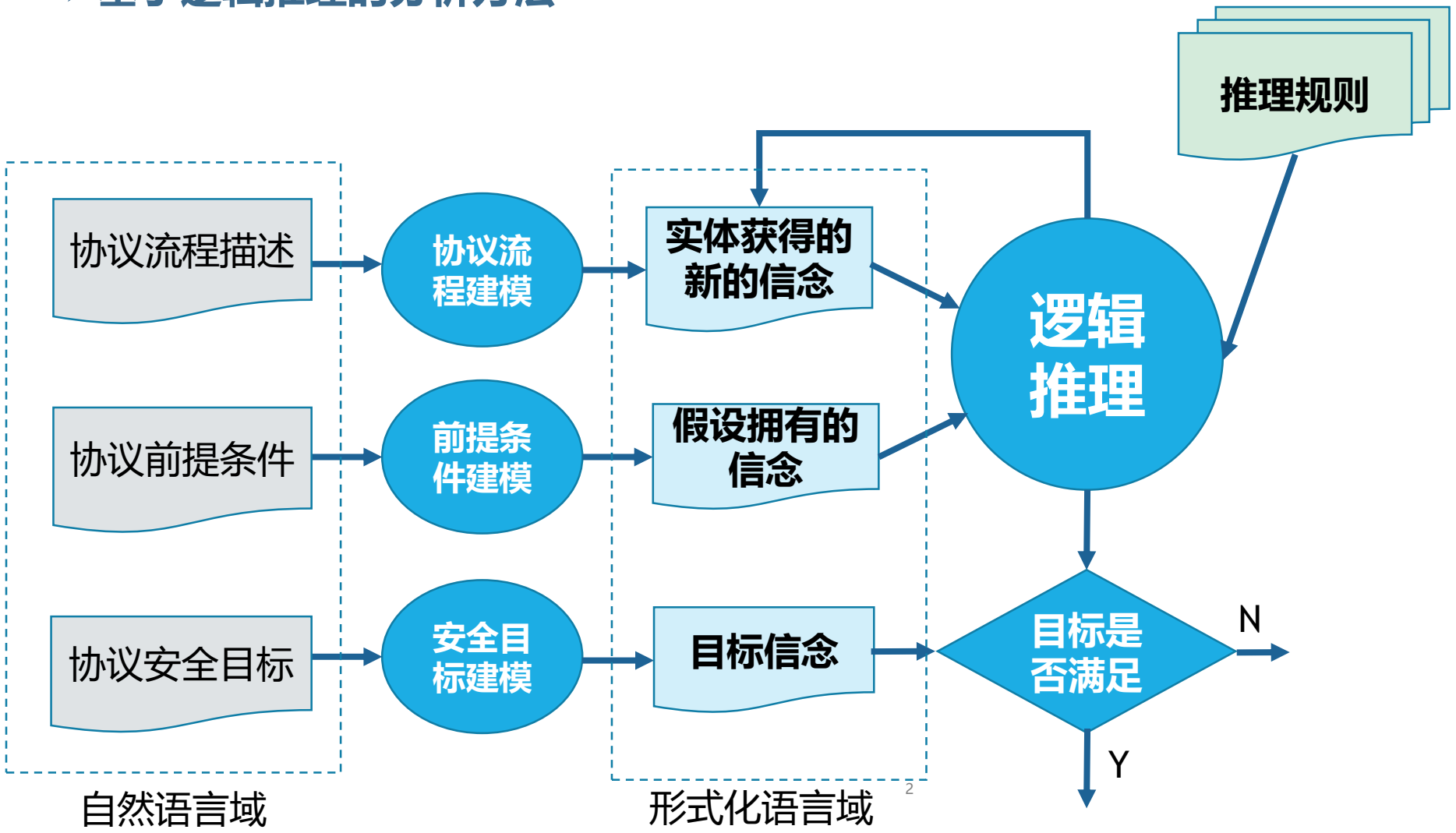
安全协议设计与分析

第十讲：基于模型检测的分析方法

李晖 网络空间安全学院

上次内容

➢基于逻辑推理的分析方法

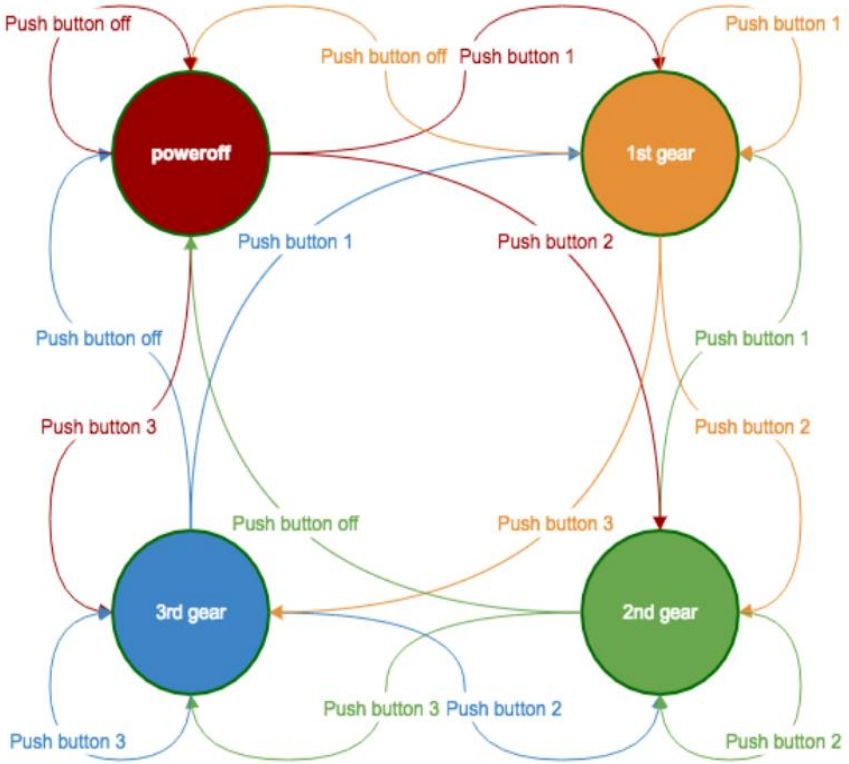


本讲内容

- 概述
- Dolev-Yao模型
- 通信进程方法
  - 通信进程方法CSP简介
  - CSP的基本概念
- FDR简介+利用FDR进行安全协议分析的流程
- CSP网络模型
- 基于CSP的安全协议分析实例
  - 安全协议系统建模
  - 协议安全性质的CSP描述与验证

概述

- 基于**模型检测**技术的安全协议分析方法
  - 也称为基于**状态检测**的方法。
- **基本思路**
  - 利用有限状态机理论，通过定义**状态集合**及**状态迁移函数**为安全协议系统建立模型；
  - 通过**穷尽搜索状态空间**来判断一些特殊的状态是否可达，或者是否可以生成一条特殊的状态转移路径，并以此检测该模型是否具备期望的安全性质。



电风扇状态图

概述

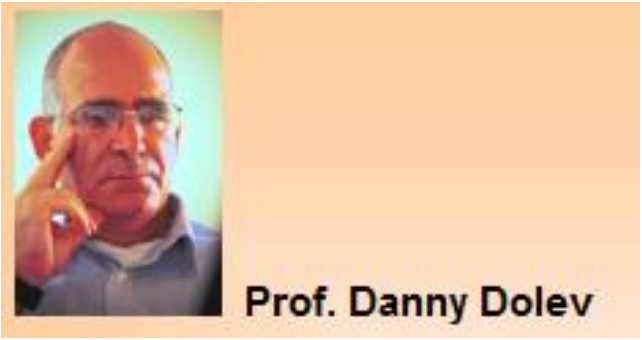
- 基本流程
  - 把安全协议看成一个**分布式系统**，单个主体涉及的协议执行部分称为局部状态，所有局部状态构成系统的全局状态；
  - 协议执行过程中，主体收发消息的动作会引起局部**状态的改变**，进而也引起全局状态的改变；
  - 在安全协议全局状态上定义安全属性或不变关系，则安全协议是否满足安全目标等价于在系统可达的每个全局状态上**安全属性或不变关系**是否都能得到满足。

概述

- **优点**
  - **自动化程度高**，可以借助自动分析工具来完成分析过程而不需要用户的参与，并且安全协议存在缺陷时能够自动生成相应的攻击实例。
- **主要方法**
  - 通信进程方法
  - NRL协议分析器
  - 模型检测工具Murφ
  - 模型检测工具ASTRAL
  - 协议分析工具BRUTUS
- **基于Dolev-Yao模型**

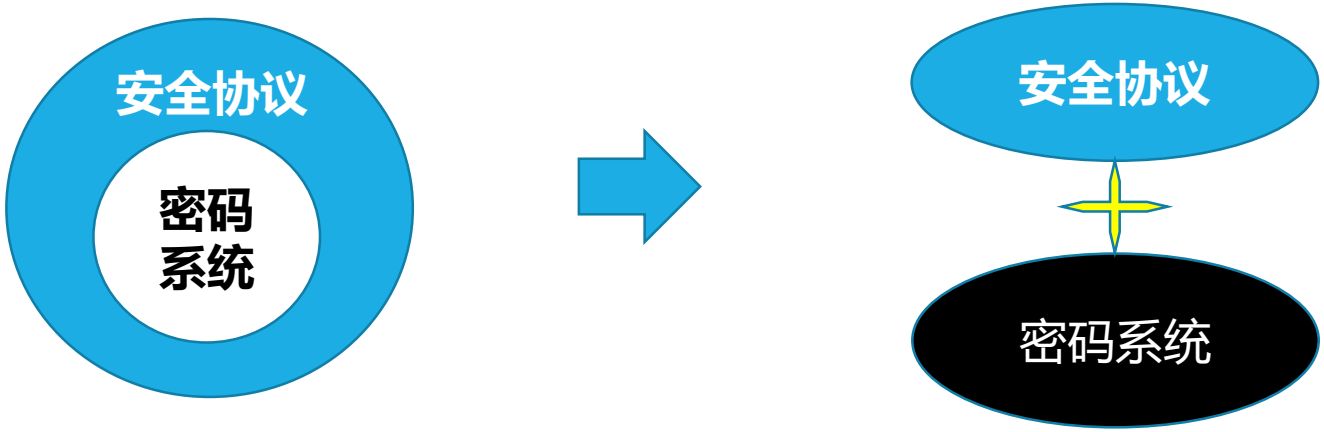
Dolev-Yao模型

- 由Dolev、Yao于1983年首次提出



Andrew Chi-Chih Yao

- 将**安全协议**本身与安全协议所采用的**密码系统**分开考虑，降低了安全协议分析问题的复杂度



Dolev-Yao模型

- **密码系统模型**
  - 将密码系统看做是一个黑盒子，并假设采用的密码算法和密码技术是完善的；
  - 主体仅在拥有了正确的解密密钥时才能进行解密，并且产生的密文必须拥有对应的明文和加密密钥。
- **攻击者模型**
  - 描述了攻击者的知识和能力

Dolev-Yao模型

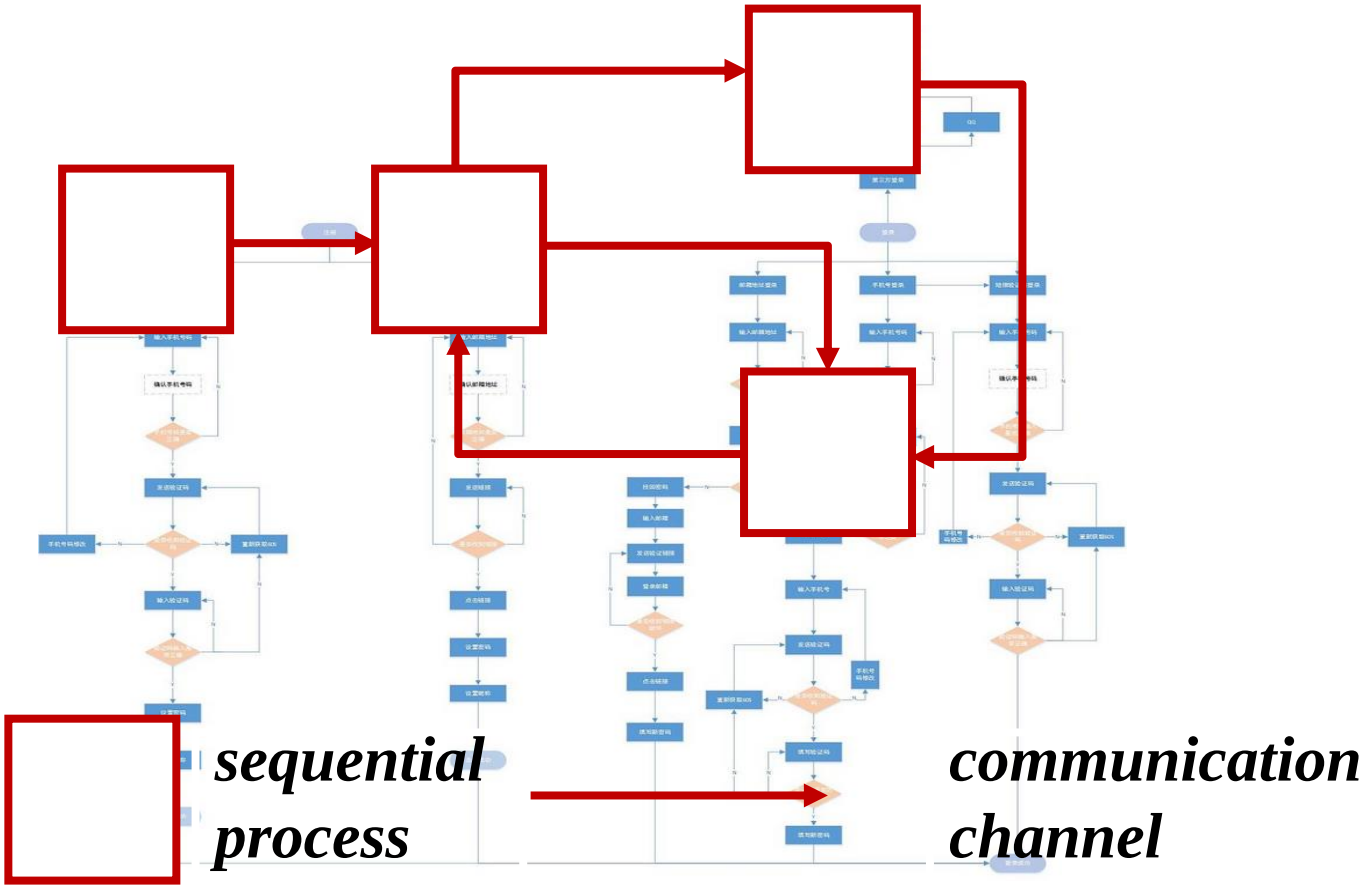
➢攻击者模型

1) 熟悉**现代密码学**，知道加解密等运算操作。
2) 知道**参与**协议运行的各**实体**及其**公钥**。
3) **拥有自己的加解密密钥**，可将窃听到的消息增加为自己的新知识。
4) **对网络具有完全的控制能力**，可窃听、拦截系统中传送的任何消息。
5) 可用他拥有的加密或解密密钥对消息进行加密或解密操作。
6) 可在系统中插入新的消息。
7) 即使不知道加密部分的内容，也可重放他所看到的任何消息。
8) 可以生成新的随机数等。

Dolev-Yao模型

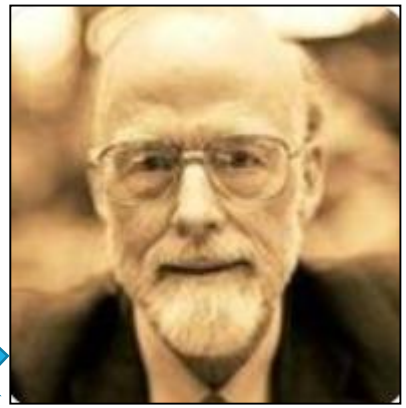
- Dolev-Yao模型的重要意义
  - 模型检测技术及定理证明技术均建立在此模型之上
  - 逻辑推理依赖于Dolev-Yao模型的黑盒假设
- Dolev-Yao模型的局限性
  - 无法对密码系统相关缺陷进行分析
  - 对攻击者的能力假设过强
    - 攻击者总能形成拒绝服务攻击（DOS）
    - 电子商务协议的匿名性也无法得到保证

通信进程方法CSP简介



通信进程方法CSP简介

- **通信顺序进程**（Communicating Sequential Process，CSP）是一种用来描述**并发系统**中通过消息交互的通信实体行为的抽象符号语言。
- 该理论最早由著名计算机科学家**C.A.R.Hoare**设计提出。
- 目前，CSP方法在安全协议的分析验证方面得到了成功的应用，
  - 将安全协议的信息交互归约为CSP
  - 将其安全目标也在CSP框架内描述为CSP要说明的问题
  - 通过有效的分析方法对这些描述进行推理
- CSP —> π-calculus—>Applied π calculus



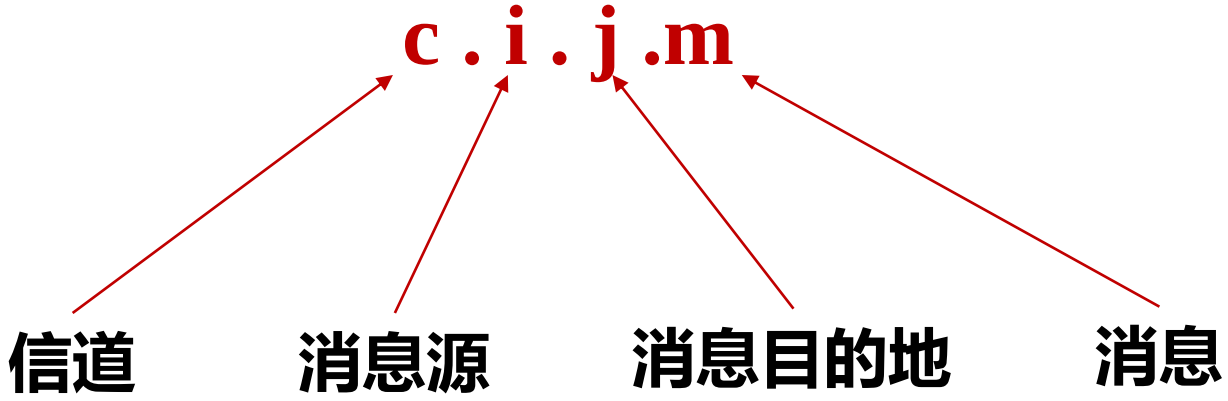
C.A.R.Hoare

于1980年获得美国计算机学会(ACM)设立的计算机界最高奖——图灵奖，2000年获得日本稻盛财团设立的国际大奖——京都奖(尖端技术领域)。同年，英国女王伊丽莎白二世授予Tony Hoare爵士爵位，以表彰他对计算机科学所做出的巨大贡献。

CSP的基本概念

1．事件

- 协议是通过实体间执行一系列的**事件（Event）**来执行的，所有可能事件的集合记为∑。
- 一个典型的CSP事件表示为



CSP的基本概念

2．进程

- CSP通过用某一进程（Processes）可能涉及到的事件来描述该进程，从而提供了一种描述进程可达状态的方法。
- 进程的表示通常由两个部分构成：执行状态中的一个**动作（事件）**、动作结束后的**状态**，两者之间可以用“→”连接。

（1）停止进程 STOP √

- 该进程不执行任何事件，等价于死锁。

（2）事件前缀 a→P √

- 假设进程P及动作a，则a→P表示执行动作a后按照进程P来执行。
- 例如，假设in和out是∑中的两个动作，进程in→out→STOP表示依次执行in和out，之后不再执行任何动作。

CSP的基本概念

- **（3）事件前缀选择 ?x: A→P** √
  - 若A为一事件集合，$A \subseteq \Sigma$，则进程?x: A→P(x)表示当选择任意的动作$x \in A$时，就执行进程P，执行后状态为P(x)，即x可以作为进程P执行时的参数。
- **（4）输入前缀选择 c?x: A→P** √
  - 设c为信道，$A \subseteq \sum$，则进程c?x:A→P(x)表示在信道c上接收任意输入$x \in A$之后，执行进程P(x)。
- **（5）输出前缀选择 c!x→P** √
  - 设c为信道，则进程c!x→P(x)表示在信道c上输出x之后，执行进程P(x)。

CSP的基本概念

- **（6）进程间选择 P□Q** √
  - 设P和Q是两个进程， P□Q表示在P和Q之间做选择，结果或为P，或为Q。作用就是给出了两个进程动作的选择项，然后按照其中一个选择的动作运行。
  - $\square_{i:I}P(i)$ 代表从I中选择一个i，执行过程P(i);
- **（7）接口并行进程** $P \underset{Y}{\|} Q$ √
  - Y是一个事件子集，进程P和Q并行执行，通过来自Y中的事件进行同步；
  - 如果P和Q不需要同步，则标记为**P|||Q;**

CSP的基本概念

- **（8）成功终止 SKIP** √
  - 成功终止通常是由一个进程主动产生的。
- **（9）顺序组合 P；Q** √
  - P、Q是两个进程，**P；Q**表示在P结束之前都是运行P的事件，P结束后才运行Q的事件，并且直到Q终止，P；Q才终止。
- **（10）递归进程 μP.F(P)** √
  - 下面以一个简单的例子来说明递归进程的含义。假设Alt进程描述的是只要环境允许就交替执行to和fro：Alt=to→fro→Alt，则可用表达式μP.to→fro→P来表示。

CSP的基本概念

3．迹

- 一个进程的**迹（Traces）**模型描述了该进程所有可能的事件序列。例如：
  - **Traces(STOP)={< >}**，其中< >是一个空序列，即不管观察STOP进程多长时间，它都是什么也不做。
  - **Traces(μP.a→P□b→SKIP)={$<a>^n$, $<a>^n<b>$,$<a>^n<b,\surd>$|n∈N}**，其中ab表示a和b的串联，s^n是s的n个副本的串联；√是发送终止信号的事件，当该信号在迹中出现时就是迹中的最后一个元素。

CSP的基本概念

- 与迹有关的表示如下：
  - **1）tr↑D：**表示tr中由D中事件组成的最大子序列。如果D是由一个事件d组成，即D={d}，则tr↑D等价于tr↑d。
  - **2）tr↓C：**表示在信道C上传输的最长消息序列。
  - **3）tr⇓C：**表示在C中某些信道上传输的消息集合。
  - **4）σ(tr)：**表示在迹tr上出现的事件集合。若将此操作扩展到进程上，则σ(P)表示在进程P的某些迹上出现的事件的集合。

CSP的基本概念

- **迹上的运算规则**有：
  - **1）traces(STOP)={< >}。** √
  - **2）traces(a→P)={< >}∪{<a>s|s∈traces(P)}**，即该进程要么什么都不做，要么第一个事件总是a，并且其后有一个P的迹。 √
  - **3）traces(?x: A→P)= {< >} ∪ {<a>s|a∈A∧ s ∈ traces(P[a/x])}**，与迹traces(a→P)类似，只是初始事件是从集合A中选择，而且后来的行为依赖于已选的行为a，其中P[a/x]表示使用a值来替代标识符x表示的任意发生事件。
  - **4）traces(c?x: A→P)= {< >}∪ {<c.a>s|a∈A∧ s ∈ traces(P[a/x])}**，与迹traces(?x: A→P)类似，只是这里使用了信道名。
  - **5）traces(P□Q)=traces(P) ∪ traces(Q)**，该进程提供了P的迹和Q的迹。 √

CSP的基本概念

- **6）traces(P || Q)=traces(P) ∩traces(Q)**，当P和Q在任何事件中都需要同步时，其组合的迹就是那些既是P的迹，又是Q的迹。
- **7）traces** $\left(P \underset{Y}{\|} Q\right) = \cup\{s \underset{Y}{\|} t \mid s \in traces(P) \wedge t \in traces(Q)\}$，这里的 $s \underset{Y}{\|} t$ 是P和Q分别执行 s 和 t 所产生的迹的集合。
- **8）traces** $(P_X \|_Y Q)=\{s \in (X \cup Y)^{*\surd}\}) \mid s \Gamma X^{\surd} traces(P) \wedge s \Gamma Y^{\surd} \in traces(Q)\}$，其中$X^{\surd}=X \cup \{\surd\}$，$s \Gamma Z$表示s受Z的约束，即s中所有Z之外的成员都被丢弃。该式表示P一定要执行X中的所有事件，而Q一定要执行Y中的所有事件，且其组合进程只有当P和Q都被终止时才终止。
- **9）traces** (SKIP)={< >，<√>}，即SKIP要做的就是终止。
- **10）traces** (P； Q)=(traces(P)∩Σ^*){st|s<√>traces(P) ∧ t ∈ traces(Q)}。

CSP的基本概念

- P is refined by Q，记为：$P \sqsubseteq_T Q$，如果Q的所有迹都是P的迹

$$P \sqsubseteq_T Q \quad \Leftrightarrow \quad traces(P) \supseteq traces(Q)$$

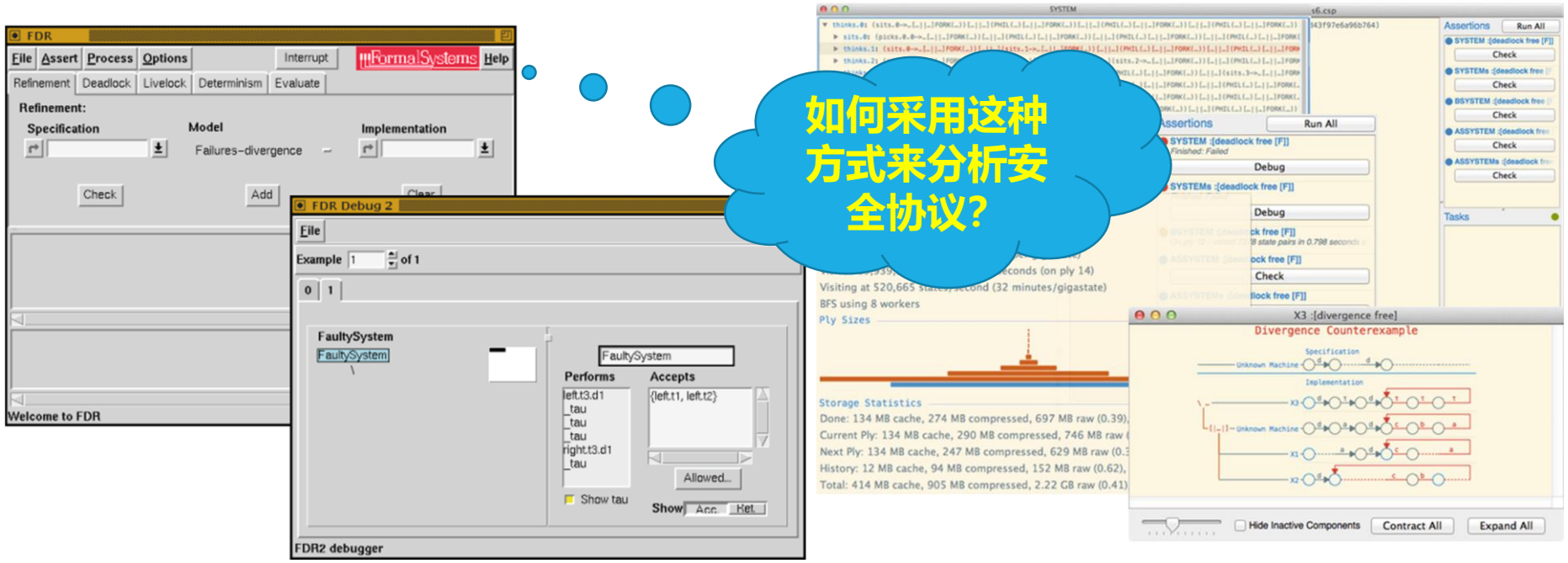
系统规约模型
Specification

系统实现模型
Implementation

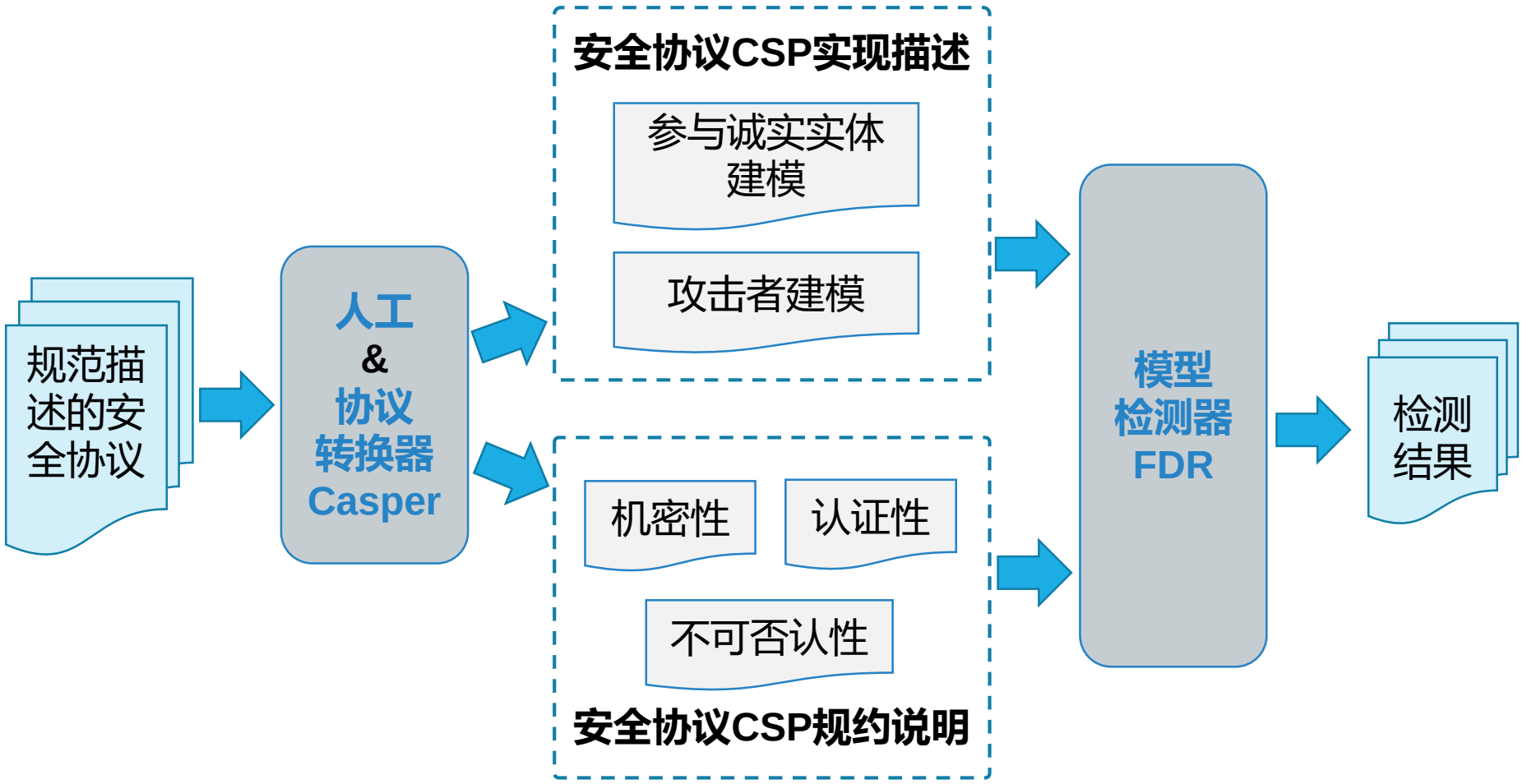
- **系统规约模型**：可看作系统任务书，也就是系统的功能要求；
- **系统实现模型**：可看作系统的某个具体实现，可以有多种；
- 通过测试**所有Q的迹是否属于P的迹**来测试某个系统实现是否符合规约，若都属于则符合规约，否则不符合。
- 测试工具：**FDR（ https://cocotec.io/fdr/ ）**。

FDR简介

- FDR：Failures Divergences Refinement
- 最新版本是2016发布的FDR4（1991年第一个版本）
- FDR is a refinement checker for the process algebra CSP. It allows processes to be defined in a machine-readable version of CSP, CSP_M, and is then able to check various assertions about these processes.



利用FDR进行协议分析的流程



CSP网络模型

可信环境

- 安全协议是由一系列传送消息的并行进程相互作用完成的。
- 在利用CSP建模时，将运行协议的主体称为**代理**，典型的代理有3类：**协议发起者**、**消息响应者**以及**服务器**。
- 各代理通过**信道**进行通信，每一个代理进程都有**接收信道（Receive）**和**发送信道（Send）**。
- 将输入输出形式分别表示为**receive.a.b.m**和**send.a.b.m**，其中a、b分别表示发送者和接收者的名字，m表示消息的内容。

CSP网络模型

- CSP网络模型下的可信环境

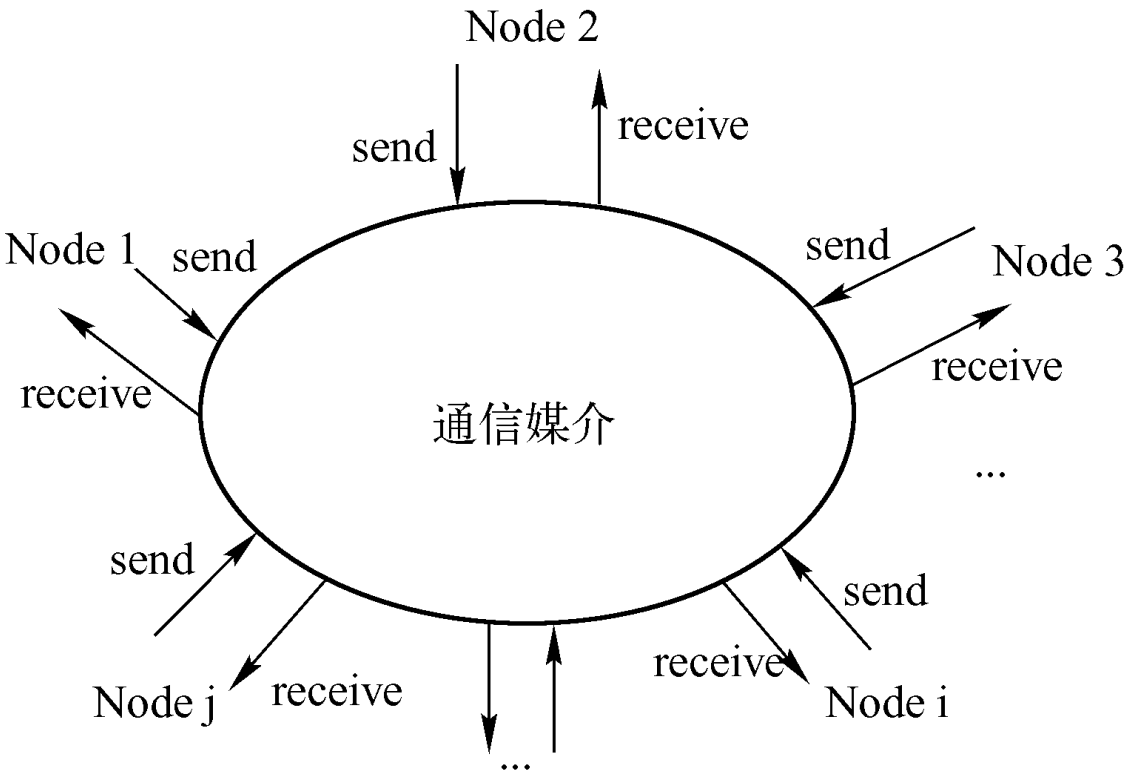


图 CSP网络模型下的可信环境

- **已发送的消息**是被发送到一个通信媒介上
- **接收到的消息**也是从通信媒介那里产生的
- **不能**肯定
  - 该消息**既定的目的地**
  - 协议所**规定的消息源**
- 总是**能**保证
  - **密钥**的新鲜性
  - **nonce值**的新鲜性

CSP网络模型

- 所有通信都通过入侵者的网络模型

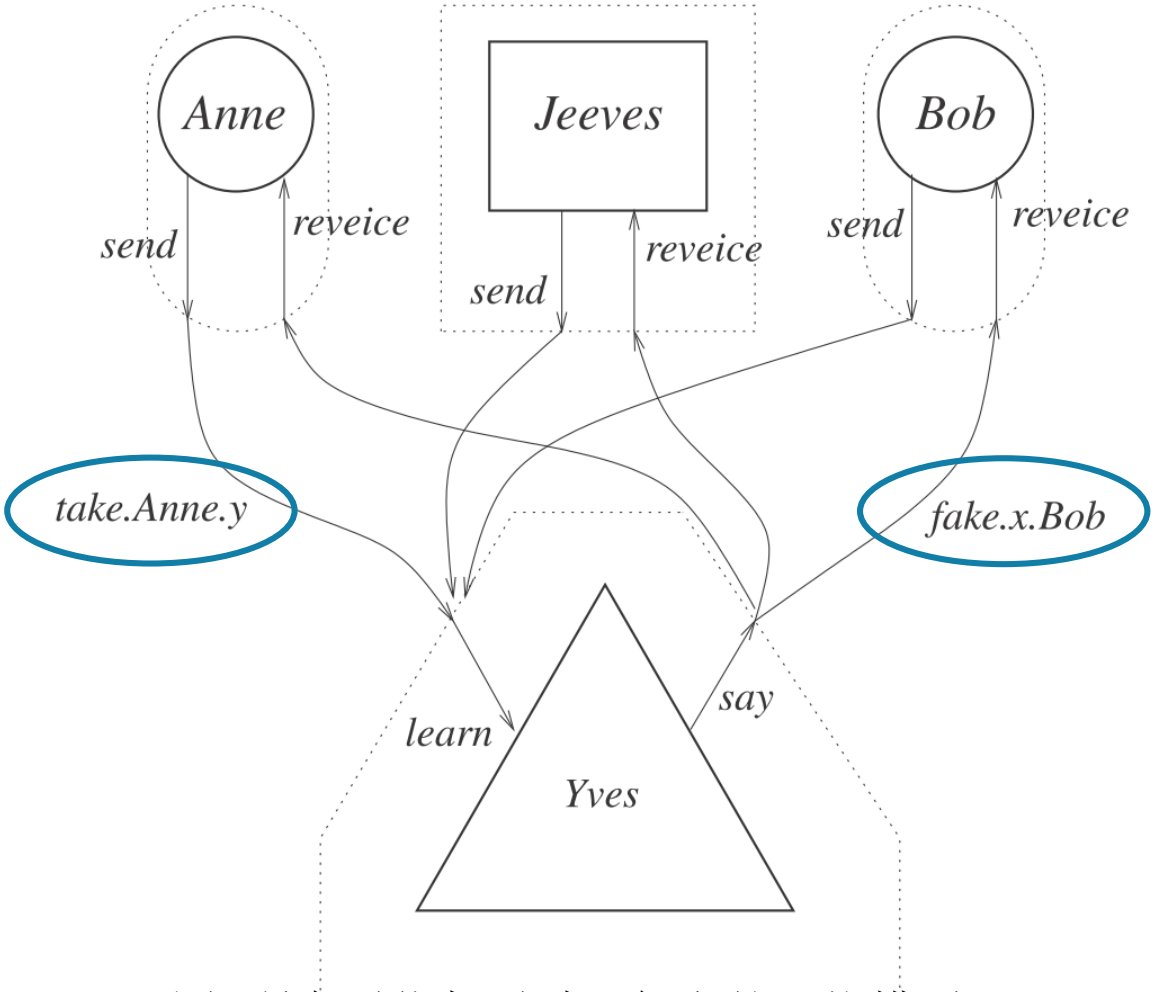


图 所有通信都通过入侵者的网络模型

CSP网络模型

- 按照保密特性要求，入侵者不能够知道k_{ab}。
  - 为了表示该特性，可注意k_{ab}是否出现在say信道中。
  - 为了把say信道的描述用法与普通的给协议代理提供消息的用法分开，新引入一个**leak信道**从say信道接受输出。

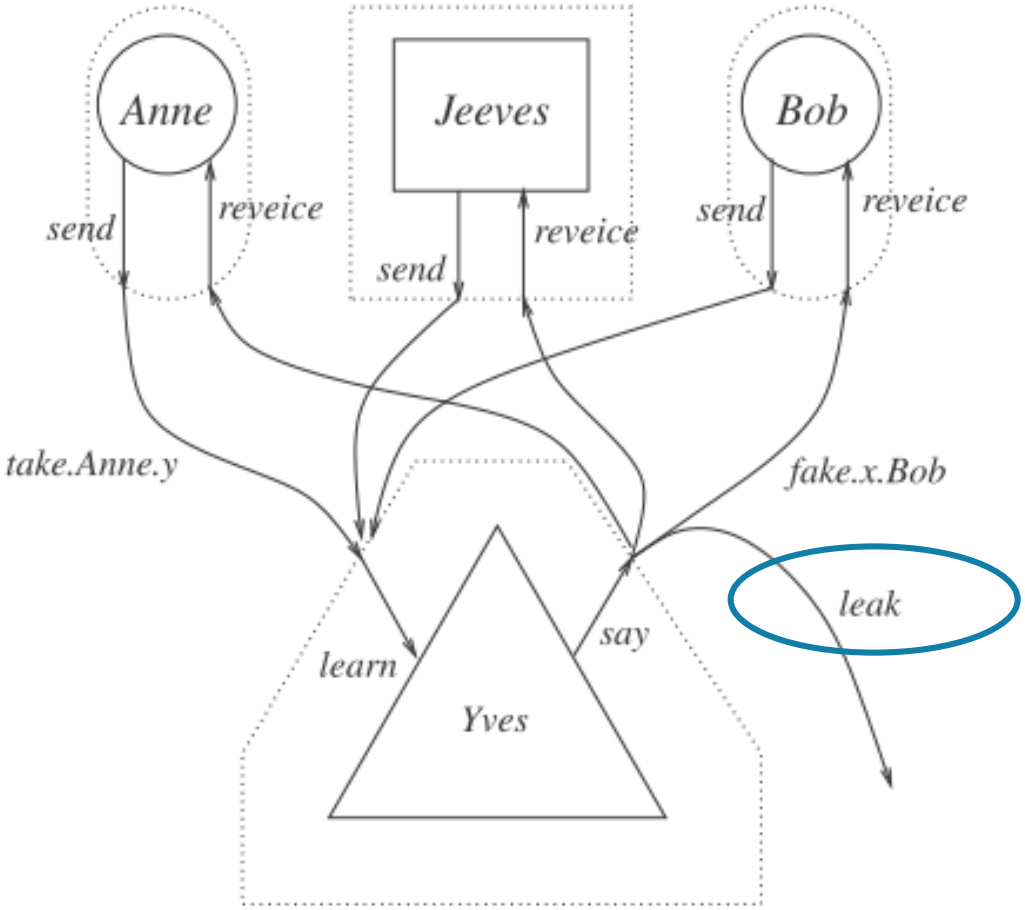


图　引入leak信道得到的网络图示

利用CSP建模协议例子

- 考虑一个简化的安全协议
- 只有协议发起者Alice和协议响应者Bob
- Alice使用的密钥为Ka，包括：$PK(a)/SK(a)$；
- Bob产生秘密信息Sb，使用的密钥为Kb，包括：$PK(b)/SK(b)$；
- **协议目标**：利用a和b的公钥分发一个会话密钥k和秘密信息s
- **协议流程**：

$$\text{Message } 1. \quad a \rightarrow b \ : \ \{\![[k]_{SK(a)}]\!\}^{\mathsf{a}}_{PK(b)}$$

$$\text{Message } 2. \quad b \rightarrow a \ : \ \{\![s]\!\}^{\mathsf{s}}_{k}$$

表示非对称加密

表示对称加密

利用CSP建模协议例子

- **系统模型：**
  - 初始者Alice, 使用会话密钥Ka
  - 响应者Bob, 使用秘密信息Sb
  - **引入攻击者Mallory，完全控制网络**

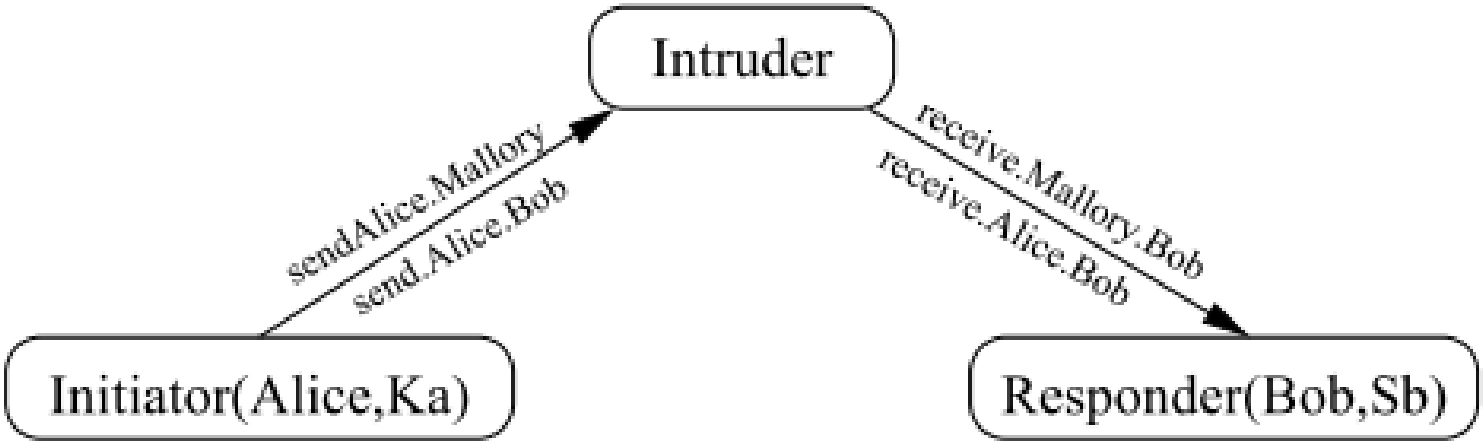


Figure 1. The system

系统实现建模

基本标记

Encryption子类型

$$PK(a) = PK\_.a, \qquad SK(a) = SK\_.a,$$
$$Agent = \{Alice, Bob, Mallory\}, \qquad Honest = Agent - \{Mallory\},$$
$$Secret = \{Sb, Sm\}, \qquad SessionKey = \{Ka, Km\},$$
$$PublicKey = \{PK(a) \mid a \in Agent\}, \qquad SecretKey = \{SK(a) \mid a \in Agent\},$$
$$AllKeys = SessionKey \cup PublicKey \cup Secretkey.$$

配对密钥计算函数

$$inverse(Ka) = Ka, \qquad inverse(Km) = Km,$$
$$inverse(PK\_.a) = SK\_.a, \qquad inverse(SK\_.a) = PK\_.a.$$

信道

channel $send, receive : Agent.Agent.Msg.$

系统实现建模

- **协议发起者：**

$\square_{i:I} P(i)$ 代表从I中选择一个i，执行过程P(i);

从环境接收消息，告诉它与b一起运行协议

$$
\begin{aligned}
&Initiator(a,k) = \\
&\quad \square_{b:Agent}\ env.a.(Env0, b) \rightarrow \\
&send.a.b.(Msg1, Encrypt.(PK(b), Encrypt.(SK(a), k))) \rightarrow \\
&\quad \square_{s:Secret}\ receive.b.a.(Msg2, Encrypt.(k, s)) \rightarrow STOP.
\end{aligned}
$$

- **协议响应者：**

$$
\begin{aligned}
&Responder(b,s) = \\
&\quad \square_{a:Agent, k:SessionKey} \\
&\qquad receive.a.b.(Msg1, Encrypt.(PK(b), Encrypt.(SK(a), k))) \rightarrow \\
&send.b.a.(Msg2, Encrypt.(k, s)) \rightarrow STOP.
\end{aligned}
$$

- **组成系统：**

$$System_0 = Initiator(Alice, Ka) \ ||| \ Responder(Bob, Sb).$$

系统实现建模

攻击者建模

攻击者了解的*Fact*

$$
\begin{aligned}
Fact = \{&Encrypt.(PK(b), Encrypt.(SK(a), k)) \mid \\
&k \in SessionKey, a \in Agent, b \in Agent\} \cup \\
&\{Encrypt.(k, s) \mid k \in SessionKey, s \in Secret\} \cup \\
&\{Encrypt.(SK(a), k) \mid k \in SessionKey, a \in Agent\} \cup \\
&Agent \cup SessionKey \cup Secret \cup SecretKey \cup PublicKey.
\end{aligned}
$$

Fact1=M1

Fact2=M2

Fact3

攻击者推演新的*Fact*的规则

$$
\begin{aligned}
\{f, k\} &\vdash Encrypt.(k, f), && \text{for } k \in AllKeys, \\
\{Encrypt.(k, f), inverse(k)\} &\vdash f, && \text{for } k \in AllKeys, \\
\{f_1, \ldots, f_n\} &\vdash Sq.\langle f_1, \ldots, f_n\rangle, && \\
\{Sq.\langle f_1, \ldots, f_n\rangle\} &\vdash f_i, && \text{for } i = 1, \ldots, n.
\end{aligned}
$$

$X \vdash f$： 如果给定一组事实X，他能够推出事实 f

系统实现建模

攻击者建模

攻击者能力描述：

表示攻击者当前的知识集合

$$
\begin{aligned}
&Intruder_0(S) = \\
&\quad hear?f : MsgBody \rightarrow Intruder_0(S \cup \{f\}) \\
&\quad \Box \\
&\quad say?f : S \cap MsgBody \rightarrow Intruder_0(S) \\
&\quad \Box \\
&\quad leak?f : S \cap Secret \rightarrow Intruder_0(S) \\
&\quad \Box \\
&\quad \Box_{f:Fact,\, X \subseteq S,\, X \vdash f,\, f \notin S} \; infer.(f, X) \rightarrow Intruder_0(S \cup \{f\}).
\end{aligned}
$$

攻击者初始知识集合

$$IIK = Agent \cup \{PK(a) \mid a \in Agent\} \cup \{SK(Mallory), Km, Sm\}.$$

攻击者知道代理的标识、所有的公钥、自己的私钥、自己的秘密密钥Km和秘密信息Sm

系统实现建模

- 攻击者建模
  - 攻击者任意随机行为模拟：

$$Intruder_1 = chase(Intruder_0 \setminus \{\!|infer|\!\}).$$

  - 信道重命名：

$$\begin{aligned} Intruder = Intruder_1 \\ & [\![send.a.b.(l,m)/hear.m \mid \\ & \quad a \in Agent - \{Mallory\}, b \in Agent, (l,m) \in Msg]\!] \\ & [\![receive.a.b.(l,m)/say.m \mid \\ & \quad a \in Agent, b \in Agent - \{Mallory\}, (l,m) \in Msg]\!] \end{aligned}$$

  - 组成系统

$$System = System_0 \underset{\{\!|send,receive|\!\}}{\parallel} Intruder.$$

协议安全性质的CSP描述

- **1．保密性**
  - 如果要在协议运行过程中规定某一数据项m是安全的，那么在协议运行描述结束的时候应该插入消息*Claim_Secret*。
  - 保密性表明：只要声明了m具有保密性，入侵者就不能够得到m。

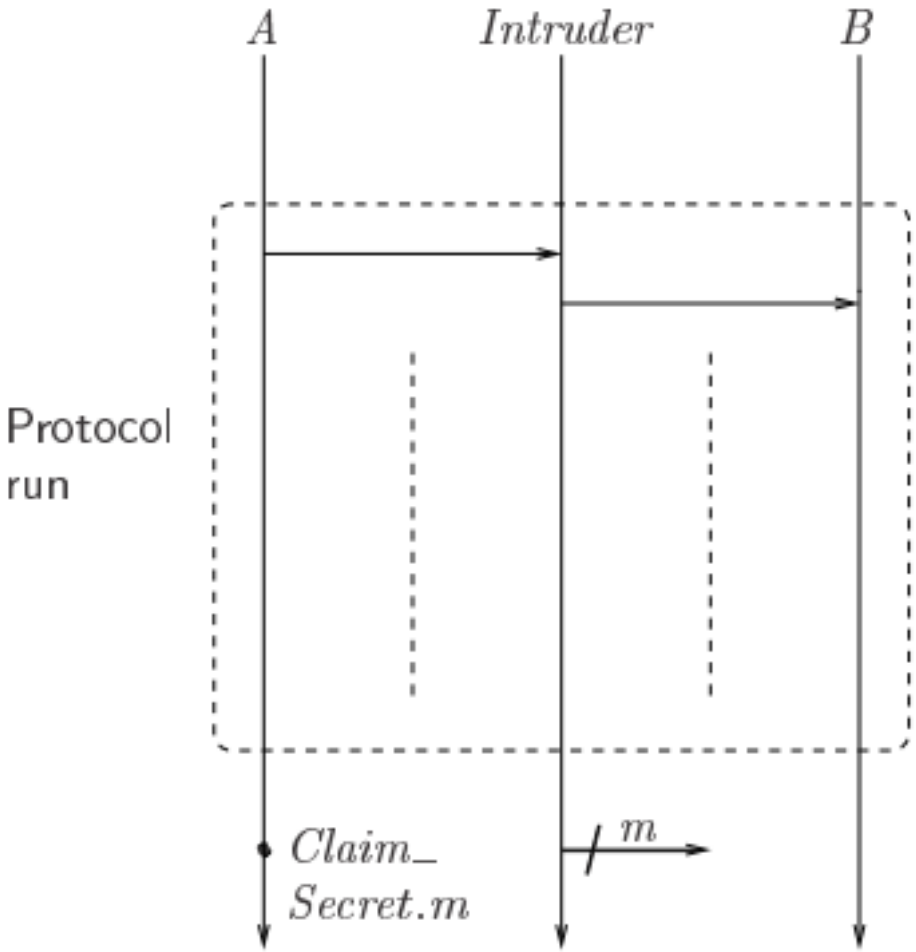


图 保密声明

机密性的CSP描述

- 上面简单认证协议的**机密性目标**描述
  - **目标1**：如果协议发起方a从响应方b接收到了**s**，b不是攻击者，则攻击者无法获得**s**；
  - **目标2**：如果协议响应方b给发起方a发送了**s**，a不是攻击者，则攻击者无法获得**s**。
- 可以同时测试这两个目标

机密性的CSP描述(系统实现建模-更新)

- **协议发起者：**

$$
\begin{aligned}
&Initiator(a,k) = \\
&\quad \Box_{b:Agent}\ env.a.(Env0,b) \rightarrow \\
&send.a.b.(Msg1, Encrypt.(PK(b), Encrypt.(SK(a),k))) \rightarrow \\
&\quad \Box_{s:Secret}\ receive.b.a.(Msg2, Encrypt.(k,s)) \rightarrow STOP.
\end{aligned}
$$

claimSecret.a.**s**.b

- **协议响应者：**

$$
\begin{aligned}
&Responder(b,s) = \\
&\quad \Box_{a:Agent,k:SessionKey} \\
&\qquad receive.a.b.(Msg1, Encrypt.(PK(b), Encrypt.(SK(a),k))) \rightarrow \\
&send.b.a.(Msg2, Encrypt.(k,s)) \rightarrow STOP.
\end{aligned}
$$

claimSecret.b.**s**.a

- **组成系统：**

$$
SecretSystem = Initiator(Alice, Ka) \ ||| \ Responder(Bob, Sb).
$$

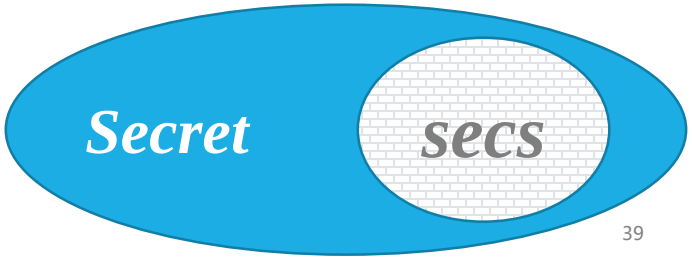
机密性的CSP描述(系统规约建模)

- 定义SecretSpec

$$
\begin{aligned}
&SecretSpec = SecretSpec'(\{\}),\\
&SecretSpec'(secs) =\\
&\quad claimSecret?a?s?b \rightarrow\\
&\qquad \text{if } b \in Honest \text{ then } SecretSpec'(secs \cup \{s\}) \text{ else } SecretSpec'(secs)\\
&\quad \Box\\
&\quad leak?s : \underline{Secret - secs} \rightarrow SecretSpec'(secs).
\end{aligned}
$$

- 通过检查SecretSystem的每个迹是否也是SecretSpec的迹来测试保密属性；
- 使用FDR来测试

$$SecretSpec \sqsubseteq_T SecretSystem.$$



机密性验证

- 通过FDR进行检查，发现机密性无法满足
- 给出的证据：SecretSystem的一个迹不是SecretSpec中的一个迹：

$$\langle claimSecret.Bob.Sb.Alice,\ leak.Sb \rangle.$$

- FDR debugger给出如下的迹

$$\begin{aligned}
&\langle\, env.Alice.(Env0, Mallory),\\
&\ send.Alice.Mallory.\\
&\qquad (Msg1, Encrypt.(PK_.Mallory, Encrypt.(SK_.Alice, Ka))),\\
&\ receive.Alice.Bob.(Msg1, Encrypt.(PK_.Bob, Encrypt.(SK_.Alice, Ka))),\\
&\ send.Bob.Alice.(Msg2, Encrypt.(Ka, Sb)),\\
&\ leak.Sb\,\rangle.
\end{aligned}$$

- 解读为：

$$\begin{aligned}
&\text{Message } 0. && \rightarrow Alice && : Mallory\\
&\text{Message } 1. && Alice \rightarrow Mallory && : \{\!|[Ka]_{SK(Alice)}|\!\}^{a}_{PK(Mallory)}\\
&\text{Message } 1. && I_{Alice} \rightarrow Bob && : \{\!|[Ka]_{SK(Alice)}|\!\}^{a}_{PK(Bob)}\\
&\text{Message } 2. && Bob \rightarrow I_{Alice} && : \{\!|Sb|\!\}^{s}_{Ka}\\
&\text{The intruder knows } Sb.
\end{aligned}$$

协议安全性质的CSP描述

2．认证性

- 一个认证协议在完成之后应建立一种保证：**如果一次协议运行完成，那么其他各方也要加入这个协议的运行。**
- 协议的一次运行完成是以**Commit事件**来标记的，而且**认证性**要求该Commit事件的发生意味着一个相应的Running事件必须在此前已被其他方执行了。

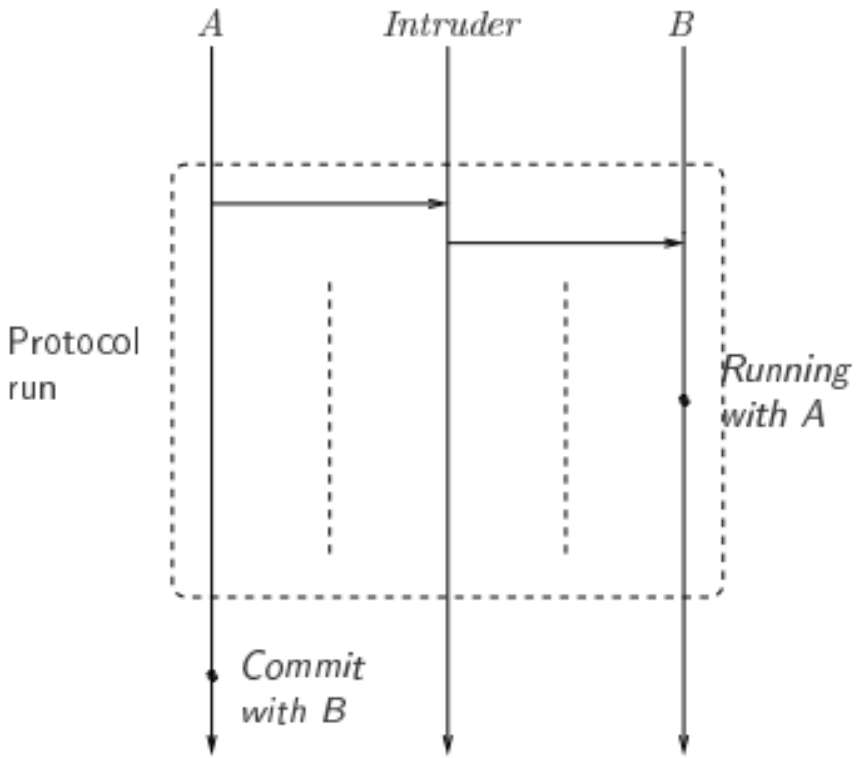


图 认证声明

协议安全性质的CSP描述

- **如果要实现A对B的认证，即B提供某个通信是与A进行通信的保证：**
  - 将事件信号**Commit.A.B**引入A运行协议的描述中，用来标记A对B完成认证的那一点。在A的协议运行中发生**Commit.A.B**就意味着代理A已经与B完成了协议的运行。
  - 同时，引入事件信号**Running.B.A**（A运行中）来标记在A执行完**Commit.A.B**事件时B所达到的点，该事件的出现只意味着代理B正在与代理A运行协议。
  - 如果到事件Commit.A.B执行为止，Running.B.A事件已经发生了，那么认证目标就达到了。

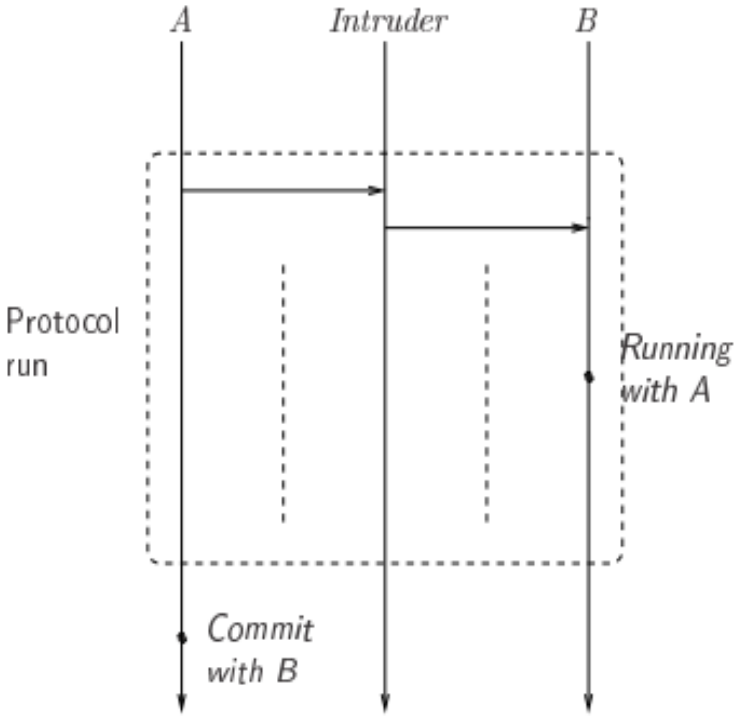


图 把信号加入协议（两个参与者）

认证性的CSP描述

- **在我们例子中的认证性需要认证：**
  - **目标1**：如果协议发起方a和b一起完成了一次协议的运行，则b也完成了一次与a的协议运行，且他们对**秘密信息s**和**会话密钥k**达成了一致？
  - **目标2**：如果协议发相应者b和a一起完成了一次协议的运行，则a也完成了一次与b的协议运行，且他们对**会话密钥k**达成了一致？
- **注意：目标2不包括秘密信息s，因为对b来说，他甚至不知道Alice是否收到了s。**
- **下面来对目标2进行建模（目标1类似）**

认证性的CSP描述(系统实现建模-更新)

- **协议发起者：**

Running.InitiatorRole.a.b.k

$$
\begin{aligned}
&Initiator(a,k) = \\
&\quad \Box_{b:Agent}\ env.a.(Env0,b) \rightarrow \\
&send.a.b.(Msg1, Encrypt.(PK(b), Encrypt.(SK(a),k))) \rightarrow \\
&\quad \Box_{s:Secret}\ receive.b.a.(Msg2, Encrypt.(k,s)) \rightarrow STOP.
\end{aligned}
$$

- **协议响应者：**

Complete.ResponderRole.b.a.k

$$
\begin{aligned}
&Responder(b,s) = \\
&\quad \Box_{a:Agent,k:SessionKey} \\
&\qquad receive.a.b.(Msg1, Encrypt.(PK(b), Encrypt.(SK(a),k))) \rightarrow \\
&send.b.a.(Msg2, Encrypt.(k,s)) \hookrightarrow STOP.
\end{aligned}
$$

- **组成系统：**

$$AuthSystem = Initiator(Alice, Ka) \ ||| \ Responder(Bob, Sb).$$

认证性的CSP描述(系统规约建模)

- 定义AuthSpec
  - 不存在一 一对应关系

$$AuthSpec = Running.InitiatorRole?a.b.k \rightarrow Chaos(\{Complete.ResponderRole.b.a.k\}).$$

  - 需要一 一对应

$$AuthSpec = Running.InitiatorRole?a.b.k \rightarrow Complete.ResponderRole.b.a.k \rightarrow STOP.$$

- 使用FDR来测试

$$AuthSpec \sqsubseteq_T AuthSystem.$$

认证性验证

- 通过FDR进行检查，发现认证目标2无法满足
- 给出如下迹：

$$\langle Complete.ResponderRole.Bob.Alice.Ka \rangle.$$

- FDR debugger给出如下的迹

$$\begin{aligned}
&\langle\, env.Alice.(Env0, Mallory) \\
&\quad send.Alice.Mallory. \\
&\qquad (Msg1, Encrypt.(PK_.Mallory, Encrypt.(SK_.Alice, Ka))) \\
&\quad receive.Alice.Bob.(Msg1, Encrypt.(PK_.Bob, Encrypt.(SK_.Alice, Ka))) \\
&\quad send.Bob.Alice.(Msg2, Encrypt.(Ka1, Sb))\,\rangle.
\end{aligned}$$

- 解读为：

$$\begin{array}{lll}
\text{Message } 0. & \rightarrow Alice & : Mallory \\
\text{Message } 1. & Alice \rightarrow Mallory & : \{\!|[Ka]_{SK(Alice)}|\!\}^{a}_{PK(Mallory)} \\
\text{Message } 1. & I_{Alice} \rightarrow Bob & : \{\!|[Ka]_{SK(Alice)}|\!\}^{a}_{PK(Bob)} \\
\text{Message } 2. & Bob \rightarrow I_{Alice} & : \{\!|Sb|\!\}^{s}_{Ka}
\end{array}$$

- 类似方法测试目标1，FDR没有发现攻击，即目标1满足

思考

- **发生攻击的原因**
  - 攻击成功是由于Bob接受了一个由Alice签名的密钥，该密钥本来是发给Mallory的，并不是发给Bob自己的。

- **如何改进协议?**

$$\text{Message } 1. \quad a \rightarrow b \ : \ \{\lceil a, b, k \rceil_{SK(a)}\}^{\mathsf{a}}_{PK(b)}$$

$$\text{Message } 2. \quad b \rightarrow a \ : \ \{\lvert s \rvert\}^{\mathsf{s}}_{k}$$

本讲小结

- 基于**模型检测**技术的安全协议分析方法
- **基本思路**是利用有限状态机理论，通过定义状态集合及状态迁移函数为安全协议系统**建立模型**；通过穷尽搜索状态空间来判断一些特殊的状态是否可达，或者是否可以生成一条特殊的状态转移路径，并以此检测该模型**是否具备期望的安全性质**。
- **优点**是自动化程度高，可以借助自动分析工具来完成分析过程而不需要用户的参与，并且安全协议存在缺陷时能够自动生成相应的攻击实例。
- 现有CSP、NRL、Murφ和BRUTUS等协议分析方法及分析工具
- **不足**主要体现在：
  - ①容易产生状态爆炸问题，虽然每一种分析工具都采用了不同的状态约减技术以克服该问题的影响，但分析复杂协议仍比较困难；
  - ②在协议描述时**需要指定运行参数**，因而**只能证明协议的不正确性**，而**不能证明协议是正确的**。

参考文献

- Lowe, Gavin. "Analysing Security Protocols using CSP." *Formal Models and Techniques for Analyzing Security Protocols* (2011).

- P.Y.A. Ryan, S.A. Schneider, M.H. Goldsmith, G. Lowe and A.W. Roscoe "The modelling and analysis of security protocols: the csp approach." (2000).

- Clarke, E.M., Henzinger, T.A., Veith, H., & Bloem, R. (2018). Handbook of Model Checking. *Cambridge International Law Journal*.

补充—进一步认识可认证性

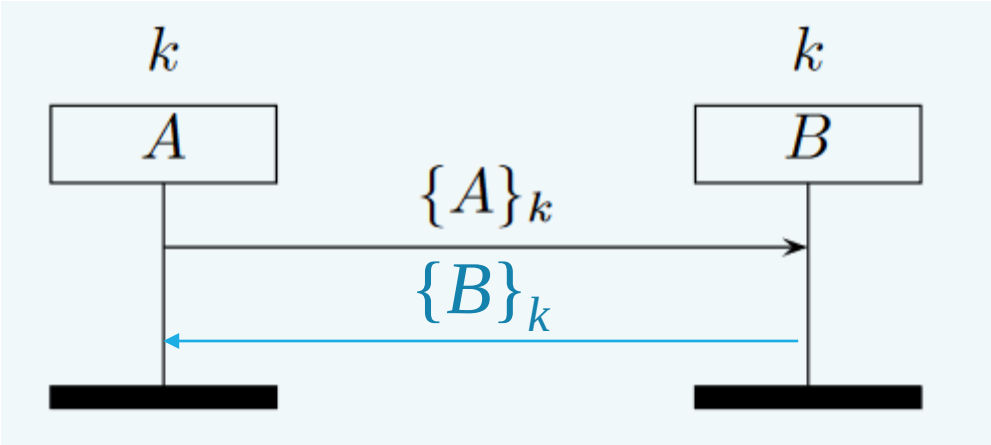
- *<From LOWE>*from an agent A’ s point of view, four levels of authentication between two agents A and B:
- **(i) aliveness,** which only ensures that B has been running the protocol previously, but not necessarily with A;
- **(ii) weak agreement,** which ensures that B has previously been running the protocol with A, but not necessarily with the same data;
- **(iii) non-injective agreement,** which ensures that B has been running the protocol with A and both agree on the data;
- **(iv) injective agreement,** which additionally ensures that for each run of the protocol of an agent there is a unique matching run of the other agent, and prevents replay attacks.

可认证性

- Others point of view:
  - Aliveness
  - recent aliveness
  - weak agreement
  - non-injective agreement
  - Agreement/injective aggreement

aliveness

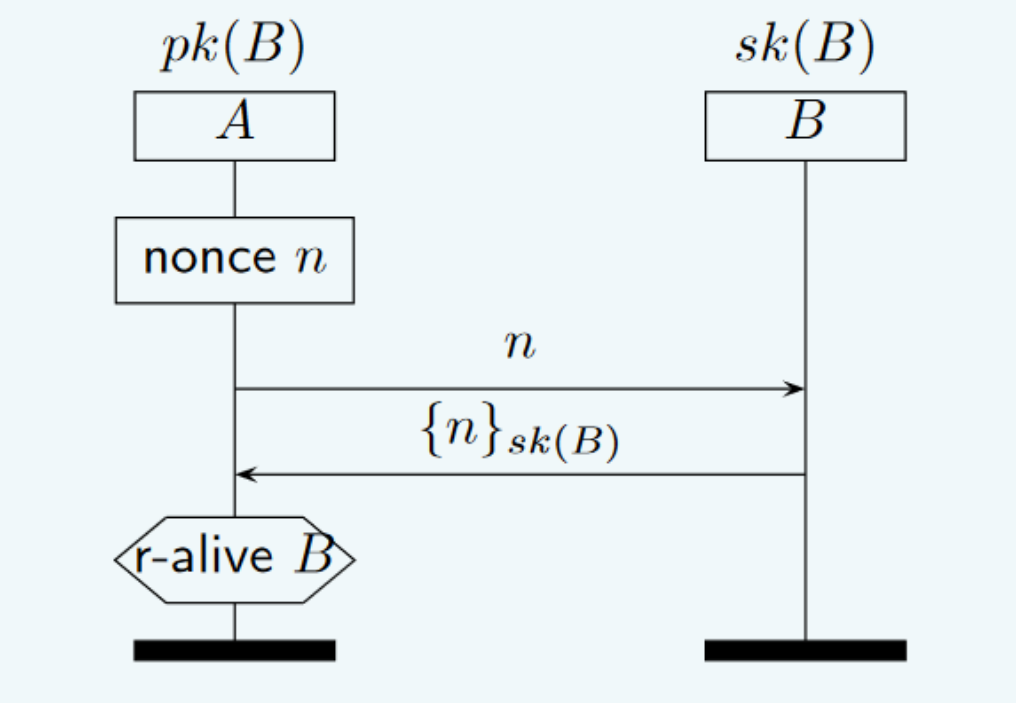
- An authentication protocol provides assurance of the identity of the communicating party in a protocol.



Definition. *(aliveness) A protocol guarantees to an agent a in role A aliveness of an agent b in role B if, whenever a completes a run of role A, believing to be communicating with b, then b has previously been running the protocol.*

recent aliveness

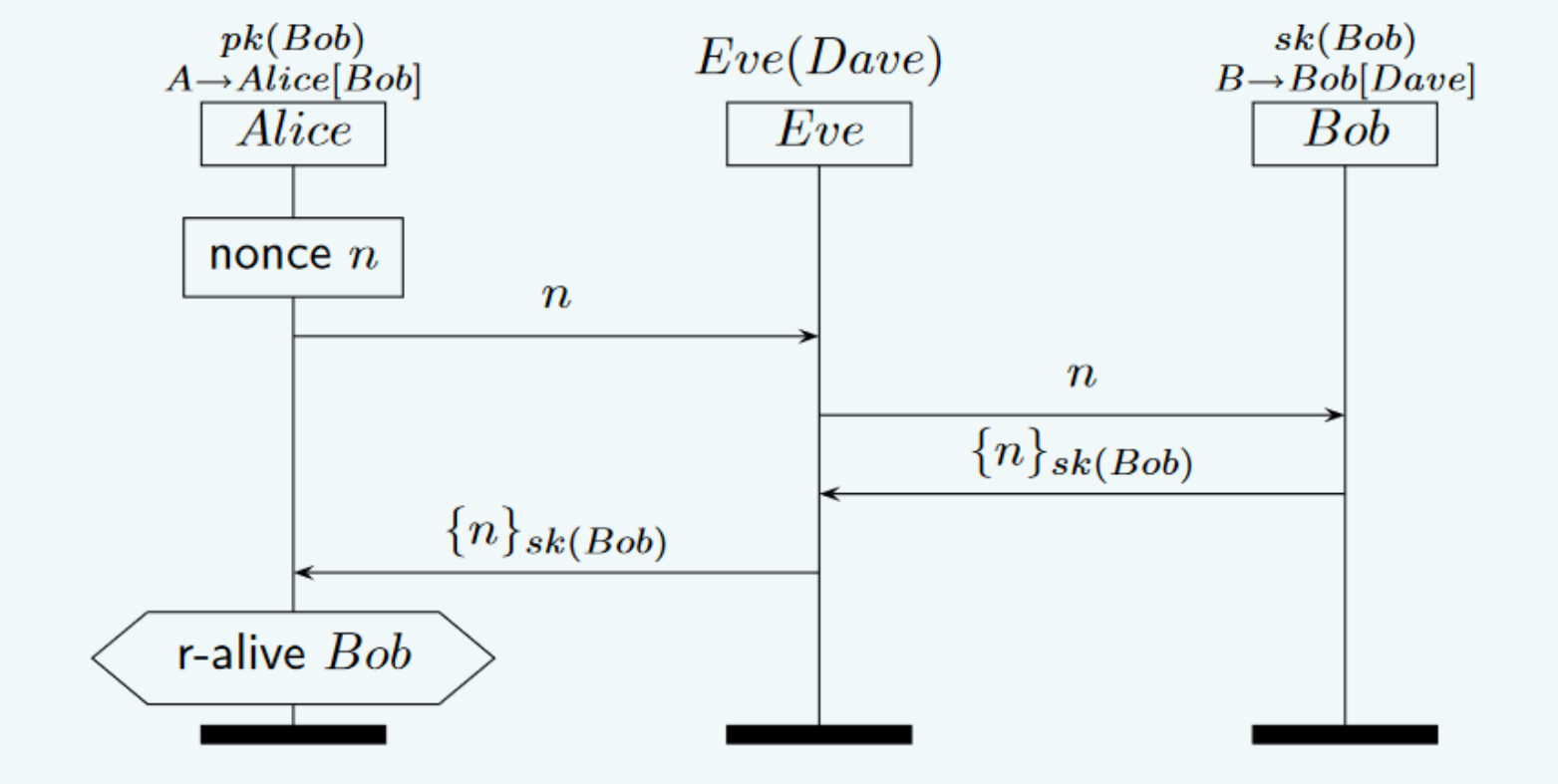
- Protocols satisfying **aliveness** guarantee that the communicating party has sent a message in the past.



Definition. *(recent aliveness) A protocol guarantees to an agent a in role A recent aliveness of an agent b if, whenever a completes a run of role A, believing to be communicating with b, then b has been running the protocol during a's run.*

recent aliveness

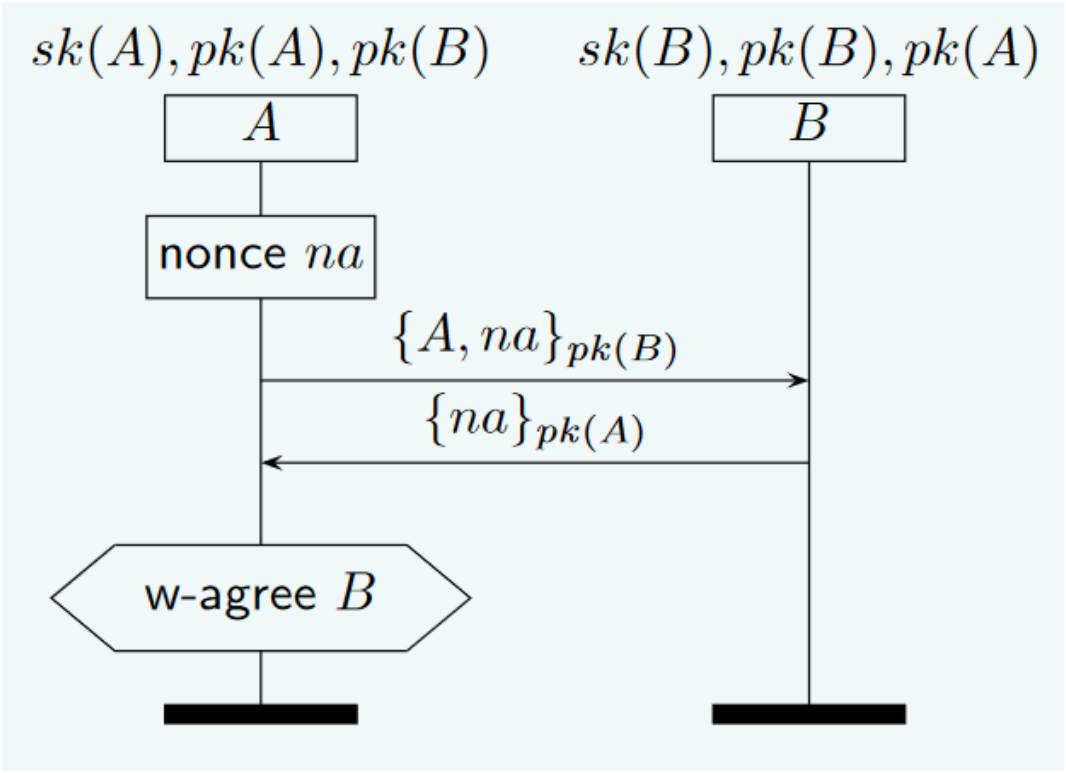
- Protocols satisfying **recent aliveness** guarantee that the communicating party has recently sent “a message”.



weak agreement

Definition. *(weak agreement) A protocol guarantees to an agent a in role A weak agreement of an agent b if, whenever a completes a run of role A, believing to be communicating with b, then*

- *b has been running the protocol believing to be communicating with a.*



non-injective agreement

Definition. *(non-injective agreement) A protocol guarantees to an agent a in role A non-injective agreement of an agent b if, whenever a completes a run of role A, believing to be communicating with b, then*

- *b has been running the protocol believing to be communicating with a and*
- *a and b agree on the contents of all the messages exchanged*

➢**Cannot prevent replay attacks！！！**

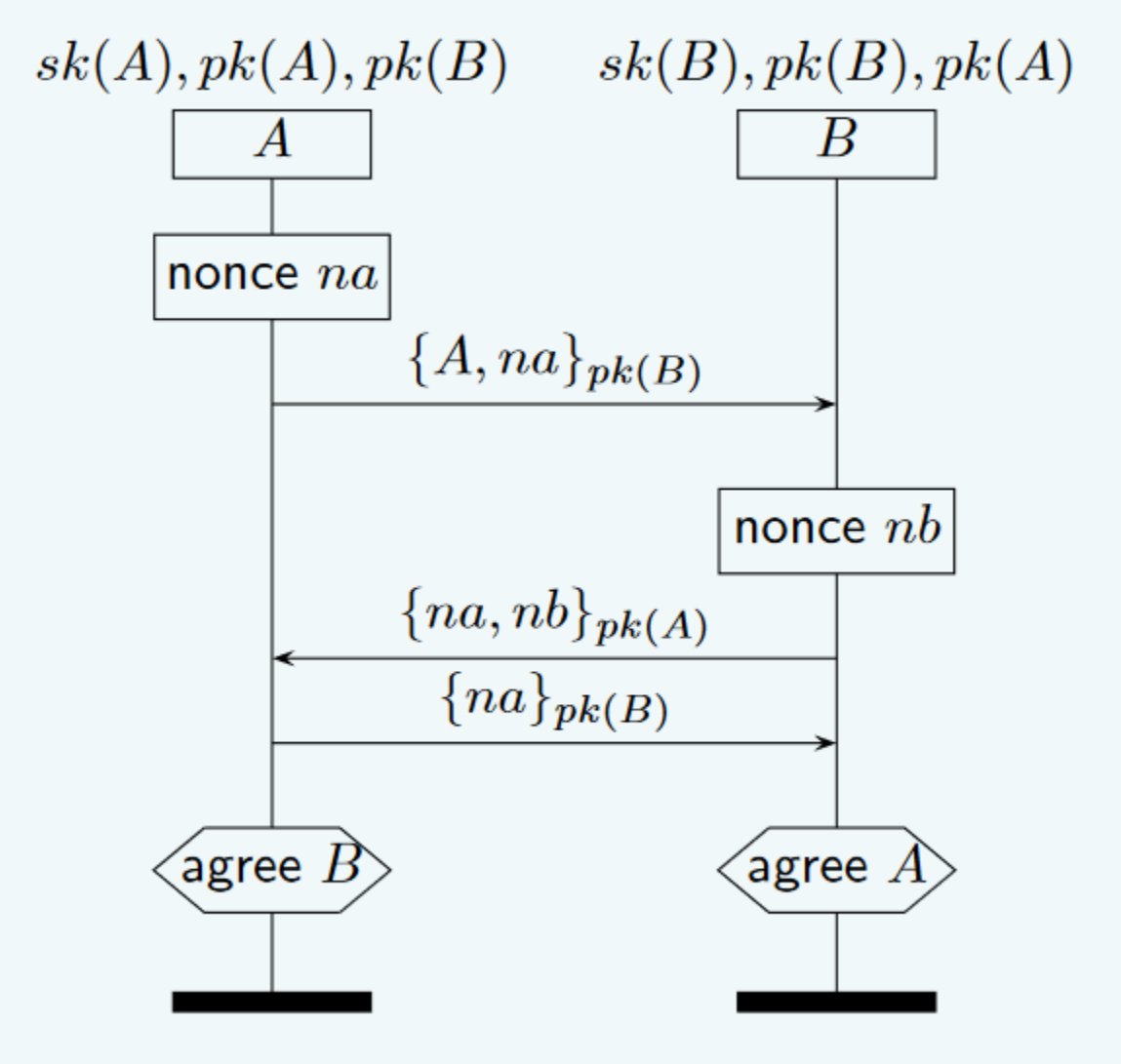
Agreement/injective agreement

Definition. *(agreement) A protocol guarantees to an agent a in role A agreement of an agent b if, whenever a completes a run of role A, believing to be communicating with b, then*

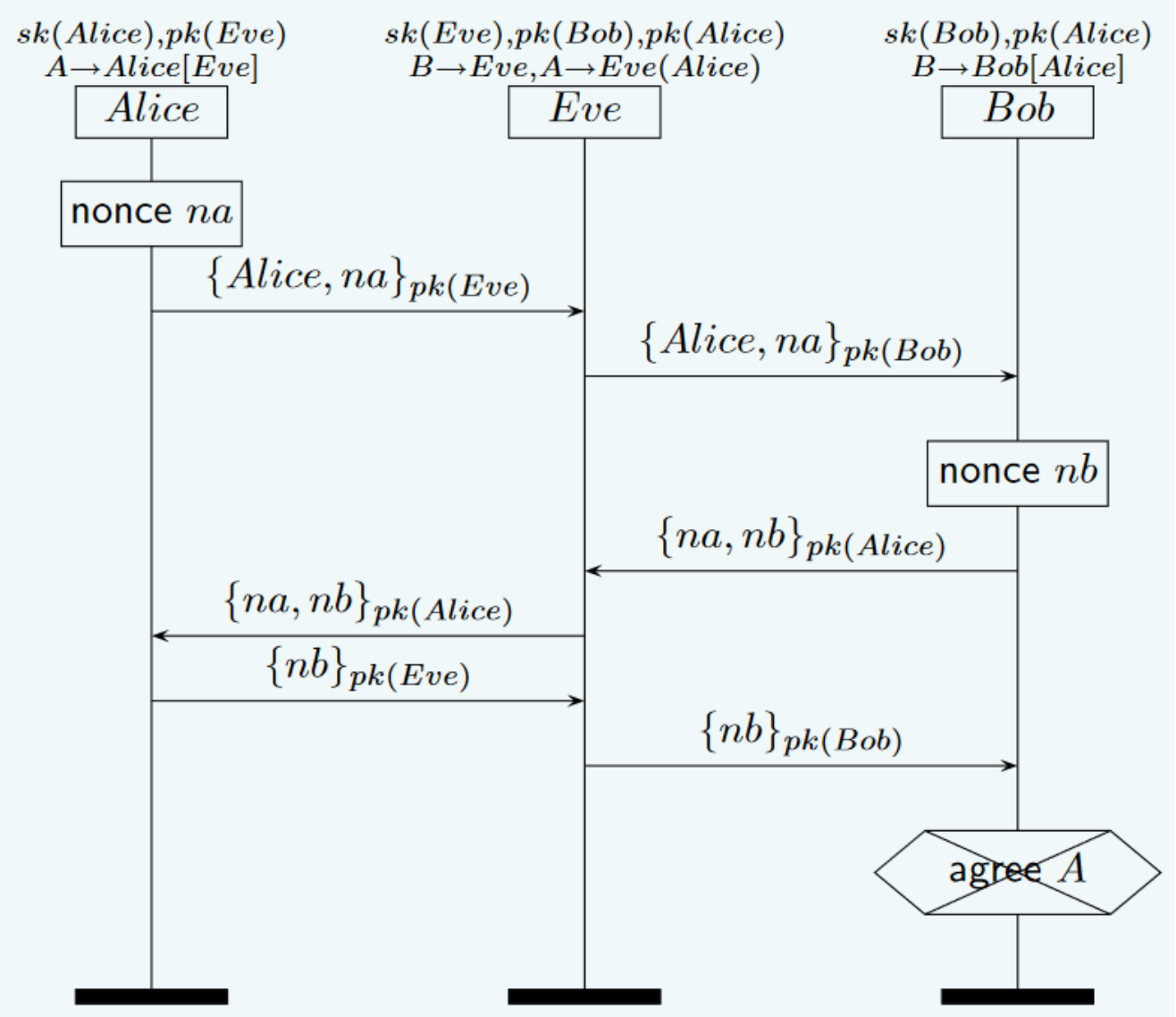
- *b has been running the protocol believing to be communicating with a,*
- *a and b agree on the contents of all the messages exchanged, and*
- *each run of A corresponds to a unique run of B.*

➢ **Can prevent replay attacks！！！**

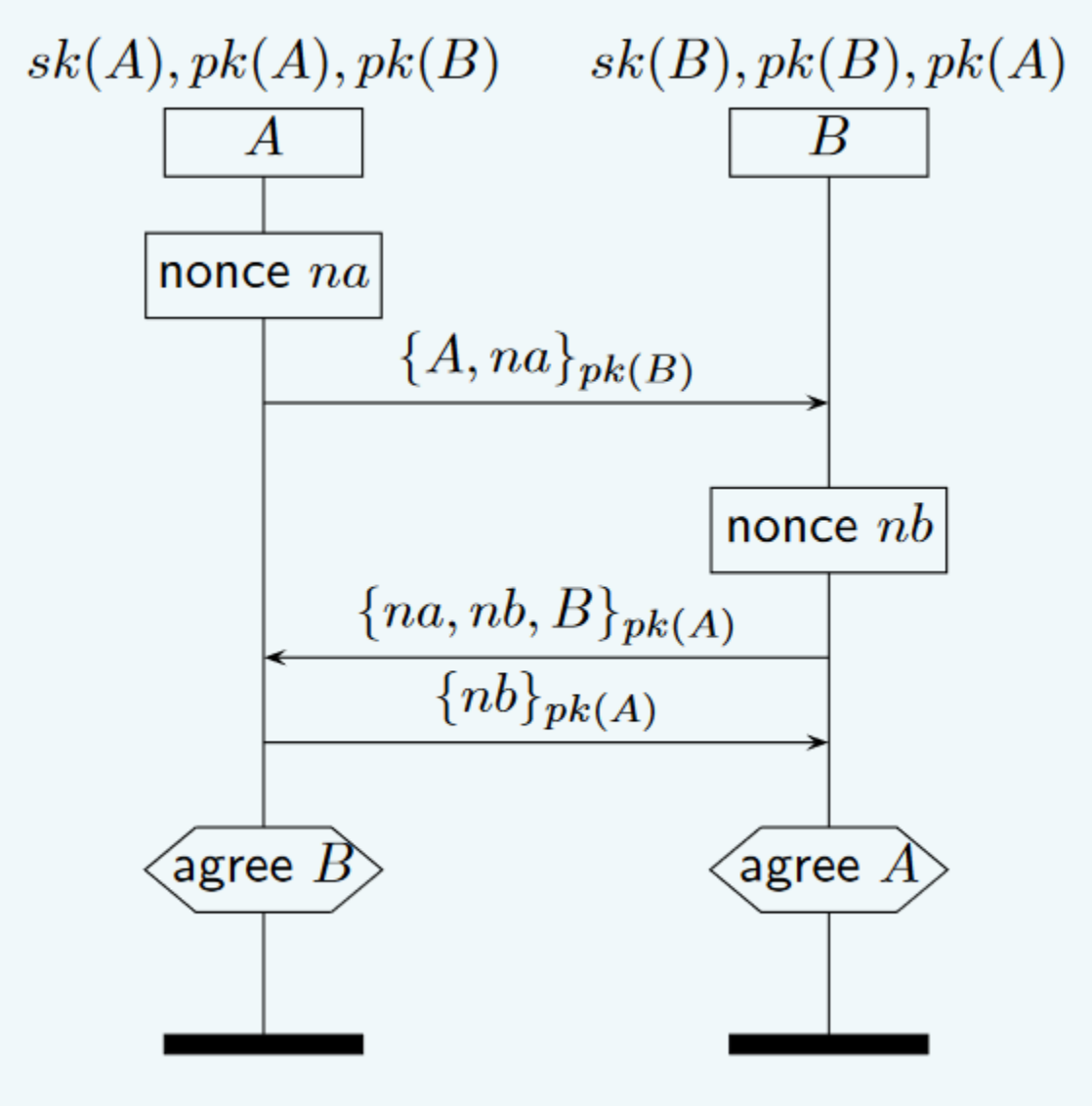
Famous example: Needham-Schroeder



Famous example: Needham-Schroeder



Famous example: Needham-Schroeder-Lowe



谢谢大家！ 欢迎提问！