第一章 数据安全概述

学习要求：了解相关数据安全相关的法律法规；掌握数据、数据安全的定义；掌握个人信息的处理规则；理解相关法律法规所定义的相关权益；了解分类分级的概念；理解数据安全相关的技术，能区分密文计算、机密计算、隐私计算等概念；了解数据安全相关典型案例，以及数据胶囊的含义。

课时：2 课时。

目前，隐私泄露的事件层出不穷，既有个人级的信息泄露，也有国家级的信息泄露。个人数据的泄露可能造成个人的财产损失乃至人身伤害（现在的电信诈骗多数是基于这种信息泄露实施的精准诈骗），而国家级的数据泄露更可能造成政治博弈和国与国之间的对抗。

习近平总书记很早就高瞻远瞩地提出了“没有网络安全，就没有国家安全”的国家战略。网络攻防和网络安全的实质是什么呢？可以用一句话来概述：保护服务和保护数据。服务指的是网络设施能够在任何正常和异常的情况下，保证基本功能的运行；而数据则是其中的核心，是重中之重。从第一次世界大战到第二次世界大战，从英格玛密码产生到图灵和波兰三杰的破译，保护的核心数据都是情报数据。随着信息科技的发展，数据变得越来越广泛，不只是战争时期机密的关键信息，个人数据的保护也变得越来越重要。

我们正处于一个智能时代，基于云计算、大数据等技术，从积累的海量数据里挖掘大量价值、改造和改善生活的时代。数据产生价值，海量的数据产生海量的价值。数据产生的价值的过程，其实是以用户信息的采集为前提的。如何保证公民的隐私数据不被泄露，又能满足日益增长的科技需求，这是一个大问题，是民生要事。2022 年，《总体国家安全观学习纲要》将国家安全的内涵进行了扩充，从 16 个领域扩展到 20 个领域，其中增加了数据安全的内容。强调：当前数据安全问题比较突出，绝不能掉以轻心。浩瀚的数据海洋就如同工业社会的石油资源，蕴含着巨大生产力和商机，谁掌握了大数据，谁就掌握了发展的资源和主动权。要把握好大数据发展的重要机遇，促进大数据产业健康发展，处理好数据安全、网络空间治理等方面的挑战。

因此，全世界都在出台法律法规，对网络安全和数据安全提出要求和约束。这些立法对数据安全和网络安全提出明确的要求和约束。现在的数据安全，已经从传统的容灾备份、数据恢复、去重存储、安全删除等传统问题，慢慢转向聚焦于怎么从数据中挖掘价值、在挖掘价值的同时怎么去保护用户的隐私等等，在这方面发现、形成、发展新的研究脉络。典型技术方法包括差分隐私、安全多方计算、同态加密、密文查询、密文机器学习等。

本节将从法律法规、技术体系、典型案例等方面概述当前数据安全的总体情况。

1.1 法规制度

1.1.1 相关法律法规

1. 国外相关法规

欧盟《通用数据保护条例》（GDPR）：该条例于 2018 年 5 月生效，是欧盟首部专门针对个人数据保护的法律，规定了个人数据处理的基本要求、责任和义务，以及个人数据保护、隐私保护等方面的内容，是全球遵守的规范参考。

美国《加州消费者隐私法案》（CCPA）：该法案于 2018 年通过，于 2020 年生效，是加州首部专门针对消费者隐私保护的法律，规定了消费者个人数据处理的基本要求、责任和义务，以及消费者隐私保护、数据泄露通知等方面的内容。

此外，还有《欧盟网络与信息安全指令》（NIS）、《美国联邦贸易委员会法案》、《澳大利亚信息与隐私权法案》等法律也涉及到数据安全方面的内容。

总体来说，全球各国都在加强数据安全法律法规的制定和实施，以保护个人隐私和数据安全，同时促进数字经济的发展。

2. 国内相关法规

《中华人民共和国网络安全法》：该法于 2016 年 11 月通过，于 2017 年 6 月施行，是中国首部针对网络安全领域的法律，规定了网络安全的基本要求、责任和义务，以及网络安全管理、网络安全技术、网络安全应急处置等方面的内容。

《中华人民共和国数据安全法》：该法于 2021 年 6 月通过，于 2021 年 9 月施行，是中国首部专门针对数据安全领域的法律，规定了数据安全的基本要求、责任和义务，以及数据安全管理、数据安全保护、数据安全监测预警等方面的内容。

《中华人民共和国个人信息保护法》：该法于 2021 年 8 月通过，2021 年 11 月施行，是中国为了保护个人信息权益，规范个人信息处理活动，促进个人信息合理利用，根据宪法，制定的法规。

此外，《中华人民共和国电子商务法》、《中华人民共和国刑法》等法律也涉及到数据安全方面的内容。

3. 数据安全法

共包含 7 章，从数据安全与发展、数据安全制度、数据安全保护义务、政务数据安全与开放、法律责任等方面给出明确规定。

（1）数据安全相关定义

《中华人民共和国数据安全法》第三条给出了数据、数据安全的官方定义，如下：

- ✧ **数据**，是指任何以电子或者其他方式对信息的记录。
- ✧ **数据处理**，包括数据的收集、存储、使用、加工、传输、提供、公开等。
- ✧ **数据安全**，是指通过采取必要措施，确保数据处于有效保护和合法利用的状态，以及具备保障持续安全状态的能力。

（2）数据安全与发展

《中华人民共和国数据安全法》鼓励数据安全与发展，在第十三条指出“国家统筹发展和安全，坚持以数据开发利用和产业发展促进数据安全，以数据安全保障数据开发利用和产业发展。”

同时，从数据安全技术研究、标准建设、检测评估、交易制度等方面给出规定：

- ✧ **第十六条** 国家支持数据开发利用和数据安全技术研究，鼓励数据开发利用和数据安全等领域的技术推广和商业创新，培育、发展数据开发利用和数据安全产品、产业体系。
- ✧ **第十七条** 国家推进数据开发利用技术和数据安全标准体系建设。国务院标准化行政主管部门和国务院有关部门根据各自的职责，组织制定并适时修订有关数据开发利用技术、产品和数据安全相关标准。国家支持企业、社会团体和教育、科研机构等参与标准制定。
- ✧ **第十八条** 国家促进数据安全检测评估、认证等服务的发展，支持数据安全检测评估、认证等专业机构依法开展服务活动。国家支持有关部门、行业组织、企业、教育和科研机构、有关专业机构等在数据安全风险评估、防范、处置等方面开展协作。
- ✧ **第十九条** 国家建立健全数据交易管理制度，规范数据交易行为，培育数据交易市场。

截止到 2023 年底，有关数据安全的检测评估尚未完全普及和展开，数据交易市场尚未形成规模，在技术保障、落地实施等方面还有很长的路要走。

（3）数据安全制度

数据安全法明确了 6 项数据安全制度：

- ✧ **数据分类分级与核心数据保护制度（第二十一条）**。确立了依据对国家安全、公共利益或者个人、组织合法权益造成的危害程度进行分类分级的原则，要求国家网信部门协调编制重要数据目录，各地区、各部门负责地方、领域的目录编制和

数据保护，特别强调对于关系国家安全、国民经济命脉、重要民生、重大公共利益等国家核心数据，实行更加严格的管理制度。

- ✧ **数据安全风险评估与工作协调机制（第二十二条）**。规定国家建立集中统一、高效权威的数据安全风险评估、报告、信息共享、监测预警机制，建立工作协调机制统筹协调有关部门加强数据安全风险信息的获取、分析、研判、预警工作。
- ✧ **数据安全应急处置机制（第二十三条）**。要求对于发生数据安全事件的，主管部门应当依法启动应急预案，采取相应的应急处置措施，防止危害扩大，消除安全隐患。
- ✧ **数据安全审查制度（第二十四条）**。要求对影响或者可能影响国家安全的数据处理活动进行国家安全审查。
- ✧ **数据出口管制制度（第二十五条）**。要求对与维护国家安全和利益、履行国际义务相关的属于管制物项的数据依法实施出口管制。
- ✧ **歧视反制制度（第二十六条）**。规定对我国采取相关歧视性的禁止、限制或者其他类似措施的国家和地区，我国可以对其采取对等措施。

目前，这些制度正处于推动落实的阶段，比如中央网信办推出《网络数据安全管理条例》等制度，逐步进行推动落实《数据安全法》相关的各项制度的建设。

其他方面不再详细介绍。

4. 个人信息保护法

《个人信息保护法》全文共计八章七十条，围绕个人信息的处理,从处理规则、跨境提供、个人权利处理者义务、保护职责部门以及法律责任等不同角度确立了相应规则，并且针对敏感个人信息和国家机关处理强调了特别规则。

（1）个人信息相关定义

《中华人民共和国个人信息保护法》第四条给出了个人信息、个人信息的处理的官方定义，如下：

- ✧ **个人信息**，是以电子或者其他方式记录的与已识别或者可识别的自然人有关的各种信息，不包括匿名化处理后的信息。
- ✧ **个人信息的处理**，包括个人信息的收集、存储、使用、加工、传输、提供、公开、删除等。

（2）个人信息的处理规则

以“告知-知情-同意”为核心的个人信息处理规则：

- ✧ 处理个人信息应当具有明确、合理的目的，并应当与处理目的直接相关，采取对个人权益影响最小的方式。收集个人信息，应当限于实现处理目的的最小范围。

- ✧ 处理个人信息应当在事先充分告知的前提下取得个人同意，个人信息处理的重要事项发生变更的应当重新向个人告知并取得同意。

（3）严格限制处理敏感个人信息

个人信息保护法规定，敏感个人信息包括生物识别、宗教信仰、特定身份、医疗健康、金融账户、行踪轨迹等信息，以及不满十四周岁未成年人的个人信息。

处理敏感个人信息比处理一般个人信息更为严格，只有在具有特定目的和充分的必要性，并采取严格保护措施的情形下，取得个人单独或书面同意才能进行。

（4）规范个人信息跨境处理

个人信息保护法规定，个人信息处理者因业务等需要，需向境外提供个人信息的：

- ✧ 应当通过国家网信部门组织的安全评估
- ✧ 按照国家网信部门的规定经专业机构进行个人信息保护认证
- ✧ 按照国家网信部门制定的标准合同与境外接收方订立合同，约定双方的权利和义务
- ✧ 符合法律、行政法规或者国家网信部门规定的其他条件
- ✧ 采取必要措施，保障境外接收方处理个人信息的活动达到本法规定的个人信息保护标准

（5）明确大型互联网平台个人信息保护义务

个人信息保护法对大型互联网平台设定特别的个人信息保护义务，包括：

- ✧ 按照国家规定建立健全个人信息保护合规制度体系，成立主要由外部成员组成的独立机构对个人信息保护情况进行监督
- ✧ 遵循公开、公平、公正的原则，制定平台规则
- ✧ 对严重违法处理个人信息的平台内产品或者服务提供者，停止提供服务
- ✧ 定期发布个人信息保护社会责任报告，接受社会监督

1.1.2 相关权益

1. 个人信息保护法

个人在个人信息处理活动中的七项权利：

（1）知情同意权

收集和使用公民个人信息必须遵循合法、正当、必要原则，且目的必须明确并经用户的知情同意；

（2）决定权

有权限制、拒绝或撤回他人对其个人信息的处理；

（3）查阅复制权

个人有权向个人信息处理者查阅、复制其个人信息；

（4）个人信息移转权

个人请求将个人信息转移至其指定的个人信息处理者，符合国家网信部门规定条件的，个人信息处理者应当提供转移的途径；

（5）更正补充权

个人发现其个人信息不准确或者不完整的，有权请求个人信息处理者更正、补充；

（6）删除权

在五种情形下，个人信息处理者应当主动删除个人信息；个人信息处理者未删除的，个人有权请求删除：

- ✧ 处理目的已实现、无法实现或者为实现处理目的不再必要，
- ✧ 个人信息处理者停止提供产品或者服务，或者保存期限已届满，
- ✧ 个人撤回同意，
- ✧ 个人信息处理者违反法律、行政法规或者违反约定处理个人信息，
- ✧ 法律、行政法规规定的其他情形；

（7）规则解释权

个人有权要求个人信息处理者对其个人信息处理规则进行解释说明。

2. 欧盟 GDPR

欧盟 GDPR 定义了多种用户权益。

（1）Transparency & Auditing 透明&审计

数据主体应当知道谁拥有他们的数据&数据是如何处理的。（第 13-14 章）

等同于个人信息保护中定义的知情同意权

（2）Consent 同意

数据主体应该明确同意收集、处理他们的数据。（第 4、7 章）

等同于个人信息保护中定义的知情同意权。

（3）Processing Control 处理控制

数据主体应该有权控制对他们的数据应用的处理类型。（第 18、22 章）

等同于个人信息保护中定义的决定权。

（4）Data Portability 数据携带权

数据主体应该能够获得与他们相关的任何数据的副本。（第 15-17、20 章）

等同于个人信息保护中定义的查阅复制权。

（5）其他权益

GDPR 还定义了删除权、被遗忘权等多种权益。删除权与被遗忘权是有区别的，简单来说，传统的删除权是一对一的，即用户个人（数据主体）对于企业（数据控制者）提出的要求，当数据主体认为数据控制者违法或违约收集、使用其个人信息的情况下，有权要求企业删除其个人数据。而 GDPR 在传统删除权上进一步扩展提出的“被遗忘权”是一对多的，不仅包含传统的删除权的权利要求，还包括要求数据控制者负责将其已经扩散出去的个人数据，采取必要的措施予以消除。

1.1.3 落地实施

1. 数据要素是第一生产力

国家互联网信息办公室在今年 5 月发布了《数字中国发展报告（2022 年）》，报告中显示我国去年的数据产量达 8.1ZB，同比增长 22.7%，全球占比 10.5%，位居世界第二。在 2020 年 4 月中共中央、国务院发布了《关于构建更加完善的要素市场化配置体制机制的意见》，“数据”作为一种新的生产要素首次写入了中央文件中。我国成为全球第一个（在国家政策层面）将数据确立为生产要素的国家。2022 年 12 月，《中共中央 国务院关于构建数据基础制度更好发挥数据要素作用的意见》进一步指出，加快构建新发展格局，坚持改革创新、系统谋划，以维护国家数据安全、保护个人信息和商业秘密为前提，以促进数据合规高效流通使用、赋能实体经济为主线，以数据产权、流通交易、收益分配、安全治理为重点，深入参与国际高标准数字规则制定，构建适应数据特征、符合数字经济发展规律、保障国家数据安全、彰显创新引领的数据基础制度，充分实现数据要素价值、促进全体人民共享数字经济发展红利，为深化创新驱动、推动高质量发展、推进国家治理体系和治理能力现代化提供有力支撑。

2023 年 3 月，中共中央、国务院印发了《数字中国建设整体布局规划》，将数字安全屏障、数字技术创新体系并列为强化数字中国的两大能力。

2. 网络数据安全管理条例

2021 年 11 月 14 日，国家网信办发布了《网络安全数据管理条例》（征求意见稿）（以下简称《条例》），共九章七十五条。 2022 年 7 月，国务院办公厅印发《国务院 2022 年度立法工作计划》，《网络数据安全管理条例》被列入拟制定、修订的行政法规中。

概括来说，《条例》是在《中华人民共和国网络安全法》《中华人民共和国数据安全法》《中华人民共和国个人信息保护法》三部上位法的基础上制定，在实施细则、责任界定、规范要求、惩罚措施等方面更加清晰细致，同时也增加了一些新的内容，进一步强化和落实数据处理者的主体责任，共同保护重要数据和个人信息的安全。

（1）法律依据

《条例》的制定是以《中华人民共和国网络安全法》《中华人民共和国数据安全法》《中华人民共和国个人信息保护法》三部上位法为依据。

（2）主体定义

数据处理者：在数据处理活动中自主决定处理目的和方式的个人和组织。

互联网平台运营者：为用户提供信息发布、社交、交易、支付、视听等互联网平台服务的数据处理者。

大型互联网平台运营者：用户超过五千万、处理大量个人信息和重要数据、具有强大社会动员能力和市场支配地位的互联网平台运营者。

（3）明确数据分类分级标准

《条例》第五条指出，国家需建立数据分类分级保护制度。

数据分类是按照数据具有的某种共同属性或特征（包括数据对象共享属性、开放属性、应用场景等），采用一定原则和方法进行区分和归类，以便于管理和使用数据。

数据分级目的是差异化保护数据安全，按照数据遭到破坏（包括攻击、泄露、篡改、非法使用等）后对国家安全、社会稳定、公共利益以及个人、法人和其他组织的合法权益（受侵害客体）的危害程度对数据进行定级，为数据全生命周期管理的安全策略制定提供支撑。

按照数据对国家安全、公共利益或者个人、组织合法权益的影响和重要程度，将数据分为一般数据、重要数据、核心数据，不同级别的数据应采取不同的保护措施：

- ✧ **核心数据：**对领域、群体、区域具有较高覆盖度或达到较高精度、较大规模、一定深度的重要数据，一旦被非法使用或共享，可能直接影响政治安全。主要包括关系国家安全重点领域的数据，关系国民经济命脉、重要民生、重大公共利益的数据，经国家有关部门评估确定的其他数据。
- ✧ **重要数据：**特定领域、特定群体、特定区域或达到一定精度和规模的数据，一旦被泄露或篡改、损毁，可能直接危害国家安全、经济运行、社会稳定、公共健康和安全等的数据。仅影响组织自身或公民个体的数据，一般不作为重要数据。
- ✧ **一般数据：**核心数据、重要数据之外的其他数据。

数据分类分级是企业数据安全治理的基础环节，也是国家、政府平衡数据保护与数据流通的重要手段，通过对敏感数据的分级，提升数据的安全性，降低企业的合规性风险。任何时候，数据的定级都离不开数据的分类。因此，在数据安全治理或数据资产管理领域都是将数据的分类和分级放在一起，统称为数据分类分级。

目前，诸如金融、工业、电信、医疗和汽车等行业均已出台了针对性的数据分类分级指南或技术规范，天津市自贸港也于 2024 年 2 月率先推出了涉及数据出境相关的重要数据分类指南。

（4）其他相关规定

《条例》还从多个方面给出规定，比如规定重要数据需落实等保等。《条例》第九条指出，数据处理者应当按照网络安全等级保护的要求，加强数据处理系统、数据传输网络、数据存储环境等安全防护，处理重要数据的系统原则上应当满足三级以上网络安全等级保护和关键信息基础设施安全保护要求，处理核心数据的系统依照有关规定从严保护。

总体上，目前技术保障、落地实施等方面，距离实现数据全生命周期的安全防护、全面落实《数据安全法》和《个人信息保护法》还有很长的路要走。

1.2 技术体系

1.2.1 技术体系

数据安全的定义强调“确保数据处于有效保护和合法利用的状态”，数据处理（收集、存储、使用、加工、传输、提供、公开等）中的持续安全状态能力的保障都可以纳入数据安全的学术研究范畴。现在的数据安全，已经从传统的容灾备份、数据恢复、去重存储、安全删除等传统问题，慢慢转向聚焦于怎么从数据中挖掘价值、在挖掘价值的同时怎么去保护用户的隐私等等，在这方面发现、形成、发展新的研究脉络。

如图 1.2.1 所示，以密码学和数据科学为基础，以区块链和访问行为模式保护（第九章）为共性的信任支撑和安全保障技术，围绕数据全生命周期安全防护，衍生出一系列典型技术方法，包括差分隐私（第六章）、安全多方计算（第十章）、同态加密（第三章）、密态数据库（第七章）等，这些技术在后续章节中进行详细讲解。

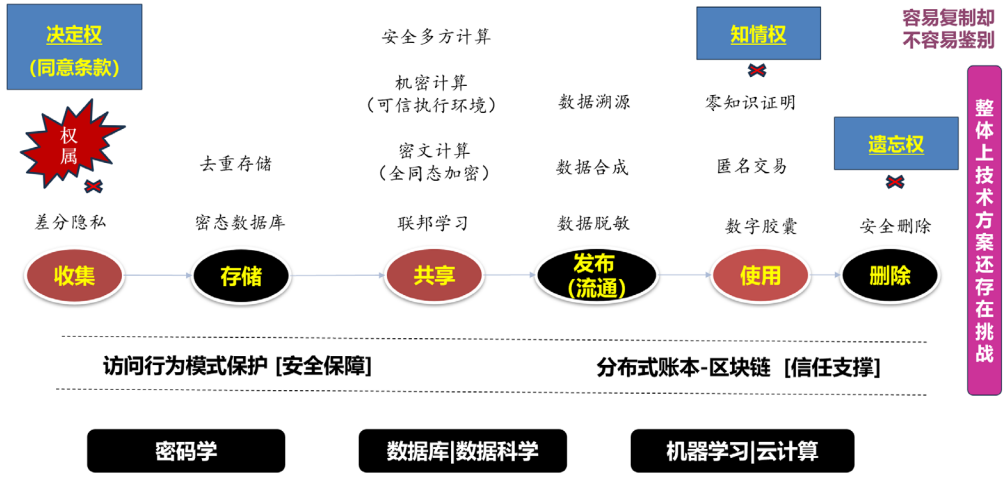


图 1.2.1 数据安全技术体系

现有技术解决了数据安全防护的特定问题。但是，因为电子数据容易复制、却不容易鉴别，在实现数据全生命周期安全防护、落实《个人信息保护法》和欧盟 GDPR 中定义的多种用户权益方面，还存在很多技术空白，比如，还没有可靠的技术支撑遗忘权、知情权等的实现。

1.2.2 基础技术

1. 密码技术

密码技术的基本思想是伪装信息，所谓伪装就是对数据进行一组可逆的数学变换。伪装前的原始数据称为明文（Plaintext），而伪装后的消息称为密文（Ciphertext）。伪装的数据变换过程称为加密（Encryption），相反的，将密文还原成明文的过程称为解密（Decryption）。

加密和解密算法的操作通常都是在一组密钥（Key）的控制下进行的，分别称为加密密钥和解密密钥，加密密钥通常用 K_e 来表示，解密密钥通常用 K_d 来表示。

（1）密码系统

一个密码系统（即密码体制），通常由五部分组成：

消息空间：所有可能明文 m 的有限集称为明文空间，通常用 M 表示；

密文空间：所有可能密文 c 的有限集称为密文空间，通常用 C 表示；

密钥空间：一切可能的密钥 k 构成的有限集称为密钥空间，通常用 K 表示；

加密算法：加密算法是基于加密密钥 k_e 将明文 m 变换为密文 c 的变换函数，相应的变换过程称为加密，通常用 E 表示，即$c = E_{k_e}(m)$，也可表示为 $c=E(m, k_e)$。

解密算法：解密算法是基于解密密钥 k_d 将密文 c 恢复为明文 m 的变换函数，相应的变换过程称为解密，通常用 D 表示，即$m = D_{k_d}(c)$，也可表示为 $m=D(c, k_d)$。

（2）分类

根据密钥的数量，可以将密码体制分为两类：对称密钥密码体制和非对称密钥密码体制。

如果一个密码体制的 $k_d = k_e$，即发送方和接收方双方使用的密钥相同，则称为对称密钥密码体制；相反地，如果 $k_d \neq k_e$，即发送方和接收方双方使用的密钥不相同，则称为非对称密钥密码体制。

在非对称密钥体制中，由于在计算上 k_e 不能由 k_e 推出，这样将 k_e 公开也不会损害 k_d 的安全，于是可以将 k_e 公开，因此，这种密码体制又被称为公开密钥密码体制。公开密钥密码体制的概念，由 Diffie 和 Hellman 于 1976 年提出，它的出现是密码史上的一个里程碑。

2. 哈希函数

哈希（hash）函数又称为散列函数、杂凑函数，它是一种单向密码体制，即它是一个从明文到密文的不可逆映射，只有加密过程，没有解密过程。

哈希函数可以将任意长度的输入经过变换后得到固定长度的输出。哈希函数的数学表述为：$h = H(m)$，其中 $H()$是哈希函数，m 是任意长度明文，h 是固定长度的哈希值。

理想的哈希函数对于不同的输入，获得的哈希值不同。如果存在 x、x'两个不同的消息，存在 $H(x)= H(x')$，则称 x 和 x'是哈希函数 H 的一个碰撞。

哈希函数的这种单向的特性以及长度固定的特征使得它可以生成消息或者数据块的“消息摘要”（也称为散列值、哈希值），因此，在数据完整性和数字签名领域有着广泛的应用。

典型的哈希函数有：消息摘要算法 MD5（Message Digest Algorithm）和安全散列算法 SHA（Secure Hash Algorithm）。

1．哈希函数的性质

哈希算法有一些特点：

① 压缩。对于任意大小的输入 x，哈希值 $H(x)$的长度很小，实际应用中哈希函数 H 产生的哈希值是固定长度的。

② 易计算。对于任意给定的消息，容易计算其哈希值。

③ 单向性。对于给定的哈希值 h，要找到 m'使得 $H(m') = h$ 在计算上是不可行的，即求哈希函数的逆很困难。

④ 抗碰撞性。理想的哈希函数是无碰撞的，但实际算法设计中很难做到。因此有两种抗碰撞性：一种是弱抗碰撞性，即对于给定的消息 x，要发现另一个消息 y，满足 $H(x)= H(y)$在计算上不可行；一种是强抗碰撞性，即对任意一对不同的消息(x, y)，使得 $H(x)= H(y)$在计算上不可行。

⑤ 高灵敏性。当一个输入位发生变化时，输出位将有一半以上会发生变化。

2．哈希函数的应用

消息认证：在一个开放通信网络的环境中，信息面临的攻击包括窃听、伪造、修改、插入、删除、否认等。因此，需要提供用来验证消息完整性的一种机制或服务，即消息认证。这种服务的主要功能包括确保收到的消息确实和发送的一样、确保消息的来源真实有效等，用于消息认证的最常见的密码技术是基于哈希函数的消息认证码。

数字签名：由于非对称算法的运算速度较慢，所以在数字签名协议中，哈希函数扮演了一个重要的角色。对哈希，又称“消息摘要”进行数字签名，在统计上可以认为与对文件本身进行数字签名是等效的。

口令的安全性：由于哈希函数具有单向性的特征，在口令保护中应用非常广泛。通常，口令仅仅将其哈希值进行保存，进行口令校验的时候比对哈希值即可，即使攻击者获得了保存的哈希值，也无法计算出口令。

数据完整性：比较熟悉的校验算法有奇偶校验和循环冗余校验码校验，这两种校验并没有抗数据篡改的能力，它们一定程度上能检测并纠正数据传输中的信道误码，但却不能防止对数据的恶意破坏。哈希算法 “消息摘要”的特性，使它成为目前应用最广泛的一种数据完整性校验和算法。

3. 区块链

区块链（英文名：blockchain 或 block chain）是一种块链式存储、不可篡改、安全可信的去中心化分布式账本，它结合了分布式存储、点对点传输、共识机制、密码学等技术，通过不断增长的数据块链记录交易和信息，确保数据的安全和透明性。

区块链起源于比特币（Bitcoin），最初由中本聪（Satoshi Nakamoto）在 2008 年提出，作为比特币的底层技术。从诞生初期的比特币网络开始，区块链逐渐演化为一项全球性技术，以太坊（Ethereum）等新一代区块链平台的出现进一步扩展了应用领域。

区块链包括三个基本要素，即交易（Transaction，一次操作，导致账本状态的一次改变）、区块（Block，记录一段时间内发生的交易和状态结果，是对当前账本状态的一次共识）和链（Chain，由一个个区块按照发生顺序串联而成，是整个状态变化的日志记录）。区块链中每个区块保存规定时间段内的数据记录（即交易），并通过密码学的方式构建一条安全可信的链条，形成一个不可篡改、全员共有的分布式账本。通俗地说，区块链是一个收录所有历史交易的账本，不同节点之间各持一份，节点间通过共识算法确保所有人的账本最终趋于一致。区块链中的每一个区块就是账本的每一页，记录了一个批次记录下来的交易条目。这样一来，所有交易的细节都被记录在一个任何节点都可以看得到的公开账本上，如果想要修改一个已经记录的交易，需要所有持有账本的节点同时修改。同时，由于区块链账本里面的每一页都记录了上一页的一个摘要信息，如果修改了某一页的账本（也就是篡改了某一个区块），其摘要就会跟下一页上记录的摘要不匹配，这时候就要连带修改下一页的内容，这就进一步导致了下一页的摘要与下下页的记录不匹配。如此循环，一个交易的篡改会导致后续所有区块摘要的修改，考虑到还要让所有人承认这些改变，这将是一个工作量巨大到近乎不可能完成的工作。正是从这个角度看，区块链具有不可篡改的特性。

区块链的特点包括去中心化、不可篡改、透明、安全和可编程性。每个数据块都链接到前一个块，形成连续的链，保障了交易历史的完整性。智能合约技术使区块链可编程，支持更广泛的应用。

1.2.3 典型技术

（1）可信计算

可信计算（Trusted Computing），是一种基于密码的运算与防护并存的计算机体系安全技术，保证全程可检测可监控。可信计算核心部分是可信根，通常是可信硬件芯片。

可信并不等同于安全，但它是安全的基础，因为安全方案、策略只有运行在未被篡改的环境下才能进一步确保安全目的。通过保证系统和应用的完整性，可以确保使用正确的

软件栈，并在软件栈受到攻击发生改变后能及时发现。总的来说，在系统和应用中加入可信验证能够减少由于使用未知或遭到篡改的系统/软件遭到攻击的可能性。

以 PC 机可信举例，通俗来讲，可信就是在每台 PC 机启动时检测 BIOS 和操作系统的完整性和正确性，保障你在使用 PC 时硬件配置和操作系统没有被篡改过，所有系统的安全措施和设置都不会被绕过；在启动后，对所有的应用，如社交软件、音乐软件、视频软件等应用可进行实时监控，若发现应用被篡改立即采取止损措施。

（2）机密计算

机密计算（Confidential Computing）是通过基于硬件的可信执行环境（TEE，Trusted Execution Environment）对使用中的数据进行保护。

可信执行环境 TEE 被定义为提供一定级别的数据完整性、数据机密性和代码完整性保证的环境，比如 Intel SGX、ARM Trustzone 等。

在受信任的硬件执行环境基础上构建安全区域，所有参与方将需要参与运算的明文数据加密传输至该安全区域内并完成运算，安全区域外部的任何非授权的用户和代码都无法获取或者篡改安全区域内的任何数据。机密计算可在可信硬件执行环境内保护数据的隐私，但当数据离开可信硬件执行环境时无能为力。

（3）密文计算

密文计算是指所有参与运算的明文数据使用指定的规则转换为密文，在密文空间中进行特定形式的代数运算并得到结果，密文运算的结果再通过相应的转换规则转换为明文运算结果，该结果与明文运算结果一致。

同态加密是密文计算代表性技术。

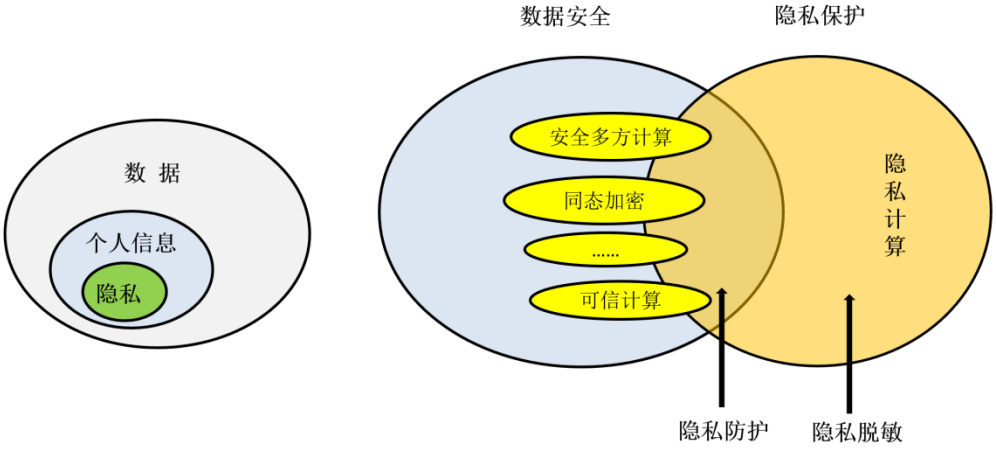
（4）安全多方计算

安全多方计算能够同时确保输入的隐私性和计算的正确性，在无可信第三方的前提下通过数学理论保证参与计算的各方成员输入信息不暴露，且同时能够获得准确的运算结果。

（5）隐私计算

中国中文信息学会大数据安全与隐私计算专业委员会李凤华主任等在《隐私计算：概念、内涵和实例》中给出隐私计算的准确定义，即：隐私计算是面向隐私信息全生命周期保护的计算理论和方法，是隐私信息的所有权、管理权和使用权分离时隐私度量、隐私泄露代价、隐私保护与隐私分析复杂性的可计算模型与公理化系统。具体是指在处理视频、音频、图像、图形、文字、数值、泛在网络行为信息流等信息时，对所涉及的隐私信息进行描述、度量、评价和融合等操作，形成一套符号化、公式化且具有量化评价标准的隐私计算理论、算法及应用技术，支持多系统融合的隐私信息保护。隐私计算涵盖了信息搜集者、发布者和使用者在信息产生、感知、发布、传播、存储、处理、使用、销毁等全生命周期过程的所有计算操作，并包含支持海量用户、高并发、高效能隐私保护的系统设计理论与架构。隐私计算是泛在互联环境下隐私信息保护的重要理论基础。

（6）小结



个人信息是数据的一种，而隐私数据属于个人信息敏感的非授权不能访问的数据，这种层次关系很清晰。数据安全是指保护数据全生命周期（包含产生、存储、销毁等）数据合法使用和有效保护的能力，它涉及的技术领域广泛。同态加密、安全多方计算等技术都是实现数据安全的相关技术，他们也是达到隐私保护目标的相关技术。而仅仅就隐私计算而言，很多人将其与安全多方计算、密文计算等等同，这个是有明显区分的，主要体现在信息量损失上，后者明显是不允许有任何信息损失的，但隐私计算允许。

1.3 典型案例

1.3.1 明文发布

在未来很长一段时间或者很多特殊应用场景里，仍然需要以明文方式使用和共享数据。如何让这些明文的数据能产生价值，同时避免泄露用户隐私？

（1）典型案例

一个典型例子就是精准医疗，即利用海量医院病人的数据，提高医疗诊治的准确率。医院已经积累了海量的病人数据，但实际上，在医疗机构要用病人数据进行分析以提升诊断准确率时，医院不敢分享这些数据，因为在分享的过程中，一定会存在泄露病人隐私的风险。

（2）相关技术

隐私保护的数据发布就是针对此的一类解决方法。像医院、大型互联网企业、电信企业这样的数据发布者，都有海量的用户信息和各种维度的数据，但他们是不敢轻易地把数据发布给外部单位的，因为在发布的过程会泄露用户敏感的隐私。一种直观的做法是删掉敏感信息再发布，例如删掉病人信息中的名字、身份证号等，而保留统计分析可能需要的信息，例如地域、住址、年龄等信息。

这种做法一般来说是没有问题的，但是对于专业的信息安全攻防人员而言，这就是大问题，为什么？举例来说，如果某大学医院的这种数据泄露了，而攻击者可以通过一些渠道购买或者通过黑客手段拿到某大学的数据库，得到该大学的具体老师名单。仅仅把名字等敏感信息去掉是不够的，因为通过准标识符，仍然是可以确定用户身份的。例如，一个计算机学院的、41 岁的男老师得了某种病，而计算机学院也没几个 41 岁的老师，那么就可能很容易地圈定到具体人员。这种<单位，年龄>类似的信息就是准标识符。

为了抵御类似的攻击，进一步的匿名方法提出来，比如泛化—将所有属性变成上一级的、更广泛的属性。例如， 把计算机学院泛化成工学院，40~45 岁的统一变成 45 岁。一个准标识符泛化后可能关联 K 条数据的话，就叫做 K 匿名。通过这样的方法，做数据分析的时候，虽然不那么精确，但一定程度上还能满足需求。但是这仍然存在问题。如果相同泛化的结果（即 K 个准标识符）只能对应一种疾病，那么攻击者也能猜出你得的是什么病。这种情况下，就需要 L 样化。L 样化是指相同泛化后的准标识符拥有 L 种疾病。此时，攻击者只能知道你有 L 种疾病可能，很难精确推测你的疾病了。

K 匿名和 L 样化是不是就安全了呢？也不一定，因为还可能存在表关联攻击等。例如，一共有 5 个人，数据中有 4 个满足条件的，那么我就知道至少有 80%概率生病。如果数据允许被查询的话，攻击者甚至可以用差分攻击的方法去差分 1 个人在不在这 4 个里，由此就可以确定地得知某个人得没得病。这种差分攻击非常致命，SQL 盲注实际利用的就是类似的方法，通过一个查询语句成功执行与否的判定，去猜测数据库的表名、字段甚至各类数据。

抵御差分攻击的方法就是差分隐私。差分隐私是通过增加特定分布的噪音保护个体的差异，使得整体数据特征不变的一种技术，在现阶段非常有用。它可以用在面向终端用户的数据采集，比如输入法里，通过增加噪音后无法让服务器知道特定用户的隐私信息，但可以利用带噪音的数据去推测，类似用户输入了“LZL”就会选择“刘哲理”这一种词语的行为习惯。因为，一旦“刘哲理”成为了网红，一些用户可能会大量的搜索，这个时候可以利用采集的部分用户数据产生的价值，服务于其它第一次输入“LZL”的用户。

差分隐私就是在查询结果上加噪音，让构造的相似的数据集的返回结果变的不确定，让差分攻击失去了本质依赖条件，即对于个体“是与不是”或者“在与不在”的回答不确定了，所以无法奏效。然而，差分隐私并非是万能药，因为加的噪音都是围绕真值的特定分布的数据，如果能多次查询，取平均值就能得到真正结果，这种重复攻击就又奏效了。

1.3.2 半密文使用

数据是机器学习的基础。而在大多数行业中，由于行业竞争、隐私安全、行政手续复杂等问题，数据常常是以孤岛的形式存在的。一个企业，它有上海分中心、天津分中心、北京分中心，内部都有大量数据，明文存储的数据并不一定能够很轻易地共享出去。不同的企业，例如腾讯、京东、阿里等，他们既存在竞争关系，又存在合作关系，他们都有大

量的数据库，在很多时候需要使数据联合起来产生价值。这种情况就要求明文存储、密文使用。

（1）典型案例

一个典型案例就是精准广告推荐。现在，广告主投放广告的要求越来越高，并不只是要求覆盖一个用户范围就行了，而是要求精准覆盖。以游戏设备厂商为例，腾讯有用户玩游戏的记录，而京东有用户买游戏设备的记录，游戏设备商就希望把广告投放给既玩某一款游戏、又买某一款设备的用户，这样广告的转化率才更高，广告主才愿意花更多地钱。

（2）相关技术

那么对两个数据孤岛，怎样去求交集，怎样把共性的元素提炼出来，这些问题都是安全业务的扩展带来的新问题。数据安全是企业拓展业务需要解决的首要问题。刚才所说的问题怎么来解决呢？用安全多方计算去解决。我们不去共享数据，但是我们通过密文上的数据计算的方式，来把任务完成。刚才说的两方精准推送广告，其实就是密文集合求交，是安全多方计算的典型例子。

我们再看第二类场景。银行有着大量的财务数据，他们也想利用这些数据来产生价值。一种做法是开放一个接口，如果有需要可以来联合做预测。例如超市有很多会员，它想增加营业额，充分利用会员机制，把一些大额的商品推送给会员。但如果会员平时就不怎么有钱，不怎么买东西，推送反而可能引起反感。所以超市就有动机宁可稍微花点钱，和银行做一次联合，以确定是否把大额商品推送给某会员。这种联合虽然要花小部分钱，但能提升超市的信誉，也能提升它的业绩。这是一种纵向联邦机器学习，也就是两方拥有不同维度的用户数据，超市有会员信息和购买意向，而银行有会员用户的财务情况。

另一种情况是，需要利用分布在不同地点的相同维度的数据，例如上海、北京、天津分中心的大量数据，去共同训练一个模型。可是把数据集中在一起再去训练机器学习模型，对数据存储带来很高要求，而且训练效率可能也不够高。而且考虑到安全性，有些数据根本就不允许分享出去。在这种情况下，使用不同孤岛里存储的相同维度数据共同训练一个任务模型，就是横向联邦机器学习，这个过程是基于密文参数交换共享、特定服务器安全聚合的方式来完成的。安全聚合的联邦机器学习已经成为信息孤岛数据价值挖掘、保护用户隐私的重要数据安全手段。

1.3.3 全密文计算

从明文存储、明文使用到明文存储、密文使用，最终我们希望存的就是密文，用的时候也是密文，这将是一个理想状态。现在在一些机密的环境里已经开始使用了，例如密态数据库。大量的信息泄露的根本原因，就是数据库里面存的是明文，但如果要把数据加了密存入数据库，那数据可能就没法用了，增、删、改、查可能都没有办法执行了。

密文查询和密文计算有助于解决这个问题，包括可搜索加密、保留顺序加密和同态加密等。可搜索加密解决的其实是关键词检索的问题。在数据库里输入一个订单号查找一条

记录，或者输入一本书名查找类似书籍信息的多条记录，都是关键词检索的问题。在 Windows 系统打开“我的电脑”，去找一个文件存在哪儿，这也是关键词检索的问题。打开邮箱，输入一个关键词，搜索过去的某一封邮件，同样是一个关键词检索的问题。这些信息存在文件或者数据库里，如果加了密，那么可搜索加密就可以替你完成这些事情，即检索密文里是否包含一个加密的关键词，这种手段是用密码技术来解决用户隐私保护的问题。数据库里经常要做范围查询，大于或小于多少、在哪个区间的范围查询。范围查询就需要在密文上保留顺序，这就是保留顺序加密能做到的事情。同态加密可以用于统计分析、求和与平均等常见数据库统计任务。上述的这些密码机制，在 NoSQL 数据库、分布式文件存储系统、云存储里都是可以工作的。

纯密文的状态是未来的必然趋势，现在核心关键的地方也应该率先采用这样的技术。随着《中华人民共和国密码法》的颁布，基于密码技术来实现数据价值的共享、用户隐私的保护，是一种必然趋势。然而，密码应用现在面临很多挑战，比如可搜索加密面临注入攻击，同态加密面临密文扩充、复杂计算效率太低等问题；再比如，现在国产数据库还不是全部基于密码、还做不到全密态，它们很多都是用加密卡来做的，只是在硬盘存储的时候加密。要完全解决这些问题，还需要很长的路。

1.4 数据胶囊

什么是数据胶囊？数据胶囊是一种保护隐私的数据交换模型，包含用户要共享的加密数据+与之相对应的访问策略。

2022 年，Dawn Song 等在 USENIX Security 发表论文《PrivGuard: Privacy Regulation Compliance Made Easier》，提出了数据胶囊管理器 PrivGuard。

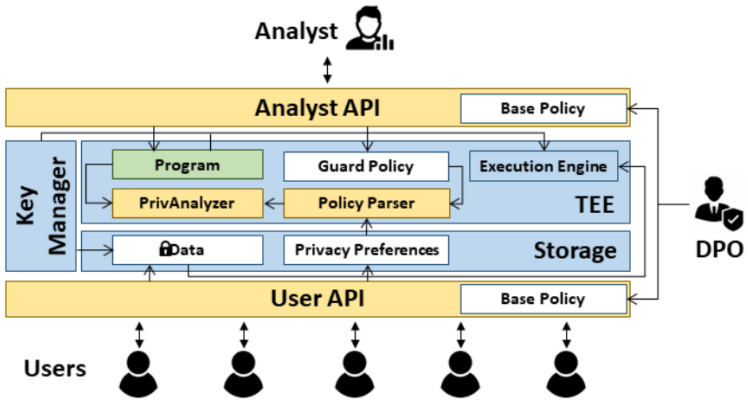


Figure 2: PRIVGUARD prototype infrastructure. White: data/policies; green: analysis programs; blue: off-the-shelf components; yellow: newly-designed components.

数据保护官员(DPOs)、法律专家和领域专家合作将隐私法规（base policy）转化为机器可读的策略语言。翻译过程需要在应用程序和隐私监管中的特定领域的知识（例如，将法律概念映射到具体的领域）。

基于 PrivGuard 的数据应用步骤：

- ✧ 在收集数据之前，数据主体将由客户端 API 进行辅助，以指定他们的隐私偏好。数据分析人员提交程序来分析收集到的数据&提交一个相应的不弱于基本政策的保护策略（Guard Policy）。
- ✧ 使用静态分析工具 PrivAnalyzer 检查分析程序，以确认其符合保护策略（Guard Policy）。同时，加载隐私偏好不大于保护政策的数据子集，进行实际分析。
- ✧ 根据 PrivGuard 的输出，结果解密给分析师或继续被保护。

举例：

DPO、法律专家和领域专家在基本政策中规定了两项要求：（1）未成年人数据不得用于任何分析；（2） 应使用差分隐私保护数据的任何统计数据。

隐私偏好与数据一起从数据主体收集。一些数据主体（第 1 组）信任公司并直接接受基本策略。一些人（第 2 组）更为谨慎，希望在分析之前对其邮政编码进行修改，其他人（第 3 组）不信任该公司，不希望其数据用于合法利益以外的目的。

数据分析师希望调查用户年龄分布。它规定了一项保护策略，即除了基本策略外，在分析中也不得使用邮政编码。分析师提交一个计算用户年龄直方图的程序给 PRIVGUARD。他记得过滤掉所有次要信息并编辑邮政编码字段，但忘记了用差分隐私保护程序。

PRIVGUARD 使用 PRIVANALYZER 检查隐私偏好，并将第 1 组和第 2 组的数据加载到 TEE 中，因为他们的隐私偏好并不比保护策略严格。PRIVGUARD 运行程序并保存生成的直方图。但是，经过检查 PRIVGUARD 发现程序没有用差分隐私保护直方图。因此，直方图被加密，转储到存储层，并由残差策略进行保护，该策略指示应在结果解密之前应用差分隐私。

PRIVGUARD 将剩余策略输出给分析员。分析人员在检查剩余策略后，提交了一个程序，该程序将噪声添加到直方图中，以满足不同的隐私要求。PRIVGUARD 然后解密直方图，将其加载到 TEE 中，并执行程序为其添加噪声。这一次，PRIVGUARD 发现保护策略中的所有要求都得到了满足，因此它将直方图解密给分析员。

第二章 密码基础

学习要求：掌握 P 问题和 NP 问题；理解对称密码的设计思想，掌握对称密码的分类和工作模式；掌握 NP 问题及其公钥密码体制的设计思想，理解公钥密码及其在数字签名中的应用，了解典型的公钥密码构造方案；掌握刻画敌手能力的四种攻击模型，理解不可区分性的安全性目标，了解安全性证明的过程和思想、常见的安全性模型；了解通用可组合安全的概念。

课时：4 课时。

建议授课进度：[2.1~~2.3]、[2.4~2.5]

2.1 基本概念

2.1.1 算法复杂度

算法复杂度包括时间复杂度和空间复杂度，分别由运行此算法所需要的计算时间 T 或者计算空间 S 来表示，这两个值往往表示为算法输入规模$\boldsymbol{n}$的函数。在分析并确定一个算法复杂度时，通常用大 O 符号 O(.)来表示。

在算法分析的应用中，通常采用时间复杂度来描述算法复杂度。从时间复杂度的角度来看，算法一般可分为三类：多项式时间算法、亚指数时间算法和指数时间算法，算法时间复杂度依次增高。

定义（多项式时间算法）：假设$\boldsymbol{n}$表示一个算法输入量的规模，k为某个常数，执行该算法的时间复杂度为$O(\boldsymbol{n}^k)$，称该算法为一个多项式时间算法。

在计算复杂度理论中，多项式时间算法被看成是一个简单的算法。如果解决一个问题的算法是多项式时间的，则该问题不是一个计算困难问题。

定义（指数时间算法）：指数时间算法是指该算法的时间复杂度为$O(t^{f(\boldsymbol{n})})$，这里t是一个大于 1 的常数，$f(\boldsymbol{n})$是关于算法输入规模$\boldsymbol{n}$的一个多项式函数。

在计算复杂度理论中，指数时间算法具有过高的时间算法复杂度，所该算法不是一个有效算法。对于一个计算问题来说，如果解决该问题的算法是指数时间算法，则该问题是一个计算困难问题。

定义（亚指数时间算法）：亚指数时间算法是指该算法的时间复杂度为$O(t^{f(\boldsymbol{n})})$，这里t是一个大于 1 的常数，$f(\boldsymbol{n})$是大于常数小于$\boldsymbol{n}$的线性多项式的一个函数。

亚指数时间算法的复杂度介于多项式时间算法和指数时间算法之间。

2.1.2 P 问题和 NP 问题

P 是 Polynomial-time，即多项式时间；NP 是指 Non-deterministic Polynomial-time，即非确定性多项式时间。P 类问题是易解问题，NP 类问题是难解问题。

P 问题：用确定性算法可以在多项式时间内求解的问题。

例如：冒泡排序、快速排序等问题。

NP 问题：用非确定算法可以在多项式时间内求解的问题。

举一个非常浅显的例子来说明 NP 问题和 P 问题。如果让别人将非常多的不规则碎片拼成一个完整的杯子，这个问题的解决方式是随机的，且解决起来非常困难，是一个 NP 问题；而 P 类问题则是说让别人去数杯子碎片有多少个，这种问题是比较容易解决的。

NP 问题不一定在多项式时间内可解，但可以在多项式时间内验证。比如上面的碎片拼杯子的问题，虽然求解过程可能算法很复杂，但是结果是一个完整的杯子，很容易验证是由给定的碎片拼成的。

换句话说，计算机可以在多项式时间复杂度内解决的问题称为 P 类问题，在多项式时间复杂度内不可以解决的问题称为 NP 类问题。

2.2 对称密码

对称密钥密码体制基本特征是发送方和接收方共享相同的密钥，即加密密钥与解密密钥相同。对称密码的优点在于加解密处理速度快。

2.2.1 对称密码分类

对称密钥体制，根据对明文的加密方式的不同而分为两类：分组密码（Block cipher）和序列密码（Stream cipher）。

分组密码，也被称为分块密码，是将明文消息编码表示后的位（Bit）或字符序列，按指定的分组长度划分成多个分组（Block）后，每个分组分别在密钥的控制下变换成等长的输出。常见的分组密码算法主要有 DES、IDEA、AES 等。

序列密码，也被称为流密码，是将明文和密钥都划分为位或字符的序列，并且对明文序列中的每一位或字符都用密钥序列中的对应分量来加密。公开的序列密码算法主要有 RC4、SEAL 等。

分组密码每一次加密一个明文分组，而序列密码每一次加密一位或者一个字符，两种密码在计算机系统中都有广泛的应用。

2.2.2 设计思想

混淆（confusion）与扩散（diffusion）是设计对称密码的主要方法，最早出现在香农 1945 年的论文《密码学的数学理论》中。

（1）混淆（Confusion）：是一种使密钥与密文之间的关系尽可能模糊的加密操作。如今实现混淆常用的一个元素就是替换；这个元素在 AES 和 DES 中都有使用。

（2）扩散（Diffusion）：是一种为了隐藏明文的统计特性而将一个明文符号的影响扩散到多个密文符号的加密操作。最简单的扩散元素就是位置换，它常用于 DES 中；而 AES 则使用更高级的 Mixcolumn 操作。

仅执行扩散不够安全，但是将扩散操作串联起来就可以建立一个更强壮的密码。将若干加密操作串联起来的思想也是 Shannon 提出的，这样的密码也叫乘积密码（Product cipher）。乘积密码就是以某种方式连续执行两个或多个密码，以使得所得到的最后结果或乘积从密码编码的角度比其任意一个组成密码都更强。

目前，所有的分组密码都是乘积密码，因为它们都是由对数据重复操作的轮组成的，其中每轮都执行一次扩散和混淆操作，如下图所示：

图 1.2.1 乘积密码示例

2.2.3 工作模式

即使有了安全的分组密码算法，也需要采用适当的工作模式来隐蔽明文的统计特性、数据的格式等，以提高整体的安全性，降低删除、重放、插入和伪造成功的机会。

分组密码的工作模式是一个算法，它刻画了如何利用分组密码提供信息安全服务。主要有五种工作模式，它们分别是电子密码本 ECB（Electronic Codebook Mode）模式、密码分组链 CBC（Cipher Block Chaining Mode）模式、密码反馈 CFB（Cipher Feedback Mode）模式、输出反馈 OFB（Output Feedback Mode）模式、计数 CTR（Counter Mode）模式。

本书主要简单介绍前两类工作模式。

（1）电子密码本模式

直接利用分组密码对明文的各分组进行加密。设明文 $M=(M_1, M_2, \cdots, M_n)$，相应的密文 $C=(C_1, C_2, \cdots, C_n)$，其中：

$$C_i = E_k(M_i), i=1, 2, \cdots, n$$

电子密码本是分组密码的基本工作模式。

ECB 的一个缺点是容易暴露明文的数据模式，比如相同的明文块的密文是完全相同的。

ECB 容易遭受重放攻击。攻击者通过重放，可以在不知道密钥情况下修改被加密过的消息，用这种办法欺骗接收者。例如在实际应用中，不同的消息可能会有一些比特序列是相

同的（消息头），攻击者重放消息头，修改消息体以欺骗接收者。

（2）密码分组链模式

明文要与前面的密文进行异或运算然后被加密，从而形成密文链。

密码分组链模式的运行原理如图 1.2.2 所示。每一分组的加密都依赖于所有前面的分组。在处理第一个明文分组时，与一个初始向量（IV，Initialization Vector）组进行异或运算。IV 不需要保密，它可以明文形式与密文一起传送。

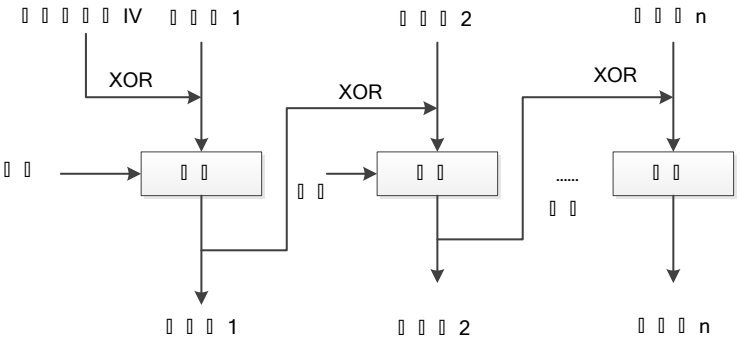


图 1.2.2 密码分组链模式

使用 IV 后，完全相同的明文被加密成不同的密文。敌手再用分组重放进行攻击是完全不可能的了。

CBC 模式除了用于加密大长度明文外，还常用于报文鉴别与认证。

2.2.4 应用示例

实验 2.1：安装 OpenSSL 并进行对称密码的 CBC 等模式进行数据加密。

1. OpenSSL 安装

一般情况下，Ubuntu 系统存在预装的 OpenSSL，可使用 openssl version 查看版本号。

(1) 使用 APT 软件包管理工具安装

可使用以下命令安装或更新 OpenSSL，需要保证处于 root 账户下或使用 sudo 命令提权

```
[sudo] apt install openssl
```

若需使用 OpenSSL 开发组件，则需要额外安装 libssl-dev 包

```
[sudo] apt install libssl-dev
```

(2) 使用源码编译安装

① 卸载预装版本

在安装新版本前，您可根据需要，使用如下命令卸载预装版本。

```
[sudo] apt autoremove openssl
```

② 下载源码包，下载链接可从官网（openssl.org）获取

```
wget https://www.openssl.org/source/openssl-<版本号>.tar.gz
```

③ 解压并进入到该目录

```
tar -zxvf openssl-<版本号>.tar.gz
cd openssl-<版本号>
```

④ 使用 config 生成 Makefile

```
./config
```

若想修改安装地址，可使用

```
./config --prefix=/opt/openssl --openssldir=/usr/local/ssl
```

--prefix 指定安装的根目录，默认地址为/usr/local；

--openssldir 指定配置文件地址，以及证书和密钥的默认存放目录，默认地址为/usr/local/ssl。

⑤ 编译并安装

```
make
[sudo] make install
```

⑥ 解决动态链接库查找问题

安装完成后，可能会出现动态连接库相关错误。这是因为 openssl 默认放置动态连接库的地址并非系统搜寻目录，可通过以下命令添加。

```
[sudo] ldconfig /usr/local/lib          #openssl 1.1.*
[sudo] ldconfig /usr/local/lib64        #openssl 3.0.*
```

2. 对称加密及工作模式

(3) 使用 OpenSSL 命令加解密文件

openssl enc 对称密码命令可使用各种块和流密码对数据进行加密或解密，通过 openssl enc -help 可查看全部选项。

① 文件加密

使用 aes-128-cbc 对 message.txt 文件进行加密并使用 base64 编码，输出到 ciphertext.txt。假设 128 位密钥为 a3171d177d1ce97ebc644ea3ff826b4e，初始向量为 8bc65f2f883f95eea10b6f940cc805f6。

```
openssl enc -e -aes-128-cbc -in message.txt -out ciphertext.txt -K a3171d177d1ce97ebc644ea3ff826b4e -iv 8bc65f2f883f95eea10b6f940cc805f6 -base64
```

② 文件解密

对 ciphertext.txt 进行 base64 解码并解密，结果输出到 plaintext.txt。

```
openssl enc -d -aes-128-cbc -in ciphertext.txt -out plaintext.txt -K a3171d177d1ce97ebc644ea3ff826b4e -iv 8bc65f2f883f95eea10b6f940cc805f6 -base64
```

以下为部分语法解释。

语法：enc [options]

选项	作用
-e	加密
-d	解密
-in infile	指定输入文件
-out outfile	指定输出文件
-base64	加密时在加密后进行 base64 编码，或在解密时首先对密文进行解码
-K val	指定密钥
-iv val	提供初始向量

(4) 加解密程序

① 编写程序文件 aes-128-cbc.cpp

```
#include <stdio.h>
#include <string.h>
#include <openssl/evp.h>

// aes-128-cbc 加密函数
bool aes_128_cbc_encrypt(const uint8_t *in, int in_len, uint8_t *out, int *out_len, const uint8_t *key, const uint8_t *iv)
{
	// 创建上下文
	EVP_CIPHER_CTX *ctx = EVP_CIPHER_CTX_new();
	if (!ctx)
		return false;
	bool ret = false;
	// 初始化加密模块
	if (EVP_EncryptInit_ex(ctx, EVP_aes_128_cbc(), NULL, key, iv) <= 0)
		goto err;
	int update_len;
	// 向缓冲区写入数据，同时将以对齐的数据加密并返回
	if (EVP_EncryptUpdate(ctx, out, &update_len, in, in_len) <= 0)
		goto err;
	int final_len;
	// 结束加密，填充并返回最后的加密数据
	if (EVP_EncryptFinal_ex(ctx, out + update_len, &final_len) <= 0)
		goto err;
	*out_len = update_len + final_len;
	ret = true;
err:
	EVP_CIPHER_CTX_free(ctx);
	return ret;
}

// aes-128-cbc 解密函数，结构与加密相似
bool aes_128_cbc_decrypt(const uint8_t *in, int in_len, uint8_t *out, int *out_len, const uint8_t *key, const uint8_t *iv)
{
	EVP_CIPHER_CTX *ctx = EVP_CIPHER_CTX_new();
	if (!ctx)
```

```
        return false;
    bool ret = false;
    if (EVP_DecryptInit_ex(ctx, EVP_aes_128_cbc(), NULL, key, iv) <= 0)
        goto err;
    int update_len;
    if (EVP_DecryptUpdate(ctx, out, &update_len, in, in_len) <= 0)
        goto err;
    int final_len;
    if (EVP_DecryptFinal_ex(ctx, out + update_len, &final_len) <=0)
        goto err;
    *out_len = update_len + final_len;
    ret = true;
err:
    EVP_CIPHER_CTX_free(ctx);
    return ret;
}

int main()
{
    // 密钥
    uint8_t key[] = {35, 31, 71, 44, 34, 42, 76, 16, 86, 27, 93, 59, 26, 62, 4, 19};
    // 初始化向量
    uint8_t iv[] = {91, 66, 51, 17, 14, 40, 65, 38, 4, 60, 89, 44, 87, 63, 67, 32};
    const char *msg = "Hello World!";
    const int msg_len = strlen(msg);
    // 存储密文
    uint8_t ciphertext[32] = {0};
    int ciphertext_len;
    // 加密
    aes_128_cbc_encrypt((uint8_t *)msg, msg_len, ciphertext, &ciphertext_len,
(uint8_t *)key, (uint8_t *)iv);
    // 存储解密后的明文
    uint8_t plaintext[32] = {0};
    int plaintext_len;
    // 解密
    aes_128_cbc_decrypt((uint8_t *)ciphertext, ciphertext_len, plaintext,
&plaintext_len, (uint8_t *)key, (uint8_t *)iv);

    // 输出解密后的内容
    printf("%s\n", plaintext);
    return 0;
}
```

② 编译并运行

```
g++ signature.cpp -o signature -lcrypto
./signature
```

2.3 公钥密码

对称加密最大的缺点在于其密钥管理困难，因为通信的双方必须首先预约相同的密钥后才能进行加密，而且在网络环境下，为了安全，密钥应当经常更换，在用户越多的情况下，

密钥的数量及管理难度相应增加。

1976 年，Diffie 和 Hellman 第一次提出了公开密钥密码的概念，开创了一个密码新时代。

2.3.1 基本概念

公开密钥密码，也就是非对称密码体制，基本思想是将对称密码的密钥 k 一份为二，分为加密密钥 k_e 和解密密钥 k_d，用加密密钥 k_e 控制加密，用解密密钥 k_d 控制解密，而且由计算复杂性确保由加密密钥 k_e 在计算上不能推出解密密钥 k_d。这样，即使是将 k_e 公开也不会暴露 k_d，也不会损害密码的安全。于是便可将 k_e 公开，而只对 k_d 保密。由于 k_e 是公开的，只有 K_d 是保密的，所以便从根本上克服了传统密码在密钥分配上的困难。

（1）加解密过程

公开密钥密钥体制中，一个用户有两个密钥，即公钥 k_e 和私钥 k_d，加解密的过程如下：

① 如果要给一个拥有公钥 k_e 的用户发送信息 m，则可以用其公钥 k_e 执行加密算法 E，以加密明文 m 得到密文 c：$c = E_{k_e}(m)$。

② 用户收到密文之后，用自己的私钥 k_d 执行解密算法 D 以恢复明文 m：$m = D_{k_d}(c)$。

可见，加密算法 E 和解密算法 D 是可逆运算，即：$D(E(m))=m$，不过加解密过程使用的密钥不同，不同于对称加密无论是加密还是解密均使用相同的密钥。

然而，由于公开密钥密码通常依赖于某个难解的问题设计，虽然安全性高，但降低了加解密效率，是公开密钥密码一大缺点。在应用中，通常采用对称密码体制实现数据加密、公钥密码体制实现密钥管理的混合加密机制。公钥密码主要用于身份认证、密钥协商，一种混合机制是通过公钥密码进行密钥协商后得到一个对称密码体制的会话密钥，进而用作加密。

（2）数字签名应用

私钥唯一、不公开且不可伪造的特性，使得非对称密码可以应用到数字签名中。

一个拥有公钥 k_e 和私钥 k_d 的用户，实现数字签名的过程如下：

① 拥有公钥 k_e 的可以用其私钥 k_d 执行签名算法 S，以产生信息 m 的签名信息 sig：$sig=S_{k_d}(m)$。

② 要验证一个签名信息 sig 是否某用户的签名时，只需要用该用户的公钥 k_e 执行验签方法计算出校验值 m'：$m'=V_{k_e}(c)$，如果 $m'=m$，那么验证成功，否则失败。

（3）优缺点

如上面提到的基本思想及其在数字签名的应用，公开密钥密码的优点在于从根本上克服了对称密码密钥分配上的困难，且易于实现数字签名。然而，由于公开密钥密码通常依赖于某个难解的问题设计，虽然安全性高，但降低了加解密效率，是公开密钥密码一大缺点。

在应用中，通常采用对称密码体制实现数据加密、公钥密码体制实现密钥管理的混合加密机制。公钥密码主要用于身份认证、密钥协商，一种混合机制是通过公钥密码进行密钥协商后得到一个对称密码体制的会话密钥，进而用作加密。

2.3.2 设计思想

在安全性方面，非对称密码通常依赖于某个难解的数学难题。也就是说，基**于难解问题设计密码是非对称密码设计的主要思想**。

如 1.1.3 节所述，**NP 问题不一定在多项式时间内可解，但可以在多项式时间内验证**。比如：大数分解问题，给定一个极大数，让你拆成两个素数相乘，可能很久都解不出来，也就是在多项式时间复杂度内不可以解决。但是，我告诉你这是 p*q 得到的，那么很简单就能在多项式时间内验证是否正确，这就是 NP 问题。

公开密钥密码，开拓了直接利用 NP 问题设计密码的技术路线。对于一个 NP 问题，若能找到一个计算序列，用于设计加解密算法，那么密码分析者在不知道计算序列的情形下求解问题（称为客观求解）成为计算上的不可能。也就是说，NP 问题在密码学中的价值便是这些 NP 问题没有已知的算法可以在多项式时间内解决，作为攻击者可能需要指数级或者更高的破译时间。

典型且流行的非对称密码包括 RSA、ElGamal、椭圆曲线密码（ECC，Elliptic Curves Cryptography）等。RSA 基于大整数因数分解难的问题设计；ElGamal 基于离散对数求解困难的问题设计；而 ECC 基于椭圆曲线离散对数求解困难的问题设计。

自 2000 年起，基于双线性对的基于身份加密（IBE）、基于属性加密（ABE）等，还有基于格的公钥密码体制逐渐兴起，并且在减少密钥管理复杂度、细粒度访问控制、抗量子攻击密码体制设计等起到了重要作用。

2.3.3 RSA 算法

1978 年美国麻省理工学院的三名密码学者 Rivest、Shamir 和 Adleman 提出了一种基于大整数因数分解困难性的公开密钥密码，简称为 RSA 密码。

1. 大整数因数分解问题

大整数因数分解问题是指：将两个大素数相乘十分容易，但想要对其乘积进行因数分解却极其困难，因此可以将乘积公开作为加密密钥。

大整数因数分解就是一个典型的 NP 问题。

2. 算法描述

（1）密钥生成

· 随机选择两个大素数 p 和 q，p 和 q 都保密；

· 计算 $n=pq$，将 n 公开；

· 计算 $\phi(n)=(p-1)(q-1)$， ϕ(n)保密；

· 随机选取一个正整数 e，$1<e<\phi(n)$且 e 与 $\phi(n)$互素，将 e 公开；e 和 n 就构成了用户的公钥；

· 根据 $ed\equiv 1 \bmod \phi(n)$（≡表示模同余运算），计算出 d，d 保密；d 和 n 构成了用户的私钥；

由以上算法可见，RSA 密码的公开加密密钥 $K_e=<n, e>$，而保密的解密密钥 $K_d=<p, q, d, \phi(n)>$，保存 $p, q, \phi(n)$是为了计算加速。

说明：算法中的 $\phi(n)$是一个数论函数，称为欧拉函数，表示在比 n 小的正整数中与 n 互素的数的个数。若 p 和 q 是素数，且 $n=pq$，则 $\phi(n)=(p-1)(q-1)$。例如，6=2*3, $\phi(6)=2$，而 1~5 之间和 6 互素的数是 1 和 5 两个数。

（2）加密

加密运算：$C=M^e \bmod n$；

（3）解密

解密运算：$M=C^d \bmod n$。

3. 正确性

给定密文 $C=M^e \bmod n$ ，可以利用私钥 $K_d=<p, q, d, \varphi(n)>$解密，即

$$(C)^d = (M^e)^d = M^{ed} = M \bmod n$$

通过 RSA 加解密运算可以看出，加密和解密运算具有可交换性：

$$D(E(m)) = (M^e)^d = M^{ed} = (M^d)^e = E(D(m)) \bmod n$$

通过该可交换性看出，如果执行解密算法产生的签名，也可以通过公钥来进行验证。

4. 安全性

密码分析者攻击 RSA 密码的一种可能的途径是截获密文 C，从中求出明文 M。他们知道：

$$M=C^d \bmod n$$

因为 n 是公开的，要从 C 中求出明文 M，必须先求出 d，而 d 是保密的。但他们知道，e 是公开的，要从中求出 d，必须先求出 $\phi(n)$，而 $\phi(n)$是保密的。但他们知道：

$$n=pq$$

要从 n 求出 p 和 q，只有对 n 进行因数分解。

由此可见，只要能对 n 进行因数分解，便可攻破 RSA 算法。虽然大合数因数分解是十分困难的，但是随着科学技术的发展，人们对于大合数因数分解的能力在不断的提高。1994

年，成功分解了 129 位的大合数；1996 年又破译了 RSA-130；1999 年又破译了 RSA-140。现在，科学家们正向更高位数的 RSA 发起冲击。因此，要应用 RSA 密码，应当采用足够大的整数 n。普遍认为，n 至少应取 1024 位，最好是 2048 位。

除了通过因数分解攻击 RSA 外，还有一些攻击方法，但是还不能构成有效威胁。因此，完全可以认为，只要合理的选择参数，正确的使用，RSA 就是安全的。

2.3.4 ElGamal 算法

ElGamal 密码是除了 RSA 密码之外最有代表性的公开密钥密码。RSA 密码建立在大整数因数分解的困难性之上，而 ElGamal 密码建立在离散对数的困难性之上。

1. 离散对数问题

设 p 为素数，若存在一个正整数 g，使得 $g, g^2, g^3, \cdots, g^{p-1}$，关于模 p 互不同余，则称为模 p 的本原元。显而易见若 g 为模 p 的本原元，则对于 $i \in \{1, 2, 3, \cdots, p-1\}$ 一定存在一个正整数 k，使得 $i \equiv g^k \bmod p$。

设 p 为素数，g 为模 p 的本原元，g 的幂乘运算为

$$Y \equiv g^X \bmod p, 1 \le X \le p-1$$

其中，X 是以 g 为底的模 p 的对数，即求解对数 X 的运算为

$$X \equiv \log_g Y, 1 \le X \le p-1$$

由于上述运算是定义在模 p 有限域上的，所以称为离散对数运算。

从 X 计算 Y 是容易的，可是从 Y 计算 X 就困难的多，利用目前最好的算法，对于小心选择的 p 将至少需用 $p^{1/2}$ 次以上的运算，只要 p 足够大，求解离散对数问题是相当困难的。

这便是著名的离散对数问题。可见，离散对数问题具有较好的单向性。

由于离散对数具有较好的单向性，所以离散对数问题在公钥密码学中得到广泛应用。除了 ElGamal 密码外，Diffie-Hellman 密钥分配协议和美国数字签名标准算法 DSA 等也都是建立在离散对数问题之上的。

2. ElGamal 密码

（1）密钥生成

随机选择一个大素数 p，且要求 p-1 有大素数因子。再选择一个模 p 的本原元 g，将 p 和 g 公开。

用户随机的选择一个整数 d 作为自己的私钥，$1 \le d \le p-2$，计算 $y \equiv g^d \bmod p$，取 y 为自己的公钥。

（2）加密

将明文消息 M ($0 \le M \le p-1$)加密成密文的过程如下：

· 随机选择一个整数 r，$1 \le r \le p-2$。

· 计算 $U = y^r \bmod p$　　$C_1 = g^r \bmod p$　　$C_2 = UM \bmod p$

· 取(C_1, C_2)作为密文

（3）解密

将密文(C_1, C_2)解密的过程如下：

· 计算 V= $C_1^d \bmod p$

· 计算 $M = C_2 V^{-1} \bmod p$

3. 正确性

解密的可还原性可证明如下：

因为，$C_2 V^{-1} \bmod p = (UM)V^{-1} \bmod p$

$= (UM)(C_1^d)^{-1} \bmod p$

$= (UM)((g^r)^d)^{-1} \bmod p$

$= (UM)((g^d)^r)^{-1} \bmod p$

$= (UM)((y)^r)^{-1} \bmod p$

$= (UM)(U)^{-1} \bmod p$

$= M \bmod p$

故解密可还原。

4. 安全性

由于 ElGamal 密码的安全性建立在 GF(p)离散对数的困难性上，而目前尚无求解有限域 GF(p)离散对数的有效算法，所以在 p 足够大时 ElGamal 密码是安全的。

为了安全，p 应该为 150 位以上的十进制数，而且 p-1 应有大素因子。

此外，为了安全加密和签名所使用的 r 必须是一次性的。这是因为，如果使用的 r 不是一次性的，时间长了就可能被攻击者获得。又因 y 是公开密钥，攻击者就可以计算出 U，进而利用 Euclid 算法求出 U^{-1}。又因为攻击者可以获得密文 C_2，因此可以通过计算 $U^{-1} C_2$ 得到明文 M。另外，假设用同一个 r 加密两个不同的明文 M 和 M'，相应的密文为(C_1, C_2) 和(C_1', C_2')。因为 $C_2 / C_2' = M / M'$，如果攻击者知道 M，则很容易求出 M'。

2.3.5 椭圆曲线密码

1. 椭圆曲线离散对数问题

在椭圆曲线加密（ECC，Elliptic curve discrete）中，利用了某种特殊形式的椭圆曲线，即定义在有限域上的椭圆曲线。其方程如下：

$$y^2 = x^3+ax+b \pmod p$$

其中 p 为一个素数或 2 的幂（有时也称为椭圆曲线的秩），且 $4a^3+27b^2\neq 0$。

椭圆曲线离散对数问题（ECDLP，Elliptic curve discrete logarithm problem）定义如下：给定素数 p 和椭圆曲线 E，给定椭圆曲线上两点 P 和 W，对 $W=kP$，求 k 的值。可以证明，已知 k 和 P 计算 W 比较容易，而由 W 和 P 计算 k 则比较困难，至今没有有效的方法来解决这个问题，这就是椭圆曲线加密算法原理之所在。

注意：kP 是指定义在加法运算符“+”上的 k 个 P 执行运算“+”，类似于 P^k 次方。具体将在 2.2.2 部分进行讲解。

类似这样的特性：向一个方向计算容易，但反方向倒推很难的算法被称作单向陷门函数（Trapdoor Function）。椭圆曲线离散对数问题可以用来构造一个很好的陷门函数。

2. 椭圆曲线加密

将 ECDLP 嵌入到加密系统中的方法有很多种，最简单的一种实现方法描述如下：

① Alice 首先选择一条椭圆曲线以及该曲线上的一点 G，然后选取一个保密数字 k，该数字将作为她的私钥。她的公钥为 G 和 P_A，其中 $P_A=kG$，并将公钥给 Bob。

② 当 Bob 要给 Alice 发送一个消息时，他需要先将明文转换成数字 m，并在曲线上找出一点 P_m，使其 x 坐标值与 y 坐标值之差为 m。再选择一个随机数 r，然后将得到的密文 $C=(C_1, C_2)$发送给 Alice，其中：

$$C_1 = rG, C_2 = P_m + rP_A$$

③ Alice 收到消息后，通过用自己的私钥与 C_1 相乘，并用第二点减去它，就可以很容易完成解密：

$$C_2 - kC_1 = P_m + rP_A - k(rG) = P_m + r(kG) - k(rG) = P_m$$

3. 安全性与性能

攻击者得到 C_1 和 C_2 后，因为椭圆曲线离散对数问题，它无法计算计算得到 r 或者 k，因此是安全的。

ECC 使用较小的密钥就可以提供比 RSA 更高的安全级别，如 160 位 ECC 与 1024 位

RSA 有相同的安全强度。

2.3.6 应用示例

公钥基础设施（PKI, Public Key Infrastructure），是一种遵循既定标准的密钥管理平台，它能够为所有网络应用提供加密和数字签名等密码服务及所必需的密钥和证书管理体系，简单来说，PKI 就是利用公钥理论和技术建立的提供安全服务的基础设施。

数字证书是指在互联网通讯中标志通讯各方身份信息的一个数字认证，人们可以在网上用它来识别对方的身份。在 PKI 体系中，建有证书管理机构 CA (Certificate Authority) 。CA 中心的公钥是公开的，因此由 CA 中心签发的内容均可以验证。

密钥的生存周期包括：密钥的产生和登记、密钥分发、密钥更新、密钥撤销、密钥销毁等。在产生密钥后，公钥需要在 PKI 中登记，并通过 CA 中心的私钥签名后形成公钥证书。由于 CA 中心的公钥公开，用户可以方便的对公钥证书进行验证，进而用户可以通过公钥证书来互相交换自己的公钥。进而，PKI 作为安全基础设施，能够提供身份认证、数据完整性、数据保密性、数据公正性、不可抵赖性和时间戳六种安全服务。

PKI 的应用非常广泛，其为网上金融、网上银行、网上证券、电子商务、电子政务等网络中的数据交换提供了完备的安全服务功能。

OpenSSL 库提供了相关的基本功能支撑，下面提供一个数字签名及认证的简单示例。

实验 2.2：在 OpenSSL 中进行数据签名及验证。

基于第 2.2.4 节的示例，在 OpenSSL 中进行数据签名及验证的实验如下所示。

(5) 使用 OpenSSL 命令签名并验证

① 生成 2048 位密钥，存储到文件 id_rsa.key

```
openssl genrsa -out id_rsa.key 2048
```

② 根据私钥文件，导出公钥文件 id_rsa.pub

```
openssl rsa -in id_rsa.key -out id_rsa.pub -pubout
```

③ 使用私钥对文件 message.txt 进行签名，输出签名到 message.sha256

```
openssl dgst -sign id_rsa.key -out rsa_signature.bin -sha256 message.txt
```

④ 使用公钥验证签名

```
openssl dgst -verify id_rsa.pub -signature rsa_signature.bin -sha256 message.txt
```

若验证成功，会输出 Verified OK 字段。

以下为部分语法解释。

语法：genrsa [options] numbits

选项	作用
-out	指定输出文件

numbits	密钥长度，存在默认值

语法：rsa [options]

选项	作用
-in	指定输入文件
-out	指定输出文件
-pubout	输出公钥

语法：dgst [options] [file...]

选项	作用
-sign val	生成签名，同时指定私钥
-versify val	使用公钥验证签名
-prverify val	使用私钥验证签名
-out outfile	输出到文件
-signature infile	指定签名文件
-sha256	使用 sha256 算法摘要
file	消息文件

(6) 数字签名程序

① 编写程序文件 signature.cpp

```
#include <stdio.h>
#include <string.h>
#include <openssl/evp.h>
#include <openssl/rsa.h>
#include <openssl/pem.h>

// 公钥文件名
#define PUBLIC_KEY_FILE_NAME "public.pem"
// 私钥文件名
#define PRIVATE_KEY_FILE_NAME "private.pem"

// RSA 生成公私钥，存储到文件
bool genrsa(int numbit)
{
	EVP_PKEY_CTX *ctx = EVP_PKEY_CTX_new_id(EVP_PKEY_RSA, NULL);
	if (!ctx)
		return false;
	EVP_PKEY *pkey = NULL;
	bool ret = false;
	int rt;
	FILE *prif = NULL, *pubf = NULL;
	if (EVP_PKEY_keygen_init(ctx) <= 0)
		goto err;
	// 设置密钥长度
	if (EVP_PKEY_CTX_set_rsa_keygen_bits(ctx, numbit) <= 0)
		goto err;
	// 生成密钥
	if (EVP_PKEY_keygen(ctx, &pkey) <= 0)
		goto err;
	prif = fopen(PRIVATE_KEY_FILE_NAME, "w");
	if (!prif)
```

```
            goto err;
        // 输出私钥到文件
        rt = PEM_write_PrivateKey(prif, pkey, NULL, NULL, 0, NULL, NULL);
        fclose(prif);
        if (rt <= 0)
            goto err;

        pubf = fopen(PUBLIC_KEY_FILE_NAME, "w");
        if (!pubf)
            goto err;
        // 输出公钥到文件
        rt = PEM_write_PUBKEY(pubf, pkey);
        fclose(pubf);
        if (rt <= 0)
            goto err;
        ret = true;
err:
        EVP_PKEY_CTX_free(ctx);
        return ret;
}

// 生成数据签名
bool gensign(const uint8_t *in, unsigned int in_len, uint8_t *out, unsigned int *out_len)
{
        FILE *prif = fopen(PRIVATE_KEY_FILE_NAME, "r");
        if (!prif)
            return false;
        // 读取私钥
        EVP_PKEY *pkey = PEM_read_PrivateKey(prif, NULL, NULL, NULL);
        fclose(prif);
        if (!pkey)
            return false;
        bool ret = false;
        EVP_MD_CTX *ctx = EVP_MD_CTX_new();
        if (!ctx)
            goto ctx_new_err;
        // 初始化
        if (EVP_SignInit(ctx, EVP_sha256()) <= 0)
            goto sign_err;
        // 输入消息，计算摘要
        if (EVP_SignUpdate(ctx, in, in_len) <= 0)
            goto sign_err;
        // 生成签名
        if (EVP_SignFinal(ctx, out, out_len, pkey) <= 0)
            goto sign_err;
        ret = true;
sign_err:
        EVP_MD_CTX_free(ctx);
ctx_new_err:
        EVP_PKEY_free(pkey);
        return ret;
}
```

```
// 使用公钥验证数字签名，结构与签名相似
bool verify(const uint8_t *msg, unsigned int msg_len, const uint8_t *sign, unsigned int sign_len)
{
	FILE *pubf = fopen(PUBLIC_KEY_FILE_NAME, "r");
	if (!pubf)
		return false;
	// 读取公钥
	EVP_PKEY *pkey = PEM_read_PUBKEY(pubf, NULL, NULL, NULL);
	fclose(pubf);
	if (!pkey)
		return false;
	bool ret = false;
	EVP_MD_CTX *ctx = EVP_MD_CTX_new();
	if (!ctx)
		goto ctx_new_err;
	// 初始化
	if (EVP_VerifyInit(ctx, EVP_sha256()) <= 0)
		goto sign_err;
	// 输入消息，计算摘要
	if (EVP_VerifyUpdate(ctx, msg, msg_len) <= 0)
		goto sign_err;
	// 验证签名
	if (EVP_VerifyFinal(ctx, sign, sign_len, pkey) <= 0)
		goto sign_err;
	ret = true;
sign_err:
	EVP_MD_CTX_free(ctx);

ctx_new_err:
	EVP_PKEY_free(pkey);
	return ret;
}

int main()
{
	// 生成长度为 2048 的密钥
	genrsa(2048);
	const char *msg = "Hello World!";
	const unsigned int msg_len = strlen(msg);
	// 存储签名
	uint8_t sign[256] = {0};
	unsigned int sign_len = 0;
	// 签名
	if (!gensign((uint8_t *)msg, msg_len, sign, &sign_len))
	{
		printf("签名失败\n");
		return 0;
	}
	// 验证签名
	if (verify((uint8_t *)msg, msg_len, sign, sign_len))
		printf("验证成功\n");
	else
```

```
            printf("验证失败\n");
    return 0;
}
```

② 编译并运行

```
g++ signature.cpp -o signature -lcrypto
./signature
```

2.4 可证明安全性

随着现代密码学理论的发展，在面向各种应用的密码方案设计中，可证明安全性（Provable Security）受到越来越多的重视。某个方案没有通过形式化验证确认其可证明安全性，是无法想象能被广泛接受而投入到现实应用当中来。

传统将某个方案的安全性简单归结为某种函数运算（大整数分解、背包问题、离散对数等）的单向性上的设计思路，并没有充分考虑适应性攻击者攻击能力和实际应用中复杂的环境条件影响，因而采用该思路而设计的方案往往很快被发现其中可被适应性攻击者利用的漏洞，例如二次剩余加密方案可被适应性攻击者利用，以不可忽略的概率对模数进行分解等。

2.4.1 基本概念

1. 安全参数

在密码学中，安全参数（Security Parameter）是用来衡量一个攻击方（adversary）攻破一个加解密机制有多困难的方式。直观的理解就是，安全参数越大，对应的破解加密系统的难度也就越大。安全参数有两种类型：计算安全参数和统计安全参数。

（1）计算安全参数

计算安全参数（computational，通常使用符号λ表示）表示攻击者在算力有限的情况下，应用离线计算破解一个问题的困难程度，例如破解一个加密算法的困难程度。计算安全参数λ定义了计算的数值空间大小，通常是用 bit 位数表示。更大的λ也就对应了所需要计算的次数更大，所花费的时间越长。

（2）统计安全参数

统计安全参数（statistical，通常使用符号σ表示）表示攻击方者在算力无限的情况下，破解加密机制的困难程度。统计安全参数依赖于不同变量分布之间的统计距离。直观上说，如果两个变量之间的统计距离越小，则越难区分猜测值和目标变量值。如果一个协议是统计安全的，那么其属于信息论安全，意指无论攻击方算力强弱，也无法破解系统。

2. 完美安全

如果一个具有无限计算能力的敌手从给定的密文中不能获取明文的任何有用信息，则说明这个加密体制具有完美安全性或信息论安全性。在 1949 年，香农在他的论文里提出了完美密码模式（定义在对称密码机制上），称为香农理论：

设（KeyGen, Enc, Dec）是一种在明文域M上的加密模式，且$|M| = |K| = |C|$，则当且仅当：

✧ 所有密钥$k \in K$都以$\frac{1}{|K|}$的概率通过 KeyGen 生成；

✧ 对所有明文$m \in M$，$c \in C$，都存在单独密钥$k \in K$使得$Enc_k(m)$输出c。

一次一密是达到完美安全的设计方法，它要求每次均使用随机变化的密钥进行加解密。这样，即使攻击者获取信道上的秘密消息，由于其不知道当前使用的密钥，也无法进行解密。

3. 计算安全

虽然一次一密理论上是安全的，但是它带来的密钥管理等开销使其难以在大多数现实场景中广泛应用。香农定理告诉我们，如果一个加密体制具有完美保密性，那么密钥数量至少要和明文数量一样多。因受香农定理制约，一次一密要求密钥长度至少要和明文一样长，用它加密 1GB 的视频文件时，密钥也至少得是 1GB。事实上，不需要实现一个绝对安全的密码系统，只需要保证密码算法的破解对攻击者来说是“计算上不可行”的即可，也就是计算安全性。对敌手的计算安全性需要考虑以下两个方面：（1）攻击的运行时间；（2）攻击的成功概率。

假设一个密钥有λ比特长，那么最暴力的攻击方式是攻击者需要尝试2^{λ}种可能的密钥。这里的λ就是安全参数，λ的增大意味着攻击者暴力破解的难度呈指数增加。而为了评估λ的上限，对敌手计算安全性的考量变更为：（1）渐进运行时间；（2）渐进攻击成功概率。

（1）渐进运行时间

那么如何使得算法的破解是计算上不可行呢？正如 1.1.2 节所讲，只需保证算法在“多项式时间”内不能被破解即可。

（2）渐进攻击成功率

密码系统对攻击者的安全性另一方面也要考虑渐进攻击成功率。如果随着λ的增加，如果攻击成功率趋近于 0，那么即使攻击时间在多么有效率的多项式时间内都无效。

为了表示这个思想，就需要使用“可忽略函数”。可忽略函数是一个极小量，定义如下：

可忽略函数 υ：N→R 是任意一个趋近于 0 的速度比任何逆多项式都快的函数。换句话说，对于任意多项式p，除了有限多个 n 以外，均有$\upsilon(n) < 1/p(n)$。

可忽略函数有以下两个性质，即若$f_1(\lambda)$和$f_2(\lambda)$为可忽略函数，则：

（1） $f_3(\lambda) = f_1(\lambda) + f_2(\lambda)$为可忽略函数；

（2） 对所有正多项式p，函数$f_4(\lambda) = p(\lambda)f_1(\lambda)$为可忽略函数。

可忽略函数的重要意义在于：如果一个事件是可忽略事件，那么由于它几乎不可能发生，那么就可以被实际目标所忽略。因此如果对密码系统的某次攻击仅具有可忽略的可能性，那么在设计密码模式时，就无需考虑这种攻击。

2.4.2 敌手能力

敌手的攻击能力是评判模型安全能力的重要角度之一。

1. 密码方案的敌手能力

在对密码方案进行尝试破解的敌手中，依照攻击者的攻击能力由弱到强，可分为以下四种基本攻击模式：

- 唯密文攻击（Cipher-only attack）：最基础的攻击方式，敌手目标是通过观察密文，尝试推测出相应的明文信息。
- 已知明文攻击（Known-plaintext attack）：敌手已知部分在同一密钥下加密的明文-密文对，敌手的目标是通过已知的明文-密文对，推测出其他一些密文对应的明文信息。
- 选择明文攻击（Chosen-plaintext attack，CPA）：敌手可以选择获取任意其所选取的明文加密后的结果，敌手目标是通过这些明文信息推测出其他密文对应的明文信息。
- 选择密文攻击（Chosen-ciphertext attack，CCA）：敌手甚至可以获取任意其所选择的密文解密后的结果，敌手目标是通过这些信息推测出其他一些（其无法随意选取的）密文对应的明文信息。

2. 安全协议的敌手能力

上述密码方案的敌手主要以破解密文或者猜测密钥位主要目的，敌手所针对的密码方案中交互次数少、形式较为单一。但是，随着研究的进展，以安全多方计算等为代表的复杂的安全协议被广泛应用，这些安全协议设计复杂、参与方多、存在多个协议并行运行的情况。

主要考虑两种敌手：半诚实攻击者和恶意攻击者。

- 半诚实（Semi-honest）攻击者可以攻陷参与方，但会遵循协议规则执行协议。换句话说，攻陷参与方会诚实地执行协议，但可能会尝试从其他参与方接收到的消息中尽可能获得更多的信息，是一种诚实但好奇（Honest-but-curious）的攻击者。半诚实攻击者也被称为*被动*（Passive）*攻击者*，因为此类攻击者只能通过观察协议执行过程中自己的视角来尝试得到秘密信息，无法采取其他任何攻击行动。
- 恶意（Malicious）攻击者，或称*主动*（Active）*攻击者*，可以让攻陷参与方任意偏离协议规则执行协议，以破坏协议的安全性。恶意攻击者分析协议执行过程的能力与半诚实攻击者相同，但恶意攻击者可以在协议执行期间采取任意行动。请注意，这意味着攻击者可以控制或操作网络，或在网络中注入任意消息。

2.4.3 安全性定义

1. 完美保密性

历史上关于第一个关于对称加密方案的安全性的严格定义是香农提出的完美保密性，发生在上世纪四十年代，如下：

如果$\forall\, m_0, m_1 \in M\ (\lvert m_0\rvert = \lvert m_1\rvert)$和$\forall\, c \in C$,有 $\Pr[\mathrm{E}(k, m_0] = c] = \Pr[\mathrm{E}(k, m_1) = c]$)， 则该对称加密体制具有完善保密性，其中$k \in K$ 是随机的。

这个定义说明，如果一个对称加密方案满足完善保密性，那么密文不会泄露明文的“任何信息”。攻击者窃听到任意一个密文后，既然明文空间中任意两个明文加密后得到该密文的概率都是一样的，那么攻击者自然不可能知道密文对应哪个明文。

一次一密。香农指出一次一密能满足完善保密性，而且超级简单。一次一密有很多不同的形式，比特串形式的一次一密是指：加密时，明文和密钥逐比特异或即得密文。解密时，密文和密钥逐比特异或即得明文。

有关完美保密性，需要注意的是：

（1）完美保密性定义在信息论安全上，即拥有无限计算资源。

（2）对称加密方案的完善保密性考虑的是唯密文攻击，只说明它在唯密文攻击下是安全的，在其他类型的攻击下未必安全。

（3）“任何信息”指的是明文内容的信息。明文长度、明文是什么时候发送的等信息不包含在内，这些东西即使不破译密码，也可以通过其他手段获知，所以不在密码安全性的考虑范畴之内。因此，定义中附加了一个限制条件：“|m0|=|m1|”，即 m0 和 m1 的长度要相等。

事实上，现在加密原语即使达到了信息论安全，但是在使用过程存在的信息泄露（大小模式、访问行为模式等）已经被证明用来做各种密码破译工作。

（4）一次一密并不实用，最主要的原因就是，为了达到完善保密性，需要的密钥长度不能短于明文长度，当加密长消息时必须使用长密钥。然而，一次一密的优点也是非常明显的，加解密只需要异或操作，软硬件实现都非常简单，运行速度很快。

如何让一次一密更加实用呢？思路很直接：在保证安全性的前提下，可以使用短密钥加密长消息。那么问题又来了，根据香农定理，如果用短密钥加密长消息，就没法达到完善保密性，因为密文势必会泄露明文的信息。明文的信息都泄露了，那又如何保证安全性呢？

我们知道完善保密性可以抵抗无限计算资源的攻击者，但在现实世界中，攻击者的计算资源总是有限的。所以，可以针对有限计算资源的攻击者重新给安全性下一个定义。如果一个加密方案在这个定义下是安全的，那么有限计算资源的攻击者就拿它没办法，这样使用时就足够了。这种加密方案被称为“计算上安全的”。实际应用中也都是使用这种密码方案。因此，解决方法就是：设计一种加密方案，可以用短密钥加密长消息；它在面对实际的攻击者时（计算资源有限），没必要达到完善保密性，密文泄露一点儿明文的信息没关系，只要这些信息对攻击者的帮助是可忽略的就行。

2. 不可区分性

如何定义攻击者的能力是可忽略的呢？不可区分性已经成为一个重要的基础概念，不可区分安全性也已经成为加密算法的重要安全目标。此外，还有一些其他安全目标，比如不可延展性（鉴于本书内容侧重，不做介绍）、存在不可伪造性等。通常关于加密算法，定义安全目标为不可区分性或者不可延展性；对于数字签名算法，定义安全目标为存在不可伪造性；对于密钥交换协议，定义安全目标为通过攻击获得的会话密钥和随机数的不可区分性。

不可区分性。令D_1和D_2为两个以安全参数为索引的概率分布，或表述为D_1和D_2是以安全参数为输入的两个算法。如果对于所有的算法 A，存在一个可忽略函数 υ，满足：

$$\Pr\left[A\left(D_1(n)\right)=1\right]-\Pr\left[A\left(D_2(n)\right)=1\right]\leq \upsilon(n)$$

则称D_1和D_2是不可区分的（Indistinguishable）。换句话说，当以根据D_1或D_2采样得到的样本作为输入时，任何一个算法 A 的执行差异都不会超过可忽略函数。

如果仅考虑非均匀、多项式时间算法 A，则此定义描述的是计算不可区分性。如果考虑所有算法而不考虑算法的计算复杂性，则此定义描述的是统计不可区分性。在统计不可区分性里，两个概率分布的差异上界为两个概率分布的统计距离（Statistical Distance）（也称为总变差距离），定义为：

$$\Delta\left(D_1(n),D_2(n)\right)=\frac{1}{2}\sum_x\left|\Pr[x=D_1(n)]-\Pr\left[x=D_2(n)\right]\right|$$

不可区分安全性定义。将不可区分性的安全目标与敌手能力结合起来评估模型安全性，已经成为目前主要安全性定义方式之一。在后续安全性定义和模型的讨论中，默认是指计算不可区分性。常见的不可区分安全性定义有两个：

（1） **IND-CPA (indistinguishable chosen-plaintext attack)**：一个具有选择明文攻击能力的敌手对模型进行攻击的结果与随机攻击的结果具有“不可区分性”，则称加密算法在“选择明文攻击下”具有不可区分性，记作“IND-CPA”。

（2） **IND-CCA (indistinguishable chosen-ciphertext attack)**：一个具有选择密文攻击能力的敌手对模型进行攻击的结果与随机攻击的结果具有“不可区分性”，则称加密算法在“选择密文攻击下”具有不可区分性，记作“IND-CCA”。

3. 语义安全性

语义安全是继香农提出的完美保密性之后第二个重要的安全性定义，是 Goldwasser 和 Micali 在 1984 年给出的安全性定义：*如果已知某段未知文段的密文不会泄露任何该文段的其余信息，那么则称该密文是语义安全的*。语义安全性意味着密文不会向任何计算能力为多项式时间的敌手泄漏有关相应明文的任何有用信息（假定明文长度不被认为是有用信息），它侧重表示被揭露的信息不会被实际窃取。

后来，Micali 等证明语义安全性和选择明文攻击（CPA）安全性是等价的。

在定义语义安全性时，不能像定义完美性加密那样只给出一个概率公式就完事了，必须得把 CPA 攻击中攻击者“自己选择一些明文”这一活动特点刻画进去。因此，需要借助一个安全模型来帮助给出定义，并通过这些安全模型来证明达到了预期的安全性。

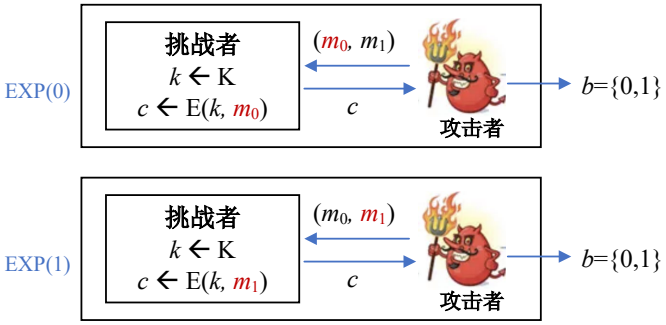


图 1.4.1 语义安全性的安全模型

如图 1.4.1 所示的安全模型里，E 表示一个对称加密体制，有一个挑战者和一个攻击者 A。定义两个黑盒子，每个里面都有一个挑战者，它会随机选择一个密钥，加密输入的最左侧的密文，并将该密文输出。然后，随机选择一个黑盒子，放在 A 面前，A 并不知道自己面对的是哪一个黑盒子：如果 A 面对的是上面那个，称之为实验 EXP(0)；否则，称之为 EXP(1)。A 的任务就是根据得到的密文 c 猜测自己到底是面对实验 EXP(0)还是实验 EXP(1)，并输出自己的猜测 b。

该安全性模型称为 Left-or-Right（LoR）模型，是因为在所设计的实验里，将 m_0 写在左边，m_1 写在右边。详细定义如下：

- ✧ 攻击者 A 随机选择两个消息 m_0 和 m_1，$|m_0|=|m_1|$，将 m_0 和 m_1 发送给挑战者。
- ✧ 挑战者从密钥空间 K 中随机选择一个密钥 k：在实验 EXP(0) 中，挑战者将加密左边的明文 m_0、输出密文 c；在实验 EXP(1)中，挑战者将加密右边的明文 m_1、输出密文 c。
- ✧ 攻击者 A 根据获得的密文 c 猜测自己面对哪个实验，并输出自己的猜测，记为 b。比如，b=0 表示攻击者 A 猜测面对的实验是 EXP(0)，加密的消息是 m_0。

注意，在这个对称密码语义安全性的 LoR 模型中，攻击者 A 只能给挑战者发送一次明文，也即只允许 A 询问一次。换句话说，挑战者选择的密钥只使用了一次。

很明显，从攻击者的角度，它能区分面对的是哪个实验只能依靠得到的密文 c，只要加密方案足够好，攻击者就无法从密文 c 获取有用信息来区分两个实验。这就是为什么这两个实验能刻画加密方案安全性的原因。另外，m_0 和 m_1 都是攻击者自己选择的（攻击者愿意选择什么样的明文都可以），这也刻画了选择明文这一特性。

为了定义攻击者的优势，设 W_0 表示事件攻击者 A 在面对 EXP(0)时返回 1，设 W_1 表示事件攻击者 A 在面对 EXP(1) 时返回 1，攻击者 A 的优势定义为：$Adv=|Pr[W_0]-Pr[W_1]|$，即 $Adv=|Pr[A(EXP(0))=1]-Pr[A(EXP(1))=1]|$。

基于这个安全模型和攻击者 A 的优势，正式定义语义安全性如下：

定义（语义安全性） 设 E 是一个对称加密方案，如果所有高效攻击者 A 在语义安全性的安全模型中的优势都是可忽略的，则 E 是语义安全的。

如果该方案达到了语义安全，可以这么理解，攻击者 A 总是输出 b=1，即猜测密文 c 来自于实验 EXP(1)，在这个情况下 EXP(0)和 EXP(1)仍然是等价的。或者说，不管这个攻击者

面对 EXP(0)还是 EXP(1)，他输出相同猜测结果（因为 W_0 和 W_1 都只关注攻击者返回 1 的情况）的概率是相等的。

概率加密。1984 年，Micali 等提出概率加密的概念，用于设计公钥密码体制。概率加密指的是，每次加密时都使用独立的随机数用于加密过程，保证了即使用同一个公钥对同一个明文加密所得密文也是以很大概率不相同的。因此，概率加密是实现 IND-CPA 安全目标不可缺少的条件。

Goldwasser 和 Micali 利用概率加密的思想给出了第一个满足 IND-CPA 安全的公钥加密方案（简称 GM 方案）。自此，IND-CPA 安全的公钥加密方案得到了快速的发展。各种满足不同困难假设的 IND-CPA 安全的公钥加密方案相继被提出，其中最经典的包括 ElGamal 方案、Paillier 方案以及 Damgård-Jurik 方案等。回顾 RSA 方案和 ElGamal 方案，很明显发现 RSA 方案设计的时候并没有引入概率加密的概念，在此背景下，相同密钥对同一明文加密的密文是相同的，并没有达到 IND-CPA 安全性。RSA 加密方案虽然开创了公钥密码理论的先河，但是没有给出明确的安全定义，也缺乏严格的安全性证明，从而导致了后来可能出现的短私钥攻击等问题。

2.4.4 安全性证明

接下来，以流密码来演示语义安全性的安全模型定义、安全性定义和安全性证明。

流密码基于一次一密基础做了改造，引入伪随机生成器（PRG，pseudo-random generator），它能够利用短密钥产生长密钥，然后用这个长密钥再和明文异或，进一步得到密文。

流密码。定义 G: K $\longrightarrow$ $\{0,1\}^n$ 为伪随机函数发生器，加密过程为 $E(k, m) = G(k) \oplus m$; 解密过程为 $D(k, c) = G(k) \oplus c$。很明显，这里使用 PRG 的输出 G(k)代替了原来的随机密钥。这种改造后的一次一密被称为流密码（stream cipher）。可以看出，k 在 PRG 中是种子，而在流密码中作为密钥使用。流密码本质上是简单套用 PRG 构造而成的，很明显，流密码的安全性必然依赖于 PRG 的安全性。

根据前面的介绍，流密码的安全性可以这样来理解：只要 PRG 的输出能以假乱真，也即能够和等长的随机序列不可区分，那么流密码加密出的密文也能以假乱真，也即和一次一密加密出的密文是不可区分的。这就是流密码为什么是安全的依据。

定理：如果 G: K $\longrightarrow$ $\{0,1\}^n$ 是一个安全的 PRG，由它构造的流密码便具有语义安全性，其中 K=$\{0,1\}^s$。

证明：令 E 表示一个对称加密体制，G 表示一个**安全的**伪随机生成器，有一个挑战者和一个攻击者 A，定义如下两个实验 EXP(0)和 EXP(1)：

- ✧ EXP(0)：攻击者 A 随机选择两个消息 m_0 和 m_1，$|m_0|=|m_1|$，将 m_0 和 m_1 发送给挑战者；挑战者从 K 中随机选择一个种子 k，计算 G(k) $\oplus$ m_0，输出左明文的密文 c。

✧ EXP(1)：攻击者 A 随机选择两个消息 m_0 和 m_1，$|m_0|=|m_1|$，将 m_0 和 m_1 发送给挑战者；挑战者从 K 中随机选择一个种子 k，计算 $G(k) \oplus m_1$，输出右明文的密文 c。

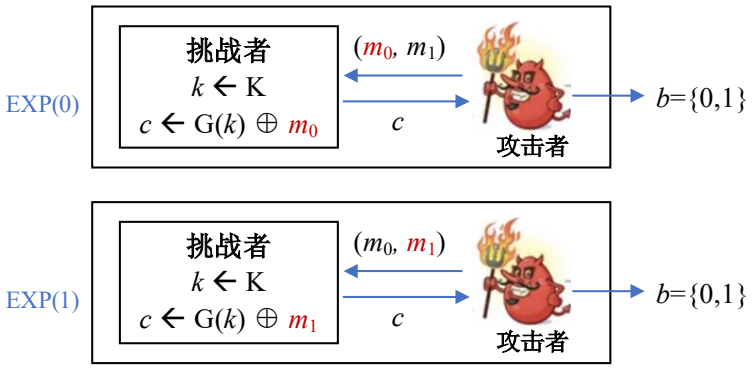


图 1.4.2 证明流密码语义安全的安全模型

如上图所示，设 W_0 表示事件攻击者 A 面对 EXP(0)中返回 1，设 W_1 表示事件攻击者 A 面对 EXP(1)中返回 1，攻击者 A 的优势定义为：$Adv=|Pr[W_0]- Pr[W_1]|$。如果 A 的优势是可忽略的，则 E 是语义安全的。

证明思路。直接证明 EXP(0)和 EXP(1)是不可区分的不好弄，所以采用间接的证明方法。间接的证明方法是这样的：要证明 EXP(0)和 EXP(1)是不可区分的，先把 EXP(0)这个实验进行一系列改造，逐步改造成 EXP(1)。每次改造都会产生一个新的中间实验，需要证明改造前的实验和改造后的实验室是不可区分的。最后利用“计算上不可区分”的性质：传递性，推导出 EXP(0)和 EXP(1)是不可区分的。

为完成证明，进一步定义如下实验：

✧ 实验 EXP(0.1)：攻击者 A 随机选择两个消息 m_0 和 m_1，$|m_0|=|m_1|$，将 m_0 和 m_1 发送给挑战者；挑战者从 $\{0,1\}^n$ 中随机选择一个随机数 r，计算 $r \oplus m_0$，输出左明文的密文 c。

✧ 实验 EXP(0.2)：攻击者 A 随机选择两个消息 m_0 和 m_1，$|m_0|=|m_1|$，将 m_0 和 m_1 发送给挑战者；挑战者从 $\{0,1\}^n$ 中随机选择一个随机数 r，计算 $r \oplus m_1$，输出右明文的密文 c。

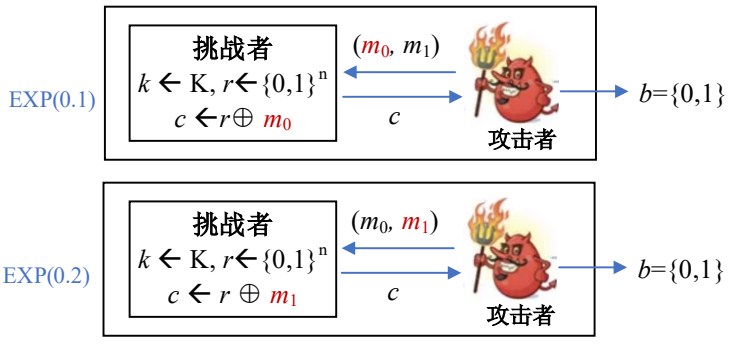


图 1.4.3 相关实验

也就是说，一共定义了四个实验：EXP(0)、EXP(0.1)、EXP(0.2)、EXP(1)。

引理 1：如果 G 是安全的 PRG，则 EXP(0)和 EXP(0.1)是计算上不可区分的。

证明思路。EXP(0) 和 EXP(0.1)不可区分的依据是 G 必须是安全的 PRG，也即 G(k)与随机序列必须计算上不可区分。不难看出，要证明的是：如果 G 是安全的 PRG，则 EXP(0)和 EXP(0.1)是计算上不可区分的。反证法的实质就是通过前提假设“EXP(0)和 EXP(0.1)是可以区分的”，以推出矛盾的结论“G 不是安全的 PRG”。相当于，有一个 PRG 安全性模型里的挑战者 C，它给了一串序列 r。r 可能等于 G(k)，也可能等于随机序列。你需要想办法设计一个高效攻击者，记为 B，它能以不可忽略的优势将之区分开。

怎么做呢？幸运的是，根据前提假设，EXP(0)和 EXP(0.1)是可以区分的，那必然存在一个高效攻击者 A 能以不可忽略的优势区分 EXP(0)和 EXP(0.1)。你需要做的就是把 A 作为子程序，让 B 去调用 A，以利用 A 区分 EXP(0)和 EXP(0.1) 的能力，去识别 r 到底是 G(k) 还是随机序列。如何设计 B 就是反证法的核心，证明的实质就是写出 B 的具体执行过程，就像写算法的伪代码一样。证明的最后还需要算出 B 的优势和时间复杂度：B 的优势必须是不可忽略的，同时 B 必须是高效的。既然存在一个高效攻击者 B，它能以不可忽略的优势区分 G(k)和随机序列，就说明 G 不是安全的 PRG，这正好与题设中“G 是安全的 PRG”相矛盾，由此证明完毕。

证明：假设存在一个概率多项式时间的攻击者 A，它能以不可忽略的优势 Adv_A 区分 EXP(0)和 EXP(0.1)。

挑战者 C 给出一个长度是 n 的比特序列 r。设计一个概率多项式时间的算法 B，它的任务是通过调用 A 为子程序，在接收到 r 以后，能以不可忽略的优势 Adv_B 识别出 r 是 G(k)还是随机序列。

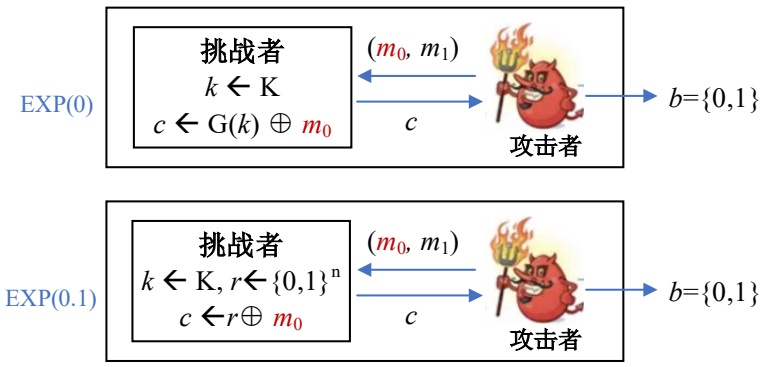


图 1.4.4 相关实验

在收到挑战者 C 发来的 r 以后，算法 B 的执行过程如下：

1) 收到 A 发来的 m_0 和 m_1 后，**计算 $c = r \oplus m_0$**，并将 c 返回给 A 作为应答。

很明显，当 r=G(k)时，c = G(k)$\oplus m_0$，A 处于 EXP(0)；当 r 是随机序列时，c= $\{0,1\}^n \oplus m_0$，A 处于 EXP(0.1)。

2) A **输出** b$\in${0,1} 作为自己的**猜测**。如果 b=0，表示 A 猜测自己处于 EXP(0)；否则（b=1），表示 A 猜测自己处于 EXP(0.1)。

3) B 将 b 直接返回给 C 作为自己的猜测。如果 b=0，表示 B 猜测 r=G(k)；否则（b=1），

表示 B 猜测 r 是随机序列。

很明显，因为 B 仅仅是做了一个异或运算，其余都是对 A 的调用，所以 $Adv_A = Adv_B$，且 B 的时间复杂度等于 A 的时间复杂度。因为假定 A 是一个可以区分两个实验的攻击者，所以 B 同样能在多项式时间内以不可忽略的优势区分 G(k)和随机序列。这与 G 是安全的 PRG 相矛盾。得证！

引理 2：EXP(0.1)和 EXP(0.2)是计算上不可区分的。

证明。因为两个实验都是利用随机数 r 异或左明文或者右明文，本质就是一次一密。根据一次一密的完善保密性，任何攻击者区分 EXP(0.1)和 EXP(0.2)的优势都等于零。

引理 3：如果 G 是安全的 PRG，则 EXP(0.2)和 EXP(1)是计算上不可区分的。

证明。参考引理 1，利用反证法证明。

归纳总结。考虑到

(1) EXP(0)和 EXP(0.1)之间是计算上不可区分的。

(2) EXP(0.1)和 EXP(0.2)之间是计算上不可区分的。

(3) EXP(0.2)和 EXP(1)之间是计算上不可区分的。

根据(1)、(2)和(3)以及传递性可知，EXP(0)和 EXP(1)是计算上不可区分的。

所以，任意高效攻击者区分 EXP(0)和 EXP(1)的优势都是可忽略的。

证毕！

2.4.5 安全性模型

1. 语义安全性模型

常用的语义安全性模型有三类：LOR 模型、比特随机猜测模型和 Real-Random 模型。这些模型之间都是等价的。也就是说，一个加密方案在其中一个模型下证明是安全的，它在另一个模型下肯定也能证明是安全的。

前面介绍的语义安全的安全性模式是 LoR 模型，即 Left-or-Right 模型，即攻击者猜测是对左侧密文还是右侧密文的加密的一种攻击者能力的定义模型。LoR 模型还有一种完全对应的比特随机猜测模型。比特随机猜测模型更为常用。

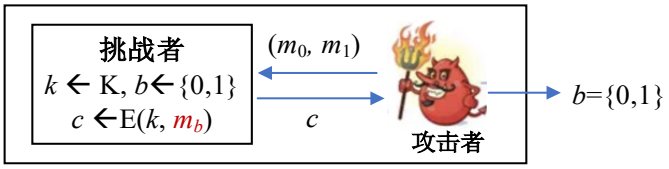


图 1.4.5 比特随机猜测模型

比特随机猜测模型。这个模型里只有一个实验。挑战者随机选择一个比特，记为 b。在收到攻击者发来的两个消息后，挑战者用随机产生的密钥 k 加密消息 m_b，产生密文 c。

很明显，b=0 时，c 对应的是消息 m_0；b=1 时，c 对应的是消息 m_1。

所以，攻击者收到的密文 c 里包含的不是 m_0 就是 m_1，他需要猜测挑战者选择的 b 到底

等于 0 还是等于 1，并输出自己的猜测，记为 b'。

如果 b'=b 就说明攻击者猜对了，攻击成功；否则就说明攻击者猜错了，攻击失败。

在这个安全性模型中，攻击者的优势定义为 |Pr[b=b']-1/2|，很容易理解，攻击者一直猜测相同结果，在随机猜测的情况下概率是 1/2。如果不比 1/2 多，那么意味着攻击者成功的概率是可忽略的。

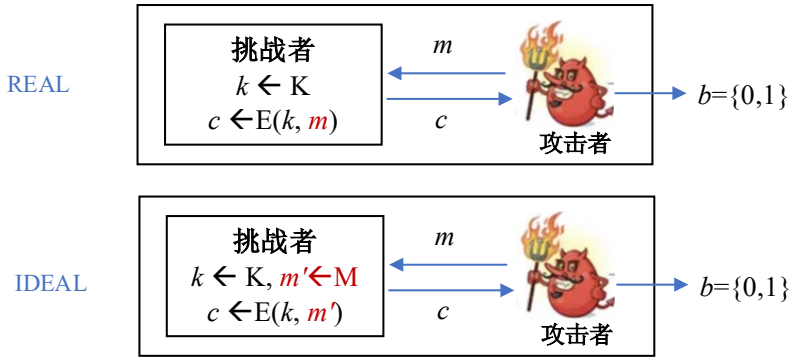


图 1.4.6 Real-Random 模型

Real-Random 模型。在这个模型里，攻击者只给挑战者发一个消息 m，挑战者收到 m 以后，会随机决定到底是直接加密 m 还是再产生一个等长的随机 m 进行加密。这个随机产生的消息记为 m'。之后，挑战者用随机产生的密钥 k 加密 m 或者 m'，然后将密文发送给攻击者。也就是说，攻击者收到的密文可能是他发送的明文 m 的密文，也可能是攻击者随机选择的 m'的密文。因此，攻击者处于两个不同的实验中，当密文是明文 m 的加密的时候，它处于的是 Real 实验，当密文是随机消息 m'的加密的时候，它处于的是 IDeal 实验。

注意，这里称为 IDeal 实验而不是随机实验。

最后，攻击者输出 b，如果 b=0，则表示攻击者猜测处于 Real 实验；否则，处于 Ideal 实验。

2. 随机预言机模型

如前所述，计算安全将攻击者的能力限定为多项式时间，一个方案是否计算安全取决于攻击者成功的优势能否规约到以不可忽略的概率解决某个已知困难问题（例如大整数分解、离散对数等），或者所有高效攻击者 A 在相应的安全模型中的优势都是可忽略的。

在公钥密码学研究中，早期基于计算安全的密码方案一般是基于标准模型（Standard Model）下，该模型首先对方案中攻击者的能力加以定义，而且必须强调攻击者一定是自适应性的。然后假设该攻击者成功的概率为某个多项式时间不可忽略的值，然后通过一定的步骤利用该攻击者，将攻击者的能力转化为攻破某已知困难问题的优势。由于该困难问题在多项式时间下无法求解，因而可以得出存在攻击者以不可忽略概率攻破方案这一假设与事实相矛盾。标准模型下的密码方案其实就是在不借助任何假想模型下所设计的方案，没有任何安全证明环节上的假想存在，仅仅建立在一些被广泛接受的假设基础上，安全性值得高度信赖。不幸的是，基于标准模型的密码方案往往需要大量的计算，难以实用。如现有最高效的标准模型下安全的公钥加密方案、数字签名方案，仍然没有广泛加以使用。

如何设计一个面向具体应用的密码方案，同时平衡可证明安全性和实用性，成为了密码方案设计中首要考虑的问题。因为一个低效率的方案与不安全的方案一样，都无法在实际当中被广大用户接受并广泛使用。

随机预言机（Random Oracle，RO）是描述哈希函数安全性的一个启发式模型，由 Bellare 和 Rogaway 在 1993 年首先提出。随机预言机的基本思想是将哈希函数看作公开的理想随机函数。在随机预言机模型中，所有参与方都可以访问用状态预言机实现的公开函数$H:\{0,1\}^* \to \{0,1\}^k$。给定字符串$x \in \{0,1\}^k$，$H$ 查找自身的调用记录。如果之前从未调用过 $H(x)$，则 H 随机选择$r_x \in \{0,1\}^k$，记录输入/输出对(x, r_x)，并返回r_x。如果之前调用过 $H(x)$，则 H 返回r_x。预言机通过这一方式实现了一个随机选择函数$\{0,1\}^* \to \{0,1\}^k$。

随机预言机模型是一个启发式模型，因为此模型只能覆盖将哈希函数 H 视为黑盒的攻击算法。随机预言机模型将公开函数（例如像 SHA-256 这样的标准哈希函数）视为固有随机对象，但现实中不存在这样的公开函数。事实上，可以构造出在随机预言机模型下安全，但当 H 被任意具体函数实例化后便不安全的方案（Canetti et al., 1998）。尽管有这些缺点，但实际应用中通常都可以接受随机预言机模型。如果能假设存在随机预言机，一般都可以设计出更高效的方案。

2.5 通用可组合安全*

前面介绍的安全性定义和安全证明过程，主要是面向简单密码协议，交互次数少、形式较为单一。但是，目前很多安全协议设计复杂、参与方多、存在多个协议并行运行的情况。

比如，在多方参与安全计算的环境中，一组参与者，他们之间互不信任，但是他们希望安全地计算一个约定的函数，这个函数的输入由这些参与者提供。每个参与者都能得到正确的计算结果，同时每个参与者的输入是保密的，也就是说一个参与者无法得知另一个参与者的输入，这就是安全多方计算的一个直观的说法。

安全多方计算问题可以用数学形式化如下：n 个协议参与者$\{P_1, \ldots, P_n\}$执行某个协议 π，每个协议参与者提供秘密输入x_i，协议 π 计算函数$F(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$，结果协议参与者$P_i$应该得到（并且仅仅得到）他的结果$y_i$，除此以外，他不应该知道任何其它敏感信息，比如其他参与者的输入。如果假设有可信第三方 T 存在，这个问题的解决是十分容易的，参与者只需将自己的输入保密传送给 T，由 T 计算这个函数，然后将计算的结果广播给每一个参与者，这样每个参与者都得到了正确的结果，同时自己的输入也是保密的。然而在现实的应用中，很难找到这样一个所有参与者都信任的 T，因此安全多方计算的研究主要是针对无可信 T 的情况下，如何安全地计算一个约定函数的问题。从另一个技术角度说，安全多方计算协议实际上就是在努力仿真一个有可信 T 的协议，虽然没有可信 T，但是希望达到有可信 T 存在的“效果”。

安全多方计算是许多密码学协议的基础，从广义上讲，所有的密码学协议都是安全多方计算的一个特例，这些密码学协议都可以看作是一组参与者之间存在着各种各样的信任关系(最弱的信任关系就是互不信任)，他们希望通过交互或者非交互的操作来完成一项工作(计算某个约定的函数)。这些协议的不同之处在于协议计算的函数是不一样的。

针对类似复杂协议并行运行或者作为其它协议的子协议时整个系统的安全情况，2001年，Canetti 等提出了通用可组合安全（UC 安全，Universally Composable Security）的概念。UC 安全的最优秀的性质就是一种模块化设计思想：可以单独设计协议，只要协议满足 UC 安全，那么就可以保证和其它协议并行运行的安全。UC 安全框架的核心由三个模型：现实模型、理想模型以及 F-混合模型搭建而成，它的主要证明和技术手段是“仿真”。

1.5.1 基本概念

定义安全性时，很自然的想法是列举一个“安全检查清单”，枚举出哪些情况属于违反安全性要求。这种安全性定义方式不仅非常烦琐，而且很容易出现错误。很难说明“安全检查清单”是否枚举出了所有的安全性要求。

现实-理想范式避免采取这种安全性要求描述方式，而是引入了一个定义明确、涵盖所有安全性要求的“理想世界”，通过论述现实世界与理想世界的关系来定义安全性。

理想世界。在理想世界中，每个参与方 P_i秘密地将自己拥有的私有输入 x_i发送给一个完全可信的参与方 T，由后者来安全地计算函数$F(x_1,\dots,x_n)$ ，并返回结果给所有参与方。通常，F 被称为理想函数（Ideal Functionality），它扮演了一个不可破可信第三方的角色，能完成协议所执行的功能。

虽然很容易理解理想世界的定义，但完全可信第三方的存在使得理想世界只是一个想象中的世界。**我们用理想世界作为判断实际协议安全性的基准**。

现实世界。现实世界中不存在可信参与方。在现实世界中，攻击者可以攻陷参与方。在协议开始执行之前就被攻陷的参与方与原始参与方就是攻击者是等价的。根据威胁模型的定义，攻陷参与方可以遵循协议规则执行协议，也可以偏离协议规则执行协议。

如果攻击者实施攻击后，其在现实世界中达到的攻击效果与其在理想世界中达到的攻击效果相同，则可以认为现实世界中的协议是安全的。换句话说，协议的目标是（在给定一系列假设的条件下）使其在现实世界中提供的安全性与其在理想世界中提供的安全性等价。

1.5.2 半诚实安全性

半诚实（Semi-honest）攻击者可以攻陷参与方，但会遵循协议规则执行协议。换句话说，攻陷参与方会诚实地执行协议，但可能会尝试从其他参与方接收到的消息中尽可能获

得更多的信息。半诚实攻击者也被称为*被动*（Passive）*攻击者*，因为此类攻击者只能通过观察协议执行过程中自己的视角来尝试得到秘密信息，无法采取其他任何攻击行动。半诚实攻击者通常也被称为诚实但好奇（Honest-but-curious）攻击者。

参与方的视角（View）包括其私有输入、协议执行期间收到的所有消息所构成的消息列表等。攻击者的视角包含所有攻陷参与方的混合视角。攻击者从协议执行过程中得到的任何信息都必须能表示为以其视角作为输入的高效可计算函数的输出。

根据现实-理想范式，**为了证明协议是安全的，在理想世界中的攻击者必须能够生成一个视角，此视角与真实世界中的攻击者视角不可区分**。请注意，理想世界中的攻击者视角只包含发送到 T 的输入和从 T 接收到的输出。因此，理想世界中的攻击者必须能够使用这些信息生成一个视角，此视角和真实世界中的攻击者视角看起来一样。因为*攻击者在理想世界中生成了一个真实世界中的"仿真"攻击者视角，所以理想世界的攻击者被称为仿真者*（Simulator）。能说明存在这样一个仿真者，就能证明攻击者在现实世界中实现的所有攻击效果都可以在理想世界中实现。

"现实-理想范式"形式化定义。令 π 为一个协议，F 为一个功能函数。令 C 为攻陷参与方集合，令 Sim 为一个仿真者算法。定义下述两个随机变量的概率分布：

- ✧ $Real_{\pi}(\kappa, C; x_1, \dots, x_n)$：在安全参数 κ 下执行协议，其中每个参与方 P_i 都将使用自己的私有输入 x_i，诚实地执行协议。令 V_i 为参与方 P_i 的最终视角，令 y_i 为参与方 P_i 的最终输出。

  输出$\{V_i | i \in C\}$，$(y_1, \dots, y_n)$。
- ✧ $Ideal_{F,Sim}(\kappa, C; x_1, \dots, x_n)$：计算$(y_1, \dots, y_n) \leftarrow F(x_1, \dots, x_n)$。

  输出$Sim(C, \{(x_i, y_i) | i \in C\})$，$(y_1, \dots, y_n)$。

在上述定义中，所有参与方的输出都包含了进来，包括诚实参与方的输出。如果现实世界中攻陷参与方所拥有的视角和理想世界中攻击者所拥有的视角不可区分，换句话说，协议在现实世界中给出的输出概率分布与理想功能函数给出的输出概率分布相同，那么协议在半诚实攻击者的攻击下是安全的，具体定义如下。

定义 给定协议 π，如果存在一个仿真者 Sim，使得对于攻陷参与方集合 C 的所有子集，对于所有的输入 $x_1, \dots, x_n$，概率分布

$$Real_{\pi}(\kappa, C; x_1, \dots, x_n) \text{ 和 } Ideal_{F,Sim}(\kappa, C; x_1, \dots, x_n)$$

是（在 κ 下）不可区分的，则称此协议**在半诚实攻击者存在的条件下安全地实现了 F**。

初看半诚实攻击模型，会感觉此模型的安全性很弱，简单地读取和分析收到的消息看起来几乎根本就不是一种攻击方法！有理由怀疑是否有必要考虑如此受限的攻击模型。实际上，构造半诚实安全的协议并非易事。更重要的是，在构造更复杂环境下可抵御更强大攻击者攻击的协议时，一般都在半诚实安全协议的基础之上进行改进。此外，很多现实场景确实可以与半诚实攻击模型相对应。一种典型的应用场景是，参与方在计算过程中的行为是可信的，但是无法保证参与方的存储环境在未来一定不会遭到攻击。

仿真证明。接下来将详细描述最关键的技术手段，也就是如何利用仿真建立起现实模型和理想模型之间的桥梁，将现实模型的安全规约到理想模型的安全。

假设协议 A 和协议 B 完成同样的功能，如果攻击者攻击协议 A 不能比攻击协议 B 获得更多的信息或者影响，那么协议 A 至少和协议 B 一样安全。如果攻击者攻击现实模型下的一个协议 π，不比攻击理想模型下的一个 Ideal Functionality F 获得更大的影响或者更多的信息，那么 π 至少和 F 一样安全。

以上的说法形式化地表述为：如果任何现实模型攻击者 A 都存在一个理想模型攻击者 S（仿真器），对于任何输入，在现实模型下运行包含 A 的协议π的全局输出，它和在理想模型下运行包含 S 的 F 的全局输出是不可区分的，那么π至少和 F 一样安全。

这里的仿真是这样完成的，S 在得到 F 后，它就仿真一个虚拟的现实协议所执行的每个动作，它必须能够完成现实模型攻击者 A 的所有动作（比如收买参与者），以及看到所有 A 能看到的信息。因为在理想模型下不存在现实的协议 π，它只有一个可信第三方 F，所以协议中参与者所有的交互和输出都需要虚构（仿真），如果这些虚构（仿真）的信息和现实模型下真实的协议 π 不可区分，那么仿真就成功了，那么也就将现实模型的安全规约到了理想模型的安全。

例子。这里以 Diffie-Hellman 协议为例，且假设攻击者是被动的，也就是仅仅能窃听协议。Diffie-Hellman 协议执行情况如下：共同输入为 p 为一个大素数，g 为F_p^*的生成元；输出为 Alice 和 Bob 共享的域F_p^*的一个元素。协议执行过程为：

- Alice 随机选择 $a \in [1, p-1)$，发送 $g_a = g^a \bmod p$ 给 Bob；
- Bob 随机选择 $b \in [1, p-1)$，发送 $g_b = g^b \bmod p$ 给 Alice；
- Alice 计算 $k = {g_b}^a \bmod p$；
- Bob 计算 $k = {g_a}^b \bmod p$。

定义 Diffie-Hellman 协议的 **Ideal Functionality** F_{DH}的描述如下：Alice 和 Bob 为了得到一个共享密钥，它们向一个F_{DH}发送自己的身份标识后，F_{DH}从F_p^*随机选取一个安全的整数 k 发送给 Alice 和 Bob 作为 Alice 和 Bob 之间的共享密钥，从而完成了现实模型下的 Diffie-Hellman 协议所需要的功能。

仿真器S_{DH}的仿真过程如下：

- S_{DH}仿真 Alice 的动作，随机选择 $a' \in [1, p-1)$，发送$g_{a'} = g^{a'} \bmod p$ 给虚构的 Bob；
- S_{DH}仿真 Bob 的动作，随机选择 $b' \in [1, p-1)$，发送$g_{b'} = g^{b'} \bmod p$ 给虚构的 Alice；
- 虚构的 Alice 计算 $k' = g_{b'}^{a'} \bmod p$（k' 也就是F_{DH}的输出）；
- 虚构的 Bob 计算 $k' = g_{a'}^{b'} \bmod p$。

显然这里仿真的虚假消息$a', b', g_{a'}, g_{b'}, k'$(全局输出)和真实的$a, b, g_a, g_b, k'$是不可区分的，两者都是落在同一区间的均匀分布。于是可以得出，对于被动攻击者而言 Diffie-Hellman 协议是安全的。

本例的仿真是很自然和简单的，但是实际上在很多情况下仿真是很困难的，这是一种构造式的证明，仿真的成败与否直接决定了所设计协议的安全与否。

1.5.3 恶意安全性

恶意（Malicious）攻击者，或称*主动*（Active）*攻击者*，可以让攻陷参与方任意偏离协议规则执行协议，以破坏协议的安全性。恶意攻击者分析协议执行过程的能力与半诚实攻击者相同，但恶意攻击者可以在协议执行期间采取任意行动。请注意，这意味着攻击者可以控制或操作网络，或在网络中注入任意消息。与之前类似，恶意攻击者场景下的安全性也将通过比较理想世界和现实世界的差异来定义，但需要考虑两个重要的附加因素。

- ✧ **对诚实参与方输出的影响**。攻陷参与方偏离协议规则执行协议，可能会对诚实参与方的输出造成影响。例如，攻击者的攻击行为可能会使两个诚实参与方得到不同的输出，但在理想世界中，所有参与方都应该得到相同的输出。此外，不能也不应该相信恶意攻击者会给出最终的输出，因为恶意参与方可以输出任何想输出的结果。
- ✧ **输入提取**。由于诚实参与方会遵循协议规则执行协议，因此可以明确定义诚实参与方的输入，并在理想世界中将此输入提供给 *T*。相反，在现实世界中无法明确定义恶意参与方的输入，这意味着在理想世界中我们需要知道将哪个输入提供给 *T*。直观上看，对于一个安全的协议，无论攻击者在现实世界中实施何种攻击行为，都可以通过为攻陷参与方选择适当的输入在理想世界中模拟实现。因此，我们让仿真者选择攻陷参与方的输入。这样的仿真过程称为输入提取，因为仿真者要从现实世界的攻击者行为中提取出有效的理想世界输入，来“解释”此输入对现实世界造成的影响。大多数安全性证明只需考虑黑盒仿真过程，即仿真者只能访问现实世界中实现攻击的预言机，不能访问攻击代码本身。

形式化定义。用 *A* 表示攻击者，用 corrupt(*A*)表示被现实世界中的攻击者 *A* 攻陷的参与方集合，用 corrupt(Sim)表示被理想世界中的攻击者 Sim 攻陷的参与方集合。与定义半诚实安全性的方式类似，定义现实世界和理想世界的概率分布，并定义一个安全协议，使这两个概率分布满足不可区分性：

- ✧ $Real_{\pi,A}(\kappa;\{x_i|i \notin corrupt(A)\})$：在安全参数 κ 下执行协议，其中每个诚实参与方 P_i使用给定的私有输入 x_i 诚实地执行协议，而攻陷参与方的消息将由 *A* 选取。令 y_i表示每个诚实参与方 P_i 的输出，令 V_i 表示参与方 P_i 的最终视角。输出 $(\{V_i|i \in corrupt(A)\},\{y_i|i \notin corrupt(A)\})$。
- ✧ $Ideal_{F,Sim}(\kappa;\{x_i|i \notin corrupt(A)\})$：执行 Sim，直至其输出一个输入集合 $\{x_i|i \in corrupt(A)\}$。计算$(y_1,\dots,y_n) \leftarrow F(x_1,\dots,x_n)$。随后，将 $\{y_i|i \in corrupt(A)\}$ 发送给 Sim。令 V^* 表示 Sim 的最终输出（输出是参与方的仿真视角集合）。输出$(V^*,\{y_i|i \notin corrupt(Sim)\})$。

定义 给定协议 π，如果对于任意一个现实世界中的攻击者 A，存在一个满足 $corrupt(A) = corrupt(Sim)$ 的仿真者 Sim，使得对于诚实参与方的所有输入 $\{x_i | i \notin corrupt(A)\}$，概率分布

$$Real_{\pi,A}(\kappa; \{x_i | i \notin corrupt(A)\})$$

和

$$Ideal_{F,Sim}(\kappa; \{x_i | i \notin corrupt(Sim)\})$$

是（在 κ 下）不可区分的，则称此协议**在恶意攻击者存在的条件下安全地实现了 F**。

需要注意的是，该定义仅描述了诚实参与方的输入 $\{x_i | i \notin corrupt(A)\}$。攻陷参与方与现实世界 Real 交互时不需要提供任何输入。而在与理想世界 Sim 交互时，攻陷参与方的输入是间接确定的（仿真者需要根据攻陷参与方的行为来选择将何种输入发送给 F）。虽然也可以在现实世界定义攻陷参与方的输入，但此输入仅仅是一个“建议”，因为攻陷参与方可以在执行协议时选择使用任何其他的输入（甚至使用与真实输入不一致的输入执行协议）。

交互功能函数。在理想世界中，功能函数仅包含一轮交互过程：提供输入，给出输出。可以进一步扩展 F 的行为方式，令 F 与参与方进行多轮交互，且在多轮交互的过程中保持其内部状态的私有性。我们称此类功能函数为交互功能函数（Reactive Functionality）。

交互功能函数的一个实例是扑克游戏中的发牌方。此功能函数必须追踪所有扑克牌的状态，获取输入命令，并通过多轮交互向所有参与方提供输出。

另一个交互功能函数实例是承诺（Commitment），这是一个非常常见的功能函数。此功能函数从 P_1 处接收一个比特值 b（更一般的情况是接收一个字符串），告知 P_2 已“承诺”b，并在内部记住 b。稍后，如果 P_1向该功能函数发送命令“披露”（或“打开”），此功能函数将 b 发送给 P_2。

可中止安全性。在几乎所有基于消息的 2PC 协议中，一个参与方会在另一个参与方之前得到最终的输出。如果此参与方是恶意的攻陷参与方，它可以简单地拒绝将最后一条消息发送给诚实参与方，从而阻止诚实参与方得到输出。然而，这种攻击行为与我们之前描述的理想世界攻击行为不兼容。在理想世界中，如果攻陷参与方可以从功能函数中得到输出，则所有参与方均可以得到输出。此性质称为输出公平性（Output Fairness）。并非所有的功能 函数在计算过程都可以满足输出公平性。

为在恶意攻击场景下覆盖此攻击行为，学者们提出了一种稍弱的安全性定义，称为可中止安全性（Security with Abort）。为此，需要按照下述方式稍微修改一下理想功能函数。首先，允许功能函数得知攻陷参与方的身份。其次，修改后的功能函数需要一些交互能力：当所有参与方提供输入后，功能函数计算输出结果，但只将输出结果交付给攻陷参与方。随后，功能函数等待来自攻陷参与方的“交付”或“中止”命令。收到“交付”命令

后，功能函数将输出交付给所有诚实参与方。收到“中止”命令后，功能函数向所有诚实参与方交付一个表示协议中止的输出（⊥）。

在修改后的理想世界中，攻击者允许在诚实参与方之前得到输出，同时可以阻止诚实参与方接收任何输出。需要特别注意此定义的一个关键点：诚实参与方是否中止协议只能依赖于攻陷参与方的命令。特别地，如果诚实参与方中止协议的概率依赖于诚实参与方的输入，则协议可能是不安全的。

在描述功能函数时，一般不会明确写出此功能函数可能会让诚实参与方无法得到输出。反之，当讨论协议在恶意攻击者攻击下的安全性时，通常会认为攻击者可以决定是否向诚实参与方交付输出，在此场景下不要期望协议可以满足输出公平性。

适应性攻陷。在已经定义的现实世界和理想世界中，如果攻陷参与方在整个交互过程中是固定不变的，则称这一安全模型的协议在静态性攻陷（Static Corruption）下是安全的。相反的，如果攻击者在协议执行期间可以根据交互过程中得到的信息选择攻陷哪些参与方，则称这一攻击行为是适应性攻陷（Adaptive Corruption）。

可以在现实-理想范式中为适应性攻陷攻击行为建立安全模型，方法是允许攻击者发出形式为“攻陷 P_i”的命令。在现实世界中，这会令攻击者得到 P_i 的当前视角（包括 P_i 的内部私有随机状态），并接管其在协议执行过程中发送消息的控制权。在理想世界中，仿真者只能得到攻陷此参与方时该参与方的输入和输出，必须使用这些信息生成仿真视角。显然，各个参与方的视角是相互关联的（如果 P_i 向 P_j 发送一条消息，则此消息会同时包含在两个参与方的视角中）。适应性安全的挑战是仿真者必须逐段生成攻陷参与方的视角。例如，当参与方 P_i 被攻陷时，我们要求仿真者生成 P_i 的视角。仿真者必须在未知 P_j 私有输入的条件下仿真出 P_j 发送给 P_i 的所有消息。随后，仿真者可能需要提供 P_j 的视角（包括 P_j 的内部私有随机状态）来“解释”之前发送的协议消息与 P_j 的私有输入是匹配的。

1.5.4 组合性

出于模块化考虑，设计协议时经常会让协议调用其他的理想功能函数。例如，要设计一个安全实现某功能函数 F 的协议 π，在 π 中，参与方除了彼此要发送消息之外，还需要与另一个功能函数 G 交互。因此，该协议在现实世界中包含 G，但在理想世界（一般来说）仅包含 F。我们称这一修改后的现实世界为 G-混合世界。

对安全模型的一个很自然的要求是组合性（Composition）：如果 π 是一个安全实现 F 的 G-混合协议（即 π 的参与方需要彼此发送消息，且需要与一个理想的 G 交互），且 ρ 是一个安全实现 G 的协议，则以最直接的方式组合使用 π 和 ρ（将调用 G 替换为调用 ρ）应该可以得到安全实现 F 的协议。一个组合性的例子是基于理想认证消息传输功能函数 F_auth 实现的 DH 密钥交换协议 π_DH。组合性要求：如果 π_DH 在给定 F_auth 的情况下安全地实现了理想 DH 密钥交换功能函数 F_DH，那么将 π_DH 中所有对 F_auth 的调用替

换为对某个安全实现了 F_auth 的子协议 π_auth 的调用后得到的 π'_DH 依然安全地实现了 F_DH。

需要指出，满足基于模拟证明（参考前述半诚实安全性和恶意安全性）的协议自然满足“孤立环境”（stand-alone setting）下的可组合性。所谓孤立环境是指当前环境中有且仅有一个该协议的实例在运行，而不存在该协议的其他实例。这种环境下的可组合性又被称为串行组合性（Sequential composition）。然而，在现实环境下，协议是有可能同时存在多个实例的，而这种环境下协议的可组合性称为并行组合性（Concurrent composition）。Canetti 已经指出，一些满足串行组合性的协议在并行组合的情况下是不安全的。

保证并行组合性的一种方法是使用 2001 年 Canetti 提出的通用可组合性（Universal Composability，UC）框架。UC 框架在之前描述的安全模型上进行了扩展，在安全模型中增加了一个称为环境（Environment）的实体，此实体也同时包含在理想世界和现实世界中。引入环境实体的目的是体现协议执行时的“上下文”（例如，当前协议被某个更大的协议所调用）。环境实体为诚实参与方选择输入，接收诚实参与方的输出。环境实体可以与攻击者进行任意交互。

现实世界和理想世界都包含相同的环境实体，而环境实体的“目标”是判断自身是在现实世界还是在理想世界中被实例化的。在此之前，我们定义安全性的方式是要求现实世界和真实世界中的特定视角满足不可区分性。在 UC 场景下，我们还可以将区分两种视角的攻击者吸收到环境实体之中。因此，不失一般性，环境实体的最终输出是一个单比特值，表示环境实体“猜测”自身是在现实世界还是在理想世界被实例化的。

接下来，我们定义现实世界和理想世界的协议执行过程，其中 Z 是一个环境实体：

- $Real_{\pi,A,Z}(\kappa)$：执行涉及攻击者 A 和环境 Z 的协议交互过程。当 Z 为某一诚实参与方生成一个输入时，此诚实参与方执行协议 π，并将输出发送给 Z。最后，Z 输出一个单比特值，作为 $Real_{\pi,A,Z}(\kappa)$的输出。
- $Ideal_{F,Sim,Z}(\kappa)$：执行涉及攻击者（仿真者）Sim 和环境 Z 的协议交互过程。当 Z 为某一诚实参与方生成一个输入时，此输入将被直接转发给功能函数 F，F 将相应的输出发送给 Z（F 完成了诚实参与方的行为）。Z 输出一个单比特值，作为 $Ideal_{F,Sim,Z}(\kappa)$的输出。

定义 给定协议 π，如果对于所有现实世界中的攻击者 A，存在一个满足 $corrupt(A) = corrupt(Sim)$的仿真者 Sim，使得对于所有的环境实体 Z：

$$|\Pr[Real_{\pi,A,Z}(\kappa) = 1] - \Pr[Ideal_{F,Sim,Z}(\kappa) = 1]|$$

是（κ 下）可忽略的，则称此协议 **UC-安全地实现了 F**。

由于定义中要求不可区分性对所有可能的环境实体都成立，因此一般会把攻击者 A 的攻击行为也吸收到环境 Z 中，只留下所谓的“无作为攻击者”（此攻击者只会简单地按照 Z 的指示转发协议消息）。

在其他（非 UC 可组合的）安全模型中，理想世界中的攻击者（仿真者）可以随意利用现实世界中的攻击者。特别地，仿真者可以在内部运行攻击者，并反复将攻击者的内部状态倒带成先前的内部状态。可以在这类较弱的模型下证明很多协议的安全性，但组合性可能会对仿真者的部分能力进行一些约束和限制。

在 UC 模型中，仿真者无法倒带攻击者的内部状态，因为攻击者的攻击行为可能会被吸收到环境实体之中，而仿真者不允许利用环境实体完成仿真过程。相反，仿真者必须是一个直线仿真者（Straight-line Simulator）：一旦环境实体希望发送一条消息，仿真者必须立刻用仿真出的回复做出应答。直线仿真者必须一次性生成仿真消息，而先前的安全模型定义没有对仿真消息或视角生成过程做出任何限制。