安全协议设计与分析：

第五讲　基础安全协议-认证协议

- 李晖
- 网络空间安全学院

本讲内容

1. 认证协议的基本概念
2. 认证协议的基本技术
3. 经典认证协议

1. 认证协议的基本概念

- 认证（authentication）

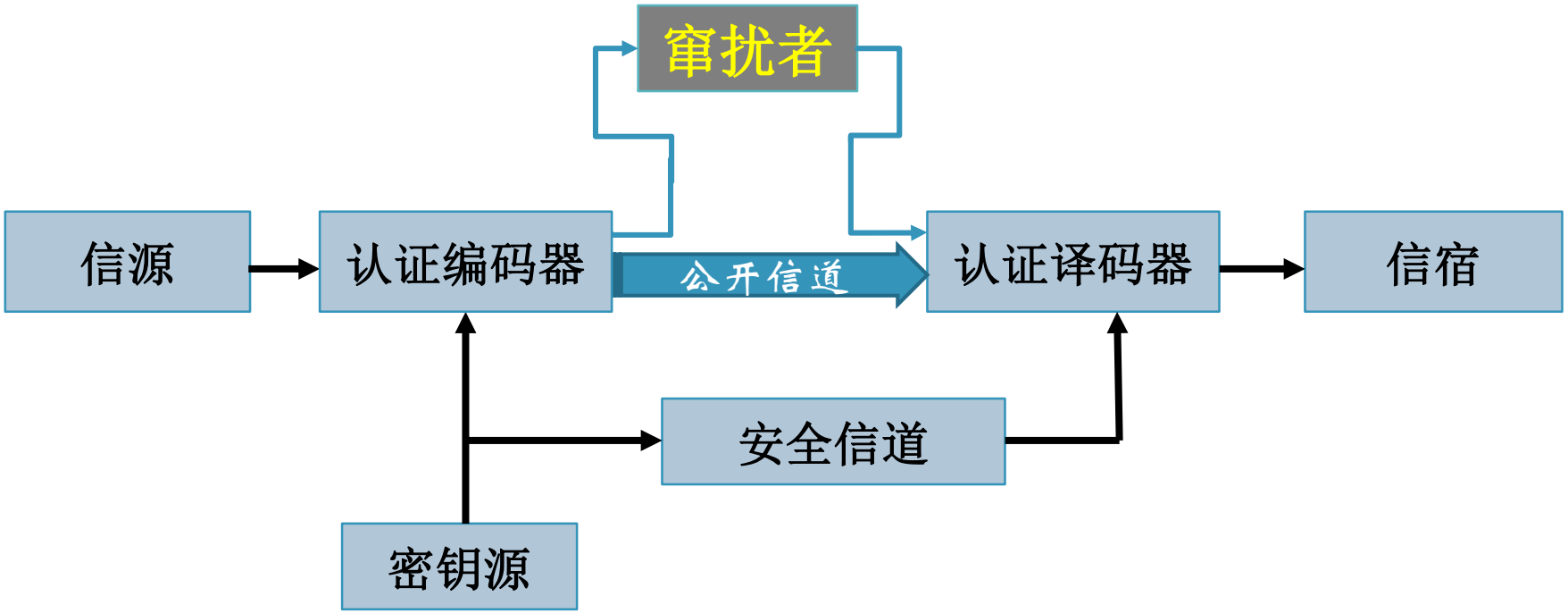
是防止主动攻击的重要技术，它对于开放的网络中的各种信息系统的安全性有重要作用。

- 认证的主要目的：
  - **实体认证**（entity authentication）：验证信息的发送者/接收者是真正的，而不是冒充的，此为信源/信宿识别；
  - **消息认证**（message authentication）：验证信息的**来源和目的地**，验证消息在传送或存储过程中**未被篡改、重放或延迟**等。

保密 ≠ 认证

认证系统基本模型

- 一个纯**认证系统**的模型如下图所示：

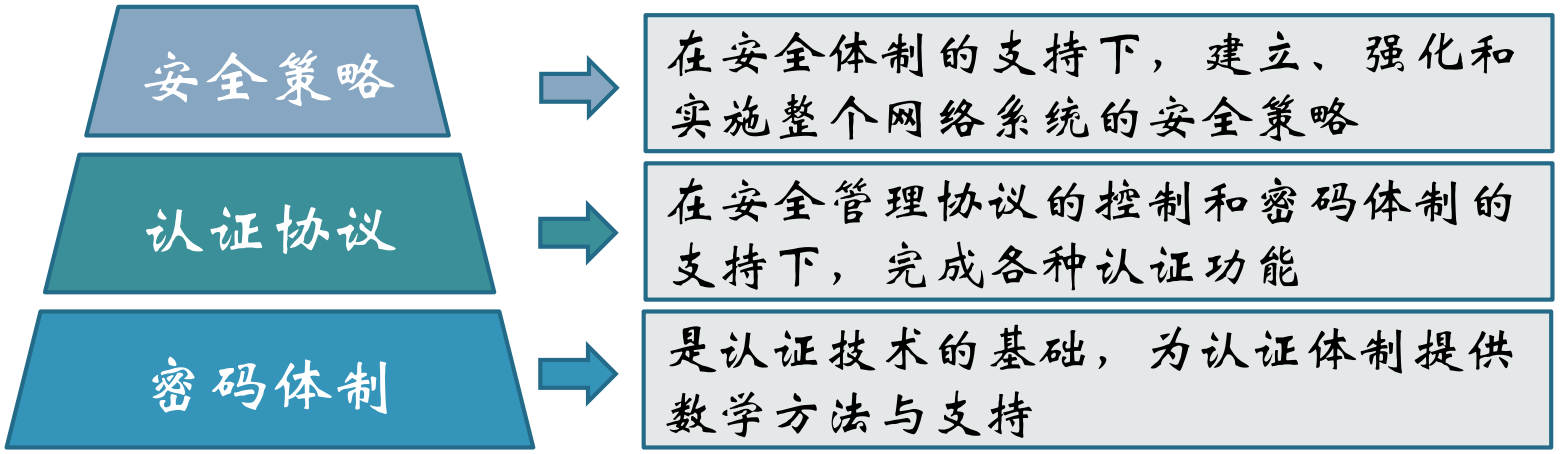


本讲内容

1. 认证协议的基本概念
2. 认证协议的基本技术
3. 经典认证协议

2. 认证技术

- 认证技术的三个层次



- 一个安全的认证系统
  - 首先选择恰当的**认证函数**
  - 然后给出合理的**认证协议**(Authentication Protocol)。
  - 最后制定理想的**安全策略**

2. 认证技术

➢ 认证函数（Authentication Functions)

对称密码体制下可用来做认证的函数分为三类：

(1) 信息加密函数(Message encryption)

用完整信息的密文作为对信息的认证。

(2) 信息认证码MAC(Message Authentication Code)

是对信源消息的一个编码函数。

(3) 散列函数(Hash Function)

是一个公开的函数，它将任意长的信息映射成一个固定长度的信息。

非对称密码体制下可用来做认证的函数：

(4)基于公钥密码学的签名与验证算法

2. 认证技术—身份认证技术

- 身份认证基本概念
  - 是用户向系统出示自己身份证明的过程，也是系统核查用户身份证明的过程
  - 又称为身份鉴别
  - 区分认证、授权与访问控制
    - **认证**：判断是谁的过程
    - **访问控制**：判断权限的过程
    - **授权**：赋予权限的过程

认证是基础，**授权**和**访问控制**一般在认证之后

2. 认证技术—身份认证技术

认证技术分类

从使用的认证手段进行划分

认证技术	使用要素	认证方式	举例
基于口令的认证	What you know	口令	用户名、密码、动态口令
基于密码学技术的认证	What you know	密码学技术	共享密钥、PKI、数字签名
基于物理设备的认证	What you have	物理设备	IC卡、加密狗等
基于生物特征的认证	Who you are	人体生物特征	指纹、虹膜等
基于地址的认证	Where you are	地址认证协议	IP认证、端口认证
基于多因素的认证	多种结合	多因素结合	指纹与智能卡相结合

2. 认证技术—身份认证技术

- **认证技术分类**
  - 认证过程**是否动态**变化
    - **静**态认证
    - **动**态认证
  - 认证**因素**的数量
    - **单**因子认证
    - **双**因子认证
    - **多**因子认证
  - 认证采用的**密码算法**类型
    - 基于**对称**密钥算法的认证方式
    - 基于**公开**密钥算法的认证方式
    - 基于Hash算法的认证方式

2. 认证技术—消息认证技术

- 消息认证技术
  - 信源和信宿的认证：验证信息的**来源和目的地**
  - 消息内容认证：验证消息在传送或存储过程中**未被篡改**
    - 常用方法：基于消息认证码的方法
  - 消息序号和操作时间：验证消息在传送或存储过程中**未被重放或延迟**等。

2. 认证协议的基本技术

- 基本符号约定

符号	含义
Alice,Bob（简称A,B）	协议参与主体名称
ID_x	主体X的标识
A → B：M	A给B发送消息M
K_{AB}	A和B的共享密钥
$Sig_A\{M\}$	主体A对消息M产生的签名
N_x	主体X产生的随机数
t_x	主体X产生的时戳

2. 认证技术—消息认证技术

- 信源认证
  - 采用对称密码体制
    - A → B: $E_{KAB}(ID_A, M)$
    - B收到报文后用K_{AB}解密，若解密所得的发送方标识与ID_A相同，则B认为报文是A发来的。
  - 采用公开密钥密码体制
    - 只要发送方对每一报文进行数字签名，接收方验证签名即可：
    - A → B: M，$Sig_A(ID_A, M)$
    - B：$VER(Sig_A(ID_A, M))$;
    - 若收方验证签名正确，则认为发方为真。

2. 认证技术—消息认证技术

- 信宿认证
  - 将信源的认证方式稍加修改，便可实现对信宿的认证
  - 采用对称密码体制
    - A → B: $E_{KAB}(ID_B, M)$
    - B收到报文后用K_{AB}解密，若解密所得的发送方标识与自己的标识ID_B相同，则B认为报文是发给B的。
  - 采用公开密钥密码体制
    - 只要发送方对每一报文用B的公钥进行加密：
    - A → B: $E_{PKB}(ID_B, M)$
    - B收到后，用私钥解密，通过检查标识，判断是否发给自己。

2. 认证技术—消息认证技术

➢ 信息认证码MAC

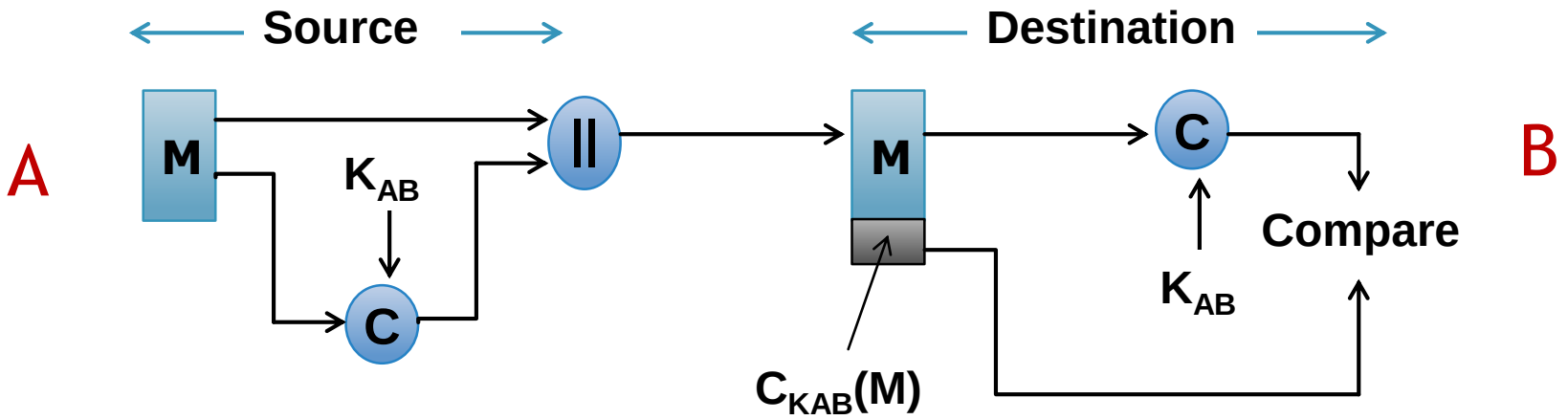
- 是消息内容和密钥的公开函数，其输出是固定长度的短数据块，又称密码校验和（cryptographic checksum）

$$MAC=C_K(M)$$

- 如果接收方计算的MAC与收到的MAC匹配，则
  - 接收者可以确信消息M未被改变。
  - 接收者可以确信消息来自所声称的发送者；
  - 如果消息中包含顺序码（如HDLC,X.25,TCP），则接收者可以保证消息的正常顺序；
- MAC函数类似于加密函数，但不需要可逆性。因此在数学上比加密算法被攻击的弱点要少。

2. 认证技术—消息认证技术

➢ MAC的基本用法



- **A → B:** $M||C_{KAB}(M)$
- 因为只有 A 和 B 共享了 K_{AB}，所以可以提供认证；
- 如果需要提供保密性，可以如下使用：
- **A → B:** $E_{KAB2}(M||C_{KAB1}(M))$

2. 认证技术——防重放基本技术

- 挑战-应答机制(Challenge-Response)
  - “ISO两次传输单方认证协议”，描述如下：
    1. A → B: N_A;
    2. B → A: $F_{KAB}(M,N_A)$
    3. A验证来自B的密文分组，如果N_A以正确形式出现，则验证通过，否则验证不通过。
  - 其中F:
    - 对称加密算法
    - 公钥算法
      1. A→ B: N_A;
      2. B→ A: M，$Sig_B(M,N_A)$
      3. A采用B的公钥验证来自B的签名。
    - 消息认证码（MAC）

2. 认证技术——防重放基本技术

- 时间戳机制
  - A根据消息中的时间戳信息，判断消息的有效性
    - 如果消息的时间戳与A所知道的当前时间足够接近
  - “ISO一次传输单方认证协议”，描述如下：
    1. A → B: $F_{KAB}(M,tt_A)$
    2. B验证来自A的密文分组，如果tt_A以正确形式出现并且有效，则验证通过，否则验证不通过。
  - 这种方法要求不同参与者之间的时钟需要同步
    - 在网络环境中，特别是在分布式网络环境中，时钟同步并不容易
    - 一旦时钟同步失败
      - 要么协议不能正常服务，影响可用性(availability)，造成DOS攻击
      - 要么放大时钟窗口，造成攻击的机会
  - 时间窗大小的选择应根据消息的时效性来确定

2.认证技术——防重放基本技术

- 序列号机制
  - 要求每对通信实体必须事先商定好序列号递增的方式，实体收到消息后，检查消息中的序列号是否有效。
  - 优点：
    - 处理简单，延迟小，适用于无线连接服务的分布式应用
  - 缺点：
    - 序列号管理开销大
    - 同样存在同步窗口
      - 同步窗口过小，影响可用性(availability)，造成DOS攻击
      - 同步窗口过大，造成攻击的机会
    - 存在序列号失步情况，需要设计序列号再同步机制

本讲内容

1. 认证协议的基本概念
2. 认证协议的基本技术
3. 经典认证协议

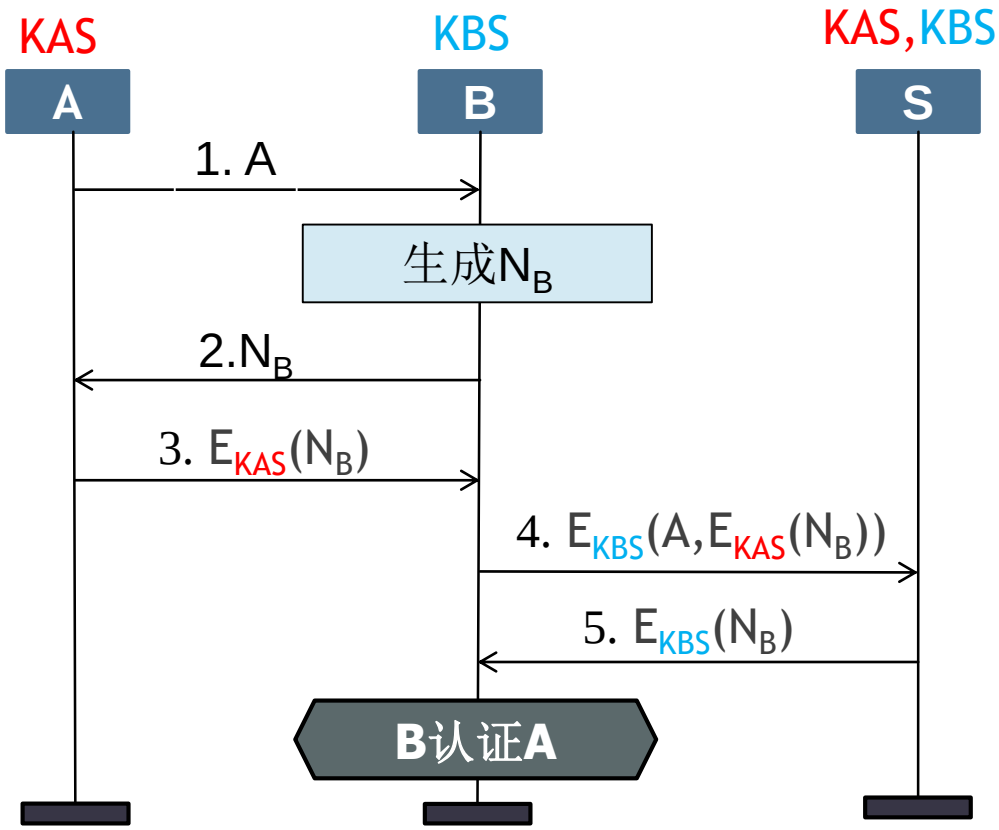
3.经典认证协议

- Woo-Lam协议
- NSSK协议
- NSPK协议
- OTWAY-REES认证协议
- YAHALOM协议
- 大嘴青蛙协议

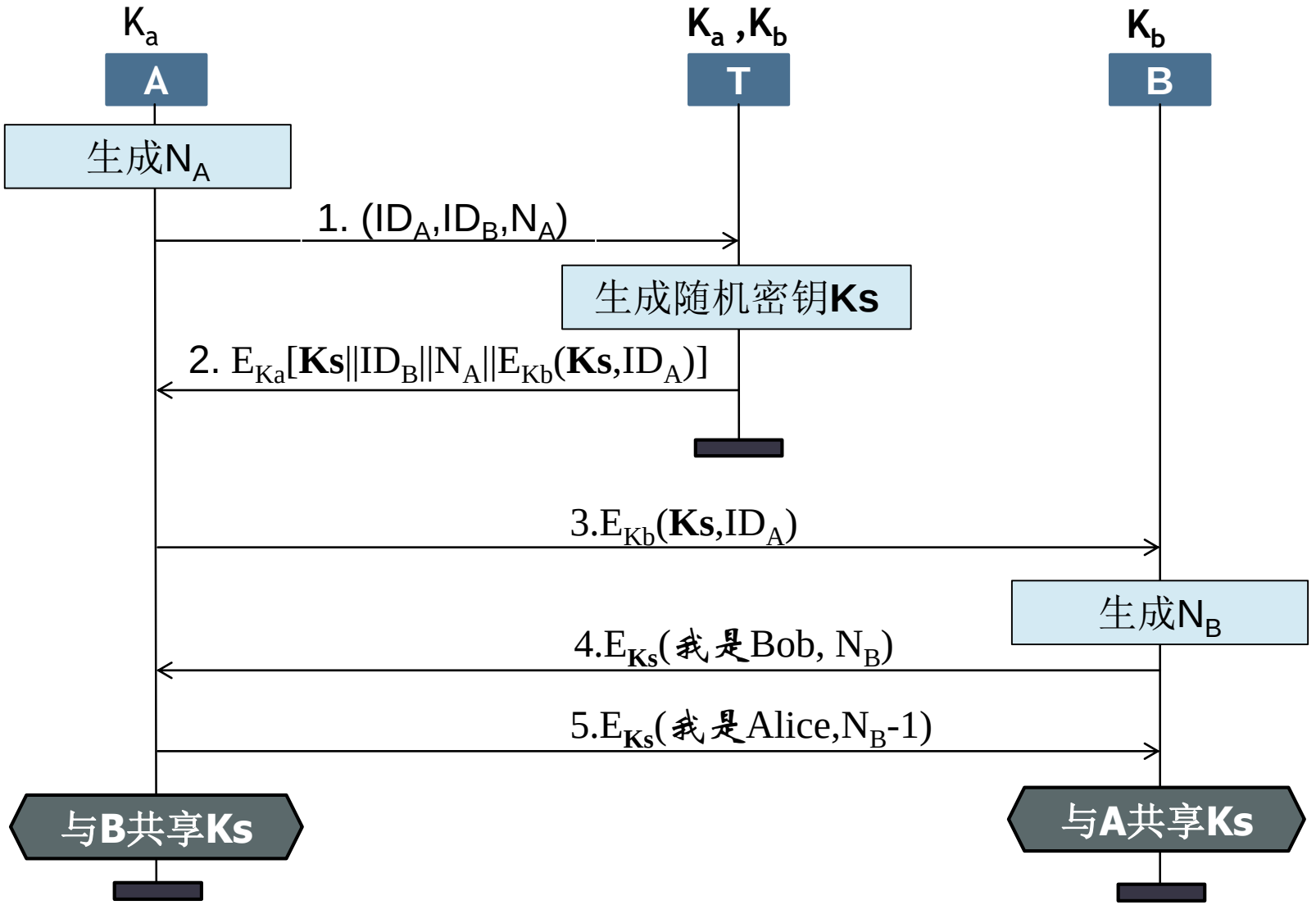
Woo-Lam 协议

- 目标：B 认证A

1. $A \rightarrow B : A$
2. $B \rightarrow A : N_B$
3. $A \rightarrow B : E_{KAS}(N_B)$
4. $B \rightarrow S : E_{KBS}(A, E_{KAS}(N_B))$
5. $S \rightarrow B : E_{KBS}(N_B)$

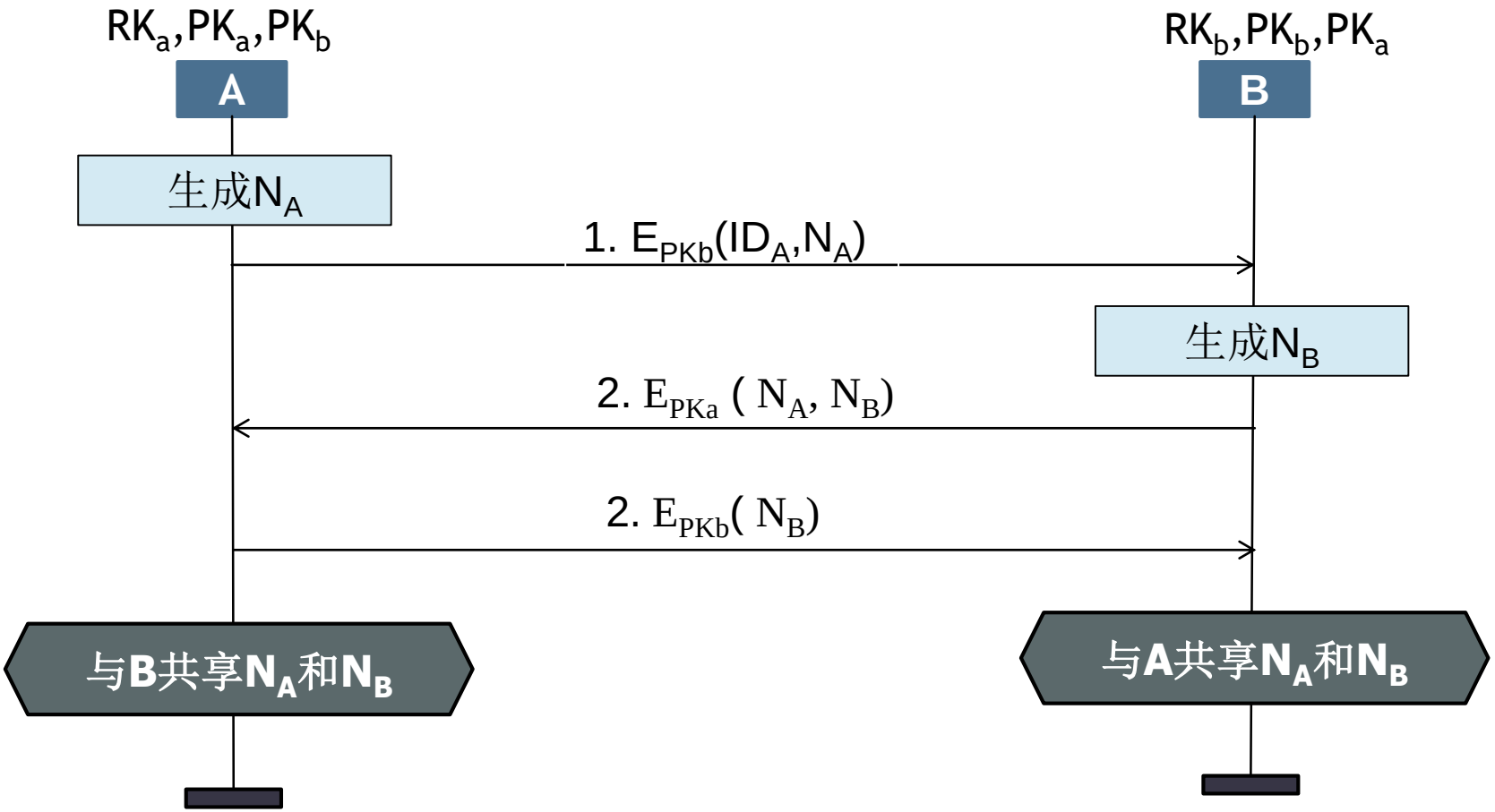


NSSK协议


K_a
A
K_a ,K_b
T
K_b
B
生成N_A
1. (ID_A,ID_B,N_A)
生成随机密钥**Ks**
2. E_{Ka}[**Ks**||ID_B||N_A||E_{Kb}(**Ks**,ID_A)]
3.E_{Kb}(**Ks**,ID_A)
生成N_B
4.E_{Ks}(我是Bob, N_B)
5.E_{Ks}(我是Alice,N_B-1)
与B共享Ks
与A共享Ks

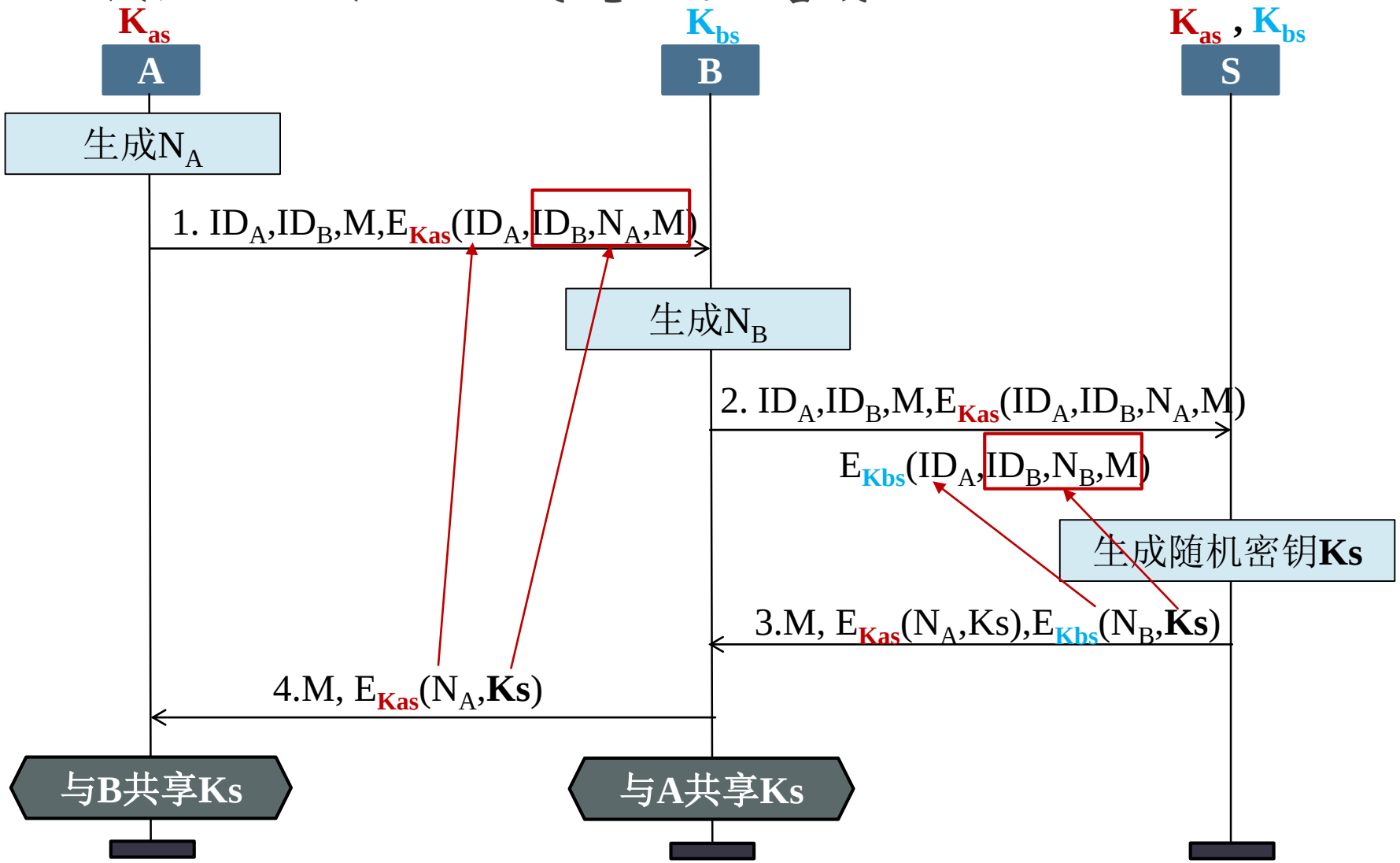
NSPK协议

- 目的：
  - 使通信双方安全地交换双方彼此的秘密



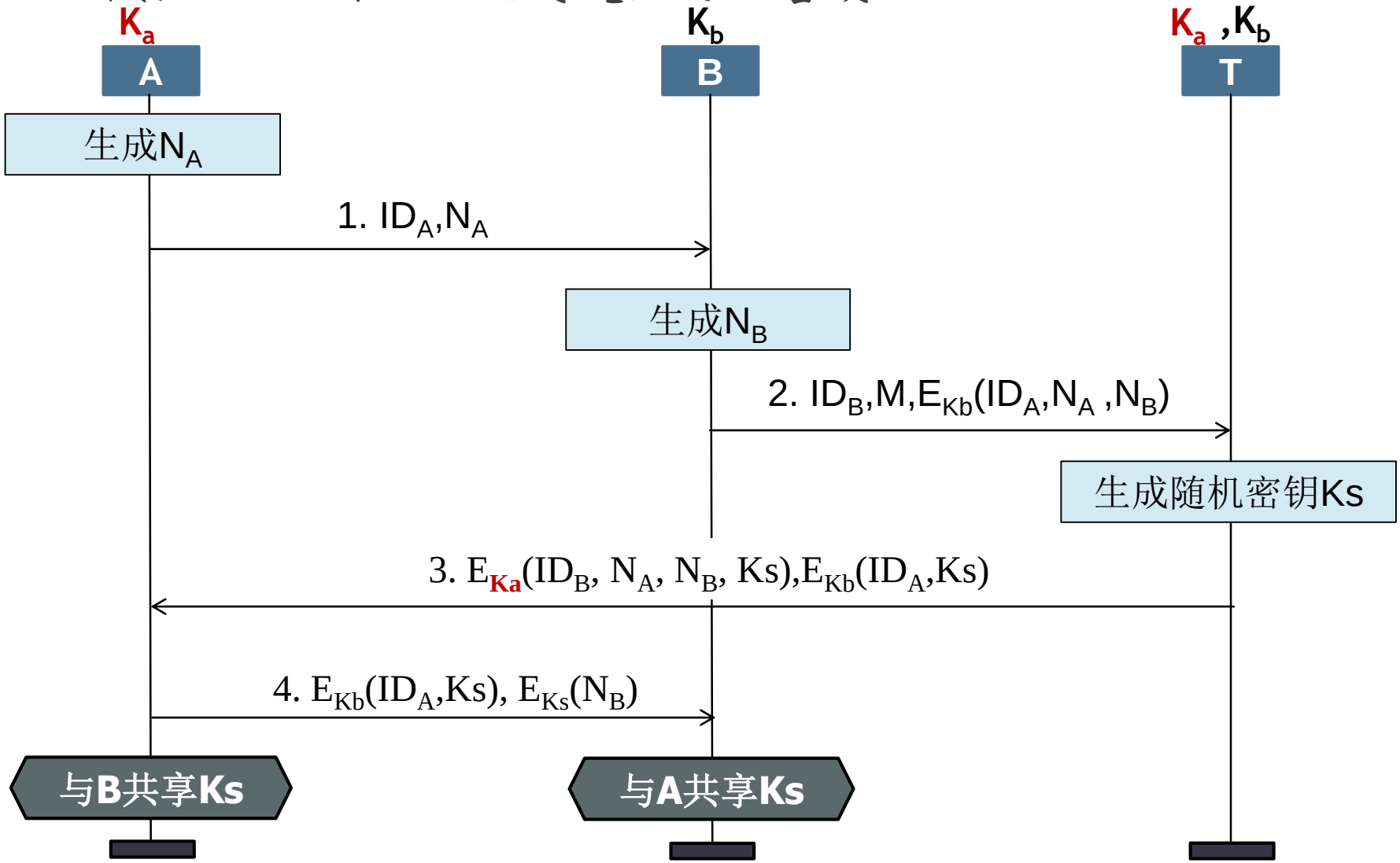
OTWAY-REES认证协议

- 目标：Alice和Bob之间建立会话密钥。



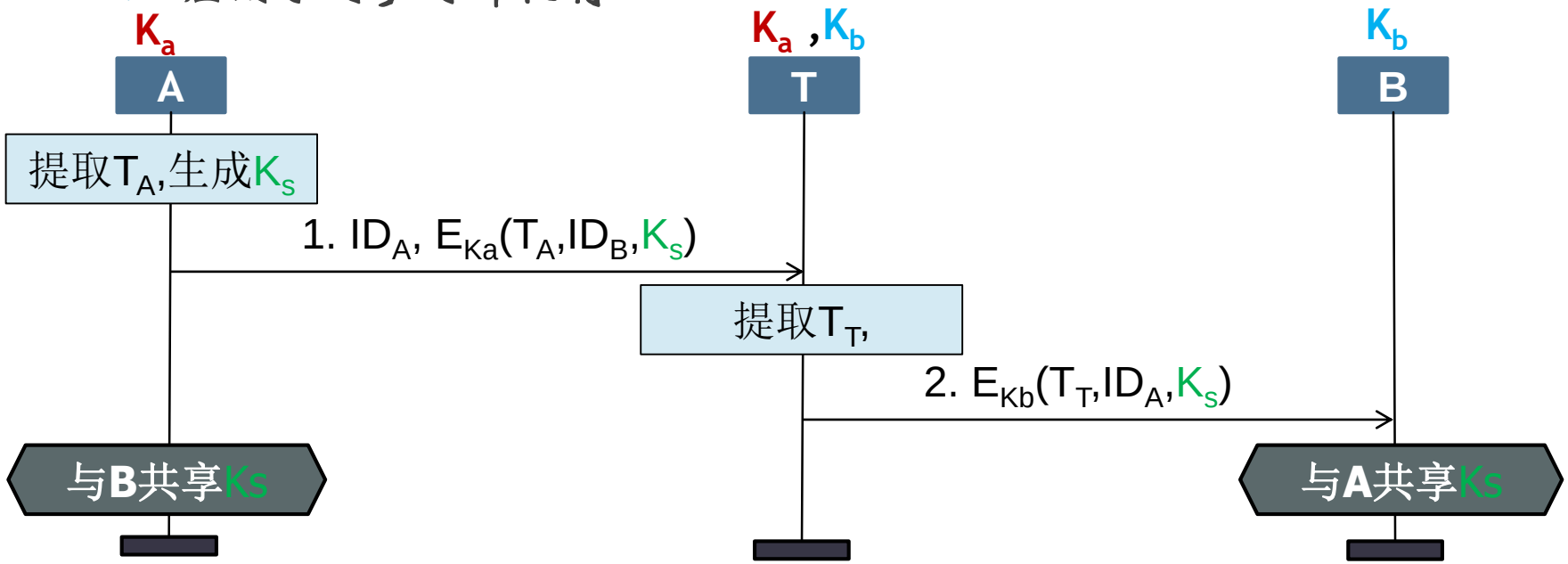
YAHALOM协议

- 目标：Alice和Bob之间建立会话密钥。



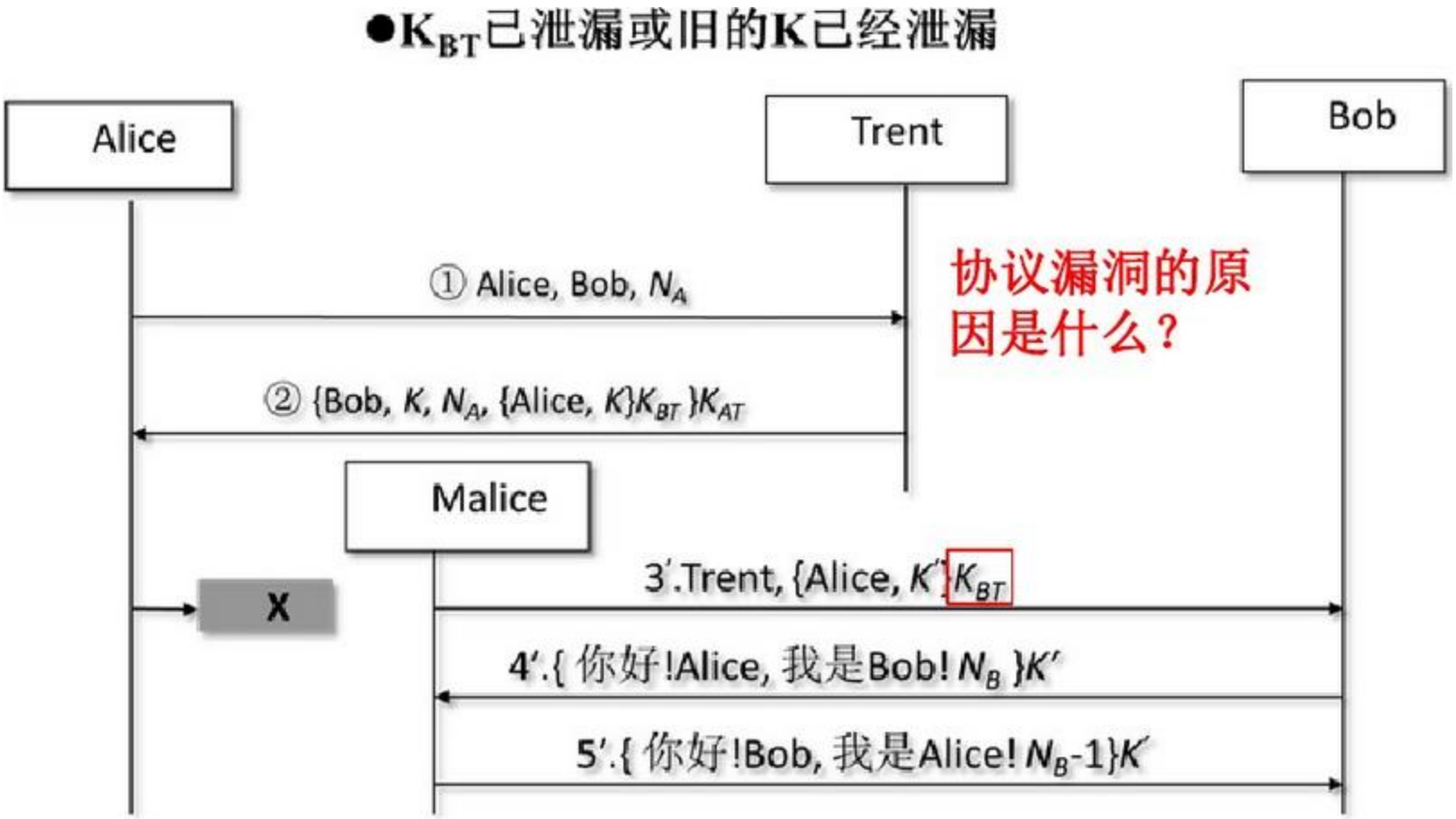
大嘴青蛙协议

- 最简单的、应用对称密码体制的三方认证协议
- 在该协议中，Alice通过可信第三方Trent向Bob传送会话密钥。
- 协议特点：
  - 会话密钥只通过两个步骤就从Alice传送给了Bob；
  - 应用了同步时钟机制。

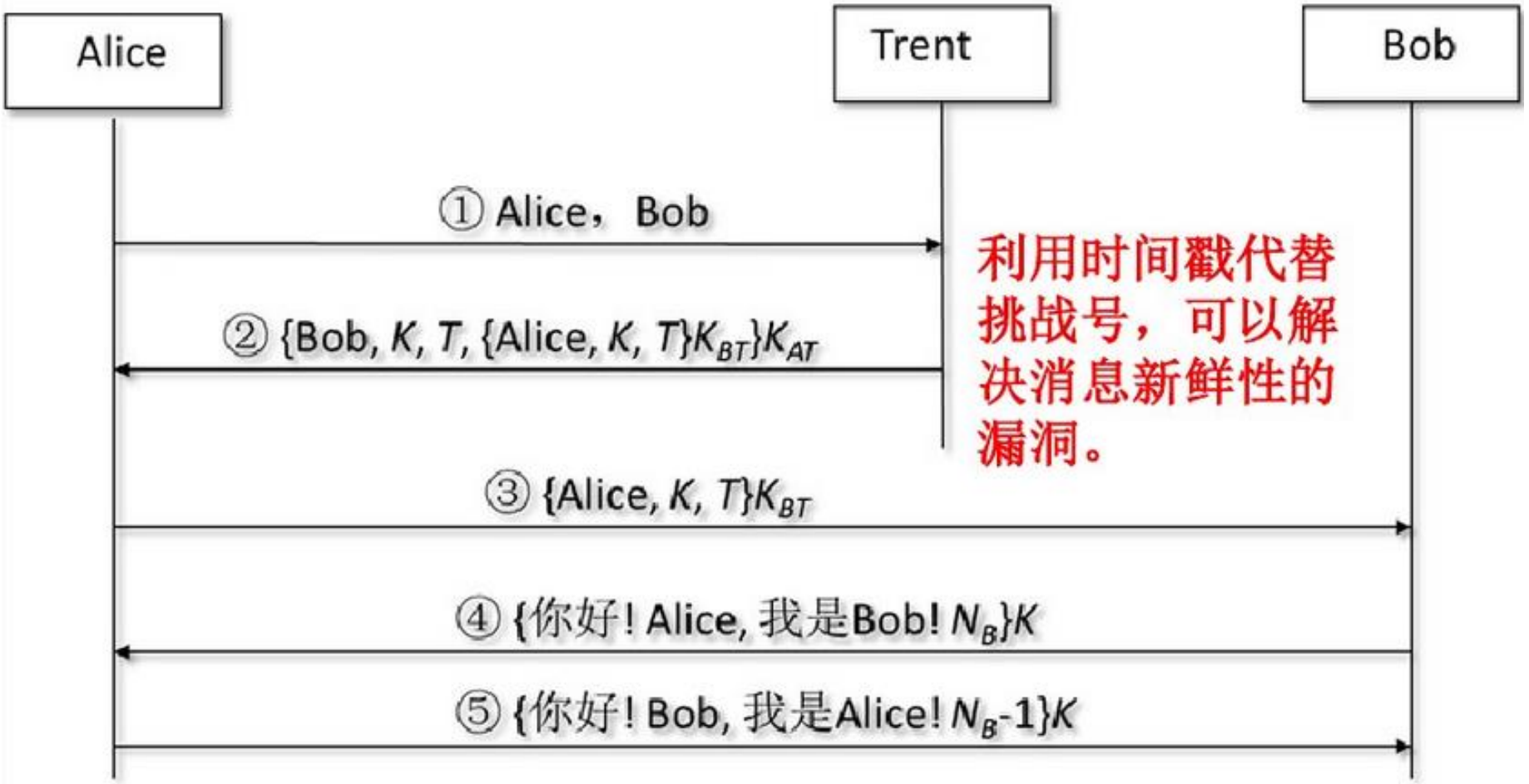


补充材料：对上述协议的攻击

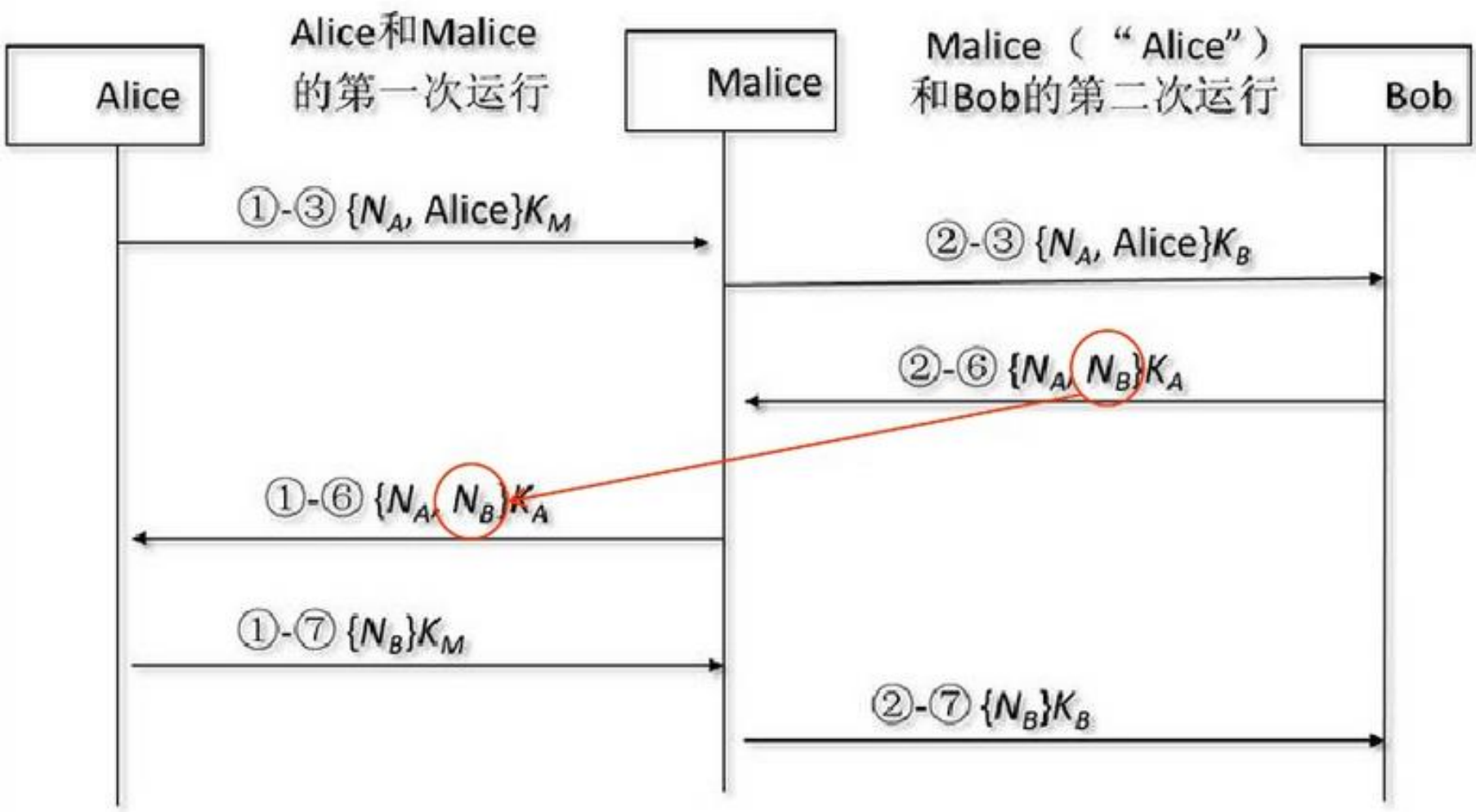
DENNING和SACCO对NSSK协议的攻击



DENNING和SACCO对NSSK协议的改进



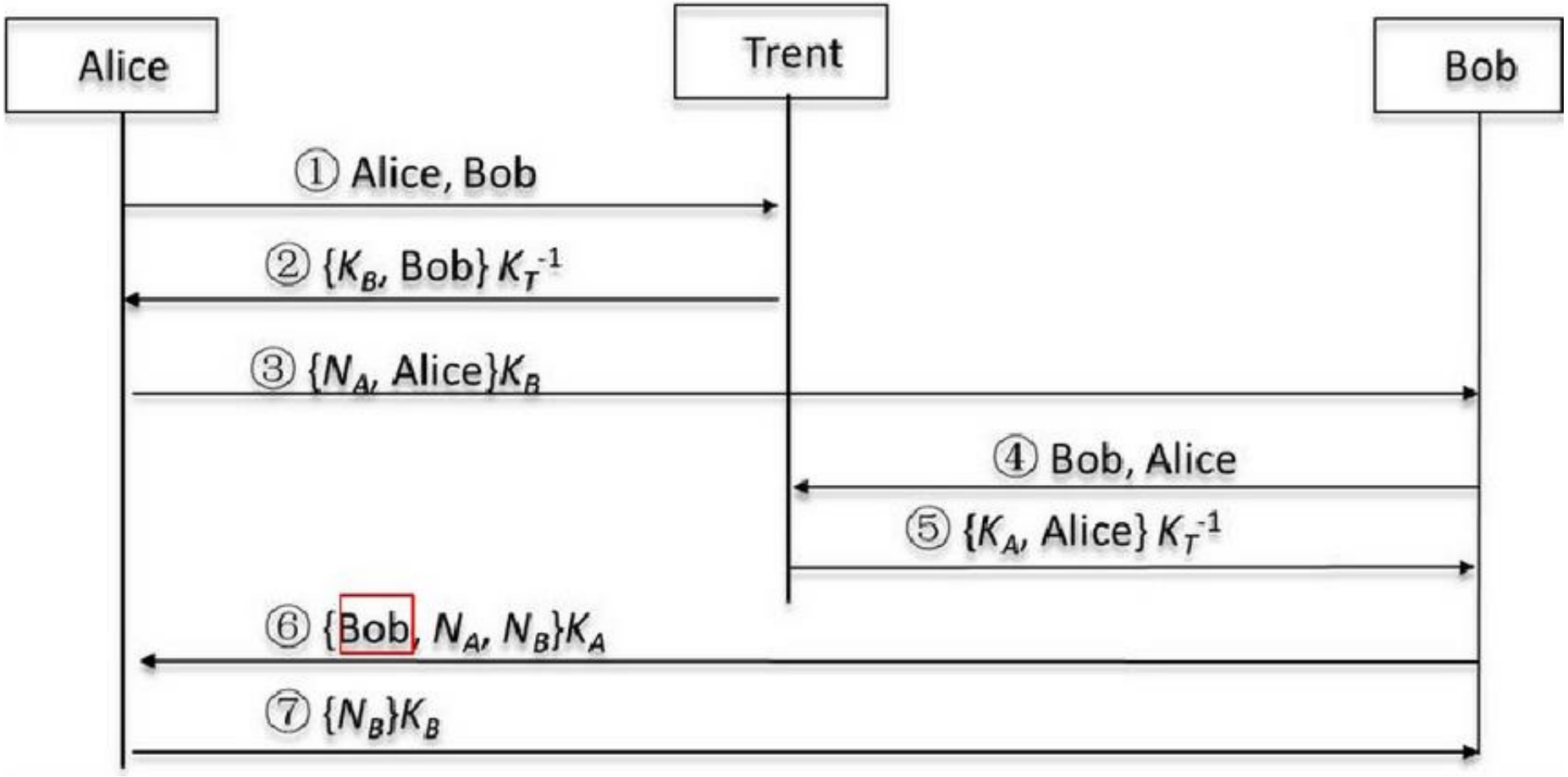
LOWE对NSPK协议的攻击



LOWE对NSPK协议的攻击

- 假设在这个系统中Malice是一个合法的主体，因此其他主体可能要和Malice建立标准会话。这个攻击过程中包括两次同时运行该协议。
- 第一次是Alice和Malice之间的运行①-③、①-⑥、①-⑦步)，结果是Alice和Malice建立了了一个合法的会话。
- 第二次是Malice假冒Alice和Bob之间的运行(②-③、②-⑥②-⑦步)，结果是Malice假冒Alice和Bob建立了一个假的会话。
- 我们分析以上攻击，Malice成功攻击的关键步骤是Alice无意地为他解密了Bob的nonce N_B。当一个主体无意地为攻击者执行了一个密码运算时，该主体就被用作预言机(oracle)或提供了预言机服务(Oracle service)。因此密码算法和协议应该设计成即使用户为攻击者提供了预言服务也是安全的。

对NSPK协议的改进


Alice
Trent
Bob
① Alice, Bob
② {K_B, Bob} K_T^{-1}
③ {N_A, Alice}K_B
④ Bob, Alice
⑤ {K_A, Alice} K_T^{-1}
⑥ {Bob, N_A, N_B}K_A
⑦ {N_B}K_B

针对OTWAY-REES协议的“类型缺陷”型攻击

“类型缺陷”型攻击的特点是利用认证协议实现时的固定格式对协议进行攻击。有些协议只检测长度的合法性。假定在Otway-Rees认证协议中，M的长度是64比特，Alice和Bob的长度各为32比特，Kas的长度为128比特。

- 攻击过程如下(Malic冒充Bob)
- 1.Alice→Malic (“Bob”) :M, Alice, Bob, $\{N_A, M, Alice, Bob\}K_{AT}$;
  - 【发给Bob的消息被Malice劫持】
- 2.malic→Trent:M,Alice,Malic,$\{N_A, M, Alice, Bob\}K_{AT}$,$\{N_B, M, Alice, Malic\}K_{MT}$;
  - 【如果Trent只是检测了长度的合法性，就导致攻击成功】
- 3.Trent→Malic:M,$\{N_A, K_{AB}\}K_{AT}$, $\{N_B, K_{AB}\}K_{MT}$;
  - 【你们验证通过，给你们密钥】
- 4.Malice(“Bob”)→Alice:M,$\{N_A, K_{AB}\}K_{AT}$
  - 【你验证通过了，给你密钥】

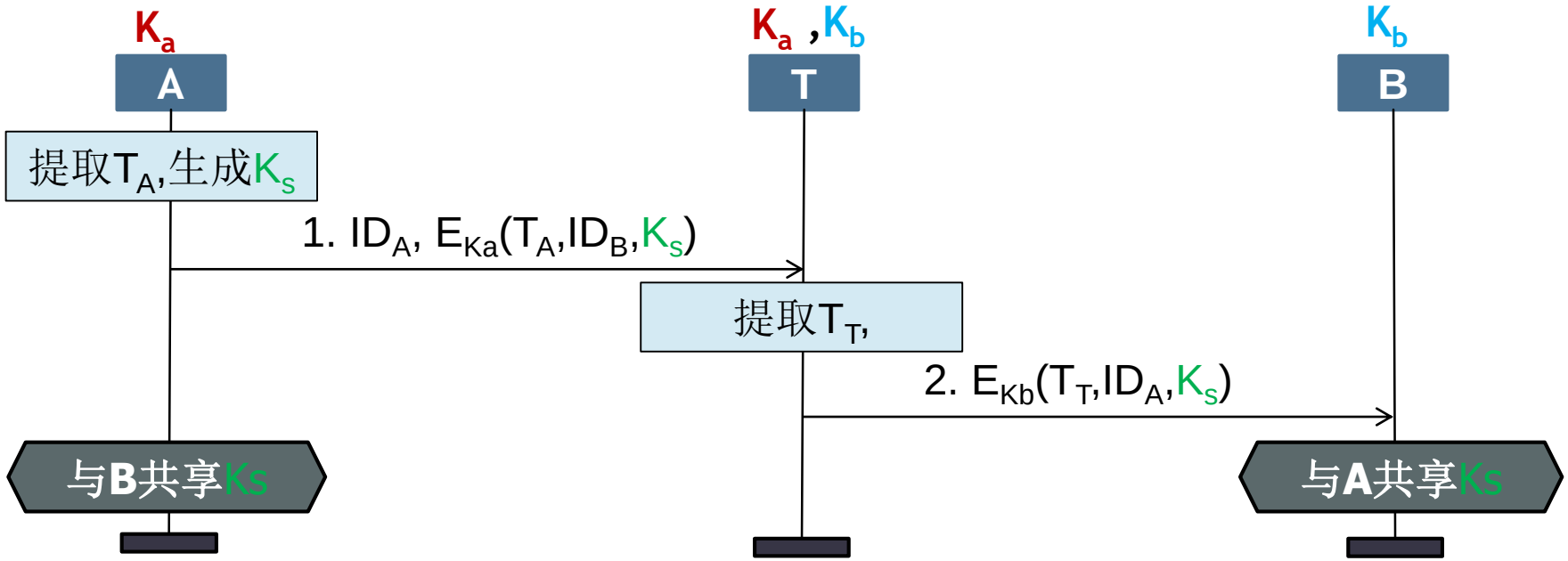
针对OTWAY-REES协议的“类型缺陷”型攻击

攻击者还可以冒充可信第三方Trent攻击Otway-Rees协议。攻击过程如下：

- 1’.Alice→Bob:M,Alice,Bob,$\{N_A,M,Alice,Bob\}K_{AT}$;
- 2’.Bob→Malic(Trent):M,Alice,Bob,$\{N_A,M,Alice,Bob\}K_{AT}$,$\{N_B,M,Alice,Bob\}K_{BT}$;
  - 【发给Trent的信息被劫持】
- 3’. Malic(Trent) → Bob :M,$\{N_A,M,Alice,Bob\}K_{AT}$,$\{N_B,M,Alice,Bob\}K_{BT}$;
  - 【利用手头上的密文进行攻击】
- 4’. Bob→Alice: M,$\{N_A,M,Alice,Bob\}K_{AT}$
  - 【攻击成功的前提是Kab的长度=M+Alice+Bob的长度】

针对大嘴青蛙协议的攻击

- 第1种方法是在有效的时间范围内重放步骤①的消息。其后果是将进行重新认证。因为根据协议，Trent将生成一个新的消息2,并生成一个新的时间戳。这样通信的时间戳被修改可能影响后续的使用。(破坏可用性的方法)



针对大嘴青蛙协议的攻击

- 第2种攻击方法攻击过程如下:(实施破坏性的攻击)

1. A→T: A, $\{T_A, B, K_{AB}\}K_{AT}$;
2. T→B: $\{T_T, A, K_{AB}\}K_{BT}$;

1'.M(B) →T: B, $\{T_T, A, K_{AB}\}K_{BT}$;

2'.T→ M(A) :$\{T'_T, B, K_{AB}\}K_{AT}$;

1*.M(A) →T: B, $\{T'_T, B, K_{AB}\}K_{AT}$;

2*.T→ M(B) :$\{T^*_T, A, K_{AB}\}K_{BT}$;

- 第1次运行，Malic监听Alice与Bob之间的一次会话；
  第2次运行，Malic假冒Bob，从Trent获得对它有用的消息2'；
  第3次运行，Malic假冒Alice，从Trent获得对它有用的消息2*。
- Malic通过重放上述消息，引起Alice与Bob之间的重新认证。

谢谢大家，欢迎提问！