的，造成计算上的浪费。

$$\{ K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs} \} K_c$$

$$Ticket_{tgs} = \{ K_{c,tgs} \| ID_c \| AD_c \| ID_{tgs} \| TS_2 \| Lifetime_2 \} K_{tgs}$$

第四版的缺陷 — 技术上的不足

2. 会话密钥

每张票据包括一个会话密钥，该密钥供客户对发往与该票据有关服务器的鉴别符进行加密。此外，客户与服务器随后可能要使用这个会话密钥来保护会话过程中的报文。

(3) $ID_v \| Ticket_{tgs} \| Authenticator_c$

$Authenticator_c = \{ ID_c \| AD_c \| TS_3 \} K_{c,tgs}$

(4) $\{ K_{e,v} \| ID_v \| TS_4 \| Ticket_v \} K_{c\ tgs}$

$Ticket_v = \{ K_{C,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime_4 \} K_v$

第四版的缺陷 — 技术上的不足

2. 会话密钥

每张票据包括一个会话密钥，该密钥供客户对发往与该票据有关服务器的鉴别符进行加密。此外，客户与服务器随后可能要使用这个会话密钥来保护会话过程中的报文。

(3) $ID_v \| Ticket_{tgs} \| Authenticator_c$

$Authenticator_c = \{ ID_c \| AD_c \| TS_3 \} K_{c,tgs}$

(4) $\{ K_{c,v} \| ID_v \| TS_4 \| Ticket_v \} K_{c\ tgs}$

$Ticket_v = \{ K_{C,v} \| ID_c \| AD_c \| ID_v \| TS_4 \| Lifetime_4 \} K_v$

然而，因为同一票据会被多次用作访问一个特定的服务器，存在对手使用与该客户或该服务器旧会话的报文进行重放攻击的危险。在第5版中，客户和服务器协商一个仅用于那个连接的子会话密钥是可能的。客户进行新的访问将导致使用新的子会话密钥。

第四版的缺陷(续4)

- 口令攻击：两个版本都容易遭到口令攻击。
  - 从AS发往客户的报文中包含用基于客户口令的密钥加密的内容。对手可以截获这个报文，并试图通过尝试不同的口令来对报文解密。如果测试解密的结果是正确的形式，那么这个对手已经发现了该客户的口令并可能随后用它获得Kerberos的鉴别证书。

$$\{K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}\} K_c$$

  - 第5版提供了称为预鉴别(preauthentication)的机制，使口令攻击变得更加困难，但并不能避免口令攻击。

Kerberos 第5版

(a) 鉴别服务交换：获得服务许可票据

(1) C→AS: $Options \| ID_C \| Realm_c \| ID_{tgs} \| Times \| Nonce_1$

(2) AS→C: $Realm_c \| ID_C \| Ticket_{tgs} \| \{ K_{C,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs} \} K_c$

$Ticket_{tgs} = \{ Flags \| K_{C,tgs} \| Realm_c \| ID_c \| AD_c \| Times \} K_{tgs}$

(b) 票据许可交换服务：获得服务许可票据

(3) $C \rightarrow TGS: Options \| ID_v \| Times \| Nonce_2 \| Ticket_{tgs} \| Authenticator_c$

(4) $TGS \rightarrow C: Reaml_c \| IDc \| Ticket_v \| \{ K_{e,v} \| Times \| Nonce_2 \| Realm_v \| ID_v \} K_{c\ tgs}$

$Ticket_{tgs} = \{ Flags \| K_{C,tgs} \| Realm_c \| ID_c \| AD_c \| Times \} K_{tgs}$

$Ticket_v = \{ Flags \| K_{C,v} \| Realm_c \| ID_c \| AD_c \| Times \} K_v$

$Authenticator_c = \{ ID_c \| Realm_c \| TS_1 \} K_{c,tgs}$

(c) 客户/服务器鉴别交换：获得服务

(5) $C \rightarrow K: Options \| Ticket_v \| Authenticator_c$

(6) $K \rightarrow C: \{ TS_2 \| Subkey \| Seq\# \} K_{c,v}$

$Ticket_v = \{ Flags \| K_{C,v} \| Realm_c \| ID_c \| AD_c \| Times \} K_v$

$Authenticator_c = \{ ID_c\ Realm_c \| TS_2 \| Subkey \| Seq\# \} K_{c,v}$

Kerberos

- Windows2000和后续的操作系统都默认Kerberos为其默认认证方法。RFC3244记录整理了微软的一些对Kerberos协议软件包的添加。
- RFC4757“微软Windows2000Kerberos修改密码并设定密码协议”记录整理了微软用RC4密码的使用。虽然微软使用了Kerberos协议，却并没有用麻省理工的软件。
- 苹果的Mac OS X也使用了Kerberos的客户和服务器版本。
- Red Hat Enterprise Linux4 和后续的操作系统使用了Kerberos的客户和服务器版本。

实践课题

- Windows 系统的安全
  - 分析和配置使用Windows系统的活动目录、Kerberos认证。
  - Windows系统中的证书服务系统的原理、配置使用。

- 参考文献
  - **Ed Bott, Carl Siechert 著， 梁超，李钦，江楠，树军 译，Windows 2000和Windows XP 安全技术，清华大学出版社，2003年6月第1版。ISBN 7-89494-093-3。**
  - **程迎春编著，Windows安全，应用策略和实施方案手册，人民邮电出版社，2005年5月第1版，ISBN 7-1115-13165—1/TP·4495。**

备份

3.4 双方认证

3.4 双方认证

ISO 公钥三次传输双方认证协议

前提：A拥有公钥证书$Cert_A$, B拥有公钥证书$Cert_B$；

目标：A和B完成双方认证；

1. B → A：R_B
2. A → B：$Cert_A$, TokenAB
3. B → A：$Cert_B$, TokenBA

$TokenAB= R_A || R_B || B || sig_A (R_A||R_B||B)$

$TokenBA= R_B || R_A || A || sig_B (R_B||R_A||A)$

3.4 双方认证

- 早期的标准草案中

$$TokenBA = R'_B \,||\, R_A \,||\, A \,||\, sig_B\,(R'_B||R_A||A)$$

- 目的在于防止B会对一个被A完全所知的字符串签名。

Wiener攻击(加拿大人攻击)

- 前提：A、B、M拥有公钥证书$Cert_A$、$Cert_B$、$Cert_M$。
- 目标：M冒充B和A完成双方认证。

$M(B) \rightarrow A \quad R_B$

$A \rightarrow M(B) \quad R_A \| R_B \| B \| sig_A(R_A \| R_B \| B)$

$M(A) \rightarrow B \quad R_A$

$M(B) \rightarrow A \quad R'_B \| R_A \| A \| sig_B(R'_B \| R_A \| A)$

$B \rightarrow M(A) \quad R'_B \| R_A \| A \| sig_B(R'_B \| R_A \| A)$

- 在发现Wiener攻击以后，ISO／IEC 9798系列用于标准化认证协议时对双方认证开始采取了谨慎的方法。

  如果TokenAB在某个单方认证协议中出现，而双方认证又是在该单方认证版本的基础上发展的，

  那么在该双方认证中TokenAB的匹配TokenBA用于双方认证时会包含一个到TokenAB的上下文相关链接，该链接通常是通过重用相同的(也就是，当前的)运行中的新鲜性标志符来实现。

3.5 平行会话攻击

Woo – Lam协议— 包含可信第三方的认证

- 前提：A和T共享对称密钥K_{AT}; B和T共享对称密钥K_{BT};
- 目标：A向B证实自己

1. A →B：Alice
2. B →A：N_B
3. A →B：$\{N_B\}K_{AT}$
4. B →T：$\{Alice||\{N_B\}K_{AT}\}K_{BT}$
5. T →B：$\{N_B\}K_{BT}$
6. Bob使用密钥K_{BT}解密密文分组，如果等于N_B则接受Alice,否则拒绝接受Alice。

对Woo-Lam的平行会话攻击

- A和T共享对称密钥K_{AT}; B和T共享对称密钥K_{BT}; M和T共享对称密钥K_{MT};

Malice冒充Alice 与Bob通信		Malice 与 Bob	
1. M(A) →B:	Alice		
2. B →M(A):	N_B		
3. M(A) →B:	$\{N_B\}K_{MT}$		
4. B →T:	$\{A, \{N_B\}K_{MT}\}K_{BT}$		
		1. M →B:	Malice
		2. B →M:	N_B'
		3. M →B:	**$\{N_B\}K_{MT}$**
		4. B →T:	$\{M, \{N_B\}K_{MT}\}K_{BT}$
5. T →B:	{“垃圾”}K_{BT}, $\{N_B\}K_{BT}$	5. T →B:	$\{N_B\}K_{BT} \neq N_B'$
6. **Bob**接受和“**A**”的运行，实际和 **M**运行。 因为**Bob**收到了N_B是与”**Alice**”通信中选择的新鲜数。		**6.** B拒绝和M的运行 因为**Bob**没有收到期望的N'_B **Bob**认为与**M**的通信收到了垃圾 = $\{\{N_B\}K_{MT}\}K_{AT}$	

- 问题：
  - Woo – Lam协议中利用加密的方法进行认证；
  - 提供了解密服务：$\{N_B\}K_{BT}$
  - 没有提供完整性服务：

    Malice与B 的通信信道中T的响应$\{N_B\}K_{BT}$ 被认为是Malice冒充Alice 与Bob通信信道中的T响应。

Woo – Lam协议的改进

- 前提：A和T共享对称密钥K_{AT}; B和T共享对称密钥K_{BT};
- 目标：A向B证实自己

1. A →B：Alice
2. B →A：N_B
3. A →B：$[N_B]K_{AT}$
4. B →T：$[Alice, N_B, [N_B]K_{AT}]K_{BT}$
5. T →B：T用K_{AT} $[N_B]$ K_{AT}是否是N_B的消息鉴别码，是则发送$[N_B]K_{BT}$
6. 如果$[N_B]K_B$通过完整性验证则接受Alice,否则拒绝接受Alice。

Malice冒充Alice 与Bob	Malice 与 Bob
1. M(A) →B：Alice	1'. M →B：Malice
2. B →M(A)：N_B	2'. B →M：N_B'
3. M(A) →B：$[N_B]K_{MT}$	3'. M →B：$[N_B]K_{MT}$
4. B →T: $[A, N_B, [N_B]K_{MT}]K_{BT}$	4'. B →T：$[M, N'_B, [N_B]K_{MT}]K_{BT}$
5. T 用K_{AT}验证$[N_B]K_{MT}$发现错误。	5'. T 用$[N_B]K_{MT}$验证N'_B, 时发现错误
6. Bob拒绝	6'. Bob拒绝。