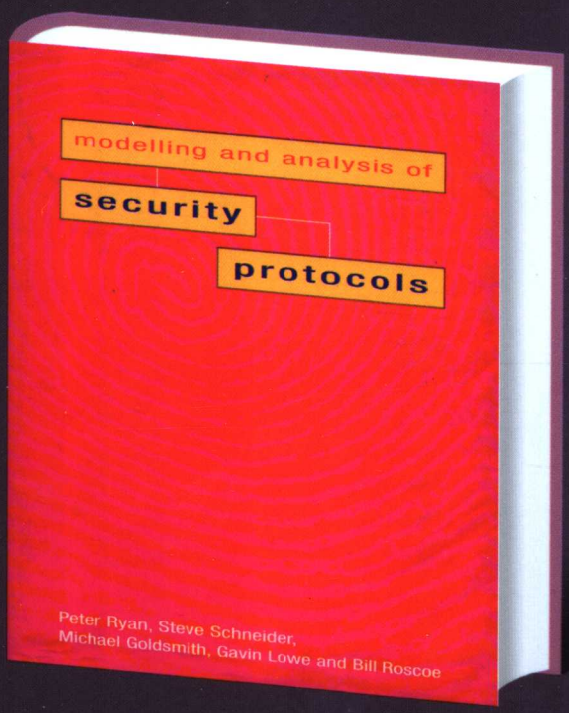
网络与信息安全丛书

安全协议的建模与分析：CSP方式

Modelling and Analysis of Security Protocols

Peter Ryan, Steve Schneider, Michael Goldsmith, Gavin Lowe and Bill Roscoe 著

张玉清　莫　燕　吴建耀　等译



机械工业出版社
CHINA MACHINE PRESS

网络与信息安全丛书

安全协议的建模与分析：CSP 方式

Modelling and Analysis of Security Protocols

Peter Ryan, Steve Schneider, Michael Goldsmith, Gavin Lowe and Bill Roscoe 著

张玉清 莫燕 吴建耀 等译

机械工业出版社

本书主要介绍了安全协议的一种建模与分析方法：CSP（Communicating Sequential Processes，通信顺序进程）方法。本书共有11章和3个附录，主要内容包括：安全协议概述、CSP方法介绍、安全协议的CSP建模方法、协议目标描述、FDR概述、Casper介绍、为FDR进行协议和入侵者编码、分析结果的定理证明、协议的简化转换、其他的安全协议分析方法以及安全协议分析所存在的问题与发展趋势。附录包括：密码学背景知识、具体实例及第8章的详细证明过程。

本书可作为高等院校信息安全、计算机、通信等专业的教学参考书，也可供从事相关专业的教学、科研和工程技术人员参考。

Original English language title: Modelling and Analysis of Security Protocols, by Peter Ryan, Steve Schneider, Michael Goldsmith, Gavin Lowe and Bill Roscoe

ISBN: 0-201-67471-8

图书在版编目（CIP）数据

安全协议的建模与分析：CSP方式/（英）瑞安（Ryan，P.Y.A.），（英）施奈德（Schneider，A.S.A.）著张玉清等译.—北京：机械工业出版社，2005.2

（网络与信息安全丛书）

书名原文：The Modelling And Analysis Of Security Protocols：The Csp Approach

ISBN 7-111-15721-4

Ⅰ.安… Ⅱ.①瑞…②施…③张… Ⅲ.计算机网络-安全-通信协议 Ⅳ.TP393.08

中国版本图书馆CIP数据核字（2004）第125139号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）

策划编辑：赵 慧

责任编辑：罗子超 版式设计：冉晓华 责任校对：李秋荣

责任印制：李 妍

北京机工印刷厂印刷·新华书店北京发行所发行

2005年1月第1版第1次印刷

787mm×1092mm $^{1}/_{16}$·15.5印张·381千字

定价：29.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

本社购书热线电话（010）68993821、88379646

68326294、68320718

封面无防伪标均为盗版

译 者 序

安全协议是建立在密码体制基础上的一种交互通信的协议，它运行在计算机通信网或分布式系统中，借助于密码算法来达到密钥分配、身份认证等目的。目前，安全协议已广泛应用于计算机网络与分布式系统中，这包括：Kerberos 协议、SSL 协议和 SET 协议等。虽然安全协议已成为安全通信和信息处理基础设施的重要元素之一，但安全协议的分析和设计仍是一个具有挑战性的工作。尤其是协议的安全性分析和证明，这是一个迄今为止仍然没有根本解决的科学问题。

本书的主要内容就是介绍安全协议的作用、工作原理、安全协议设计所要确保的一些安全特性，以及如何有效地设计和分析安全协议。

安全协议的设计和分析工作是一个非常精细、枯燥，但常常容易出错的过程。虽然安全协议看似简单，有的协议仅由几个执行步骤组成，但因其异常敏感，往往从刚一公布出来，就会被人陆续发现不少漏洞和错误，从而导致了对安全协议的有效攻击。

1996 年，本书作者之一——英国学者 Gavin Lowe 首先启用 CSP 和模型检测技术（model checking）对安全协议进行分析。通信顺序进程 CSP（Communicating Sequential Processes）是著名计算机科学家 C. A. R. Hoare 为解决并发现而提出的代数理论。Gavin Lowe 采用了 CSP 模型和 CSP 模型检测工具 FDR 来分析 Needham-Schroeder 公钥协议，并成功地找到一个以前从未发现的攻击。模型检测技术应用于安全协议分析的成功，使学者们相继投入到安全协议形式分析这个研究热点领域，并取得了一些卓有成效的成绩。

本书的内容就是在这种背景下形成的，五位作者都是国际安全协议研究领域的权威学者，为此我们将本书推荐到机械工业出版社，并将其翻译出来供国内信息安全同行们参考。同时本书也是我们在国家自然科学基金（编号：60102004，60273027）资助下完成的一项工作。

参加本书翻译的人员有：张玉清、莫燕、吴建耀、王春玲等，全书由张玉清统稿。本书不当之处在所难免，恳请读者批评指正。联系 E-mail：zhangyq@nipc.org.cn。

译 者

原 书 序

信息的价值和它所能传达的力量已经被世人公认。现在与以前相比，信息对社会的作用越来越重要。与此同时，也必须保证信息的完整性、机密性和真实性。

安全协议是保密通信和信息处理基础设施的重要组成之一。当然，安全协议并不是保证安全特性的惟一要素。比如，好的加密算法和用于保护重要资料的系统保密措施，是必不可少的。然而，协议可被视为安全体系中的基石。它使得各代理之间能够互相认证，并建立具有新鲜性的会话密钥以进行相互信任的通信，它还可以保证数据和服务的真实性等。

本书的目的

本书的主要内容是介绍有关安全协议的作用，安全协议是如何工作的，设计安全协议所要确保的一些安全特性，以及如何设计和分析安全协议。

几乎从安全协议一产生，它的设计和分析工作就被认为是一件非常精细、枯燥，并且容易出错的过程。安全协议给人以看似简单的假象，但是其中往往隐藏了惊人的细节和错误。虽然开发相应的体系和工具以对安全协议的属性进行推理验证的尝试，可以追溯到20年前，但是它在安全研究团体中依然被认为是非常活跃并易出成果的领域之一。相关的历史背景综述请参见第9章。

本书将提出一种特殊的方法去验证安全协议，此方法是由本书作者们所开发的。这种方法首次用进程代数和模型检测解决问题，所讨论的这种进程代数就是CSP（Communicating Sequential Processes，通信顺序进程）。

只要在体系结构上实施良好的密码算法就能使体系结构安全，这是一个错误的概念。诚然，好的密码算法是重要的，但同时也要看到，采用了高级密码算法的体系结构，由于使用了不好的协议设计，从而使得门户大开的情况依然可能出现。

我们希望广大读者在学完本书后，能够对安全协议的作用有一个正确的理解，能够了解安全协议是如何运作的，并且对安全协议所寻找的种种漏洞有所掌握。特别是能够更好地领会那些将安全目标准确化的精妙之处，安全协议不仅是安全目标的保证，还是制定安全目标的关键，就像那些基本原理和环境假定一样重要。

希望读者能够通过阅读本书获取足够的知识和热情，并能将这些所学的手

段和技术应用于自身现有的协议中，无论是现实的协议还是虚构的协议。或许一部分读者将会被这个具有挑战性和吸引力的领域中的一些公开问题所吸引，而投身到这项工作中来。

本书的结构

本书主要是应用基于进程代数 CSP 的特殊方法分析和验证安全协议。这种特殊的方法有许多方面，本书将采用一个连续的例子：Yahalom 协议，将这些方面联系起来。

绪论部分对安全协议的概况作了简要介绍。绪论包括了安全协议设计中出现的问题，安全协议结构中使用的密码体制，安全协议所期望的应有属性，破坏安全协议的种种攻击手段。这部分内容还讨论了 CSP 方法及其支持的工具，还介绍了 Yahalom 协议和其他几种协议的例子。

第 1 章主要对 CSP 方法及其相关方面作了概括性介绍。CSP 包含了一种语言以及为系统建立模型和对这种模型进行形式化分析所需要的基本理论，其中这种模型是由相互作用的多个部分组成的。本章介绍了能够描述各个要素的语言模块，并且讨论了这些要素如何构成一个系统。此外，本章还包含巧妙的论述、验证和性质导向规范的内容等。本章的结尾讨论了如何对离散时间建立模型。

第 2 章讲述了如何用 CSP 方法对安全协议建立模型，这里提到的安全协议包含了很多通信要素，并且非常适合用 CSP 方法分析。对安全协议的各种可能的攻击也必须是模型的一部分。本章还介绍了如何联合 Dolev-Yao 方法建立一个恶意环境生成适合于分析的系统描述方法。

第 3 章描述了安全协议期望所具有的各种属性，以及如何在 CSP 的体系结构中对安全协议进行形式化描述。本书中主要探讨的是保密性、认证性，也介绍了其他的一些特性。非否认性和匿名性在本章也有了讨论。

第 4 章介绍了 CSP 所支持的模型检测工具：故障发散改进检测器（Failures Divergences Refinement checker，FDR）。本章讨论了该工具的工作原理以及改进检测的本质。

第 5 章主要讲解 Casper 工具。Casper 是一个安全协议的编辑器，它可以把安全协议的高级描述以及安全协议所必需的性质，转化为在第 2 章讨论过的 CSP 的协议模型，并且在此将证明一些命题。然后再用第 4 章的 FDR 模型检测器对这种 CSP 协议模型进行分析。

第 6 章详细地讨论了 Casper 所实现的一些 CSP 模型，尤其讨论了恶意环境中如何建立模型以使模型检测器的分析更加有效。

第 7 章对安全协议的 CSP 模型进行了直接的证明。本章引入了“阶函数”

来证明协议的正确性。这使得证明更加结构化，从而可以对根据需求的任意大小的协议描述进行验证。此外，本章还讨论了 CSP 方法的定理—求证过程及其所支持的特定工具。

第 8 章讲解了简化尺度问题。众所周知，现实世界中协议的内容都是庞大的，并且在协议的描述过程中存在着大量的细节，这就使得对他们的分析十分困难。本章主要讲述了“简化转换”，这种转换可以在分析完整的协议过程中，首先通过对完整协议的额外细节加以简化，从而再通过特定的方式去正确分析简化后的协议。本章是通过 CyberCash 主序列协议的例子来验证这种方法。

第 9 章对安全协议验证的历史和现状进行了总结。本书中对其他的安全协议验证方法进行了描述，本章主要介绍了那些在协议验证领域较有影响力的分析方法。

第 10 章主要讨论了一些更为广阔的议题，以及一些公开问题和当前研究的领域，并且对这一领域更深层的开发和研究给出指导。本章所讨论的当前研究领域，主要是依靠“数据独立”技术发展的，这对于本书所提出的模型检测方法是非常重要的。“数据独立”技术可以使模型检测的结果应用于任何大小的协议模型中。

附录分为三个部分。附录 A 介绍了一些基础的数学和密码学的背景知识，包括 RSA 算法和 ElGamal 公钥体系；附录 B 主要是将 Casper 应用于 Yahalom 协议的实例，包括 Casper 生成的输入文件和 CSP 模型；附录 C 对第 8 章提到的简化的 CyberCash 协议描述应用 rank 函数进行了验证。

本书有一个相应的网页：www. cs. rhbnc. ac. uk/books/secprot/。该网页提供了本书所有讨论过的工具和使用过的协议实例以及其他。我们建议读者在阅读本书的过程中能够下载相关的工具去对协议进行验证分析。同时，这个网页还提供了大量练习题和答案以及其他一些相关的材料。

致谢

作者在这里要感谢 DERA（防卫和评估研究所，英国）和 MoD 原计划要对战略研究项目（SRP）“安全协议的建模与分析”的资助，虽然他们最终放弃了。作者还要感谢 EPRSC（英国工程学和物理科学研究委员会）和 ONR（美国海军研究办公室）对项目的后期发展进行资助。同时，还要感谢 Inmos，ONR，DERA 和 ESPRIT，数年来对 FDR 的发展所进行的资助。

Peter Ryan 在此感谢皇家霍洛威学院计算机科学系以及剑桥大学微软研究院在完成本书的过程中所给予的热情帮助。

本书的完成得益于与下列人员的合作，他们是：Philippa Broadfoot，Neil Evans，James Heather，Mei Lin Hui，Ranko Lazić 以及正式系统（Formal Systems）的

工作人员。同时，本书的完成还受到与下列人员探讨以及他们建议的影响，他们是：Giampaolo Bella, Steve Brackin, Dieter Gollmann, Andy Gordon, Roberto Gorrieri, Joshua Guttman, Richard Kemmerer, John McLean, Cathy Meadows, Larry Paulson, Matthias Schunter, Paul Syverson 和 Paulo Verissimo。

最后，还要特别感谢 Coby, Helen, Liz, Kate 和 Eleanor 对我们精神上的支持。

目　录

第0章 绪 论

0.1 安全协议

与任何协议一样，安全协议是为了达到某种目的，实体之间相互作用的一系列指定序列组成的。一个成熟的协议，通常包括诸如对理解备忘录等内容的交换，以使得有着潜在利益冲突的参与方之间达成一致。通信协议的设计目的是建立代理之间的通信。也就是说，建立链接、使句法达成一致等。甚至，许多日常生活中人们的行为，譬如从自动取款机中取钱、与其他人拐弯抹角的谈判等，都包含了协议。

安全协议，有时也被称作密码协议，其最终目标是在分布式系统中，提供各种各样的安全服务。这些目标包括：对代理和结点的认证、建立结点之间的会话密钥、确保安全性、完整性、匿名性以及不可否认性等。为了达到以上目的，它们会涉及到结点间消息的交换，而且经常需要可信第三方或者会话服务器的参与。一般而言，它们可以不拘于任何限制地自由运用各种密码机制，譬如对称加密、非对称加密、哈希函数和数字签名等。在一些特定的情况下，还会更多地使用到一些机制（例如，时戳）。我们稍后将会对这些术语进行更详细的解释。

人们很早就认识到，设计与分析安全协议是很难的。这些困难主要来源于以下几方面的考虑：

- 安全协议所要确保的各种性质是非常细微的。即使是表面上定义很简单的消息认证概念，也容纳了若干个细节问题，并且在每个细节上，都有不同的含义。对于这个概念的准确含义，或者退一步，不用准确二字，只说其含义，都依然有着激烈的争论。
- 这些协议存在于一个复杂且充满敌手的环境中。为了能合适地评价他们，我们需要准确地描述和模拟这个环境，这就不得不考虑代理蓄意地去破坏协议的可能性。在本书中，我们把这种充满敌意的代理称作入侵者，为了文字上不至于太单调，有时也将其称作间谍、敌人、攻击者、窃听者或渗透者。
- 完全获知入侵者的能力无疑是极端困难的，或者说是不可能的。但是，最起码我们希望能在最大限度上对这些能力进行模拟，并且这种模拟可以随着新的攻击类型的发现而日益加强。除了操纵在网络上经过的消息以外，入侵者们还掌握密码分析技术，能够进行追踪混乱散发的消息、定时功能、能量波动、概率观测以及其他恶意的活动。譬如，rubber hose-pipe 密码分析（利用非数学方法提取密码变量元素）。
- 高度同步的安全协议，总是使得分析更加具有挑战性。

事实上，安全协议是严格分析技术的最佳选择。它们是任何一个分布式安全结构的重要组成部分，它们非常容易表达，并且很难通过手工来评价。它们看似很简单：文献中到处都是看起来是安全的协议，但到后来，有时是若干年后，才被发现已经成了一些狡猾攻击的牺牲品。在 Roger Needham 的评论中写到“它们只不过是一个只有三行的程序，但人们却依然

能找出错误。”

安全协议已经成为形式化方法研究团体所研究的对象之一，就像水果双翼昆虫与基因研究群体的关系一样。他们简洁且容易操作，但包含了很多问题，对于分析和评估安全，评估系统的工具和技术而言，都是一个挑战。

为了使安全协议更具体化，我们来考虑一个例子：Needham-Schroeder 密钥协议（NSSK）。这是最早的安全协议之一，除了一些细微的弱点（后面会有介绍）以外，NSSK 经受了时间的考验。它形成了广为人知的 Kerberos 认证和授权系统的基本框架[65]。它应用完全对称的加密算法，设计成能够使得两个代理方——Anne 和 Bob，在一个可以信任的服务器 Jeeves 的帮助下，建立一个安全通信信道。如图 0.1 所示。

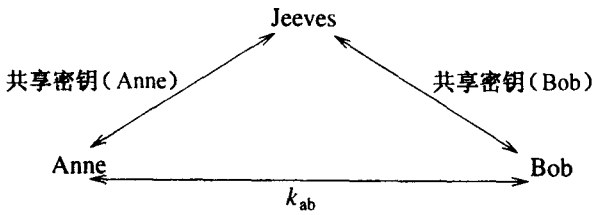


图 0.1　安全通道的建立

最开始，所有已注册的代理，包括 Anne 和 Bob，分别与服务器 Jeeves 共享秘密的长期密钥。这些密钥是截然不同的，所以即使每个人都可以和 Jeeves 安全地通信，但他们之间不能直接地通话。现在假设 Anne 想与 Bob 开始私人谈话。对于他们而言，最大的一种可能就是通过 Jeeves 来传达：Anne 用她与 Jeeves 共享的密钥加密一个消息，并发送给 Jeeves，然后 Jeeves 先解密这个消息，并且用明文把他与 Bob 共享的密钥加密，最终将这个消息发送给 Bob。这种通信方式可以实现，但是很繁琐：它包括了很多的计算，而且 Jeeves 成为了瓶颈，只要他失败，整个对话就会失败。

骗局正是利用了服务器 Jeeves，在必要的时候在 Anne 和 Bob 之间建立一个新的密钥，这样他们之间便可直接通信，而不必通过服务器 Jeeves 的处理。一个很明显的问题是：为什么不直接在最开始的时候在每一对代理之间都提供一个密钥？原因是这样做虽然可以节省一个节点，但是一开始时就需要分配大约 N^2 个互不相同的密钥，这里的 N 是代理的个数。随着 N 的增大，这个方案很快就变得不切实际了，特别是通常这些被分配的密钥中的绝大部分可能永远不会被用到。更糟糕的是一开始我们并不知道将来的用户情况。特别是注册的用户在不断的变化。另外很重要的一点是：密钥必须经常更换，以增加密码破译者的破译难度，并且这样可以避免任何威胁的影响。有了像 Needham-Schroeder 密钥协议一类的体制后，可以使建立所需的新密钥变得更容易，并且使长期密钥，如 ServerKey（a），ServerKey（b）等，变得更坚固。当密码破译者试图只通过传送小量的信息来破译密码时不容易成功。

再回到协议上，已知的事实是如果开始时有 N 个用户，则初始时只需要建立 N 个密钥。当有新的用户注册时，便依照公平的原则直接为每个新用户分配一个新的密钥。

现在假设 Anne 与服务器 Jeeves 共享密钥 ServerKey（Anne），同时 Bob 与服务器 Jeeves 共享密钥 ServerKey（Bob）。

协议步骤如下：

消息 1　$a \rightarrow J$：$a.\ b.\ n_a$

消息 2　$J \rightarrow a$：$\{n_a.\ b.\ k_{ab}.\ \{k_{ab}.\ a\}_{ServerKey(b)}\}_{ServerKey(a)}$

消息 3　$a \rightarrow b$：$\{k_{ab}.\ a\}_{ServerKey(b)}$

消息 4　$b \rightarrow a$：$\{n_b\}_{k_{ab}}$

消息 5　$a \rightarrow b: \{n_b - 1\}_{k_{ab}}$

首先解释一下这些符号，协议的每一步都用一行描述来表示。如

消息　n　$a \rightarrow b$: data

该式表示协议的第 n 步，代理 a 给 b 发送信息 data。消息通常由很多部分互相连接构成。实际上，这些消息也有可能不能到达 b 或者消息被发送给了其他用户。现在只介绍按照协议步骤正常发展的过程。

形如 n_a 的术语表示所谓的 nonce，即它是一个即时产生的、（通常情况下）惟一的且不可被预知的值。下标表明是由谁来产生的，但是需要注意的是：这里使用符号 n_a 只是为了方便标记。然而在实际的协议中，通常这样写并不表示这个值就是由 a 产生的。下面将更详细地讨论 nonce 在协议中的作用。

各种术语的组合可以有以下形式：

- $\{data\}_k$：该符号代表将 data 用密钥 k 加密后得到的值。
- m. n：该符号表示正文 m 后紧跟（连接）着文本 n。

现在按照协议过程一步一步地运行。第 1 步是用户 Anne 通知服务器 Jeeves，她将要与用户 Bob 通话，并且将 nonce 的值 n_a 提供给服务器 Jeeves。收到了用户 Anne 发出的请求后，服务器 Jeeves 将产生一个新密钥 k_{ab}，并在第 2 步中将嵌套加密后的消息返回给用户 Anne。由于外层加密使用的密钥 ServerKey(a) 是用户 Anne 已知的，所以她可以将其解密，解密后用户 Anne 将得到如下内容：

$$n_a.\ b.\ k_{ab}.\ \{k_{ab}.\ a\}_{ServerKey(b)}$$

上式中第 1 项 n_a 就是用户 Anne 在消息 1 中发送给服务器 Jeeves 的 nonce 的值，用户 Anne 将验证这个值与她最初发送的值是否一致，稍后我们再讨论它的重要性。第 2 项应该是用户 Bob 的名字。用户 Anne 将再次检查这个值是否与她当初在消息 1 中发送邀请的用户名一致。而第 3 项就是新产生的密钥 k_{ab}。第 4 项则是采用用户 Bob 与服务器 Jeeves 共享的密钥加密的内容。

协议运行到这一步，用户 Bob 已经知道了新密钥 k_{ab} 的值，并且确定了将与用户 Anne 进行通信。他将新建一个他自己的 nonce 值，用密钥 k_{ab} 加密后，在消息 4 中再将它发回给用户 Anne。用户 Anne 解密这个消息后，得到 n_b 的值。她以某种标准方法（减 1 变换）修改这个值后，将修改后的值用密钥 k_{ab} 加密，再发送给用户 Bob。最后用户 Bob 检验解密后的结果是否满足预定的条件：与 $n_b - 1$ 相等。

那么经过了这么多复杂的步骤后，到底要达到什么目标呢？假设协议顺利进行，用非正式的语言来表述，就是用户 Anne 和 Bob 最终可以用密钥 k_{ab} 共享信息了。实际上，当用户 Bob 收到消息 3 后他们已经共享了信息，那为什么还需要发送后面的 2 条消息，还要使用另外一个 nonce 值呢？其实，消息 4 和消息 5 只是认证信息，是为了确保对方也知道密钥 k_{ab} 而发送的。可知，当用户 Bob 收到消息 3 时，他已经确认用户 Anne 是知道密钥 k_{ab} 的。一旦用户 Anne 得到消息 4，她就能确认用户 Bob 也知道密钥 k_{ab}，从而她也知道用户 Bob 已经确认自己是知道密钥 k_{ab} 的。我们可以继续构造一系列无限多的认证消息，这些消息就会导致形成一个基于知识的知识状态塔状结构。

需要强调的是，以上在用户 Anne 和 Bob 之间达到的结果还只是建立在非正式推理的基础上，但是当收到消息 3 后，用户 Bob 的推理就可以建立在下列几项原则的基础上了：

必须肯定的一点是：除了用户 Bob，只有服务器 Jeeves 知道密钥 ServerKey（b），所以除了他自己，别的用户无法产生这些数据项。同时假设服务器 Jeeves 是诚实可信的，并且只有在收到用户 Anne 发出的请求后，服务器 Jeeves 才能产生这些项，因此这些数据项必须包含在一个运用用户 Anne 的长期密钥加密过的消息中。所以当用户 Anne 向自己发出通话邀请后，只有自己才能收到这个消息。因此，用户 Anne 将收到服务器 Jeeves 发出的用密钥 k_{ab} 加密的正确消息……

但是以上一系列的推理有着相当错误的倾向。例如，在对 Needham-Schroeder 公钥（NSPK）协议的分析中，就能够看到这一点。本书将介绍如何使用一种完全形式化和真正自动化的方法对协议进行推理。

这个协议的最终目标是将经过认证的密钥分配给用户。通常，这个协议可以满足用户 Anne 的请求，为她与用户 Bob 提供一条私有的、经过认证的会话通道。通过此协议，用户 Anne 最终想要的结果是确认服务器 Jeeves 确实为她和用户 Bob 提供了一系列新鲜的密钥 K，以供他们在进一步的通信中使用，并且这些密钥必须是只能被他们使用的。只要这些密钥只为他们所掌握，用户 Anne 就能确信她用密钥 K 加密的消息发出后只有用户 Bob 可以阅读，而且她收到的由用户 Bob 发出的用密钥 K 加密的信息也只有她能阅读，不会被攻击者所窃取。

这个协议说明了很多有趣的问题，在本书中还将出现很多这类问题。要确保在协议提供给用户 Anne 和 Bob 的服务中保持真正的公平，绝非一件容易办到的事情。首先，我们必须确保攻击者无法通过对消息进行一系列的（非密码学）方法的处理，窃取到上述目标的值。要做到这一点依赖于协议设计的正确性，而且它正是本书将主要讨论的内容。蹩脚的设计或者开始时选择了不合适的密码体制，都可能导致产生脆弱点，被攻击者利用。基本上，我们的讨论不会涉及这些问题，除非是在密码代数的性质与协议本身相互作用而使脆弱性提高的情况下，才会讨论这种相互作用的结果。另外一个值得注意的问题是，从用户 Anne 的观点而言，她需要在某种程度上信任系统的远程部分。她必须信任服务器 Jeeves，相信他会按照协议描述的步骤执行，并且使用了一个完善的密钥产生体制。同时，她还需要信任用户 Bob。因为如果 Bob 泄漏了密钥 K，无论是蓄意的还是由于粗心大意（如密钥的存储媒介不安全），都会导致整个方案的安全性或认证性化为泡影。另一方面，用户 Anne 不应该过分信任通信媒介。相反，通常情况下她应该假设媒介是完全开放的，恶意的代理可以在媒介上竭尽所能来破坏安全目标。

注意，初始发起者 Anne 发出的第 1 个消息，在发送时是不会受到攻击的，因此，我们没有必要保证这个消息的保密性和完整性。对于攻击者能够得知用户 Anne 想要与 Bob 通信这一事件，我们总假设它是容易达到的。稍后也将看到，攻击者知道 n_a 的值对他并无用处。所以缺少这个值的保密性并不会造成大的问题，除非攻击者采用流量分析攻击。在这种情况下，所谓完整性是指如何防止攻击者篡改这些值，这样做的惟一后果就是当 Anne 发现从服务器 Jeeves 发来的消息中的验证值与她记录的不一致时，她会停止发送消息。用这种方法可以发动一场拒绝服务攻击，但是保密性和认证性没有被破坏。

另外，现在假设协议采用分组密码体制（明文和密文之间的转换以固定长度的字符为单位进行），而不是流密码体制（明文和密文之间的转换一次只转换一个字符）。我们将在第 0.3 节更为详细地介绍这两种密码体制的区别，但是这一点对用户 Bob 的 nonce 询问是非

常重要的。因为如果采用的是流密码体制，攻击者将能够伪造 $\{n_b - 1\}_{k_{ab}}$。甚至他不需要知道 n_b 和 k_{ab} 的值，只需要对消息 5 的密文进行适当的移位就能实现了。

进一步，我们假设将很多项放在一起加密就会产生绑定这些项的效果。也就是说，攻击者为了替换其中的一项而剪切或者粘贴密文的一部分时，不可能保持原加密的有效性。在需要加密的内容长度超过分组密码长度的情况下，需要一个适当的加密模式来达到以上的效果，我们将在后面详细讨论它。假若缺少这个机制，攻击者 Yves 将有可能对密文的一部分进行剪切、粘贴，将他知道的密文中包含用户 Anne 的身份认证的部分改为自己的，这样就欺骗了 Bob，让 Bob 认为他是与用户 Yves 建立了一条秘密通道，而不是 Anne。

引用这个例子，主要是想用它来说明安全协议所发挥的作用，以及它们通常是如何实现的。这个典型的协议主要的功能是实现各个用户之间的相互认证，而且如果需要的话，用户之间的通信可以是保密的。其他的安全目标当然也可以实现，我们将在下一节介绍几个相关的例子。这里出现的古典加密体制在此有好几种作用：加密以实现保密性；绑定各项内容以确保认证性；使用 nonce 以阻止重放攻击以及诸如此类的功能。这些古典加密体制将会在本书中一再出现。为了使读者能很好地理解这些古典加密体制，我们将在第 0.3 节中对密码学的背景知识作简单介绍。

0.2 安全特性

在此，将对一个安全协议所需要提供的众多重要特性或者服务给出一个直观的描述。当然，在通常情况下，一个协议需要提供的特性和服务只是它的一个子集，这取决于它的应用领域。

这些术语的含义通常被认为是显而易见且广为人知的，然而很多人却发现，如果要他们为这些术语作出精确的定义却相当困难。此外，有时即使在一个简单的文件或设计方案内，经常也会对这些看起来广为人知的术语作出不同的阐述。由于上述原因，我们很有必要为这些术语作出精确的书面定义。例如，绝不应该声称某个协议是“安全的”，甚至是“正确的”。一个协议只能说是对某些给定的精确定义过的性质是正确的，甚至是只对某些特定的几类威胁在各种假想环境中才是正确的。

在后续章节中将概括介绍这些特性的可能形式，这些介绍只是可能的描述而非惟一的。在第 2 章和第 3 章中的介绍 CSP 形式化的过程中，将更为详细地介绍这些术语。

保密性

保密性（secrecy）或者机密性（confidentiality），通常会有很多不同的含义，这取决于应用领域的不同。也许可以将协议设计为一个严格的描述，使得入侵者无法发现合法用户的任何活动。最好的结果是入侵者甚至不能做任何流量分析，更别说推断任何消息的内容了。对此就需要一个无冲突的描述来断言：一个高级用户的活动不应该对系统的低级用户或者外部观察者产生任何明显的影响。可以在譬如文献［82］中看到更详细的论述。这是一个非常严格的解释，而且执行时需要相对的精确。在通信网络中，就需要对诸如热线形式的虚假流量或者匿名路由作出严格的解释。

对大多数应用而言，这样做将导致过分强调严格的描述，其实通常情况下简单的形式化