



# 密码科学技术丛书

## 公钥加密的设计方法

作者：陈宇 & 秦宝东

组织：山东大学 & 西安邮电大学

版本：0.1

微言大义 高屋建瓴

# 前言

本书是在“2017-2019 中国科学院大学研究生暑期课程”和“2022 北京大学应用数学专题讲习班”的讲义基础上,结合多年在公钥加密方面的研究成果编写而成。目的是尽快引导读者到达公钥加密这一极重要的现代密码学领域,力求从高屋建瓴的视角介绍公钥加密的设计方法,对重要的思想剥丝抽茧、对关键的技术条分缕析。写作过程中根据教学和研究经历感悟对内容做了取舍和精简,参阅了国际顶级会议期刊的前沿论文,也融入了作者的独立思考。

计算机网络技术的飞速发展引发了人类社会组织形态的根本性变革,从集中式迁移为分布式,“海内存知己,天涯若比邻”从诗歌意象走进现实世界。面向分布式环境下的隐私保护需求,1976 年 Diffie 和 Hellman 开创了现代密码学的新方向—公钥密码学。迄今为止的半个多世纪以来,公钥密码学一直处于最活跃的前沿,引领驱动了密码学的研究进展,极大丰富了密码学的学科内涵。公钥加密作为公钥密码学最重要的分支,在理论方面孕育了可证明安全方法、将各类数学困难问题纳入工具库、启发了一系列密码原语和重要概念,已有多项历史性成果获得 Turing 奖和 Gödel 奖;在应用方面则是各类网络通信安全协议的核心组件,时刻保护着公开信道上消息传输的机密性。

近期,公钥加密仍处于快速发展阶段,在安全性方面,各类超越传统语义安全的高级安全属性研究已经日趋成熟,基于复杂性弱假设的细粒度安全的研究正在兴起;在功能性方面,函数加密的研究方兴未艾,全同态加密的研究如火如荼。我们已经有幸见证了公钥加密之旅的美妙风景,但还有更广袤深邃的领域待探索征服。

公钥加密历经多年发展,各类方案层出不穷,相关概念定义繁多,因此想深入学习的读者往往会感觉陷入书山文海,难识庐山真面目。本书试图快速引导读者登高俯瞰,将公钥加密的设计方法尽收眼底,达到万变不离其宗的认知。为此,本书的内容偏重基于一般假设的通用构造。这样的选择有诸多好处,从理论角度,通用构造剥离了不必要的细节、凸显了核心要素,从而更容易洞察公钥加密的本质和复杂性下界;从实际角度,通用构造可以启发更多的具体构造,可根据安全或应用的需求灵活选择新的困难假设给出实例化方案。

本书的第一章简述了公钥加密的发展历程,第二章介绍了准备知识,为后续章节做好铺垫。第三章回顾经典的公钥加密方案,方便读者先获得具象的认识,预备一些重要的例子。第四章是核心部分,从各类密码组件出发展示了公钥加密的通用构造,并在最后获得更高阶的抽象,与对称加密的构造相互呼应、完美契合。第五章和第六章分别从安全性增强和功能性扩展两个维度介绍公钥加密的重要成果和前沿进展。第七章简介了公钥加密的标准化与工程实践,打通理论与实践的最后一公里。

书中有不少看似不起眼的注记,它们大多来源于作者科研过程中的心得体会,领悟其中蕴含的思辨方式之后或许能看到更美的风景。总的来说,切实掌握本书的内容之后,可使读者进入可证明安全密码学的新层次,为进一步的研究打好基础。

密码科学技术国家实验室对本书的出版给予了极大支持,作者深表感谢。作者也借此机会对在密码学研究给予作者热情支持帮助的上海交通大学的郁昱教授、刘胜利教授和山东大学的王美琴教授深表感谢。最后,感谢家人们的默默支持,没有你们的全情支持,我们不可能完成此书。

由于水平有限,时间紧迫,定有许多不当之处。诚恳欢迎批评指正。

陈宇 & 秦宝东

2023 年夏

# 目录

前言	i
前言	1
<b>第一章 公钥加密的概述</b>	<b>2</b>
1.1 背景与起源 . . . . .	3
1.2 发展历程简介 . . . . .	4
1.2.1 安全性增强 . . . . .	4
1.2.2 功能性丰富 . . . . .	4
<b>第二章 准备知识</b>	<b>5</b>
2.1 符号与记号 . . . . .	6
2.2 可证明安全方法 . . . . .	7
2.2.1 如何书写安全性证明 . . . . .	8
2.3 困难问题 . . . . .	11
2.3.1 整数分解类假设 . . . . .	11
2.3.2 离散对数类假设 . . . . .	13
2.3.3 格类假设 . . . . .	14
2.4 复杂性理论初步 . . . . .	17
2.5 信息论工具 . . . . .	19
2.5.1 熵的概念 . . . . .	19
2.5.2 随机性提取 . . . . .	20
2.6 公钥加密基本安全模型 . . . . .	21
2.6.1 公钥加密方案 . . . . .	21
2.6.1.1 基本性质 . . . . .	24
2.6.2 密钥封装机制 . . . . .	24
2.6.3 两类混合加密范式的比较 . . . . .	26
<b>第三章 经典公钥加密方案回顾</b>	<b>27</b>
3.1 基于数论问题的经典方案 . . . . .	28
3.1.1 Goldwasser-Micali PKE . . . . .	28
3.1.2 Rabin PKE . . . . .	29
3.2 基于离散对数类问题的经典方案 . . . . .	30
3.2.1 ElGamal PKE . . . . .	30
3.2.2 Twisted ElGamal PKE . . . . .	31
3.3 基于格问题的经典方案 . . . . .	33
3.3.1 Regev PKE . . . . .	33
3.3.2 GPV PKE . . . . .	34
<b>第四章 通用构造方法</b>	<b>37</b>
4.1 单向陷门函数类 . . . . .	38
4.1.1 单向陷门函数 . . . . .	38
4.1.2 有损陷门函数 . . . . .	41

4.1.3 相关积单向陷门函数 . . . . .	47
4.1.4 自适应单向陷门函数 . . . . .	49
4.2 哈希证明系统类 . . . . .	59
4.2.1 起源释疑 . . . . .	60
4.2.2 HPS 的构造 . . . . .	61
4.2.3 基于 HPS 构造 IND-CPA KEM . . . . .	63
4.2.4 基于 HPS 构造 IND-CCA KEM . . . . .	64
4.3 可提取哈希证明系统类 . . . . .	68
4.3.1 起源释疑 . . . . .	68
4.3.2 EHPS 的构造 . . . . .	69
4.3.3 基于 EHPS 构造 IND-CPA KEM . . . . .	70
4.3.4 基于 EHPS 构造 IND-CCA KEM . . . . .	72
4.3.5 小结 . . . . .	75
4.3.5.1 HPS 与 EHPS 的分析比较 . . . . .	75
4.4 程序混淆类 . . . . .	77
4.4.1 背景知识 . . . . .	77
4.4.2 受限伪随机函数 . . . . .	79
4.4.3 基于不可区分混淆的 KEM 构造 . . . . .	82
4.5 可公开求值伪随机函数类 . . . . .	86
4.5.1 定义与安全性 . . . . .	86
4.5.2 基于 PEPRF 的 KEM 构造 . . . . .	88
4.5.3 PEPRF 的构造 . . . . .	89
4.5.3.1 基于 DDH 假设的 PEPRF . . . . .	89
4.5.3.2 基于 QR 假设的 PEPRF . . . . .	89
4.5.3.3 基于 TDF 的 PEPRF . . . . .	90
4.5.3.4 基于 HPS 的 PEPRF 构造 . . . . .	91
4.5.3.5 基于 EHPS 的 PEPRF 构造 . . . . .	92
4.5.3.6 基于 $i\mathcal{O}$ 的 PEPRF 构造 . . . . .	93
4.5.4 小结 . . . . .	94
<b>第五章 公钥加密的安全性增强</b> . . . . .	<b>95</b>
5.1 抗泄漏安全 . . . . .	96
5.1.1 安全模型 . . . . .	97
5.1.2 NS09 方案 . . . . .	97
5.1.2.1 基于哈希证明系统的通用构造 . . . . .	97
5.1.2.2 基于 DDH 问题的实例化方案 . . . . .	99
5.1.3 QL13 方案 . . . . .	100
5.1.3.1 一次有损过滤器 . . . . .	100
5.1.3.2 基于一次有损过滤器的通用构造 . . . . .	101
5.1.3.3 基于 DDH 问题的实例化方案 . . . . .	105
5.1.4 CQX18 方案 . . . . .	106
5.1.4.1 规则有损函数及扩展 . . . . .	106
5.1.4.2 全除一规则有损函数的具体构造 . . . . .	109
5.1.4.3 全除一规则有损函数的通用构造 . . . . .	110
5.1.4.4 规则有损函数的应用 . . . . .	112

5.1.5 CWZ18 方案 . . . . .	117
5.1.5.1 LR-PEPRFs 的定义 . . . . .	117
5.1.5.2 LR-PEPRFs 的构造 . . . . .	118
5.1.5.3 LR-PEPRFs 的应用 . . . . .	122
5.2 抗篡改安全 . . . . .	123
5.2.1 安全模型 . . . . .	123
5.2.2 Wee12 方案 . . . . .	123
5.2.2.1 自适应单向陷门关系的特殊性质 . . . . .	123
5.2.2.2 基于自适应单向陷门关系的 RKA-CCA 方案 . . . . .	124
5.2.3 CQZ+22 方案 . . . . .	126
5.2.3.1 不可延展函数 . . . . .	127
5.2.3.2 不可延展性与单向性之间的关系 . . . . .	128
5.2.3.3 不可延展函数的构造 . . . . .	130
5.2.3.4 不可延展函数的应用 . . . . .	132
5.3 消息依赖密钥安全 . . . . .	135
5.3.1 安全模型 . . . . .	135
5.3.2 Wee16 方案 . . . . .	136
5.3.2.1 同态平滑仿射哈希 . . . . .	136
5.3.2.2 基于同态平滑仿射哈希的 KDM-CPA 方案 . . . . .	137
5.3.3 HLL16 方案 . . . . .	139
5.3.3.1 支持辅助输入的认证加密方案 . . . . .	139
5.3.3.2 KDM-CCA 方案的通用设计方法 . . . . .	141
5.3.3.3 实例化方案 . . . . .	142
5.3.4 QLH13 方案 . . . . .	144
5.3.4.1 KDM 函数族定义 . . . . .	144
5.3.4.2 KDM-CCA 安全的 Cramer-Shoup 方案 . . . . .	145
<b>第六章 公钥加密的功能性扩展</b>	<b>147</b>
6.1 确定性公钥加密 . . . . .	148
6.1.1 安全模型 . . . . .	148
6.1.2 基于对偶仿射哈希的 DPKE 方案 . . . . .	148
6.1.2.1 对偶仿射哈希 . . . . .	148
6.1.2.2 DPKE 方案的构造 . . . . .	149
6.1.3 基于 DLIN 的实例化方案 . . . . .	151
6.2 可搜索公钥加密 . . . . .	152
6.2.1 PEKS 方案 . . . . .	152
6.2.1.1 形式化定义 . . . . .	152
6.2.1.2 安全模型 . . . . .	154
6.2.1.3 PEKS 与 IBE 之间的关系 . . . . .	154
6.2.1.4 方案构造 . . . . .	155
6.2.2 PKE-PEKS 方案 . . . . .	158
6.2.2.1 形式化定义 . . . . .	158
6.2.2.2 安全模型 . . . . .	159
6.2.2.3 方案构造 . . . . .	160
6.2.3 PAEKS 方案 . . . . .	169

---

6.2.3.1	形式化定义 . . . . .	170
6.2.3.2	安全模型 . . . . .	171
6.2.3.3	方案构造 . . . . .	173
6.2.3.4	应用 . . . . .	177
<b>第七章</b>	<b>标准化及工程实践</b>	<b>179</b>
7.1	标准化与工程实践 . . . . .	180
7.1.1	公钥加密的标准化 . . . . .	180
7.1.1.1	国内外标准化组织简介 . . . . .	180
7.1.1.2	公钥加密标准方案 . . . . .	181
7.1.2	公钥加密的工程实践 . . . . .	182
7.1.2.1	重要方案的优秀开源实现 . . . . .	182
7.1.2.2	重要的开源密码库 . . . . .	182
7.1.3	密码学的工程实践经验 . . . . .	183
<b>参考文献</b>		<b>184</b>
<b>后记</b>		<b>192</b>

# 前言

# 第一章 公钥加密的概述

## 1.1 背景与起源

密码的历史甚至早于文字的出现。密码的历史进程大概可以分为以下几个阶段：

- 古典阶段：
- 现代阶段：Shannon 和 DES
- Diffie-Hellman 的密码学新方向。

## 1.2 发展历程简介

目前的设想是从两个维度梳理公钥加密的发展历史: 功能性的丰富和安全性的增强.

自 Diffie 和 Hellman 的划时代论文 [**DH-IEEE-IT-1976**] 后, 公钥密码学的发展一日千里、日新月异, 热潮持续至今, 始终是现代密码学的核心和重要技术的摇篮.

公钥密码学的发展大体可以分为两条主线: 一条是安全性的增强, 从最初的直觉安全演进到严格健壮的语义安全, 再到不可区分选择密文安全和各类超越传统安全模型的高级安全性, 如抗泄漏安全、抗篡改安全和消息依赖密钥安全; 另一条是功能性的丰富, 从最初的一对一加解密到基于身份加密, 再到属性加密乃至极致泛化的函数加密.

### 1.2.1 安全性增强

最初的 RSA PKE 只满足朴素单向安全. Goldwasser 和 Micali 思考了公钥加密应该满足何种最低安全性这一问题, 提出了严格的安全模型, 设计了首个可证明安全的公钥加密方案 Goldwasser-Micali PKE.

### 1.2.2 功能性丰富

## 第二章 准备知识

## 2.1 符号与记号

对于正整数  $n$ , 用  $[n]$  表示集合  $\{1, \dots, n\}$ . 对于集合  $X$ ,  $|X|$  表示其大小,  $x \xleftarrow{R} X$  表示从  $X$  中均匀采样  $x$ ,  $U_X$  表示  $X$  上的均匀分布. 如果一个概率算法的运行时间是关于输入规模  $n$  的多项式函数  $\text{poly}(n)$ , 则称其是概率多项式时间的算法, 简记为 (probabilistic polynomial time, PPT). 令  $\mathcal{A}$  是一个随机算法,  $z \leftarrow \mathcal{A}(x; r)$  表示  $\mathcal{A}$  在输入为  $x$  和随机带为  $r$  时输出  $z$ , 当上下文明确时, 常隐去随机带  $r$  简记为  $z \leftarrow \mathcal{A}(x)$ . 令  $f(\cdot)$  是关于  $n$  的函数, 如果对于任意的多项式  $p(\cdot)$  均存在常数  $c$  使得  $n > c$  时总有  $f(n) < 1/p(n)$  成立, 则称  $f$  是关于  $n$  的可忽略函数, 记为  $\text{negl}(n)$ . 在本书中,  $\kappa \in \mathbb{N}$  表示计算安全参数,  $\lambda \in \mathbb{N}$  表示统计安全参数. 令  $X = \{X_k\}_{k \in \mathbb{N}}$  和  $Y = \{Y_k\}_{k \in \mathbb{N}}$  是两个由  $k$  索引的分布簇, 如果  $X$  和  $Y$  之间的统计距离是关于  $\lambda$  的可忽略函数, 则称  $X$  和  $Y$  统计不可区分, 记为  $X \approx_s Y$ ; 如果任意 PPT 敌手区分  $X$  和  $Y$  的优势函数为  $\text{negl}(\kappa)$ , 则称  $X$  和  $Y$  计算不可区分, 记为  $X \approx_c Y$ .

对于实数  $x \in \mathbb{R}$ , 令  $\lfloor x \rfloor$  表示  $x$  的下取整,  $\lfloor x \rfloor := \lfloor x + 1/2 \rfloor$  表示与  $x$  最接近的整数.

令  $F$  是带密钥的函数,  $F_k(x)$  表示函数  $F$  在密钥  $k$  控制下对  $x$  的求值, 也常记作  $F(k, x)$ .

表 2.1: 缩略词及其含义对照表

缩略词	英文表达	中文含义
CPA	chosen-plaintext attack	选择明文攻击
CCA	chosen-ciphertext attack	选择密文攻击
PKE	public-key encryption	公钥加密方案
-	hardcore function	硬核函数
-	hardcore predicate	硬核谓词
-	oracle	预言机
-	one-way trapdoor function	单向陷门函数

## 2.2 可证明安全方法

Talk is cheap. Show me the code.

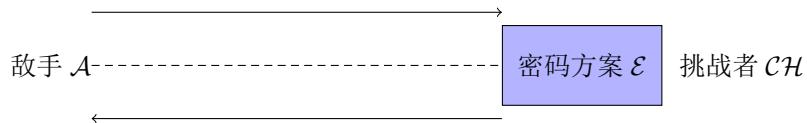
— Linus Torvalds

长久以来, 密码方案的安全性分析没有系统标准的方式, 传统方式由密码分析者通过尝试各类攻击来检验密码方案的安全性. 容易看出, 传统的分析方式存在以下局限: (1) 分析结果严重依赖分析者的个人能力 (如细致的观察和敏锐的直觉); (2) 分析者难以穷尽所有可能的攻击. 因此, 绝大多数古典密码陷入“设计—攻破—修补—攻破”的循环往复怪圈, 难以称之为真正的科学.

直到上世纪八十年代, Goldwasser 和 Micali [GM-STOC-1982] 借鉴计算复杂性理论的归约技术, 开创了可证明安全方法. 从此, 密码方案的安全性分析手段由反向攻击转为正向证明, 安全性由“声称安全”变为“可证安全”, 密码学也从此由艺术蝶变为真正的科学.

简言之, 可证明安全方法的核心是以下三要素组成:

- **精确的安全模型:** 通常由攻击者和挑战者之间的交互式游戏进行刻画, 如图 ?? 所示, 包括:
  - 敌手的计算能力: 常见的有概率多项式时间和指数时间
  - 敌手能够获取的信息, 包括:
    1. 固定信息: 如方案的公开参数等公开信息
    2. 非固定信息: 在攻击过程中获得的信息, 形式化为访问相应预言机获得的输出
  - 敌手的攻击效果: 以加密方案为例, 敌手恢复密钥和恢复明文是不同的攻击效果



$$\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S] - \Pr[T]|$$

图 2.1: 安全模型

令事件  $S$  表示敌手攻击成功这一事件,  $t$  表示某固定的基准优势 (如区分类游戏定义为  $1/2$ , 搜索性游戏定义为  $0$ ), 定义  $\mathcal{A}$  的优势函数为  $\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S] - t|$ , 其中  $S$  所在的概率空间由  $\mathcal{A}$  和  $\mathcal{CH}$  的随机带确定. 如果对于所有 PPT 敌手  $\mathcal{A}$ ,  $\text{Adv}_{\mathcal{A}}(\kappa)$  均是关于安全参数  $\kappa$  的可忽略函数, 则称密码方案  $\mathcal{E}$  在既定的安全模型下是安全的.

- 清晰的困难性假设
- 严格的归约式证明: 通过反证式论证将方案的安全性归结到困难性假设.

图 ?? 是归约式证明的交换图表. 归约式证明的步骤如下:

1. 假设存在 PPT 的敌手  $\mathcal{A}$  在既定安全模型下针对密码方案  $\mathcal{E}$  具有不可忽略的优势  $\epsilon_1(\kappa)$ ;
2. 利用  $\mathcal{A}$  的能力, 构建 PPT 的算法  $\mathcal{B}$  以不可忽略的优势  $\epsilon_2(\kappa)$  打破困难问题. 这里  $\mathcal{R}$  通常以扮演敌手  $\mathcal{A}$  的挑战者的方式调用  $\mathcal{A}$ , 因此也常称  $\mathcal{R}$  是模拟算法或归约算法.  $\mathcal{R}$  调用敌手  $\mathcal{A}$  的方式又可细分为两类:
  - 黑盒方式:  $\mathcal{R}$  以黑盒的方式调用  $\mathcal{A}$ , 从算法的角度理解就是  $\mathcal{R}$  以  $\mathcal{A}$  作为子程序调用. 黑盒方式实质上证明了比所需更强的结果, 即  $\exists \mathcal{R} \forall \mathcal{A}$ . 此类证明最为常见, 被称为黑盒归约 (black-box reduction) 或者一致归约 (universal reduction).
  - 非黑盒方式:  $\mathcal{R}$  以非黑盒的方式调用  $\mathcal{A}$ , 充分利用了  $\mathcal{A}$  的个体信息, 如算法结构、运行时间等. 这类证明是完全契合可证明安全思想的, 即  $\forall \mathcal{A} \exists \mathcal{R}$ . 此类证明相对少见, 被称为个体归约 (individual reduction) [Deng-EUROCRYPT-2017], 常可以突破黑盒归约下的安全性下界.

上述两步归约式论证的逻辑是: 构造出的算法  $\mathcal{R}$  与困难假设相矛盾, 因此不存在算法  $\mathcal{R}$ , 进而得出  $\mathcal{A}$  不存在的结论.

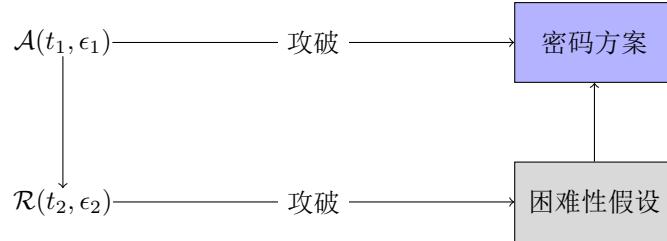


图 2.2: 归约式证明的交换图表

**安全性强弱.** 在明确可证明安全方法后, 密码方案的强弱可以根据三要素的强弱定性分析:

- 安全模型的强弱: 敌手的计算能力越强大、获得的信息越多、攻击的效果越弱, 则所确定的安全模型越强, 反之越弱.
- 困难假设的强弱: 通常搜索类假设强于判定类假设, 平均情形 (average-case) 假设强于最坏情形 (worst-case) 假设.
- 归约质量的优劣: 笼统的说,  $(t_2, \epsilon_2)$  越接近  $(t_1, \epsilon_1)$ , 归约的质量越高. 在归约算法的运行时间和优势函数两个指标中, 通常更关注后者. 定义归约松紧因子  $r = \epsilon_2/\epsilon_1$ , 如果  $r$  是一个常数, 称归约是紧的; 如果  $r$  是一个可察觉的函数 (noticeable function), 称归约多项式松弛的, 归约有效; 如果  $r$  是一个可忽略函数, 称归约超多项式松弛, 归约无效.

#### 注记 2.1

阿基米德曾说过:“给我一个支点, 我能撬起地球!” 可证明安全方法与这句名言有共通之处, 地球可以理解为待证明方案的安全性, 支点和杠杆可以理解为归约式证明方法, 而施加在杠杆上的力可以理解为困难性假设. 如果支点在困难性假设和方案安全性正中, 代表归约最优, 方案的安全性可以紧归约到假设困难性上; 如果力臂过短, 则代表归约松弛, 困难性假设无法有意义的保证密码方案的安全性.

### 2.2.1 如何书写安全性证明

很多初学者对方案/协议的安全性有隐约的直觉, 但是很难写出严格精确的证明. 密码学中的安全性证明如同吉他中的大横按, 是横亘在所有初学者面前的一个障碍.

本小节将以极为精炼的方式归纳总结安全性证明的构建方式. 给出安全性证明大致有两种外在形式, 分别是单一归约和游戏序列.

单一归约适用于密码方案/协议仅依赖单一困难问题的简单形式. 拟基于惟一困难假设  $\mathcal{P}$  证明密码方案/协议

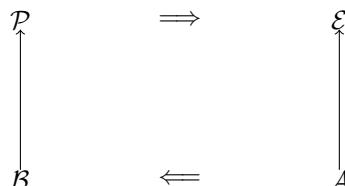


图 2.3: 单一归约

$\mathcal{E}$  的安全性时, 证明的方式是构建如图 ?? 所示的交换图表, 即首先假设存在 PPT 的敌手  $\mathcal{A}$  打破  $\mathcal{E}$  的安全性, 再利用  $\mathcal{A}$  构造算法  $\mathcal{B}$  打破困难假设  $\mathcal{P}$ . 构造  $\mathcal{B}$  的方法通常是令  $\mathcal{B}$  在方案  $\mathcal{E}$  的安全游戏中模拟挑战者的角色, 模拟的方式是将困难问题的实例直接嵌入到安全游戏的参数中. 此时, 基于  $\mathcal{P}$  的困难性便可得到任意 PPT 敌手针对  $\mathcal{E}$  的优势函数  $\text{Adv}_{\mathcal{A}}^{\mathcal{E}}(\kappa) \leq \text{negl}(\kappa)$  的结论.

## 注记 2.2

需要特别指出, 单一归约的适用范围有限, 仅适用于困难问题单一且能够直接嵌入安全游戏的场景, 这就要求安全游戏的目标和困难问题的类型必须相同, 比如同是计算性或者判定性. 有时, 在困难问题单一的情况下, 我们也需按照下面介绍的游戏序列方式组织证明. 如章节中的 ElGamal PKE 的证明. [add ref here](#)



对于基于多个困难问题的密码方案/协议, 即待证明的定理形如  $\mathcal{P}_1 + \cdots + \mathcal{P}_n \Rightarrow \mathcal{E}$ , 就难以使用单一归约进行证明了. Shoup [Shoup-ePrint-2004] 针对该情形, 系统地提出了“游戏序列”的方式组织证明. 游戏序列的证明框架如下:

1. 引入一系列游戏, 记为  $\text{Game}_0, \dots, \text{Game}_m$ . 敌手在  $\text{Game}_i$  中成功的事件记作  $S_i$ , 优势基准为  $t$ , 则敌手在  $\text{Game}_i$  的优势函数为:

$$\text{Adv}_{\mathcal{A}}^{\text{Game}_i} = |\Pr[S_i] - t|$$

通常情况下  $\text{Game}_0$  刻画原始真实的安全游戏,  $\text{Game}_m$  刻画最终游戏且  $\text{Adv}_{\mathcal{A}}^{\text{Game}_m} = |\Pr[S_m] - t| = \text{negl}(\kappa)$ , 即敌手在  $\text{Game}_m$  中的优势函数是可忽略的.

2. 证明对于所有的  $i \in [m]$  均有  $|\Pr[S_i] - \Pr[S_{i-1}]| \leq \text{negl}(\kappa)$ ;
3. 通过混合论证 (hybrid argument) 得出  $\text{Adv}_{\mathcal{A}}^{\text{Game}_m} = |\Pr[S_0] - t| = \text{negl}(\kappa)$  的结论.

对于同一密码方案/协议, 在使用游戏序列进行证明时存在多种可能得游戏序列组织方式. 尽管在游戏序列的设定没有严格的规定, 但有以下两个经验准则:

- 相邻游戏的差异需最小化, 下一个游戏与上一个游戏仅有一个差异为宜;
- 差异应易于分析.

相邻游戏之间的差异通常有以下三种类型:

1. 差异源于不可区分的分布;
2. 差异基于某特定事件是否发生;
3. 差异仅是概念上调整, 为后续分析做铺垫.

对于第一类差异,  $\text{Game}_i$  和  $\text{Game}_{i+1}$  的变化可以归结为分布的不可区分性, 如  $Z_0 \approx Z_1$ , 其中  $Z_0$  和  $Z_1$  是两个分布. 换言之, 存在归约算法  $B$ , 以  $Z_0$  为输入时, 可以完美模拟敌手在  $\text{Game}_i$  中的视图; 以  $Z_1$  为输入时, 可以完美模拟敌手在  $\text{Game}_{i+1}$  中的视图. 我们令  $\text{View}_i$  表述敌手在  $\text{Game}_i$  中的视图. 在上下文清晰没有歧义时, 也常用  $\text{Game}_i$  直接代指敌手的视图.

- 当  $Z_0 \approx_s Z_1$  时, 利用复合引理 (composition lemma) 可以立刻得出任意敌手在两个游戏中的输出统计不可区分, 进而得出  $\Pr[S_0] - \Pr[S_1] \leq \text{negl}(\kappa)$
- 当  $Z_0 \approx_c Z_1$  时, 如果敌手成功这一事件  $\mathcal{B}$  可准确判定, 则同样可以得出  $\Pr[S_0] - \Pr[S_1] \leq \text{negl}(\kappa)$ , 论证过程如下图 ?? 所示,  $\mathcal{B}$  在事件  $S$  发生时输出 “1”, 否则输出 “0”. 根据游戏定义, 有  $\Pr[\mathcal{B}(Z_0)] = \Pr[S_i]$ ,  $\Pr[\mathcal{B}(Z_1)] = \Pr[S_{i+1}]$ , 因此有:

$$|\Pr[S_i] - \Pr[S_{i+1}]| = |\Pr[\mathcal{B}(Z_0) = 1] - \Pr[\mathcal{B}(Z_1) = 1]| \leq \text{negl}(\kappa)$$

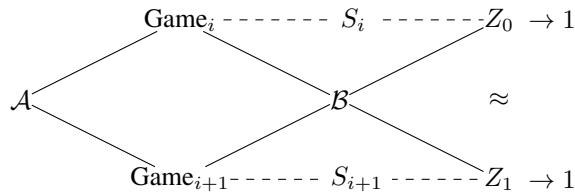


图 2.4: 基于分布不可区分的游戏演变

**注记 2.3**

许多学术论文在证明的过程中为了简便往往先证明敌手在两个相邻游戏中的视图不可区分, 再据此得出敌手优势函数的差是可忽略这一结论。在多数情形, 这种论证并无问题, 但需要特别小心的是在视图计算不可区分时, 必须同时确保归约算法能够准确判定敌手成功事件才能够确保证明严谨。在有些特殊情形, 归约算法无法有效判定敌手是否在游戏中成功, 此时归约算法无法利用敌手成功概率差异打破底层区分性假设, 从而导致归约失效。请读者参考文献 [HLL-ASIACRYPT-2016] 加深对该证明技术细节的理解和掌握。



第二类差异取决于某个特定事件是否发生, 即定义在同一概率空间的两个相邻游戏  $\text{Game}_i$  和  $\text{Game}_{i+1}$  仅在某特定事件  $F$  发生时存在差异, 在  $F$  不发生时完全一致, 概率描述如下:

$$S_i \wedge \neg F = S_{i+1} \wedge \neg F$$

为了分析敌手在相邻游戏  $\text{Game}_i$  和  $\text{Game}_{i+1}$  中的优势函数差, 需要以下的“差异引理”(difference lemma):

**引理 2.1 (Difference Lemma)**

令  $A, B, F$  是定义在同一概率空间中的事件, 如果  $A \wedge \neg F = B \wedge \neg F$ , 那么则有  $|\Pr[A] - \Pr[B]| \leq \Pr[F]$ 。



**证明** 差异引理的证明如下, 仅需使用古典概率中简单的缩放技巧:

$$\begin{aligned} |\Pr[A] - \Pr[B]| &= |\Pr[A \wedge F] + \Pr[A \wedge \neg F] - \Pr[B \wedge F] - \Pr[B \wedge \neg F]| // \text{全概率展开} \\ &= |\Pr[A \wedge F] - \Pr[B \wedge F]| // \text{化简} \\ &\leq \max\{\Pr[A \wedge F], \Pr[B \wedge F]\} \leq \Pr[F] // \text{缩放} \end{aligned}$$

证毕!



根据差异引理, 若需证明  $|\Pr[S_i] - \Pr[S_{i+1}]| \leq \text{negl}(\kappa)$ , 仅需证明  $\Pr[F] \leq \text{negl}(\kappa)$ , 证明细分为以下两种情形:

- $F$  发生的概率取决于敌手的计算能力, 如敌手找到哈希函数的碰撞或者成功伪造消息认证码。该情形需要建立安全归约, 即若  $F$  发生, 则存在敌手打破困难问题  $X_i$ 。
- $F$  发生的概率与敌手的计算能力无关。该情形仅需纯粹的信息论论证 (information-theoretic argument)。

第三类差异称为桥接差异。在分析游戏序列之间的差异时, 有时需要引入桥接步骤对某个变量的生成方式以等价的方式重新定义, 以确保差异分析的良定义。桥接步骤引入的差异仅是挑战者侧的概念性变化, 敌手侧的视图完全相同, 因此  $\Pr[S_i] = \Pr[S_{i+1}]$ 。桥接步骤看似可有可无, 实则必要, 若不引入必要的桥接步骤, 则会使得证明跳跃难以理解、游戏序列间的差异无法精确分析。

## 2.3 困难问题

密码学中常见的困难问题可大致分为数论类和格类, 其中前者是代数问题, 后者是数的几何问题, 这也正是格类困难问题具备抗量子攻击的原因之一.

数论类的假设又可进一步分为整数分解类和离散对数类两个分支. 本章首先介绍常见的数论类假设, 再介绍常见的格类困难问题.

### 2.3.1 整数分解类假设

整数分解类假设定义在群  $\mathbb{Z}_N^*$  上. 其中:

$$\mathbb{Z}_N^* \stackrel{\text{def}}{=} \{b \in \{1, \dots, N\} \mid \gcd(b, N) = 1\}$$

也即  $\mathbb{Z}_N^*$  是整数集合  $\{1, \dots, N-1\}$  中所有与  $N$  互质元素构成的子集, 在模乘运算  $ab \stackrel{\text{def}}{=} [ab \bmod N]$  下构成交换群.

以下令  $\text{GenModulus}$  是多项式时间算法, 其以安全参数  $1^\kappa$  为输入, 输出  $(N = pq, p, q)$ , 其中  $p$  和  $q$  (以压倒性概率) 是两个  $\kappa$ -bit 的素数.

#### 定义 2.1 (整数分解假设)

整数分解问题指分解大整数在平均意义上是困难的. 我们称整数分解假设相对于  $\text{GenModulus}$  成立当且仅当对于任意 PPT 敌手:

$$\Pr[\mathcal{A}(N) = (p', q') \text{ s.t. } p'q' = N] \leq \text{negl}(\kappa)$$

上述概率建立在敌手  $\mathcal{A}$  和  $\text{GenModulus}(1^\kappa) \rightarrow (N, p, q)$  的随机带上.



尽管整数分解假设历经百年的分析攻击仍然健壮, 但是它并不直接蕴含高效的密码系统. 这就引发密码科技工作者研究与整数分解问题困难性相关问题的研究. 1978 年, Rivest, Shamir 和 Adleman 提出了 RSA 问题.

令  $\text{GenRSA}$  是多项式时间算法, 其以安全参数  $1^\kappa$  为输入, 输出两个  $\kappa$ -bit 素数的乘积  $N$  作为模数, 同时输出正整数  $(e, d)$  满足  $ed = 1 \bmod \phi(N)$ .

#### 定义 2.2 (RSA 假设)

RSA 问题指在平均意义下求解  $\mathbb{Z}_N^*$  的  $e$  次方根是困难的. 我们称 RSA 假设相对于  $\text{GenModulus}$  成立当且仅当对于任意 PPT 敌手:

$$\Pr[\mathcal{A}(N, e, y) = x \text{ s.t. } x^e = y \bmod N] \leq \text{negl}(\kappa)$$

上述概率建立在敌手  $\mathcal{A}$ 、 $\text{GenRSA}(1^\kappa) \rightarrow (N, e, d)$  和随机选取  $x \in \mathbb{Z}_N^*$  随机带上.



#### 注记 2.4

RSA 问题刻画的是在不知晓  $\phi(N)$  的情况下计算  $\mathbb{Z}_N^*$  中随机元素的  $e$  次方根是困难的. 容易看出, 如果敌手能够打破整数分解问题, 则可以通过分解  $N$  求出  $\phi(N)$ , 进而通过 Fermat 小定理计算  $e$  次方根. 因此, 整数分解问题难于 RSA 问题, 整数分解假设弱于 RSA 假设. 两个假设是否等价仍然未知.



给定群  $\mathbb{G}$ , 称  $y \in \mathbb{G}$  是一个二次剩余当且仅当  $\exists x \in \mathbb{G} \text{ s.t. } x^2 = y$ .  $x$  成为  $y$  的平方根. 如果一个元素不是二次剩余则称其为二次非剩余. 在 abelian 群中, 二次剩余构成子群.

首先考察群  $\mathbb{Z}_p^*$  中的二次剩余, 其中  $p$  是素数. 定义函数  $\text{sq}_p : \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*$  为  $\text{sq}_p(x) \stackrel{\text{def}}{=} [x^2 \bmod p]$ . 当  $p$  是大于 2 的素数时,  $\text{sq}_p$  是 2-对-1 函数, 因此立刻可知  $\mathbb{Z}_p^*$  中恰好一半元素是二次剩余. 记模  $p$  的二次剩余集合为  $\mathcal{QR}_p$ , 模

$p$  的二次非剩余集合为  $\mathcal{QNR}_p$ , 我们有:

$$|\mathcal{QR}_p| = |\mathcal{QNR}_p| = \frac{|\mathbb{Z}_p^*|}{2} = \frac{p-1}{2}$$

定义元素  $x \in \mathbb{Z}_p^*$  模  $p$  的 Jacobi 符号如下:

$$\mathcal{J}_p(x) \stackrel{\text{def}}{=} \begin{cases} +1 & \text{if } x \in \mathcal{QR}_p \\ -1 & \text{if } x \in \mathcal{QNR}_p \end{cases}$$

再考察群  $\mathbb{Z}_N^*$  中的二次剩余, 其中  $N$  是两个互异素数  $p$  和  $q$  的乘积. 由中国剩余定理可知:  $\mathbb{Z}_N^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^*$ , 令  $y \leftrightarrow (y_p, y_q)$  表示上述同构映射给出的分解, 易知  $y$  是模  $N$  的二次剩余当且仅当  $y_p$  和  $y_q$  分别是模  $p$  和模  $q$  的二次剩余. 定义函数  $\text{sq}_N : \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$  为  $\text{sq}_N(x) \stackrel{\text{def}}{=} [x^2 \bmod N]$ , 当  $N$  为互异素数乘积时,  $\text{sq}_N$  是 4-对-1 函数. 记模  $N$  的二次剩余集合为  $\mathcal{QR}_N$ , 由  $\mathcal{QR}_N$  与  $\mathcal{QR}_p \times \mathcal{QR}_q$  之间的一一对应关系可知:

$$\frac{|\mathcal{QR}_N|}{|\mathbb{Z}_N^*|} = \frac{|\mathcal{QR}_p| \cdot |\mathcal{QR}_q|}{|\mathbb{Z}_N^*|} = \frac{1}{4}$$

从二次剩余的角度可以对  $\mathbb{Z}_N^*$  中的元素进行如下的划分: (i)  $\mathbb{Z}_N^*$  可以划分为相同大小的  $\mathcal{J}_N^{+1}$  和  $\mathcal{J}_N^{-1}$  (Jacobi 符号分别为 1 和 -1); (ii)  $\mathcal{J}_N^{+1}$  有可以划分为  $\mathcal{QR}_N$  和  $\mathcal{QNR}_N^{+1}$ , 其中  $\mathcal{QNR}_N^{+1} \stackrel{\text{def}}{=} \{x \in \mathbb{Z}_N^* \mid x \notin \mathcal{QR}_N \wedge \mathcal{J}_N(x) = +1\}$ .

### 定义 2.3 (模二次剩余假设)

模二次剩余 (QR, quadratic residue) 假设指  $\mathcal{QR}_N$  上的均匀分布与  $\mathcal{QNR}_N^{+1}$  上的均匀分布计算不可区分. 我们称模二次剩余假设相对于 GenModulus 成立当且仅当对于任意 PPT 敌手:

$$|\Pr[\mathcal{A}(N, y_0) = 1] - \Pr[\mathcal{A}(N, y_1) = 1]| \leq \text{negl}(\kappa)$$

上述概率建立在敌手  $\mathcal{A}$ 、 $\text{GenModulus}(1^\kappa) \rightarrow (N, p, q)$  和随机选取  $y_0 \in \mathcal{QR}_N, y_1 \in \mathcal{QNR}_N^{+1}$  的随机带上. ♣

与 QR 假设应用紧密相关的技术细节是如何对  $\mathcal{QR}_N$  和  $\mathcal{QNR}_N^{+1}$  进行高效的均匀采样.

- 对  $\mathcal{QR}_N$  进行均匀采样较为简单: 仅需随机选取  $x \in \mathbb{Z}_N^*$  再令  $y := x^2 \bmod N$  即可. 注意到  $x^2 \bmod N$  是一个 4-to-1 的正则函数, 因此当  $x \xleftarrow{R} \mathbb{Z}_N^*$  时, 输出  $y$  服从  $\mathcal{QR}_N$  上的均匀分布.
- 对  $\mathcal{QNR}_N^{+1}$  进行均匀采样稍显复杂, 当  $N$  的分解未知时如何均匀采样未知. 我们可以借助辅助信息  $z \in \mathcal{QNR}_N^{+1}$  完成采样, 即随机选取  $x \in \mathbb{Z}_N^*$ , 输出  $y := z \cdot x^2 \bmod N$ . 可以验证, 当  $x \xleftarrow{R} \mathbb{Z}_N^*$  时, 输出  $y$  服从  $\mathcal{QNR}_N^{+1}$  上的均匀分布.

### 注记 2.5

显然, 整数分解问题难于二次剩余判定问题, 因此整数分解假设弱于模二次剩余判定假设. 两个假设是否等价仍然未知. ♠

### 定义 2.4 (模平方根假设)

模平方根 (SQR, square root) 假设指对  $\mathcal{QNR}_N$  中的随机元素求平方根是困难的. 我们称模平方根假设相对于 GenModulus 成立当且仅当对于任意 PPT 敌手:

$$\Pr[\mathcal{A}(N, y) = x \text{ s.t. } x^2 = y \bmod N] \leq \text{negl}(\kappa)$$

上述概率建立在敌手  $\mathcal{A}$ 、 $\text{GenModulus}(1^\kappa) \rightarrow (N, p, q)$  和随机选取  $y \in \mathcal{QNR}_N$  的随机带上. ♣

令  $p$  和  $q$  是两个互异的模 4 余 3 的素数, 则称  $N = pq$  是 Blum 整数. 我们有以下推论:

### 命题 2.1

当  $N$  时 Blum 整数时, 则每个模  $N$  的二次剩余有且仅有一个平方根是二次剩余.



上述推论保证了当  $N$  是 Blum 整数时, 函数  $f_N \stackrel{\text{def}}{=} [x^2 \bmod N]$  构成  $\mathcal{QR}_N$  上的置换. 这一性质在构造加密方案时至关重要.

### 注记 2.6

模平方根假设等价于整数分解假设, 即在未知  $N$  分解的情况下求模平方根与分解  $N$  一样困难.



综上, 整数分解类问题的困难性关系如图 ?? 所示:

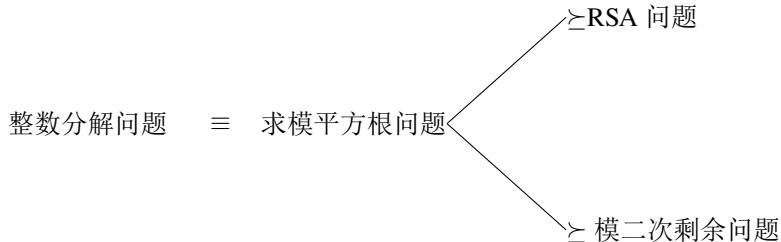


图 2.5: 整数分解类问题的困难性关系

## 2.3.2 离散对数类假设

离散对数类假设定义在循环群  $\mathbb{G}$  中. 令  $\text{GenGroup}$  是多项式时间算法, 其以安全参数  $1^\kappa$  为输入, 输出  $q$  阶循环群  $\mathbb{G} = \langle g \rangle$  的描述, 其中,  $q$  是  $\kappa$ -bit 的整数, 简记为  $(\mathbb{G}, q, g) \leftarrow \text{GenGroup}(1^\kappa)$ . 为了行文方便, 本书中假设  $\mathbb{G}$  为加法群, 用 “.” 表示群运算. 由循环群的定义可知,  $\mathbb{G}$  中的元素为  $\{g^0, g^1, \dots, g^{q-1}\}$ . 因此, 对于任意  $h \in \mathbb{G}$  存在唯一的  $x \in \mathbb{Z}_q$  使得  $g^x = h$ , 我们称  $x$  是  $h$  相对于生成元  $g$  的离散对数并记为  $x = \log_g h$ , 这里称其为离散对数强调其取值均为非负整数, 有别于标准算术对数的取值为实数.

### 定义 2.5 (离散对数假设)

离散对数问题指在平均意义下求解群元素的离散对数是困难的. 我们称离散对数 (*DLOG*) 假设相对于  $\text{GenGroup}$  成立当且仅当对于任意 PPT 敌手:

$$\Pr[\mathcal{A}(\mathbb{G}, q, g, h) = \log_g h] \leq \text{negl}(\kappa)$$

上述概率建立在敌手  $\mathcal{A}$ 、 $\text{GenGroup}(1^\kappa) \rightarrow (\mathbb{G}, q, g)$  和随机采样  $h \in \mathbb{G}$  的随机带上.



显然, 离散对数假设说明了  $x \mapsto g^x$  是从  $\mathbb{Z}_q$  到  $\mathbb{G}$  的单向函数. 单向函数能够蕴含的密码方案有限, 下面介绍与离散对数假设相关的其它假设, 它们能够作为更多密码方案的安全基础. 这类困难假设的起源于 Diffie 和 Hellman [**DH-IEEE-IT-1976**] 在 1976 年的划时代论文, 后来被称为 Diffie-Hellman 假设. 为了叙述方便, 我们首先定义 DH 函数  $\text{DH}_g : \mathbb{G}^2 \rightarrow \mathbb{G}$ ,

$$\text{DH}_g(h_1, h_2) \stackrel{\text{def}}{=} g^{\log_g h_1 \cdot \log_g h_2}$$

Diffie-Hellman 类假设可细分为两类, 一类是计算性 Diffie-Hellman (CDH) 问题, 一类是判定性 Diffie-Hellman (DDH) 问题. 下面依次介绍.

**定义 2.6 (CDH 假设)**

CDH 问题指在平均意义上计算  $\text{DH}_g$  函数是困难的. 我们称 *CDH* 假设相对于 *GenGroup* 成立当且仅当对于任意 PPT 敌手:

$$\Pr[\mathcal{A}(\mathbb{G}, q, g, g^a, g^b) = g^c] \leq \text{negl}(\kappa)$$

上述概率建立在敌手  $\mathcal{A}$ 、 $\text{GenGroup}(1^\kappa) \rightarrow (\mathbb{G}, q, g)$  和随机采样  $a, b \in \mathbb{Z}_q$  的随机带上.

**定义 2.7 (DDH 假设)**

对于四元组  $(g, g^a, g^b, g^c)$ , 如果  $g^c = \text{DH}_g(g^a, g^b)$  也即  $ab = c \pmod{q}$ , 则称其为 DH 元组. DDH 假设刻画的是随机 DH 元组和随机四元组是计算不可区分的. 我们称 *DDH* 假设相对于 *GenGroup* 成立当且仅当对于任意 PPT 敌手:

$$|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^a, g^b, g^c) = 1]| \leq \text{negl}(\kappa)$$

上述概率建立在敌手  $\mathcal{A}$ 、 $\text{GenGroup}(1^\kappa) \rightarrow (\mathbb{G}, q, g)$  和随机采样  $a, b, c \in \mathbb{Z}_q$  的随机带上.



离散对数类问题的困难性关系如图 ?? 所示:

离散对数问题  $\succeq$  CDH 问题  $\succeq$  DDH 问题

**图 2.6:** 离散对数类问题的困难性关系

**注记 2.7**

注意到任何  $q$  阶循环群均与  $\mathbb{Z}_q$  是同构的, 而  $\mathbb{Z}_q$  上的离散对数问题是容易的. 因此在实例化循环群  $\mathbb{G}$  必须谨慎审慎, 这也从一个方面说明离散对数类问题的困难性与底层代数结构的具体特性(如群的表示)紧密相关. 对于  $\mathbb{G}$  的实例化, 通常既可以选择  $\mathbb{F}_{p^k}^*$  的素数阶乘法子群, 也可以选择椭圆曲线上的素数阶乘法群. 另外强调一点, 存在这样的循环群  $\mathbb{G}$ (如双线性映射群)使得离散对数、CDH 假设成立, 而 DDH 假设不成立.



### 2.3.3 格类假设

1997 年, Shor [Shor-SIAM-1997] 给出了数论类问题(包括整数分解类和离散对数类)的有效量子算法. 在未来, 如果大规模量子计算机研制成功, 则数论类假设将不再成立. 迄今为止, 尚未有针对格基困难问题的有效量子算法, 通用的量子算法仅相对非量子算法有些许优势. 目前普遍的共识是格基困难问题具备抗量子安全能力, 这正是该类问题倍受关注的主要原因.

本小节中将介绍两个主要的平均意义上的格基困难问题, 短整数解问题和带误差学习问题. 需要提前说明的是, 格基困难问题的困难性与参数的选取密切相关, 因此格基问题的描述相比数论类问题要复杂得多.

Ajtai [Ajtai-STOC-1996] 在 1996 年的开创性论文中正式提出了短整数解(short integer solution, SIS)问题. SIS 问题不仅可以作为所有 Minicrypt 世界中密码组件的安全基础, 包括单向函数、身份鉴别协议、数字签名, 还可以用来构造抗碰撞哈希函数. 非正式的, SIS 问题指在给定许多较大的有限加法群中随机选取的元素, 找到足够“短”的整系数组合使得其和是 0 是困难的. SIS 问题由以下参数刻画:

- 正整数  $n$  和  $q$ , 用于刻画加法群  $\mathbb{Z}_q^n$ ;
- 正实数  $\beta$ , 用于刻画解向量的长度;
- 正整数  $m$ , 用于表征群元素的个数.

其中  $n$  是主要的参数(如:  $n \geq 100$ ),  $q > \beta$  通常设定为关于  $n$  的小多项式.

**定义 2.8 (短整数解假设 ( $SIS_{n,q,\beta,m}$ ))**

我们称  $SIS$  假设成立当且仅当对于任意 PPT 敌手:

$$\Pr[\mathcal{A}(\mathbf{a}_1, \dots, \mathbf{a}_m) = \mathbf{z} \neq \mathbf{0} \in \mathbb{Z}^m \text{ s.t. } \sum_i^m \mathbf{a}_i z_i = \mathbf{0} \in \mathbb{Z}_q^n \wedge \|\mathbf{z}\| \leq \beta] \leq \text{negl}(\kappa)$$

上述概率建立在敌手  $\mathcal{A}$  和随机选取  $\mathbf{a}_i \in \mathbb{Z}^m$  的随机带上.



以上定义中  $m$  个  $\mathbb{Z}_q^n$  上的随机向量可以按列向量的方式组成矩阵  $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ . 因此,  $SIS$  假设实质上在要求找到函数  $f_{\mathbf{A}}(\mathbf{z}) := \mathbf{A}\mathbf{z}$  的短整数非零向量原像是困难的.

下面简单讨论参数选取与问题困难性之间的关联:

- 如果不对  $\|\mathbf{z}\|$  进行限制, 那么可以轻易利用 Gaussian 消元法找到一个整数解. 同时, 我们必须要求  $\beta < q$ , 否则  $\mathbf{z} = (q, 0, \dots, 0) \in \mathbb{Z}^m$  即构成一个合法的非平凡解.
- 注意到任何关于矩阵  $\mathbf{A}$  的短整数解可通过补 0 平凡地延展为关于矩阵  $[\mathbf{A} | \mathbf{A}']$  的解. 换言之,  $SIS$  问题的困难性随着  $m$  的增大变得容易. 对应的,  $SIS$  问题的困难性随着  $n$  增加变得困难.
- 向量范数界  $\beta$  和向量  $\mathbf{a}_i$  的个数  $m$  必须足够大以保证解的存在性. 令  $\bar{m}$  是大于  $n \log q$  的最小正整数, 则我们必须有  $\beta > \sqrt{\bar{m}}$  和  $m \geq \bar{m}$ . 不失一般性, 不妨假设  $m = \bar{m}$ , 则存在超过  $q^n$  个向量  $\mathbf{x} \in \{0, 1\}^m$ , 根据鸽巢原理, 则必有  $\mathbf{x} \neq \mathbf{x}'$  使得  $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x}' \in \mathbb{Z}_q^n$ , 从而它们的差值  $\mathbf{z} = \mathbf{x} - \mathbf{x}' \in \{0, \pm 1\}^m$  是范数小于  $\beta$  的短整数解.
- 上述的鸽巢原理论证事实上蕴含更多深意: 函数族  $\{f_{\mathbf{A}} : \{0, 1\}^m \rightarrow \mathbb{Z}_q^n\}$  基于  $SIS$  假设是抗碰撞的. 若不然, 给定关于  $f_{\mathbf{A}}$  的一对碰撞  $\mathbf{x}, \mathbf{x}' \in \{0, 1\}^m$ , 则立刻诱导出关于  $\mathbf{A}$  的一个短整数解.

$SIS$  问题可以被理解为在以下特定  $q$  元  $m$  维整数格中的平均意义短向量问题 (short-vector problem, SVP), 该整数格的定义为:

$$\mathcal{L}^\perp(\mathbf{A}) \stackrel{\text{def}}{=} \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = \mathbf{0} \in \mathbb{Z}_q^n\} \supseteq q\mathbb{Z}^m$$

从编码的角度理解,  $\mathbb{A}$  扮演着格/码字  $\mathcal{L}^\perp(\mathbf{A})$  校验矩阵的角色.  $SIS$  问题的困难性指对于随机选取的  $\mathbf{A}$ , 找到一个短的码字是困难的.

Regev [Regev-STOC-2005] 在 2005 年的开创性论文中提出了另一个平均意义上的重要格基困难问题-带误差学习问题 (learning with errors, LWE). LWE 问题与  $SIS$  问题互相对偶, 能够蕴含 Minicrypt 之外的密码体制.

在正式定义 LWE 问题之前, 首先引入 LWE 分布的概念. 称向量  $\mathbf{s} \in \mathbb{Z}_q^n$  为秘密, LWE 分布  $A_{\mathbf{s}, \chi}$  定义在  $\mathbb{Z}_q^n \times \mathbb{Z}_q$  上, 采样算法为随机选取  $\mathbf{a} \in \mathbb{Z}_q^n$ , 选取  $e \leftarrow \chi$ , 输出  $(\mathbf{a}, b = \langle \mathbf{s}, \mathbf{a} \rangle + e \bmod q)$ .

LWE 问题有两个版本, 其中搜索版本要求给定 LWE 采样求解秘密, 判定版本要求区分 LWE 采样和随机采样. LWE 问题由以下参数刻画:

- 正整数  $n$  和  $q$ , 和  $SIS$  问题一样, 用于刻画加法群  $\mathbb{Z}_q^n$ ;
- 正整数  $m$  表征采样的个数, 通常选取的足够大以保证秘密的惟一性;
- $\mathbb{Z}$  上的误差分布  $\chi$ , 通常的选取是宽度为  $\alpha q$  的离散 Gaussian 分布, 其中  $\alpha < 1$  称为相对错误率.

**定义 2.9 (搜索 LWE 假设)**

搜索 LWE 问题指给定  $m$  个  $A_{\mathbf{s}, \chi}$  的独立随机采样, 求解秘密向量  $\mathbf{s}$  是困难的. 我们称搜索  $LWE$  假设成立当且仅当对于任意 PPT 敌手:

$$\Pr[\mathcal{A}(\{\mathbf{a}_i, b\}_{i=1}^m \leftarrow A_{\mathbf{s}, \chi}) = \mathbf{s}] \leq \text{negl}(\kappa)$$

上述概率建立在敌手  $\mathcal{A}$ 、随机选取  $\mathbf{s} \in \mathbb{Z}_q^n$  和采样  $A_{\mathbf{s}, \chi}$  的随机带上.



**定义 2.10 (判定 LWE 假设)**

判定 LWE 问题指区分  $m$  个独立采样是来自  $A_{\mathbf{s}, \chi}$  分布还是随机分布是困难的。我们称判定 LWE 假设成立当且仅当对于任意 PPT 敌手：

$$|\Pr[\mathcal{A}(\{\mathbf{a}_i, b\}_{i=1}^m \leftarrow A_{\mathbf{s}, \chi}) = 1] - \Pr[\mathcal{A}(\{\mathbf{a}_i, b\}_{i=1}^m \leftarrow U_{\mathbb{Z}_q^n \times \mathbb{Z}_q}) = 1]| \leq \text{negl}(\kappa)$$

上述概率建立在敌手  $\mathcal{A}$ 、随机选取  $\mathbf{s} \in \mathbb{Z}_q^n$  和采样  $A_{\mathbf{s}, \chi}$  以及  $U_{\mathbb{Z}_q^n \times \mathbb{Z}_q}$  的随机带上。

**注记 2.8**

LWE 问题时 LPN 问题 (learning parities with noise) 的一般化。在 LPN 问题中， $q = 2$ ,  $\chi$  为  $\{0, 1\}$  上的 Bernoulli 分布。



下面简单讨论参数选取与问题困难性之间的关联：

- 如果没有误差分布  $\chi$ , 则 LWE 问题的搜索版本和判定版本均可利用 Gaussian 消元法快速求解。
- 和 SIS 问题类似, 可以用矩阵的语言更简洁的描述 LWE 问题: (i) 将  $m$  个向量  $\mathbf{a}_i \in \mathbb{Z}_q^n$  汇聚为矩阵  $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ ; (ii) 将  $m$  个  $b_i \in \mathbb{Z}_q$  汇聚为向量  $\mathbf{b} \in \mathbb{Z}_q^m$ , 因此对于 LWE 采样我们有:

$$\mathbf{b}^t = \mathbf{s}^t \mathbf{A} + \mathbf{e}^t (\bmod q),$$

其中  $\mathbf{e} \leftarrow \chi^m$ .

LWE 问题可以被理解为在以下特定  $q$  元  $m$  维整数格中的平均意义有界距离解码问题 (bounded-distance decoding problem, BDD), 该整数格的定义为:

$$\mathcal{L}(\mathbf{A}) \stackrel{\text{def}}{=} \{\mathbf{A}^t \mathbf{s} : \mathbf{s} \in \mathbb{Z}_q^n\} + q\mathbb{Z}^m$$

从编码的角度理解,  $\mathbf{A}$  扮演着格/码字  $\mathcal{L}(\mathbf{A})$  生成矩阵的角色。对于 LWE 采样,  $\mathbf{b}$  与格中的惟一向量/码字相近, 搜索版本要求计算秘密向量  $s$ , 即根据带误差的码字进行解码。对于随机采样,  $\mathbf{b}$  以大概率远离格  $\mathcal{L}(\mathbf{A})$  中所有向量。SIS 问题的困难性指对于随机选取的  $\mathbf{A}$ , 找到一个短的码字是困难的。

## 2.4 复杂性理论初步

在复杂性理论中, 困难问  $P$  通常定义在  $L \subseteq X$  上,  $X$  是所有实例的集合,  $L$  是  $X$  中满足特定性质的一个子集. 我们称  $P$  是可高效判定/求解的, 如果存在确定性时间的 Turing 机  $M$  满足:

$$x \in L \iff M(x) = 1$$

所有可高效判定/求解问题的合集组成  $\mathcal{P}$  复杂性类.

**笔记** 实例集合  $X$  的学术术语是词 (words), 子集  $L$  对应的学术术语是语言 (language). 术语的来源是一个自然的类比: 不妨设世界上所有的词汇构成一个集合, 那么汉语、英语、法语、德语、C++ 语言、Rust 语言等多种多样的语言自然构成了这个集合的各个子集. 通常, 称语言内的元素为 Yes 实例, 语言外的元素为 No 实例.

密码学中的困难问题可以分为计算和判定两类:

- 计算类(也称搜索类) 问题要求计算出问题的解: 如 RSA 问题、离散对数问题、计算 Diffie-Hellman 问题、短整数解问题等.
- 判定类问题要求判定是或否: 如二次剩余问题、判定 Diffie-Hellman 问题、判定 LWE 问题等.

从解空间的角度理解, 判定问题可以看做计算问题的特例, 即输出解为 1 比特. 通常, 同一个问题的计算版本难于判定版本, 对应的计算假设弱于判定假设.

困难的二元关系是对密码学中各种计算类困难问题的抽象.

### 定义 2.11 (二元关系 (binary relation))

令  $L \subseteq X$  是一个  $\mathcal{NP}$  语言.  $L$  由二元关系  $R_L : X \times W$  定义, 其中  $W$  之证据集合:

$$x \in L \Leftrightarrow \exists w \in W \text{ s.t. } (x, w) \in R_L$$

如果  $R_L$  满足如下两个性质, 则称其是困难的 (hard):

- 易采样 (easy to sample):  $\exists$  PPT 算法 SampR 对关系  $R_L$  进行随机采样, 其以公开参数  $pp$  为输入, 输出“实例-证据”元组  $(x, w) \in R_L$ .
- 难抽取 (hard to extract):  $\forall$  PPT 敌手  $\mathcal{A}$ :

$$\Pr[(x, \mathcal{A}(x) = w') \in R_L : (x, w) \leftarrow \text{SampR}(pp)] \leq \text{negl}(\kappa)$$

**笔记** 单向函数自然诱导了一个困难的二元关系. 易采样的性质由单向函数的定义域可高效采样和单向函数可高效求值两点保证, 难抽取的性质由单向函数的单向性保证.

问题任务: 计算/搜索      解空间:  $\{0, 1\}^{\text{poly}(\kappa)}$

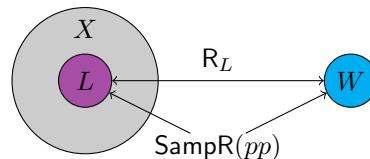


图 2.7: 计算类困难问题图示

子集成员判定问题 (SMP, Subset Membership Problem) 则是密码学中各种判定类问题的抽象.

**定义 2.12 (子集成员判定问题)**

令  $L \subset X$  是一个语言, 公开参数是  $pp$ . 定义以 3 个 PPT 采样算法:

- $\text{SampAll}(pp)$ : 输出  $X$  中的随机元素.
- $\text{SampYes}(pp)$ : 输出  $L$  中的随机元素, 即随机 Yes 实例.
- $\text{SampNo}(pp)$ : 输出  $X \setminus L$  中的随机元素, 即随机 No 实例.

SMP 问题有两种类型:

- Type 1:  $U_X \approx_c U_L$
- Type 2:  $U_{X \setminus L} \approx_c U_L$

**注记 2.9**

定义  $\rho = |L|/|X|$  为语言  $L$  相对于  $X$  的密度. 容易证明:

- 当  $\rho = \text{negl}(\kappa)$  时: Type 1  $\iff$  Type 2
- 当  $\rho$  已知时: Type 2  $\Rightarrow$  Type 1
  - 归约的方法是对给定分布和  $U_L$  分布根据  $\rho$  进行加权重构: 如果给定分布是  $U_{X \setminus L}$ , 则重构结果  $U_X$ ; 如果给定分布是  $U_L$ , 则重构结果仍为  $U_L$ . 因此, Type 2 的实例可以归约到 Type 1 的实例.



**问题任务:** 区分/判定      **解空间:**  $\{0, 1\}$

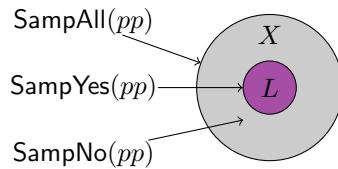


图 2.8: 判定类计算问题示例

## 2.5 信息论工具

### 2.5.1 熵的概念

Shannon 在 1948 年开创了信息论这一全新领域, 其工作的开创性不啻于 Einstein 关于引力场理论的工作, 一举奠定了学科的基础, 解答了最初所有最重要的问题.

Shannon 的信息论关注的重点是“消息”和它们在(有噪)信道中的传播. 容易直观理解的是, 消息所包含的信息量取决于消息令人感到意外的程度(如果消息平平无奇, 那么它包含的信息为 0). 相比而言, 稍显反直觉的是随机消息包含最多的信息.

信息论中最重要的概念是熵(entropy). 熵量化了期望的意义下消息包含的信息量, 单位通常是比特. 从概率论的角度看, 熵是对随机变量不确定性的测度. 以下令令  $X$  是定义在  $\Omega$  上的随机变量.

#### 定义 2.13 (熵)

$X$  的熵刻画了平均意义下  $X$  取值的(不)可预测性:

$$H(X) = - \sum_{\omega \in \Omega} \Pr[X = \omega] \log \Pr[X = \omega]$$

#### 注记 2.10

一个消息的熵就是消息所包含信息的比特数, 用编码的语言刻画, 就是编码该消息所需的最短比特数.

密码方案/协议的安全性分析均是针对恶意敌手展开的. 敌手单次正确预测某随机变量(如私钥)值的概率与密码方案/协议的安全性紧密相关. 显然, 敌手的最佳策略是猜测最大似然值. 在本章中, 我们用大写字母  $X$  表示随机变量, 用小写字母  $x$  表示  $X$  的取值, 用花体字母  $\mathcal{X}$  表示  $X$  的支撑集.

一个随机变量  $X$  的最大可预测性是  $\max_{\omega \in \Omega} \Pr[X = \omega]$ . 最大可预测性对应最小熵(min-entropy), 严格定义如下:

#### 定义 2.14 (最小熵)

$X$  的最小熵刻画了  $X$  的最大可预测性:

$$H_{\infty}(X) = - \log \left( \max_{\omega \in \Omega} \Pr[X = \omega] \right)$$

#### 注记 2.11

最小熵可以看做“最坏情形”(worst-case)的熵.

在很多场景中, 随机变量  $X$  与另一随机变量  $Y$  相关, 并且敌手知晓  $Y$  的取值. 因此, Dodis 等 [DORS-SIAM-2008] 引入了平均最小熵(average min-entropy)来刻画  $X|Y$  的(不)可预测性:

$$\tilde{H}_{\infty}(X|Y) = - \log \left( \mathbb{E}_{y \leftarrow Y} \left[ 2^{H_{\infty}(X|Y=y)} \right] \right) = - \log \left( \mathbb{E}_{y \leftarrow Y} \left[ \max_{\omega \in \Omega} \Pr[X = \omega | Y = y] \right] \right) \quad (2.1)$$

以下浅释平均最小熵的定义直觉. 考虑一对变量  $X$  和  $Y$ (两者可能相关). 如果敌手知晓  $Y$  的取值  $y$ , 则  $X$  在敌手视角中的可预测性是  $\max_x \Pr[X = x | Y = y]$ . 在平均的意义下(对  $Y$  做期望), 敌手成功预测  $X$  的概率为  $\mathbb{E}_{y \leftarrow Y} [\max_x \Pr[X = x | Y = y]]$ .

平均最小熵的定义在对  $Y$  做加权平均的前提下( $Y$  的取值不受敌手控制)测度  $X$  最坏情形下的可预测性(敌手知晓  $y$  后对  $X$  的预测是恶意行为). 一个微妙的细节是平均最小熵的定义(?)先对预测成功的概率做期望后再取对数, 那能否交换  $\log$  和  $\mathbb{E}$  的次序呢? 定义平均最小熵  $\mathbb{E}_{y \leftarrow Y} [H_{\infty}(X|Y=y)]$  是否合理呢? 交换次序后的定义

失去了原本的意义. 考虑以下的例子, 令  $X$  和  $Y$  都是定义在  $\Omega = \{0, 1\}^{1000}$  上的随机变量,  $Y$  是  $\Omega$  上的随机分布, 当  $Y$  的取值  $y$  的首 bit 为 0 时,  $X$  的取值与  $y$  相同, 否则随机分布. 因此对于  $Y$  的半数取值  $y$ ,  $H_\infty(X|Y=y) = 0$ , 对另外半数取值,  $H_\infty(X|Y=y) = 1000$ , 所以  $\mathbb{E}_{y \leftarrow Y}[H_\infty(X|Y=y)] = 500$ . 然而, 声称  $X$  具有 500 比特的安全性显然不符合逻辑. 事实上, 知晓  $Y$  取值  $y$  的敌手直接输出  $y$ , 既能够以大于  $1/2$  的概率猜对  $X$  的取值. 平均最小熵标准的定义准确刻画了至少  $1/2$  得可预测性, 因为  $\tilde{H}_\infty(X|Y)$  略小于 1. 我们也可以从数学的角度解释如下,  $\mathbb{E}$  是线性算子, 而  $\log$  是非线性算子, 因此次序交换后意义不同.

平均最小熵和最小熵之间存在何种关系呢? Dodis 等 [DORS-SIAM-2008] 证明了如下的 cChaining Lemma, 建立了两者之间的关系, 给出了平均最小熵的一个下界.

### 引理 2.2 (Chaining Lemma)

令  $X, Y$  和  $Z$  是三个随机变量(可任意相关), 其中  $Y$  的支撑集包含至多  $2^r$  个元素. 我们有  $\tilde{H}_\infty(X|(Y, Z)) \geq H_\infty(X|Z) - r$ . 特别的, 当  $Z$  为空时, 上述不等式简化为:  $\tilde{H}_\infty(X|Y) \geq H_\infty(X) - r$ .



## 2.5.2 随机性提取

随机性是密码学的主旋律, 几乎所有已知密码方案/协议都离不开均匀随机采样. 然而, 均匀无偏的完美信源不易得, 很多场景下存在的是有偏的弱信源. 如何在信源有偏的情况下进行均匀随机采样呢? 这就是随机性提取器所要完成的工作.

### 定义 2.15 (强随机性提取器)

令  $X$  是最小熵  $H_\infty(X) \geq n$  的随机变量,  $\text{ext} : \mathcal{X} \times \mathcal{S} \rightarrow \mathcal{Y}$  是一个可高效计算的函数. 我们称  $\text{ext}$  是对信源  $X$  的  $(n, \epsilon)$ -强随机性提取器当且仅当以下成立:

$$\Delta((\text{ext}(X, S), S), (Y, S)) \leq \epsilon,$$

其中  $S$  是定义在  $\mathcal{S}$  上的均匀随机变量,  $Y$  是定义在  $\mathcal{Y}$  上的均匀随机变量.



类比于平均最小熵和最小熵之间的关系, 当信源  $X$  与另一变量  $Z$  相关时, 我们需要引入平均强随机性提取器来对信源  $X$  进行萃取.

### 定义 2.16 (平均强随机性提取器)

令  $(X, Z)$  是满足约束  $\tilde{H}_\infty(X|Z) \geq n$  的任意变量对,  $\text{ext} : \mathcal{X} \times \mathcal{S} \rightarrow \mathcal{Y}$  是一个可高效计算的函数. 我们称  $\text{ext}$  是对信源  $X$  的平均意义  $(n, \epsilon)$ -强随机性提取器当且仅当以下成立:

$$\Delta((\text{ext}(X, S), S, Z), (Y, S, Z)) \leq \epsilon,$$

其中  $S$  是定义在  $\mathcal{S}$  上的均匀随机变量,  $Y$  是定义在  $\mathcal{Y}$  上的均匀随机变量.



Dodis 等 [DORS-SIAM-2008] 的 Leftover Hash Lemma(剩余哈希引理) 证明了任何强随机性提取性在适当的参数设定下都是平均强随机性提取器. 作为一个特例, Dodis 等证明了任何一族一致哈希函数 (universal hash functions) 都是平均强随机性提取器.

### 引理 2.3 (Leftover Hash Lemma)

令  $X$  和  $Z$  是满足约束  $\tilde{H}_\infty(X|Z) \geq n$  的任意变量对,  $\mathcal{H} = \{h_s : \mathcal{X} \rightarrow \mathcal{Y}\}_{s \leftarrow S}$  是一族一致哈希函数. 那么当  $n \geq \log |\mathcal{Y}| + 2 \log(1/\epsilon)$  时,  $\text{ext}(x, s) := h_s(x)$  是  $(n, \epsilon)$ -平均强随机性提取器.



## 2.6 公钥加密基本安全模型

### 2.6.1 公钥加密方案

公钥加密的概念由 Diffie 和 Hellman [DH-IEEE-IT-1976] 在 1976 年的划时代论文中正式提出, 其与对称加密的最大不同在于每个用户自主生成一对密钥, 公钥用于加密、私钥用于解密, 发送方仅需知晓接收方的公钥即可向接收方发送密文.

#### 定义 2.17 (公钥加密方案)

正式的, 公钥加密方案由以下的四个多项式时间组成:

- $\text{Setup}(1^\kappa)$ : 系统生成算法以安全参数  $1^\lambda$  为输入, 输出系统公开参数  $pp$ , 其中  $pp$  包含对公钥空间  $PK$ 、私钥空间  $SK$ 、明文空间  $M$  和密文空间  $C$  的描述. 该算法由可信第三方生成并公开, 系统中的所有用户共享, 所有算法均将  $pp$  作为输入. 当上下文明确时, 常常为了行文简洁省去  $pp$ .
- $\text{KeyGen}(pp)$ : 密钥生成算法以系统公开参数  $pp$  为输入, 输出一对公/私钥对  $(pk, sk)$ , 其中公钥公开, 私钥秘密保存.
- $\text{Encrypt}(pk, m; r)$ : 加密算法以公钥  $pk \in PK$ 、明文  $m \in M$  为输入, 输出密文  $c \in C$ .
- $\text{Decrypt}(sk, c)$ : 解密算法以私钥  $sk \in SK$  和密文  $c \in C$  为输入, 输出明文  $m \in M$  或者  $\perp$  表示密文非法. 解密算法通常为确定性算法.



**正确性.** 该性质保证公钥加密的功能性, 即使用私钥可以正确恢复出对应公钥加密的密文. 正式的, 对于任意明文  $m \in M$ , 有:

$$\Pr[\text{Decrypt}(sk, \text{Encrypt}(pk, m)) = m] = 1 - \text{negl}(\kappa). \quad (2.2)$$

公式 (2.2) 的概率建立在  $\text{Setup}(1^\kappa) \rightarrow pp$ 、 $\text{KeyGen}(pp) \rightarrow (pk, sk)$  和  $\text{Encrypt}(pk, m) \rightarrow c$  的随机带上. 如果上述概率严格等于 1, 则称公钥加密方案满足完美正确性.

#### 注记 2.12

通常基于数论假设的公钥加密方案满足完美正确性, 而格基方案由于底层困难问题的误差属性, 解密算法存在可忽略的误差.

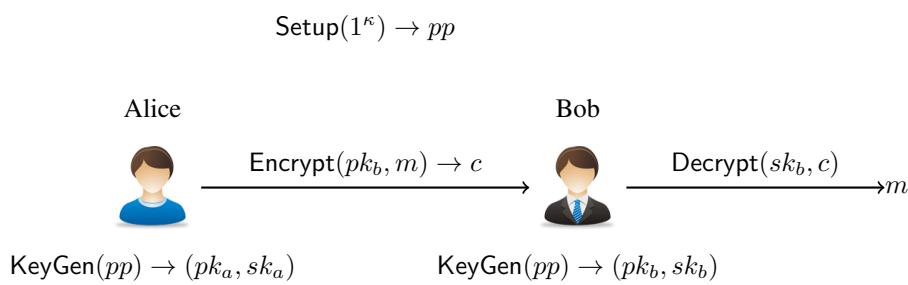


图 2.9: 公钥加密方案示意图

**安全性.** 定义公钥加密方案敌手  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  的优势函数如下:

$$\text{Adv}_{\mathcal{A}}(\kappa) = \Pr \left[ \beta' = \beta : \begin{array}{l} pp \leftarrow \text{Setup}(1^\lambda); \\ (pk, sk) \leftarrow \text{KeyGen}(pp); \\ (m_0, m_1, state) \leftarrow \mathcal{A}_1^{\text{O}_{\text{decrypt}}(\cdot)}(pp, pk); \\ \beta \xleftarrow{R} \{0, 1\}; \\ c^* \leftarrow \text{Encrypt}(pk, m_\beta); \\ \beta' \leftarrow \mathcal{A}_2^{\text{O}_{\text{decrypt}}(\cdot)}(pp, pk, state, c^*); \end{array} \right] - \frac{1}{2},$$

在上述定义中,  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  表示敌手  $\mathcal{A}$  可划分为两个阶段, 划分界线是接收到挑战密文  $c^*$  前后,  $state$  表示  $\mathcal{A}_1$  向  $\mathcal{A}_2$  传递的信息, 记录部分攻击进展.  $\mathcal{O}_{\text{decrypt}}(\cdot)$  表示解密预言机, 其在接收到密文  $c$  的询问后输出  $\text{Decrypt}(sk, c)$ . 如果任意的 PPT 敌手  $\mathcal{A}$  在上述游戏中的优势函数均为可忽略函数, 则称公钥加密方案是 IND-CPA 安全的; 如果任意的 PPT 敌手在阶段 1 可自适应访问  $\mathcal{O}_{\text{decrypt}}(\cdot)$  的情形下仍仅具有可忽略优势, 则称公钥加密方案是 IND-CCA1 安全的; 如果任意的 PPT 敌手在阶段 1 和阶段 2 均可自适应访问  $\mathcal{O}_{\text{decrypt}}(\cdot)$  的情形下仍仅具有可忽略优势, 则称公钥加密方案是 IND-CCA2 或 IND-CCA 安全的.

以下阐述公钥加密安全性定义的一些细微之处:

- 自适应的含义是敌手的攻击行为可根据学习到的知识动态调整, 如我们称敌手能够自适应的访问解密预言机指敌手可以根据历史询问结果发起新的询问. 简而言之, 自适应性极大的增强了敌手的攻击能力.
- IND-CCA 安全性远强于 IND-CCA1 和 IND-CPA 安全性, 这是因为敌手可以在观察到挑战密文  $c^*$  后有针对性的发起更加有威胁的解密询问.
- $(m_0, m_1)$  由敌手任意选择, 从而巧妙精准的刻画了密文不泄漏明文任何一比特信息的直觉.
- 为了避免定义无意义, 在 IND-CCA 的安全游戏中禁止敌手在第二阶段向  $\mathcal{O}_{\text{decrypt}}(\cdot)$  询问挑战密文  $c^*$ .

 **笔记** 对于密码方案, 给出恰当的安全性定义非常重要: 一方面安全性定义必须足够强以刻画现实中存在的攻击, 另一方面安全性定义不能过强使得无法构造满足其的密码方案. 公钥加密的安全性定义是逐渐演化的.

上世纪 70 年代, Diffie 和 Hellman 提出了公钥加密的概念, 随后 Rivest、Shamir 和 Adi 构造出了首个公钥加密方案——RSA 加密. 在这一阶段, 公钥加密的安全性仅具备符合直觉的单向性, 即在平均意义上从密文中恢复出明文是计算困难的. 到了上世纪 80 年代, 人们逐渐认识到单向性并不能满足应用需求, 这是因为对于单向安全的公钥加密方案, 敌手有可能从密文恢复出明文的部分信息, 而在应用中, 由于数据来源的多样性和不确定性, 明文的每一比特都可能包含关键的机密信息(比如股票交易指令中的“买”或“卖”).

1982 年, Goldwasser 和 Micali [GM-STOC-1982] 指出单向安全的不足, 提出了语义安全性(semantic security). 语义安全性的直观含义是密文对敌手求解明文没有帮助. 严格定义颇为精妙, 定义的形式是基于模拟的, 即敌手掌握密文的视角可以由一个 PPT 的模拟器在计算意义上模拟出来. 语义安全性可以看做 Shannon 完美安全性在计算意义的推广放松, 然而在论证的时候稍显笨重.

Goldwasser 和 Micali 给出了另一个等价的定义(等价性的证明参见 Dodis 和 Ruhl 的短文 [DR-Web-1999]), 即选择明文攻击下的不可区分性(IND-CPA, indistinguishability against chosen-plaintext attack). IND-CPA 安全定义的直觉是密文在计算意义上不泄漏明文的任意一比特信息, 即对任意两个明文对应的密文分布是计算不可区分的, 其中选择明文攻击刻画了公钥公开特性使得任意敌手均可通过自行加密获得任意明文对应密文这一事实. 使用 IND-CPA 安全进行安全论证相比语义安全要便捷很多, 因此被广为采用.

注意到 IND-CPA 安全仅考虑被动敌手, 即敌手只窃听信道上的密文. 1990 年, Naor 和 Yung [NY-STOC-1990] 认为敌手有能力发起一系列主动攻击, 比如重放密文、修改密文等, 进而提出选择密文攻击(CCA, chosen-ciphertext attack)刻画这一系列主动攻击行为, 即敌手可以自适应的获取指定密文对应的明文. Naor 和 Yung 考虑了两种选择密文攻击, 一种是弱化版本, 称为午餐时间攻击(lunch-time attack), 含义是敌手只能在极短的时间窗口(收到挑战密文之前)进行选择密文攻击; 另一种是标准版本, 敌手可以长时间窗口(收到挑战密文前后)进行选择密文攻击.

1998 年, Bleichenbacher [Bleichenbacher-CRYPTO-1998] 展示了针对 PKCS#1 标准中公钥加密方案的有效选择密文攻击, 实证了关于选择密文安全的研究并非杞人忧天. Shoup [Shoup-TechReport-1998] 进一步深入探讨了选择密文安全的重要性与必要性. 从此, IND-CCA 安全成为了公钥加密方案的事实标准.

事实上, 公钥加密还存在另外一种更符合直觉的安全定义, 大意是对于明文空间中的随机明文进行加密, 敌手正确猜测出加密明文的概率与随机猜测的正确概率相近. 我们称这种安全性为消息恢复选择明文安全(message-recovery-CPA security), 严格定义如下:

**消息恢复选择明文安全.** 令  $M$  是明文空间. 定义公钥加密方案敌手  $\mathcal{A}$  的优势函数如下:

$$\text{Adv}_{\mathcal{A}}(\lambda) = \Pr \left[ \begin{array}{l} pp \leftarrow \text{Setup}(1^\kappa); \\ (pk, sk) \leftarrow \text{KeyGen}(pp); \\ i = i' : M' = \{m_1, \dots, m_n\} \subseteq M \leftarrow \mathcal{A}, |M'| \geq 2; \\ i \xleftarrow{\text{R}} [n]; \\ c^* \leftarrow \text{Encrypt}(pk, m_i); \\ i' \leftarrow \mathcal{A}(pk, M', c^*); \end{array} \right] - \frac{1}{n}$$

如果任意的 PPT 敌手  $\mathcal{A}$  在上述游戏中的优势函数均为可忽略函数, 则称公钥加密方案是 Message-Recovery-CPA 安全的.

### 定理 2.1

公钥加密的 IND-CPA 与 Message-Recovery-CPA 安全性是等价的.



### 证明

Message-Recovery-CPA  $\Rightarrow$  IND-CPA. 容易看出, 当限定  $|M'| = 2$  时, MessageRecovery-CPA 蕴含 IND-CPA. 令  $\mathcal{A}$  是针对 IND-CPA 安全的敌手, 以下展示如何基于  $\mathcal{A}$  构造  $\mathcal{B}$  打破 MessageRecovery-CPA 安全.  $\mathcal{B}$  选择两个互异的明文  $m_0, m_1$ , 提交  $M' = \{m_0, m_1\}$ , 获得  $c^* \leftarrow \text{Encrypt}(pk, m_\beta)$  作为挑战.  $\mathcal{B}$  将  $c^*$  发送给  $\mathcal{A}$ , 并将  $\mathcal{A}$  的输出作为自己的输出发送给 MessageRecovery-CPA 的挑战者. 显然,  $\mathcal{B}$  完美的模拟了 IND-CPA 安全实验,  $\text{Adv}_{\mathcal{B}}(\kappa) = \text{Adv}_{\mathcal{A}}(\kappa)$ .

IND-CPA  $\Rightarrow$  Message-Recovery-CPA. 我们通过以下的游戏序列完成证明:

Game 0. 对应标准的 MessageRecovery-CPA 安全实验. 在挑战阶段,  $\mathcal{CH}$  在收到敌手选择的  $M'$  后, 随机选择  $i \xleftarrow{\text{R}} [n]$ , 发送  $c^* \leftarrow \text{Encrypt}(pk, m_i)$  to  $\mathcal{A}$ . 根据定义, 我们有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/|M'||$$

Game 1. 与标准的 Message-Recovery-CPA 安全实验基本相同, 唯一的不同是在挑战阶段收到  $M'$  后,  $\mathcal{CH}$  随机选择  $i \xleftarrow{\text{R}} [n], j \xleftarrow{\text{R}} [n]$ , 发送  $c^* \leftarrow \text{Encrypt}(pk, m_j)$  to  $\mathcal{A}$ . 因为  $i$  在信息论意义上隐藏于  $\mathcal{A}$ , 因此我们有:

$$\Pr[S_1] = |1/|M'||$$

此处需要特别强调上述概率仅建立在  $\mathcal{CH}$  随机选择  $i$  的随机带上.

最后, 我们证明 IND-CPA 安全保证了  $|\Pr[S_0] - \Pr[S_1]| = \text{negl}(\kappa)$ .

令  $\mathcal{B}$  是针对 IND-CPA 安全的敌手.  $\mathcal{B}$  随机选择  $i \xleftarrow{\text{R}} [n], j \xleftarrow{\text{R}} [n]$ , 令  $m_0 = m_i, m_1 = m_j$ , 将  $(m_0, m_1)$  提交给它的挑战者并获得关于  $m_\beta$  的挑战密文  $c^*$ .  $\mathcal{B}$  发送挑战密文  $c^*$  给  $\mathcal{A}$ ,  $\mathcal{A}$  输出对  $i$  的猜测  $i'$ . 如果  $i' = i$ ,  $\mathcal{B}$  输出 0. 否则,  $\mathcal{B}$  输出对  $\beta$  的随机猜测. 当  $\beta = 0$  时,  $\mathcal{B}$  向  $\mathcal{A}$  完美的模拟了 Game<sub>0</sub>, 因此正确输出  $\beta$  的概率是  $\Pr[S_0]$ ; 当  $\beta = 1$  时,  $\mathcal{B}$  向  $\mathcal{A}$  完美的模拟了 Game<sub>1</sub>, 因此正确输出  $\beta$  的概率是  $(1 - \Pr[S_1])$ ; 我们有:

$$\begin{aligned} \text{Adv}_{\mathcal{B}}(\kappa) &= |\Pr[\beta = 0] \Pr[S_0] + \Pr[\beta = 1](1 - \Pr[S_1]) - 1/2| \\ &= \left| \frac{1}{2} (\Pr[S_0] - \Pr[S_1]) \right| \end{aligned}$$

IND-CPA 的安全性保证了  $\text{Adv}_{\mathcal{B}}(\kappa)$  可忽略, 因此  $|\Pr[S_0] - \Pr[S_1]| \leq \text{negl}(\kappa)$ . 综上我们有  $\text{Adv}_{\mathcal{A}}(\kappa) = \text{negl}(\kappa)$ .

以上展示了 Message-Recovery-CPA 和 IND-CPA 的双向蕴含, 定理得证.

### 2.6.1.1 基本性质

**同态性.** 公钥加密方案的正确性隐式保证了解密算法自然诱导出从密文空间  $C$  到明文空间  $M$  的一个映射  $\phi = \text{Dec}(sk, \cdot)$ . 如果  $\phi$  具备同态性, 则第三方可对密文进行相应的公开计算, 得到的密文与对明文施加同样计算所得结果对应. 正式的, 令  $\mathcal{C} = \{f\}$  是从  $M^n \rightarrow M$  的某个电路族, 其中  $n$  是正整数;  $\text{Eval}$  为密文求值算法, 以公钥  $pk$ 、 $f \in \mathcal{C}$  和密文向量  $\mathbf{c} = (c_1, \dots, c_n)$  为输入, 输出  $c' \in C$ , 记作  $c' \leftarrow \text{Eval}(pk, f, \mathbf{c})$ . 如果对于任意  $f \in \mathcal{C}$  和任意明文  $\mathbf{m} = (m_1, \dots, m_n) \in M^n$  以下公式成立:

$$\Pr \left[ \begin{array}{l} (pk, sk) \leftarrow \text{KeyGen}(1^\kappa); \\ \text{Dec}(sk, c') = f(\mathbf{m}): \mathbf{c} \leftarrow \text{Enc}(pk, \mathbf{m}); \\ c' \leftarrow \text{Eval}(pk, f, \mathbf{c}); \end{array} \right] = 1$$

则称公钥加密方案是  $\mathcal{C}$ -同态的,  $\mathcal{C}$  刻画了同态所支持的公开计算类型. 两种常见的同态类型如下:

- 部分同态 (partially homomorphic): 不失一般性, 若明文空间  $M$  为加法群, 密文空间  $C$  为乘法群, 若  $\mathcal{C}$  仅包含  $M^2 \rightarrow M$  的群运算, 则称加密方案是部分同态或者加法同态的. 此时同态性刻画如下:

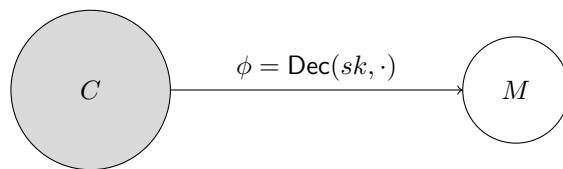
$$\Pr \left[ \text{Dec}(sk, c_1 \cdot c_2) = f(m_1, m_2): \begin{array}{l} (pk, sk) \leftarrow \text{KeyGen}(1^\kappa); \\ c_1 \leftarrow \text{Enc}(pk, m_1), c_2 \leftarrow \text{Enc}(pk, m_2); \end{array} \right] = 1$$

- 全同态 (fully homomorphic): 若  $\mathcal{C}$  包含了  $M^n \rightarrow M$  的所有多项式时间可计算函数, 则称方案是全同态的.

 **笔记** 几乎所有公钥加密方案都构建在代数性质良好的结构上, 且大部分方案均天然满足部分同态, 如 RSA、ElGamal、Goldwasser-Micali、Benaloh、Paillier、Sander-Young-Yung、Boneh-Goh-Nissim、Ishai-Paskin 等. 在 RSA 公钥加密方案横空出世仅一年后, Rivest、Adleman 和 Dertouzos [RAD-FOSC-1978] 即提出了全同态公钥加密的概念. 直到 31 年后, 才由 Gentry [Gentry-STOC-2009] 通过引入理想格构给出首个全同态加密方案的构造. 自此突破之后, 全同态加密迅猛发展, 理论成果百花齐放, 效率不断提升, 成为了隐私保护技术中重要且实用的密码学工具. 感兴趣的读者请参阅 Halevi 的综述文章 [Halevi-Tutorial-2017].

#### 注记 2.13

对于密码方案和协议, 安全性和功能效率之间通常存在权衡关系 (trade-off). 对于公钥加密方案, CPA 安全性与同态性可以共存, 而更强的 CCA 安全性与同态性之间就存在冲突, 无法兼得. 在现实世界中应用公钥加密方案时, 需根据应用场景的具体需求在安全性和功能效率之间做出恰当的选择, 切不可教条.



### 2.6.2 密钥封装机制

主流的公钥加密方案基于数论或者格基困难问题构造. 基于数论问题的公钥加密方案因需要进行高精度算术运算导致加解密速率较低, 基于格基困难问题的公钥加密方案存在公钥和密文尺寸较大的问题. 而对称加密方案因其功能简单, 仅需异或等逻辑运算即可完成, 且硬件支持良好 (如定制的指令), 因此在较公钥加密具有较大的性能优势, 在加密长明文的场景下更为显著.

如何解决公钥加密在加密长消息时的性能短板呢? 解决思路是混合加密 (hybrid encryption), 朴素的实现方式是 PKE+SKE, 如图 ?? 所示:

1. 发送方首先随机选择对称密钥  $k$ , 调用公钥加密算法用接收方的公钥  $pk$  加密  $k$  得到  $c$ , 再调用对称加密算法用  $k$  加密明文  $m$  得到  $c_1$ , 最终的密文  $(c, c')$ .

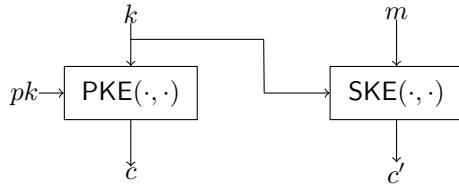


图 2.10: 混合加密: PKE+SKE

2. 接收方在接收到密文  $(c, c')$  后, 首先使用私钥  $sk$  解密  $c$  恢复对称密钥  $k$ , 再使用  $k$  解密  $c'$ .

混合加密方法既保留了公钥加密的功能性, 同时性能几乎与对称加密相当, 因此是公钥加密加密长明文时的通用范式. Cramer 和 Shoup [CS-EUROCRYPT-2002] 观察到公钥加密在混合加密范式中起到的关键作用是发送方向接收方传输对称密钥, 而传递的方式并非必须是加解密. 基于该观察, Cramer 和 Shoup 提出了“密钥封装-数据封装”范式, 简称为 KEM-DEM(key/data-encapsulation mechanism), 该范式可以看作是混合加密的另一种实现方式, 如图 ?? 所示. 顾名思义, KEM-DEM 范式包含 KEM 和 DEM 两个组件, DEM 可以粗略的等同为对称加密, KEM 是该范式的核心. 简言之, KEM 与 PKE 的不同在于发送方不再先显式选择对称密钥再加密, 而是封装一个随机的对称密钥.

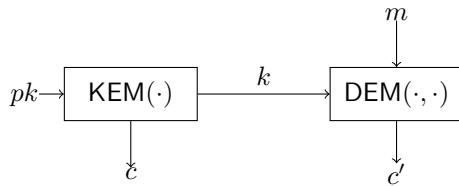


图 2.11: 混合加密: KEM+DEM

### 定义 2.18 (密钥封装机制)

正式的, KEM 由以下的四个多项式时间组成:

- $\text{Setup}(1^\kappa)$ : 系统生成算法以安全参数  $1^\kappa$  为输入, 输出系统公开参数  $pp$ , 其中  $pp$  包含对公钥空间  $PK$ 、私钥空间  $SK$ 、对称密钥空间  $K$  和密文空间  $C$  的描述. 该算法由可信第三方生成并公开, 系统中的所有用户共享, 所有算法均将  $pp$  作为输入. 当上下文明确时, 常常为了行文简洁省去  $pp$ .
- $\text{KeyGen}(pp)$ : 密钥生成算法以系统公开参数  $pp$  为输入, 输出一对公/私钥对  $(pk, sk)$ , 其中公钥公开, 私钥秘密保存.
- $\text{Encaps}(pk; r)$ : 封装算法以公钥  $pk \in PK$  为输入, 输出对称密钥  $k \in K$  和封装密文  $c \in C$ .
- $\text{Decaps}(sk, c)$ : 解封装算法以私钥  $sk \in SK$  和密文  $c \in C$  为输入, 输出对称密钥  $k \in K$  或者  $\perp$  表示封装密文非法. 解封装算法通常为确定性算法.

### 注记 2.14

在 KEM 中, 对称密钥  $k$  起到的作用是在发送方和接收方之间建立安全的会话信道, 因此也常称为会话密钥.

**正确性.** 该性质保证 KEM 的功能性, 即使用私钥可以正确恢复出封装密文所封装的会话密钥. 正式的, 对于任意会话密钥  $k \in K$ , 有:

$$\Pr[\text{Decaps}(sk, c) = k : (c, k) \leftarrow \text{Encaps}(pk)] = 1 - \text{negl}(\kappa). \quad (2.3)$$

公式 (??) 的概率建立在  $\text{Setup}(1^\kappa) \rightarrow pp$ 、 $\text{KeyGen}(pp) \rightarrow (pk, sk)$  和  $\text{Encaps}(pk) \rightarrow (c, k)$  的随机带上. 如果上述概率严格等于 1, 则称 KEM 方案满足完美正确性.

安全性. 定义 KEM 敌手  $\mathcal{A}$  的优势函数如下:

$$\text{Adv}_{\mathcal{A}}(\kappa) = \Pr \left[ \beta' = \beta : \begin{array}{l} pp \leftarrow \text{Setup}(1^\lambda); \\ (pk, sk) \leftarrow \text{KeyGen}(pp); \\ (c^*, k_0^*) \leftarrow \text{Encaps}(pk), k_1^* \xleftarrow{\text{R}} K; \\ \beta \xleftarrow{\text{R}} \{0, 1\}; \\ \beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{decaps}}(\cdot)}(pp, pk, c^*, k_\beta^*); \end{array} \right] - \frac{1}{2},$$

在上述定义中,  $\mathcal{O}_{\text{decaps}}(\cdot)$  表示解封装谕言机, 其在接收到密文  $c$  的询问后输出  $\text{Decaps}(sk, c)$ . 如果任意的 PPT 敌手  $\mathcal{A}$  在上述游戏中的优势函数均为可忽略函数, 则称 KEM 方案是 IND-CPA 安全的; 如果任意的 PPT 敌手在可自适应访问  $\mathcal{O}_{\text{decaps}}(\cdot)$  的情形下仍仅具有可忽略优势, 则称 KEM 方案是 IND-CCA 安全的.

下面给出安全性定义的一些注记:

- KEM 的安全游戏中分阶段定义敌手不再必要, 因为挑战密文的生成不受敌手控制, 正是这点不同使得 KEM 的安全性定义要比 PKE 的安全性定义简单.
- 为了避免定义无意义, 在 IND-CCA 的安全游戏中禁止敌手向  $\mathcal{O}_{\text{decaps}}(\cdot)$  询问挑战密文  $c^*$ .

补充 DEM 部分和 KEM+DEM 的安全性组合

### 2.6.3 两类混合加密范式的比较

以上两种混合方法的共性都是首先生成对称密钥, 再利用对称密钥加密明文, 因此效率方面的差异体现在第一阶段. PKE+SKE 范式的非对称部分是先选择一个随机的密钥  $k$ , 再使用 PKE 对其加密得到  $c$ , 而 KEM+DEM 范式的非对称部分是两步并做一步完成. 如果使用 PKE-SKE 范式, 密文  $c$  必然存在密文扩张, 这是由概率加密的本质决定的; 而如果使用 KEM+DEM 的方法, 密文  $c$  相比  $k$  可能不存在扩张, 原因是此时  $c$  是对  $k$  的封装, 而非加密. 综上, 使用 KEM 代替 PKE, 不仅能够缩减整体密文尺寸, 也能够提升效率.

#### 注记 2.15

通常 KEM 要比 PKE 构造简单, 这是因为 KEM 可以看作功能受限的 PKE, 因为其只允许加密随机的明文.

相比效率提升, KEM-DEM 的理论价值更大. 首先, KEM-DEM 范式实现了对 PKE 的功能解耦, 将 PKE 中的非对称内核抽取出来凝练为 KEM, 意义如下:

- KEM-DEM 范式极大简化了 PKE 的可证明安全. 我们只需证明 KEM 和 DEM 满足一定性质即可. 对比安全模型即可发现, 对于 PKE 有 CPA/CCA1/CCA 三个依次增强的安全性, 而 KEM 只有 CPA/CCA 两个依次增强的安全性. 最关键的是: 在 PKE 中敌手  $\mathcal{A}$  对挑战密文  $c^*$  有一定的控制能力, 而 KEM 中  $c^*$  完全由挑战者控制, 这一区别使得 KEM 安全证明中的归约算法更容易设计.
- KEM-DEM 范式有助于简化 PKE 的设计. 该范式将 PKE 的设计任务简化为对应的 KEM, 在后面的章节中可以看到, 在设计高等级安全的 PKE 时, 仅需设计满足相应安全性的 KEM 即可.
- KEM-DEM 范式有助于洞悉 PKE 本质. 该范式揭示了构造 PKE 的核心机制在于构造 KEM. 后续的章节揭示了 KEM 的本质是公开可求值的伪随机函数, 是伪随机函数在 minicrypt 中的对应. 认识到这一点后, 不仅可以将几乎所有公钥加密的构造统一在同一框架下, 还可以将 SKE 和 PKE 的构造在伪随机函数的视角下实现高度统一.

#### 注记 2.16

目前, 格基的 KEM 设计仍是 PKE-SKE 的方式, 显得不够灵巧纯粹, 如何设计精巧纯粹的格基 KEM 是很有挑战意义的研究课题.

### 第三章 经典公钥加密方案回顾

## 3.1 基于数论问题的经典方案

### 3.1.1 Goldwasser-Micali PKE

Goldwasser 和 Micali [GM-JCSS-1984] 在 1984 年基于 QR 假设构造出首个可证明安全的公钥加密方案。该方案仅能加密一比特消息，设计的思想可类比编码：当明文为 0 时，随机选取二次剩余元素作为密文；当明文为 1 时，随机选取 Jacobi 符号为 +1 的非二次剩余元素作为密文。

#### 构造 3.1 (Goldwasser-Micali PKE)

- $\text{Setup}(1^\kappa)$ : 全局参数生成  $pp$ , 包含对明文空间  $M = \{0, 1\}$  的描述。
- $\text{KeyGen}(pp)$ : 从  $pp$  中解析出  $\kappa$ , 运行  $\text{GenModulus}(1^\kappa) \rightarrow (N, p, q)$ , 随机选取  $z \in \mathcal{QR}_N^{+1}$ , 输出公钥  $pk = (N, z)$  和私钥  $sk = (p, q)$ 。
- $\text{Encrypt}(pk, m)$ : 以公钥  $pk = (N, z)$  和明文  $m \in \{0, 1\}$  为输入, 随机选择  $x \xleftarrow{\text{R}} \mathbb{Z}_N^*$ , 输出密文  $c = z^m \cdot x^2 \bmod N$ 。
- $\text{Decrypt}(sk, c)$ : 以私钥  $sk = (p, q)$  和密文  $c$  为输入, 利用私钥判定  $c$  是否是模  $N$  的二次剩余。若是, 输出 0; 否则输出 1.



Goldwasser-Micali PKE 的正确性显然, 安全性由以下定理保证。

#### 定理 3.1

如果 QR 假设成立, 那么 Goldwasser-Micali PKE 是 IND-CPA 安全的。



**证明** 令  $S_i$  表示敌手在 Game<sub>i</sub> 中成功概率。以游戏序列的方式组织证明如下:

Game<sub>0</sub>: 该游戏是标准的 IND-CPA 游戏, 挑战者  $\mathcal{CH}$  和敌手  $\mathcal{A}$  交互如下:

- 初始化:  $\mathcal{CH}$  运行  $\text{Setup}(1^\kappa)$  生成公开参数  $pp$ , 同时运行  $\text{KeyGen}(pp)$  生成公私钥对  $pk = (N, z)$  和  $sk = (p, q)$ .  $\mathcal{CH}$  将  $(pp, pk)$  发送给  $\mathcal{A}$ .
- 挑战:  $\mathcal{A}$  选择  $m_0, m_1 \in \mathbb{G}$  并发送给  $\mathcal{CH}$ .  $\mathcal{CH}$  选择随机比特  $\beta \in \{0, 1\}$ , 随机选择  $x \in \mathbb{Z}_N^*$ , 计算  $c^* = z^{m_\beta} \cdot x^2 \bmod N$  并发送给  $\mathcal{A}$ .
- 猜测:  $\mathcal{A}$  输出对  $\beta$  的猜测  $\beta'$ .  $\mathcal{A}$  成功当且仅当  $\beta' = \beta$ .

根据定义, 我们有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/2|$$

Game<sub>1</sub>: 与 Game<sub>0</sub> 的唯一不同在于密钥对的生成方式,  $\mathcal{CH}$  将  $pk$  中元素  $z$  的选取由 Jacobi 符号为 +1 的随机非二次剩余元素切换为随机二次剩余元素。在 Game<sub>1</sub> 中, 无论  $m_\beta$  是 0 还是 1, 密文分布均是  $\mathcal{QR}_N$  上的均匀分布, 完美掩盖了  $\beta$  的信息。因此, 即使对于拥有无穷计算能力的敌手, 我们也有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_1] - 1/2| = 0$$

#### 引理 3.1

如果 QR 假设成立, 那么对于任意 PPT 敌手我们均有  $|\Pr[S_0] - \Pr[S_1]| \leq \text{negl}(\kappa)$ .



**证明** 证明的思路是反证。若存在 PPT 敌手  $\mathcal{A}$  在 Game<sub>0</sub> 和 Game<sub>1</sub> 中成功的概率之差不可忽略, 则可构造出 PPT 算法  $\mathcal{B}$  打破 QR 困难问题。令  $\mathcal{B}$  的 QR 挑战实例为  $(N, z)$ ,  $\mathcal{B}$  的目标是区分挑战实例  $z$  选自  $\mathcal{QR}_N^{+1}$  还是  $\mathcal{QR}_N$  上的均匀分布。为此  $\mathcal{B}$  扮演 IND-CPA 游戏中的挑战者与  $\mathcal{A}$  交互如下:

- 初始化:  $\mathcal{B}$  根据它的挑战实例生成  $pp$ , 令  $pk = (N, z)$ , 将  $(pp, pk)$  发送给  $\mathcal{A}$ .
- 挑战:  $\mathcal{A}$  选择  $m_0, m_1 \in \mathbb{G}$  并发送给  $\mathcal{B}$ .  $\mathcal{B}$  随机选择  $\beta \xleftarrow{\text{R}} \{0, 1\}$ , 随机选取  $x \in \mathbb{Z}_N^*$  设置  $c^* = z^{m_\beta} \cdot x^2$  并发送给  $\mathcal{A}$ .

- 猜测:  $\mathcal{A}$  输出对  $\beta$  的猜测  $\beta'$ . 如果  $\beta' = \beta$ ,  $\mathcal{B}$  输出 1.

对上述交互分析可知, 如果  $z \xleftarrow{\text{R}} \mathcal{QR}_N^{+1}$ , 那么  $\mathcal{B}$  完美的模拟了 Game<sub>0</sub>; 如果  $z \xleftarrow{\text{R}} \mathcal{QR}_N$ , 那么  $\mathcal{B}$  完美的模拟了 Game<sub>1</sub>. 因此,  $\mathcal{B}$  解决 QR 挑战的优势  $\text{Adv}_{\mathcal{B}}^{\text{QR}}(\kappa) = |\Pr[S_0] - \Pr[S_1]|$ . 如果 QR 假设成立, 我们有  $|\Pr[S_0] - \Pr[S_1]| \leq \text{negl}(\kappa)$ .

综上, 定理得证. □

### 3.1.2 Rabin PKE

Rabin [Rabin-TechReport-1979] 在 1979 年基于 SQR 假设构造出  $\mathcal{QR}_N$  上的单向陷门置换  $f_N \stackrel{\text{def}}{=} [x^2 \bmod N]$ , 称为 Rabin TDP. 可以证明, 最低有效位 (lsb, least significant bit) 函数是 Rabin TDP 的 hardcore 谓词. 基于 Rabin TDP, 可以构造公钥加密方案如下:

#### 定义 3.1 (Rabin PKE)

- $\text{Setup}(1^\kappa)$ : 全局参数生成  $pp$ , 包含对明文空间  $M = \{0, 1\}$  的描述.
- $\text{KeyGen}(pp)$ : 从  $pp$  中解析出  $\kappa$ , 运行  $\text{GenModulus}(1\kappa) \rightarrow (N, p, q)$ , 其中  $N$  是 Blum 整数. 输出公钥  $pk = N$  和私钥  $sk = (p, q)$ .
- $\text{Encrypt}(pk, m)$ : 以公钥  $pk = N$  和明文  $m \in \{0, 1\}$  为输入, 随机选择  $x \xleftarrow{\text{R}} \mathcal{QR}_N$ , 计算  $c_0 = x^2 \bmod N$ , 计算  $c_1 = m \oplus \text{lsb}(x)$ , 输出  $c = (c_0, c_1)$  作为密文.
- $\text{Decrypt}(sk, c)$ : 以私钥  $sk = (p, q)$  和密文  $c = (c_0, c_1)$  为输入, 计算  $x$  满足  $x^2 = c_0 \bmod N$ , 输出  $m' = c_1 \oplus \text{lsb}(x)$ .



Rabin PKE 的正确性由  $f_N \stackrel{\text{def}}{=} [x^2 \bmod N]$  是陷门置换这一事实保证, IND-CPA 安全性由陷门置换的单向性保证.

## 3.2 基于离散对数类问题的经典方案

### 3.2.1 ElGamal PKE

1985 年, ElGamal [ElGamal-IEEE-IT-1985] 基于 Diffie-Hellman 构造了 ElGamal PKE 方案. 该方案设计简洁精巧, 对后续的研究有深远的影响.

#### 定义 3.2 (ElGamal PKE)

- $\text{Setup}(1^\kappa)$ : 运行  $\text{GenGroup}(1^\kappa) \rightarrow (\mathbb{G}, q, g)$ , 输出  $pp$  包含循环群描述, 同时包含对公钥空间  $PK = \mathbb{G}$ 、私钥空间  $SK = \mathbb{Z}_q$ 、明文空间  $M = \mathbb{G}$  和密文空间  $C = \mathbb{G}^2$ .
- $\text{KeyGen}(pp)$ : 随机选取  $sk \in \mathbb{Z}_q$  作为私钥, 计算公钥  $pk := g^{sk}$ .
- $\text{Encrypt}(pk, m)$ : 以公钥  $pk$  和明文  $m \in \mathbb{G}$  为输入, 随机选择  $r \xleftarrow{R} \mathbb{Z}_q$ , 计算  $c_0 = g^r$ ,  $c_1 = pk^r \cdot m$ , 输出密文  $c = (c_1, c_0) \in C$ .
- $\text{Decrypt}(sk, c)$ : 以私钥  $sk$  和密文  $c = (c_0, c_1)$  为输入, 输出  $m' := c_1 / c_0^{sk}$ .



**正确性.** 以下公式 ?? 说明方案具有完美正确性:

$$m' = c_1 / c_0^{sk} = pk^r \cdot m / (g^r)^{sk} = m \quad (3.1)$$

#### 定理 3.2

如果 DDH 假设成立, 那么 ElGamal PKE 是 IND-CPA 安全的.



**证明** 令  $S_i$  表示敌手在 Game<sub>i</sub> 中成功概率. 以游戏序列的方式组织证明如下:

Game<sub>0</sub>: 该游戏是标准的 IND-CPA 游戏, 挑战者  $\mathcal{CH}$  和敌手  $\mathcal{A}$  交互如下:

- 初始化:  $\mathcal{CH}$  运行  $\text{Setup}(1^\kappa)$  生成公开参数  $pp$ , 同时运行  $\text{KeyGen}(pp)$  生成公私钥对  $(pk, sk)$ .  $\mathcal{CH}$  将  $(pp, pk)$  发送给  $\mathcal{A}$ .
- 挑战:  $\mathcal{A}$  选择  $m_0, m_1 \in \mathbb{G}$  并发送给  $\mathcal{CH}$ .  $\mathcal{CH}$  选择随机比特  $\beta \in \{0, 1\}$ , 随机选择  $r \in \mathbb{Z}_q$ , 计算  $c^* = (g^r, pk^r \cdot m_\beta)$  并发送给  $\mathcal{A}$ .
- 猜测:  $\mathcal{A}$  输出对  $\beta$  的猜测  $\beta'$ .  $\mathcal{A}$  成功当且仅当  $\beta' = \beta$ .

根据定义, 我们有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/2|$$

Game<sub>1</sub>: 与 Game<sub>0</sub> 的唯一不同在于挑战密文的生成方式,  $\mathcal{CH}$  不再计算  $pk^r$  作为会话密钥掩蔽  $m_\beta$ , 而是随机选取  $z \xleftarrow{R} \mathbb{Z}_q$ , 用  $g^z$  作为会话密钥掩蔽  $m_\beta$ . 挑战密文  $c^* = (g^r, g^z \cdot m_\beta)$ . 在 Game<sub>1</sub> 中, 由于  $r$  和  $z$  均有挑战者从  $\mathbb{Z}_q$  中独立随机选取, 因此挑战密文  $c^*$  在  $\mathbb{G} \times \mathbb{G}$  上均匀分布, 完美掩盖了  $\beta$  的信息. 因此, 即使对于拥有无穷计算能力的敌手, 我们也有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_1] - 1/2| = 0$$

#### 引理 3.2

如果 DDH 假设成立, 那么对于任意 PPT 敌手我们均有  $|\Pr[S_0] - \Pr[S_1]| \leq \text{negl}(\kappa)$ .



**证明** 证明的思路是反证. 若存在 PPT 敌手  $\mathcal{A}$  在 Game<sub>0</sub> 和 Game<sub>1</sub> 中成功的概率差不可忽略, 则可构造出 PPT 算法  $\mathcal{B}$  打破 DDH 困难问题. 令  $\mathcal{B}$  的 DDH 挑战实例为  $(g, g^a, g^b, g^c)$ ,  $\mathcal{B}$  的目标是区分挑战实例是 DDH 四元组还是随机四元组. 为此  $\mathcal{B}$  扮演 IND-CPA 游戏中的挑战者与  $\mathcal{A}$  交互如下:

- 初始化:  $\mathcal{B}$  根据它的挑战实例生成  $pp$ , 令  $pk = g^a$ , 将  $(pp, pk)$  发送给  $\mathcal{A}$ . 注意,  $\mathcal{B}$  并不知晓  $a$  (这是符合逻辑的, 不然归约无意义).

- 挑战:  $\mathcal{A}$  选择  $m_0, m_1 \in \mathbb{G}$  并发送给  $\mathcal{B}$ .  $\mathcal{B}$  随机选择  $\beta \xleftarrow{\text{R}} \{0, 1\}$ , 设置  $c^* = (g^b, g^c \cdot m_\beta)$  并发送给  $\mathcal{A}$ . 该设定隐式的设定  $r = b$ .
- 猜测:  $\mathcal{A}$  输出对  $\beta$  的猜测  $\beta'$ . 如果  $\beta' = \beta$ ,  $\mathcal{B}$  输出 1.

对上述交互分析可知, 如果  $c = ab$ , 那么  $\mathcal{B}$  完美的模拟了 Game<sub>0</sub>; 如果  $c$  在  $\mathbb{Z}_q$  中随机选择, 那么  $\mathcal{B}$  完美的模拟了 Game<sub>1</sub>. 因此,  $\mathcal{B}$  解决 DDH 挑战的优势  $\text{Adv}_{\mathcal{B}}^{\text{DDH}}(\kappa) = |\Pr[S_0] - \Pr[S_1]|$ . 如果 DDH 假设成立, 我们有  $|\Pr[S_0] - \Pr[S_1]| \leq \text{negl}(\kappa)$ .

综上, 定理得证.  $\square$

 **笔记** [具有实际应用价值的同态] ElGamal PKE 构建在  $q$  阶循环群  $\mathbb{G} = \langle g \rangle$  上, 明文空间是  $\mathbb{G}$ , 使用公钥  $pk$  对明文  $m$  的加密所得密文为  $(g^r, pk^r \cdot m)$ . 容易验证, ElGamal PKE 相对于  $\mathbb{G}$  中的群运算“.”同态, 然而, 这种类型的同态并无实际意义, 现实应用中需要的是相对于  $\mathbb{Z}_q$  上的模加运算“+”同态. 面向实际需求, ISO/IEC 标准化了 exponential ElGamal PKE 方案. 该方案同样构建在  $q$  阶循环群  $\mathbb{G} = \langle g \rangle$  上, 所不同的是明文空间设定为  $\mathbb{G}$  的自然同构  $\mathbb{Z}_q$ , 使用公钥  $pk$  对明文  $m$  加密时, 首先计算  $m$  的自然同构映射结果  $g^m$ , 再如常加密, 最终密文为  $(g^r, pk^r \cdot g^m)$ . 容易验证, exponential ElGamal 相对于  $\mathbb{Z}_q$  中的“+”运算同态.

### 3.2.2 Twisted ElGamal PKE

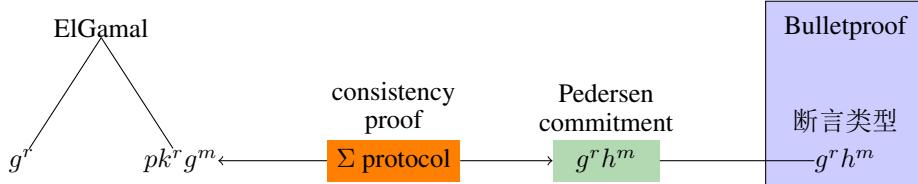
近半个世纪, 随着网络技术的飞速发展, 计算模式逐渐由集中式迁移分布式. 新型计算模式对加密方案的需求也从单一的机密性保护扩展到对隐私计算的支持. 上一节注记中提到的 Exponential ElGamal PKE 支持  $\mathbb{Z}_p$  上的模加运算“+”同态, 适用于密态计算场景. 在区块链和机器学习等涉及恶意敌手的计算场景中, 还需要加法同态加密方案具有零知识证明友好的特性, 即易与零知识证明协议进行套接, 证明密文加密的明文满足声称的约束关系, 如在指定的区间内.

当前最高效的零知识区间范围证明系统是构建在离散对数群上的 Bulletproof [Bunz-BulletProof-SP-2018], 其接受的断言类型为 Pedersen 承诺. 仅管 exponential ElGamal PKE 密文的第二项  $pk^r \cdot g^m$  也是 Pedersen 承诺的形式, 但是若证明者为密文发送方, 则其知晓承诺密钥  $(pk, g)$  之间的离散对数关系, 从而无法调用 Bulletproof 完成证明(合理性得不到保证).

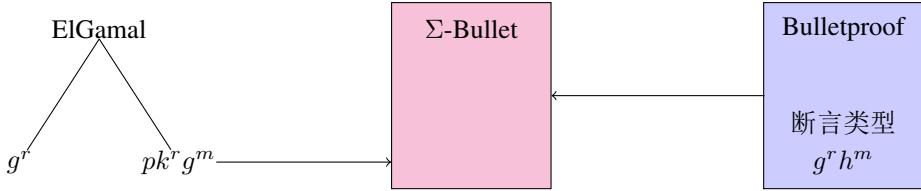


解决该问题有两种技术手段:

- 文献 [Quisquis-ASIACRYPT-2019] 中的方法: 证明者引入新的 Pedersen 承诺作为桥接, 并设计  $\Sigma$  协议证明新承诺的消息与明文的一致性, 再调用 Bulletproof 对桥接承诺进行证明. 该方法的缺点是需要引入桥接承诺的额外的  $\Sigma$  协议, 增大了证明和验证的开销.



- 文献 [Zether-FC-2020] 中的方法: 结合待证明的 ElGamal PKE 密文对 Bulletproof 进行重新设计, 使用特制的  $\Sigma$ -Bulletproof 完成证明. 该方法的缺点是需要对 Bulletproof 进行定制化的改动, 不具备模块化.



上述两种技术手段均存在不足. 针对这一问题, Chen 等 [Chen-ESORICS-2020] 对 exponential ElGamal PKE 进行变形扭转, 将封装密文与会话密钥的位置对调, 同时更改同构映射编码的基底, 得到 twisted ElGamal PKE. 新的加密方案与 exponential ElGamal PKE 的性能和安全性相当, 同样满足  $\mathbb{Z}_q$  上的模加同态; 特别的, 密文的第二部分恰好是标准的 Pedersen 承诺形态 (承诺密钥陷门未知), 因此可无缝对接 Bulletproof 等区间范围证明, 具备零知识证明友好的特性.

### 定义 3.3 (Twisted ElGamal PKE)

- $\text{Setup}(1^\kappa)$ : 运行  $\text{GenGroup}(1^\kappa) \rightarrow (\mathbb{G}, q, g)$ , 随机选取  $\mathbb{G}$  的另一生成元  $h$ , 输出  $pp$  包含循环群和  $h$  的描述, 同时包含对公钥空间  $PK = \mathbb{G}$ 、私钥空间  $SK = \mathbb{Z}_q$ 、明文空间  $M = \mathbb{Z}_q$  和密文空间  $C = \mathbb{G}^2$ .
- $\text{KeyGen}(pp)$ : 随机选取  $sk \in \mathbb{Z}_q$  作为私钥, 计算公钥  $pk := g^{sk}$ .
- $\text{Encrypt}(pk, m)$ : 以公钥  $pk$  和明文  $m \in \mathbb{Z}_q$  为输入, 随机选择  $r \xleftarrow{R} \mathbb{Z}_q$ , 计算  $c_0 = pk^r, c_1 = pk^r \cdot h^m$ , 输出密文  $c = (c_0, c_1) \in C$ .
- $\text{Decrypt}(sk, c)$ : 以私钥  $sk$  和密文  $c = (c_0, c_1)$  为输入, 输出  $m' := \log_h c_1 / c_0^{sk^{-1}}$ .



正确性. 以下公式 ?? 说明方案具有完美正确性:

$$c_1 / c_0^{sk^{-1}} = g^r \cdot h^m / (pk^r)^{sk^{-1}} = h^m \quad (3.2)$$

### 定理 3.3

如果 DDH 假设成立, 那么 twisted ElGamal PKE 是 IND-CPA 安全的.



证明与标准的 ElGamal PKE 证明类似, 我们留给作者作为练习.

笔记 为获得  $\mathbb{Z}_q$  上的加法同态, exponential ElGamal PKE 和 twisted ElGamal PKE 均将明文空间设定为  $\mathbb{Z}_q$ , 加密时必须先进行同构编码, 解密时则在最后需要进行解码. 解码的过程等同于求解离散对数, 因此为了确保解密高效, 必须将明文空间限制在较小的范围内, 如  $[2^{40}]$ .

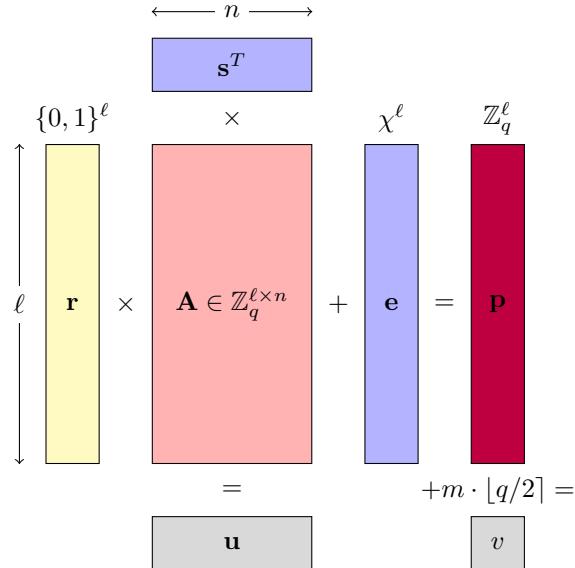


图 3.1: Regev PKE 加密方案示意图

## 3.3 基于格问题的经典方案

### 3.3.1 Regev PKE

Regev [Regev-STOC-2005] 中提出了 LWE 困难问题, 并基于该问题构造了一个公钥加密方案, 称为 Regev PKE.

#### 定义 3.4 (Regev PKE)

- $\text{Setup}(1^\kappa)$ : 以安全参数  $\kappa$  为输入, 生成随机矩阵  $\mathbf{A} \in \mathbb{Z}_q^{\ell \times n}$  作为公开参数.
- $\text{KeyGen}(pp)$ : 以公开参数  $pp$  为输入, 随机选取向量  $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n$  作为私钥, 随机选取噪声向量  $\mathbf{e} \xleftarrow{R} \chi^\ell$  (其中  $\chi^\ell = D_{\mathbb{Z}^\ell, r}$ ), 计算  $\mathbf{p} \leftarrow \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^\ell$  作为公钥.
- $\text{Encrypt}(pk, m)$ : 以公钥  $pk = \mathbf{p}$  和明文  $m \in \{0, 1\}$  为输入, 随机选取向量  $\mathbf{r} \xleftarrow{R} \{0, 1\}^\ell$  计算  $\mathbf{u}^T = \mathbf{r}^T \mathbf{A}$ ,  $v = \mathbf{r}^T \mathbf{p} + m \cdot \lfloor q/2 \rfloor$  作为密文.
- $\text{Decrypt}(\mathbf{s}, c)$ : 以私钥  $sk = \mathbf{s}$  和密文  $c = (\mathbf{u}, v)$ , 计算  $y = v - \mathbf{u}^T \mathbf{s} \in \mathbb{Z}_q$ , 若  $y$  接近 0 则输出 0, 若  $y$  接近  $\lfloor q/2 \rfloor$  则输出 1.



**正确性.** 观察以下等式:

$$\begin{aligned}
 y &= v - \mathbf{u}^T \mathbf{s} \\
 &= \mathbf{r}^T \mathbf{p} + m \cdot \lfloor q/2 \rfloor - \mathbf{r}^T \mathbf{A} \mathbf{s} \\
 &= \mathbf{r}^T (\mathbf{A} \mathbf{s} + \mathbf{e}) + m \cdot \lfloor q/2 \rfloor - \mathbf{r}^T \mathbf{A} \mathbf{s} \\
 &= \mathbf{r}^T \mathbf{e} + m \cdot \lfloor q/2 \rfloor
 \end{aligned}$$

由上述推导可知, 当累计误差  $|\langle \mathbf{r}, \mathbf{e} \rangle| \leq q/4$  时解密正确. 因此, 在参数选取时应令  $q$  的取值相对于误差分布  $\chi$  和  $\ell$  相对大. 比如, 当  $\chi = D_{\mathbb{Z}, r}$  是离散 Gaussian 分布时,  $\langle \mathbf{r}, \mathbf{e} \rangle$  是参数至多为  $r\sqrt{\ell}$  的亚 Gaussian 分布, 其尺寸小于  $r\sqrt{\ell \ln(1/\varepsilon)/\pi}$  的概率至少为  $1 - 2\varepsilon$ . 为了确保解密错误的概率可忽略, 可设定  $r = \Theta(\sqrt{n})$ ,  $q = \tilde{O}(n)$ , 对应 LWE 错误率  $\alpha = r/q = 1/\tilde{O}(n)$ .

**定理 3.4**

如果判定性 LWE 假设成立, 则 Regev PKE 是 IND-CPA 安全的.



**证明** 令  $S_i$  表示敌手在 Game<sub>i</sub> 中成功概率. 以游戏序列的方式组织证明如下:

Game<sub>0</sub>: 该游戏是标准的 IND-CPA 游戏. 挑战者  $\mathcal{CH}$  和敌手  $\mathcal{A}$  交互如下:

- 初始化:  $\mathcal{CH}$  运行  $\text{Setup}(1^\kappa)$  生成公开参数  $\mathbf{A} \in \mathbb{Z}_q^{\ell \times n}$ , 同时生成公私钥对, 其中私钥  $sk$  为随机向量  $\mathbf{s} \in \mathbb{Z}_q^n$ , 公钥  $pk$  为  $\mathbf{p} = \mathbf{As} + \mathbf{e} \in \mathbb{Z}_q^\ell$ , 其中  $\mathbf{e} \leftarrow \chi^\ell$ .
- 挑战:  $\mathcal{A}$  选取  $(m_0, m_1)$  发送给  $\mathcal{CH}$ .  $\mathcal{CH}$  随机选取  $\mathbf{r} \xleftarrow{R} \{0, 1\}^\ell$ ,  $\beta \xleftarrow{R} \{0, 1\}$ , 计算  $\mathbf{u} = \mathbf{r}^T \mathbf{A}$ ,  $v = \mathbf{r}^T \mathbf{p} + m_\beta \cdot \lfloor q/2 \rfloor$ , 发送  $(\mathbf{u}, v)$  给  $\mathcal{A}$  作为挑战密文.
- 猜测:  $\mathcal{A}$  输出对  $\beta$  的猜测  $\beta'$ .  $\mathcal{A}$  成功当且仅当  $\beta = \beta'$ .

根据定义, 我们有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/2|$$

Game<sub>1</sub>: 与 Game<sub>0</sub> 惟一不同的是  $\mathcal{CH}$  生成公钥的方式由计算  $\mathbf{As} + \mathbf{e}$  变为随机选取  $\mathbb{Z}_q^\ell$  上的向量. 在 Game<sub>1</sub> 中,  $\vec{\mathbf{A}} = \mathbf{A} \mid \mathbf{p}$  是  $\mathbb{Z}_q^{\ell \times n}$  上的随机矩阵, 容易验证  $f_{\vec{\mathbf{A}}}(\mathbf{r}) = \mathbf{r}^T \vec{\mathbf{A}}$  是从  $\{0, 1\}^\ell$  到  $\mathbb{Z}_q^{n+1}$  的 universal hash, 由参数选取  $\ell > n \log q$  和剩余哈希引理 (leftover hash lemma) 可知, 函数的输出服从  $\mathbb{Z}_q^{n+1}$  上的均匀分布. 因此, 挑战密文完美掩盖了  $\beta$  的信息. 因此, 即使对于拥有无穷计算能力的敌手, 我们也有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_1] - 1/2| = 0$$

**断言 3.1**

如果判定性 LWE 假设成立, 那么对于任意 PPT 敌手均有  $|\Pr[S_0] - \Pr[S_1]| \leq \text{negl}(\kappa)$ .



**证明** 证明的思路是反证. 若存在 PPT 敌手  $\mathcal{A}$  在 Game<sub>0</sub> 和 Game<sub>1</sub> 中成功的概率差不可忽略, 则可构造出 PPT 算法  $\mathcal{B}$  打破 LWE 困难问题. 令  $\mathcal{B}$  的 LWE 挑战实例为  $(\mathbf{A}, \mathbf{p})$ ,  $\mathcal{B}$  的目标是区分挑战实例是随机采样还是 LWE 采样. 为此  $\mathcal{B}$  扮演 IND-CPA 游戏中的挑战者与  $\mathcal{A}$  交互如下:

- 初始化:  $\mathcal{B}$  发送  $(\mathbf{A}, \mathbf{p})$  给  $\mathcal{A}$ . 该操作将  $pk$  隐式地设定为  $\mathbf{p}$ .
- 挑战:  $\mathcal{A}$  选取  $(m_0, m_1)$  发送给  $\mathcal{CH}$ .  $\mathcal{CH}$  随机选取  $\mathbf{r} \xleftarrow{R} \{0, 1\}^\ell$ ,  $\beta \xleftarrow{R} \{0, 1\}$ , 计算  $\mathbf{u} = \mathbf{r}^T \mathbf{A}$ ,  $v = \mathbf{r}^T \mathbf{p} + m_\beta \cdot \lfloor q/2 \rfloor$ , 发送  $(\mathbf{u}, v)$  给  $\mathcal{A}$  作为挑战密文.
- 猜测:  $\mathcal{A}$  输出对  $\beta$  的猜测  $\beta'$ . 如果  $\beta = \beta'$ ,  $\mathcal{B}$  输出 1.

对上述交互分析可知, 如果  $\mathbf{p}$  是 LWE 采样, 那么  $\mathcal{B}$  完美模拟了 Game<sub>0</sub>; 如果  $\mathbf{p}$  是随机采样, 那么  $\mathcal{B}$  完美模拟了 Game<sub>1</sub>. 因此,  $\mathcal{B}$  解决 LWE 挑战的优势  $\text{Adv}_{\mathcal{B}}^{\text{LWE}}(\kappa) = |\Pr[S_0] - \Pr[S_1]|$ . 如果 LWE 假设成立, 我们有  $|\Pr[S_0] - \Pr[S_1]| \leq \text{negl}(\kappa)$ .  $\square$

综上, 定理得证.  $\square$

**注记 3.1**

Regev PKE 和 Goldwasser-Micali PKE 在设计上有异曲同工之处, 均采用的是有损加密思想, 即公钥存在正常和有损这两种计算不可区分的类型, 正常公钥生成的密文可以正确解密, 而有损公钥生成的密文丢失了明文的全部信息. 在安全性证明时, 利用两种类型公钥的计算不可区分性以及有损加密的性质, 即可完成 IND-CPA 安全的论证.



### 3.3.2 GPV PKE

Gentry, Peikert 和 Vaikuntanathan [GPV-STOC-2008] 同样基于 LWE 假设设计出一个 PKE 方案, 称为 GPV PKE.

GPV PKE 由以下 4 个多项式时间算法组成:

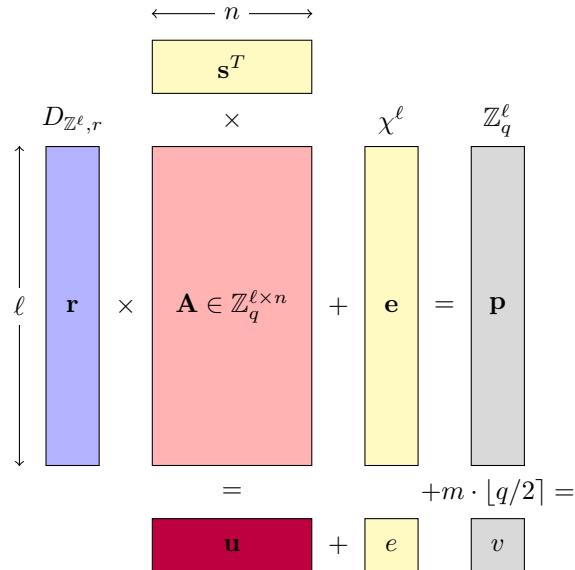


图 3.2: GPV PKE 加密方案示意图

- **Setup( $1^\kappa$ ):** 以安全参数  $\kappa$  为输入, 生成随机矩阵  $\mathbf{A} \in \mathbb{Z}_q^{\ell \times n}$  作为公开参数.
  - **KeyGen( $pp$ ):** 以公开参数  $pp$  为输入, 随机选取噪声向量  $\mathbf{r} \xleftarrow{\text{R}} D_{\mathbb{Z}^\ell, r}$  作为私钥, 计算  $\mathbf{u}^T \leftarrow \mathbf{r}^T \mathbf{A}$  作为公钥. 从编码的角度,  $\mathbf{u}$  可以理解为  $\mathbf{e}$  相对于  $\mathbf{A}$  的 syndrome.
  - **Encrypt( $pk, m$ ):** 以公钥  $pk = \mathbf{u}$  和明文  $m \in \{0, 1\}$  为输入, 随机选取向量  $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$  和  $\mathbf{e} \xleftarrow{\text{R}} \chi^\ell$ , 随机选取  $e \xleftarrow{\text{R}} \chi$ , 计算  $\mathbf{p} = \mathbf{As} + \mathbf{e} \in \mathbb{Z}_q^\ell$ ,  $v = \mathbf{u}^T \mathbf{s} + e + m \cdot \lfloor q/2 \rfloor$  作为密文.
  - **Decrypt( $\mathbf{r}, c$ ):** 以私钥  $sk = \mathbf{r}$  和密文  $c = (\mathbf{p}, v)$ , 计算  $y = v - \mathbf{r}^T \mathbf{p} \in \mathbb{Z}_q$ , 若  $y$  接近 0 则输出 0, 若  $y$  接近  $\lfloor q/2 \rfloor$  则输出 1.

**正确性.** 观察以下等式:

$$\begin{aligned}
y &= v - \mathbf{r}^T \mathbf{p} \\
&= \mathbf{u}^T \mathbf{s} + e + m \cdot \lfloor q/2 \rfloor - \mathbf{r}^T (\mathbf{A}\mathbf{s} + \mathbf{e}) \\
&= \mathbf{u}^T \mathbf{s} + e + m \cdot \lfloor q/2 \rfloor - \mathbf{u}^T \mathbf{s} - \mathbf{r}^T \mathbf{e} \\
&= m \cdot \lfloor q/2 \rfloor + e - \mathbf{r}^T \mathbf{e}
\end{aligned}$$

由上述推导可知, 当累计误差  $|\langle e - \mathbf{r}^T \mathbf{e} \rangle| \leq q/4$  时解密正确。通过恰当的参数选择, 可确保累计误差以接近 1 的绝对优势概率小于等于  $q/4$ , 更多细节请参考 [GPV-STOC-2008]

### 定理 3.5

如果判定性 LWE 假设成立，则 GPV PKE 是 IND-CPA 安全的。

1

**证明** 令  $S_i$  表示敌手在 Game<sub>i</sub> 中成功概率，以游戏序列的方式组织证明如下：

Game<sub>0</sub>: 该游戏是标准的 IND-CPA 游戏，挑战者  $CH$  和敌手  $A$  交互如下：

- 初始化:  $\mathcal{CH}$  运行  $\text{Setup}(1^\kappa)$  生成公开参数  $\mathbf{A} \in \mathbb{Z}_q^{\ell \times n}$ , 同时生成公私钥对, 其中私钥  $sk$  为随机向量  $\mathbf{r} \in D_{\mathbb{Z}^\ell, r}$ , 公钥  $pk$  为  $\mathbf{u} = \mathbf{r}^T \mathbf{A}$ .
  - 挑战:  $\mathcal{A}$  选取  $(m_0, m_1)$  发送给  $\mathcal{CH}$ .  $\mathcal{CH}$  随机选取  $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n$ , 随机选取  $\mathbf{e} \xleftarrow{R} \chi^\ell$  和  $e \xleftarrow{R} \chi$ ,  $\beta \xleftarrow{R} \{0, 1\}$ , 计算  $\mathbf{p} = \mathbf{As} + \mathbf{e} \in \mathbb{Z}_q^\ell$ ,  $v = \mathbf{u}^T \mathbf{s} + e + m_\beta \cdot \lfloor q/2 \rfloor$  作为密文. 发送  $(\mathbf{u}, v)$  给  $\mathcal{A}$  作为挑战密文.
  - 猜测:  $\mathcal{A}$  输出对  $\beta$  的猜测  $\beta'$ .  $\mathcal{A}$  成功当且仅当  $\beta = \beta'$ .

根据定义, 我们有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/2|$$

**Game<sub>1</sub>**: 与 Game<sub>0</sub> 惟一不同的是  $\mathcal{CH}$  生成公钥的方式由计算  $\mathbf{u}^T = \mathbf{r}^T \mathbf{A}$  变为随机选取  $\mathbf{u} \xleftarrow{\text{R}} \mathbb{Z}_q^n$  上的向量. 在 Game<sub>1</sub> 中,  $(\mathbf{A}, \mathbf{p} = \mathbf{As} + \mathbf{e}, \mathbf{u}, \mathbf{u}^T \mathbf{s} + x)$  恰好构成  $\ell + 1$  个 LWE 采样结果. 有 LWE 假设立刻可知, 敌手在 Game<sub>1</sub> 中的视角计算意义上隐藏了  $\beta$  的信息, 因此基于 LWE 假设有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_1] - 1/2| \leq \text{negl}(\kappa)$$

### 断言 3.2

对于任意的敌手  $\mathcal{A}$ (即使拥有无穷计算能力), 均有  $|\Pr[S_0] - \Pr[S_1]| \leq \text{negl}(\kappa)$



**证明** 根据  $\ell \geq 2n \log q$  的参数选择可知, 公钥  $\mathbf{u}$  的分布与  $\mathbb{Z}_q^n$  上的均匀分布统计不可区分, 因此敌手在 Game<sub>0</sub> 和 Game<sub>1</sub> 中的视图统计不可区分, 从而  $|\Pr[S_0] - \Pr[S_1]| \leq \text{negl}(\kappa)$ .  $\square$

综上, 定理得证.  $\square$

### 注记 3.2

Regev PKE 和 GPV PKE 在形式上相似, 构造使用了相同的元素  $\mathbf{A}, \mathbf{s}, \mathbf{r}, \mathbf{e}, \mathbf{p}, \mathbf{u}, v$ , 但用途含义不完全相同, 构造互为对偶. Regev PKE 中,  $\mathbf{p}$  为公钥,  $(\mathbf{s}, \mathbf{e})$  为私钥,  $\mathbf{u}$  为密文; GPV PKE 中  $\mathbf{p}$  为密文,  $(\mathbf{s}, \mathbf{e})$  为加密随机数,  $\mathbf{u}$  为公钥. 感兴趣的读者可以参考 Micciancio [Micciancio-PKC-2010] 了解更多格密码学中的对偶性. Regev PKE 中, 公钥空间是稀疏的; GPV PKE 中, 公钥空间是稠密的, 正是利用公钥空间的稠密性, 我们可以借助随机预言机将 GPV PKE 升级为身份加密方案 GPV IBE.



## 第四章 通用构造方法

### 内容提要

- 单向陷门函数类
- 哈希证明系统类
- 可提取哈希证明系统类
- 程序混淆类
- 统一构造框架

## 4.1 单向陷门函数类

### 4.1.1 单向陷门函数

单向陷门函数 (TDF) 是单向函数 (OWF) 在 Cryptomania 中的对应, 简言之, 其正向计算容易, 逆向计算困难但在有陷门信息辅助时同样容易.

#### 定义 4.1 (单向陷门函数)

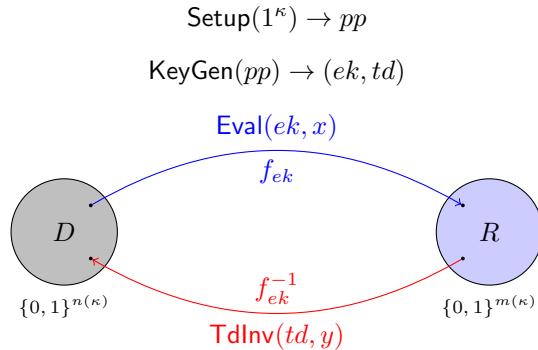
TDF 由以下四个多项式时间算法组成:

- $\text{Setup}(1^\kappa)$ : 系统生成算法以安全参数  $\kappa$  为输入, 输出公开参数  $pp$ , 其中  $pp$  包含对定义域  $D$ , 值域  $R$ , 求值公钥空间  $EK$ 、求逆陷门空间  $TD$  和单向陷门函数族  $f : EK \times D \rightarrow R$  的描述. 换言之,  $f$  是由求值公钥索引的函数族. 不失一般性,  $D$  支持高效的随机采样, 即存在 PPT 算法  $\text{SampDom}$  可以从  $D$  中随机选取一个元素. 在多数情况下,  $D$  和  $R$  是与求值公钥无关的(该性质也被称为 index-independent), 但在有些情形下,  $D$  和  $R$  是由求值公钥索引的空间簇. 为了叙述简洁, 以下均假设  $D$  和  $R$  是单一空间. 空间簇的情形由单一集合的情形自然推广得到.
- $\text{KeyGen}(pp)$ : 以公共参数  $pp$  为输入, 输出密钥对  $(ek, td)$ , 其中  $ek$  为求值公钥,  $td$  为求逆陷门.
- $\text{Eval}(ek, x)$ : 以求值公钥  $ek$  和定义域元素  $x \in D$  为输入, 输出  $y \leftarrow f_{ek}(x)$ .
- $\text{TdInv}(td, y)$ : 以求逆陷门  $td$  和值域元素  $y \in R$  为输入, 输出  $x \in D$  或特殊符号  $\perp$  指示  $y$  不存在原像.



定义以下两条性质:

- 单射:  $\forall ek$ , 称  $f_{ek}$  是单射的当且仅当  $x \neq x' \Rightarrow f_{ek}(x) \neq f_{ek}(x')$ .
- 置换:  $\forall ek$ ,  $\text{Img}(f_{ek}) = D = R$ .



正确性.  $\forall \kappa \in \mathbb{N}, \forall pp \leftarrow \text{Setup}(1^\kappa), \forall (ek, td) \leftarrow \text{KeyGen}(pp)$  和  $\forall x \in D$  和  $y = \text{Eval}(ek, x)$ , 有:

$$\Pr[\text{TDInv}(td, y) \in f_{ek}^{-1}(y)] = 1$$

单向性. 定义单向陷门函数对手  $\mathcal{A}$  的优势函数如下:

$$\text{Adv}_{\mathcal{A}}(\kappa) = \Pr \left[ x \in f_{ek}^{-1}(y^*) : \begin{array}{l} pp \leftarrow \text{Setup}(1^\kappa); \\ (ek, td) \leftarrow \text{KeyGen}(pp); \\ x^* \xleftarrow{\text{R}} D, y^* \leftarrow \text{Eval}(ek, x^*); \\ x \leftarrow \mathcal{A}(pp, ek, y^*) \end{array} \right],$$

如果对于任意的 PPT 敌手  $\mathcal{A}$ , 其优势函数均是可忽略的, 则称该陷门函数是单向的.

**注记 4.1**

1. 不失一般性, 假定  $D$  和  $R$  均存在经典表示, 分别是  $\{0, 1\}^{n(\kappa)}$  和  $\{0, 1\}^{m(\kappa)}$ , 其中  $n(\cdot)$  和  $m(\cdot)$  是关于  $\kappa$  的多项式函数. 容易验证, 长度函数不能过大, 如果  $n(\cdot)$  或  $m(\cdot)$  是超多项式函数, 则函数无法高效计算; 长度函数也不能过小, 如果  $n(\cdot)$  或  $m(\cdot)$  是亚线性函数, 则函数不可能满足单向性.
2. 在抽象定义中, 只限定了  $\text{TDFInv}(td, \cdot)$  在输入为像集元素时返回原像, 而未限定其输入为非像集元素时的行为. 在具体构造时,  $\text{TDFInv}(td, \cdot)$  在输入为非像集元素时的行为往往需要精心设定, 以方便安全性证明.
3. 在单向性的定义中, 敌手  $\mathcal{A}$  仅观察到  $ek$  和  $y^*$  的信息.  $x^* \xleftarrow{R} D$  可以放宽至  $x^*$  选自  $D$  上的高最小熵分布, 即  $H_\infty(x^*) \geq \omega(\log \kappa)$ .



在展示如何使用单向陷门函数构造公钥方案之前, 先展示一个基于单向陷门置换的构造. 该构造并不安全, 但对得到正确的构造很有启发意义.

**构造 4.1 (朴素的 TDF-based PKE(不安全))**

- $\text{Setup}(1^\kappa)$ : 运行  $\text{TDP}.\text{Setup}(1^\kappa)$  生成公开参数  $pp$ , 其中明文空间和密文空间均为单向陷门置换的定义域  $D$ .
- $\text{KeyGen}(pp)$ : 运行  $\text{TDP}.\text{KeyGen}(pp) \rightarrow (ek, td)$ , 其中  $ek$  作为加密公钥,  $td$  作为解密私钥.
- $\text{Encrypt}(ek, m)$ : 以公钥  $ek$  和明文  $m \in D$  为输入, 运行  $\text{TDP}.\text{Eval}(ek, m)$  计算  $c \leftarrow f_{ek}(m)$  作为密文.
- $\text{Decrypt}(td, c)$ : 以私钥  $td$  和密文  $c \in D$  为输入, 运行  $\text{TDP}.\text{TdInv}(td, c)$  计算  $m \leftarrow f_{ek}^{-1}(c)$  恢复明文.



上述构造来自 Diffie 和 Hellman 的经典论文 [DH-IEEE-IT-1976], 原始的 RSA 公钥加密方案就是该构造的具体实例化. 该构造的想法直观, 利用单向陷门置换将明文转化为密文, 同时利用陷门可以求逆从密文中恢复明文. 但其仅仅满足较弱的 OW-CPA 安全, 并不满足现在公认的最低要求 IND-CPA 安全, 因此其也被成为公钥加密的 textbook 构造<sup>1</sup>. 朴素构造不满足 IND-CPA 安全的根本原因是加密算法是确定型的而非概率型的, 因此敌手可以通过“加密-比较”即可打破 IND-CPA 安全. 因此, 强化朴素构造的第一步是选择定义域中的随机元素  $x$ , 计算其函数值  $f_{ek}(x)$  作为封装密文, 再用  $x$  作为会话密钥掩蔽明文. 强化构造仍然不满足 IND-CPA 安全性, 原因是  $f_{ek}(\cdot)$  是公开可计算函数, 其函数值泄漏了原像信息, 使得原像在敌手的视角中不再伪随机. 针对性的强化方法是计算  $x$  的 hardcore function 值作为会话密钥.

以下首先介绍 hardcore function 的概念.

**定义 4.2 (hardcore function)**

称多项式时间可计算的确性型函数  $hc : D \rightarrow K$  是函数  $f : D \rightarrow R$  的 hardcore function 当且仅当:

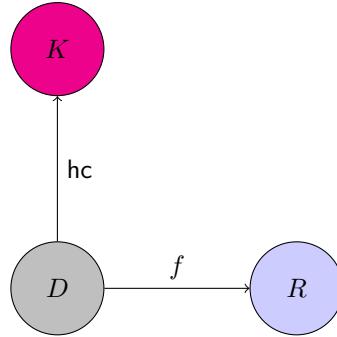
$$(f(x^*), hc(x^*)) \approx_c (f(x^*), U_K)$$

其中概率空间建立在  $x^* \xleftarrow{R} D$  的随机带上. 以安全实验的方式可如下定义, 即对于任意 PPT 敌手  $\mathcal{A}$ , 其安全优势可忽略:

$$\text{Adv}_{\mathcal{A}}(\kappa) = \Pr \left[ \beta' = \beta : \begin{array}{l} pp \leftarrow \text{Setup}(1^\kappa); \\ x^* \xleftarrow{R} D, y^* \leftarrow f(x^*); \\ k_0^* \leftarrow hc(x^*), k_1^* \xleftarrow{R} K, \beta \xleftarrow{R} \{0, 1\}; \\ \beta' \leftarrow \mathcal{A}(pp, ek, y^*, k_\beta^*); \end{array} \right] - \frac{1}{2}$$



<sup>1</sup>textbook 指其仅适合作为以科普为目的教学.



#### 定理 4.1 (Goldreich-Levin Theorem)

如果  $f : \{0,1\}^n \rightarrow \{0,1\}^m$  是单向函数, 那么  $GL(x) = \bigoplus_{i=1}^n x_i r_i$  是从  $\{0,1\}^n$  到  $\{0,1\}$  的单比特输出硬核函数, 或称为硬核谓词.



#### 注记 4.2

Goldreich-Levin 定理是现代密码学中极为重要的结论, 它的意义在于通过显式构造硬核函数, 建立起单向性与伪随机性之间的关联. 从另一个角度理解, GL 硬核谓词可以看做一个计算意义上的随机性提取器, 对  $x|f(x)$  的计算熵进行随机性提取, 萃取出伪随机的一比特. 还需要特别说明的是, 到目前为止尚不知晓如何针对任意单向函数  $f$  设计一个确定型的硬核谓词. GL 是相对于  $g(x, r) := f(x)||r$  的硬核谓词, 或者可以将  $r \xleftarrow{R} \{0,1\}^n$  理解为硬核谓词的描述, 将 GL 理解为  $f$  的随机性硬核谓词. 本书中采用第二种观点. 另一方面, GL 硬核谓词是通用的 (universal), 即构造对于任意单向函数均成立. 强通用性的代价是效率较低, 输出仅是单比特. 当单向函数具有特殊的结构 (如函数是置换) 或者依赖额外困难假设 (如判定性假设、差异输入程序混淆假设) 时, 存在更高效的构造.



以下我们展示如何基于单射的单向陷门函数构造构造 KEM 方案.

#### 构造 4.2 (TDF-based KEM)

- $\text{Setup}(1^\kappa)$ : 运行  $\text{TDF}.\text{Setup}(1^\kappa)$  生成公开参数  $pp$ .  $pp$  中不仅包含单向陷门函数  $f_{ek} : D \rightarrow R$  的描述, 还包括相应硬核函数  $hc : D \rightarrow K$  的描述. KEM 方案的密文空间是 TDF 的定义域  $D$ , 密钥空间是硬核函数的值域  $K$ .
- $\text{KeyGen}(pp)$ : 运行  $\text{TDF}.\text{KeyGen}(pp) \rightarrow (ek, td)$ , 其中  $ek$  作为封装公钥  $pk$ ,  $td$  作为解封装私钥  $sk$ .
- $\text{Encaps}(pk, m)$ : 以公钥  $pk = ek$  为输入, 随机选取  $x \xleftarrow{R} D$ , 运行  $\text{TDF}.\text{Eval}(ek, x)$  计算  $c \leftarrow f_{ek}(m)$  作为封装密文, 计算  $k \leftarrow hc(x)$  作为会话密钥.
- $\text{Decaps}(sk, c)$ : 以私钥  $sk = td$  和密文  $c$  为输入, 运行  $\text{TDF}.\text{TdInv}(td, c)$  计算  $x \leftarrow f_{ek}^{-1}(c)$ , 输出  $k \leftarrow hc(x)$ .



**正确性.** 由单向陷门函数的单射性质和求逆算法的正确性可知, 上述 KEM 构造满足正确性.

#### 定理 4.2

如果  $f_{ek}$  是一族单射单向陷门函数, 那么上述 KEM 构造是 IND-CPA 安全的.



**证明** 证明的思路是单一归约, 即若存在敌手  $\mathcal{A}$  打破 KEM 方案的 IND-CPA 安全性, 则存在敌手  $\mathcal{B}$  打破  $hc$  的伪随机性, 进而与  $f_{ek}$  的单向性矛盾. 令  $\mathcal{B}$  的挑战实例为  $(pp, ek, y^*, k_\beta^*)$ , 其中  $pp$  为单射单向陷门函数的公开参数,  $ek$  为随机生成的求值密钥,  $y^* \leftarrow f_{ek}(x^*)$  是随机选取原像  $x^*$  的像,  $k_0^* \leftarrow hc(x^*)$ ,  $k_1^* \xleftarrow{R} K$ .  $\mathcal{B}$  的目标是判定  $\beta = 0$  抑或  $\beta = 1$ .  $\mathcal{B}$  与  $\mathcal{A}$  交互如下:

- 初始化:  $\mathcal{B}$  根据  $pp$  生成 KEM 方案的公开参数, 并设定公钥  $pk := ek$ , 将  $(pp, ek)$  发送给  $\mathcal{A}$ .
- 挑战:  $\mathcal{B}$  设定  $c^* := y^*$ , 将  $(c^*, k_\beta^*)$  发送给  $\mathcal{A}$ .
- 猜测:  $\mathcal{A}$  输出  $\beta'$ ,  $\mathcal{B}$  将  $\beta'$  转发给它自身的挑战者.

容易验证,  $\mathcal{B}$  完美地模拟了 KEM 方案中的挑战者,  $\mathcal{B}$  成功当且仅当  $\mathcal{A}$  成功. 因此我们有:

$$\text{Adv}_{\mathcal{A}}^{\text{KEM}}(\kappa) = \text{Adv}_{\mathcal{B}}^{\text{hc}}(\kappa)$$

由  $f_{ek}$  的单向性可知, hc 伪随机, 从 KEM 构造满足 IND-CPA 安全性.  $\square$

以上的结果展示了单射单向陷门函数蕴含 IND-CPA 的公钥加密. 需要确认 TDF 与 CCA PKE 之间是否存在分离一个自然的问题是, 单向陷门函数需要满足何种性质才能蕴含 IND-CCA 的公钥加密. 以下, 我们按照时间先后顺序依次介绍单向陷门函数的三个增强版本, 并展示如何基于这些增强版本的单向陷门函数构造 IND-CCA 的公钥加密.

## 4.1.2 有损陷门函数

天之道, 损有余而补不足, 是故虚胜实, 不足胜有余.

— 宋·黄裳《九阴真经》

理想世界中的镜中月和水中花体现的是信息完美复刻, 而现实世界中更多的现象体现的是信息有损, 如拍照、录音, 无论设备和手段多么先进, 都无法做到完美复刻信源信息, 只能做到尽可能的高保真. 单射函数可以形象的理解为理想世界中信息无损的编码过程, 那么什么形式的函数刻画了现实世界中信息有损的编码过程呢? Peikert 和 Waters [PW-STOC-2008] 正是基于上述的思考, 在 2008 年开创性提出了有损陷门函数的概念. 简言之, 有损陷门函数有两种模式, 即正常和有损模式. 在单射模式下, 函数是单射的, 像完全保留了原像的全部信息; 在有损模式下, 函数是有损的, 像在信息论意义上丢失了原像的部分信息. 两种模式之间的关联是计算不可区分.

### 定义 4.3 (有损陷门函数 LTDF)

有损陷门函数 LTDF 由  $n$  和  $\tau$  两个参数刻画, 包含以下五多项式时间算法:

- **Setup( $1^\kappa$ )**: 系统生成算法以安全参数  $\kappa$  为输入, 输出公开参数  $pp$ , 其中  $pp$  包含对定义域  $X$  和值域  $Y$  的描述. 其中  $|X| = 2^{n(\kappa)}$ .
- **GenInjective( $pp$ )**: 以公共参数  $pp$  为输入, 输出密钥对  $(ek, td)$ , 其中  $ek$  为求值公钥,  $td$  为求逆陷门. 该算法输出的  $ek$  定义了从  $X$  到  $Y$  的单射函数  $f_{ek}$ , 拥有对应  $td$  可以对  $f_{ek}$  进行高效求逆.
- **GenLossy( $pp$ )**: 以公共参数  $pp$  为输入, 输出密钥对  $(ek, \perp)$ , 其中  $ek$  为求值公钥,  $\perp$  表示陷门不存在无法求逆. 该算法输出的  $ek$  定义了从  $D$  到  $R$  的有损函数  $f_{ek}$ , 像集的大小至多为  $2^{\tau(\kappa)}$ .
- **Eval( $ek, x$ )**: 以求值公钥  $ek$  和定义域元素  $x \in X$  为输入, 输出  $y \leftarrow f_{ek}(x)$ .
- **TdInv( $td, y$ )**: 以求逆陷门  $td$  和值域元素  $y \in Y$  为输入, 输出  $x \in X$  或特殊符号  $\perp$  指示  $y$  不存在原像.

模式不可区分性. GenInjective( $pp$ ) 和 GenLossy( $pp$ ) 的第一个输出构成的分布在计算意义上不可区分, 即任意 PPT 敌手无法判定求值公钥  $ek$  属于单射模式还是有损模式.

相比常规的单向陷门函数, 有损陷门函数额外具备一个计算不可区分的有损模式, 这正是其威力的来源. 在利用有损陷门函数设计密码方案/协议时, 通常按照如下的步骤:

1. 在单射模式下完成密码方案/协议的功能性构造 (功能性通常需要函数单射可逆)
2. 在有损模式下完成密码方案/协议的安全性论证 (论证通常在信息论意义上进行)
3. 利用单射模式和有损模式的计算不可区分性证明密码方案/协议在正常模式下计算安全性.

细心的读者可能已经发现了有损陷门函数的定义中并没有显式的要求函数在单射模式下具备单向性, 这是因为单射和有损模式的计算不可区分性已经隐式的保证了这一点. 以下进行严格证明, 具体展示应用有损陷门函数设计密码方案/协议的过程.

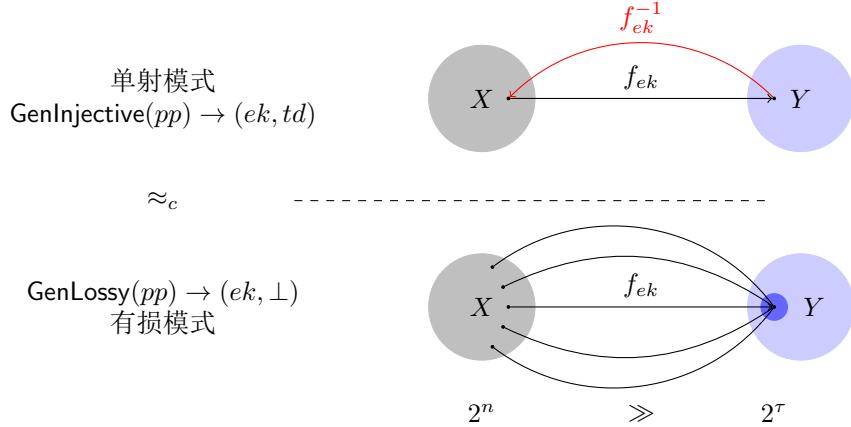


图 4.1: 有损陷门函数 (LTDF) 示意图

**定理 4.3**

令  $\mathcal{F}$  是一族  $(n, \tau)$ -LTDF, 当  $n - \tau \geq \omega(\log \lambda)$  时,  $\mathcal{F}$  的单射模式构成一族单射单向陷门函数.



**证明** 证明通过游戏序列组织.

Game<sub>0</sub>: 该游戏是标准的单射单向陷门函数安全游戏. 挑战者  $\mathcal{CH}$  和敌手  $\mathcal{A}$  交互如下:

- 初始化:  $\mathcal{CH}$  运行  $pp \leftarrow \text{Setup}(\kappa)$ ,  $(ek, td) \leftarrow \text{GenInjective}(pp)$ , 发送  $(pp, ek)$  给  $\mathcal{A}$ .
- 挑战阶段:  $\mathcal{CH}$  随机选择  $x^* \xleftarrow{R} X$ , sends  $y^* \leftarrow f_{ek}(x^*)$  给  $\mathcal{A}$ .
- 猜测阶段:  $\mathcal{A}$  输出  $x'$ ,  $\mathcal{A}$  赢得游戏当且仅当  $x' = x^*$ .

根据定义, 我们有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = \Pr[S_0]$$

Game<sub>1</sub>: 该游戏与上一个游戏完全相同, 唯一不同的是将单射模式切换到有损模式

- 初始化:  $\mathcal{CH}$  运行  $(ek, \perp) \leftarrow \text{GenLossy}(pp)$  生成求值公钥  $ek$ .

根据定义, 我们有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = \Pr[S_1]$$

**断言 4.1**

单射和有损两种模式的计算不可区分性保证了  $|\Pr[S_0] - \Pr[S_1]| \leq \text{negl}(\kappa)$ .



**证明** 我们利用反证法完成归约论证: 若  $|\Pr[S_0] - \Pr[S_1]|$  不可忽略, 则可构造出 PPT 的敌手  $\mathcal{B}$  打破模式的不可区分性.  $\mathcal{B}$  在收到模式不可区分性的挑战  $(pp, ek)$  后, 将  $(pp, ek)$  发送给  $\mathcal{A}$ , 随后随机选取  $x^* \xleftarrow{R} X$ , 计算并发送  $y^* \leftarrow f_{ek}(x^*)$  给  $\mathcal{A}$ . 当收到  $\mathcal{A}$  的输出  $x'$  后, 若  $x' = x^*$ ,  $\mathcal{B}$  输出 ‘1’, 否则输出 ‘0’. 分析可知, 当  $ek$  来自单射模式时,  $\mathcal{B}$  完美的模拟了 Game<sub>0</sub>; 当  $ek$  来自有损模式时,  $\mathcal{B}$  完美的模拟了 Game<sub>1</sub>. 因此, 我们有:

$$|\Pr[\mathcal{B}(ek) = 1 : ek \leftarrow \text{GenInjective}(pp)] - \Pr[\mathcal{B}(ek) = 1 : ek \leftarrow \text{GenLossy}(pp)]| = |\Pr[S_0] - \Pr[S_1]|$$

其中  $pp \leftarrow \text{Setup}(1^\kappa)$ .

**断言 4.2**

对于任意的敌手  $\mathcal{A}$  (即使拥有无穷计算能力), 其在 Game<sub>1</sub> 中的优势也是可忽略的.



**证明** Game<sub>1</sub> 处于有损模式, 因此由 Chaining Lemma ?? 可知,  $x^*$  的平均条件最小熵  $\tilde{H}_\infty(x^* | y^*) \geq n - \tau \geq \omega(\log \kappa)$ , 从而断言得证.



综合以上, 定理得证! □

### 注记 4.3

有损陷门函数相比标准单向陷门函数多了有损模式, 也正因为如此, 其具有标准单向陷门函数很多不具备的优势.

在安全方面, 根据上述论证容易验证只要参数设置满足一定约束, 则有损(陷门)函数在泄漏模型下仍然安全. 具体的, 在敌手获得关于原像任意长度为  $\ell$  有界泄漏的情形下, 只要  $n - \tau - \ell \geq \omega(\log \kappa)$ , 则单向性依然成立. 因此, 有损(陷门)函数是构造抗泄漏单向函数的重要工具 [Komargodski-TCC-2016; Chen-ASIACRYPT-2018].

在效率方面, 令  $\mathcal{H}$  是一族从  $X$  到  $\{0,1\}^{m(\kappa)}$  的对独立哈希函数族 (pairwise-independent hash family), 只要  $n - \tau - m \geq \omega(\log \kappa)$ , 那么从  $\mathcal{H}$  中随机选择的  $h$  即构成单向函数的多比特输出 hardcore function. 论证的方式是应用 Leftover Hash Lemma ?? 和对独立哈希函数族构成强随机性提取器的事实, 得到 hardcore function 输出和均匀随机输出不可区分的结论.



有损陷门函数还有一个非平凡的扩展, 称为全除一 (ABO, All-But-One) 有损陷门函数. 简言之, ABO-TDF 存在一个分支集合 (branch set), 记为  $B$ . 求值密钥  $ek$  和分支值  $b \in B$  共同定义了从  $X$  到  $Y$  的函数  $f_{ek,b}$ , 该函数当且仅当  $b$  等于某特定一个分支值时有损, 在其它分支均单射可逆. 严格定义如下:

### 定义 4.4 (全除一有损陷门函数)

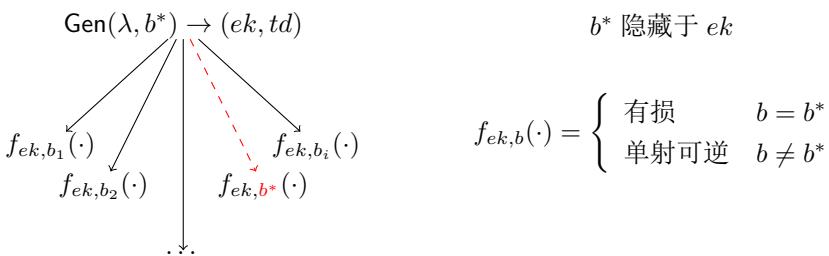
ABO-TDF 由  $n$  和  $\tau$  两个参数刻画, 包含以下五个多项式时间算法:

- $\text{Setup}(1^\kappa)$ : 系统生成算法以安全参数  $\kappa$  为输入, 输出公开参数  $pp$ , 其中  $pp$  包含对定义域  $X$ 、值域  $Y$  和分支集合  $B$  的描述. 其中  $|X| = 2^{n(\kappa)}$ .
- $\text{Gen}(pp, b^*)$ : 以公共参数  $pp$  和给定分支值  $b^* \in B$  为输入, 输出密钥对  $(ek, td)$ , 其中  $ek$  为求值公钥,  $td$  为求逆陷门. 该算法输出的  $ek$  和分支值  $b \in B$  定义了从  $X$  到  $Y$  的函数  $f_{ek,b}$ . 当  $b \neq b^*$  时,  $f_{ek,b}$  单射且拥有对应  $td$  可高效求逆; 当  $b = b^*$  时,  $f_{ek,b^*}$  有损, 像集的大小至多为  $2^{\tau(\kappa)}$ ,  $b^*$  因此称为有损分支.
- $\text{Eval}(ek, b, x)$ : 以求值公钥  $ek$ 、分支值  $b \in B$  和定义域元素  $x \in X$  为输入, 输出  $y \leftarrow f_{ek,b}(x)$ .
- $\text{TdInv}(td, b, y)$ : 以求逆陷门  $td$ 、分支值  $b \in B$  和值域元素  $y \in Y$  为输入, 输出  $x \in X$  或特殊符号  $\perp$  指示  $y$  不存在原像.

有损分支隐藏性. 该性质刻画的安全要求是求值公钥不泄漏有损分支的信息. 严格定义类似公钥加密的不可区分安全或是承诺的隐藏性, 即  $\forall b_0, b_1 \in B$ , 我们有:

$$\text{Gen}(pp, b_0) \approx_c \text{Gen}(pp, b_1)$$

其中  $pp \leftarrow \text{Setup}(1^\kappa)$ .



**注记 4.4**

ABO-TDF 可以理解为 LTDF 的扩展, 分支集合由  $\{0, 1\}$  延拓至  $\{0, 1\}^b$ . LTDF 已经有较为丰富的应用, 如 IND-CPA 的公钥加密方案、不经意传输、抗碰撞哈希函数等; LTDF 与 ABO-TDF 结合有着更强的应用, 如 IND-CCA 的公钥加密方案. IND-CCA 的公钥加密方案构造原理蕴含在如何基于 LTDF 和 ABO-LTDF 构造更高级的单向陷门函数中(将在章节中阐述), 为了避免重复, 此处不再详述.



以下展示如何给出 LTDF 和 ABO-TDF 的具体构造. 构造的难点是需要巧妙设计密钥对生成算法, 使其可以工作在单射可逆和有损两个模式, 且两种模式在计算意义上不可区分. 设计的思路是令定义域  $X$  是向量空间, 输入  $x$  是向量空间中的元素, 求值公钥  $ek$  是刻画线性变换的矩阵, 函数求值  $f(ek, x)$  的过程就是对输入进行线性变换, 当  $ek$  满秩时, 函数单射可逆; 当  $ek$  非满秩时, 函数有损. 隐藏  $ek$  工作模式的思路则是对其“加密”. 我们称上述技术路线为矩阵式方法.

下面展示矩阵式构造的一个具体例子, 以剥丝抽茧的方式阐明设计思想和关键技术.

**隐藏矩阵生成.** 最简单的满秩矩阵是单位阵, 最简单的非满秩矩阵是全零阵, 两者之间差异显著, 为了保证计算不可区分性, 思路是生成一个伪随机的隐藏矩阵 (concealer matrix)  $\mathbf{M}$  对其加密. 我们期望  $\mathbf{M}$  满足如下结构:  $\mathbf{M}$  的所有行向量均处于同一个一维子空间, 后面可以看到子空间的描述将作为陷门信息使用. 具体的, 隐藏矩阵生成算法  $\text{GenConcealMatrix}(n)$  细节如下:

1. 随机选择  $\mathbf{r} = (r_1, \dots, r_n) \xleftarrow{\text{R}} \mathbb{Z}_p^n$  和  $\mathbf{s} = (s_1, \dots, s_m, 1) \xleftarrow{\text{R}} \mathbb{Z}_p^n \times \{1\}$
2. 计算张量积  $\mathbf{V} = \mathbf{r} \otimes \mathbf{s} = \mathbf{r}^t \mathbf{s} \in \mathbb{Z}_p^{n \times (n+1)}$

$$\mathbf{V} = \left( \begin{array}{cccc|c} r_1 s_1 & r_1 s_2 & \dots & r_1 s_n & r_1 \\ r_2 s_1 & r_2 s_2 & \dots & r_2 s_n & r_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ r_n s_1 & r_n s_2 & \dots & r_n s_n & r_n \end{array} \right)$$

3. 输出  $\mathbf{M} = g^{\mathbf{V}} \in \mathbb{G}^{n \times (m+1)}$  作为隐藏矩阵,  $\mathbf{s}$  作为陷门信息.

$$\mathbf{M} = \left( \begin{array}{cccc|c} g^{r_1 s_1} & g^{r_1 s_2} & \dots & g^{r_1 s_n} & g^{r_1} \\ g^{r_2 s_1} & g^{r_2 s_2} & \dots & g^{r_2 s_n} & g^{r_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g^{r_n s_1} & g^{r_n s_2} & \dots & g^{r_n s_n} & g^{r_n} \end{array} \right)$$

算法前两步的作用是生成特定结构: 通过张量积确保  $\mathbf{V}$  中所有行向量均处于向量  $(s_1, \dots, s_n, 1)$  张成的一维子空间中. 当前向量定义在有限域  $\mathbb{F}_p$  上, 而  $ek$  矩阵不可以定义在有限域  $\mathbb{F}_p$  上, 否则存在高效的算法判定  $ek$  对应的矩阵是否满秩. 令  $\mathbb{G}$  是  $p$  阶循环群, 其中 DDH 假设成立. 可以证明, 如果  $ek$  矩阵定义在  $\mathbb{G}$  上, 那么满秩和非满秩无法有效判定. 因此, 算法的第三步利用从  $\mathbb{F}_p$  到  $\mathbb{G}$  的同构映射  $\phi: t \rightarrow g^t$  将  $\mathbf{V}$  中的所有元素从  $\mathbb{F}_p$  提升到  $\mathbb{G}$  中.

**注记 4.5**

如果将  $\mathbf{s}$  截断为  $\mathbf{s}' = (s_1, \dots, s_n)$ , 那么  $g^{\mathbf{r} \otimes \mathbf{s}'} = (g^{r_i s_j}) \in \mathbb{G}^{n \times n}$  恰好是 Naor-Reingold 基于 DDH 假设的伪随机合成器构造 (pseudorandom synthesizer)

- 伪随机合成器  $f(r, s)$  是满足如下性质的函数: 令  $r_1, \dots, r_n$  和  $s_1, \dots, s_m$  独立随机分布, 当输入  $(r, s)$  取遍  $(r_i, s_j)$  组合时, 输出伪随机.
- Naor 和 Reingold 证明了从  $\mathbb{Z}_p \times \mathbb{Z}_p$  映射到  $\mathbb{G}$  的函数  $f(r, s) = g^{rs}$  是基于 DDH 假设的伪随机合成器.

**引理 4.1**

如果 DDH 假设成立, 那么由  $\text{GenConcealMatrix}(n)$  生成的矩阵  $\mathbf{M} = g^{\mathbf{V}}$  在  $\mathbb{G}^{n \times (n+1)}$  上伪随机.



**证明** 证明的过程分为两个步骤, 我们首先在一行上从左至右逐个列元素进行混合论证, 证明其与  $\mathbf{G}^{n+1}$  上的随机向量计算不可区分, 再利用该结论从上到下逐行进行混合论证, 从而证明隐藏矩阵  $\mathbf{M}$  在  $\mathbf{G}^{n \times (n+1)}$  上伪随机分布.

- 逐列论证: 令  $r \xleftarrow{\text{R}} \mathbb{Z}_p, \mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_p^n, \mathbf{t} \xleftarrow{\text{R}} \mathbb{Z}_p^n$ , 证明如下两个分布计算不可区分:

$$(g^{\mathbf{s}}, g^r, \mathbf{y} = g^{r \cdot \mathbf{s}}) \approx_c (g^{\mathbf{s}}, g^r, \mathbf{y} = g^{\mathbf{t}})$$

证明的方法是设计如下的游戏序列进行混合论证:

$$\begin{array}{cccccc} \text{Hyb}_0 : & g^{\mathbf{s}} & g^{rs_1} & \dots & g^{rs_n} & g^r \\ \text{Hyb}_1 : & g^{\mathbf{s}} & g^{\mathbf{t}_1} & \dots & g^{rs_n} & g^r \\ \text{Hyb}_j : & g^{\mathbf{s}} & g^{t_1} & g^{\mathbf{t}_j} & g^{rs_n} & g^r \\ \text{Hyb}_n : & g^{\mathbf{s}} & g^{t_1} & \dots & g^{\mathbf{t}_n} & g^r \end{array}$$

基于 DDH 假设, 可以证明任意两个相邻的游戏中的分布簇均计算不可区分, 利用 hybrid lemma 立刻可得:  $\text{Hyb}_0 \approx_c \text{Hyb}_1$ .

- 逐行论证: 基于上述结果, 我们再逐行变换, 每次将一行替换成  $\mathbf{G}^{n+1}$  上的随机向量, 再次利用 hybrid lemma 即可证明

$$(g^{\mathbf{s}}, \mathbf{M}) \approx_c (g^{\mathbf{s}}, U_{\mathbf{G}^{n \times (n+1)}}) \quad (4.1)$$

综上,  $\mathbf{M}$  在  $\mathbf{G}^{n \times (n+1)}$  上伪随机分布. □

#### 注记 4.6

公式 (??) 事实上证明了比引理更强的结果, 即在敌手观察到  $g^{\mathbf{s}}$  的情形下,  $\mathbf{M}$  仍与  $\mathbf{G}^{n \times (n+1)}$  上随机矩阵计算不可区分. 在以上两个步骤的证明过程中, 横向的归约损失是  $n$ , 纵向的归约损失为  $n$ , 因此证明的总归约损失是  $n^2$ . 可以利用 DDH 类假设的随机自归约性质 (random self-reducibility) 将归约损失降为  $n$  (to be confirmed).



以下首先展示基于 DDH 假设的 LTDF 构造.

#### 构造 4.3 (DDH-based LTDF)

- Setup( $1^\kappa$ ):** 运行  $\text{GenGroup}(1^\kappa) \rightarrow (\mathbb{G}, g, p)$ , 其中  $\mathbb{G}$  是一个阶为素数  $p$  的循环群, 生成元为  $g$ . 输出  $pp = (\mathbb{G}, g, p)$ .  $pp$  还包括了定义域  $X = \{0, 1\}^n$  和值域  $Y = \mathbb{G}$  的描述.
- GenInjective( $n$ ):** 运行  $\text{GenConcealMatrix}(n) \rightarrow (g^{\mathbf{V}}, \mathbf{s})$ , 输出  $g^{\mathbf{Z}} = g^{\mathbf{V} + \mathbf{I}'}$  作为公钥  $ek$ , 其中  $\mathbf{I}' \in \mathbb{Z}_p^{n \times (n+1)}$  由  $n$  阶单位阵在最右侧补上全零列扩展得来 (即  $(\mathbf{e}_1, \dots, \mathbf{e}_n, \mathbf{0})$ ), 输出  $\mathbf{s}$  作为函数的陷门  $td$ .

$$g^{\mathbf{Z}} = \left( \begin{array}{cccc|c} g^{r_1 s_1 + 1} & g^{r_1 s_2} & \dots & g^{r_1 s_n} & g^{r_1} \\ g^{r_2 s_1} & g^{r_2 s_2 + 1} & \dots & g^{r_2 s_n} & g^{r_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g^{r_n s_1} & g^{r_n s_2} & \dots & g^{r_n s_n + 1} & g^{r_n} \end{array} \right)$$

- GenLossy( $n$ ):**  $\text{GenConcealMatrix}(n) \rightarrow g^{\mathbf{V}}$ , 输出  $g^{\mathbf{Z}} = g^{\mathbf{V}}$  作为公钥  $ek$ , 陷门  $td$  为  $\perp$ .

$$g^{\mathbf{Z}} = \left( \begin{array}{cccc|c} g^{r_1 s_1} & g^{r_1 s_2} & \dots & g^{r_1 s_n} & g^{r_1} \\ g^{r_2 s_1} & g^{r_2 s_2} & \dots & g^{r_2 s_n} & g^{r_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g^{r_n s_1} & g^{r_n s_2} & \dots & g^{r_n s_n} & g^{r_n} \end{array} \right)$$

- $\text{Eval}(ek, \mathbf{x})$ : 以  $ek = g^{\mathbf{Z}}$  和  $\mathbf{x} \in \{0, 1\}^n$  为输入, 计算  $\mathbf{y} \leftarrow g^{\mathbf{xZ}} \in \mathbb{G}^{n+1}$ .
- $\text{TdInv}(td, \mathbf{y})$ : 解析  $td = \mathbf{s} = (s_1, \dots, s_n)$ , 对每个  $i \in [n]$ , 计算  $a_i = y_i / y_{n+1}^{s_i}$  并输出  $x_i \in \{0, 1\}$  s.t.  $a_i = g^{x_i}$ .



#### 定理 4.4

基于 DDH 假设, 上述构造是一族  $(n, \log p)$ -LTDF.



**证明** 单射可逆模式的正确性由算法  $\text{TdInv}$  的正确性保证. 在有损模式下, 所有输出  $\mathbf{y}$  都具有  $g^{c\mathbf{s}}$  的形式, 其中  $c = \langle \mathbf{x}, \mathbf{r} \rangle \in \mathbb{Z}_p$ . 向量  $\mathbf{s}$  被  $ek$  固定, 因此  $\text{Img}(f_{ek}) \leq p$ .

单射可逆模式和有损模式的计算不可区分性由  $\text{GenConcealMatrix}$  输出的伪随机性 (引理 ??) 保证.  $\square$

下面展示如何基于 DDH 假设构造 ABO-LTDF.

#### 构造 4.4 (DDH-based ABO-LTDF)

- $\text{Setup}(1^\kappa)$ : 运行  $\text{GenGroup}(1^\kappa) \rightarrow (\mathbb{G}, g, p)$ , 其中  $\mathbb{G}$  是一个阶为素数  $p$  的循环群, 生成元为  $g$ . 输出  $pp = (\mathbb{G}, g, p)$ .  $pp$  还包括定义域  $X = \{0, 1\}^n$ 、值域  $Y = \mathbb{G}$  和分支集合  $B = \mathbb{Z}_p$  的描述.
- $\text{Gen}(pp, b^*)$ : 运行  $\text{GenConcealMatrix}(n) \rightarrow (g^{\mathbf{V}}, \mathbf{s})$ , 输出  $g^{\mathbf{Z}} = g^{\mathbf{V} - b^* \mathbf{I}'}$  作为公钥  $ek$ , 其中  $\mathbf{I}' = (\mathbf{e}_1, \dots, \mathbf{e}_n, \mathbf{0}) \in \mathbb{Z}_p^{n \times (n+1)}$ , 输出  $(b^*, \mathbf{s})$  作为陷门  $td$ .
- $\text{Eval}(ek, b, \mathbf{x})$ : 以  $ek = g^{\mathbf{Z}}$  和  $\mathbf{x} \in \{0, 1\}^n$  为输入, 计算  $\mathbf{y} \leftarrow g^{\mathbf{x}(\mathbf{Z} + b(\mathbf{e}_1, \dots, \mathbf{e}_n, \mathbf{0}))} \in \mathbb{G}^{n+1}$ , 记为  $y \leftarrow f(ek, b, x)$  或  $y \leftarrow f_{ek,b}(x)$ .
- $\text{TdInv}(td, b, \mathbf{y})$ : 解析  $td$  为  $\mathbf{s} = (s_1, \dots, s_n)$ , 对每个  $i \in [n]$ , 计算  $a_i = y_i / y_{n+1}^{s_i}$  并输出  $x_i \in \{0, 1\}$  s.t.  $a_i = g^{(b-b^*)x_i}$ .



$$\text{Gen}(pp, b^*) \rightarrow (ek, \mathbf{s})$$

$$\begin{aligned} \text{GenConcealMatrix}(n) &= g^{\mathbf{V}} \\ \mathbf{x} \in \mathbb{Z}_2^n \times &\left( \begin{array}{cccc|c} g^{r_1 s_1} & g^{r_1 s_2} & \dots & g^{r_1 s_n} & g^{r_1} \\ g^{r_2 s_1} & g^{r_2 s_2} & \dots & g^{r_2 s_n} & g^{r_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g^{r_n s_1} & g^{r_n s_2} & \dots & g^{r_n s_n} & g^{r_n} \end{array} \right) -b^*(\mathbf{e}_1, \dots, \mathbf{e}_n, \mathbf{0}) \\ &+ b(\mathbf{e}_1, \dots, \mathbf{e}_n, \mathbf{0}) \\ &\rightarrow \mathbf{y} \in \mathbb{G}^{n+1} \end{aligned}$$

$$\text{DDH} \Rightarrow \approx_c U_{\mathbb{G}^{n \times (n+1)}}$$

#### 定理 4.5

基于 DDH 假设, 上述构造是一族分支集合为  $B = \mathbb{Z}_p$  的  $(n, \log p)$ -ABO-TDF.



**证明** 容易验证, 当  $b \neq b^*$  时,  $\mathbf{V} + (b - b^*)\mathbf{I}'$  矩阵满秩,  $f_{ek,b}$  单射且可高效求逆; 当  $b = b^*$  时, 矩阵  $\mathbf{V} + (b - b^*)\mathbf{I}'$  的秩为 1,  $\text{Img}(f_{ek,b}) \leq p$ . 有损分支隐藏性由  $\text{GenConcealMatrix}$  输出的伪随机性 (引理 ??) 保证.  $\square$

#### 注记 4.7

为了确保求逆算法的高效性, 以上构造有两个重要的设定: (1) 首先在  $\text{ConcealMatrix}$  设置了辅助列  $(g^{r_1}, \dots, g^{r_n})^T$ , 便于计算出  $a_i = g^{x_i}$ ; (2) 从  $a_i$  中计算  $x_i$  需要求解离散对数, 因此定义域  $X$  设定为  $\mathbb{Z}_2^n$ , 其中 2 可以进一步放宽至  $\kappa^{O(1)}$  (关于  $\kappa$  的多项式规模), 以保证可以在多项式时间完成离散对数求解.



## 扩展与深化

注意到在公钥加密的选择密文安全定义中敌手对解密预言机的访问权限是全除一的, 由此可以看出全除一有损陷门函数的应用局限于“全除一”类安全的密码方案设计. Hofheinz [Hofheinz-EUROCRYPT-2012] 引入了全

除多有损陷门函数, 将有损分支的数量从 1 扩展到  $\text{poly}(\kappa)$ , 并展示了其在选择打开选择密文安全 (selective opening chosen-ciphertext security) 中的应用. 在有损陷门函数的应用中, 我们通常期望有损模式下函数丢失的信息尽可能的多, 即像集尽可能的小. 这是因为单射和有损模式的反差越大, 所蕴含的结果越强, 如更高的泄漏容忍能力、更紧的安全归约等. 但凡事有度, 物极必反, 在常规的一致归约 (universal reduction) 模型下, 有损模式的像集尺寸  $2^\tau$  不能过小, 至少是关于计算安全参数  $\kappa$  的超多项式规模, 否则 PPT 的敌手可以通过生日攻击有效的区分单射和有损模式. Zhandry [Zhandry-CRYPTO-2016] 创造性的提出了极度有损函数 (ELF, extremely lossy functions). 在 ELF 中, 有损模式下函数的像集可以缩小至关于计算安全参数  $\kappa$  的多项式规模, 只要在指定 PPT 敌手的生日攻击能力之外即可. ELF 的有损模式之所以能够打破像集多项式界的关键在更为精细的个体归约 (individual reduction) 模型 [Deng-EUROCRYPT-2017] 下进行安全性证明. Zhandry 基于不可区分程序混淆给出了 ELF 的构造, 并展示了其强大的应用. 在无须求逆的应用场景中, 不仅不需要陷门, 甚至是单射的性质也可以弱化. 陈等 Chen-CT-RSA-2018 根据这一观察, 提出了规则有损函数 (RLF, regular lossy functions). 相比标准的 LTDF, RLF 将单射可逆模式放宽至规则有损, 即每个像的原像集合大小相同. 正是这一弱化, 使得 RLF 不仅有更加高效的具体构造, 也可由哈希证明系统通用构造得出, 并在抗泄漏密码学领域有着重要的应用.

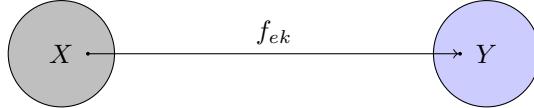
### 4.1.3 相关积单向陷门函数

山重水复疑无路, 柳暗花明又一村.

—宋·陆游《过山西村》

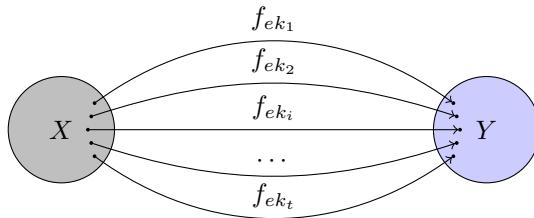
令  $\mathcal{F} = \{f_{ek} : X \rightarrow Y\}$  是一族单向陷门函数, 可以自然对  $\mathcal{F}$  进行  $t$  重延拓, 得到  $\mathcal{F}^t = \{g_{ek_1, \dots, ek_t} : X^t \rightarrow Y^t\}$ , 其中  $f_{ek_1, \dots, ek_t}^t(x_1, \dots, x_t) := (f_{ek_1}(x_1), \dots, f_{ek_t}(x_t))$ . 我们称  $\mathcal{F}^t$  为  $\mathcal{F}$  的  $t$  重积 ( $t$ -wise product).

$$\mathcal{F} = \{f_{ek} : X \rightarrow Y\}$$



$t$ -wise product

$$\mathcal{F}^t = \{f_{ek_1, \dots, ek_t}^t : X^t \rightarrow Y^t\}$$



#### 定理 4.6

如果  $\mathcal{F} = \{f_{ek}\}$  是一族单向函数, 那么它的  $t$  重积  $\mathcal{F}^t = \{f_{ek_1, \dots, ek_t}^t\}$  也是一族单向函数.



**证明** 证明的思路简述如下: 如果存在 PPT 的敌手  $\mathcal{A}$  打破  $\mathcal{F}^t$  的单向性, 那么其必然以不可忽略的优势对  $t$  个单向函数的实例  $f_{ek_i}(\cdot)$  求逆, 这显然与  $\mathcal{F}$  的单向性冲突, 因此得证.

#### 注记 4.8

上述定理在  $ek_1 = \dots = ek_t$  (即所有  $f_{ek_i}$  相同) 时仍然成立, 该情形恰好对应单向函数的单向性放大 (one-wayness amplification).



需要注意的是,  $f_{ek_1, \dots, ek_t}^t$  单向性成立的前提是各分量输入  $x_i$  独立随机采样, 而当各分量输入相关时, 单向性则未必成立, 这是因为多个像的分量交叉组合可能会泄漏原像的信息.

**构造 4.5 (反例构造)**

令  $\hat{f}_{ek} : X = \{0,1\}^n \rightarrow Y$  是一个单向函数, 构造一个新的函数  $f_{ek} : \{0,1\}^{2n} \rightarrow Y||\{0,1\}^n$  如下:

$$f_{ek}(x_l||x_r) := \hat{f}_{ek}(x_l)||x_r$$



在上述构造中,  $f_{ek}$  以  $\hat{f}_{ek}$  为核, 因此如果  $\hat{f}_{ek}$  是单向的, 那么  $f_{ek}$  也是单向的. 考察 2 重积  $f_{ek_1,ek_2}^2$  在相关输入  $(x_1 = x_l||x_r, x_2 = x_r||x_l)$  下的行为:

$$f_{ek_1,ek_2}^2(x_1, x_2) := (f_{ek_1}(x_1), f_{ek_2}(x_2)) = \hat{f}_{ek_1}(x_l)||x_r||\hat{f}_{ek_2}(x_r)||x_l$$

根据  $f_{ek}$  的设计,  $f_{ek_1,ek_2}^2$  的原像信息  $(x_1, x_2)$  可以从像中的  $(x_r, x_l)$  完全恢复出来, 因此在输入呈如上相关时并不满足单向性. 上述反例构造的精髓是设计具有特殊结构的单向函数.

反例 ??说明单向函数的  $t$  重积在输入相关时并不一定仍然单向. Alon 和 Rosen [RS-TCC-2009] 引入了相关积 (correlated products) 单向陷门函数, 定义如下: 要求函数的  $t$  重积在输入分量相关时仍然保持单向性

**定义 4.5 (相关积单向性)**

令  $\mathcal{F} : X \rightarrow Y$  是一族单向函数,  $\mathcal{C}_t$  是定义在  $X^t$  上的分布 (分量相关). 如果  $\mathcal{F}$  的  $t$  重积  $\mathcal{F}^t : X^t \rightarrow Y^t$  在  $\mathcal{C}_t$  相关积下仍然是单向的, (即对于任意 PPT 敌手  $\mathcal{A}$ , 其在如下的安全实验中优势是可忽略的)

$$\Pr \left[ f_{ek_1, \dots, ek_t}^t(x') = y^* : \begin{array}{l} ek_i \leftarrow \text{Gen}(\kappa); \\ (x_1^*, \dots, x_t^*) \xleftarrow{\text{R}} \mathcal{C}_t; \\ y^* \leftarrow (f_{ek_1}(x_1^*), \dots, f_{ek_t}(x_t^*)); \\ x' \leftarrow \mathcal{A}(ek_1, \dots, ek_t, y^*); \end{array} \right]$$

则称  $\mathcal{F}$  是  $\mathcal{C}_t$  相关积安全的 (correlated-product secure). 该定义可以自然延拓到陷门函数场景.



在给出 CP-TDF 的定义后, 接下来需要研究的问题是分析什么样的  $\mathcal{F}$  在何种相关积下仍然单向. 本书中聚焦最为典型的一种  $\mathcal{C}_t$  相关积——均匀重复相关积  $\mathcal{U}_t$ , 即  $x_1 \xleftarrow{\text{R}} X$  且  $x_1 = \dots = x_t$ . Rosen 和 Segev [RS-TCC-2009] 基于 LTDF 给出了 CP-TDF 的一个通用构造, 揭示了两者之间的联系.

**定理 4.7**

令  $\mathcal{F}$  是一族  $(n, \tau)$ -LTDF, 那么  $\mathcal{F}$  在相关积  $\mathcal{U}_t$  下仍然单向, 其中  $t \leq (n - \omega(\log \kappa))/\tau$ .



**证明** 证明通过以下的游戏序列完成, 敌手在 Game<sub>i</sub> 中成功的事件为  $S_i$ .

Game<sub>0</sub>: 对应真实的相关积单向性实验, 函数以单射模式运作

- $\mathcal{CH}$  独立运行  $\mathcal{F}.\text{GenInjective}(\kappa)$  算法  $t$  次, 生成  $ek = (ek_1, \dots, ek_t)$  并将其发送给  $\mathcal{A}$ .
- $\mathcal{CH}$  随机采样  $x^* \xleftarrow{\text{R}} X$ , 计算  $y^* \leftarrow (f_{ek_1}(x^*), \dots, f_{ek_t}(x^*))$  并将  $y^*$  发送给  $\mathcal{A}$ .
- $\mathcal{A}$  输出  $x'$ , 当且仅当  $x' = x^*$  时成功.

根据定义, 我们有:

$$\text{Adv}_{\mathcal{A}}^{\text{Game}_0}(\kappa) = \Pr[S_0]$$

Game<sub>1</sub>: 与上一游戏相同, 区别在于函数切换到有损模式运作

- $\mathcal{CH}$  独立运行  $\mathcal{F}.\text{GenLossy}(\kappa)$  算法  $t$  次, 生成  $ek = (ek_1, \dots, ek_t)$  并将其发送给  $\mathcal{A}$ .

$$\text{Adv}_{\mathcal{A}}^{\text{Game}_1}(\kappa) = \Pr[S_1]$$

**断言 4.3**

基于 LTDF 的单射/有损模式不可区分性, 任意 PPT 敌手  $\mathcal{A}$  在 Game<sub>0</sub> 和 Game<sub>1</sub> 中的成功概率差可忽略.



**证明** Game<sub>0</sub> 和 Game<sub>1</sub> 的差别在于  $(ek_1, \dots, ek_t)$  的生成模式. 基于 LTDF 的单射/有损模式不可区分性和 hybrid argument, 可以推出 Game<sub>0</sub>  $\approx_c$  Game<sub>1</sub>, 进而保证  $|\Pr[S_0] - \Pr[S_1]| \leq \text{negl}(\kappa)$ .

#### 断言 4.4

对于任意敌手  $\mathcal{A}$ (即使拥有无穷计算能力),  $\Pr[S_1] = \text{negl}(\lambda)$ .



**证明** 在 Game<sub>1</sub> 中, 函数工作在有损模式, 因此像集的大小至多为  $2^{t\tau}$ , 由 chainning lemma ??可知  $x^*$  的平均最小熵  $\tilde{H}_\infty(x^*|y^*) \geq n - t\tau$ . 根据定理前提条件中的参数选取, 有  $\tilde{H}_\infty(x^*|y^*) \geq \omega(\log \kappa)$ , 因此断言得证.  $\square$

综上, 我们有  $\Pr[S_0] \leq \text{negl}(\kappa)$ . 定理得证!



**笔记** 追求简洁、消除冗余在科学和文学领域似乎都是真理. 然而, 正如知乎上一篇文章 [zhihu-essay] 所说:“尽管我们偏爱简洁, 但冗余让一切皆有可能”. 相关积单向函数的定义和构造就充分诠释了冗余的力量.

### 4.1.4 自适应单向函数

他强由他强, 清风拂山冈; 他横由他横, 明月照大江; 他自狠来他自恶, 我自一口真气足.

— 达摩《九阳真经》

构造 ??仅具备 IND-CPA 安全性, 并不一定能够满足 IND-CCA 安全性. 这是因为底层的 TDF 可能具备诸如同态等优良的代数性质, 使得上层 PKE/KEM 方案具有可延展性. 从安全归约的角度分析, 归约算法无法对解密/解封装询问做出正确的应答. 基于以上分析, 一个自然的问题是: TDF 满足何种增强的性质才能够使得构造 ??满足 IND-CCA 安全性.

Kiltz, Mohassel 和 O’Neill [KMO-EUROCRYPT-2010] 提出了自适应单向性 (adaptive one-wayness), 该性质要求 TDF 的单向性在敌手能够访问求逆谕言机的情况下仍然成立.

#### 定义 4.6 (自适应单向性)

令  $\mathcal{F}$  是一族陷门函数, 定义敌手  $\mathcal{A}$  的优势如下:

$$\Pr \left[ x' \in f_{ek}^{-1}(y^*) : \begin{array}{l} pp \leftarrow \text{Setup}(\kappa); \\ (ek, td) \leftarrow \text{KeyGen}(pp); \\ x^* \xleftarrow{\text{R}} X, y^* \leftarrow f_{ek}(x^*); \\ x' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{inv}}}(ek, y^*); \end{array} \right]$$

其中  $\mathcal{O}_{\text{inv}}$  是求逆谕言机,  $\forall x \neq x^*, \mathcal{O}_{\text{inv}}(y) = \text{TdInv}(td, y)$ . 如果任意 PPT 敌手  $\mathcal{A}$  在上述安全试验中的优势均为  $\text{negl}(\kappa)$ , 那么则称  $\mathcal{F}$  是自适应单向的.



为了方便在公钥加密场景中的应用, 引入自适应伪随机性如下.

#### 定义 4.7 (自适应伪随机性)

令  $\mathcal{F}$  是一族单向函数,  $\text{hc}$  是其 hardcore function. 定义敌手  $\mathcal{A}$  的优势如下:

$$\Pr \left[ \beta' = \beta : \begin{array}{l} (ek, td) \leftarrow \mathcal{F}.\text{Gen}(\lambda); \\ x^* \xleftarrow{\text{R}} X, y^* \leftarrow f_{ek}(x^*); \\ k_0^* \leftarrow \text{hc}(x^*), k_1^* \xleftarrow{\text{R}} K, \beta \leftarrow \{0, 1\}; \\ \beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{inv}}}(ek, y^*, k_\beta^*); \end{array} \right] - \frac{1}{2}$$

其中  $\mathcal{O}_{\text{inv}}$  是求逆谕言机,  $\text{hc}$  是  $\mathcal{F}$  的 hardcore function. 如果任意 PPT 敌手  $\mathcal{A}$  在上述安全试验中的成功概率均为  $\text{negl}(\kappa)$ , 那么则称  $\text{hc}$  的是自适应伪随机的.



**推论 4.1**

$\mathcal{F}$  的自适应单向性蕴含 hardcore function 的自适应伪随机性.



**证明** Goldreich-Levin 定理的证明可以平行推广到求逆预言机  $\mathcal{O}_{\text{inv}}$  存在的情形下,  $\text{hc}(x^*)$  自适应伪随机性由  $x^*$  的自适应单向性保证.  $\square$

自适应单向陷门函数 (ATDF, adaptive TDF) 定义简洁, 威力强大, 将 ATDF 代入构造 ?? 中, 得到的 KEM 满足 IND-CCA 安全. 从安全归约的角度观察, ATDF 的自适应单向性是为 KEM 的 CCA 安全性量身定制的, 都是“全除一”类型的安全定义. 那么, 如何构造 ATDF 呢? 文献 [KMO-EUROCRYPT-2010] 一方面基于实例独立 (instance-independent) 假设给出 ATDF 的具体构造, 一方面分别基于 LTDF 和 CP-TDF 给出了 ATDF 的两个通用构造.

以下我们聚焦 ATDF 的通用构造, 首先展示如何基于 LTDF 构造 ATDF. 构造的技术困难点在于 ATDF 的安全试验中挑战者  $\mathcal{CH}$  向对手  $\mathcal{A}$  提供了“全除一”式解密预言机  $\mathcal{O}_{\text{inv}}$ , 而 LTDF 的安全试验中并没有提供类似的预言机访问接口. 因此, 构造的思路是通过引入精巧的结构完成解密预言机  $\mathcal{O}_{\text{inv}}$  的模拟. 总体的思路如下:

- 令 ATDF 的像  $y$  形如  $(y_0, y_1)$ , 确保  $y_1$  由  $y_0$  唯一确定, 可行的设计是计算原像  $x$  的 LTDF 值作为  $y_0$ , 再以  $y_0$  为分支编号计算  $x$  的 ABO-TDF 值作为  $y_1$ .

$$y_0 \leftarrow f(ek_{\text{ltdf}}, x), y_1 \leftarrow g(ek_{\text{abo}}, y_0, x)$$

- 上述设计利用 ABO-TDF 的相对分支标签的“全除一”求逆陷门嵌入了相对于像的“全除一”可逆结构.

**构造 4.6 (基于 LTDF 和 ABO-TDF 的 ATDF 构造)**

构造所需的组件是:

- $(n, \tau_1)$ -LTDF -  $\mathcal{F} : X \rightarrow Y_1$ ;
- $(n, \tau_2)$ -ABO-TDF -  $\mathcal{G} : X \rightarrow Y_2$  w.r.t.  $Y_1$  作为分支集合;

其中  $\log_2 |X| = n$ ,  $\log_2 |Y_1| = m_1$ ,  $\log_2 |Y_2| = m_2$ .

构造 ATDF :  $X \rightarrow Y_1 \times Y_2$  如下:

- $\text{Setup}(1^\kappa)$ : 以安全参数  $\kappa$  为输入, 计算  $pp_{\text{ltdf}} \leftarrow \mathcal{F}.\text{Setup}(1^\kappa)$ ,  $pp_{\text{abo}} \leftarrow \mathcal{G}.\text{Setup}(1^\kappa)$ , 输出  $pp = (pp_{\text{ltdf}}, pp_{\text{abo}})$ .
- $\text{Gen}(pp)$ : 以公开参数  $pp = (pp_{\text{ltdf}}, pp_{\text{abo}})$  为输入, 计算  $(ek_{\text{ltdf}}, td_{\text{ltdf}}) \leftarrow \mathcal{F}.\text{GenInjective}(pp_{\text{ltdf}})$ ,  $(ek_{\text{abo}}, td_{\text{abo}}) \leftarrow \mathcal{G}.\text{Gen}(pp_{\text{abo}}, 0^{m_1})$ , 输出求值公钥  $ek = (ek_{\text{ltdf}}, ek_{\text{abo}})$  和陷门  $td = (td_{\text{ltdf}}, td_{\text{abo}})$ .
- $\text{Eval}(ek, x)$ : 以求值公钥  $ek = (ek_{\text{ltdf}}, ek_{\text{abo}})$  和  $x \in \{0, 1\}^n$  为输入, 计算  $y_1 \leftarrow f_{ek_{\text{ltdf}}}(x)$ ,  $y_2 \leftarrow g_{ek_{\text{abo}}}(y_1, x)$ , 输出  $y = (y_1, y_2)$ .
- $\text{TdInv}(td, y)$ : 以陷门  $td = (td_{\text{ltdf}}, td_{\text{abo}})$  和  $y = (y_1, y_2)$  为输入, 计算  $x \leftarrow \mathcal{F}.\text{TdInv}(td_{\text{ltdf}}, y_1)$ , 验证  $y_2 \stackrel{?}{=} g_{ek_{\text{abo}}}(y_1, x)$ : 如果是输出  $x$ , 否则输出  $\perp$ .



上述构造的正确性显然成立. 安全性由如下定理保证:

**定理 4.8**

基于 LTDF 和 ABO-TDF 的安全性, 上述构造在  $n - \tau_1 - \tau_2 \geq \omega(\log \kappa)$  构成一族 ATDF.



**证明** 令  $(x^*, y^* = (y_1^*, y_2^*))$  为单向挑战实例, 其中  $x^*$  是原像,  $y^*$  是像. 证明的思路将像  $y^* = (y_1^*, y_2^*)$  的计算方式从单射无损模式逐步切换到有损模式, 最终在信息论意义上论证单向性.

Game<sub>0</sub>: 真实的 ATDF 单向性试验.  $\mathcal{CH}$  与  $\mathcal{A}$  交互如下:

- 初始化:  $\mathcal{CH}$  进行如下操作
  - 运行  $pp_{\text{ltdf}} \leftarrow \mathcal{F}.\text{Setup}(1^\kappa)$ ,  $pp_{\text{abo}} \leftarrow \mathcal{G}.\text{Setup}(1^\kappa)$ ;
  - 计算  $(ek_{\text{ltdf}}, td_{\text{ltdf}}) \leftarrow \mathcal{F}.\text{GenInjective}(pp_{\text{ltdf}})$ ,  $(ek_{\text{abo}}, td_{\text{abo}}) \leftarrow \mathcal{G}.\text{Gen}(pp_{\text{abo}}, 0^{m_1})$ ;

3. 发送  $pp = (pp_{\text{ltdf}}, pp_{\text{abo}})$  和  $ek = (ek_{\text{ltdf}}, ek_{\text{abo}})$  给  $\mathcal{A}$ .
- 挑战:  $\mathcal{CH}$  随机选取  $x^* \xleftarrow{\text{R}} X$ , 计算  $y_1^* \leftarrow f_{ek_{\text{ltdf}}}(x^*)$ ,  $y_2^* \leftarrow g_{ek_{\text{abo}}}(y_1^*, x^*)$ , 发送  $y^* = (y_1^*, y_2^*)$  给  $\mathcal{A}$ .
  - 求逆询问: 当  $\mathcal{A}$  向  $\mathcal{O}_{\text{inv}}$  询问  $y = (y_1, y_2)$  的原像时,  $\mathcal{CH}$  分情况应答如下:

- $y_1 = y_1^*$ : 直接返回  $\perp$ .

- $y_1 \neq y_1^*$ : 首先计算  $x \leftarrow \mathcal{F}.\text{TdInv}(td_{\text{ltdf}}, y_1)$ , 如果  $y_2 = g_{ek_{\text{abo}}}(y_1, x)$  则返回  $x$ , 否则返回  $\perp$ .

根据 ATDF 像的生成方式可知, 第一部分完全确定了第二部分, 当  $y_1 = y_1^*$  时, 如  $y_2 = y_2^*$  则  $\mathcal{A}$  的询问为禁讯点, 如  $y_2 \neq y_2^*$  则像的格式不正确. 基于以上分析,  $\mathcal{CH}$  在应答形如  $(y_1^*, y_2)$  的求逆询问时, 无须进一步检查第二部分  $y_2$ , 直接返回  $\perp$  即可保证应答的正确性.

**Game<sub>1</sub>**: 在 Game<sub>0</sub> 中  $\mathcal{CH}$  使用  $\mathcal{F}$  的陷门进行求逆, 因此  $\mathcal{F}$  必须工作在单射可逆模式. 为了将  $y_1^*$  的计算模式切换到有损模式, 需要利用  $\mathcal{G}$  的陷门进行求逆. 注意到在 Game<sub>0</sub> 中  $\mathcal{G}$  的“全除一”陷门根据预先设定的有损分支  $0^{m_1}$  生成, 因此必须先激活再使用, 因此 Game<sub>1</sub> 的设计目的是为激活做准备:

- $\mathcal{CH}$  在初始化阶段即随机采样  $x^* \xleftarrow{\text{R}} X$ , 并计算  $y_1^* \leftarrow f_{ek_{\text{ltdf}}}(x^*)$ .

与 Game<sub>0</sub> 相比, Game<sub>1</sub> 仅将上述操作从挑战阶段提前至初始化阶段, 敌手的视图没有发生任何变化, 因此有:

$$\text{Game}_0 \equiv \text{Game}_1$$

**Game<sub>2</sub>**: 上一游戏已经做好激活  $\mathcal{G}$  陷门的准备, 因此在 Game<sub>2</sub> 中将预设的有损分支值由  $0^{m_1}$  替换为  $y_1^*$  完成激活:

- $(ek_{\text{abo}}, td_{\text{abo}}) \leftarrow G.\text{Gen}(pp_{\text{abo}}, y_1^*)$

由 ABO-TDF 的有损分支隐藏性质, 可以得到:

$$\text{Game}_1 \approx_c \text{Game}_2$$

**Game<sub>3</sub>**: 使用  $\mathcal{G}$  的陷门  $td_{\text{abo}}$  应答求逆询问, 当  $\mathcal{A}$  发起询问  $y = (y_1, y_2)$  时,  $\mathcal{CH}$  分情形应答如下:

- $y_1 = y_1^*$ : 直接返回  $\perp$ .
- $y_1 \neq y_1^*$ : 计算  $x \leftarrow \mathcal{G}.\text{TdInv}(td_{\text{abo}}, y_1, y_2)$ , 如果  $y_1 = f_{ek_{\text{ltdf}}}(x)$  则返回  $x$ , 否则返回  $\perp$ .

像的生成方式和  $\mathcal{G}$  求逆算法的正确性和保证了  $\mathcal{O}_{\text{inv}}$  应答的正确性, 因此有:

$$\text{Game}_2 \equiv \text{Game}_3$$

**Game<sub>4</sub>**: 将  $y_1^*$  的生成方式切换到有损模式

- $\mathcal{CH}$  在初始化阶段计算  $(ek_{\text{ltdf}}, \perp) \leftarrow \mathcal{F}.\text{GenLossy}(pp_{\text{ltdf}})$

LTDf 的单射/有损模式的计算不可区分性保证了

$$\text{Game}_3 \approx_c \text{Game}_4$$

#### 断言 4.5

对任意的敌手  $\mathcal{A}$ (即使拥有无穷的计算能力) 均有  $\text{Adv}_{\mathcal{A}}(\text{Game}_4) = \text{negl}(\kappa)$ .



**证明** 在 Game<sub>4</sub> 中, 函数  $f_{ek_{\text{ltdf}}}(\cdot)$  有损且像集大小至多为  $2^{\tau_1}$ , 函数  $g_{ek_{\text{abo}}}(y_1^*, \cdot)$  有损且像集大小至多为  $2^{\tau_2}$ . 因此  $y_1^*$  和  $y_2^*$  均在信息论意义上损失了原像  $x^*$  的信息, 在敌手  $\mathcal{A}$  的视图中,  $x^*$  的平均最小熵为  $\tilde{H}_{\infty}(x^* | (y_1^*, y_2^*)) \geq H_{\infty}(x^*) - \tau_1 - \tau_2 = n - \tau_1 - \tau_2 \geq \omega(\log \kappa)$ . 从而对于任意敌手  $\mathcal{A}$  均有:

$$\text{Adv}_{\mathcal{A}}(\text{Game}_4) = \text{negl}(\kappa)$$

断言得证! □

综上, 定理得证! □

### 注记 4.9

上述构造的设计思想值得读者反复拆解，体会其精妙之处。上述 ATDF 构造在形式上与 Naor-Yung 的双钥加密有异曲同工之处：分别使用  $f_{ek_{\text{tdf}}}(\cdot)$  和  $g_{ek_{\text{abo}}}(\cdot, \cdot)$  两个函数计算原像的函数值作为像。一个自然的想法是上述构造显得冗余，是否仅用 ABO-TDF 即可呢？答案是否定的，如果仅依赖 ABO-TDF 构造 ATDF，需要满足以下四点：

- 求值分支可由输入公开确定计算得出，以确保 ATDF 是公开可计算函数。
- 像所对应的求值分支可由像中计算得出，以确保 ATDF 的求逆算法可以基于 ABO-TDF 的求逆算法设计。
- 在安全归约中势必需要将 ATDF 的单向性建立在 ABO-TDF 的信息有损性上，也即  $y^*$  是  $x^*$  在有损分支的求值。

上述三点潜在要求 ATDF 的像包含两个部分，一部分是原像对应的分支值，一部分是 ABO-TDF 在该分支值下的像，这使得在安全证明时存在如下两个障碍：

1. 分支值泄漏原像的多少信息难以确定
2. 敌手可以从挑战的像中计算出有损分支值，从而可以发起关于有损分支的求逆询问，而归约算法无法应答

通过上述的拆解分析，便可看出 ATDF 设计的必然性。引入 LTDF 并将分支值设定为原像的 LTDF 值有三重作用：

- LTDF 的陷门确保了 ATDF 构造存在功能完备的陷门。
- 可将分支值泄漏的关于原像信息量控制在指定范围。
- 分支值完全确定了像，从而使得 ABO-TDF 的陷门在归约证明中可用于模拟求逆预言机  $\mathcal{O}_{\text{inv}}$ 。

$\text{LTDF+ABO-TDF} \Rightarrow \text{ATDF}$  的设计思路有如二级运载火箭，第一级运载火箭 (LTDF) 在完成推动后从单射切换到有损模式，同时激活第二级运载火箭 (ABO-TDF)。



我们再展示如何基于 CP-TDF 构造 ATDF。构造的难点是在归约证明中归约算法如何在不掌握全部 CP-TDF 实例陷门的情况下正确模拟  $\mathcal{O}_{\text{inv}}$ 。大体的设计思路和以上基于有损陷门函数构造 LTDF 相似，通过多重求值引入冗余结构，从而使得归约算法在掌握部分 CP-TDF 实例陷门时能够正确应答求逆询问。

- 设计像  $y$  形如  $(y_0, y_1, \dots, y_n)$ ，确保  $y_0$  能够惟一确定  $(y_1, \dots, y_n)$ 
  - 令  $y_0$  是原像的 CP-TDF 函数值，目的是确保  $y_0$  不会破坏最终 ATDF 函数的单向性
  - 令  $(y_1, \dots, y_n)$  是关于原像  $x$  的  $|y_0| = n$  重冗余函数求值
- 嵌入“全除一”求逆结构
  - 对  $y_0^*$  进行比特分解：归约算法使用 Dolev-Dwork-Naor(DDN) 类技术逐比特嵌入对应的陷门，使得对于点  $y = (y_0, y_1, \dots, y_n)$  处的求逆询问：
    1.  $y_0 = y_0^*$ : 归约算法可根据  $\mathcal{O}_{\text{inv}}$  的定义直接拒绝，返回  $\perp$
    2.  $y_0 \neq y_0^*$ :  $\mathcal{R}$  可至少寻找到一个可用陷门用于应答  $\mathcal{O}_{\text{inv}}$ 。

### 构造 4.7 (基于 CP-TDF 的 ATDF)

构造所需组件：单射 CP-TDF  $\mathcal{F} : X \rightarrow \{0, 1\}^n$

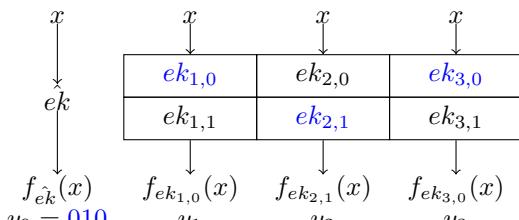
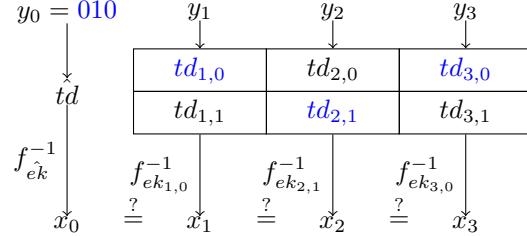
构造 ATDF:  $X \rightarrow \{0, 1\}^{n(n+1)}$  如下：

- $\text{Setup}(1^\kappa)$ : 运行  $pp \leftarrow \mathcal{F}.\text{Setup}(1^\kappa)$ ，输出  $pp$  作为公开参数。
- $\text{KeyGen}(pp)$ : 以公开参数  $pp$  为输入
  1. 计算  $(\hat{ek}, \hat{td}) \leftarrow \mathcal{F}.\text{KeyGen}(\lambda)$ ；
  2. 对于  $b \in \{0, 1\}$  和  $i \in [n]$ ，计算  $(ek_{i,b}, td_{i,b}) \leftarrow \mathcal{F}.\text{KeyGen}(\lambda)$ ；
  3. 输出  $(\hat{ek}, (ek_{i,0}, ek_{i,1}), \dots, (ek_{n,0}, ek_{n,1}))$  作为求值公钥，输出  $(\hat{td}, (td_{i,0}, td_{i,1}), \dots, (td_{n,0}, td_{n,1}))$  作为求逆陷门。

- $\text{Eval}(ek, x)$ : 以求值公钥  $ek = \hat{ek} || (ek_{1,0}, ek_{1,1}) \dots (ek_{n,0}, ek_{n,1})$  和原像  $x$  为输入, 计算
  1. 计算  $y_0 \leftarrow f_{\hat{ek}}(x)$ ;
  2. 令  $b_i \leftarrow y_0[i]$ , 对  $i \in [n]$  计算  $y_i \leftarrow f_{ek_{i,b_i}}(x)$ ;
  3. 输出  $y = y_0 || y_1 || \dots || y_n$ .
- $\text{TdInv}(td, y)$ : 以陷门  $td = (\hat{td}, \{(td_{i,0}, td_{i,1})\}_{i \in [n]})$  和像  $y = y_0 || y_1 || \dots || y_n$  为输入:
  1. 计算  $x_0 \leftarrow \mathcal{F}.TdInv(\hat{td}, y_0)$ ;
  2. 令  $b_i \leftarrow y_0[i]$ , 对所有  $i \in [n]$  计算  $x_i \leftarrow \mathcal{F}.TdInv(td_{i,b_i}, y_i)$ ;
  3. 检查  $x_i = x_0$  是否对于  $i \in [n]$  均成立, 若是则输出  $x_0$ , 否则输出  $\perp$ .



$\hat{ek}$	$ek_{1,0}$	$ek_{2,0}$	$ek_{3,0}$
$\hat{td}$	$td_{1,0}$	$td_{2,0}$	$td_{3,0}$
	$ek_{1,1}$	$ek_{2,1}$	$ek_{3,1}$
	$td_{1,1}$	$td_{2,1}$	$td_{3,1}$

图 4.2:  $n = 3$  时的求值公钥和求逆陷门图示图 4.3:  $n = 3, y = 010$  时的求值图示图 4.4:  $n = 3, y = 010$  时的求逆图示

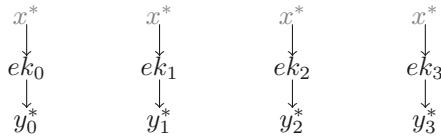
上述 ATDF 构造的正确性显然. 构造中, 函数的像  $y = y_0 || y_1 || \dots || y_n$  是对原像的  $n + 1$  重求值, 其中  $y_0$  确定了使用哪些求值公钥  $ek_{i,b}$  计算  $y_i$ , 因此当底层的 CP-TDF 是单射函数时,  $y_0$  可惟一确定  $y_1, \dots, y_n$ . 下面的定理就是利用上述结构特性模拟求逆预言机  $\mathcal{O}_{\text{inv}}$ .

#### 定理 4.9

如果  $\mathcal{F}$  是一族相对于  $\mathcal{U}_t$  安全的 CP-TDF, 那么上述构造一族 A 自适应单向陷门函数.



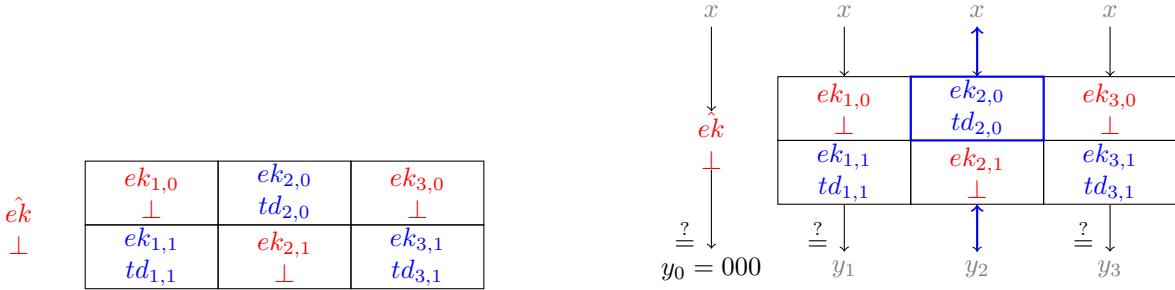
**证明** 使用反证法通过单一游戏完成归约证明. 假设存在 PPT 的对手  $\mathcal{A}$  能以不可忽略的优势打破 ATDF 的自适应单向性, 那么可以黑盒调用  $\mathcal{A}$  的能力构造 PPT 的  $\mathcal{B}$  打破 CP-TDF 相对于  $\mathcal{U}_{n+1}$  的单向性.  $\mathcal{B}$  的 CP-TDF 挑战是公开参数  $pp$ 、求值公钥  $(ek_0, ek_1, \dots, ek_n)$  和像  $y^* = (y_0^*, y_1^*, \dots, y_n^*)$ , 其中  $y_i^* \leftarrow f_{ek_i}(x^*)$ ,  $x^* \xleftarrow{R} X$ .  $\mathcal{B}$  并不知晓  $x^*$ , 其攻击目标是求解  $x^*$ .

图 4.5:  $n = 3$  时  $\mathcal{B}$  的 CP-TDF 挑战实例

令  $b_i^*$  是  $y_0^*$  的第  $i$  比特,  $\mathcal{B}$  (扮演挑战者) 与  $\mathcal{A}$  在 ATDF 的自适应单向性游戏中交互如下:

- 初始化:  $\mathcal{B}$  将 CP-TDF 的  $pp$  设为 ATDF 的公开参数, 设定  $\hat{ek} := ek_0$ , 对  $i \in [n]$  设定  $ek_{i,b_i^*} := ek_i$ , 计算  $(ek_{i,1-b_i^*}, td_{i,1-b_i^*}) \leftarrow \mathcal{F}.KeyGen(\kappa)$ .
- 挑战阶段:  $\mathcal{B}$  发送  $(y_0^*, y_1^*, \dots, y_n^*)$  给  $\mathcal{A}$  作为挑战.
- 求逆询问:  $\mathcal{A}$  向  $\mathcal{B}$  发起求逆询问  $y = (y_0, y_1, \dots, y_n)$ ,  $\mathcal{B}$  分情况应答如下:
  1.  $y_0 = y_0^*$ : 直接返回  $\perp$ , 应答的正确性由以下两种细分情况保证:

1.  $y_0 = y_0^*$ : 直接返回  $\perp$ , 应答的正确性由以下两种细分情况保证:

图 4.6:  $y_0^* = 010$  时生成求值公钥和求逆陷门的过程图示图 4.7:  $y_0 = 000$  时的求逆过程图示

- 对于所有的  $i \in [n]$  均有  $y_i = y_i^*$ : 询问为禁询点, 因此根据  $\mathcal{O}_{\text{inv}}$  的定义需返回  $\perp$ .
- 对于某个  $i \in [n]$  使得  $y_i \neq y_i^*$ :  $\mathcal{F}$  的单射性质和像的生成方式保证了像的首项  $y_0$  确定了其余  $n$  项  $y_1, \dots, y_n$ .
- $y_0 \neq y_0^*$ : 必然存在  $\exists j \in [n]$  s.t.  $b_j \neq b_j^*$  且  $y_j = f_{ek_j, b_j}(x)$ , 其中  $x$  是未知原像. 此时,  $\mathcal{B}$  拥有关于  $ek_{j, b_j}$  的求逆陷门  $td_{j, b_j}$ ,  $\mathcal{B}$  可计算  $x \leftarrow f_{ek_j, b_j}^{-1}(y_j)$ 
  - 如果  $y_0 = f_{ek_0}$  且  $y_i = f_{ek_i, b_i}(x)$  对其余所有  $i \neq j$  也均成立, 那么返回  $x$ , 否则返回  $\perp$ .
- 求解:  $\mathcal{A}$  输出  $x$  作为 ATDF 的挑战应答,  $\mathcal{B}$  将  $x$  转发给 CP-TDF 的挑战者.

容易验证,  $\mathcal{B}$  的优势与  $\mathcal{A}$  的优势相同. 定理得证!  $\square$

#### 注记 4.10 (优化)

以上 ATDF 构造的像  $(y_0, y_1, \dots, y_n)$  包含了对原像的  $(n+1)$  重 CP-TDF 求值:

- $y_0$  构造中起到的作用求值的公钥选择向量, 在归约证明中起到的作用是“全除一”求逆陷门的激活扳机 (trigger), 当  $y_0 \neq y_0^*$  时即可激活求逆陷门.

$y_0$  的编码长度决定了像的冗余重数. 能否缩减  $|y_0|$  以提高效率呢? 答案是肯定的, 可以使用密码组件进行值域扩张 (domain extension) 的通用技术, 使用  $y_0$  的抗碰撞哈希值代替  $y_0$ . 在上述构造中, 我们贴合 ATDF 的安全定义进行更为精细的处理, 使用 TCRHF(target collision resistant hash function) 代替 CRHF. 具体的, 令  $\text{TCR} : \{0, 1\}^n \rightarrow \{0, 1\}^m$ , 使用  $\text{TCR}(y_0)$  代替  $y_0$  作为公钥选择向量和陷门激活扳机. 从而利用 TCR 压缩的性质将像的重数从  $1+n$  缩减到  $1+m$ . 安全论证仍然成立, 这是因为 TCR 的抗碰撞性质保证了在计算意义上:

$$y_0 \neq y_0^* \iff \text{TCR}(y_0) \neq \text{TCR}(y_0^*)$$

类似的优化技术同样可以用于 LTDF+ABO-TDF  $\Rightarrow$  ATDF 的构造中: 可以使用  $y_0$  的 TCR 哈希值代替  $y_0$  作为分支值. 这样处理的好处是增加分支集合选择的灵活性.



**笔记** LTDF+ABO-TDF  $\Rightarrow$  ATDF 与 CP-TDF  $\Rightarrow$  ATDF 的构造分别与 Naor-Yung 范式 [NY-STOC-1990] 和 Dolev-Dwork-Naor 范式 [DDN-STOC-1991] 在思想上极为相似, 总体思路都是通过冗余的结构来保证求逆谕言机的完美模拟.

## 自适应单向陷门关系

将 ATDF 中的确定性函数泛化为可公开高效验证的二元关系可得到自适应单向陷门关系 (ATDR, adaptive trapdoor relation).

- 确定性函数  $\sim$  概率关系
- 可高效计算  $\sim$  可高效采样

**定义 4.8 (单向陷门关系)**

一族单向陷门关系包含以下算法:

- $\text{Setup}(\kappa)$ : 以安全参数  $\kappa$  为输入, 输出公开参数  $pp = (X, Y, EK, TD, R)$ , 其中  $R = \{R_{ek} : X \times Y\}_{ek \in EK}$  是定义在  $X \times Y$  上由  $ek$  索引的一族二元单向关系.
- $\text{KeyGen}(pp)$ : 以公开参数  $pp$  为输入, 输出公钥  $ek$  和陷门  $td$ .
- $\text{Sample}(ek)$ : 输出二元关系的一个随机采样  $(x, y) \xleftarrow{R} R_{ek}$ .
- $\text{TdInv}(td, y)$ : 以  $td$  和  $y \in Y$  为输入, 输出  $x \in X \cup \perp$ .



**正确性:**  $\forall (ek, td) \leftarrow \text{KeyGen}(pp), \forall (x, y) \leftarrow \text{Sample}(ek)$ , 总有  $(\text{TdInv}(td, y), y) \in R_{ek}$ .

我们可以将函数的单射性质平行推广至二元关系的场景下: 如果  $\forall (x_1, y_1), (x_2, y_2) \in R_{ek}$  均有  $x_1 \neq x_2 \Rightarrow y_1 \neq y_2$ , 即  $y$  惟一确定了  $x$ , 那么则称二元关系满足单射性.

 **笔记**  $\text{Sample}$  是概率算法, 因此当  $y_1 \neq y_2$  时, 存在  $x_1 = x_2$  的可能.

**定义 4.9 (自适应单向性)**

令  $R$  是一族二元关系, 定义敌手  $\mathcal{A}$  的优势如下:

$$\Pr \left[ (x', y^*) \in R_{ek} : \begin{array}{l} pp \leftarrow \text{Setup}(\kappa); \\ (ek, td) \leftarrow \text{KeyGen}(pp); \\ (x^*, y^*) \leftarrow \text{Sample}(ek); \\ x' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{inv}}}(ek, y^*); \end{array} \right]$$

其中  $\mathcal{O}_{\text{inv}}$  是求逆谕言机,  $\forall x \neq x^*, \mathcal{O}_{\text{inv}}(y) = \text{TdInv}(td, y)$ . 如果任意 PPT 敌手  $\mathcal{A}$  在上述安全试验中的优势均为  $\text{negl}(\kappa)$ , 那么则称  $R$  是自适应单向的.



ATDR 是 ATDF 的弱化, 弱化允许我们可以给出更加高效灵活的设计, 同时不严重降低可用性. 在给出 ATDR 的构造之前, 我们首先回顾基于 CP-TDF 的 ATDF 构造. 构造的关键之处是将像  $y$  设计为  $y_0$  和  $(y_1, \dots, y_n)$  两部分, 其中  $y_0$  设定为  $f_{ek}(x)$ , 通过单射性完美绑定了  $(y_1, \dots, y_n)$ , 同时在归约证明中起到了“全除一”陷门触发器的作用: 当目标不再是构造确定性单向函数而是概率二元关系时, 我们有着更加灵活的选择: 使用一次性签名 (OT, one-time signature) 的验证公钥作为  $(y_1, \dots, y_n)$  的求值选择器和求逆陷门触发器.

**构造 4.8 (基于 CP-TDF 和 OTS 的 ATDR 构造)**

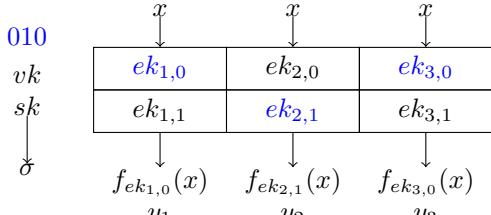
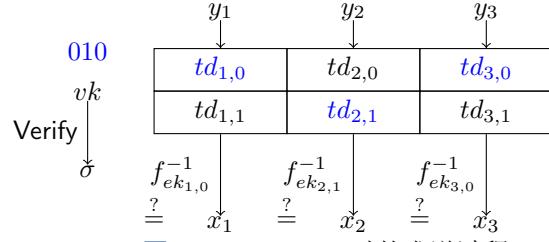
构造组件: 单射 CP-TDF  $\mathcal{F} : X \rightarrow Y$  和 strong OTS (令  $|vk| = \{0, 1\}^n$ , 签名空间为  $\Sigma$ );

构造目标: ATDR  $X \rightarrow VK \times Y^n \times \Sigma$

- $\text{Setup}(1^\kappa)$ : 运行  $pp_{\text{cptdf}} \leftarrow \mathcal{F}.Setup(1^\kappa)$ ,  $pp_{\text{ots}} \leftarrow \text{OTS}.Setup(1^\kappa)$ , 输出  $pp = (pp_{\text{cptdf}}, pp_{\text{ots}})$ .
- $\text{KeyGen}(pp)$ : 以  $pp = (pp_{\text{cptdf}}, pp_{\text{ots}})$  为输入, 对  $b \in \{0, 1\}$  和  $i \in [n]$  运行  $(ek_{i,b}, td_{i,b}) \leftarrow \mathcal{F}.KeyGen(pp_{\text{cptdf}})$  输出  $ek = ((ek_{i,0}, ek_{i,1}), \dots, (ek_{n,0}, ek_{n,1}))$ ,  $td = ((td_{i,0}, td_{i,1}), \dots, (td_{n,0}, td_{n,1}))$ .
- $\text{Sample}(ek)$ : 以  $ek = (ek_{1,0}, ek_{1,1}) \dots (ek_{n,0}, ek_{n,1})$  为输入, 采样如下:
  1. 生成  $(vk, sk) \leftarrow \text{OTS.KeyGen}(pp_{\text{ots}})$ ;
  2. 随机选择  $x \in X$ , 对  $i \in [n]$  计算  $y_i \leftarrow f_{ek_{i,b_i}}(x)$ , 其中  $b_i \leftarrow vk[i]$ ;
  3. 计算  $\sigma \leftarrow \text{OTS.Sign}(sk, y_1 || \dots || y_n)$ ;
 输出  $y = (vk, y_1 || \dots || y_n, \sigma)$ .
- $\text{TdInv}(td, y)$ : 以  $td = (\{(td_{i,0}, td_{i,1})\}_{i \in [n]})$  和  $y = (vk, y_1 || \dots || y_n, \sigma)$  为输入, 求逆如下:
  1. 检查  $\text{OTS.Verify}(vk, y_1 || \dots || y_n, \sigma) \stackrel{?}{=} 1$ , 如果签名无效则返回  $\perp$ ;
  2. 对所有  $i \in [n]$  计算  $x_i \leftarrow \mathcal{F}.TdInv(td_{i,b_i}, y_i)$ , 其中  $b_i = vk[i]$ .
  3. 如果对所有  $i \in [n]$  均有  $x_i = x_1$  则返回  $x_1$ , 否则返回  $\perp$ .



$ vk  = 3$	$ek_{1,0}$	$ek_{2,0}$	$ek_{3,0}$
	$td_{1,0}$	$td_{2,0}$	$td_{3,0}$
	$ek_{1,1}$	$ek_{2,1}$	$ek_{3,1}$
	$td_{1,1}$	$td_{2,1}$	$td_{3,1}$

图 4.8:  $|vk| = 3$  时的求值公钥和求逆陷门生成图示图 4.9:  $vk = 010$  时的采样过程图 4.10:  $vk = 010$  时的求逆过程

构造的正确性显然, 构造的以下三个特性使得归约算法能够成功模拟  $\mathcal{O}_{\text{inv}}$ .

- $R_{ek}$  是单射的并且  $y_1 \parallel \dots \parallel y_n$  是对原像  $x$  的  $n$  重冗余求值.
- $vk$  是求值公钥的选择比特向量.
- 利用 OTS 的 sEUF-CMA 安全性,  $vk$  在计算意义下绑定了  $(y_1, \dots, y_n)$ .

#### 定理 4.10

如果 OTS 是 sEUF-CMA 安全的, 并且  $\mathcal{F}$  是  $\mathcal{U}_n$  相关积单向的, 那么上述二元关系的构造满足自适应单向性.



**证明** 证明通过以下游戏序列完成.

Game<sub>0</sub>: 对应真实的 ATDR 自适应单向性安全试验. 令  $y^* = (vk^*, y_1^* \parallel \dots \parallel y_n^*, \sigma^*)$  是挑战的像.

Game<sub>1</sub>: 与 Game<sub>0</sub> 相同, 唯一的区别是挑战者对于求逆询问  $y = (vk^*, y_1 \parallel \dots \parallel y_n, \sigma)$  直接返回  $\perp$ . 应答的合理性分情况解释如下:

1.  $(y_1 \parallel \dots \parallel y_v, \sigma) = (y_1^* \parallel \dots \parallel y_v^*, \sigma^*)$ : 禁查询点
2.  $(y_1 \parallel \dots \parallel y_v, \sigma) \neq (y_1^* \parallel \dots \parallel y_v^*, \sigma^*)$ : 构成 OTS 的存在性伪造

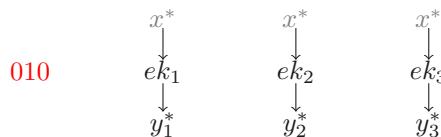
记敌手发起第二种类型求逆询问的事件为  $F$ , 那么利用 Difference Lemma 可以证明  $|\Pr[S_1] - \Pr[S_0]| \leq \Pr[F]$ , 而基于 OTS 的 sEUF-CMA 安全性, 可以推出  $\Pr[F] \leq \text{negl}(\kappa)$ , 从而  $|\Pr[S_1] - \Pr[S_0]| \leq \text{negl}(\kappa)$ .

#### 断言 4.6

如果  $\mathcal{F}$  是  $\mathcal{U}_t$  相关积安全的, 那么对于任意的 PPT 敌手均有  $\Pr[S_1] = \text{negl}(\kappa)$ .



**证明** 论证通过单一归约完成. 假设存在 PPT 的敌手  $\mathcal{A}$  在 Game<sub>1</sub> 中的优势不可忽略, 那么尝试构造 PPT 算法  $\mathcal{B}$ , 通过黑盒调用  $\mathcal{A}$  的能力打破 CP-TDF 相对  $\mathcal{U}_n$  的相关积单向性.  $\mathcal{B}$  的 CP-TDF 挑战是公开参数  $pp_{\text{cptdf}}$ , 求值公钥  $(ek_1, \dots, ek_n)$  和像  $(y_1^*, \dots, y_n^*)$ , 其中  $y_i^* \leftarrow f_{ek_i}(x^*)$ ,  $x^* \xleftarrow{R} X$ .  $\mathcal{B}$  并不知晓  $x^*$ , 其攻击目标是求解  $x^*$ .

图 4.11:  $n = 3$  时  $\mathcal{B}$  的 CP-TDF 挑战实例

$\mathcal{B}$ (扮演挑战者) 与  $\mathcal{A}$  在 Game<sub>1</sub> 中交互如下:

- 初始化:  $\mathcal{B}$  运行  $pp_{\text{ots}} \leftarrow \text{OTS}.\text{Setup}(1^\kappa)$ , 生成  $(vk^*, sk^*) \leftarrow \text{OTS}.\text{KeyGen}(pp_{\text{ots}})$ . 令  $b_i^*$  是  $vk^*$  的第  $i$  比特,  $\mathcal{B}$  进行如下操作:

1. 对  $i \in [n]$  设定  $\text{ek}_{i,b_i^*} := ek_i$ .
  2. 对  $i \in [v]$  计算  $(\text{ek}_{i,1-b_i^*}, td_{i,1-b_i^*}) \leftarrow \mathcal{F}.\text{KeyGen}(pp_{\text{cptdf}})$ .
- $\mathcal{B}$  发送  $pp = (pp_{\text{cptdf}}, pp_{\text{ots}})$  和  $ek = (ek_{1,0}, ek_{1,1}, \dots, ek_{n,0}, ek_{n,1})$  给  $\mathcal{A}$ .

$vk^*$	$ek_{1,0}$ $\perp$	$ek_{2,0}$ $td_{2,0}$	$ek_{3,0}$ $\perp$
$sk^*$	$ek_{1,1}$ $td_{1,1}$	$ek_{2,1}$ $\perp$	$ek_{3,1}$ $td_{3,1}$

图 4.12:  $|vk| = 010$  时归约算法设定求值公钥和求逆陷门的过程图示

- 挑战:  $\mathcal{B}$  计算  $\sigma^* \leftarrow \text{OTS.Sign}(sk^*, (y_1^*, \dots, y_n^*))$ , 发送  $(vk^*, y_1^*, \dots, y_n^*, \sigma^*)$  给  $\mathcal{A}$  作为挑战.
  - 求逆询问: 对于求逆询问  $y = (vk, y_1 || \dots || y_v, \sigma)$ ,  $\mathcal{B}$  应答如下:
    1.  $vk = vk^*$ : 直接返回  $\perp$ .
    2.  $vk \neq vk^*$ : 必然存在  $\exists j \in [n]$  s.t.  $b_j \neq b_j^*$  且  $y_j = f_{ek_j, b_j}(x)$ , 其中  $x$  是未知原像. 此时,  $\mathcal{B}$  拥有关于  $ek_{j, b_j}$  的求逆陷门  $td_{j, b_j}$ ,  $\mathcal{B}$  可计算  $x \leftarrow f_{ek_j, b_j}^{-1}(y_j)$ 
      - 如果  $y_i = f_{ek_i, b_i}(x)$  对所有的  $i \neq j$  也均成立, 那么返回  $x$ , 否则返回  $\perp$ .
- 由  $\mathcal{F}$  的单射性可知,  $\mathcal{B}$  完美的模拟了 Game<sub>1</sub> 中的  $\mathcal{O}_{\text{inv}}$  应答.

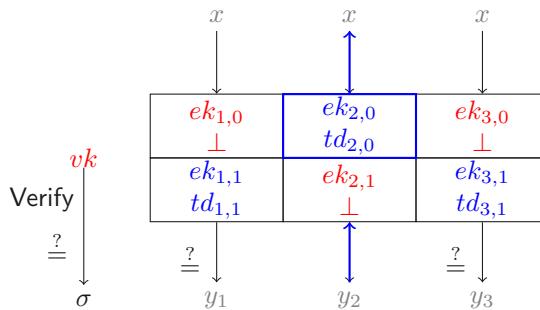


图 4.13:  $vk = 010$  时归约算法求逆过程图示

- 求解:  $\mathcal{A}$  输出  $x$  作为 Game<sub>1</sub> 中 ATDF 的挑战应答,  $\mathcal{B}$  将  $x$  转发给 CP-TDF 的挑战者.
- 容易验证,  $\mathcal{B}$  的优势与  $\mathcal{A}$  的优势相同. 断言得证. □
- 综上, 定理得证! □

## 小结

Rosen 和 Regev [**RS-TCC-2009**] 证明了 CP-TDF 与 LTDF 之间存在黑盒分离, Kiltz、Mohassel 和 O'Neill [**KMO-EUROCRYPT**] 证明了 ATDF 与 CP-TDF 之间也存在黑盒分离. 因此, 在黑盒的意义下, ATDF 和 ATDR 是目前单向函数类中构造 CCA-KEM 所需的最弱组件.

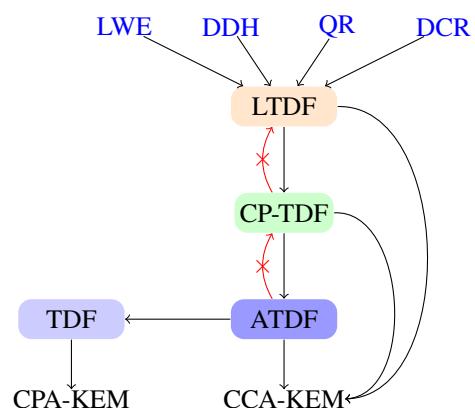


图 4.14: 各类单向函数之间的蕴含关系

## 4.2 哈希证明系统类

一阴一阳之谓道, 继之者善也, 成之者性也.

—《易经·系辞上》

1998 年, Cramer 和 Shoup [CS-CRYPTO-1998] 基于判定性 Diffie-Hellman 问题构造出首个标准模型下高效的公钥加密方案, 成为 CS98-PKE. 2002 年, Cramer 和 Shoup [CS-EUROCRYPT-2002] 再度合作, 提出了哈希证明系统 (HPS, hash proof system) 的概念, 给出了标准模型下构造 CCA-secure PKE 的全新范式, 完美的阐释了 CS98-PKE 的设计原理. 在同一篇论文中, 作者还正式提出了 KEM+DEM 的公钥加密工作模式, 相比朴素混合加密更加现代、模块化. 以下首先介绍 HPS 的定义和相关性质.

### 定义 4.10 (哈希证明系统)

HPS 包含以下多项式时间算法:

- $\text{Setup}(1^\kappa)$ : 以安全参数  $\kappa$  为输入, 输出公开参数  $pp = (\mathsf{H}, SK, PK, X, L, W, \Pi, \alpha)$ , 其中  $\mathsf{H} : SK \times X \rightarrow \Pi$  是由私钥集合  $SK$  索引的一族带密钥哈希函数 (keyed hash function),  $L$  是定义在  $X$  上的  $\mathcal{NP}$  语言,  $W$  是对应的证据集合,  $\alpha$  是从私钥集合  $SK$  到公钥集合  $PK$  的投射函数.
- $\text{KeyGen}(pp)$ : 以公开参数  $pp$  为输入, 随机采样  $sk \xleftarrow{\text{R}} SK$ , 计算  $pk \leftarrow \alpha(sk)$ , 输出  $(pk, sk)$ .
- $\text{PrivEval}(sk, x)$ : 以私钥  $sk$  和  $x \in X$  为输入, 输出  $\pi = \mathsf{H}_{sk}(x)$ .
- $\text{PubEval}(pk, x, w)$ : 以公钥  $pk$ 、 $x \in L$  以及相应的  $w$  为输入, 输出  $\pi = \mathsf{H}_{sk}(x)$ , 其中  $\alpha(sk) = pk$ .



$$pp \leftarrow \text{Setup}(1^\kappa)$$

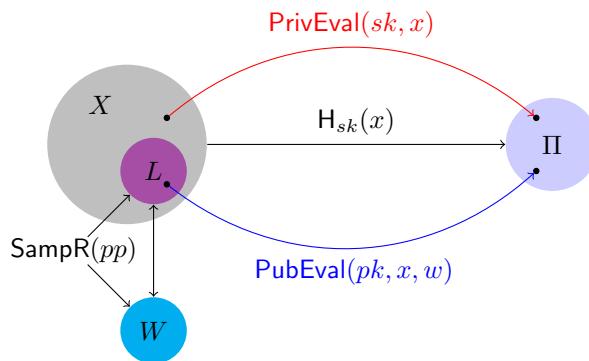


图 4.15: HPS 示意图

HPS 的定义围绕  $L \subset X$  展开, 引入了 KeyGen, PrivEval 和 PubEval 这三个核心算法. 以下性质刻画了哈希函数在输入  $x \in L$  上的行为, 用于保证上层密码方案的功能性.

**投射性 (Projectivity):**  $\forall x \in L$ , 函数值  $\mathsf{H}_{sk}(x)$  由  $x$  和私钥的投射  $pk \leftarrow \alpha(sk)$  完全确定.

以下性质由弱到强刻画了哈希函数在输入  $x \in X \setminus L$  上的行为, 用于保证上层密码方案的安全性.

**平滑性 (Smooth):**  $\mathsf{H}_{sk}(\cdot)$  在输入  $x \xleftarrow{\text{R}} X \setminus L$  时的输出与  $\Pi$  上的均匀分布统计接近, 即:

$$(pk, \mathsf{H}_{sk}(x)) \approx_s (pk, \pi)$$

其中  $(pk, sk) \leftarrow \text{KeyGen}(pp)$ ,  $\pi \xleftarrow{R} \Pi$ .

**1-一致性 (1-Universal):**  $H_{sk}(\cdot)$  在任意输入的输出与  $\Pi$  上的均匀分布统计接近, 即  $\forall x \in X \setminus L$ , 有:

$$(pk, H_{sk}(x)) \approx_s (pk, \pi)$$

其中  $(pk, sk) \leftarrow \text{KeyGen}(pp)$ ,  $\pi \xleftarrow{R} \Pi$ .

**2-一致性 (2-Universal):** 在给定某点  $x^* \in X \setminus L$  哈希函数值的情形下,  $H_{sk}(\cdot)$  在任意输入的输出仍与  $\Pi$  上的均匀分布统计接近, 即  $\forall x, x^* \in X \setminus L$  且  $x \neq x^*$ , 有:

$$(pk, H_{sk}(x^*), H_{sk}(x)) \approx_s (pk, H_{sk}(x^*), \pi)$$

其中  $(pk, sk) \leftarrow \text{KeyGen}(pp)$ ,  $\pi \xleftarrow{R} \Pi$ .

**笔记** 以上三条性质由弱到强. smooth 性质同时建立在  $sk \xleftarrow{R} SK$  和  $x \xleftarrow{R} X \setminus L$  两根随机带上, 1-universal 性质仅建立在  $sk \xleftarrow{R} SK$  一根随机带上, 而 2-universal 性质则可解读为要求 1-universal 性质在随机带  $sk \xleftarrow{R} SK$  有偏时(将  $H_{sk}(x^*)$  理解为关于  $sk$  的泄漏)仍然成立. 特别注意, 三条性质均刻画的是输入在语言外时哈希函数的行为.

### 4.2.1 起源释疑

相信很多读者在阅读 HPS 早期的文献时, 都会对这个范式的命名和引入动机感到疑惑. 事实上, HPS 是一类指定验证者的非交互式零知识证明系统 (designated verifier NIZK), 引入的动机来自以下的思考: Naor-Yung 双重加密范式使用标准的 NIZK 来证明密文的合法性 (well-formedness), 然而密文的合法性并非一定是可公开验证的 (public verifiable), 解密私钥  $sk$  的持有者可验证即可. 指定可验证弱于公开可验证, 因此 DV-NIZK 的效率通常高于 NIZK. 想必 Cramer 和 Shoup 正是基于以上的思考, 引入了 HPS, 目的是在标准模型下构造高效的 CCA-secure PKE.

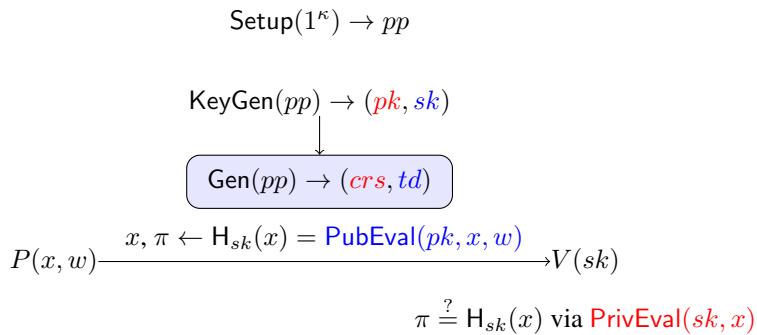


图 4.16: 从 DV-NIZK 的视角解析 HPS

**笔记** 上图解释了 HPS 的命名渊源, 其本质上是指定验证者零知识证明, 证明的形式是实例的哈希值, 故名哈希证明系统.

- DV-NIZK 的完备性由  $H_{sk}(\cdot)$  的投射性保证:

$$\forall x \in L, H_{sk}(x) = \text{PubEval}(pk, x, w)$$

- DV-NIZK 的合理性由 1-universal 性质保证  $\forall x \notin L, H_{sk}(x)$  随机分布, 即使拥有无限计算能力的证明者  $P^*$  也无法预测, 因此通过验证的概率可忽略. 2-universal 性质则保证了更强的合理性, 即敌手在看到一个 No 实例的有效证明后, 也无法为一个新的 No 实例生成有效证明.
- DV-NIZK 的零知识性是显然且平凡的: 指定验证者拥有私钥, 因此可以对任意的  $x \in L$ (甚至是  $x \in X$ ) 生成正确的证明

此外, 证明系统是有效的, 即证明者在拥有证据时可以高效计算出实例的证明, 这对于基于 HPS 密码方案的功能性至关重要.

### 4.2.2 HPS 的构造

我们首先以针对  $L_{\text{DDH}}$  语言的 HPS 构造为例, 获得对 HPS 设计方式的直观认识. 令  $(\mathbb{G}, p, g)$  是算法  $\text{GroupGen}(1^\kappa)$  的输出, 其中  $\mathbb{G}$  是阶为素数  $p$  的群,  $g$  是生成元. 随机选取  $\mathbb{G}$  中的两个生成元  $g_1, g_2$ . 令  $pp = (\mathbb{G}, p, g_1, g_2)$  是公开参数, 定义由  $pp$  索引的  $\mathcal{NP}$  语言如下:

$$L_{\text{DDH}} = \{(x_1, x_2) \in X : \exists w \in W \text{ s.t. } x_1 = g_1^w, x_2 = g_2^w\}$$

其中  $X = \mathbb{G} \times \mathbb{G}$ ,  $W = \mathbb{Z}_p$ .

容易验证, 语言中的元素是 DH 对, 语言外的元素是非 DH 对,  $(x_1, x_2) \xleftarrow{\text{R}} L_{\text{DDH}}$ . DDH 假设蕴含  $L \subset X$  上的 SMP 困难问题成立, 即:

- $U_L \approx_c U_X$ : 随机 DH 对与  $X$  中的随机二元组计算不可区分
- 由于  $|L|/|X| = 1/p = \text{negl}(\kappa)$ ,  $L$  在  $X$  中稀疏, 所以可以进一步得到  $U_L \approx_c U_{X \setminus L}$ : 随机 DH 对与随机非 DH 对计算不可区分

#### 构造 4.9 ( $L_{\text{DDH}}$ 语言的 HPS 构造)

$L_{\text{DDH}}$  的 HPS 构造如下, 如图 ?? 所示:

- $\text{Setup}(1^\kappa)$ : 以安全参数  $\kappa$  为输入, 输出公开参数  $pp = (\mathbb{G}, p, g_1, g_2)$ .  $pp$  还包括了对  $SK = \mathbb{Z}_p \times \mathbb{Z}_p$ ,  $PK = \mathbb{G}$ ,  $L_{\text{DDH}}$ ,  $X = \mathbb{G} \times \mathbb{G}$  和  $W = \mathbb{Z}_p$  的描述.
- $\text{KeyGen}(pp)$ : 以公开参数  $pp$  为输入, 随机采样  $sk \xleftarrow{\text{R}} \mathbb{Z}_p^2$ , 计算  $pk \leftarrow \alpha(sk) = g_1^{sk_1} g_2^{sk_2}$ , 输出  $(pk, sk)$ .
- $\text{PrivEval}(sk, x)$ : 以私钥  $sk$  和  $x \in X$  为输入, 输出  $\pi = \mathsf{H}_{sk}(x) = x_1^{sk_1} x_2^{sk_2}$ .
- $\text{PubEval}(pk, x, w)$ : 以公钥  $pk$ 、 $x \in L_{\text{DDH}}$  以及相应的  $w$  为输入, 输出  $\pi = pk^w$ , 其中  $\alpha(sk) = pk$ . 以下等式说明了公开求值算法的正确性:

$$pk^w = (g_1^{sk_1} g_2^{sk_2})^w = x_1^{sk_1} x_2^{sk_2} = \mathsf{H}_{sk}(x)$$

#### 引理 4.2

以上关于  $L_{\text{DDH}}$  的 HPS 满足 1-universal 性质.



**证明** 证明的目标是

$$\forall x \in X \setminus L, (pk, \mathsf{H}_{sk}(x)) \approx_s (pk, \pi)$$

其中  $(pk, sk) \leftarrow \text{KeyGen}(pp)$ ,  $\pi \xleftarrow{\text{R}} \Pi$ .

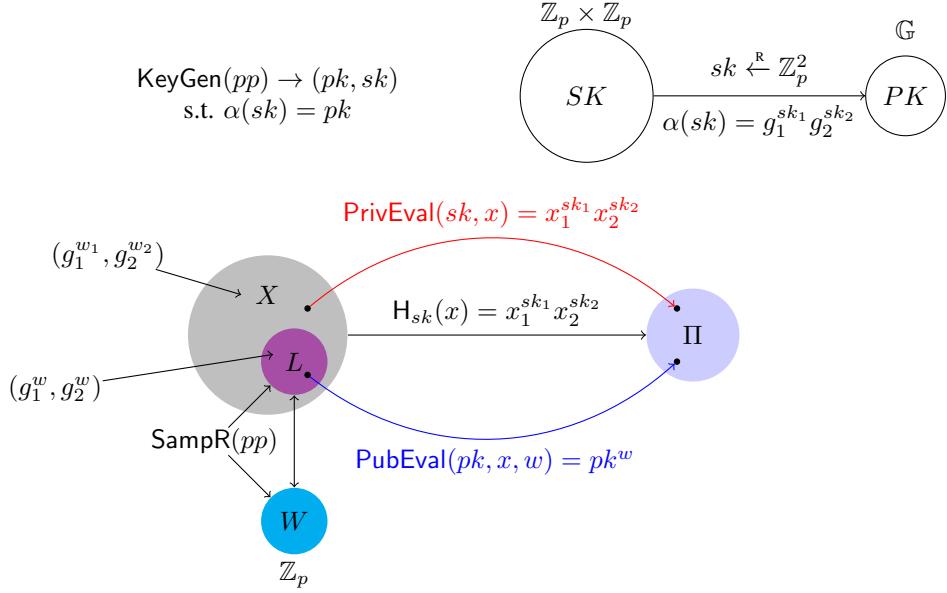
首先固定  $x = (x_1 = g_1^{w_1}, x_2 = g_2^{w_2}) \in X \setminus L$ , 其中  $w_1 \neq w_2$ . 将左式表示为关于  $sk$  函数的形式:

$$(pk, \mathsf{H}_{sk}(x)) = f_{g_1, g_2, x_1, x_2}(sk_1, sk_2) := (g_1^{sk_1} g_2^{sk_2}, x_1^{sk_1} x_2^{sk_2})$$

用矩阵的形式描述函数作用过程:

$$\begin{pmatrix} g_1 & g_2 \\ g_1^{w_1} & g_2^{w_2} \end{pmatrix} \begin{pmatrix} sk_1 \\ sk_2 \end{pmatrix} = \begin{pmatrix} pk \\ \mathsf{H}_{sk}(x) \end{pmatrix}$$

$$pp = (\mathbb{G}, p, g_1, g_2) \leftarrow \text{Setup}(1^\kappa)$$

图 4.17:  $L_{\text{DDH}}$  的 HPS

令  $g_2 = g_1^\beta$ , 其中  $\beta \in \mathbb{Z}_p$ , 将最左边矩阵进行等价变形:

$$\begin{pmatrix} g_1 & g_2 \\ g_1^{w_1} & g_2^{w_2} \end{pmatrix} = \begin{pmatrix} g_1 & g_1^\beta \\ g_1^{w_1} & g_1^{w_2\beta} \end{pmatrix} = g_1 \underbrace{\begin{pmatrix} 1 & \beta \\ w_1 & w_2\beta \end{pmatrix}}_M$$

$\det(M) = \beta(w_1 - w_2) \Rightarrow M$  满秩  $\Rightarrow f$  单射. 又由于函数的定义域和值域大小相等, 最终得出:

$$\underbrace{\begin{pmatrix} g_1 & g_2 \\ g_1^{w_1} & g_2^{w_2} \end{pmatrix}}_{\text{full rank } 2 \times 2} \underbrace{\begin{pmatrix} sk_1 \\ sk_2 \end{pmatrix}}_{\text{uniform over } \mathbb{Z}_p^2} = \underbrace{\begin{pmatrix} pk \\ \text{H}_{sk}(x) \end{pmatrix}}_{\text{uniform over } \mathbb{G}^2}$$

从而 1-universal 性质得证!

#### 注记 4.11

HPS 并不一定要求  $L \subseteq X$  之上一定存在 SMP 问题, 但只有当  $L \subseteq X$  之上存在 SMP 问题时, 相应的 HPS 有密码学意义. 这是因为 HPS 中所有关于哈希函数的性质均是针对输入在语言外时定义的, 只有当 SMP 问题存在时, 才可以间接刻画出哈希函数在输入在语言内时的行为.

HPS 存在两个局限:

- 证明只支持私密验证, 不满足公开验证性
- 证明的表达能力有限, 目前仅能对证明群中的子群成员归属问题, 尚未知能否延伸到任意的 NP 语言.

在很多具体的零知识证明应用场合, 公开验证性和强大的表达能力均不是必须, 因此用标准的零知识证明系统有大材小用之嫌, 哈希证明系统可以做的更快更好, 其中效率的优势恰恰源自局限. 以下展示如何基于 HPS 设计 IND-CPA 和 IND-CCA 的 KEM 方案.

### 4.2.3 基于 HPS 构造 IND-CPA KEM

作为暖场应用, 我们首先介绍如何基于 HPS 构造 CPA 安全的 KEM. 设计的思路如下:

- 发送方扮演 HPS 中的证明者, 选择  $L$  中的随机实例  $x$  作为密文  $c$ , 利用公钥  $pk$  和相应的证据  $w$  计算其哈希证明  $\pi$  作为会话密钥  $k$ .
- 接收方扮演 HPS 中的验证者, 使用私钥  $sk$  计算  $x$  的哈希证明以恢复会话密钥  $k$ .

#### 构造 4.10 (基于 HPS 的 CPA 安全 KEM)

从 smooth 的 HPS 出发, 构造 CPA 安全的 KEM 如下:

- $\text{Setup}(\kappa)$ : 运行  $pp \leftarrow \text{HPS}.\text{Setup}(1^\kappa)$ , 输出  $pp = (\mathsf{H}, SK, PK, X, L, W, \Pi, \alpha)$  作为公开参数, 其中  $X$  作为密文空间,  $\Pi$  作为会话密钥空间.
- $\text{KeyGen}(pp)$ : 运行  $(pk, sk) \leftarrow \text{HPS}.\text{KeyGen}(pp)$ , 输出公钥  $pk$  和私钥  $sk$ .
- $\text{Encaps}(pk; r)$ : 以公钥  $pk$  和随机数  $r$  为输入, 执行如下步骤:
  1. 运行  $(x, w) \leftarrow \text{SampR}(r)$  生成随机实例和相应的证据;
  2. 通过  $\text{HPS}.\text{PubEval}(pk, x, w)$  计算实例  $x$  的哈希证明  $\pi \leftarrow \mathsf{H}_{sk}(x)$ ;
  3. 输出实例  $x$  作为密文  $c$ , 输出哈希证明  $\pi$  作为会话密钥  $k$ .
- $\text{Decaps}(sk, c)$ : 以私钥  $sk$  和密文  $c$  为输入, 通过  $\text{HPS}.\text{PrivEval}(sk, c)$  计算  $c$  的哈希证明  $\pi \leftarrow \mathsf{H}_{sk}(x)$  以恢复会话密钥  $k$ .

KEM 方案的正确性有 HPS 的完备性保证. 安全性由如下定理保证.

#### 定理 4.11

如果  $L \subseteq X$  上的 SMP 困难问题成立, 那么构造 ?? 中的 KEM 是 IND-CPA 安全的.



**证明** 我们将通过游戏序列组织证明. 游戏序列的编排次序由如下证明思路指引:

- 将诚实生成的密文分布  $x \xleftarrow{\text{R}} L$  切换为  $x \xleftarrow{\text{R}} X \setminus L$
- 论证当  $x \xleftarrow{\text{R}} X \setminus L$  时,  $(pk, \pi = \mathsf{H}_{sk}(x))$  的分布与  $(pk, \pi \xleftarrow{\text{R}} \Pi)$  统计接近.

$\text{Game}_0$ : 对应真实的游戏, 其中挑战密文  $x^* \xleftarrow{\text{R}} L$ , 计算会话密钥的方式是对  $\mathsf{H}_{sk}(x^*)$  进行公开求值

- 初始化:  $\mathcal{CH}$  计算  $pp \leftarrow \text{HPS}.\text{Setup}(1^\kappa)$ ,  $(pk, sk) \leftarrow \text{HPS}.\text{KeyGen}(pp)$ , 将  $pp$  和  $pk$  发送给  $\mathcal{A}$ .
- 挑战:  $\mathcal{CH}$  按照以下步骤生成挑战
  1. 随机采样  $(x^*, w^*) \leftarrow \text{SampR}(r^*)$ ;
  2. 通过  $\text{HPS}.\text{PubEval}(pk, x^*, r^*)$  公开计算  $\pi^* \leftarrow \mathsf{H}_{sk}(x^*)$ ;
  3. 令  $c^* = x^*$ ,  $k_0^* = \pi^*$ , 随机采样  $k_1^* \xleftarrow{\text{R}} \Pi$ ;
  4. 随机选取  $\beta \xleftarrow{\text{R}} \{0, 1\}$ , 将  $(c^*, k_\beta^*)$  发送给  $\mathcal{A}$  作为挑战.

敌手  $\mathcal{A}$  在游戏中的视图包括  $(pp, pk, x^*, k_\beta^*)$ .

- 应答:  $\mathcal{A}$  输出对  $\beta$  的猜测  $\beta'$ ,  $\mathcal{A}$  成功当且仅当  $\beta' = \beta$ .

为了准备将挑战密文的分布从  $x^* \in L$  切换到  $x^* \in X \setminus L$ , 我们首先需要引入以下的游戏作为过渡, 这是因为分布切换后  $x^*$  已经不在语言  $L$  内,  $\mathcal{CH}$  无法再以公开求值的方式计算哈希证明, 所以需要提前改变  $\mathcal{CH}$  的求值方式.

$\text{Game}_1$ : 与  $\text{Game}_0$  相比唯一的区别在于挑战阶段的步骤 2,  $\mathcal{CH}$  通过  $\text{HPS}.\text{PrivEval}(sk, x^*)$  秘密计算  $\pi^* \leftarrow \mathsf{H}_{sk}(x^*)$ .  $\mathsf{H}_{sk}(\cdot)$  的投射性质保证了当  $x^* \in L$  时,  $\text{PubEval}(pk, x^*, w^*) = \mathsf{H}_{sk}(x^*) = \text{PrivEval}(sk, x^*)$  因此在敌手的视角中,  $\mathcal{CH}$  所作出的改变完全不可察觉, 我们有:

$$\text{Game}_0 \equiv \text{Game}_1$$

经过  $\text{Game}_1$  的铺垫, 我们可以顺利过渡到以下的  $\text{Game}_2$ .

$\text{Game}_2$ : 与  $\text{Game}_1$  唯一的区别是调用  $\text{SampNo}(r^*)$  采样  $x^* \xleftarrow{\text{R}} X \setminus L$ . SMP 问题的困难性保证了敌手在相邻游戏中

的视图计算不可区分:

$$\text{Game}_1 \approx_c \text{Game}_2$$

$\text{Game}_3$ : 与  $\text{Game}_2$  的唯一不同是在挑战阶段随机采样  $\pi^* \xleftarrow{R} \Pi$  替代  $\pi^* \leftarrow H_{sk}(x^*)$ . 由  $H_{sk}(\cdot)$  的平滑性保证:

$$\text{Game}_2 \approx_s \text{Game}_3$$

在  $\text{Game}_3$  中,  $k_0^*$  和  $k_1^*$  均是  $\Pi$  上的均匀分布, 因此即使对于拥有无穷计算能力的敌手  $\mathcal{A}$ , 其优势也为 0. 综合以上, 定理得证!  $\square$

#### 4.2.4 基于 HPS 构造 IND-CCA KEM

我们首先以自问自答的方式分析基于 HPS 构造 CCA-secure KEM 的难点.

构造 ?? 中的 KEM 方案是 IND-CCA 安全的么?

- 从归约证明的角度粗略分析似乎并没有技术困难, 因为归约算法  $\mathcal{R}$  始终掌握私钥  $sk$ , 可以回答任意的解封装询问. 然而细致分析后发现并非如此. 与 IND-CPA 安全游戏相比, 在 IND-CCA 安全游戏中敌手的视图中额外包括了对解封装询问的应答. 当解封装询问  $c = x$  的密文  $x \notin L$  时, 应答会泄漏更多关于  $sk$  的信息 (公钥  $pk$  可以看做关于  $sk$  的部分泄漏). 因此我们无法再使用平滑性得出  $\text{Game}_2 \approx_s \text{Game}_3$  的结论.

接上问, 既然当  $x \in X \setminus L$  时的解封装询问会泄漏  $sk$  的信息, 那拒绝此类询问是否可以达到 IND-CCA 安全性呢?

- 不可以. 这是因为 SMP 问题的困难性使得 PPT 的解密者无法判定是否  $x \in L$ . 善于思考的读者很能发现解密者还拥有解密私钥  $sk$ , 然而解密者 (对应诚实用户) 仅拥有一个解密私钥, 依然无法判定是否  $x \in L$ . 那是否有巧妙的方案设计使得解密者拥有多个解密私钥, 从而解密者可以通过检测多个私钥求值的一致性来判定  $x \in L$  了. 答案依然是否定的, 因为 SMP 的困难性否定了此类方案设计算法的存在性. 反过来, 如果解密者拥有了对应 SMP 问题公开参数对应的秘密参数, 那么确实可以设计方案使得解密者拥有多个解密私钥, 比如考虑  $L_{DDH}$  语言的 HPS ??, 如果解密者知晓  $\alpha$  使得  $g_1^\alpha = g_2$ , 那么任取  $\Delta \in \mathbb{Z}_p$ , 均有:

$$(sk_1, sk_2) \sim (sk'_1 = sk_1 + \alpha\Delta, sk'_2 = sk_2 - \Delta) \Leftrightarrow g_1^{sk_1} g_2^{sk_2} = g_1^{sk'_1} g_2^{sk'_2}$$

上述设计方案已经暗含了 SMP 问题的困难性对解密者不复存在, 这使得安全归约将会在  $\text{Game}_1 \approx_c \text{Game}_2$  的步骤失败, 原因是归约算法 (针对 SMP 问题的敌手) 不掌握  $\alpha$ , 从而无法模拟解密者的行为.

通过以上的分析, 不难得出基于 HPS 构造 CCA-secure KEM 的一种思路是杜绝“危险”的解密询问:

- $x \in L$  属于安全的解密询问, 这是因为应答  $\pi = \text{HPS.PubEval}(pk, x, w)$  没有额外泄漏关于  $sk$  的信息, 因此不会破坏平滑性.
- $x \notin L$  属于危险的解密询问, 杜绝的思路在密文中嵌入某种 **私密认证结构**, 使得 PPT 的敌手无法生成有效的 (valid) 危险密文, 同时解密者能够判定密文是否有效. 具体的设计思路是将哈希证明作为信息论意义下的一次性消息验证码 (information-theoretic one-time MAC), 此处需要满足 2-universal 性质的 HPS.

#### 构造 4.11 (基于 HPS 的 CCA-secure KEM)

构造的组件是:

- 满足 smooth 性质的  $\text{HPS}_1$
- 满足 2-universal 性质的  $\text{HPS}_2$
- $\text{Setup}(1^\kappa)$ :
  - 运行  $pp_1 \leftarrow \text{HPS}_1.\text{Setup}(1^\kappa)$ , 其中  $pp_1 = (\mathsf{H}_1, SK_1, PK_1, X, L, W, \Pi_1, \alpha_1)$ ;
  - 运行  $pp_2 \leftarrow \text{HPS}_2.\text{Setup}(1^\kappa)$ , 其中  $pp_2 = (\mathsf{H}_2, SK_2, PK_2, X, L, W, \Pi_2, \alpha_2)$ ;
  - 输出公开参数  $pp = (pp_1, pp_2)$ . 公钥空间  $PK = PK_1 \times PK_2$ , 私钥空间  $SK = SK_1 \times SK_2$ , 密

文空间  $C = X \times \Pi_2$ , 会话密钥空间  $K = \Pi_1$ .

- $\text{KeyGen}(pp)$ : 解析  $pp = (pp_1, pp_2)$ , 执行以下步骤:
  1. 计算  $(pk_1, sk_1) \leftarrow \text{HPS}_1.\text{KeyGen}(pp_1)$ ;
  2. 计算  $(pk_2, sk_2) \leftarrow \text{HPS}_2.\text{KeyGen}(pp_2)$ ;
  3. 输出公钥  $pk = (pk_1, pk_2)$  和私钥  $sk = (sk_1, sk_2)$ .
- $\text{Encaps}(pk; r)$ : 以公钥  $pk = (pk_1, pk_2)$  和随机数  $r$  为输入, 执行以下步骤:
  1. 运行  $(x, w) \leftarrow \text{SampR}(r)$  随机采样语言  $L_1$  中的实例和相应证据;
  2. 通过  $\text{HPS}_1.\text{PubEval}(pk_1, x, w)$  计算实例  $x$  在  $\text{HPS}_1$  中的哈希证明  $\pi_1 \leftarrow \text{H}_1(sk_1, x)$ ;
  3. 通过  $\text{HPS}_2.\text{PubEval}(pk_2, x, w)$  计算实例  $x$  在  $\text{HPS}_2$  中的哈希证明  $\pi_2 \leftarrow \text{H}_2(sk_2, x)$ ;
  4. 输出实例  $x$  和  $\pi_2$  作为密文  $c$ , 其中  $\pi_2$  可以看做  $x$  的 MAC 值; 输出哈希证明  $\pi_1$  作为会话密钥  $k$ .
- $\text{Decap}(sk, c)$ : 以私钥  $sk = (sk_1, sk_2)$  和密文  $c = (x, \pi_2)$  为输入, 通过  $\text{HPS}_2.\text{PrivEval}(sk_2, x)$  计算  $x$  的哈希证明  $\pi'_2 \leftarrow \text{H}_2(sk_2, x)$ ; 如果  $\pi_2 \neq \pi'_2$  则输出  $\perp$ , 否则通过  $\text{HPS}_1.\text{PrivEval}(sk_1, x)$  计算  $x$  的哈希证明  $\pi_1 \leftarrow \text{H}_1(sk_1, x)$  以恢复会话密钥  $k$ .



构造 ??的正确性由  $\text{HPS}_1$  和  $\text{HPS}_2$  的完备性保证, 安全性由以下定理保证.

#### 定理 4.12

如果  $L \subseteq X$  上的 SMP 问题成立, 那么构造 ??中的 KEM 是 IND-CCA 安全的.



**证明** 为了便于安全分析, 首先对密文  $c = (x, \pi_2)$  做如下的分类:

- 良生成的 (well-formed)  $\iff x \in L$
- 有效的 (valid)  $\iff \text{H}_{sk_2}^2(x) = \pi_2$

根据以上定义, 良生成的密文有可能是无效的, 有效的密文也可能是非良生成的. 在基于 HPS 构造的 KEM 中, 非良生成的密文是“危险的”, 因为解封装询问的结果会泄漏关于私钥的信息.

以下通过游戏序列完成定理证明:

Game<sub>0</sub>: 对应真实的游戏

- 初始化:  $\mathcal{CH}$  生成  $pp_1 \leftarrow \text{HPS}_1.\text{Setup}(1^\kappa)$ ,  $pp_2 \leftarrow \text{HPS}_2.\text{Setup}(1^\kappa)$ , 计算  $(pk_1, sk_1) \leftarrow \text{HPS}_1.\text{KeyGen}(pp_1)$ ,  $(pk_2, sk_2) \leftarrow \text{HPS}_2.\text{KeyGen}(pp_2)$ , 发送  $pp = (pp_1, pp_2)$  和  $pk = (pk_1, pk_2)$  给对手  $\mathcal{A}$ .
- 挑战:  $\mathcal{CH}$  执行以下操作生成挑战
  1. 运行  $(x^*, w^*) \leftarrow \text{SampR}(r^*)$  随机采样  $L$  中的实例和证据;
  2. 通过  $\text{HPS}_1.\text{PubEval}(pk_1, x^*, r^*)$  计算哈希证明  $\pi_1^* \leftarrow \text{H}_1(sk_1, x^*)$ ;
  3. 通过  $\text{HPS}_2.\text{PubEval}(pk_2, x^*, w^*)$  计算哈希证明  $\pi_2^* \leftarrow \text{H}_2(sk_2, x^*)$ ;
  4. 令  $c^* = (x^*, \pi_2^*)$ ,  $k_0^* = \pi_1^*$ ,  $k_1^* \xleftarrow{R} \Pi$ ;
  5. 选择随机比特  $\beta \xleftarrow{R} \{0, 1\}$ , 发送  $(c^*, k_\beta^*)$  给  $\mathcal{A}$  作为挑战.
- 解封装询问: 当对手发起解封装询问  $c = (x, \pi_2)$  时,  $\mathcal{CH}$  分情况应答如下
  - $c = c^*$ : 返回  $\perp$ ;
  - $c \neq c^*$ : 如果  $\pi_2 = \text{HPS}_2.\text{PrivEval}(sk_2, x)$  返回  $\text{HPS}_1.\text{PrivEval}(sk_1, x)$ ; 否则返回  $\perp$ .

Game<sub>1</sub>: 与 CPA 构造情形类似, 该游戏的引入是为了将密文  $c^*$  由语言  $L$  内切换到语言外. 在挑战阶段,  $\mathcal{CH}$  通过  $\text{HPS}_1.\text{PrivEval}(sk_1, x^*)$  计算  $\pi_1^* \leftarrow \text{H}_1(sk_1, x^*)$ , 通过  $\text{HPS}_2.\text{PrivEval}(sk_2, x^*)$  计算  $\pi_2^* \leftarrow \text{H}_2(sk_2, x^*)$ . HPS 的投射性保证了 Game<sub>0</sub>  $\equiv$  Game<sub>1</sub>.

Game<sub>2</sub>: 将随机采样  $L$  中的实例和证据  $(x^*, w^*) \xleftarrow{R} \text{SampR}(r^*)$  切换为随机采样  $X \setminus L$  中的实例  $x^* \leftarrow \text{SampNo}(r^*)$ .

SMP 问题的困难性保证了敌手在相邻游戏中的视图计算不可区分:

$$\text{Game}_1 \approx_c \text{Game}_2$$

在游戏序列演进过程中, 仅在论证  $\text{Game}_1 \approx_c \text{Game}_2$  时依赖计算困难假设; 其余的分析均在信息论意义下 (information-theoretic) 完成, 从此刻起挑战者  $\mathcal{CH}$  拥有无穷计算能力.

$\text{Game}_3$ : 微调解密规则, 将直接拒绝非良生成但有效的 (ill-formed but valid) 密文. 对于解封装询问  $c = (x, \pi_2)$ , 只要  $x \notin L$ , 那么即使  $\pi_2 = \mathsf{H}_{sk_2}^2(x)$  也直接返回  $\perp$  表示拒绝. 改变规则的目的是拒绝所有危险密文, 从而确保解封装查询的应答不泄漏关于私钥的信息.

#### 断言 4.7

$$|\Pr[S_3] - \Pr[S_2]| \leq \text{negl}(\kappa).$$



**证明** 注意到正常的解封装算法会对此类密文返回解封装结果, 并不是直接返回  $\perp$  拒绝. 为了分析规则改变引发的差异, 引入如下事件  $E$ :

- $\mathcal{A}$  发起非良生成但有效的解封装询问, 即:  $x \notin L \wedge \pi_2 = \mathsf{H}_{sk_2}^2(x)$

显然如果事件  $E$  不发生, 那么  $\text{Game}_2$  与  $\text{Game}_3$  完全相同. 令  $Q$  表示  $\mathcal{A}$  发起解封装询问的最大次数,  $\mathsf{HPS}_2$  的 2-universal 保证了 (存疑):

$$\Pr[E] \leq Q/|\Pi_2| = \text{negl}(\kappa)$$

利用差异引理, 断言得证.  $\square$

$\text{Game}_4$ : 对所有良生成的解封装询问  $c = (x, \pi_2)$  也即  $x \in L$ ,  $\mathcal{CH}$  使用公钥  $pk = (pk_1, pk_2)$  和相应的证据  $w$  应答. 注意到  $\mathcal{CH}$  拥有无穷计算能力, 因此能够计算出  $x \in L$  的证据  $w$ . 该规则变化仅是为了说明对良生成密文的解封装不会额外泄漏关于私钥的信息, 不会引发敌手视图的任何改变, 因此  $\text{Game}_3 \equiv \text{Game}_4$ .

$\text{Game}_5$ : 随机采样  $\pi_1^* \xleftarrow{\text{R}} \Pi_1$  代替  $\pi^* \leftarrow \mathsf{H}_1(sk_1, x^*)$ .

#### 断言 4.8

敌手  $\mathcal{A}$  在  $\text{Game}_4$  和  $\text{Game}_5$  中的视图统计不可区分.



**证明** 敌手  $\mathcal{A}$  在  $\text{Game}_4$  和  $\text{Game}_5$  中的视图均由以下部分组成:

- 公开参数:  $pp = (pp_1, pp_2)$ ;
- 公钥:  $pk = (pk_1, pk_2)$ ;
- 挑战: 密文  $c^* = (x^*, \pi_2^*)$  和会话密钥  $k_\beta^*$ ;
- 解封装询问: 由公钥  $pk$  和敌手  $\mathcal{A}$  的询问确定.

接下来, 我们通过递增分布项的方式证明断言:

- 首先由  $\mathsf{HPS}_1$  的平滑性可知, 当  $x^* \xleftarrow{\text{R}} X \setminus L$  是有:

$$(pk_1, x^*, \boxed{\mathsf{H}_1(sk_1, x^*)}) \approx_s (pk_1, x^*, \boxed{U_{\Pi_1}})$$

- 将  $(pk_2, \pi_2^*)$  表示为  $g_{sk_2}(x^*)$ , 其中  $g_{sk_2}(x) := (\alpha_2(sk_2), \mathsf{H}_2(sk_2, x))$ . 复合引理 (composition lemma) 可推出  $X \approx_s Y \Rightarrow f(X) \approx_s f(Y)$ , 其中  $f$  可以是任意 (概率) 函数. 将上面公式左右两边的分布分别看成  $X$  和  $Y$ , 令  $f(pk_2, x^*, \pi_2) = (g_{sk_2}(x^*), pk_2, x^*, \pi_2)$ , 应用复合引理即可得:

$$(\cancel{pk_2}, \cancel{\pi_2^*}, \cancel{pk_1}, \cancel{x^*}, \cancel{\mathsf{H}_{sk_1}^1(x^*)}) \approx_s (\cancel{pk_2}, \cancel{\pi_2^*}, \cancel{pk_1}, \cancel{x^*}, U_{\Pi_1})$$

令  $view' = (pk, x^*, \pi_2^*, k_\beta^*)$ , 上面公式可以简写为  $view'_4 \approx_s view'_5$ .

3. 在左右两边添加解封装结果.  $\mathcal{CH}$  对解封装询问的应答总可以表示为  $f_{\text{decaps}}(view')$ ,  $f_{\text{decaps}}$  编码了敌手  $\mathcal{A}$  选择密文  $\{c_i\}$  的策略和解封装算法, 易知  $f_{\text{decaps}}$  是一个 PPT 算法. 再次应用复合引理, 可以得到:

$$(f_{\text{decaps}}(view'_4), view'_4) \approx_s (f_{\text{decaps}}(view'_5), view'_5)$$

根据敌手视图的定义, 可以得到  $\text{Game}_4 \approx_s \text{Game}_5$ , 断言得证.  $\square$

在  $\text{Game}_5$  中,  $k_0^*$  和  $k_1^*$  均从  $\Pi_1$  中随机采样. 因此对于任意敌手均有  $\Pr[S_5] = 0$ . 综合以上, 定理得证!  $\square$

## 小结

HPS 给出了基于 SMP 类型判定性问题构造公钥加密的范式, 在论证安全性时遵循如下的三步走 (三板斧) 套路:



1. 真实游戏中挑战密文为语言中的随机实例  $x \in L$ ;
2. 理想游戏中挑战密文为语言外的随机实例  $x \notin L$ , 在信息论意义下证明敌手优势可忽略;
3. 利用 SMP 完成语言内外的切换, 论证 PPT 敌手在真实游戏和理想游戏中的优势差可忽略.

论证过程与中国道家的“阴阳相生”思想暗合.

在很多情形下, 公钥加密的私钥嵌入于底层困难问题, 因此设计高等级安全公钥加密的一个常见难点是归约证明过程中, 归约算法  $\mathcal{R}$  需要在未知私钥的情形下模拟与私钥相关的预言机. 一个具体的例子就是难以证明 ElGamal PKE 具备私钥抗泄漏安全性, 因为私钥嵌入在底层 DDH 困难问题中. Cramer 和 Shoup 另辟蹊径, 绕过了该难点, 关窍是在基于 HPS 的公钥加密设计中, 公钥加密的密文嵌入于底层困难问题, 归约算法  $\mathcal{R}$  始终掌握私钥, 从而可以完美模拟任意与私钥相关的预言机. 正是该特性使得 HPS 的用途极为广泛, 远远超越了最初的 CCA 安全的公钥加密, 如 HPS 在基于口令的密钥交换 (PAKE: Password authenticated key exchange)、不经意传输 (Oblivious transfer) 的构造中均有重要应用, 更是达成密钥泄漏安全、消息依赖密钥安全等高等级安全的主流技术工具.

## 4.3 可提取哈希证明系统类

事要知其所以然。

—《朱子语类·卷九·论行知》

1991年, Rackoff 和 Simon [RS-CRYPTO-1991] 提出了构造 CCA 安全 PKE 的另一条技术路线:

1. 发送方随机选择会话密钥  $k$  并使用接收方的公钥对其加密得到密文  $c$ , 同时生成关于  $k$  的非交互零知识知识证明  $\pi$ , 将  $c$  和  $\pi$  一起发送给接收方;
2. 接收方先验证  $\pi$  的正确性, 若验证通过则利用私钥解密恢复会话密钥;

该条技术路线被称为 Rackoff-Simon 范式, 与 Naor-Yung 范式/Sahai 范式的不同之处是前者需要使用非交互零知识知识的证明 (NIZKPoK), 而后者使用的势非交互零知识证明 (NIZK)。

Cramer 和 Shoup 于 2002 年正式提出的哈希证明系统是 NIZK 的弱化: 公开可验证弱化为指定验证者, 表达能力由任意  $\mathcal{NP}$  语言限制为群论语言, 证明的形式特化为哈希值。2010 年, Wee [Wee-CRYPTO-2010] 提出了可提取哈希证明系统 (EHPS, extractable hash proof system), 并展示了如何基于 EHPS 以一种简洁、模块化的方式构造 CCA 安全的 PKE。该构造范式统一了几乎所有已知的基于计算性假设的 CCA 安全 PKE 方案。相对 HPS 是 NIZK 的弱化, EHPS 是 NIZKPoK 的弱化。以下首先介绍 EHPS 的定义和相关性质。

### 定义 4.11 (可提取哈希证明系统)

EHPS 包含以下多项式时间算法:

- $\text{Setup}(1^\kappa)$ : 以安全参数  $\kappa$  为输入, 输出公开参数  $pp = (\mathsf{H}, \mathsf{PK}, \mathsf{SK}, L, W, \Pi)$ , 其中  $L$  是由困难关系  $R_L$  定义的平凡  $\mathcal{NP}$  语言,  $\mathsf{H} : \mathsf{PK} \times L \rightarrow \Pi$  是由公钥集合  $\mathsf{PK}$  索引的一族带密钥哈希函数。关系  $R_L$  支持随机采样, 即存在 PPT 算法  $\text{SampRel}$  以随机数  $r$  为输入, 输出随机的“实例-证据”元组  $(x, w) \in R_L$ 。为了方便后续的应用,  $\text{SampRel}$  可以进一步分解为  $\text{Samplns}$  和  $\text{SampWit}$ , 前者随机采样语言中的实例, 后者随机采样证据, 对于任意随机数  $r \in R$ , 我们有  $(\text{Samplns}(r), \text{SampWit}(r)) \in R_L$ 。
- $\text{KeyGen}(pp)$ : 以公开参数  $pp$  为输入, 输出公钥  $pk$  和私钥  $sk$ 。
- $\text{PubEval}(pk, x, r)$ : 以公钥  $pk$ 、 $x \in L$  和随机数  $r$  为输入, 输出证明  $\pi \in \Pi$ 。正确性要求是当  $r$  是采样  $x$  的随机数时(即  $(x, w) \leftarrow \text{SampRel}(r)$ ), 算法正确计算出哈希证明:  $\pi = \mathsf{H}_{pk}(x)$ 。注意, 当给定采样随机数  $r$  时, 可以运行算法  $\text{SampRel}$  恢复  $x$ , 因此算法的第 2 项输入  $x$  可以省去。
- $\text{Ext}(sk, x, \pi)$ : 以私钥  $sk$ ,  $x \in L$  和证明  $\pi \in \Pi$  为输入, 输出证据  $w \in W \cup \perp$ 。正确性要求是:

$$\pi = \mathsf{H}_{pk}(x) \iff (x, \text{Ext}(sk, x, \pi)) \in R_L$$

- $\text{KeyGen}'(pp)$ : 以公开参数  $pp$  为输入, 输出公钥  $pk$  和私钥  $sk'$ 。
- $\text{PrivEval}(sk', x)$ : 以私钥  $sk'$  和  $x \in L$  为输入, 输出证明  $\pi \in \Pi$ 。正确性要求是  $\text{PrivEval}$  正确计算出哈希证明:  $\pi = \mathsf{H}_{pk}(x)$ 。

以上算法中,  $\text{KeyGen}$ 、 $\text{PubEval}$  和  $\text{Ext}$  工作在真实模式,  $\text{KeyGen}'$  和  $\text{PrivEval}$  工作在模拟模式, 两种模式共享同一个  $\text{Setup}$  算法生成公开参数。两种模式之间的关联是公钥的分布统计不可区分, 即:

$$\text{KeyGen}(pp)[1] \approx_s \text{KeyGen}'(pp)[1]$$



### 4.3.1 起源释疑

 **笔记** 上图解释了 EHPS 的命名渊源, 其本质上是指定验证者零知识知识的证明, 证明的形式是实例的哈希值, 故名可提取哈希证明系统。

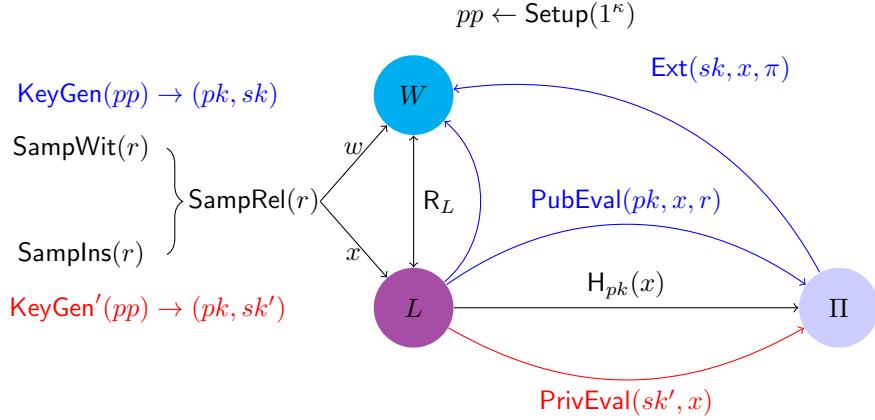


图 4.18: EHPS 的示意图

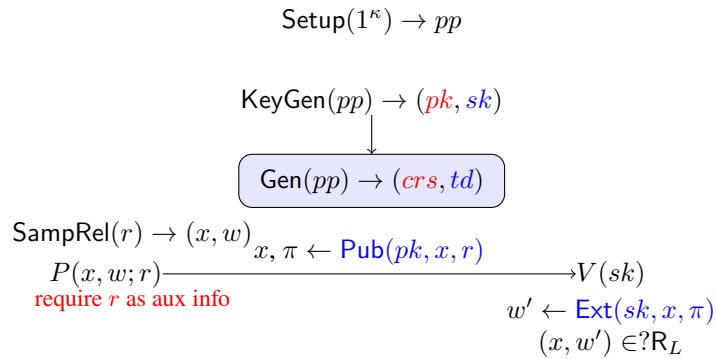


图 4.19: 从 DV-NIZKPoK 的视角解析 EHPS

- DV-NIZKPoK 的完备性和可提取性由  $\text{Ext}$  的正确性保证, 即在正常模式下,

$$\pi = \mathsf{H}_{\mathbf{pk}}(x) \iff (x, \mathsf{Ext}(sk, x, \pi)) \in R_L$$

其中  $\text{KeyGen}(pp) \rightarrow (pk, sk)$ .

- DV-NIZKPoK 的零知识性论证如下, 令  $\text{KeyGen}(pp) \rightarrow (\mathbf{pk}, sk)$ ,  $\text{KeyGen}'(pp) \rightarrow (\mathbf{pk}', sk')$ , 对于  $\forall x \in L$ , 我们有:

$$\mathbf{pk} \approx_s \mathbf{pk}' \Rightarrow (\mathbf{pk}, \mathsf{H}_{\mathbf{pk}}(x)) \approx_s (\mathbf{pk}', \mathsf{H}_{\mathbf{pk}}(x))$$

再由秘密求值算法的正确性  $\mathsf{H}_{\mathbf{pk}}(x) = \mathsf{PrivEval}(sk', x)$  可以得到:

$$(\mathbf{pk}, \mathsf{H}_{\mathbf{pk}}(x)) \approx_s (\mathbf{pk}', \mathsf{PrivEval}(sk', x))$$

### 4.3.2 EHPS 的构造

我们以针对  $L_{\text{CDH}}$  语言的 EHPS 构造为例, 获得对 EHPS 设计方式的直观认识. 令  $(\mathbb{G}, p, g)$  是算法  $\text{GroupGen}(1^\kappa)$  的输出, 其中  $\mathbb{G}$  是阶位素数  $p$  的群,  $g$  是生成元. 随机选取  $\mathbb{G}$  中的另一生成元  $g^\alpha$ , 其中  $\alpha \stackrel{\text{R}}{\leftarrow} \mathbb{Z}_p$ . 令  $pp = (\mathbb{G}, p, g, g^\alpha)$  是公开参数, 定义由  $pp$  索引的平凡  $\mathcal{NP}$  语言如下:

$$L_{\text{CDH}} = \{x \in X : \exists w \in W \text{ s.t. } w = x^\alpha\}$$

其中  $L = X = \mathbb{G}$ ,  $W = \mathbb{G}$ . 定义  $L_{\text{CDH}}$  的二元关系为  $R_{\text{CDH}}$ ,  $(x, w) \in R_{\text{CDH}} \iff w = x^\alpha$ . 容易验证:

- $R_{\text{CDH}}$  基于 CDH 假设是困难的.

- $R_{CDH}$  是高效可采样的: 存在 PPT 采样算法 SampRel 随机选取  $r \xleftarrow{R} \mathbb{Z}_p$ , 输出  $(g^r, (g^\alpha)^r) \in R_{CDH}$ .
- 如果  $\mathbb{G}$  是双线性映射群, 则  $R_{CDH}$  是公开可验证的.

#### 构造 4.12 ( $L_{CDH}$ 语言的 EHPS 构造)

$L_{CDH}$  的 EHPS 构造如下, 如图所示:

- $\text{Setup}(1^\kappa)$ : 以安全参数  $\kappa$  为输入, 输出公开参数  $pp = (\mathbb{G}, p, g, g^\alpha)$ . 其中  $pp$  还包括了对  $SK = \mathbb{Z}_p$ ,  $PK = \mathbb{G}$ ,  $L_{CDH} = X = \mathbb{G}$  和  $W = \mathbb{G}$  的描述.
- $\text{KeyGen}(pp)$ : 以公开参数  $pp$  为输入, 随机采样  $sk \xleftarrow{R} \mathbb{Z}_p$ , 计算  $pk = g^{sk} \in \mathbb{G}$ , 输出  $(pk, sk)$ .
- $\text{PubEval}(pk, x, r)$ : 以公钥  $pk$ 、实例  $x \in L_{CDH}$  和  $r \in \mathbb{Z}_p$  为输入, 输出  $\pi \leftarrow (g^\alpha \cdot pk)^r$ .
- $\text{Ext}(sk, x, \pi)$ : 以私钥  $sk$ 、实例  $x \in L_{CDH}$  和  $\pi$  为输入, 计算  $w \leftarrow \pi/x^{sk}$ , 如果  $(x, w) \in R_L$  则返回  $w$ , 否则返回  $\perp$ . 正确性由以下公式保证:

$$\pi/x^{sk} = (g^\alpha \cdot pk)^r / x^{sk} = (g^\alpha \cdot g^{sk})^r / g^{r \cdot sk} = (g^\alpha)^r = w$$

- $\text{KeyGen}'(pp)$ : 以公开参数  $pp$  为输入, 随机采样  $sk' \xleftarrow{R} \mathbb{Z}_p$ , 计算  $pk \leftarrow g^{sk'}/g^\alpha$ .
- $\text{PrivEval}(sk', x)$ : 以私钥  $sk'$  和实例  $x \in L_{CDH}$  为输入, 输出  $w \leftarrow x^{sk'}$ . 正确性由以下公式保证:

$$H_{pk}(x) = (g^\alpha \cdot pk)^r = (g^\alpha \cdot \underline{g^{sk'}/g^\alpha})^r = (g^{sk'})^r = x^{sk'}$$

容易验证, 两种模式下生成的  $pk$  服从同样的分布— $\mathbb{G}$  上的均匀分布.

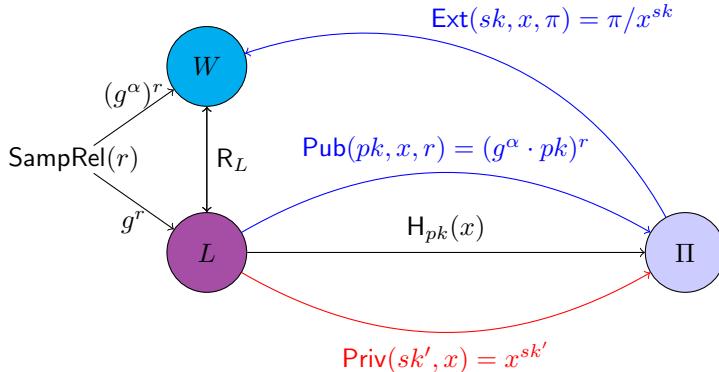


图 4.20:  $L_{CDH}$  的 EHPS

#### 4.3.3 基于 EHPS 构造 IND-CPA KEM

作为暖场应用, 我们首先介绍如何基于 EHPS 构造 CPA 安全的 KEM. 设计的思路源自 Rackoff-Simon 范式. 令  $R_L$  为定义在  $X \times W$  上的单向关系,  $hc : W \rightarrow K$  为相应的 hardcore function.

- 发送方扮演 EHPS 中的证明者, 运行  $\text{SampRel}(r)$  算法随机采样  $(x, w) \in R_L$ , 利用公钥  $pk$  和随机数  $r$  计算  $x$  的哈希证明  $\pi$ , 生成密文  $c = (x, \pi)$ , 计算证据  $w$  的 hardcore function 值作为会话密钥  $k$ .
- 接收方扮演 EHPS 中的验证者: 使用私钥  $sk$  从密文  $(x, \pi)$  中恢复  $w$ , 进而恢复会话密钥.

#### 构造 4.13 (基于 EHPS 的 CPA 安全 KEM)

从语言  $L$  的 EHPS 出发, 构造 CPA 安全的 KEM 如下:

- $\text{Setup}(1^\kappa)$ : 运行  $pp \leftarrow \text{EHPS}.\text{Setup}(1^\kappa)$ , 输出  $pp = (\mathbb{H}, SK, PK, X, L, W, \Pi)$  作为公开参数, 其中  $X \times \Pi$  作为密文空间,  $R_L$  hardcore function 的值域  $K$  作为会话密钥空间.

- $\text{KeyGen}(pp)$ : 运行  $(pk, sk) \leftarrow \text{EHPS.KeyGen}(\lambda)$ , 输出公钥  $pk$  和私钥  $sk$ .
- $\text{Encaps}(pk; r)$ : 以公钥  $pk$  和随机数  $r$  为输入, 执行如下步骤:
  1. 运行  $(x, w) \leftarrow \text{SampRel}(r)$  生成随机实例和相应证据;
  2. 通过  $\text{EHPS.Pub}(pk, x, r)$  计算实例  $x$  的哈希证明  $\pi \leftarrow H_{pk}(x)$ ;
  3. 输出  $(x, \pi)$  作为密文, 计算  $k \leftarrow hc(w)$  作为会话密钥.
- $\text{Decaps}(sk, c)$ : 以私钥  $sk$  和密文  $c = (x, \pi)$  为输入, 计算  $w \leftarrow \text{EHPS.Ext}(sk, x, \pi)$ , 如果  $(x, w) \notin R_L$  则输出  $\perp$ , 否则输出  $k \leftarrow hc(w)$ .



KEM 的正确性由 EHPS 的完备性和  $R_L$  的单向性保证, 安全性由如下定理保证.

### 定理 4.13

如果  $R_L$  是单向的, 那么构造 ?? 中的 KEM 是 IND-CPA 安全的.



**证明** 证明的目标是论证会话密钥  $hc(w^*)$  在敌手  $\mathcal{A}$  的视图中是伪随机的, 其中  $\mathcal{A}$  的视图包括:

- 公开参数  $pp$ ;
- 公钥  $pk$ : 与  $w^*$  无关;
- 密文  $c^* = (x^*, \pi^*)$ :  $R_L$  的单向性保证了  $x^*$  隐藏了  $w^*$ , EHPS 的零知识性进一步保证了  $\pi^*$ (相对于  $x^*$ ) 不会额外泄漏关于  $w^*$  的信息.

我们通过以下的游戏序列组织证明.

$\text{Game}_0$ : 对应真实的游戏.  $\mathcal{CH}$  在真实模式下运行 EHPS 与敌手  $\mathcal{A}$  交互.

- 初始化:  $\mathcal{CH}$  计算  $pp \leftarrow \text{EHPS.Setup}(1^\kappa)$ ,  $(pk, sk) \leftarrow \text{EHPS.KeyGen}(pp)$ , 将  $pp$  和  $pk$  发送给  $\mathcal{A}$ .
- 挑战:  $\mathcal{CH}$  按照以下步骤生成挑战
  1. 随机采样  $(x^*, w^*) \leftarrow \text{SampRel}(r^*)$ ;
  2. 通过  $\text{EHPS.PubEval}(pk, x^*, r^*)$  公开计算  $\pi^* \leftarrow H_{pk}(x^*)$ ;
  3. 计算  $k_0^* \leftarrow hc(w^*)$ , 随机采样  $k_1^* \xleftarrow{R} K$ ;
  4. 随机选取  $\beta \xleftarrow{R} \{0, 1\}$ , 将  $(c^* = (x^*, \pi^*), k_\beta^*)$  发送给  $\mathcal{A}$  作为挑战.
- 应答:  $\mathcal{A}$  输出对  $\beta$  的猜测  $\beta'$ ,  $\mathcal{A}$  成功当且仅当  $\beta' = \beta$ .

为了利用 EHPS 的零知识性论证  $\pi^*$  不额外泄漏关于  $w^*$  的信息, 需要将 EHPS 由真实模式切换到模拟模式.

$\text{Game}_1$ :  $\mathcal{CH}$  在模拟模式下运行 EHPS 与敌手  $\mathcal{A}$  交互.

- 初始化:  $\mathcal{CH}$  计算  $(pk, sk') \leftarrow \text{EHPS.KeyGen}'(pp)$ .
- 挑战:  $\mathcal{CH}$  在第二步通过  $\text{EHPS.Priv}(sk', x^*)$  计算  $\pi^* \leftarrow H_{pk}(x^*)$ .

敌手  $\mathcal{A}$  在游戏中的视图为  $(pp, pk, x^*, k_\beta^*)$ . 容易验证, EHPS 的零知识性保证了  $\text{Game}_0 \approx_s \text{Game}_1$ :

### 断言 4.9

如果  $R_L$  是单向的,  $\text{Adv}_{\mathcal{A}}^{\text{Game}_1} = \text{negl}(\kappa)$ .



**证明** 证明思路是如果存在  $\mathcal{A}$  以不可忽略的优势赢得  $\text{Game}_1$ , 那么可以构造出  $\mathcal{B}$  以不可忽略的优势打破  $hc$  的伪随机性, 从而与单向性假设冲突. 给定关于  $hc$  的伪随机性挑战  $pp$  和  $(x^*, k_\beta^*)$ , 其中  $(x^*, w^*) \leftarrow \text{SampRel}(r^*)$ ,  $\mathcal{B}$  模拟  $\text{Game}_1$  中的挑战者  $\mathcal{CH}$  与  $\mathcal{A}$  交互, 目标是猜测  $\beta$ .

- $\mathcal{B}$  运行 EHPS 的模拟模式与  $\mathcal{A}$  在  $\text{Game}_1$  进行交互, 在初始化阶段不再采样  $x^*$  而是直接嵌入接收到的  $x^*$ , 在挑战阶段将  $R_L$  的挑战  $(x^*, k_\beta^*)$  作为  $\mathcal{A}$  的 KEM 挑战. 最终,  $\mathcal{B}$  输出  $\mathcal{A}$  的猜测  $\beta'$ .

容易验证,  $\mathcal{B}$  在  $\text{Game}_1$  中的模拟是完美的. 因此  $\mathcal{B}$  打破  $R_L$  伪随机性的优势与  $\text{Adv}_{\mathcal{A}}^{\text{Game}_1}$  相同. 断言得证! □

综上, 定理得证!



### 4.3.4 基于 EHPS 构造 IND-CCA KEM

我们首先从安全归约的角度分析基于 EHPS 构造 IND-CCA KEM 的难点. EHPS 模拟模式下的  $sk'$  可以在不知晓采样实例  $x$  随机数的情况下正确计算出相应的哈希证明, 但无法提取出证据, 因此归约算法无法应答解密询问. 因此, 为了构造 IND-CCA 的 KEM, 需要赋予 EHPS 更丰富的功能.

PKE/KEM 的选择密文安全游戏是“全除一”(ABO, all-but-one) 式的— $\mathcal{A}$  可以发起除挑战密文  $x^*$  以外的任意解密/解封装询问. Wee [Wee-CRYPTO-2010] 引入了量身定制的 ABO-EHPS.

#### 定义 4.12 (ABO-EHPS)

ABO-EHPS 与 EHPS 的定义差别集中在模拟模式, 真实模式下完全相同. 与 EHPS 相比, ABO-EHPS 在模拟模式下的功能更加丰富.

- $\text{KeyGen}'(pp, x^*)$ : 以公开参数  $pp$  和  $x^* \in L$  为输入, 输出  $(pk, sk')$ .
- $\text{PrivEval}(sk', x^*)$ : 以私钥  $sk'$  和  $x^*$  为输出, 输出证明  $\pi^* = H_{pk}(x^*)$ .
- $\text{Ext}'(sk', x, \pi)$ : 以私钥  $sk'$ 、 $x \neq x^*$  和  $\pi \in \Pi$  为输入, 输出证据  $w \in W$ . 正确性的要求是:

$$\pi = H_{pk}(x) \iff (x, \text{Ext}'(sk', x, \pi)) \in R_L$$

$\text{KeyGen}'$  算法以预先嵌入的点  $x^*$  为输入, 输出相应的密钥对  $(pk, sk')$ . ABO 的含义是模拟模式中的  $sk'$  具备以下功能:

- “一除全” 哈希求值 (one-out-all evaluation):  $sk'$  可以计算  $x^*$  的哈希值  $H_{pk}(x^*)$ .
- “全除一” 证据抽取 (all-but-one extraction):  $sk'$  可以从除  $x^*$  以外的点  $x$  和相应的证明中正确抽出证据  $\text{Ext}'(sk', x, \pi)$ .



#### 注记 4.12

模拟模式下  $sk'$  的功能在 CCA 安全归约中起到如下作用:

- “一除全” 哈希求值允许归约算法  $\mathcal{R}$  生成挑战密文  $c^* = (x^*, \pi^*)$ .
- “全除一” 证据抽取允许归约算法  $\mathcal{R}$  回答所有合法的解封装询问  $c \neq c^*$ .



Wee [Wee-CRYPTO-2010] 展示了如何基于 EHPS 构造 ABO-EHPS.

#### 构造 4.14 (基于 EHPS 的 ABO-EHPS 构造)

起点: 二元关系  $R_L$  的 EHPS

设计目标: 二元关系  $R_L$  的 ABO-EHPS

设计思路: 不妨设  $L$  中每个实例均可编码为  $n$  长的比特串, 利用 DDN 结构 [DDN-STOC-1991] 实现 ABO 功能. 具体构造如下:

- $\text{Setup}(1^\kappa)$ : 运行  $pp \leftarrow \text{EHPS}.\text{Setup}(1^\kappa)$ ;
- $\text{KeyGen}(pp)$ : 独立运行  $\text{EHPS}.\text{KeyGen}(pp)$  算法  $2n$  次, 生成  $\{(pk_{i,b}, sk_{i,b})\}_{i \in [n], b \in \{0,1\}}$ , 输出公钥  $pk = \{pk_{i,0}, pk_{i,1}\}_{i \in [n]}$  和私钥  $sk = \{sk_{i,0}, sk_{i,1}\}_{i \in [n]}$ .
- $\text{PubEval}(pk, x, r)$ : 对所有的  $i \in [n]$  计算  $\pi_i \leftarrow \text{EHPS}.\text{PubEval}(pk_{i,x_i}, x, r)$ , 输出  $\pi = (\pi_1, \dots, \pi_n)$ .
- $\text{Ext}(sk, x, \pi)$ : 对所有的  $i \in [n]$  计算  $w_i \leftarrow \text{EHPS}.\text{Ext}(sk_{i,x_i}, x, \pi_i)$ , 如果所有结果一致则输出, 否则返回  $\perp$ .
- $\text{KeyGen}'(pp, x^*)$ : 独立运行  $\text{EHPS}.\text{KeyGen}'(pp)$  算法  $n$  次生成  $\{(pk_{i,x_i^*}, sk_{i,x_i^*})\}_{i \in [n]}$ , 独立运行  $\text{EHPS}.\text{KeyGen}(pp)$  算法  $n$  次生成  $\{(pk_{i,1-x_i^*}, sk_{i,1-x_i^*})\}_{i \in [\ell]}$ , 输出  $pk = (pk_{i,0}, pk_{i,1})_{i \in [n]}$  and  $sk' = (sk_{i,0}, sk_{i,1})_{i \in [n]}$ .
- $\text{PrivEval}(sk', x^*)$ : 对所有的  $i \in [n]$  计算  $\pi_i \leftarrow \text{EHPS}.\text{Priv}'(sk_{i,x_i^*}, x^*)$ , 输出  $\pi = (\pi_1, \dots, \pi_n)$ .
- $\text{Ext}'(sk', x, \pi)$ : 对所有满足  $x_i^* = x_i$  的索引  $i \in [n]$  验证  $\pi_i = \text{EHPS}.\text{PrivEval}(sk_{i,x_i}, x_i)$  是否成立, 如

果否则输出  $\perp$ , 如果是则继续对所有满足  $x_i^* \neq x_i$  的索引  $i \in [n]$  计算  $\text{EHPS}.\text{Ext}(sk_{i,x_i}, x, \pi_i)$ , 如果提取结果一致则输出, 否则输出  $\perp$ .



ABO-EHPS 真实模式下算法的正确性由 EHPS 对应算法保证.

$n = 3$	$pk_{1,0}$ $sk_{1,0}$	$pk_{2,0}$ $sk_{2,0}$	$pk_{3,0}$ $sk_{3,0}$
	$pk_{1,1}$ $sk_{1,1}$	$pk_{2,1}$ $sk_{2,1}$	$pk_{3,1}$ $sk_{3,1}$

图 4.21: 真实模式下  $n = 3$  时密钥结构图示

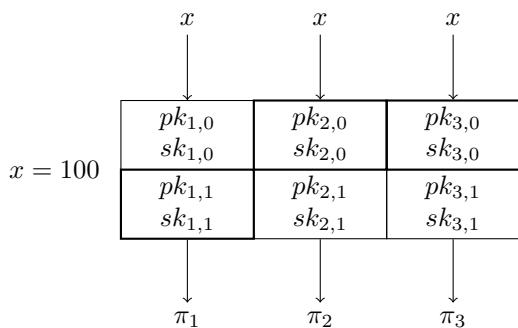


图 4.22: 真实模式下  $x = 010$  时哈希证明计算图示

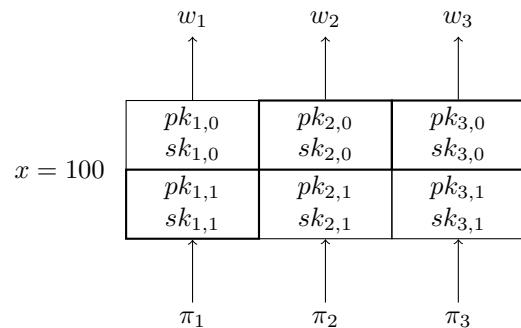


图 4.23: 真实模式下  $x = 010$  时证据提取图示

ABO-EHPS 模拟模式下算法的正确性由 DDN 结构和 EHPS 对应算法保证. ABO-EHPS 两种模式下公钥分布的统计不可区分性由 EHPS 两种模式下公钥分布的统计不可区分性与各公钥分量生成的独立性保证.

$x^* = 010$	$pk_{1,0}$ $sk'_{1,0}$	$pk_{2,0}$ $sk_{2,0}$	$pk_{3,0}$ $sk'_{3,0}$
	$pk_{1,1}$ $sk_{1,1}$	$pk_{2,1}$ $sk'_{2,1}$	$pk_{3,1}$ $sk_{3,1}$

图 4.24: 模拟模式下  $n = 3, x^* = 010$  时的密钥生成

基于 ABO-EHPS 设计 IND-CCA KEM 的方式与构造 ?? 完全相同. KEM 构造的正确性由 ABO-EHPS 的正确性和  $R_L$  的单射性保证, 安全性由以下定理保证.

#### 定理 4.14

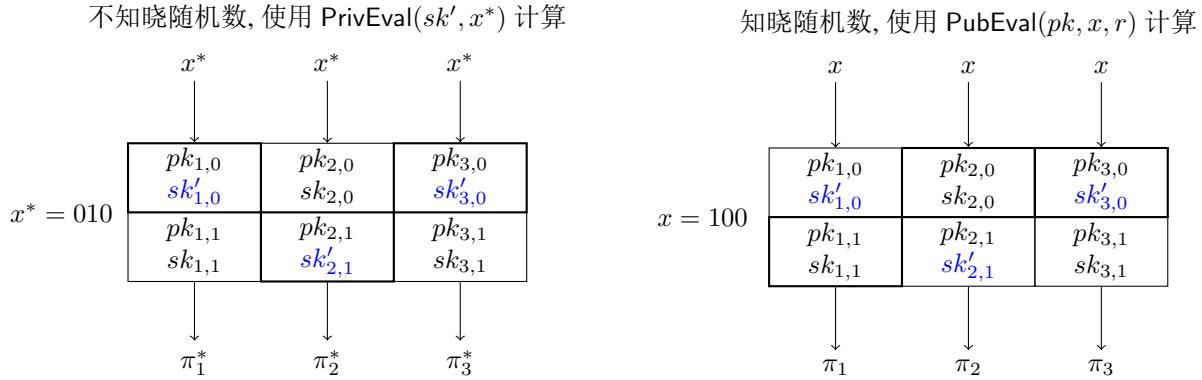
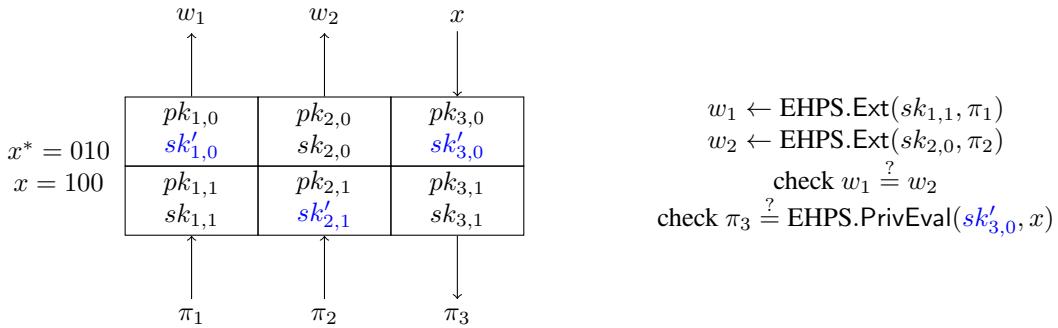
如果  $R_L$  是单向的, 那么 KEM 是 IND-CCA 安全的.



**证明** 证明的思路仍然是首先由真实模式切换到模拟模式, 再在模拟模式下利用零知识性证明安全性. 证明的要点是保证两种模式下对解密谕言机  $\mathcal{O}_{\text{decap}}$  回复的一致性. 以下通过游戏序列完成定理证明:

$\text{Game}_0$ : 对应真实的游戏.  $\mathcal{CH}$  在真实模式下运行 ABO-EHPS 与敌手  $\mathcal{A}$  交互.

- 初始化:  $\mathcal{CH}$  计算  $pp \leftarrow \text{Setup}(1^\kappa), (pk, sk) \leftarrow \text{KeyGen}(\lambda)$ , 将  $pp$  和  $pk$  发送  $\mathcal{A}$ .
- 挑战:  $\mathcal{CH}$  按照以下步骤生成挑战
  1. 随机采样  $(x^*, w^*) \leftarrow \text{SampRel}(r^*)$ ;
  2. 通过  $\text{PubEval}(pk, x^*, r^*)$  公开计算  $\pi^* \leftarrow H_{pk}(x^*)$ ;
  3. 计算  $k_0^* \leftarrow \text{hc}(w^*)$ , 随机采样  $k_1^* \xleftarrow{R} K$ ;
  4. 随机选取  $\beta \xleftarrow{R} \{0, 1\}$ , 将  $(c^* = (x^*, \pi^*), k_\beta^*)$  发送给  $\mathcal{A}$  作为挑战.

图 4.25: 模拟模式下  $x^* = 010$  时哈希证明计算图 4.26: 模拟模式下  $x = 100$  时哈希证明计算图 4.27: 模拟模式下  $x = 100$  时证据提取过程

- 解封装询问  $c = (x, \pi) \neq c^*$ : 计算  $w \leftarrow \text{Ext}(sk, x, \pi)$ , 如果  $(x, w) \in R_L$  则输出  $\text{hc}(w)$ , 否则输出  $\perp$ .
- 应答:  $\mathcal{A}$  输出对  $\beta$  的猜测  $\beta'$ ,  $\mathcal{A}$  成功当且仅当  $\beta' = \beta$ .

为了利用 ABO-EHPS 的零知识性论证  $\pi^*$  和解封装询问不额外泄漏关于  $w^*$  的信息, 需要将 ABO-EHPS 由真实模式切换到模拟模式. 为此, 先引入以下游戏作为过渡.

**Game<sub>1</sub>:** 与 Game<sub>0</sub> 完全相同, 惟一的区别是  $\mathcal{CH}$  将  $(x^*, w^*) \leftarrow \text{SampRel}(r^*)$  由挑战阶段提前至初始化阶段. 显然, 该变化不会对敌手的视图有任何改变. 因此有:

$$\text{Game}_0 \equiv \text{Game}_1$$

**Game<sub>2</sub>:** 本游戏对解封装应答方式稍加改动, 以便于后续游戏将密文的 ABO 解封装询问转化为 ABO-EHPS 相对于  $x^*$  的 ABO 证据抽取. 对于解封装询问  $c = (x, \pi) \neq c^*$ ,  $\mathcal{CH}$  应答如下:

- $x = x^* \wedge \pi \neq \pi^*$ : 直接返回  $\perp$ .
- $x \neq x^*$ : 计算  $w \leftarrow \text{Ext}(sk, x, \pi)$ , 如果  $(x, w) \in R_L$  则返回  $\text{hc}(w)$ , 否则返回  $\perp$ .

由于  $H_{pk}$  是确定性算法, 因此 Game<sub>2</sub> 与 Game<sub>1</sub> 中的解封装应答完全相同.

**Game<sub>3</sub>:**  $\mathcal{CH}$  在模拟模式下运行 ABO-EHPS 与敌手  $\mathcal{A}$  交互.

- 初始化:  $\mathcal{CH}$  与上一游戏的区别在于通过  $(pk, sk') \leftarrow \text{KeyGen}'(pp, x^*)$  生成密钥对.
- 挑战:  $\mathcal{CH}$  与上一游戏的区别在于通过  $\text{PrivEval}(sk', x^*)$  计算  $\pi^* \leftarrow H_{pk}(x^*)$ .
- 解封装询问  $c = (x, \pi) \neq c^*$ :  $\mathcal{CH}$  应答如下
  - $x = x^* \wedge \pi \neq \pi^*$ : 直接返回  $\perp$ .
  - $x \neq x^*$ : 计算  $w \leftarrow \text{Ext}'(sk', x, \pi)$ , 如果  $(x, w) \in R_L$  则返回  $\text{hc}(w)$ , 否则返回  $\perp$ .

基于以下事实, 我们有:  $\text{Game}_2 \approx_s \text{Game}_3$

- $\text{KeyGen}(pp)[1] \approx_s \text{KeyGen}'(pp, x^*)[1]$
- $\text{PubEval}(pk, x^*, r^*) = H_{pk}(x^*) = \text{PrivEval}(sk', x^*)$

- 对于解封装询问  $c = (x, \pi)$ : 当  $x = x^*$  时, 均返回  $\perp$ ; 当  $x \neq x^*$  时, ABO-EHPS 真实模式和模拟模式的正确性以及解封装算法“提取-检验”的设计保证了应答一致.

#### 断言 4.10

如果  $R_L$  是单向的, 那么  $\text{Adv}_{\mathcal{A}}^{\text{Game}_3} = \text{negl}(\kappa)$ .



**证明** 证明思路是如果存在  $\mathcal{A}$  以不可忽略的优势赢得  $\text{Game}_3$ , 那么可以构造出  $\mathcal{B}$  以不可忽略的优势打破  $\text{hc}$  的伪随机性, 从而与  $R_L$  的单向性假设冲突. 给定关于  $\text{hc}$  的伪随机性挑战  $pp$  和  $(x^*, k_\beta^*)$ , 其中  $(x^*, w^*) \leftarrow \text{SampRel}(r^*)$ ,  $\mathcal{B}$  模拟  $\text{Game}_3$  中的挑战者  $\mathcal{CH}$  与  $\mathcal{A}$  交互, 目标是猜测  $\beta$ .

- $\mathcal{B}$  运行 ABO-EHPS 的模拟模式与  $\mathcal{A}$  进行交互, 其在初始化阶段不再采样  $x^*$  而是直接嵌入接收到的  $x^*$ , 在挑战阶段将  $\text{hc}$  的挑战  $(x^*, k_\beta^*)$  作为  $\mathcal{A}$  的 KEM 挑战. 最终,  $\mathcal{B}$  输出  $\mathcal{A}$  的猜测  $\beta'$ .

容易验证,  $\mathcal{B}$  在  $\text{Game}_3$  中的模拟是完美的. 因此  $\mathcal{B}$  打破  $\text{hc}$  伪随机性的优势与  $\text{Adv}_{\mathcal{A}}^{\text{Game}_3}$  相同. 断言得证!

综上, 定理得证!

Wee [Wee-CRYPTO-2010] 展示了 ABO-EHPS 蕴含 ATDR.

#### 构造 4.15 (基于 ABO-EHPS 的 ATDR 构造)

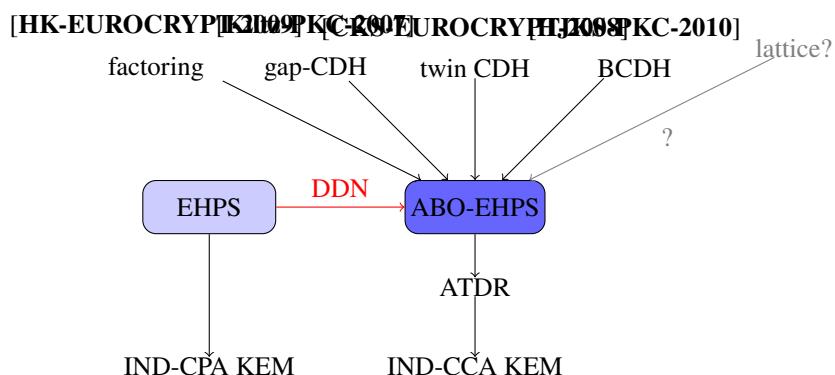
- $\text{Setup}(1^\kappa)$ : 运行  $pp \leftarrow \text{ABO-EHPS}.\text{Setup}(1^\kappa)$ .
- $\text{KeyGen}(pp)$ : 运行  $(pk, sk) \leftarrow \text{ABO-EHPS}.\text{KeyGen}(pp)$ , 令  $pk$  为求值公钥  $ek$ ,  $sk$  为求逆陷门  $td$ .
- $\text{Sample}(pk; r)$ : 运行  $(x, w) \leftarrow \text{SampRel}(r)$ , 通过  $\text{ABO-EHPS}.\text{PubEval}(pk, x, r)$  计算  $\pi \leftarrow H_{pk}(x)$ , 输出  $(w, (x, \pi))$ .
- $\text{TdInv}(td, (x, \pi))$ : 计算  $w \leftarrow \text{ABO-EHPS}.\text{Ext}(sk, (x, \pi))$ , 如果  $(x, w) \in R$  则返回  $w$ , 否则返回  $\perp$ .



上述 ATDR 构造的自适应单向性由 ABO-EHPS 的性质  $R_L$  的单向性保证. 该构造也在更抽象的层面解释了基于 ABO-EHPS 设计 CCA 安全 KEM 的实质是在构造 ATDR.

### 4.3.5 小结

EHPS 的重要理论意义在于它阐释统一了一大类标准模型下的基于计算性假设的 IND-CCA PKE 方案 [Kiltz-PKC-2007; CKS-EUROCRYPT-2008; HK-EUROCRYPT-2009; HJKS-PKC-2010]. 绝大多数标准模型下的 PKE 构造都可纳入 EHPS 和 HPS 的设计范式, 这也从公钥加密的角度再次证实了零知识证明的强大威力. 目前尚未知晓如何基于 lattice 上的困难问题构造 EHPS.



#### 4.3.5.1 HPS 与 EHPS 的分析比较

相同点

- 均可看成指定验证者的零知识证明系统 (DV-NIZK).
- 证明的形式是哈希值.

不同点

- HPS 是标准的证明系统, 而 EHPS 是知识的证明系统.
- HPS 中哈希函数族  $H_{sk}$  由私钥索引, EHPS 中哈希函数族  $H_{pk}$  有公钥索引.
- 在基于 HPS 的 PKE 构造中, 密文  $c$  是实例  $x$ , 会话密钥  $k$  是证明  $\pi$ .
  - HPS 的正确性保证了 PKE 的正确性
  - HPS 的合理性(哈希函数的平滑性、一致性)与 SMP 问题的困难性保证了 PKE 的安全性, 在证明过程中, 挑战实例需要从语言  $L$  上切换到语言外  $X \setminus L$ .
- 在基于 EHPS 的 PKE 构造中, 密文  $c$  由实例  $x$  和证明  $\pi$  组成, 会话密钥  $k$  是证据  $w$ .
  - EHPS 的知识提取性质保证了 PKE 的正确性
  - EHPS 的零知识性和二元关系的单向性保证了 PKE 的安全性, 在证明过程中, EHPS 需要由真实模式切换为模拟模式.

## 4.4 程序混淆类

只要代码足够乱, 就没人发现我写错.

— 网上·佚名

### 4.4.1 背景知识

程序混淆 (program obfuscation) 是一种编译的方法技术, 它将容易理解的源程序转化成难以理解的形式, 同时保持原有功能性不变. 程序混淆概念起源于上世界 70 年代的代码混淆领域, 在软件保护领域 (如软件水印、防逆向工程) 有着广泛的应用, 然而一直缺乏严格的安全定义.

```

319 int KDF(ZZn2 x,char *s)
320 { // Hash an Fp2 to an n-byte string
321     sha256 sh;
322     Big a,b;
323     int m;
324
325     shs256_init(&sh);
326     x.get(a,b);
327
328     while (a>0)
329     {
330         m=a%256;
331         shs256_process(&sh,m);
332         a/=256;
333     }
334     while (b>0)
335     {
336         m=b%256;
337         shs256_process(&sh,m);
338         b/=256;
339     }
340     shs256_hash(&sh,s);

```

```

#include<stdio.h> #include<string.h> main()
{
    /*acg017771cp_-,08r8)N38X4D+ACW(+01D5793
     fpsta(1445,954,stdin) (+1=|[strlen(D)[D-1]=0
     while(0){switch(*I&&isalnum(*D)-1){case
     strpmb(0,1+2)+2,-0,34;while(*I&3&8&0|0|-1
     putchar((0&337*I&8)|((I==mchr( 1 , 0 , 44
     break;case 1 : ;}+1+(*&0x31)(1-1+*(+0x81)*32);
     (*I+=1)<2>>1>3B);case 0:putchar(((+0,32));}

```

图 4.28: 程序混淆

Barak 等 [Barak-CRYPTO-2001] 首次将程序混淆引入密码学领域, 将程序从狭义的代码泛化为广义的算法, 同时提出了虚拟黑盒 (VBB, virtual black-box) 混淆的严格定义.

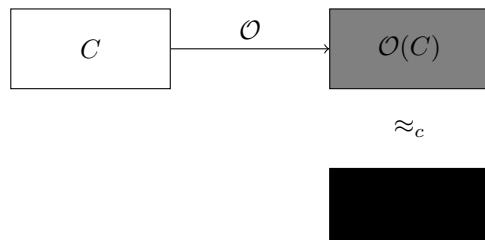


图 4.29: 虚拟黑盒混淆

我们回顾不可区分混淆 (indistinguishability obfuscator,  $i\mathcal{O}$ ) [Garg-FOCS-2013] 如下.

#### 定义 4.13 (虚拟黑盒混淆)

我们称一个 uniform PPT machine  $i\mathcal{O}$  是电路簇  $\{\mathcal{C}_\kappa\}$  的不可区分混淆器当且仅当其满足以下两个条件:

- 功能保持: 对于任意安全参数  $\kappa \in \mathbb{N}$ 、任意的  $C \in \mathcal{C}_\kappa$  和所有输入  $x \in \{0,1\}^*$  有:

$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\kappa, C)] = 1$$

- 虚拟黑盒混淆: 存在 PPT 的模拟器  $\mathcal{S}$ , 对于任意  $C \in \{\mathcal{C}_\kappa\}$ , 对于任意 PPT 敌手  $\mathcal{A}$ , 我们有:

$$\mathcal{A}(\mathcal{O}(C)) \approx_c \mathcal{S}^C$$

其中公式左边表示  $\mathcal{A}$  的视图, 公式右边表示  $\mathcal{S}$  在通过对  $C$  进行黑盒访问所输出的视图.



**笔记** 虚拟黑盒混淆的定义是基于模拟的游戏, 刻画的是 PPT 敌手从混淆程序  $\mathcal{O}(C)$  中获取的任何信息不会比黑盒访问  $C$  获得的信息更多. 换言之, 掌握  $\mathcal{O}(C)$  的敌手视图可以由模拟器通过黑盒访问  $C$  模拟得出. VBB 试图隐

藏程序  $C$  的所有细节. 比如  $C$  以平方差公式计算  $x^2 - 1$ , 即  $C(x) = (x + 1)(x - 1)$ . 那么敌手在获得  $\mathcal{O}(C)$  后, 掌握的所有信息等价输入/输出元组  $(x, x^2 - 1)$ , i.e.,  $(1, 0), (2, 3), (3, 8), \dots$

VBB 的混淆定义强到极致, 因此在密码学中应用起来颇为简单直观. 事实上, 在 1976 年 Diffie 和 Hellman 的划时代论文 [DH-IEEE-IT-1976] 中, 就已经提出了利用混淆器将对称加密方案编译为公钥加密方案的想法(如图 ??):

1. 将对 SKE 的加密算法  $\text{Enc}(sk, m, r)$  中的第一个输入 hardwired 进电路, 得到  $\text{Enc}_{sk}(m, r)$ ;
2. 利用混淆器编译  $\text{Enc}_{sk}(\cdot, \cdot)$ , 将得到的混淆程序作为公钥  $pk$ .

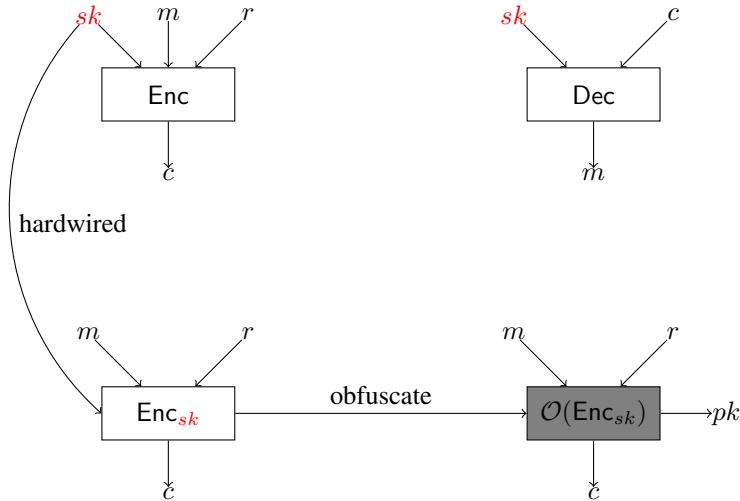


图 4.30:  $\text{SKE} \Rightarrow \text{PKE}$  via obfuscation

VBB 的安全定义至强, Barak 等 [Barak-CRYPTO-2001] 指出 VBB“too good to be true!”—不存在针对任意电路(通用, general-purpose)的 VBB. VBB 因为安全太强以至于不存在, Garg 等 [Garg-FOCS-2013] 降低了安全性要求, 引入了不可区分混淆(indistinguishability obfuscator,  $i\mathcal{O}$ ).

#### 定义 4.14 (不可区分混淆 ( $i\mathcal{O}$ ))

我们称一个 uniform PPT machine  $i\mathcal{O}$  是电路簇  $\{\mathcal{C}_\kappa\}$  的不可区分混淆器当且仅当其满足以下两个条件:

- 功能保持: 对于任意安全参数  $\kappa \in \mathbb{N}$ 、任意的  $C \in \mathcal{C}_\kappa$  和所有输入  $x \in \{0, 1\}^*$  有:

$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\kappa, C)] = 1$$

- 不可区分混淆: 对于任意 PPT 敌手  $(\mathcal{S}, \mathcal{D})$ , 存在关于安全参数的可忽略函数  $\alpha$  使得: 如果  $\Pr[\forall x, C_0(x) = C_1(x) : (C_0, C_1, aux) \leftarrow \mathcal{S}(\kappa)] \geq 1 - \alpha(\kappa)$ , 那么我们有:

$$|\Pr[\mathcal{D}(aux, i\mathcal{O}(\kappa, C_0)) = 1] - \Pr[\mathcal{D}(aux, i\mathcal{O}(\lambda, C_1)) = 1]| \leq \alpha(\kappa)$$



#### 注记 4.13

- 不可区分混淆的定义类似加密方案的不可区分性, 对于任意功能相同的电路  $C_0$  和  $C_1$ , 均有  $i\mathcal{O}(C_0) \approx_c i\mathcal{O}(C_1)$ . 这里, 我们可以把电路  $C$  类比为消息,  $i\mathcal{O}$  类比为加密算法. 与 VBB 试图隐藏电路的所有信息不同,  $i\mathcal{O}$  只试图隐藏电路的部分信息: 比如  $C_0(x) = (x + 1)(x - 1)$ ,  $C_1(x) = (x + 2)(x - 2) + 3$ , 那么如果混淆后的程序均是  $x^2 - 1$  即可满足不可区分安全性. 非严格的说,  $i\mathcal{O}$  试图在以统一的方式完成同质的计算.
- 在上述定义中, 条件  $\Pr[\forall x, C_0(x) = C_1(x) : (C_0, C_1, aux) \leftarrow \mathcal{S}(\kappa)] \geq 1 - \alpha(\kappa)$  并不意味着  $C_0$  和  $C_1$  存在差异输入 (differing-inputs), 而指的是  $C_0$  和  $C_1$  以极高的概率功能性完全相同, 这一点体现在概

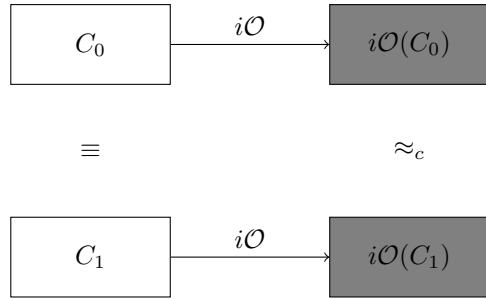


图 4.31: 不可区分混淆示意图

率空间定义在  $\mathcal{S}$  的随机带而与  $x$  无关.

- $aux$  表示  $\mathcal{S}$  在采样  $C_0, C_1$  过程中得到的任意信息, 用于辅助  $\mathcal{D}$  区分  $iO(C_0)$  和  $iO(C_1)$ .



**差异输入混淆.** 如果在上述  $iO$  定义中, 将  $\mathcal{S}$  所采样两个电路的要求由功能性完全相同放宽为允许存在差异输入, 则得到的是更强的混淆器, 称为差异输入混淆 ( $diO$ , differing-input obfuscation). [BCP-TCC-2014] 中给出了正面结果: 证明了  $iO$  蕴含多项式级别差异输入规模的  $diO$ . [BSW-EUROCRYPT-2016] 中给出了负面结果: 证明了亚指数安全 (sub-exponentially secure) 的单向函数存在, 则针对无界输入 Turing machine(TMs with unbounded inputs) 亚指数安全的  $diO$  不存在.

我们再把注意力转回不可区分混淆. 如上所述, VBB 易用但对于通用电路并不存在,  $iO$  弱化了安全要求, 从而有了基于合理困难性假设的构造. 安全性弱化后  $iO$  是否还有着强大的威力? 如何去应用呢? 直观上: 混淆后的程序既可以保持功能性, 又能够在某种程度上隐藏常量. 常量皆程序. 在密码学中, 公钥和私钥均可以看做一段程序, 其中硬编码 (hardwired) 原本的公钥和私钥作为常量: 比如加密就是以明文和随机数为输入, 运行“公钥程序”, 输出密文; 解密就是以密文为输入, 运行“解密程序”, 输出明文. 混淆在密码学中的一类强大应用就是完成从 Minicrypt 到 Cryptomania 的穿越, 因为借助混淆, 可以在不泄漏秘密的情况下以公开的方式执行某个任务.

- 保持功能性  $\Rightarrow$  确保密码方案的功能性
- 在某种程度上隐藏常量 (对应需要保护的秘密)  $\Rightarrow$  确保密码方案的安全性

#### 注记 4.14

混淆的威力强大如魔法, 其力量的来源在于对底层密码组件的调用方式是非黑盒的 (non-black-box), 因此可以绕过黑盒意义下的不可能结果 (black-box impossibilities).



在  $iO$  提出后最初的一段时间, 应用只局限于属性加密 (ABE). 原因是应用  $iO$  设计密码方案并非易事, 需要解决的技术难题是精准的隐藏“部分信息”. 2014 年, Sahai 和 Waters [SW-STOC-2014] 创造性的发展了可穿孔编程技术 (puncture program technique), 以此给出了应用  $iO$  的范式, 展示了  $iO$  的巨大威力—结合单向函数和  $iO$  重构了几乎所有的密码组件, 包括公钥加密/密钥封装、可否认加密、数字签名、单向陷门函数、非交互式零知识证明、不经意传输等.

#### 4.4.2 受限伪随机函数

以下首先介绍可穿孔编程技术的关键密码组件—受限伪随机函数 (constrained PRF). 对于标准的伪随机函数  $F : K \times X \rightarrow Y$ , 密钥  $k$  不支持代理, 因此求值方式是“完全或无”(all-or-nothing):

- 知晓密钥  $k$  时, 可以对定义域中的所有点  $x \in X$  进行函数求值, 得到  $F_k(x)$ .
- 不知晓密钥  $k$  时,  $F_k(\cdot)$  是伪随机的.

三组科学家 [BW-ASIACRYPT-2013; KPTZ-CCS-2013; BGI-PKC-2014] 独立并行的提出了受限伪随机函数 (constrained PRF). 在受限伪随机函数中,  $k$  可以进一步代理得到受限密钥, 受限密钥可以对定义域中的部分输入

进行求值. 正式的定义如下:

#### 定义 4.15 (受限伪随机函数 (constrained PRF))

受限伪随机函数包含以下多项式时间算法:

- $\text{Setup}(1^\kappa)$ : 以安全参数为输入, 输出公开参数  $pp$ , 其中包含了函数  $F : K \times X \rightarrow Y$  的描述以及电路族  $\mathcal{C} = \{f : X \rightarrow \{0, 1\}\}$  的描述.
- $\text{KeyGen}(pp)$ : 随机采样密钥  $k \xleftarrow{R} K$ .
- $\text{Constrain}(k, f)$ : 以密钥  $k$  和  $c \in \mathcal{C}$  为输入, 输出受限密钥  $k_f$ .
- $\text{Eval}(k/k_f, x) = F_k(x)$ : 以密钥  $k$  或受限密钥  $k_c$  和  $x \in X$  为输入, 当第一输入为  $k$  时输出  $F_k$ , 当第一输入为  $k_c$  时, 如果  $f(x) = 1$  则输出  $F_k(x)$ , 否则输出  $\perp$ .



**安全性.** 受限伪随机函数要求对于没有被受限密钥和求值询问覆盖的输入, 其输出仍然是伪随机的. 定义敌手  $\mathcal{A} = (\mathcal{A})$  的优势函数如下:

$$\Pr \left[ \begin{array}{l} pp \leftarrow \text{Setup}(1^\lambda); \\ k \leftarrow \text{KeyGen}(pp); \\ \beta = \beta' : (x^*, \text{state}) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{eval}}, \mathcal{O}_{\text{constrain}}}(pp); \\ \beta \xleftarrow{R} \{0, 1\}, y_0^* = F_k(x^*), y_1^* \xleftarrow{R} Y; \\ \beta' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{eval}}, \mathcal{O}_{\text{constrain}}}(\text{state}, y_\beta^*); \end{array} \right] - \frac{1}{2}.$$

其中  $\mathcal{O}_{\text{eval}}$  表示求值预言机, 以  $x \in X$  为输入, 返回  $F_k(x)$ ;  $\mathcal{O}_{\text{constrain}}$  表示受限密钥预言机, 以  $c \in \mathcal{C}$  为输入, 输出  $k_c$ . 在安全试验过程中, 挑战者隐式的维护集合  $H$  以记录被已发起的受限密钥询问和求值询问覆盖的定义域中元素. 为了避免定义无意义,  $\mathcal{A}_1$  在第一阶段被禁止选择  $H$  中的元素作为挑战点,  $\mathcal{A}_2$  在第二阶段被禁止发起能够覆盖挑战点的受限密钥询问和求值询问. 如果任意的 PPT 敌手  $\mathcal{A}$  在上述游戏中的优势函数均为可忽略函数, 则称受限伪随机函数是安全的.

#### 注记 4.15

对任意  $\mathcal{C}$ , 均存在平凡的受限伪随机函数构造: 受限密钥生成函数通过枚举的方式生成受限密钥, 即  $k_f = \{y = F_k(x)\}_{f(x)=1}$ . 为了排除此类平凡的构造, 我们要求受限密钥是紧致的, 即对于  $\forall f \in \mathcal{C}$ , 均有  $|k_f| = \kappa^{O(1)}$ .



Sahai 和 Waters [SW-STOC-2014] 引入了受限伪随机函数的特例——可穿孔伪随机函数 (puncturable PRF), 正式定义如下:

#### 定义 4.16 (可穿孔伪随机函数 (puncturable PRF))

可穿孔伪随机函数包含以下多项式时间算法:

- $\text{Setup}(1^\kappa)$ : 以安全参数为输入, 输出公开参数  $pp$ , 其中包含了函数  $F : K \times X \rightarrow Y$  的描述和电路族  $\mathcal{C} = \{f_{x^*} : X \rightarrow \{0, 1\}\}_{x^* \in X}$  的描述.  $f_{x^*}(\cdot)$  的具体定义是  $f_{x^*}(x) = \neg x^* \stackrel{?}{=} x$ . 为了表述简洁, 以下在不引起混淆的情况下使用  $x^*$  表征  $f_{x^*}$ .
- $\text{KeyGen}(pp)$ : 随机采样密钥  $k \xleftarrow{R} K$ .
- $\text{Puncture}(k, x^*)$ : 以密钥  $k$  和  $x^* \in X$  为输入, 输出受限密钥  $k_{x^*}$ .
- $\text{Eval}(k/k_{x^*}, x)$ : 以密钥  $k$  或受限密钥  $k_{x^*}$  和  $x \in X$  为输入, 当第一输入为  $k$  时输出  $F_k$ , 当第一输入为  $k_{x^*}$  时, 如果  $x \neq x^*$  时输出  $F_k(x)$ , 否则输出  $\perp$ .



可穿孔伪随机函数的安全定义直觉是对于没有被受限密钥覆盖的输入, 其输出仍然是伪随机的. 存在以下两种等价定义.

选择伪随机性. 定义敌手  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  的优势函数如下:

$$\Pr_{\beta = \beta'} \left[ \begin{array}{l} (x^*, \text{state}) \leftarrow \mathcal{A}_1(\kappa); \\ pp \leftarrow \text{Setup}(1^\lambda); \\ k \leftarrow \text{KeyGen}(pp); \\ k_{x^*} \leftarrow \text{Puncture}(k, x^*); \\ \beta \stackrel{\text{R}}{\leftarrow} \{0, 1\}, y_0^* = F_k(x^*), y_1^* \stackrel{\text{R}}{\leftarrow} Y; \\ \beta' \leftarrow \mathcal{A}_2(\text{state}, k_{x^*}^*, y_\beta^*); \end{array} \right] - \frac{1}{2}.$$

如果任意的 PPT 敌手  $\mathcal{A}$  在上述游戏中的优势函数均为可忽略函数, 则称可穿孔伪随机函数是选择伪随机的.

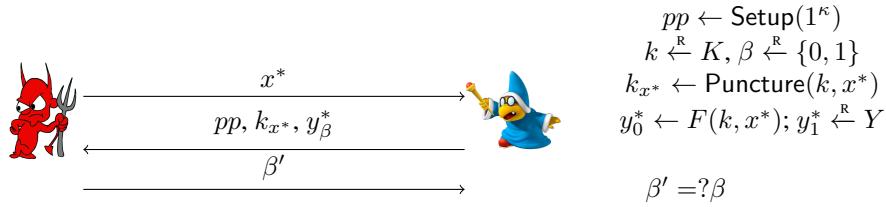


图 4.32: 可穿孔伪随机函数选择伪随机性示意图

弱伪随机性. 定义敌手  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  的优势函数如下:

$$\Pr_{\beta = \beta'} \left[ \begin{array}{l} pp \leftarrow \text{Setup}(1^\lambda); \\ k \leftarrow \text{KeyGen}(pp), x^* \stackrel{\text{R}}{\leftarrow} X; \\ k_{x^*} \leftarrow \text{Puncture}(k, x^*); \\ \beta \stackrel{\text{R}}{\leftarrow} \{0, 1\}, y_0^* = F_k(x^*), y_1^* \stackrel{\text{R}}{\leftarrow} Y; \\ \beta' \leftarrow \mathcal{A}(pp, x^*, k_x^*, y_\beta^*); \end{array} \right] - \frac{1}{2}.$$

如果任意的 PPT 敌手  $\mathcal{A}$  在上述游戏中的优势函数均为可忽略函数, 则称可穿孔伪随机函数是弱伪随机的.

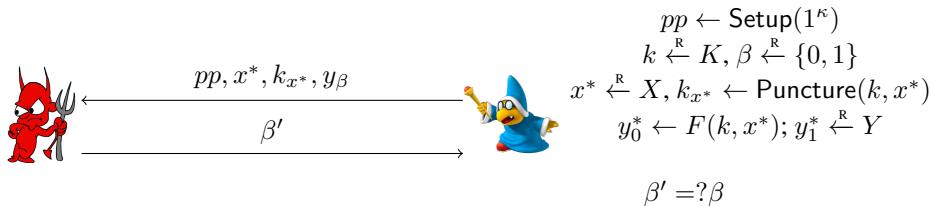


图 4.33: 可穿孔伪随机函数弱伪随机性示意图

文献 [Chen-ASIACRYPT-2018] 证明了可穿孔弱伪随机函数的两种安全定义等价.

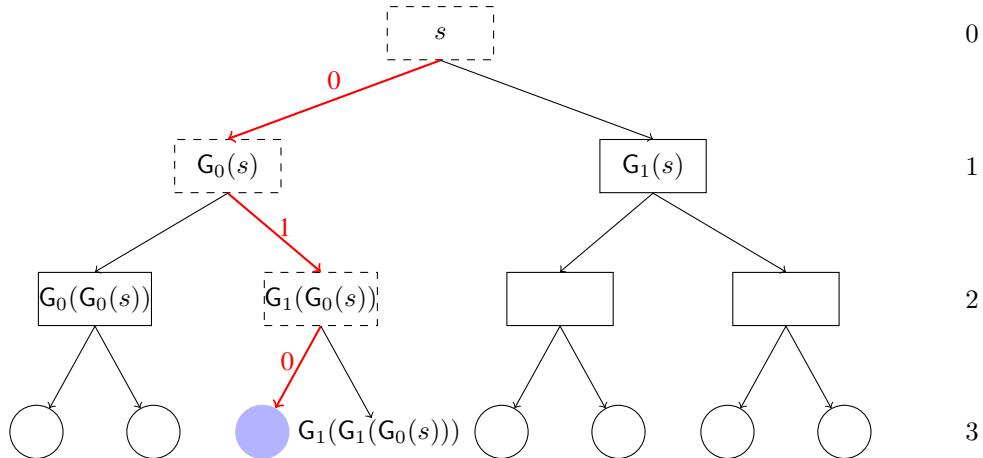
#### 注记 4.16

在定义方面, 可穿孔伪随机函数中的受限密钥  $k_{x^*}$  求值功能是“全除一”的, 是 ABO 类型密钥. 在安全性方面, 可穿孔伪随机函数要求弱伪随机性, 即对于挑战者随机选择的点  $x^*$ , 即使拥有  $k_x^*$ ,  $F(x^*)$  在敌手的视图中依然伪随机, 即:

$$(pp, x^*, k_{x^*}, \textcolor{red}{F}_k(x^*)) \approx_c (pp, x^*, k_{x^*}, y^*), \text{ 其中 } y^* \stackrel{\text{R}}{\leftarrow} Y$$

由于可穿孔伪随机函数的受限类型特殊, 因此安全定义相比一般的受限伪随机函数在形式上更加简单.

可穿孔伪随机函数可以通过 GGM 树形伪随机函数自然得出 [BW-ASIACRYPT-2013; KPTZ-CCS-2013; BGI-PKC-2014], 因此仍属于 Minicrypt.

图 4.34: GGM construction:  $k_{010} = \{G_0(G_0(s)), G_1(G_0(G_0(s))), G_1(s)\}$ 

#### 4.4.3 基于不可区分混淆的 KEM 构造

本章将逐步的展示如何基于  $i\mathcal{O}$  构造 KEM, 实现 Diffie-Hellman 当年的梦想.

**起点方案.** 首先将对称场景下基于 PRF 的 KEM 封装算法表达为程序的形式, 如下图所示.

Encaps	
Input: secret key $sk$ , randomness $x$ 1. output $c = x, k \leftarrow \text{PRF}(sk, c)$ .	

再对程序进行微调, 将  $sk$  由输入变为 hardwired 的常量, 如下图所示:

Encaps	
<u>Constants: secret key <math>sk</math></u> Input: randomness $x$ 1. output $c = x, k \leftarrow F(sk, c)$ .	

由于通用的 VBB 并不存在, 因此尝使用  $i\mathcal{O}$  对程序混淆, 将混淆后的结果作为公钥

$$pk \leftarrow i\mathcal{O}(\text{Encaps})$$

**技术困难 1.** 在将 KEM 的 IND-CCA 安全性归约到 PRF 的伪随机性时, 会遇到以下矛盾点:

- 在构造层面, 归约算法  $\mathcal{R}$  需要掌握  $sk$  以生成  $pk$
- 为了让归约有意义, 归约算法  $\mathcal{R}$  不能掌握私钥  $sk$

观察到 KEM 的 IND-CCA 安全仅要求随机挑战密文  $c^*$  封装的会话密钥是伪随机的, 因此消除矛盾点的核心想法是使用可穿孔伪随机函数替代标准伪随机函数, 在挑战密文  $c^*$  处穿孔:

- 生成  $sk_{c^*}$  得以对  $c^*$  外的所有点求值, 同时保持  $F_{sk}(c^*)$  的伪随机性.
- 利用  $sk_{c^*}$  替代  $sk$  构建程序并混淆生成公钥.

**Encaps**

Constants: secret key  $sk$

Input: randomness  $x$

1. output  $c = x, k \leftarrow F(sk, c)$ .

方案构造:  $pk \leftarrow i\mathcal{O}(\text{Encaps})$

**Encaps\***

Constants: secret key  $sk_{c^*}, c^*$

Input: randomness  $x$

1. output  $c = x, k \leftarrow F(sk_{c^*}, c)$ .

归约证明:  $pk \leftarrow i\mathcal{O}(\text{Encaps}^*)$

**技术困难 2.** 我们首先来分析归约证明中将要遇到的困难. 在模拟游戏中, 归约算法  $\mathcal{R}$  仅需要使用  $sk_{c^*}$  即可构建程序  $\text{Encaps}$ , 因此会话密钥  $k^* \leftarrow F(sk, c^*)$  的伪随机性可以归约到可穿孔伪随机函数的安全性上. 我们仍需证明敌手在真实游戏与模拟游戏中的视图不可区分. 在此过程中, 遇到的第一个障碍是由于在  $x^* := c^*$  处穿孔, 敌手可以通过观察程序在  $x^*$  的输出从而轻易区分真实游戏与模拟游戏:

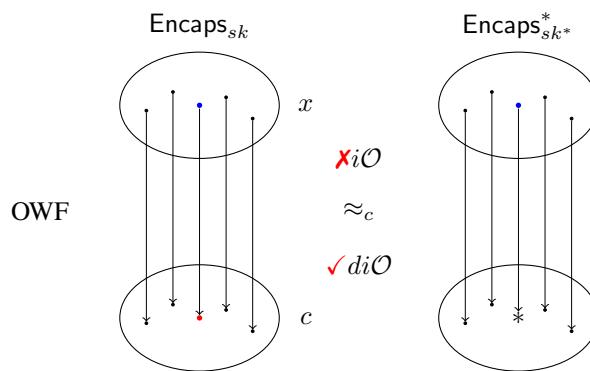
- 真实游戏:  $\text{Encaps}(x^*)$  返回  $k^*$  (已经不安全)
- 模拟游戏:  $\text{Encaps}^*(x^*)$  返回  $\perp$

以上设计不成立的根本原因是

- 密文的设定  $c = x \Rightarrow$  差异输入  $x^*$  将被挑战密文  $c^*$  直接暴露

为了隐藏差异输入, 初步的尝试为:

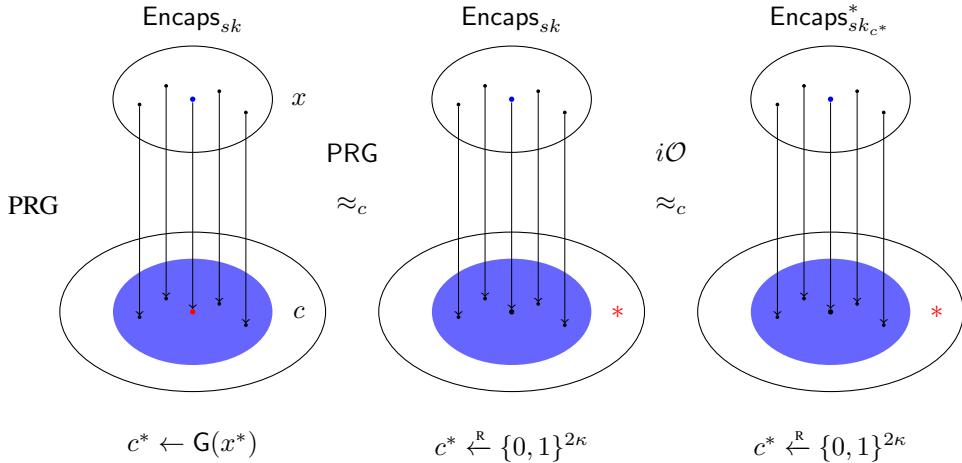
- $c = x \rightsquigarrow c = f(x)$ , 其中  $f$  是单向函数.



使用单向函数对挑战点进行隐藏并没有消除差异输入,  $\text{Encaps}_{sk}$  与  $\text{Encaps}_{sk_{c^*}}^*$  的输入输出行为存在不一致, 因此不满足  $i\mathcal{O}$  的应用条件, 需要使用更强的  $di\mathcal{O}$ .

消除差异输入的方法是将穿孔点  $c^*$  以敌手不可察觉的方式移到输入计算路径之外. 大致的技术路线是:

- 真实构造:  $c \leftarrow \text{OWF}(x) \rightsquigarrow c \leftarrow G(x)$ , 其中  $G$  是伪随机数发生器;
- 过渡游戏: 将  $c^* \leftarrow G(x^*)$  切换为  $c^* \xleftarrow{R} \{0, 1\}^{2\lambda}$ , 利用 PRG 的安全性保证切换不可察觉;
- 最终游戏: 利用  $sk_{c^*}$  替代  $sk$ , 利用  $i\mathcal{O}$  的安全性保证替代不可察觉.



综合以上, 最终的构造如下:

#### 构造 4.16 (基于不可区分混淆的构造 IND-CCA KEM)

构造所需的组件是:

- 不可区分混淆  $i\mathcal{O}$
- 伪随机数发生器 PRG  $G : \{0,1\}^\kappa \rightarrow \{0,1\}^{2\kappa}$
- 可穿孔伪随机函数 PPRF  $F : SK \times \{0,1\}^{2\kappa} \rightarrow Y$

构造 KEM 如下:

- $\text{Setup}(1^\kappa)$ : 运行  $pp \leftarrow \text{PPRF}.\text{Setup}(1^\kappa)$  生成公开参数, 私钥空间  $SK$  为可穿孔伪随机函数的密钥空间  $K$ , 密文空间  $C = \{0,1\}^{2\kappa}$ , 会话密钥空间  $K = Y$ .
- $\text{KeyGen}(pp)$ : 随机采样  $sk \xleftarrow{R} SK$ , 计算  $pk \leftarrow i\mathcal{O}(\text{Encaps})$ .
- $\text{Encaps}(pk; r)$ : 运行  $(c, k) \leftarrow pk(r)$ .
- $\text{Decaps}(sk, c)$ : 输出  $k \leftarrow F(sk, c)$ .



#### Encaps

Constants: PPRF key  $sk$

Input: randomness  $x \in \{0,1\}^\lambda$

1. output  $c \leftarrow G(x)$ ,  $k \leftarrow F(sk, c)$

构造 ?? 的正确性显然, 安全性由以下定理保证.

#### 定理 4.15

如果  $F$  是安全的可穿孔伪随机函数、 $G$  是安全的伪随机数发生器、 $i\mathcal{O}$  是不可区分混淆, 则构造 ?? 满足 IND-CCA 安全性.



**证明** 以下通过游戏序列完成定理证明.

Game<sub>0</sub>: 对应真实游戏

- 初始化:  $\mathcal{CH}$  运行  $\text{PPRF}.\text{Setup}(1^\kappa)$  生成公开参数, 随机采样  $sk \xleftarrow{R} SK$ , 生成公钥  $pk \leftarrow i\mathcal{O}(\text{Encaps})$
- 挑战阶段:  $\mathcal{CH}$  随机采样  $x^* \xleftarrow{R} \{0,1\}^\kappa$ , 计算  $c^* \leftarrow G(x^*)$ ,  $k_0^* \leftarrow F(sk, c^*)$ , 随机采样  $k_1^* \xleftarrow{R} K$ ,  $\beta \leftarrow \{0,1\}$ , 将  $(c^*, k_\beta^*)$  发送给  $\mathcal{A}$  作为挑战.
- 解封装询问:  $\mathcal{A}$  发起询问  $c \in C$ ,  $\mathcal{CH}$  返回  $k \leftarrow F(sk, c)$ .
- 猜测:  $\mathcal{A}$  输出对  $\beta$  的猜测  $\beta'$ , 攻击成功当且仅当  $\beta = \beta'$ .

$\text{Game}_1$ : 与  $\text{Game}_0$  的区别是  $\mathcal{CH}$  在挑战阶段随机采样  $c^* \xleftarrow{R} \{0,1\}^{2\kappa}$  而非计算  $c^* \leftarrow G(x^*)$ . PRG 的伪随机性保证了:

$$\text{Game}_0 \approx_c \text{Game}_1$$

$\text{Game}_2$ : 与  $\text{Game}_0$  的区别是  $\mathcal{CH}$  将  $c^*$  的生成从挑战阶段提前到初始化阶段 (为后续使用可穿孔伪随机函数做准备). 该变化完全隐藏于对手, 因此有:

$$\text{Game}_1 \equiv \text{Game}_2$$

$\text{Game}_3$ :  $\mathcal{CH}$  在初始化阶段计算  $pk \leftarrow i\mathcal{O}(\text{Encap}^*)$  而非之前的  $pk \leftarrow i\mathcal{O}(\text{Encap})$ ; 在应答解封装询问时, 使用  $sk_{c^*}$  计算并返回  $k \leftarrow F(sk_{c^*}, c)$ , 代替之前使用  $k$  计算并返回  $k \leftarrow F(sk, c)$ .

$\text{Encaps}^*$

Constants: PPRF punctured key  $sk_{c^*}, c^*$   
Input: randomness  $x \in \{0,1\}^\lambda$   
1. output  $c \leftarrow G(x), k \leftarrow F(sk_{c^*}, c)$ .

- 由于  $\Pr[c^* \in \text{Img}(G)] = 1/2^\kappa$ , 因此  $c^*$  落在  $G$  的像集中的概率可忽略, 故而穿孔导致程序输入输出行为差异的概率可忽略, 即  $\Pr[\text{Encaps}_{sk} \equiv \text{Encaps}_{sk_{c^*}}] = 1 - 1/2^\kappa$ .  $i\mathcal{O}$  的安全性保证了公钥的分布计算不可区分

$$i\mathcal{O}(\text{Encaps}) \approx_c i\mathcal{O}(\text{Encaps}^*)$$

- 对于所有合法的解密询问  $c \neq c^*$ , 可穿孔伪随机函数的正确性保证了  $F(sk, c) = F(sk_{c^*}, c)$ .  
因此, 我们有

$$\text{Game}_2 \approx_c \text{Game}_3$$

$\text{Game}_4$ :  $\mathcal{CH}$  随机采样  $k_0^* \xleftarrow{R} K$  代替上一游戏的  $k_0^* \leftarrow F(sk, c^*)$ . 可穿孔伪随机函数的弱伪随机性保证了

$$\text{Game}_3 \approx_c \text{Game}_4$$

在  $\text{Game}_4$ ,  $k_0^*$  和  $k_1^*$  均从  $K$  中均匀随机采样, 因此即使  $\mathcal{A}$  拥有无穷的计算能力, 其在  $\text{Game}_4$  中的优势也是 0.

综合以上, 定理得证! □

#### 注记 4.17

构造 ?? 中的 KEM 也具备可穿孔性质. 该构造充分展示了  $i\mathcal{O}$  的魔力——使得在不暴露秘密的情况下可以公开执行“内嵌秘密值”的程序:

- 将私钥组件编译为公钥组件

## 4.5 可公开求值伪随机函数类

明修栈道, 暗渡陈仓.

—汉·司马迁《史记·高祖本纪》

前面的章节已经展示了若干种构造公钥加密的通用方法, 包括单向陷门函数、哈希证明系统、可提取哈希证明系统以及不可区分混淆结合可穿孔伪随机函数. 这些通用构造阐释了绝大多数公钥加密方案, 然而令人颇感意外的是, 它们并无法阐释最经典的 ElGamal PKE [ElGamal-IEEE-IT-1985] 和 Goldwasser-Micali PKE [GM-JCSS-1984]. 另一方面, 伪随机函数是密码学的核心基本组件之一, 应用范围极其广泛, 特别的, 伪随机函数蕴含了简洁优雅的、也是目前唯一的 IND-CPA SKE 通用构造.

$$\text{Enc}(sk, m; r) \rightarrow (r, F(sk, x) \oplus m)$$

然而伪随机函数属于 Minicrypt, 因此在黑盒意义下无法蕴含 PKE.

以上的现象促使我们考虑如下的问题:

### 思考 4.1

是否存在新型的伪随机函数能够让 PRF-based SKE 延拓到公钥场景? 新型的伪随机函数是否能蕴含统一上述的不同构造, 并阐释经典 PKE 方案的设计机理?

我们首先分析基于 PRF 构造 PKE 的技术难点:

- 密文必须可以公开计算: 显然, PRF-based SKE 的构造正是因为这个原因无法延拓到公钥加密场景中

$$(x, F(sk, x) \oplus m)$$

因为  $F$  的伪随机性意味着其不可能公开求值.

解决上述问题的关键在于探求伪随机性 (pseudorandomness) 和可公开求值性 (public evaluability) 是否能够共存. 标准的伪随机函数处处伪随机 (universal pseudorandom), 即对于定义域中任意  $x \in X$ , PPT 敌手  $\mathcal{A}$  都无法区分  $F_k(x)$  和随机值.

- 观察 1: 构造 IND-CPA KEM 仅需要弱伪随机性 (weak pseudorandomness), 即对于挑战者随机选择的挑战输入, 其 PRF 值是伪随机的
  - 观察 2: 如果掌握输入  $x$  的某些辅助信息  $aux$  (比如采样  $x$  的随机数), 是有可能在不使用  $sk$  的情形下对  $F_{sk}(x)$  公开求值. 如果  $aux$  在平均意义上是难以抽取的, 则公开求值性与弱伪随机性不冲突.
- 综合以上, 在 KEM 中由发送方生成  $x$ , 因此其知晓  $aux$  信息, 从而以下两点成为可能:
- 功能性方面: 发送方可以借助  $aux$  对  $F_{sk}(x)$  公开求值从而生成密文.
  - 安全性方面:  $F_{sk}(x)$  在  $\mathcal{A}$  的视图中仍然伪随机.

### 4.5.1 定义与安全性

正是基于上面的思考, 陈等 [Chen-SCN-2014] 提出了可公开求值伪随机函数 (PEPRF, Publicly Evaluable PRFs). PEPRF 考虑了定义域  $X$  包含  $\mathcal{NP}$  语言  $L$  的情形, 使用私钥可以对全域求值, 而使用公钥和证据可以对语言  $L$  内的元素求值. 在安全性上, PEPRF 要求函数在语言  $L$  上弱伪随机.

#### 定义 4.17 (可公开求值伪随机函数)

- $\text{Setup}(1^\kappa)$ : 以安全参数  $\kappa$  为输入, 输出公开参数  $pp = (F, PK, SK, X, L, W, Y)$ , 其中  $F : SK \times X \rightarrow Y \cup \perp$  是由  $SK$  索引的一族函数,  $L \subseteq X$  是由困难关系  $R_L$  定义的  $\mathcal{NP}$  语言, 其中  $W$  是相应的证据集合.  $R_L$  是高效可采样的, 存在 PPT 算法 SampR 以随机数  $r$  为输入, 输出实例证据元组  $(x, w) \in R_L$ .

- $\text{KeyGen}(pp)$ : 以公开参数  $pp$  为输入, 输出公钥  $pk$  和私钥  $sk$ .
- $\text{PrivEval}(sk, x)$ : 以私钥  $sk$  和元素  $x \in X$  为输入, 输出  $y \in Y \cup \perp$ .
- $\text{PubEval}(pk, x, w)$ : 以公钥  $pk$ 、实例  $x \in L$  以及相应的证据  $w \in W$  为输入, 输出  $y \in Y$ .



#### 注记 4.18

在有些场景中有必要将单一语言  $L$  泛化为由  $PK$  索引的一族语言  $\{L_{pk}\}_{pk \in PK}$ . 相应的, 采样算法  $\text{SampRel}$  将以  $pk$  为额外输入, 随机采样  $(x, w) \in R_{L_{pk}}$ .



**正确性.** 对于任意  $pp \leftarrow \text{Setup}(1^\lambda)$  和  $(pk, sk) \leftarrow \text{KeyGen}(pp)$ , 我们有:

$$\begin{aligned} \forall x \in X : \quad F_{sk}(x) &= \text{PrivEval}(sk, x) \\ \forall x \in L \text{ 以及证据 } w : \quad F_{sk}(x) &= \text{PubEval}(pk, x, w) \end{aligned}$$

**(自适应) 弱伪随机性.** 定义敌手  $\mathcal{A}$  的优势函数如下:

$$\Pr \left[ \beta = \beta' : \begin{array}{l} pp \leftarrow \text{Setup}(1^\kappa); \\ (pk, sk) \leftarrow \text{KeyGen}(pp); \\ r^* \xleftarrow{\text{R}} R, (x^*, w^*) \leftarrow \text{SampR}(r^*); \\ y_0^* \leftarrow F_{sk}(x^*), y_1^* \leftarrow Y; \\ \beta \leftarrow \{0, 1\}; \\ \beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{eval}}(\cdot)}(pp, pk, x^*, y_b^*); \end{array} \right] - \frac{1}{2}.$$

其中  $\mathcal{O}_{\text{eval}}$  表示求值预言机, 以  $x \neq x^* \in X$  为输入, 返回  $F_{sk}(x)$ . 如果任意 PPT 敌手  $\mathcal{A}$  在上述游戏中的优势函数均为可忽略函数, 则称可公开求值伪随机函数是弱伪随机的. 如果敌手在上述游戏中可以访问  $\mathcal{O}_{\text{eval}}$  预言机, 则称可公开求值伪随机函数是自适应弱伪随机的.

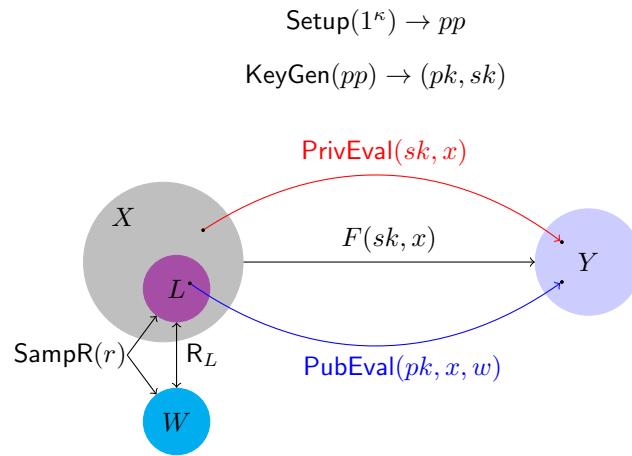


图 4.35: PEPRF 示意图

#### 注记 4.19

在 PEPRF 中, 私钥用于秘密求值, 公钥则可在知晓相应证据时对语言内的元素进行公开求值. 密钥成对出现这一点对于 PEPRF 是自然的, 因为 PEPRF 是作为 PRF 在 Cryptomania 中的对应引入的. 另一方面, 标准的 PRF 也总是可以设置公钥用于发布与私钥相关联但可公开的信息, 例如在基于 DDH 假设的 Naor-Reingold PRF [NR-JACM-2004] 中,  $F_{\vec{a}}(x) = (g^{a_0})^{\prod_{x_i=1} a_i}$ , 其中  $\vec{a} = (a_0, a_1, \dots, a_n) \in \mathbb{Z}_p^n$  是私钥,

	PRF	PEPRF
带密钥函数	✓	✓
可公开求值	$\forall x \in X \ x$	$x \in L \ \checkmark$
安全性	$\forall x \in X$ pseudorandom	$x \xleftarrow{R} L$ weak pseudorandom

表 4.1: PRF 与 PEPRF 的比较

$\{g^{a_i}\}_{1 \leq i \leq n}$  则可发布为公钥. 如果没有信息可公开, 可设定  $pk = \{\perp\}$ . 如此可保持 PRF 与 PEPRF 的语法定义保持一致.

为什么 PEPRF 只定义了弱伪随机性呢? 这是因为在公开求值算法 PubEval 存在的前提下, 这是可达的最强安全性.

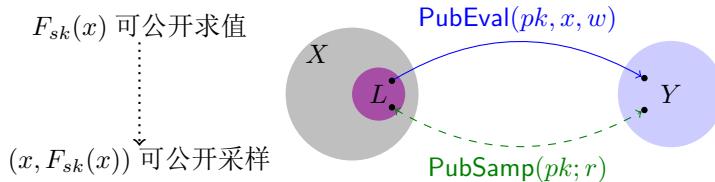
为了加深对概念的理解, 表 ?? 对比分析 PRF 与 PEPRF 的异同. 正是由于上述区别, 我们可以基于 PEPRF 构造 KEM.

PEPRF 的定义可以进一步泛化以包容更多实例化构造.

#### 定义 4.18 (可公开采样伪随机函数 (PSPRF, Publicly Sampleable PRF))

PSPRF 将 PEPRF 的可公开求值功能放宽为可公开采样功能, 即 PubEval 算法由以下的 PPT 随机采样算法替代:

- $\text{PubSamp}(pk; r) \rightarrow (x, y) \in L \times Y$  s.t.  $y = F_{sk}(x)$



显然, 可以综合关系采样算法和函数公开求值算法构造公开采样算法, 因此 PEPRF 蕴含 PSPRF:

- $\text{PubSamp}(pk; r)$ : 运行  $(x, w) \leftarrow \text{SampRel}(r)$ , 输出  $(x, \text{PEPRF.PubEval}(pk, x, w))$ .

### 4.5.2 基于 PEPRF 的 KEM 构造

本章将展示如何基于 PEPRF 构造 KEM.

#### 构造 4.17 (基于 PEPRF 的 KEM 构造)

构造思路: 随机采样语言中的元素作为密文, 计算其函数值作为会话密钥  $k$ .

起点:  $\text{PEPRF } F : SK \times X \rightarrow Y \cup \perp$ , 其中  $L \subseteq X$  是定义在  $X$  上的  $\mathcal{NP}$  语言.

构造如下:

- $\text{Setup}(1^\kappa)$ : 运行  $pp \leftarrow \text{PEPRF.Setup}(1^\kappa)$ , 其中密文空间  $C = X$ , 会话密钥空间  $K = Y$ .
- $\text{KeyGen}(pp)$ : 运行  $(pk, sk) \leftarrow \text{PEPRF.KeyGen}(pp)$ .
- $\text{Encaps}(pk; r)$ : 随机采样  $(x, w) \leftarrow \text{SampR}(r)$ , 输出  $c = x$  作为密文, 通过  $\text{PEPRF.PubEval}(pk, x, w)$  公开计算  $k \leftarrow F_{sk}(x)$  作为会话密钥.
- $\text{Decaps}(sk, c)$ : 通过运行  $\text{PEPRF.PrivEval}(sk, c)$  秘密计算  $k \leftarrow F_{sk}(x)$  恢复会话密钥.

构造 ?? 的正确性由 PEPRF 的正确性保证, 安全性由以下定理保证.

**定理 4.16**

如果 PEPRF 是弱伪随机的, 则构造 ?? 是 IND-CPA 安全的; 如果 PEPRF 是自适应弱伪随机的, 则构造 ?? 是 IND-CCA 安全的.



**证明** IND-CPA 安全性的归约是显然的, 建立 IND-CCA 安全性的关键是令归约算法利用  $\mathcal{O}_{\text{eval}}$  模拟  $\mathcal{O}_{\text{decaps}}$ .  $\square$

**注记 4.20**

在上述的 KEM 构造中, 可以将 PEPRF 弱化为 PSPRF.



### 4.5.3 PEPRF 的构造

天下同归而殊途, 一致而百虑.

—《周易·系辞下》

本章节展示如何基于具体的困难假设和(半)通用的密码组件构造 PEPRF.

#### 4.5.3.1 基于 DDH 假设的 PEPRF

图 ??展示了基于 DDH 假设的 PEPRF 构造, 其中可公开求值功能利用了 DH 函数的可交换性, 弱伪随机性建立在 DDH 假设之上. 将实例化代入构造 ??中, 得到的正是经典的 ElGamal PKE 方案 [ElGamal-IEEE-IT-1985].

**构造 4.18 (基于 DDH 假设的 PEPRF)**

- $\text{Setup}(1^\kappa)$ : 运行  $(\mathbb{G}, p, g) \leftarrow \text{GroupGen}(1^\kappa)$ , 生成公开参数  $pp = (F, PK, SK, X, L, W, Y)$ , 其中  $X = Y = PK = L = \mathbb{G}$ ,  $SK = W = \mathbb{Z}_p$ ,  $F : SK \times X \rightarrow Y$  定义为  $F_{sk}(x) = x^{sk}$ , 语言  $L = \{x : \exists w \in W \text{ s.t. } x = g^w\}$ , 相应的采样算法  $\text{SampRel}$  以随机数  $r$  为输入, 随机采样证据  $w \xleftarrow{R} \mathbb{Z}_p$ , 计算实例  $x = g^w$ .
- $\text{KeyGen}(pp)$ : 随机采样私钥  $sk \xleftarrow{R} \mathbb{Z}_p$ , 计算公钥  $pk = g^{sk}$ .
- $\text{PrivEval}(sk, x)$ : 输出  $x^{sk}$ .
- $\text{PubEval}(pk, x, w)$ : 输出  $pk^w$ .

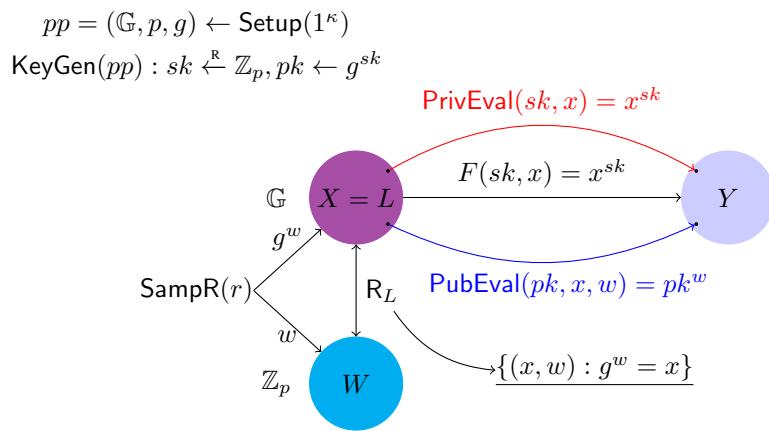


图 4.36: 基于 DDH 假设的 PEPRF

#### 4.5.3.2 基于 QR 假设的 PEPRF

图 ??展示了基于 QR 假设的 PEPRF 构造, 其中可公开求值功能利用了语言  $L$  的 OR 型定义, 弱伪随机性建立在 QR 假设之上. 将实例化代入构造 ??中, 得到的正是 Goldwasser-Micali PKE 方案 [GM-JCSS-1984] 内蕴的

KEM.

#### 构造 4.19 (QR-based-PEPRF)

- $\text{Setup}(1^\kappa)$ : 输出  $pp = \kappa$ .
- $\text{KeyGen}(pp)$ : 运行  $(N, p, q) \leftarrow \text{GenModulus}(1^\kappa)$ , 选取  $z \in \mathbb{QNR}_N^{+1}$ , 输出公钥  $pk = (N, z)$  和私钥  $sk = (p, q)$ .  $pk$  还包含了以下信息: 函数定义域  $X = \mathbb{Z}_N^*$ , 值域  $Y = \{0, 1\}$ , 证据集合  $W = \mathbb{Z}_N^*$ , 语言  $L_{pk} = \{x : \exists w \in W \text{ s.t. } x = w^2 \pmod{N} \vee x = zw^2 \pmod{N}\}$  ( $\mathbb{Z}_N^*$  中 Jacobi 符号为 +1 的元素). 采样算法  $\text{SampRel}$  以公钥  $pk$  和随机数  $r$  为输入, 随机采样  $w \xleftarrow{R} \mathbb{Z}_p$ , 随机生成实例  $x = w^2 \pmod{N}$  或  $x = zw^2 \pmod{N}$ .
- $\text{PrivEval}(sk, x)$ : 如果  $x \in \mathbb{QNR}_N$  则输出 1, 如果  $x \in \mathbb{QNR}_N^{+1}$  则输出 0.
- $\text{PubEval}(pk, x, w)$ : 如果  $x = w^2 \pmod{N}$  则输出 1, 如果  $x = zw^2 \pmod{N}$  则输出 0.



$$\begin{aligned} pp &= \kappa \leftarrow \text{Setup}(1^\kappa) \\ \text{KeyGen}(pp) &: sk \leftarrow (p, q), pk \leftarrow (N, z \xleftarrow{R} \mathbb{QNR}_N^{+1}) \end{aligned}$$

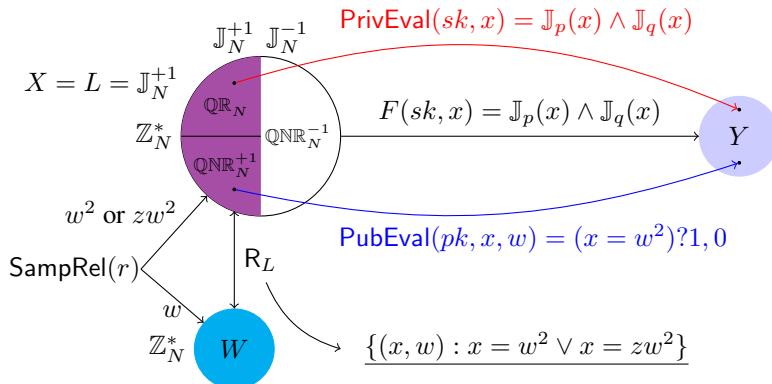


图 4.37: 基于 QR 假设的 PEPRF 构造

#### 4.5.3.3 基于 TDF 的 PEPRF

通过扭转单射 TDF, 可以构造 PEPRF 如下.

#### 构造 4.20 (基于 TDF 的 PEPRF 构造)

- $\text{Setup}(1^\kappa)$ : 运行  $pp = (G, EK, TD, S, U) \leftarrow \text{TDF.Setup}(\lambda)$ , 令  $\text{hc} : S \rightarrow K$  是相应的 hardcore function; 生成 PEPRF 的公开参数  $pp = (F, PK, SK, X, L, W, Y)$ , 其中  $PK = EK$ ,  $SK = TD$ ,  $Y = K$ ,  $X = U$ ,  $W = S$ ,  $F_{sk}(x) = \text{hc}(G_{td}^{-1}(x))$ . 算法 TDF.Eval 自然定义了一族定义在  $X$  上的  $\mathcal{NP}$  语言  $L = \{L_{pk}\}_{pk \in PK}$ , 其中  $L_{pk} = \{x : \exists w \in W \text{ s.t. } x = \text{TDF.Eval}(pk, w)\}$ . 采样算法  $\text{SampRel}$  以随机数  $r$  为输入, 首先随机采样定义域中元素  $s \leftarrow \text{SampDom}(r)$ , 再计算  $u \leftarrow \text{TDF.Eval}(pk, s)$ , 输出实例  $x = u$  和证据  $w = s$ .
- $\text{KeyGen}(pp)$ : 运行  $(ek, td) \leftarrow \text{TDF.KeyGen}(pp)$ , 输出  $pk = ek$  和  $sk = td$ .
- $\text{PrivEval}(sk, x)$ : 以私钥  $sk$  和元素  $x \in X$  为输入, 输出  $y \leftarrow F_{sk}(x) = \text{hc}(\text{TDF.TdInv}(sk, x))$ .
- $\text{PubEval}(pk, x, w)$ : 以公钥  $pk$ 、实例  $x \in L_{pk}$  和证据  $w$  为输入, 输出  $y \leftarrow \text{hc}(w)$ .



构造 ?? 的正确性由陷门单向函数的正确性和单射性保证, 安全性由如下定理保证.

#### 定理 4.17

如果起点 TDF 是 (自适应) 单向的, 那么构造 ?? 中的 PEPRF 是 (自适应) 弱伪随机的.



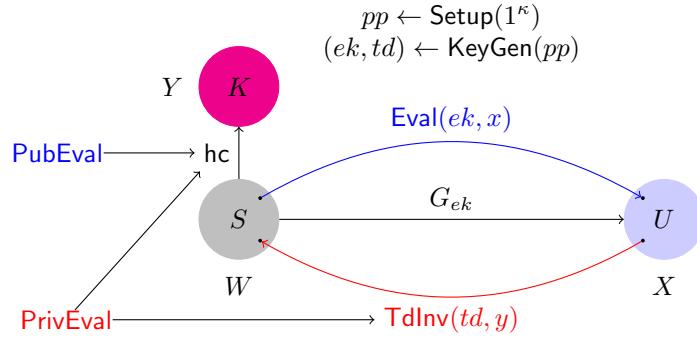


图 4.38: 基于 TDF 的 PEPRF 构造

	HPS	PEPRF
投射性	✓	not necessary
$L$ 与 $X$ 的关系	$L \subset X$	$L \subseteq X$
弱伪随机性	$x \xleftarrow{R} X \setminus L$	$x \xleftarrow{R} L$

表 4.2: HPS 与 PEPRF 的不同

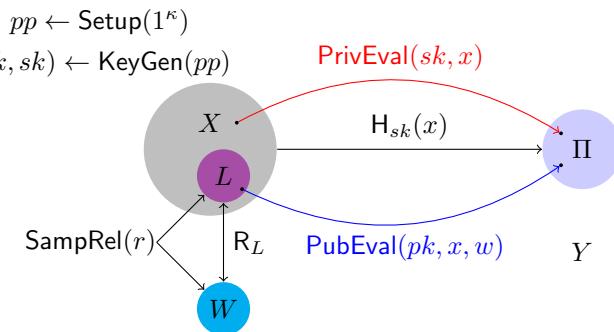
#### 4.5.3.4 基于 HPS 的 PEPRF 构造

本章节展示如何基于哈希证明系统构造具有不同安全性质的可公开求值伪随机函数.

首先展示如何基于平滑 HPS 构造弱伪随机的 PEPRF.

##### 构造 4.21 (基于平滑 HPS 的 PEPRF 构造)

- $\text{Setup}(1^\kappa)$ : 运行  $\text{HPS}.\text{Setup}(1^\lambda)$  生成 HPS 的公开参数  $pp = (\mathsf{H}, PK, SK, X, L, W, \Pi, \alpha)$ , 输入 PEPRF 的公开参数  $pp = (F, PK, SK, X, L, W, Y)$ , 其中  $F = \mathsf{H}$ ,  $Y = \Pi$ .
- $\text{KeyGen}(pp)$ : 运行  $(pk, sk) \leftarrow \text{HPS}.\text{KeyGen}(pp)$  生成密钥对.
- $\text{PrivEval}(sk, x)$ : 以私钥  $sk$  和元素  $x \in X$  为输入, 计算  $y \leftarrow \text{HPS}.\text{PrivEval}(sk, x)$ .
- $\text{PubEval}(pk, x, w)$ : 以公钥  $pk$ 、语言中的元素  $x \in L$  和相应的证据  $w \in W$  为输入, 计算  $y \leftarrow \text{HPS}.\text{PubEval}(pk, x, w)$ .



构造 ?? 的正确性由平滑 HPS 的正确性保证, 安全性由如下定理保证:

##### 定理 4.18

基于  $L \subset X$  上的 SMP 假设, 构造 ?? 中的 PEPRF 满足弱伪随机性.

PEPRF 与 HPS 在语法上非常相似, 但存在以下微妙的不同, 如表 ?? 所示:

下面展示如何基于平滑和一致 HPS 构造自适应伪随机的 PEPRF.

**构造 4.22 (基于平滑和一致 HPS 的 PEPRF 构造)**

构造组件: 针对同一语言  $\tilde{L} \subset \tilde{X}$  的 smooth HPS<sub>1</sub> 和 2-universal HPS<sub>2</sub>:

构造如下:

- $\text{Setup}(1^\kappa)$ : 运行  $pp_1 = (\mathsf{H}_1, PK_1, SK_1, \tilde{X}, \tilde{L}, W, \Pi_1, \alpha_1) \leftarrow \text{HPS}_1.\text{Setup}(1^\kappa)$  生成 smooth HPS 的公开参数, 运行  $pp_2 = (\mathsf{H}_2, PK_2, SK_2, \tilde{X}, \tilde{L}, \tilde{W}, \Pi_2, \alpha_2) \leftarrow \text{HPS}_2.\text{Setup}(1^\kappa)$  生成 2-universal HPS 的公开参数, 基于  $pp_1$  和  $pp_2$  生成 PEPRF 的公开参数  $pp = (F, PK, SK, X, L, W, Y)$ , 其中  $X = \tilde{X} \times \Pi_2$ ,  $Y = \Pi_1 \cup \perp$ ,  $PK = PK_1 \times PK_2$ ,  $L = \{L_{pk}\}_{pk \in PK}$  定义在  $X = \tilde{X} \times \Pi_2$  上, 其中  $L_{pk} = \{x = (\tilde{x}, \pi_2) : \exists w \in W \text{ s.t. } \tilde{x} \in \tilde{L} \wedge \pi_2 = \text{HPS}_2.\text{PubEval}(pk_2, \tilde{x}, w)\}$ , 相应的采样算法 SampRel 以公钥  $pk = (pk_1, pk_2)$  和随机数  $r$  为输入, 首先随机采样语言  $\tilde{L}$  的随机实例证据元组  $(\tilde{x}, w)$ , 计算  $\pi_2 \leftarrow \text{HPS}_2.\text{PubEval}(pk_2, \tilde{x}, w)$ , 输出语言  $L$  的实例  $x = (\tilde{x}, \pi_2)$  和证据  $w = \tilde{w}$ . 不失一般性, 令  $pp$  包含  $pp_1$  和  $pp_2$  中的所有信息.
- $\text{KeyGen}(pp)$ : 从  $pp$  中解析出  $pp_1$  和  $pp_2$ , 运行  $(pk_1, sk_1) \leftarrow \text{HPS}_1.\text{KeyGen}(pp_1)$  和  $(pk_2, sk) \leftarrow \text{HPS}_2.\text{KeyGen}(pp_2)$ , 输出  $pk = (pk_1, pk_2)$  和  $sk = (sk_1, sk_2)$ .
- $\text{PrivEval}(sk, x)$ : 以私钥  $sk = (sk_1, sk_2)$  和  $x = (\tilde{x}, \pi_2)$  为输入, 如果  $\pi_2 = \text{HPS}_2.\text{PrivEval}(sk_2, \tilde{x})$  则返回  $\perp$  否则返回  $y \leftarrow \text{HPS}_1.\text{PrivEval}(sk_1, \tilde{x})$ . 该算法定义了  $F : SK \times X \rightarrow Y \cup \perp$ .
- $\text{PubEval}(pk, x, w)$ : 以公钥  $pk = (pk_1, pk_2)$ 、元素  $x = (\tilde{x}, \pi_2) \in L_{pk}$  以及证据  $w$  为输入, 输出  $y \leftarrow \text{HPS}_1.\text{PubEval}(pk_1, \tilde{x}, w)$ .

**定理 4.19**

基于  $\tilde{L} \subset \tilde{X}$  上的 SMP 假设, 构造 ?? 中的 PEPRF 是自适应弱伪随机的.

**注记 4.21**

构造 ?? 相对直接, 构造 ?? 稍显复杂, 其中蕴含的设计思想与基于哈希证明系统构造 CCA-secure PKE 相似: 使用“弱”HPS 封装随机会话密钥, 使用“强”HPS 生成证明以杜绝“危险”解密询问.

**4.5.3.5 基于 EHPS 的 PEPRF 构造**

本章节展示如何基于 (ABO-)EHPS 构造 PEPRF.

**构造 4.23 (基于 (ABO-)EHPS 的 PEPRF 构造)**

- $\text{Setup}(1^\kappa)$ : 运行  $pp = (\mathsf{H}, PK, SK, \tilde{L}, \tilde{W}, \Pi) \leftarrow \text{EHPS}.\text{Setup}(1^\kappa)$  生成 EHPS 的公开参数, 令  $hc : \tilde{W} \rightarrow Z$  是单向关系  $R_{\tilde{L}}$  的 hardcore function; 生成 PEPRF 的公开参数  $pp = (F, PK, SK, X, L, W, Y)$ , 其中  $X = \tilde{L} \times \Pi$ ,  $Y = Z$ ,  $W = R$ ,  $L = \{L_{pk}\}_{pk \in PK}$  定义在  $X = \tilde{L} \times \Pi$  上, 其中  $L_{pk} = \{x = (\tilde{x}, \pi) : \exists w \in W \text{ s.t. } \tilde{x} = \text{SamplIns}(w) \wedge \pi = \text{EHPS}.\text{PubEval}(pk, \tilde{x}, w)\}$ , 相应的采样算法以公钥  $pk$  和随机数  $w$  为输入, 首先以  $w$  作为证据生成实例  $\tilde{x} \xleftarrow{R} \tilde{L}$ , 再计算  $\pi \leftarrow \text{EHPS}.\text{Ext}(pk, \tilde{x}, w)$ , 输出实例  $x = (\tilde{x}, \pi)$  和证据  $w$ .  $F_{sk}(x) := hc(\text{EHPS}.\text{Ext}(sk, x))$ .
- $\text{KeyGen}(pp)$ : 运行  $(pk, sk) \leftarrow \text{EHPS}.\text{KeyGen}(pp)$  生成密钥对.
- $\text{PrivEval}(sk, x)$ : 以私钥  $sk$  和  $x \in X$  为输入, 将  $x$  解析为  $(\tilde{x}, \pi)$ , 计算  $\tilde{w} \leftarrow \text{EHPS}.\text{Ext}(sk, \tilde{x}, \pi)$ , 输出  $y \leftarrow hc(\tilde{w})$ .
- $\text{PubEval}(pk, x, w)$ : 以公钥  $pk$ 、 $x \in L_{pk}$  以及相应的证据  $w$  为输入, 计算  $\tilde{w} \leftarrow \text{SampWit}(w)$ , 输出  $y \leftarrow hc(\tilde{w})$ .

**定理 4.20**

如果  $R_{\tilde{L}}$  是单向的, 基于 (ABO-)EHPS 的 PEPRF 是 (自适应) 弱伪随机的.



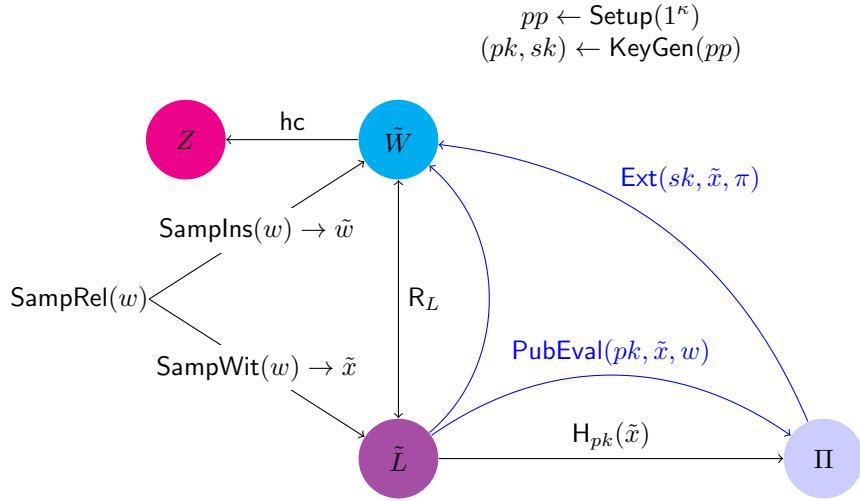


图 4.39: 基于 EHPS 的 PEPRF 构造

#### 4.5.3.6 基于 $i\mathcal{O}$ 的 PEPRF 构造

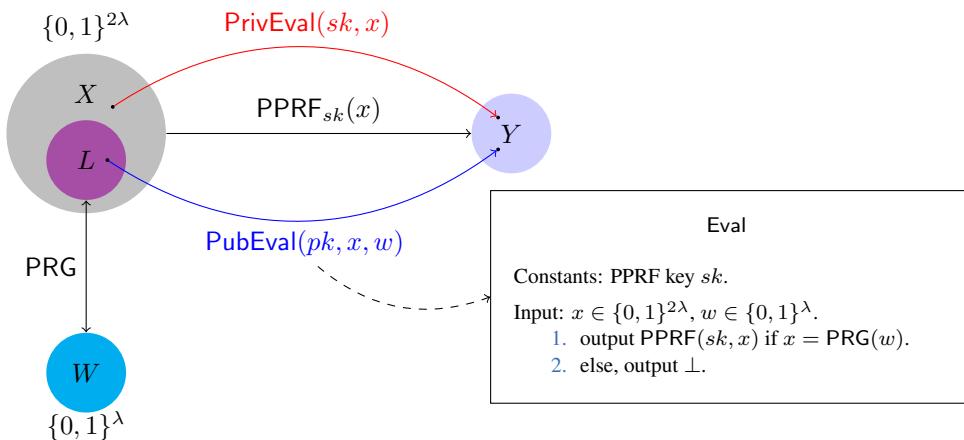
本章节展示如何基于不可区分程序混淆  $i\mathcal{O}$ 、伪随机数发生器和可穿孔伪随机函数构造 PEPRF.

##### 构造 4.24 (基于 $i\mathcal{O}$ 和 PPRF 的 PEPRF 构造)

构造组件: 不可区分程序混淆  $i\mathcal{O}$ 、伪随机数发生器和可穿孔伪随机函数

构造如下:

- $\text{Setup}(1^\kappa)$ : 生成对电路族  $\mathcal{C}_\kappa$  的不可区分程序混淆  $i\mathcal{O}$ , 选取长度倍增的伪随机数发生器 PRG :  $\{0, 1\}^\kappa \rightarrow \{0, 1\}^{2\kappa}$  和可穿孔伪随机函数 PPRF :  $K \times \{0, 1\}^{2\kappa} \rightarrow Y$ ; 生成 PEPRF 的公开参数  $pp = (F, PK, SK, X, L, W, Y)$ , 其中  $PK = i\mathcal{O}(\mathcal{C}_\kappa)$ ,  $SK = K$ ,  $X = \{0, 1\}^{2\kappa}$ , 其中  $F := \text{PPRF}$ ,  $W = \{0, 1\}^\kappa$ ,  $L = \{x \in X : \exists w \in W \text{ s.t. } x = \text{PRG}(w)\}$ , 相应的采样算法 SampRel 以随机数  $r \in \{0, 1\}^\kappa$  为输入, 输出实例  $x \leftarrow \text{PRG}(r)$  和证据  $w = r$ .
- $\text{KeyGen}(pp)$ : 随机采样  $k \in K$  作为私钥  $sk$ , 计算  $pk \leftarrow i\mathcal{O}(\text{Eval})$  作为公钥.
- $\text{PrivEval}(sk, x)$ : 输出  $y \leftarrow \text{PPRF}_{sk}(x)$ .
- $\text{PubEval}(pk, x, w)$ : 将公钥  $pk$  解析为程序, 计算  $y \leftarrow pk(x, w)$ .



**定理 4.21**

基于不可区分程序混淆、伪随机数发生器和可穿孔伪随机函数的安全性, 构造 ?? 中的 PEPRF 满足自适应弱伪随机性.

**注记 4.22**

上述构造实质上展示了  $i\mathcal{O}$  可以将 Minicrypt 中的可穿孔伪随机函数编译为 Cryptomania 中的可公开求值伪随机函数.

**4.5.4 小结**

本章中引入了 PEPRF 这一全新的密码组件, 展示了它与已有密码组件之间的联系以及它的应用, 如图 ?? 所示. 引入 PEPRF 最大的理论意义在于它不仅首次阐明了经典的 Goldwasser-Micali PKE 和 ElGamal PKE 的构造机理, 还统一了几乎所有已知的构造范式. 作为首个实用的公钥加密, RSA PKE 影响深远, 令单向陷门函数的概念深入人心, 使得人们常有“构造公钥加密必须有陷门”的错觉. PEPRF 树立了正确的认知, 指出构造公钥加密的实质在于构造可公开求值的伪随机函数, 核心技术是“令同一函数存在两种求值方法”. 基于 PEPRF 的 PKE 构造恰与 Minicrypt 中基于 PRF 的 SKE 构造形成完美的形式契合与思想共鸣.

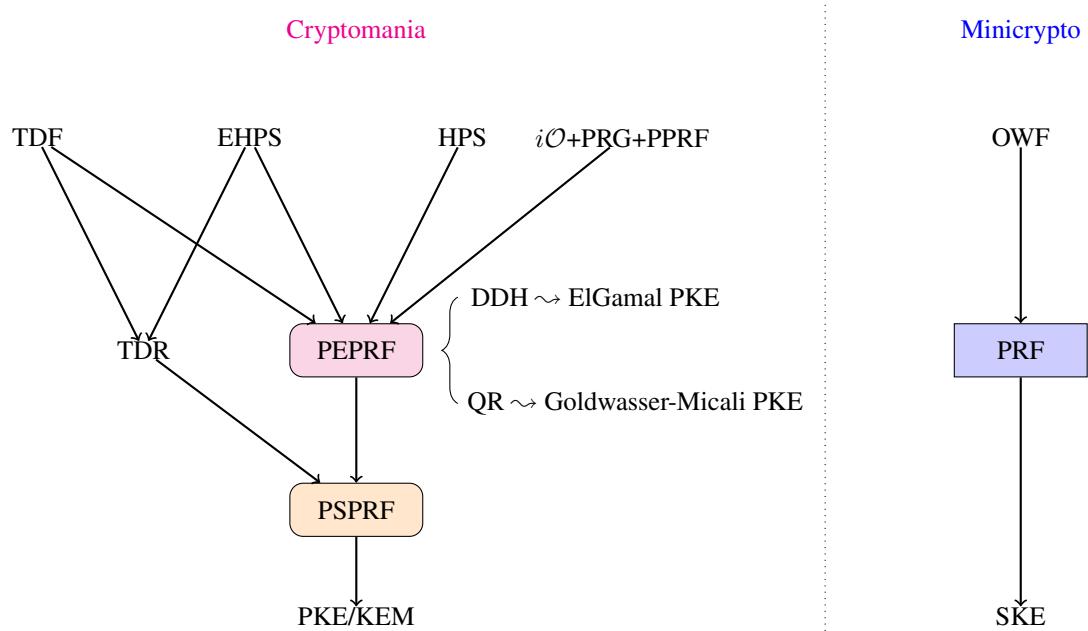


图 4.40: PEPRF 的构造与应用

PEPRF 的强大威力来源于其高度抽象, 它诠释了公钥加密设计的“万法同源, 殊途同归”.

**笔记** 抽象的概念是美妙的, 相信读者能够通过 PEPRF 感受到“大繁至简”的优雅与“高屋建瓴”的力量. 然而抽象概念是果, 具体构造是因. 切不能刻意过度的抽象而忽视具体构造, 正是多种多样具体构造才让我们能够有机缘洞见事物本质, 使得高度凝练的概念内涵丰富、意义深刻.

## 第五章 公钥加密的安全性增强

### 内容提要

- 抗泄漏安全
- 消息依赖密钥安全
- 抗篡改安全

## 5.1 抗泄漏安全

侧信道攻击,又称边信道攻击或旁路攻击,利用密码算法在实现过程中泄漏的物理信息,如运行时间 [**Kocher-CRYPTO-1999**]、电磁辐射 [**GMO-CHES-2001**]、能量功耗 [**Kocher-CRYPTO-1999**] 等来攻击密码算法的安全性。图 ??展示了侧信道攻击的方法,其中,  $F$  代表一种密码算法函数,如签名算法、解密算法等。除了私钥  $sk$  外,算法的输入可能还包括签名消息或解密密文  $x$ 。敌手可以利用上述侧信道攻击方法在算法  $F$  执行过程中获取私钥  $sk$  的部分信息,记作  $\text{leak}(sk)$ 。2008 年, Halderman 等人 [**Halderman-USENIX-Security-2008**] 还发现另一类特殊的侧信道攻击方法,即“冷启动”攻击(又称内存攻击)。简单来说,计算机断电后内存中存储的信息并不是立即被擦除掉,通过短暂的物理访问可以恢复动态随机存取存储器中的数据或密钥。上述这些物理信息都有可能泄漏密钥的部分信息,因此这类侧信道攻击统称为密钥泄漏攻击。与传统的数学分析方法相比,这类新型分析技术更有效,对密码算法的安全性构成巨大的威胁。早期抵御侧信道攻击的方法主要通过在算法实现过程中引入一些随机信息以减少泄漏的物理信息中含有的密钥信息,可参考文献 [**EHCC-Book-2005**] 第 29 章及其引文。然而这种方式难以同时抵抗多种类型的侧信道攻击技术并且这些方法缺少严格的安全性证明。类似不可区分选择密文攻击模型,如何建立合理的抗密钥泄漏攻击的安全模型,并从算法角度设计可证明安全的抗密钥泄漏密码方案是抗泄漏密码学研究的主要问题。

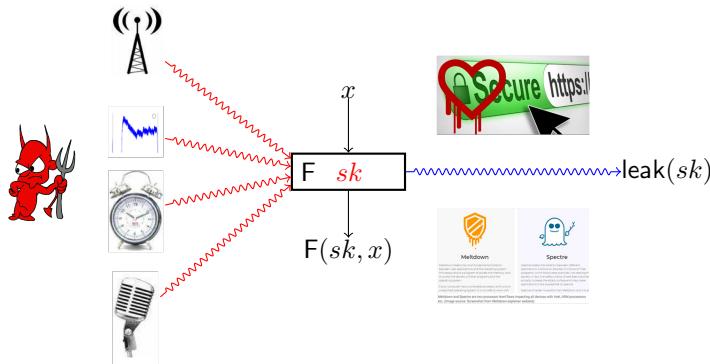


图 5.1: 侧信道攻击方法示意图

根据泄漏函数的形式不同,密钥泄漏模型可以分为有界密钥泄漏和无界密钥泄漏。2009 年, Akavia、Goldwasser 和 Vaikuntanathan [**AGV-TCC-2009**] 受“冷启动”攻击的启发,提出一种非常实用的刻画密钥泄漏的模型。该模型允许攻击者获得密钥的任意信息,只要所有获取的信息比特长度不超过某一阈值  $\lambda$  即可。因此,该模型一般称为有界密钥泄漏模型 (Bounded-Leakage Model, 简称 BLM 模型)。具体来讲,攻击者可以通过一系列有效、可计算函数  $f$ ,称之为泄漏函数,适应性地访问私钥  $sk$ ,并获取相应的泄漏信息  $f(sk)$ ,而对攻击者的要求是所有泄漏函数的输出长度之和不超过该阈值。由于 BLM 模型既简单又能涵盖广泛的侧信道攻击方法,近年来,该模型得到了密码学界的广泛关注。特别地,有界密钥泄漏模型可以涵盖以下两种密钥泄漏情形: 相对泄漏 (Relative Leakage) 和绝对泄漏 (Absolute Leakage)。

- **相对泄漏:** 指总体泄漏量与私钥长度的比率是相对固定的。这一比率通常称为相对泄漏比率。例如,攻击者得到的泄漏信息长度不超过私钥长度的一半。相对泄漏能够刻画多种侧信道攻击的情景,包括 Halderman 等人的“冷启动”攻击、针对智能卡的微波攻击等。因此,很多抗密钥泄漏方案都是在相对泄漏模型下设计的。
- **绝对泄漏:** 指相对泄漏量可以非常巨大。这种模型在一些场合是非常实用的。例如,当系统中存有恶意软件时,病毒程序可能会将用户大量的敏感数据传送给远程的控制服务器。但是在很多情况下,病毒程序下载巨量数据消耗的时间和代价很大。因此,抵御这种类型侧信道攻击最好的方法是将私钥变得巨大,以至于攻击者无法获取超过阈值的信息量。Crescenzo 等人 [**CLW-TCC-2006**] 和 Dziembowski [**Stefan-TCC-2006**] 将这一模型称为有界恢复模型 (Bounded Retrieval Model, 简称 BRM 模型)。在有界恢复模型中,设计密码算法的基本方式是通过增加敏感数据的存储空间来实现安全性,但是不能影响系统其他方面的性能。特别地,合法

用户仅需要访问很小一部分的密钥信息, 而他的计算和通信开销并不会有太大的增加.

有界恢复模型可以看做是内存泄漏模型的推广. 在 BRM 模型中设计方案的困难性主要在于方案效率仅能依赖方案的安全参数, 而不能依赖私钥的大小. 在相对泄漏模型中, 方案的效率一般与私钥大小有关. 通过扩大私钥空间来提高私钥泄漏量的同时, 方案的效率往往会显著下降. 尽管如此, 在 BRM 模型中设计方案通常先在相对泄漏模型中进行设计. 为此, 本节重点介绍相对泄漏模型下的抗泄漏公钥加密方案的几种典型设计方法.

### 5.1.1 安全模型

本节主要介绍公钥加密方案的有界密钥泄漏安全模型  $\lambda$ -BKL-CCA 安全模型. 在该模型中, 敌手不仅可以访问解密预言机, 而且可以获得密钥的部分信息. 密钥泄漏查询由任意一组输出长度之和不超过泄漏上界  $\lambda$  的函数组成. 敌手可以适应性地选择函数  $f$  并获得密钥的函数值  $f(sk)$ . 很显然, 如果函数  $f$  的输出没有任何限制, 则任何(公钥)加密方案都不可能抵抗密钥泄漏攻击.

**BKL-CCA 安全性.** 定义公钥加密方案敌手  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  的优势函数如下:

$$\text{Adv}_{\mathcal{A}}(\kappa) = \Pr_{\beta' = \beta : \begin{array}{l} pp \leftarrow \text{Setup}(1^\kappa); \\ (pk, sk) \leftarrow \text{KeyGen}(pp); \\ (m_0, m_1, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{decrypt}}(\cdot), \mathcal{O}_{\text{leak}}(\cdot)}(pp, pk); \\ \beta \stackrel{\$}{\leftarrow} \{0, 1\}; \\ c^* \leftarrow \text{Encrypt}(pk, m_\beta); \\ \beta' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{decrypt}}(\cdot)}(pp, pk, state, c^*); \end{array}} \left[ - \frac{1}{2} \right],$$

在上述定义中,  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  的含义类似 IND-CCA 安全模型中的敌手, 表示敌手  $\mathcal{A}$  可划分为两个阶段, 划分界线是接收到挑战密文  $c^*$  前后,  $state$  表示  $\mathcal{A}_1$  向  $\mathcal{A}_2$  传递的信息, 记录部分攻击进展.  $\mathcal{O}_{\text{decrypt}}(\cdot)$  表示解密预言机, 其在接收到密文  $c$  的询问后输出  $\text{Decrypt}(sk, c)$ .  $\mathcal{O}_{\text{leak}}(\cdot)$  表示密钥泄漏预言机, 其在接收到泄漏函数  $f_i : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda_i}$  的询问后输出  $f_i(sk)$  且所有泄漏函数输出长度之和满足  $\sum_i \lambda_i \leq \lambda$ . 如果任意的 PPT 敌手  $\mathcal{A}$  在上述游戏中的优势函数均为可忽略函数, 则称公钥加密方案是  $\lambda$ -BKL-CCA 安全的.

 **笔记** 在 BKL-CCA 模型中, 敌手在获得挑战密文后是不允许再访问密钥泄漏服务的. 否则, 敌手可以通过编辑挑战密文的解密函数来获得明文的部分比特信息, 从而区分挑战密文加密的是  $m_0$  还是  $m_1$ . 不可区分或语义安全性的要求是非常高的, 它不允许敌手获得任何除消息空间分布之外的有用信息. 如果允许敌手在看到挑战密文后继续访问密钥泄漏函数, 则模型的安全目标必然会降低. 为此, 2011 年 Halevi 和 Lin [HL-TCC-2011] 提出“After-the-fact”密钥泄漏模型, 利用明文的剩余熵来刻画方案的安全性. 在上述定义中, 若不允许敌手访问任何解密服务, 则这就是  $\lambda$ -BKL-CPA 安全模型的定义; 若令  $\lambda = 0$ , 即敌手没有访问密钥泄漏预言机, 则上述定义即是标准 IND-CCA 安全性的定义.

### 5.1.2 NS09 方案

#### 5.1.2.1 基于哈希证明系统的通用构造

2009 年, Naor 和 Segev [NS-CRYPTO-2009] 基于哈希证明系统提出一种抗泄漏公钥加密方案的通用构造方法. 该方案结构简单, 是哈希证明系统在抗泄漏密码学中的一个经典应用案例.

##### 构造 5.1 (Naor-Segev BKL-CPA 方案)

令  $\lambda = \lambda(\kappa)$  为密钥泄漏量的上界,  $\epsilon_1$  和  $\epsilon_2$  是两个可忽略的量. 该方案依赖一个  $\epsilon_1$ -universal 哈希证明系统  $\text{HPS} = (\text{HPS}.\text{Setup}, \text{HPS}.\text{KeyGen}, \text{HPS}.\text{PubEval}, \text{HPS}.\text{PrivEval})$  和一个平均情况下  $(\log \Pi - \lambda, \epsilon_2)$ -强提取器  $\text{Ext} : \Pi \times \{0, 1\}^t \rightarrow \{0, 1\}^m$ .

- $\text{Setup}(1^\kappa)$ : 运行  $\text{HPS}.\text{Setup}(1^\kappa)$ , 输出  $\text{HPS}$  的一个实例参数  $\text{params} = (\mathsf{H}, SK, PK, X, L, W, \Pi, \alpha)$ . 选

择一个平均情况下  $(\log \Pi - \lambda, \epsilon_2)$ -强提取器  $\text{Ext} : \Pi \times \{0, 1\}^t \rightarrow \{0, 1\}^m$ . 将  $pp = (\text{params}, \text{Ext})$  作为公开参数, 其中  $\{0, 1\}^m$  作为明文空间为和  $X \times \{0, 1\}^t \times \{0, 1\}^m$  作为密文空间.

- $\text{KeyGen}(pp)$ : 运行  $(pk, sk) \leftarrow \text{HPS.KeyGen}(pp)$ , 输出公钥  $pk$  和私钥  $sk$ .
- $\text{Encrypt}(pk, M; r)$ : 以公钥  $pk$ , 明文  $M \in \{0, 1\}^m$  和随机数  $r$  为输入, 执行如下步骤:
  1. 运行  $(x, w) \leftarrow \text{SampR}(r)$  生成随机实例和相应的证据;
  2. 通过  $\text{HPS.PubEval}(pk, x, w)$  计算实例  $x$  的哈希证明  $\pi \leftarrow \text{H}_{sk}(x)$ ;
  3. 随机选择  $s \xleftarrow{R} \{0, 1\}^t$ , 计算  $\psi = \text{Ext}(\pi, s) \oplus M$ ;
  4. 输出  $(x, s, \psi)$  作为密文  $c$ .
- $\text{Decrypt}(sk, c)$ : 以私钥  $sk$  和密文  $c = (x, s, \psi)$  为输入, 通过  $\text{HPS.PrivEval}(sk, x)$  计算  $x$  的哈希证明  $\pi \leftarrow \text{H}_{sk}(x)$ , 再恢复明文  $M' := \psi \oplus \text{Ext}(\pi, s)$ .



**正确性.** 根据哈希证明系统的正确性, 即  $\text{HPS.PrivEval}(sk, x) = \text{HPS.PubEval}(pk, x, w) = \text{H}_{sk}(x)$ , 以下公式 ??说明方案具有完美正确性:

$$\begin{aligned}
 M' &= \psi \oplus \text{Ext}(\text{HPS.PrivEval}(sk, x), s) \\
 &= \text{Ext}(\text{HPS.PubEval}(pk, x, w) \oplus M \oplus \text{Ext}(\text{HPS.PrivEval}(sk, x), s)) \\
 &= \text{Ext}(\text{H}_{sk}(x), s) \oplus M \oplus \text{Ext}(\text{H}_{sk}(x), s) \\
 &= M
 \end{aligned} \tag{5.1}$$

### 定理 5.1

如果 HPS 是  $\epsilon_1$ -universal 且  $\text{Ext}$  是一个平均情况下  $(\log \Pi - \lambda, \epsilon_2)$ -强提取器, 那么 NS-PKE 是  $\lambda$ -BKL-CPA 安全的, 其中  $\lambda \leq \log |\Pi| - \omega(\kappa) - m$ ,  $m$  是明文的比特长度.



**证明** 令  $S_i$  表示敌手在 Game<sub>i</sub> 中成功概率. 以游戏序列的方式组织证明如下:

Game<sub>0</sub>: 该游戏是标准的 BKL-CPA 游戏, 挑战者  $\mathcal{CH}$  和敌手  $\mathcal{A}$  交互如下:

- 初始化:  $\mathcal{CH}$  运行  $\text{Setup}(1^\kappa)$  生成公开参数  $pp$ , 同时运行  $\text{KeyGen}(pp)$  生成公私钥对  $(pk, sk)$ .  $\mathcal{CH}$  将  $(pp, pk)$  发送给  $\mathcal{A}$ .
- 询问: 假设  $f_i : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda_i}$  是  $\mathcal{A}$  的第  $i$  次泄漏预言机  $\mathcal{O}_{\text{leak}}(\cdot)$  查询.  $\mathcal{CH}$  首先判断  $\sum_i \lambda_i \leq \lambda$  是否成立, 若成立则返回  $f_i(sk)$ , 否则返回  $\perp$ .
- 挑战:  $\mathcal{A}$  选择  $M_0, M_1 \in \mathbb{G}$  并发送给  $\mathcal{CH}$ .  $\mathcal{CH}$  选择随机比特  $\beta \in \{0, 1\}$  和随机数  $r$ , 作如下计算:
  1. 运行  $(x, w) \leftarrow \text{SampR}(r)$  生成随机实例和相应的证据;
  2. 通过  $\text{HPS.PubEval}(pk, x, w)$  计算实例  $x$  的哈希证明  $\pi \leftarrow \text{H}_{sk}(x)$ ;
  3. 随机选择  $s \xleftarrow{R} \{0, 1\}^t$ , 计算  $\psi = \text{Ext}(\pi, s) \oplus M_b$ ;
  4. 输出  $(x, s, \psi)$  作为挑战密文  $c^*$  并发送给  $\mathcal{A}$ .
- 猜测:  $\mathcal{A}$  输出对  $\beta$  的猜测  $\beta'$ .  $\mathcal{A}$  成功当且仅当  $\beta' = \beta$ .

根据定义, 我们有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/2|$$

Game<sub>1</sub>: 与 Game<sub>0</sub> 的唯一不同在于挑战密文中哈希证明的生成方式,  $\mathcal{CH}$  不再通过  $\text{HPS.PubEval}(pk, x, w)$  计算哈希证明, 而是通过  $\text{HPS.PrivEval}(sk, x)$  计算  $x$  的哈希证明  $\pi \leftarrow \text{H}_{sk}(x)$ . 根据 HPS 两种工作模式的等价性可知, 敌手  $\mathcal{A}$  在游戏 Game<sub>0</sub> 和 Game<sub>1</sub> 中的视图是一样的. 故, 我们有

$$\text{Game}_0 \equiv \text{Game}_1$$

Game<sub>2</sub>: 与 Game<sub>1</sub> 的唯一不同在于挑战密文中随机实例  $x$  的选取方式.  $\mathcal{CH}$  调用  $\text{SampNo}(r)$  采样  $x \leftarrow X \setminus L$ . 根据

SMP 问题的困难性, 敌手  $\mathcal{A}$  在游戏 Game<sub>1</sub> 和 Game<sub>2</sub> 中的视图计算不可区分:

$$\text{Game}_1 \approx_c \text{Game}_2$$

Game<sub>3</sub>: 与 Game<sub>2</sub> 的唯一不同在于挑战密文中哈希证明  $\pi$  的选择方式.  $\mathcal{CH}$  从随机选取  $\pi \leftarrow \Pi$ . 根据 HPS 的 universal1 性质, 我们可以证明敌手  $\mathcal{A}$  在游戏 Game<sub>2</sub> 和 Game<sub>3</sub> 中的视图统计上不可区分:

$$\text{Game}_2 \approx_s \text{Game}_3$$

: 这是因为, 在没有任何密钥泄漏的情况下, 根据 HPS 的  $\epsilon_1$ -universal 性质, 我们有

$$\Delta((pk, x, \mathsf{H}_{sk}(x)), (pk, x, \pi)) \leq \epsilon_1$$

令密钥泄漏预言机的输出信息为  $\mathsf{aux}$ , 它是公钥  $pk$  和私钥  $sk$  的一个函数. 但是,  $\mathsf{aux}$  的分布由公钥  $pk$ , 随机实例  $x$  和哈希证明  $\mathsf{H}_{sk}(x)$  完全确定, 即  $\mathsf{aux} = \mathsf{aux}(pk, x, \mathsf{H}_{sk}(x))$ . 根据统计距离的性质, 可得,

$$\Delta((pk, x, \mathsf{H}_{sk}(x), \mathsf{aux}(pk, x, \mathsf{H}_{sk}(x))), (pk, x, \pi, \mathsf{aux}(pk, x, \pi))) \leq \epsilon_1$$

利用随机种子为  $s$  的强提取器  $\mathsf{Ext}$  作用在上述两个分布上不会增加它们的统计距离, 故我们可得

$$\Delta((pk, x, \mathsf{Ext}(\mathsf{H}_{sk}(x), s), s, \mathsf{aux}), (pk, x, \mathsf{Ext}(\pi, s), s, \mathsf{aux})) \leq \epsilon_1,$$

由于上述分析可知, 敌手  $\mathcal{A}$  在上述两个游戏中的视图统计距离相差不超过  $\epsilon_1$ .

Game<sub>4</sub>: 与 Game<sub>3</sub> 的唯一不同在于挑战密文中提取器  $\mathsf{Ext}(\pi, s)$  的选取方式.  $\mathcal{CH}$  随机选择  $k \leftarrow \{0, 1\}^m$ , 再计算  $\psi = k \oplus M_b$ . 由于  $k$  是随机且独立于消息  $m_b$  选取的, 所以在该游戏中敌手没有任何优势猜测挑战消息, 即  $\mathcal{A}$  成功的概率为:

$$\Pr[S_0] = 1/2$$

最后, 我们证明即使在泄漏  $\lambda$  比特密钥信息的情况下, 敌手在游戏 Game<sub>3</sub> 和 Game<sub>4</sub> 两个游戏中的视图仍然是不可区分的.

对于分布  $(pk, x, k, \mathsf{Ext}(\pi, s), s, \mathsf{aux})$ ,  $\lambda$  比特的密钥泄漏量  $\mathsf{aux}$  最多使  $\pi$  的平均极小熵减少  $\lambda$ , 即  $H_\infty(\pi | (pk, x, \mathsf{aux})) \geq \tilde{H}_\infty(\pi | (pk, x)) - \lambda = \log \Pi - \lambda$ . 利用强提取器  $\mathsf{Ext}$  的性质, 可得

$$\Delta((pk, x, \mathsf{Ext}(\pi, s), s, \mathsf{aux}), (pk, x, k, s, \mathsf{aux})) \leq \epsilon_2.$$

其中  $k \in \{0, 1\}^m$  是独立且随机选取的. 由此, 我们可知敌手  $\mathcal{A}$  在上述两个游戏中的视图统计距离相差不超过  $\epsilon_2$ .

综上, 定理得证. □

### 5.1.2.2 基于 DDH 问题的实例化方案

下面给出一个基于 DDH 问题的实例化构造方案.

#### 构造 5.2 (DDH-based NS-PKE)

- **Setup( $1^\kappa$ )**: 运行  $\text{GenGroup}(1^\kappa)$ , 生成一个循环群  $(\mathbb{G}, q, g)$ . 令  $\lambda = \lambda(\kappa)$  是泄漏参数 (泄漏量上界). 选择一个平均情况下  $(\log q - \lambda, \epsilon)$ -强提取器  $\mathsf{Ext} : \mathbb{G} \times \{0, 1\}^t \rightarrow \{0, 1\}^m$ . 输出公开参数  $pp = (\mathbb{G}, q, g, \mathsf{Ext})$ .
- **KeyGen( $pp$ )**: 随机选择  $x_1, x_2 \in \mathbb{Z}_q$  和  $g_1, g_2 \in \mathbb{G}$ . 计算  $h = g_1^{x_1} g_2^{x_2}$ . 输出公钥  $pk = (g_1, g_2, h)$  和私钥

$sk = (x_1, x_2)$ .

- **Encrypt**( $pk, M$ ): 输入公钥  $pk$  和明文  $M \in \{0, 1\}^m$ , 随机选择  $r \in \mathbb{Z}_q$  和  $s \in \{0, 1\}^t$ , 输出密文  $c = (g_1^r, g_2^r, s, \text{Ext}(h^r, s) \oplus M)$ .
- **Decrypt**( $sk, c$ ): 输入私钥  $sk = (x_1, x_2)$  和密文  $c = (u_1, u_2, s, e)$ , 输出明文  $m' := e \oplus \text{Ext}(u_1^{x_1} u_2^{x_2}, s)$ .

上述方案利用的哈希证明系统的具体描述见第四章的构造 4.9. 该哈希证明系统基于 DDH 语言构造, 是一个  $\frac{1}{q}$ -1-universal 的哈希证明系统. 根据定理 ??, 我们直接可以得到如下引理:

### 引理 5.1

如果 DDH 假设成立, 那么 NS-PKE 的实例化构造方案 ?? 是  $\lambda$ -BKL-CPA 安全的, 其中  $\lambda = (\log q - \omega(\log \kappa) - m)$ .



**笔记** 由上述引理可知, 实例化方案的密钥泄漏量可达  $L(1/2 - o(1))$ , 其中  $L$  表示私钥的长度. 然而, 该方案仅是 BKL-CPA 安全的. 为了实现 BKL-CCA 安全性, 一种直接的方法是将 Naor-Yung”双加密”模式应用与一个  $\lambda$ -BKL-CPA 安全的公钥加密方案上, 从而得到一个密钥泄漏比率不变且抗选择密文攻击安全的 BKL-CCA 安全公钥加密方案. 该方法需要引入适应性安全的非交互零知识证明系统, 在效率上具有一定的局限性. 下一小节, 我们介绍一种提高该方法效率且密钥泄漏比率不变的通用构造方法.

## 5.1.3 QL13 方案

2013 年, Qin 和 Liu [Qin-ASIACRYPT-2013] 在基于 1-universal 哈希证明系统的通用构造方案基础上, 通过引入一个一次有损过滤器 (One-time lossy filter) 实现从 BKL-CPA 到 BKL-CCA 安全性的提升且不降低密钥泄漏比率. 下面主要介绍一次有损过滤器的定义、基于一次有损过滤器的抗泄漏公钥加密通用构造及实例化方案.

### 5.1.3.1 一次有损过滤器

一个  $(\text{Dom}, \ell_{\text{LF}})$ -一次有损过滤器是一个以公钥  $ek_{\text{LF}}$  和标签  $t$  为指标的函数族:  $\{\text{LF}_{ek_{\text{LF}}, t} : \text{Dom} \rightarrow \mathcal{Y}\}$ . 函数族中的任意函数  $\text{LF}_{ek_{\text{LF}}, t}$  将  $X \in \text{Dom}$  映射到  $\text{LF}_{ek_{\text{LF}}, t}(X)$ . 给定公钥  $ek_{\text{LF}}$ , 标签集合  $\mathcal{T}$  可以分解为两个计算上不可区分的子集合: 单射标签集合  $\mathcal{T}_{\text{inj}}$  和有损标签集合  $\mathcal{T}_{\text{loss}}$ . 如果  $t$  属于单射标签, 则函数  $\text{LF}_{ek_{\text{LF}}, t}$  也是单射的并且像的大小为  $|\text{Dom}|$ ; 如果  $t$  是有损的, 则函数最多有  $2^{\ell_{\text{LF}}}$  个可能的输出结果. 因此, 若  $t$  是有损标签, 则  $\text{LF}_{ek_{\text{LF}}, t}(X)$  最多泄漏  $X$  的  $\ell_{\text{LF}}$  比特信息. 这一性质在方案证明中是至关重要的. 下面给出一次有损过滤器的形式化定义.

#### 定义 5.1 (一次有损过滤器)

一个  $(\text{Dom}, \ell_{\text{LF}})$ -一次有损过滤器 LF 包含以下三个 (概率) 多项式时间算法 ( $\text{LF.Gen}$ ,  $\text{LF.Eval}$ ,  $\text{LF.LTag}$ ) 并满足以下性质:

- (Key Generation)  $\text{LF.Gen}(1^\kappa)$ : 输入安全参数  $1^\kappa$ , 输出一对密钥  $(ek_{\text{LF}}, td_{\text{LF}})$ . 其中, 公钥  $ek_{\text{LF}}$  定义了标签集合  $\mathcal{T} = \{0, 1\}^* \times \mathcal{T}_c$ , 它由两个不相交的有损标签集合  $\mathcal{T}_{\text{loss}} \subseteq \mathcal{T}$  和单射标签集合  $\mathcal{T}_{\text{inj}} \subseteq \mathcal{T}$  构成. 每个标签  $t = (t_a, t_c) \in \mathcal{T}$  由辅助标签  $t_a \in \{0, 1\}^*$  和核心标签  $t_c \in \mathcal{T}_c$  两部分组成.  $td_{\text{LF}}$  是一个陷门, 利用它可以有效地从有损标签集合中进行抽样.
- (Evaluation)  $\text{LF.Eval}(ek_{\text{LF}}, t, X)$ : 给定公钥  $ek_{\text{LF}}$  和标签  $t$ , 将  $X \in \text{Dom}$  映射到  $\text{LF}_{ek_{\text{LF}}, t}(X)$ .
- (Lossy Tag Generation)  $\text{LF.LTag}(td_{\text{LF}}, t_a)$ : 利用陷门  $td_{\text{LF}}$  计算辅助标签  $t_a$  对应的核心标签  $t_c$ , 使其满足  $t = (t_a, t_c)$  是有损的.
- (Lossiness) 如果  $t$  是单射的, 则函数  $\text{LF}_{ek_{\text{LF}}, t}(\cdot)$  也是单射的; 如果  $t$  是有损的, 则  $\text{LF}_{ek_{\text{LF}}, t}(X)$  的像集合最多包含  $2^{\ell_{\text{LF}}}$  个元素. (在实际应用中, 通过调整公钥的其他参数, 原像集合可以逐渐增大而参数  $\ell_{\text{LF}}$  始终保持不变.)

- (Indistinguishability) 对于任意 PPT 算法  $\mathcal{A}$ , 区分有损标签和随机选取的标签是困难的. 严格来说, 对于任意 PPT 算法  $\mathcal{A}$ , 下面的优势函数

$$\text{Adv}_{\text{LF}, \mathcal{A}}^{\text{ind}}(\kappa) := |\Pr[\mathcal{A}(ek_{\text{LF}}, (t_a, t_c^{(0)})) = 1] - \Pr[\mathcal{A}(ek_{\text{LF}}, (t_a, t_c^{(1)})) = 1]|$$

是可忽略的. 其中  $(ek_{\text{LF}}, td_{\text{LF}}) \leftarrow \text{LF.Gen}(1^\kappa)$ ,  $t_a \leftarrow \mathcal{A}(ek_{\text{LF}})$ ,  $t_c^{(0)} \leftarrow \text{LF.LTag}(td_{\text{LF}}, t_a)$ ,  $t_c^{(1)} \leftarrow \mathcal{T}_c$ .

- (Evasiveness) 对于任意 PPT 敌手  $\mathcal{A}$ , 即使给定一个有损标签情况下, 也无法计算一个新的非单射标签<sup>a</sup>. 具体来说, 下面定义的敌手优势是可忽略的:

$$\text{Adv}_{\text{LF}, \mathcal{A}}^{\text{eva}}(\kappa) := \Pr \left[ \begin{array}{l} (ek_{\text{LF}}, td_{\text{LF}}) \leftarrow \text{LF.Gen}(1^\kappa); \\ (t'_a, t'_c) \neq (t_a, t_c) \wedge (t'_a, t'_c) \in \mathcal{T} \setminus \mathcal{T}_{inj} : t_a \leftarrow \mathcal{A}(ek_{\text{LF}}); t_c \leftarrow \text{LF.LTag}(td_{\text{LF}}, t_a); \\ (t'_a, t'_c) \leftarrow \mathcal{A}(ek_{\text{LF}}, (t_a, t_c)) \end{array} \right]$$

<sup>a</sup>在有些情况下, 一个标签可能既不是单射的也不是有损的.



**笔记** 一次有损过滤器是一种简化的有损代数过滤器 (lossy algebraic filters) [Hofheinz-EUROCRYPT-2013]. 两者存在以下不同之处: 一是前者要求敌手最多知道一个有损标签; 而后者要求敌手可以获得多个有损标签, 这导致后者比前者实现难度大且实现的方案效率非常差. 二是前者不需要具有特定的代数结构, 而后者必须有特定的代数结构, 可用于多挑战密文的环境, 例如 KDM-CCA 安全性. 此外, 一次有损过滤器也可看作是一种不带求逆陷门的全除一有损陷门函数. 在单射模式下, OT-LF 没有求逆陷门, 而 ABO-TDF 则需要一个陷门能够用于求逆. 因此, 在相同定义域下, OT-LF 的计算效率一般比 ABO-TDF 高.

### 5.1.3.2 基于一次有损过滤器的通用构造

下面介绍如何利用 1-universal HPS 和 OT-LF 构造 BKL-CCA 安全的公钥加密方案. 令  $\text{HPS} = (\text{HPS.Setup}, \text{HPS.KeyGen}, \text{HPS})$  是一个  $\epsilon_1$ -universal HPS, 其中  $\text{HPS.KeyGen}$  生成一个投影哈希函数的实例  $pp = (\mathsf{H}, SK, PK, X, L, W, \Pi, \alpha)$ .  $\text{LF} = (\text{LF.Gen}, \text{LF.Eval}, \text{LF.LTag})$  是一个  $(\Pi, \ell_{\text{LF}})$ -OT-LF. 定义  $\nu := \log(1/\epsilon_1)$ . 令  $\lambda$  是私钥泄漏的上界;  $\text{Ext} : \Pi \times \{0, 1\}^t \rightarrow \{0, 1\}^m$  是一个平均情况下  $(\nu - \lambda - \ell_{\text{LF}}, \epsilon_2)$ -强提取器, 其中  $\epsilon_2$  是一个可忽略量. 加密方案  $\text{PKE} = (\text{PKE.Gen}, \text{PKE.Enc}, \text{PKE.Dec})$  (明文空间为  $\{0, 1\}^m$ ) 的构造如下.

#### 定义 5.2 (Qin-Liu BKL-CCA-PKE)

令  $\lambda = \lambda(\kappa)$  为密钥泄漏量的上界,  $\epsilon_1$  和  $\epsilon_2$  是两个可忽略的量. 该方案依赖一个  $\epsilon_1$ -universal 哈希证明系统  $\text{HPS} = (\text{Setup}, \text{KeyGenPubEval}, \text{PrivEval})$  和一个平均情况下  $(\log \Pi - \lambda, \epsilon_2)$ -强提取器  $\text{Ext} : \Pi \times \{0, 1\}^t \rightarrow \{0, 1\}^m$ .

- $\text{Setup}(1^\kappa)$ : 运行  $\text{HPS.Setup}(1^\kappa)$ , 输出  $\text{HPS}$  的一个实例参数  $\text{params} = (\mathsf{H}, SK, PK, X, L, W, \Pi, \alpha)$ . 选择一个平均情况下  $(\log \Pi - \lambda, \epsilon_2)$ -强提取器  $\text{Ext} : \Pi \times \{0, 1\}^t \rightarrow \{0, 1\}^m$ . 将  $pp = (\text{params}, \text{Ext})$  作为公开参数, 其中  $\{0, 1\}^m$  作为明文空间和  $X \times \{0, 1\}^t \times \{0, 1\}^m \times \mathcal{Y} \times t_c$  作为密文空间.
- $\text{KeyGen}(pp)$ : 运行  $(pk, sk) \leftarrow \text{HPS.Gen}(pp)$  和  $(ek_{\text{LF}}, td_{\text{LF}}) \leftarrow \text{LF.Gen}(1^\kappa)$ . 输出加密方案的公钥  $PK = (pk, ek_{\text{LF}})$  和私钥  $SK = sk$ .
- $\text{Encrypt}(PK, M; r)$ : 以公钥  $PK = ((pk, ek_{\text{LF}})$ , 明文  $M \in \{0, 1\}^m$  和随机数  $r$  为输入, 执行如下步骤:
  - 运行  $(x, w) \leftarrow \text{SampR}(r)$  生成随机实例和相应的证据;
  - 通过  $\text{HPS.PubEval}(pk, x, w)$  计算实例  $x$  的哈希证明  $\pi \leftarrow \mathsf{H}_{sk}(x)$ ;
  - 随机选择  $s \xleftarrow{R} \{0, 1\}^t$ , 计算  $\psi = \text{Ext}(\pi, s) \oplus M$ ;
  - 随机选择  $t_c \xleftarrow{R} \mathcal{T}_c$ , 计算  $\tau \leftarrow \text{LF}_{ek_{\text{LF}}, t}( \pi )$ , 其中  $t = (t_a, t_c)$ ,  $t_a = (x, s, \psi)$ ;
  - 输出密文  $C = (x, s, \psi, \tau, t_c)$ .
- $\text{Decrypt}(SK, C)$ : 以私钥  $SK = sk$  和密文  $C = (x, s, \psi, \tau, t_c)$  为输入, 执行如下步骤:

1. 计算  $\pi' \leftarrow \mathsf{H}_{sk}(x)$  和  $\tau' \leftarrow \mathsf{LF}_{ek_{\text{LF}}, t}(\pi')$ , 其中  $t = ((x, s, \psi), t_c)$ ;
2. 验证  $\tau' = \tau$  是否成立. 如果不成立, 则返回  $\perp$ ; 否则输出明文文  $M' := \psi \oplus \mathsf{Ext}(\pi', s)$ .



**正确性.** 方案的正确性可以通过哈希证明系统和一次有损过滤器的正确性直接验证. 由于  $\mathsf{HPS}.\mathsf{PrivEval}(sk, x) = \mathsf{HPS}.\mathsf{PubEval}(pk, x, w) = \mathsf{H}_{sk}(x)$ , 所以  $\mathsf{LF}_{ek_{\text{LF}}, t}(\pi') = \mathsf{LF}_{ek_{\text{LF}}, t}(\pi)$ . 从而, 解密算法中的验证等式  $\tau' = \tau$  成立. 进一步, 根据下面的公式 ?? 可以说明方案具有完美正确性:

$$\begin{aligned}
 M' &= \psi \oplus \mathsf{Ext}(\mathsf{HPS}.\mathsf{PrivEval}(sk, x), s) \\
 &= \mathsf{Ext}(\mathsf{HPS}.\mathsf{PubEval}(pk, x, w) \oplus M \oplus \mathsf{Ext}(\mathsf{HPS}.\mathsf{PrivEval}(sk, x), s), s) \\
 &= \mathsf{Ext}(\mathsf{H}_{sk}(x), s) \oplus M \oplus \mathsf{Ext}(\mathsf{H}_{sk}(x), s) \\
 &= M
 \end{aligned} \tag{5.2}$$

### 定理 5.2

假设  $\mathsf{HPS}$  是一个  $\epsilon_1$ -universal 哈希证明系统,  $\mathsf{LF}$  是一个  $(\Pi, \ell_{\text{LF}})$ -一次有损过滤器,  $\mathsf{Ext} : \Pi \times \{0, 1\}^t \rightarrow \{0, 1\}^m$  是一个平均情况下  $(\nu - \lambda - \ell_{\text{LF}}, \epsilon_2)$ -强提取器. 如果  $\lambda \leq \nu - m - \ell_{\text{LF}} - \omega(\log \kappa)$ , 则加密方案  $\mathsf{PKE}$  是  $\lambda$ -BKL-CCA 安全的, 其中  $m$  是明文长度,  $\nu := \log(1/\epsilon_1)$ . 具体有,

$$\mathsf{Adv}_{\mathsf{PKE}, \mathcal{A}}^{\text{bkl-cca}}(\kappa) \leq \mathsf{Adv}_{\mathsf{LF}, \mathcal{B}_1}^{\text{ind}}(\kappa) + \mathsf{Adv}_{\mathsf{HPS}, \mathcal{B}_2}^{\text{smp}}(\kappa) + Q(\kappa) \cdot \mathsf{Adv}_{\mathsf{LF}, \mathcal{B}_3}^{\text{eva}}(\kappa) + \frac{Q(\kappa)2^{\lambda + \ell_{\text{LF}} + m}}{2^\nu - Q(\kappa)} + \epsilon_2$$

其中  $Q(\kappa)$  表示  $\mathcal{A}$  解密询问的次数.



在给出定理的证明之前, 我们先概括介绍一下该方案证明的思路. 该方案首先使用哈希证明系统生成一个对称密钥  $\pi$ , 它既用作隐藏明文又用作验证密文的正确性. 为了允许私钥泄漏, 方案使用一个提取器从  $\pi$  中提取比较短的均匀密钥来掩盖消息, 使用一次有损过滤器  $\mathsf{LF}_{ek_{\text{LF}}, t}(\pi)$  来验证密文完整性. 在挑战密文  $C^*$  中, 过滤器工作在有损模式下, 因此密文泄漏  $\pi$  的信息量是固定的. 对于非法密文, 过滤器以压倒性的概率工作在单射模式下, 因此过滤器输出的结果不会降低  $\pi$  的熵. 这就要求敌手必须完全知道  $\pi$  的值, 否则拒绝解密查询. 对于敌手来说, 若  $\pi$  的剩余熵足够大, 则正确猜测  $\pi$  的概率是可忽略的.

**证明** 定理 ?? 的证明. 我们通过不可区分游戏的思想 [Shoup-ePrint-2004] 来证明上述定理的结论. 每个游戏的参与者包括挑战者 (模拟者)  $\mathcal{CH}$  和一个 PPT 敌手 (算法)  $\mathcal{A}$ , 敌手通过与挑战者交互通信, 最终输出一比特信息  $b'$  作为对模拟者选择的随机比特  $b$  的猜测. 在游戏  $\text{Game}_i$  中, 用  $S_i$  表示事件  $b = b'$ , 用  $C^* = (x^*, s^*, \psi^*, \tau^*, t_c^*)$  表示挑战密文. 初始游戏  $\text{Game}_0$  及后续游戏的定义如下:

$\text{Game}_0$ : 该游戏是标准的 BKL-CCA 游戏, 在该游戏中, 挑战者  $\mathcal{CH}$  运行  $\mathsf{Setup}(1^\kappa)$  生成公开参数  $pp$ , 同时运行  $\mathsf{KeyGen}(pp)$  生成公私钥对  $(PK, SK)$ .  $\mathcal{CH}$  将  $(pp, PK)$  发送给  $\mathcal{A}$ . 对于每个解密询问  $C$  或私钥泄漏询问  $f_i$ , 挑战者利用私钥  $SK$  作出回答  $\mathsf{Decrypt}(SK, CT)$  或  $f_i(SK)$ . 当敌手提交两个等长消息  $M_0$  和  $M_1$  时, 挑战者将密文  $C^* \leftarrow \mathsf{Encrypt}(PK, M_b)$  发送给敌手  $\mathcal{A}$ . 只要  $C \neq C^*$ , 挑战者可以继续回答敌手的解密询问. 最后, 敌手  $\mathcal{A}$  输出一个比特  $b'$ , 作为对  $b$  的猜测. 根据 BKL-CCA 安全性的定义, 我们有:

$$\mathsf{Adv}_{\mathsf{PKE}, \mathcal{A}}^{\text{bkl-cca}}(\kappa) := \left| \Pr[S_0] - \frac{1}{2} \right|.$$

$\text{Game}_1$ : 该游戏与  $\text{Game}_0$  的不同之处在于密钥生成方式和挑战密文中核心标签的选取方式. 具体地, 当运行  $\mathsf{KeyGen}(pp)$  生成加密方案的公钥/私钥对时, 挑战者除了保留解密私钥  $SK$  外, 还保留一次有损过滤器  $\mathsf{LF}$  的陷门  $td_{\text{LF}}$ . 在选择核心标签时, 模拟者利用  $\mathsf{LF}.LT_{\text{Tag}}(td_{\text{LF}}, t_a^*)$  来计算  $t_c^*$  (其中  $t_a^* = (x^*, s^*, \psi^*)$ ), 而不是从  $\mathcal{T}_c$  中随

机选取. 利用 LF 的有损标签和随机标签不可区分性, 可直接得出

$$|\Pr[S_0] - \Pr[S_1]| \leq \text{Adv}_{\text{LF}, \mathcal{B}_1}^{\text{ind}}(\kappa)$$

其中  $\mathcal{B}_1$  是一个攻击 LF 不可区分性的敌手.

**Game<sub>2</sub>**: 该游戏与 Game<sub>1</sub> 的唯一不同之处在于增加了一条特殊规则用于拒绝解密. 具体来讲, 当敌手解密询问的密文  $C = (x, s, \psi, \tau, t_c)$  满足  $t = (t_a, t_c) = (t_a^*, t_c^*) = t^*$  时, 解密服务立即返回  $\perp$  并终止查询服务. 为简便起见, 我们将这种标签称为重用的有损过滤器标签. 下面证明, 若解密询问中的标签是重用的, 则在 Game<sub>1</sub> 和 Game<sub>2</sub> 中, 这种解密询问都将被拒绝服务. 考虑下面两种情况:

- Case 1:  $\tau = \tau^*$ . 这意味着  $C = C^*$ , 由于  $\mathcal{A}$  是不允许对挑战密文进行解密询问的, 这种情况在两个游戏中都将被拒绝解密.
- Case 2:  $\tau \neq \tau^*$ . 由  $t = ((x, s, \psi), t_c) = ((x^*, s^*, \psi^*), t_c^*) = t^*$ , 可知  $\pi = \pi^*$ ,  $\text{LF}_{ek_{\text{LF}}, t}(\pi) = \text{LF}_{ek_{\text{LF}}, t^*}(\pi^*) = \tau^*$ . 因此, 这种解密询问在 Game<sub>1</sub> 中已经被拒绝了.

根据以上分析, 可知敌手  $\mathcal{A}$  在游戏 Game<sub>1</sub> 和 Game<sub>2</sub> 中的视图是一样的. 故, 我们有

$$\Pr[S_1] = \Pr[S_2].$$

**Game<sub>3</sub>**: 该游戏与 Game<sub>2</sub> 的唯一不同之处在于挑战密文中  $\pi^*$  的生成方式. 在该游戏中, 挑战者通过哈希证明系统的私钥运算 HPS.PrivEval( $sk, x^*$ ) 替代公开运算 HPS.PubEval( $pk, x^*, w^*$ ) 来计算  $\pi^*$ . 根据 HPS 的投影性质, 这只是一种概念上的改变, 对计算结果没有任何影响. 故, 我们有

$$\Pr[S_3] = \Pr[S_2].$$

**Game<sub>4</sub>**: 该游戏与 Game<sub>3</sub> 的唯一不同之处在于挑战密文中随机实例  $x^*$  的选取方式. 在该游戏中, 挑战调用 SampNo( $r$ ) 采样  $x^* \leftarrow X \setminus L$ . 根据 SMP 问题的困难性, 敌手  $\mathcal{A}$  在游戏 Game<sub>4</sub> 和 Game<sub>3</sub> 中的视图计算不可区分, 即

$$|\Pr[S_3] - \Pr[S_4]| \leq \text{Adv}_{\text{HPS}, \mathcal{B}_2}^{\text{smp}}(\kappa)$$

其中  $\mathcal{B}_2$  为攻击 SMP 问题的敌手.

**Game<sub>5</sub>**: 该游戏与 Game<sub>4</sub> 的唯一不同之处在于增加了一种特殊的解密规则. 该规则为: 如果敌手解密询问的密文  $C = (x, s, \psi, \tau, t_c)$  满足  $x \in X \setminus L$ , 则解密预言机立即返回  $\perp$  并终止服务. 令事件  $\text{bad}_x$  表示一个解密查询在游戏 Game<sub>5</sub> 中被拒绝解密服务, 而在游戏 Game<sub>4</sub> 中可通过解密规则的验证. 因此, 当且仅当事件  $\text{bad}_x$  发生, 敌手在游戏 Game<sub>5</sub> 和 Game<sub>4</sub> 中的视图不同. 根据差分引理, 可知

$$|\Pr[S_4] - \Pr[S_5]| \leq \Pr[\text{bad}_x].$$

下面的结论保证了事件  $\text{bad}_x$  发生的概率是可忽略的. 我们稍后再给出它的证明.

### 引理 5.2

假设敌手最多询问  $Q(\kappa)$  次解密服务, 则

$$\Pr[\text{bad}_x] \leq Q(\kappa) \cdot \text{Adv}_{\text{LF}, \mathcal{B}}^{\text{eva}}(\kappa) + \frac{Q(\kappa)2^{\lambda+\ell_{\text{LF}}+m}}{2^\nu - Q(\kappa)}$$

其中  $\mathcal{B}_3$  是一个攻击一次有损过滤器“evasiveness”的敌手.



**Game<sub>6</sub>**: 该游戏与 Game<sub>5</sub> 的唯一不同之处在于挑战密文中  $\psi^*$  的生成方式. 在该游戏中, 模拟者从  $\{0, 1\}^m$  中随机选择  $\psi^*$ , 而不是通过  $\text{Ext}(\text{H}_{sk}(x^*), s^*) \oplus M_b$  计算所得.

下面来证明(对于敌手来说)游戏 Game<sub>5</sub> 和 Game<sub>6</sub> 定义的环境是不可区分的. 首先, 从敌手的视图角度(记做  $\text{view}'_{\mathcal{A}}$ ) 分析  $\mathsf{H}_{sk}(x^*)$  的极小熵. 因为非法密文直接被拒绝解密, 所以敌手利用解密服务不可能获得关于  $\mathsf{H}_{sk}(x^*)$  的信息. 所有关于密钥的信息只可能来自公钥, 挑战密文和私钥泄漏, 即  $pk, x^*, \pi^*$  和  $\lambda$  比特的私钥泄漏信. 根据平均最小熵的性质并结合  $\tau^*$  仅有  $2^{\ell_{LF}}$  个可能取值和  $\tilde{H}_\infty(\mathsf{H}_{sk}(x^*) | (pk, x^*)) \geq \nu$  (对于所有  $pk$  和  $x^* \in X \setminus L$  都成立) 这一事实, 可得

$$\begin{aligned}\tilde{H}_\infty(\mathsf{H}_{sk}(x^*) | \text{view}'_{\mathcal{A}}) &= \tilde{H}_\infty(\mathsf{H}_{sk}(x^*) | pk, x^*, \lambda\text{-leakage}, \tau^*) \\ &\geq \tilde{H}_\infty(\mathsf{H}_{sk}(x^*) | pk, x^*) - \lambda - \ell_{LF} \\ &\geq \nu - \lambda - \ell_{LF}.\end{aligned}$$

因此, 利用  $(\nu - \lambda - \ell_{LF}, \epsilon_2)$ -提取器  $\text{Ext} : \Pi \times \{0, 1\}^t \rightarrow \{0, 1\}^m$  从信息源  $\mathsf{H}_{sk}(x^*)$  中提取的随机串  $\text{Ext}(\mathsf{H}_{sk}(x^*), s^*)$  与均匀分布的统计距离不超过可忽略量  $\epsilon_2$ . 故, 我们有

$$|\Pr[S_5] - \Pr[S_6]| \leq \epsilon_2.$$

在游戏 Game<sub>6</sub> 中, 挑战密文与加密的消息是完全独立的. 因此

$$\Pr[S_6] = 1/2.$$

综上可得定理 ?? 的结论. 证毕! □

下面证明引理 ??.

**证明** (引理 ?? 的证明) 令事件  $F$  表示在游戏 Game<sub>4</sub> 中, 存在一个解密询问  $C = (x, s, \psi, \tau, t_c)$  使得  $t = ((x, s, \psi), t_c)$  既不是单射标签也不是重用标签. 则

$$\Pr[\text{bad}_x] = \Pr[\text{bad}_x \wedge F] + \Pr[\text{bad}_x \wedge \bar{F}] \leq \Pr[F] + \Pr[\text{bad}_x | \bar{F}]$$

假设敌手  $\mathcal{A}$  最多询问  $Q(\kappa)$  次解密服务, LF 是一个一次有损过滤器, HPS 是一个  $\epsilon_1$ -universal 哈希证明系统. 下面证明

$$\Pr[F] \leq Q(\kappa) \cdot \text{Adv}_{LF, \mathcal{B}}^{\text{eva}}(\kappa) \tag{5.3}$$

$$\Pr[\text{bad}_x | \bar{F}] \leq \frac{Q(\kappa)2^{\lambda + \ell_{LF} + m}}{2^\nu - Q(\kappa)} \tag{5.4}$$

其中  $\nu = \log(1/\epsilon_1)$ .

**证明** (公式 ?? 的证明) 给定有损过滤器的公钥  $ek_{LF}^*$ ,  $\mathcal{B}$  通过模拟  $\mathcal{A}$  在游戏 Game<sub>4</sub> 中的环境来攻击有损过滤器的“evasiveness”. 除了令  $ek_{LF} = ek_{LF}^*$  外,  $\mathcal{B}$  按照游戏 Game<sub>4</sub> 中的方式来生成公钥  $PK$  的其他参数. 值得注意的是,  $\mathcal{B}$  知道 PKE 的私钥, 因此可以正确回答敌手  $\mathcal{A}$  的解密和私钥泄漏查询. 为了模拟挑战密文(其中过滤器的标签必须是有损的),  $\mathcal{B}$  通过一次有损过滤器提供的服务获得辅助标签  $t_a^* = (x^*, s^*, \psi^*)$  对应的有损标签  $t_c^*$ . 最后,  $\mathcal{B}$  随机选择  $i \in \{1, \dots, Q(k)\}$ , 并从  $\mathcal{A}$  的第  $i$  个解密询问中提取相应的过滤器标签  $t = ((x, s, \psi), t_c)$ . 很显然, 如果事件  $F$  发生了, 则至少以  $1/Q(\kappa)$  的概率,  $t$  是一个非单射标签. 也就是说  $\Pr[F] \leq Q(\kappa) \cdot \text{Adv}_{LF, \mathcal{B}}^{\text{eva}}(\kappa)$ .

公式 ?? 证毕! □

**证明** (公式 ?? 的证明) 在事件  $F$  未发生的前提下, 假设  $C = (x, s, \psi, \tau, t_c)$  是第一个令事件  $\text{bad}_x$  发生的解密查询, 即  $x \in X \setminus L$ ,  $\Pi = \text{LF}_{ek_{LF}, t}(\mathsf{H}_{sk}(x))$  且  $t = ((x, s, \psi), t_c)$  是单射标签. 为简化起见, 如果密文  $C = (x, s, \psi, \tau, t_c)$  中  $x \in X \setminus L$ , 则称该密文是非法的. 将敌手提交第一个非法密文之前获得的所有信息记做  $\text{view}_{\mathcal{A}}$ . 注意到, 敌手只

可能从公钥中的  $pk$ , 挑战密文  $C^*$  和  $\lambda$  比特的私钥泄漏中获得有关私钥的信息. 由此可得

$$\begin{aligned}\tilde{H}_\infty(H_{sk}(x)|\text{view}_{\mathcal{A}}) &= \tilde{H}_\infty(H_{sk}(x)|pk, x, C^*, \lambda\text{-leakage}) \\ &\geq \tilde{H}_\infty(H_{sk}(x)|pk, x, C^*) - \lambda \\ &\geq H_\infty(H_{sk}(x)|(pk, x)) - \lambda - \ell_{LF} - m \end{aligned}\quad (5.5)$$

$$\geq \nu - \lambda - \ell_{LF} - m \quad (5.6)$$

其中公式 (??) 的结论依据以下事实: 在挑战密文  $C^*$  中, 仅有  $\psi^*$  和  $\tau^*$  两部分可能泄漏私钥信息, 而  $\psi^*$  和  $\tau^*$  分别只有  $2^m$  和  $2^{\ell_{LF}}$  个可能的取值. 特别指出,  $t_c^*$  可能泄漏私钥的信息完全取决于  $\psi^*$ , 因为  $t_c^* = \text{LF.LTag}(td_{LF}, (x^*, s^*, \psi^*))$  可以看做是  $\psi^*$  的函数. 公式 (??) 的结果依据哈希证明系统的性质及引理 2.2. 也就是说对于哈希证明系统所有的公钥  $pk$  及  $x \in X \setminus L$ , 都有  $H_\infty(H_{sk}(x)|(pk, x)) \geq \log(1/\epsilon_1) = \nu$ . 因为有损过滤器工作在单射模式下, 所以  $\tilde{H}_\infty(\text{LF}_{ek_{LF}, t}(H_{sk}(x))|\text{view}_{\mathcal{A}}) \geq \nu - \lambda - \ell_{LF} - m$ . 这说明在 Game<sub>4</sub> 中, 解密规则接受第一个非法密文的概率最多为  $2^{\lambda + \ell_{LF} + m}/2^\nu$ . 通过被拒绝解密, 敌手每次最多可以排除一个可能的  $\tau$  值, 所以第  $i$  个非法密文被接受的概率最多为  $2^{\lambda + \ell_{LF} + m}/(2^\nu - i + 1)$ . 因为  $\mathcal{A}$  最多询问  $Q(\kappa)$  次解密服务, 所以

$$\Pr[\text{bad}_C|\overline{F}] \leq \frac{Q(\kappa)2^{\lambda + \ell_{LF} + m}}{2^\nu - Q(\kappa)}. \quad (5.7)$$

若  $\lambda \leq \nu - m - \ell_{LF} - \omega(\log \kappa)$ , 则上面的概率是可忽略的.

公式 (??) 证毕! □

根据公式 (??) 和公式 (??), 可以直接得到引理 ?? 的结论.

引理 ?? 证毕! □

 **笔记** 上述方案容忍密钥的泄漏量最大为  $\lambda = \log 1/\epsilon_1 - m - \ell_{LF} - \omega(\log \kappa)$ , 与哈希证明系统的参数  $\epsilon_1$  和一次损耗过滤器的参数  $\ell_{LF}$  密切相关. 若哈希证明系统“足够强”, 如哈希证明系统作用在元素  $x \in X \setminus L$  上的哈希值在空间  $\Pi$  上是均匀分布的, 则有  $\epsilon_1 = 1/|\Pi|$ . 在这种情况下,  $\nu = \log(1/\epsilon_1) = \log |\Pi|$ . 用  $L$  表示哈希证明系统的私钥长度. 一般地, 1-universal HPS 的哈希值空间大小  $\log |\Pi|$  要小于甚至只有私钥长度的一半. 例如, 构造 4.9 中的哈希证明系统,  $|\Pi| = \log q$ ,  $L = 2 \log q$ . 因此, 当私钥长度  $L$  足够大且参数  $\ell_{LF}$  不变时, 私钥泄漏比率接近  $(\log |\Pi|)/L$ . 因此, 设计性能良好的 HPS 和 OT-LF 对于提高方案的密钥泄漏比率至关重要.

### 5.1.3.3 基于 DDH 问题的实例化方案

下面以 DDH 问题为例, 简要介绍如何实现所需的 1-universal HPS 和 OT-LF.

**基于 DDH 假设的并行哈希证明系统.** 在 HPS 的构造 4.9 基础上, 可以利用并行化技术, 基于一个固定的有限循环群设计一个私钥空间足够大且哈希值空间与私钥空间比率固定的强安全 1-universal HPS. 在构造 4.9 中, 哈希值空间为  $\mathbb{G}$  且  $\epsilon_1 = 1/\log q$ . 利用图 ?? 的并行化方法, 选择  $n$  个构造 4.9 中的 HPS 公钥  $pk_i$  和私钥  $sk_i$  可以得到一个私钥空间为  $(\mathbb{Z}_q \times \mathbb{Z}_q)^n$ , 哈希值空间为  $\mathbb{G}^n$  且  $\epsilon_1 = 1/q^n$  的哈希证明系统.

**基于 DDH 假设的一次有损过滤器.** 令  $(\mathbb{G}, q, g)$  是一个有限循环群,  $\text{CH} : \{0, 1\}^* \times \mathcal{R}_{\text{CH}} \rightarrow \mathbb{Z}_q$  表示一个变色龙哈希函数. 利用同态矩阵加密和变色龙哈希函数的思想, 可以设计一个  $(\mathbb{Z}_q^n, \log q)$ -一次有损过滤器, 其标签空间与变色龙哈希函数的定义域相同, 即  $\mathcal{T} = \{0, 1\}^* \times \mathcal{R}_{\text{CH}}$ . 设计的基本思路是构造一个如图 ?? 所示的公钥矩阵  $E$ , 其中  $r_i, s_i \xleftarrow{R} \mathbb{Z}_q$ ,  $b^*$  可以看作是一个嵌入公钥矩阵  $E$  中的有损标签的变色龙哈希函数值  $b^* = \text{CH}(t_a^*, t_c^*)$  计算而来.

对于 OT-LF 定义域上的任意元素  $X = (X_1, X_2, \dots, X_n) \in \mathbb{Z}_q^n$  和任意标签  $b \in \mathbb{Z}_q$ , OT-LF 的运算方式为  $y = X \cdot (E \otimes g^{bI})$ . 其中  $I$  表示  $n \times n$  阶单位矩阵, 运算符  $\otimes$  表示矩阵对应位置元素两两相乘. 对于矩阵  $E = (E_{i,j}) \in \mathbb{G}^{n \times n}$ ,  $X \cdot E$  的运算方式为

$$X \cdot E = \left( \prod_{i=1}^n E_{i,1}^{X_i}, \prod_{i=1}^n E_{i,2}^{X_i}, \dots, \prod_{i=1}^n E_{i,n}^{X_i} \right).$$

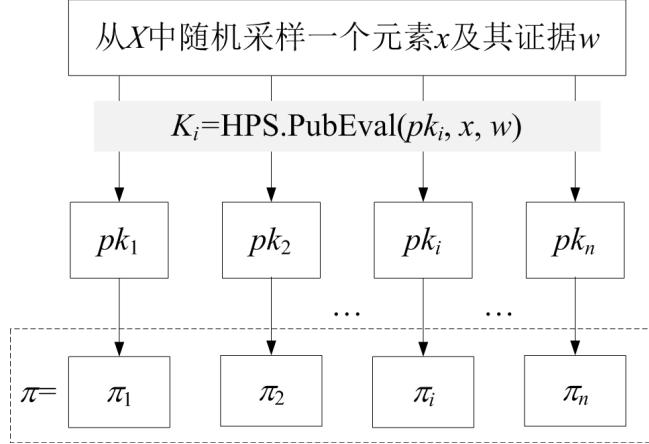


图 5.2: 并行化哈希证明系统构造示意图

$$E = \begin{pmatrix} g^{r_1 s_1} \cdot g^{-b^*} & g^{r_1 s_2} & \cdots & g^{r_1 s_n} \\ g^{r_2 s_1} & g^{r_2 s_2} \cdot g^{-b^*} & \cdots & g^{r_2 s_n} \\ \vdots & \vdots & \ddots & \vdots \\ g^{r_n s_1} & g^{r_n s_2} & \cdots & g^{r_n s_n} \cdot g^{-b^*} \end{pmatrix}$$

图 5.3: 一次有损过滤器的公钥矩阵

由此可见, 当  $b = b^*$  时, OT-LF 的值由  $g^{\sum_{i=1}^n r_i X_i}$  完全确定了, 此时 OT-LF 工作在有损模式下. 因此, 对于有损标签, OT-LF 值泄漏  $X$  的信息量不超过  $\log q$  比特. 当  $b \neq b^*$  时,  $X_i$  由  $g^{\sum_{i=1}^n r_i X_i} \cdot g^{(b-b^*)X_i}$  完全确定, 此时 OT-LF 工作在单射模式下. 此外, 利用变色龙哈希函数的性质, 对于任意  $t_a \in \{0, 1\}^*$ , 可以利用变色龙哈希函数的陷门及  $(t_a^*, t_c^*)$  计算出另一个有损标签  $(t_a, t_c)$  使得  $\text{CH}(t_a, t_c) = \text{CH}(t_a^*, t_c^*)$ .

**笔记** 根据上述基于 DDH 问题构造的 1-universal HPS 和 OT-LF 的性质, 我们可以得到一个容忍密钥泄漏量为  $L(1/2 - o(1))$  的 BKL-CCA 安全公钥加密方案. 类似地, 基于 DCR 问题也可构造容忍同等水平密钥泄漏的 BKL-CCA 安全公钥加密方案. 然而, 要到达最优密钥泄漏量  $L(1 - o(1))$ , 我们需要基于一些特殊的困难问题, 如加强的子群不可区分问题 (Refined Subgroup Indistinguishability Problem) [Qin-PKC-2014], 来设计哈希证明系统及一次有损过滤器.

### 5.1.4 CQX18 方案

2018 年, Chen 等人 [Chen-TCS-2018; Chen-CTRSA-2018] 提出规则有损函数的概念并用于设计抗泄漏密码学原语, 包括抗泄漏单向函数、抗泄漏消息认证码、抗泄漏密钥封装方案等.

#### 5.1.4.1 规则有损函数及扩展

**规则有损函数.** 下面给出规则有损函数 (Regular Lossy Functions) 的形式化定义.

##### 定义 5.3 (规则有损函数)

假设定义域为  $2^{n(\kappa)}$ , 其中  $n(\kappa) = \text{poly}(\kappa)$ . 定义  $\nu(\kappa) \leq 2^{n(\kappa)}$  表示非单射集合,  $2^{\tau(\kappa)} \leq 2^{n(\kappa)}$  表示像空间. 一个  $(\nu, \tau)$ -RLFs 由以下四个多项式时间算法组成并满足以下性质:

- $\text{Setup}(\kappa)$ : 以安全参数  $\kappa$  为输入, 输出公开参数  $pp$ , 其中  $pp$  包含对求值公钥空间  $EK$ , 定义域  $X$  和值域  $Y$  的描述.
- $\text{GenNormal}(pp)$ : 以公开参数  $pp$  为输入, 输出求值公钥  $ek$ .  $f_{ek}(\cdot) : X \rightarrow Y$  是一个  $\nu$ -规则函数.

- $\text{GenLossy}(pp)$ : 以公开参数  $pp$  为输入, 输出求值公钥  $ek$ .  $f_{ek}(\cdot) : X \rightarrow Y$  是一个有损函数, 像空间最大为  $2^\tau$ . 损耗定义为  $n - \tau$ .
- $\text{Eval}(ek, x)$ : 以求值公钥  $ek$  和原像  $x \in X$  为输入, 输出  $y \leftarrow f_{ek}(x)$ .

模式不可区分性. 对于任意公开参数  $pp \leftarrow \text{Setup}(\kappa)$ ,  $\text{GenNormal}(pp)$  和  $\text{GenLossy}(pp)$  输出的求值公钥在计算意义上不可区分.



图 ?? 给出了规则有损函数的示意图. 在  $\nu$ -标准模式中, 每个像都有  $2^\nu$  相同大小的原像空间. 相应地, 像空间也缩减为  $2^{n-\nu}$ . 在  $\tau$ -有损模式中, 像空间大小仅有  $2^\tau$ , 并且  $2^n \gg 2^\tau$ .

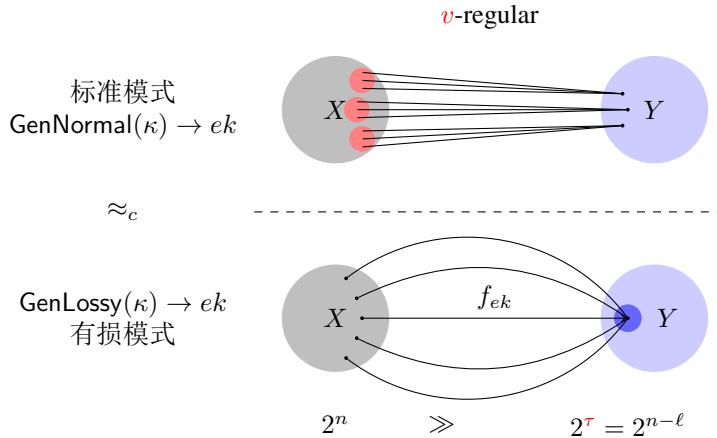


图 5.4: 规则有损函数 (RLFs) 示意图



**笔记** 规则有损函数可以看作是有损函数的一般化形式. 当  $\nu = 1$  时, 规则有损函数即是有损函数. “规则有损 (regular lossy)” 这一概念在 [KPS-CRYPTO-2013; Seurin-PKC-2014] 等文献中也有介绍. 但是与本文中的概念区别较大. 前者要求有损模式是规则有损的, 而本文要求的是在标准模式中是(近似) 规则有损的. 对于一个近似规则有损函数, 在标准模式下, 函数值的熵几乎保存了原像的熵, 可以很容易得到如下的引理:

### 引理 5.3

假设  $f : D \rightarrow R$  是一个  $\nu$ -到-1 规则函数,  $X$  是定义域  $D$  熵的一个随机变量, 则有:

$$H_\infty(f(X)) \geq H_\infty(X) - \log \nu$$



**全除一规则有损函数.** 类似于全除一有损函数 (ABO-LFs), 规则有损函数也可以推广为全除一规则有损函数 (ABO-RLFs), 其形式化定义见定义 ??.

### 定义 5.4 (全除一有损函数)

在定义中, 参数  $n, \nu, \tau$  的含义同规则有损函数,  $B$  表示分支空间. 一个  $(\nu, \tau)$ -ABO-RLFs 包含以下三个多项式时间算法并满足以下性质:

- $\text{Setup}(\kappa)$ : 以安全参数  $\kappa$  为输入, 输出公共参数  $pp$ , 其中  $pp$  包含对求值密钥空间  $EK$ , 分支空间  $B$ , 定义域  $X$  和值域  $Y$  的描述.
- $\text{Gen}(pp, b^*)$ : 以公开参数  $pp$  和分支  $b^* \in B$  为输入, 输出一个求值密钥  $ek$ . 对于任意  $b \neq b^*$ ,  $f_{ek,b}(\cdot)$  是一个从  $X$  到  $Y$  的  $\nu$ -规则函数, 而  $f_{ek,b^*}(\cdot)$  是一个从  $X$  到  $Y$  的有损函数, 其像空间大小最多为  $2^\tau$ .
- $\text{Eval}(ek, b, x)$ : 以求值密钥  $ek$ , 分支  $b \in B$  和  $x \in X$  为输入, 输出  $y \leftarrow f_{ek,b}(x)$ .

隐藏有损分支性质. 对于任意  $b_0^*, b_1^* \in B \times B$ ,  $\text{Gen}(pp, b_0^*)$  输出的求值密钥  $ek_0$  和  $\text{Gen}(pp, b_1^*)$  输出的求值

密钥  $ek_1$  在计算上是不可区分的. are computationally indistinguishable.



图 ??给出了全除一有损函数的示意图. 由图示可知, 每个求值函数都会额外输入一个分支  $b$ , 并且有损分支  $b^*$  隐藏于求值密钥  $ek$  中. 当  $b = b^*$  时, 函数才处于有损模式, 其他情况都是规则模式.

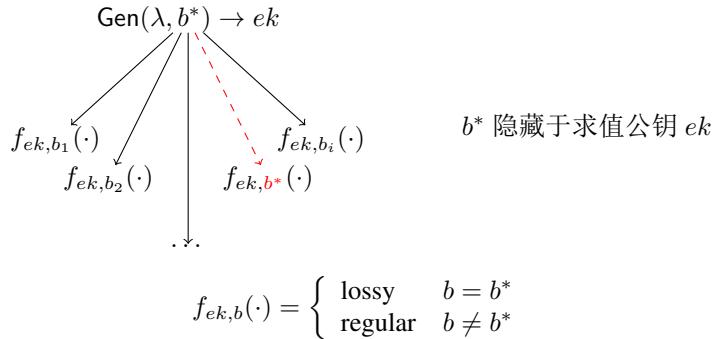


图 5.5: 全除一规则有损函数 (ABO-RLFs) 示意图

**一次规则有损过滤器.** 在 ABO-RLFs 中, 求值公钥  $ek$  完全确定了有损分支  $b^*$  的值. 因此, 在归约证明中, 有损分支需要提前选取, 并不适应于自适应攻击的敌手环境. 类似前面的 OT-LFs, 我们可以将 RLFs 推广到一次规则有损过滤器 (One-Time Regular Lossy Filters, 简称 OT-RLFs). OT-RLFs 的形式化定义见定义 definition:ch5-OT-RLFs.

### 定义 5.5 (一次规则有损过滤器)

一个  $(\nu, \tau)$ -OT-RLFs 包含以下四个多项式时间算法并满足以下性质:

- $\text{Setup}(\kappa)$ : 以安全参数  $\kappa$  为输入, 输出公开参数  $pp$ , 其中  $pp$  包含对求值密钥空间  $EK$ , 分支集合  $B = B_c \times B_a$  (其中,  $B_c$  是核心分支集合,  $B_a$  是辅助输入分支集合), 定义域  $X$  和值域  $Y$  的描述.
- $\text{Gen}(pp)$ : 以公开参数  $pp$  为输入, 输出一个求值密钥  $ek$  和一个陷门  $td$ . 分支集合  $B$  包含两个不相交的子集: 规则分支集合  $B_{\text{normal}}$  和有损分支集合  $B_{\text{lossy}}$ . 对于规则分支集合中的任意分支  $b \in B_{\text{normal}}$ ,  $f_{ek,b}(\cdot)$  确定了一个从  $X$  到  $Y$  的  $\nu$ -规则函数. 对于有损分支集合中的任意分支  $b \in B_{\text{lossy}}$ ,  $f_{ek,b}(\cdot)$  确定了一个从  $X$  到  $Y$  且像空间最大为  $2^\tau$  的有损函数.
- $\text{SampLossy}(td, b_a)$ : 以陷门  $td$  和一个辅助分支  $b_a$  为输入, 输出一个核心分支  $b_c$ , 使得  $b = (b_c, b_a)$  是集合  $B_{\text{lossy}}$  中的一个有损分支.
- $\text{Eval}(ek, b, x)$ : 以求值密钥  $ek$ , 分支  $b \in B$  和元素  $x \in X$  为输入, 输出  $y \leftarrow f_{ek,b}(x)$ .

**Indistinguishability.** 对于任意辅助分支  $b_a \in B_a$ , 由算法生成的核心分支  $b_c \leftarrow \text{SampLossy}(td, b_a)$  与随机选取的核心分支  $b_c \stackrel{R}{\leftarrow} B_c$  在计算意义上是不可区分的.

**Evasiveness.** 对于任意 PPT 敌手, 在给定一个有损分支的条件下, 再生成一个新的有损分支是困难的.



### 注记 5.1

规则有损函数的几个相关概念之间存在一定的联系. 我们知道, ABO-RLFs 是 RLFs 的推广. 实际上, 若存在一个  $(\nu, \tau)$ -ABO-RLFs 函数族, 我们只需要把 ABO-RLFs 的分支标签作为 RLFs 的求值密钥参数, 即可以得到一个  $(\nu, \tau)$ -RLFs 函数族. 相反, 类似 LTFs 与 ABO-LTFs 的转化关系 [PW-STOC-2008], 若存在一个  $(\nu, \tau)$ -RLFs 函数族, 则必然存在一个分支集合为  $B = \{0, 1\}$  的  $(\nu, \tau)$ -ABO-RLFs 函数族. 进一步, 可以推广到分支集合为  $B = \{0, 1\}^\ell$  的  $(\nu, \ell\tau)$ -ABO-RLFs 函数族. 此外, 根据文献 [Qin-PKC-2014], 一个  $(\nu, \tau)$ -ABO-RLFs 函数族结合一个变色龙哈希函数, 可以构造一个  $(\nu, \tau)$ -OT-RLFs 函数族.



### 5.1.4.2 全除一规则有损函数的具体构造

本节分别基于 DDH 问题和 DCR 问题, 介绍 ABO-RLFs 函数族的两种具体构造.

**基于 DDH 的 ABO-RLFs.** 首先回顾群上的伪随机矩阵生成算法 GenConceal. 该算法的输入是两个正整数  $n$  和  $m$  (其中  $n \geq m$ ), 输出是一个群元素构成的秩为 1 的  $n \times m$  矩阵  $\mathbb{G}^{n \times m}$  且与随机选取的  $n \times m$  矩阵不可区分. 该算法的具体执行过程如下:

- 随机选择两个向量  $\mathbf{r} = (r_1, \dots, r_n) \leftarrow \mathbb{Z}_p^n$  和  $\mathbf{s} = (s_1, \dots, s_m) \leftarrow \mathbb{Z}_p^m$ .
- 令  $\mathbf{V} = \mathbf{r} \otimes \mathbf{s} = \mathbf{r}^t \mathbf{s} \in \mathbb{Z}_p^{n \times m}$  表示  $\mathbf{r}$  与  $\mathbf{s}$  的外积.
- 输出  $\mathbf{C} = g^{\mathbf{V}} \in \mathbb{G}^{n \times m}$  作为 concealer matrix.

#### 引理 5.4 ([PW2008])

令  $n, m = \text{poly}(\lambda)$ . 在 DDH 假设下, 矩阵  $\mathbf{C} = g^{\mathbf{V}} \leftarrow \text{GenConceal}(n, m)$  在空间  $\mathbb{G}^{n \times m}$  上是伪随机的.



基于 DDH 假设的 ABO-RLFs 构造如下:

#### 构造 5.3 (基于 DDH 的 ABO-RLFs)

- **Setup( $\kappa$ ):** 以安全参数  $\kappa$  为输入, 运行  $(\mathbb{G}, g, p) \leftarrow \text{GroupGen}(\lambda)$ , 输出公开参数  $pp = (\mathbb{G}, g, p)$  和分支空间  $B = \mathbb{Z}_p$ .
- **Gen( $pp, b^*$ ):** 以公开参数  $pp$  和分支  $b^* \in \mathbb{Z}_p$  为输入, 调用算法  $\text{GenConceal}(n, m)$  生成矩阵  $\mathbf{C} = g^{\mathbf{V}} \in \mathbb{G}^{n \times m}$ , 输出求值密钥  $ek = g^{\mathbf{Y}} = g^{\mathbf{V}-b^*\mathbf{I}'}$ , 其中  $\mathbf{I}' \in \mathbb{Z}_p^{n \times m}$ , 即, 第  $i$  列向量是标准的单位向量  $\mathbf{e}_i \in \mathbb{Z}_p^n$ , 其中  $i \leq n$ , 而余下的列向量为零向量.
- **Eval( $ek, b, \mathbf{x}$ ):** 以求值密钥  $ek = g^{\mathbf{Y}}$ , 分支  $b \in \mathbb{Z}_p$  和元素  $\mathbf{x} \in \mathbb{Z}_p^n$  为输入, 输出  $\mathbf{y} = g^{\mathbf{x}(\mathbf{Y}+b\mathbf{I}')} = g^{\mathbf{x}(\mathbf{V}+(b-b^*)\mathbf{I}')} \in \mathbb{G}^m$ .



#### 引理 5.5

在 DDH 假设下, 构造 ?? 是一个  $(p^{n-m}, \log p)$ -ABO-RLFs, 其中  $n > 1$ .



**证明** 对于任意  $b \neq b^*$ ,  $(\mathbf{V}, b)$  确定了一个  $p^{n-m}$ -到-1 函数, 这是因为矩阵  $(\mathbf{Y} + b\mathbf{I}')$  的秩是  $m$ , 对于每个  $y \in \mathbb{G}^m$ , 其解空间大小为  $p^{n-m}$ . 当  $b = b^*$  时, 每个输出结果  $\mathbf{y}$  的形式是  $g^{r'\mathbf{s}}$ , 其中  $r' = \mathbf{x}\mathbf{r}^t \in \mathbb{Z}_p$ . 因为  $\mathbf{s}$  由函数索引  $\mathbf{V}$  确定, 所以由  $(\mathbf{V}, b^*)$  确定的每个函数最多有  $p$  个不同的输出结果. 故, 损耗为  $(n-1)\log p$ .

在 DDH 假设下, 通过归约可以证明隐藏有损分支性质: 对于任意分支  $b^* \in \mathbb{Z}_p$ ,  $\text{Gen}(\lambda, b^*)$  输出的求值密钥矩阵与  $\mathbb{G}^{n \times m}$  上的随机矩阵在计算意义上是不可区分的.

#### 注记 5.2

在构造中, 参数  $n$  用于控制定义域的大小, 而参数  $m$  用于调节 ABO 分支的规则性. 当  $m = n$  时, ABO 分支是单射的, 故上述构造方案变成了标准的 ABO-LFs.

在基于 DDH 的 ABO-LTF 构造方案中 [PW-STOC-2008], 定义域限制在  $\{0, 1\}^n$  且  $m$  必须大于  $n$  才能保证函数可求逆. 在上述构造中, 由于 ABO-RLF 不需要可求逆的性质, 在不改变求值密钥矩阵大小的情况下, 定义域可以由  $\{0, 1\}^n$  扩展到  $\mathbb{Z}_p^n$ . 特别地, ABO-RLF 不需要单射性质, 所以矩阵参数  $m$  可以小于  $n$ . 在基于矩阵的构造中, 求值密钥大小和计算开销的复杂性取决于参数  $n$  和  $m$ . 因此, 与基于 DDH 的 ABO-LTFs 相比, ABO-RLFs 允许的输入空间更大, 计算效率更高.



类似地, 利用 Hemenway 和 Ostrovsky 的方法 [HO-PKC-2012b], 可以将上述基于 DDH 的构造方案扩展到基于 eDDH 的构造, 而 eDDH 问题包含了 DDH、QR 和 DCR 等问题. 所以, 上述构造也蕴含了基于 DCR 的 ABO-RLFs. 尽管如此, 直接基于 DCR 假设, 可以构造出更高效的 ABO-RLFs.

**基于 DCR 的 ABO-RLFs.** 基于 DCR 假设的 ABO-RLFs 构造如下:

### 构造 5.4 (基于 DCR 的 ABO-RLFs)

- $\text{Setup}(\kappa)$ : 运行  $N \leftarrow \text{GenModulus}(\lambda)$  生成 RSA 模  $N$ , 随机选择  $z \in \mathbb{Z}_N$  并计算  $g = z^{2N} \pmod{N^2}$ , 输出公开参数  $pp = (N, y)$ , 并令分支空间为  $B = \mathbb{Z}_N$ .
- $\text{Gen}(pp, b^*)$ : 以公开参数  $pp$  和有损分支  $b^* \in \mathbb{Z}_N$  为输入, 随机选择  $r \in \mathbb{Z}_N$ , 计算并输出求值密钥  $ek = g^r(1+N)^{-b^*}$ .
- $\text{Eval}(ek, b, x)$ : 以求值密钥  $ek$ , 分支  $b \in \mathbb{Z}_N$  和元素  $x \in \{0, \dots, \lfloor N^2/4 \rfloor\}$  为输入, 输出  $y = [ek/(1+N)^b]^x(1+N)^{(b-b^*)x} \in \mathbb{Z}_{N^2}$ .



### 引理 5.6

基于 DCR 假设, 构造方案 ?? 是一个  $(1, \phi(N)/4)$ -ABO-RLFs 函数族.



**证明** 对于任意  $b \neq b^*$ ,  $f_{ek,b}$  是一个单射函数, 这是因为  $g$  以压倒性的概率是  $2N$  次剩余群的一个生成元. 令  $\phi$  表示欧拉函数. 则  $g$  的阶至少为  $\phi(N)/4$ ,  $g^r(1+N)^{b-b^*}$  的阶至少为  $N\phi(N)/4$ . 当  $b = b^*$  时, 每个输出结果  $g^{rx}$  是一个  $2N$  次剩余元素. 因此, 所有像元素至多为  $\phi(N)/4$  个, 损耗至少为  $\log N$ .

隐藏有损分支性质源于 Paillier 加密方案基于 DCR 假设的安全性 [**Paillier-EUROCRYPT-1999**]: 算法  $\text{Gen}(pp, b^*)$  的输出结果实际上是 Paillier 方案选择随机数  $r$  加密消息  $b^*$  的一密文. 因此, 对于任意  $b_0^*, b_1^* \in \mathbb{Z}_N$ ,  $\text{Gen}(pp, b_0^*)$  的输出结果和  $\text{Gen}(pp, b_1^*)$  的输出结果在计算意义上是不可区分的. 由于规则损耗参数为  $\nu = 1$ , 所以方案 ?? 实际上是一个 ABO-LFs 函数族.

#### 5.1.4.3 全除一规则有损函数的通用构造

由于全除一规则有损函数可以由规则有损函数构造. 为此, 本节重点介绍如何基于哈希证明系统构造规则有损函数.

基于哈希证明系统的一般构造. 2012 年, Wee 提出利用对偶哈希证明系统构造有损陷门函数. 对偶哈希证明系统的概念可以参考文献 [**Wee-EUROCRYPT-2012**] 或第 6 章定义 ?? . 假设  $(H, SK, PK, X, L, W, \Pi, \alpha)$  是对偶哈希证明系统的公开参数, 其中  $H : X \times SK \rightarrow \Pi$ . Wee 基于对偶哈希证明系统的有损陷门函数  $f$  构造方式如公示 (??) 所示.

$$f_x(sk) = \alpha(sk) \parallel H_x(sk) \quad (5.8)$$

在上述构造中, 哈希证明系统的密钥  $sk \in SK$  充当了有损陷门函数的定义域, 而子集成员  $x \in X$  充当了函数的求值密钥. 当  $x \in X \setminus L$  时, 根据对偶哈希证明系统的可逆性质,  $f_x$  是单射函数且可逆; 根据对偶哈希证明系统的仿射性质, 当  $x \in L$  时, 函数  $f_x$  是有损函数. 基于子集成员判定困难问题, 这两种模式在计算意义上是不可区分的.

有趣的是, 利用上述构造方式, 可以基于任意的哈希证明系统构造规则有损函数. 由于规则有损函数比有损陷门函数的性质要弱, 仅需要哈希证明系统的仿射性质即可, 并不需要哈希证明其他额外的性质, 比如平滑性、一致性、可逆性等. 令  $(H, SK, PK, X, L, W, \Pi, \alpha)$  是哈希证明系统的一个公开参数. 对于任意  $x \in X \setminus L$ , 假设  $f_x(sk) = \alpha(sk) \parallel H_x(sk)$  是一个从定义域  $SK$  到值域  $\Pi$  的  $\nu$ -到-1 函数. 我们有以下引理

### 引理 5.7

基于子集成员判定问题假设, 公式 (??) 是一个  $(\nu, \log |\text{Img}(\alpha)|)$ -规则有损函数.



**证明** 标准模式的正确性源于  $f_x(\cdot)$  是一个  $\nu$ -到-1 函数. 有损模式的损耗性源于哈希证明系统的仿射性质. 对于任意  $x \in L$ ,  $\text{Img}(f_x) = \text{Img}(\alpha)$ . 两种模式的不可区分性可以归约到子集成员判定问题的困难性.

至此, 利用哈希证明系统构造全除一规则有损函数可以通过以下两步实现: (1) 利用哈希证明系统构造一个规则有损函数族; (2) 将具有二元分支空间  $\{0, 1\}$  的规则有损函数放大到分支空间为  $\{0, 1\}^\ell$  的全除一规则有损函数. 然而, 第二步扩大分支空间的效率并不高, 需要进行  $\ell$  次独立地规则有损函数计算, 且损耗量也会降低.

**基于哈希证明系统的高效构造.** 在前面的构造方案中, 哈希证明系统所基于的子集成员判定问题没有任何代数结构限制. 如果子集成员判定问题具有一定的代数结构, 则可以构造更加高效的 ABO-RLFs 函数族. 构造的主要思想是利用.

**代数子集成员判定问题 (ASMPs).** 该问题是一种特殊的子集成员判定问题  $(X, L)$ , 满足以下几个性质:

1.  $X$  是一个有限阿贝尔群,  $L$  是  $X$  的一个子群.
2. 商群  $H = X/L$  是一个阶为  $p = |X|/|L|$  的循环群.

基于上述性质, 我们可以推出以下两个实用的结论:

- 令  $\bar{a} = aL$ , 其中  $a \in X \setminus L$  是  $H$  的一个生成元, 则陪集  $(aL, 2aL, \dots, (p-1)aL, paL = L)$  构成了  $X$  的一个划分.
- 对于任意  $x \in L, ia + x \in X \setminus L$ , 其中  $1 \leq i < p$ .

ASMPs 问题的困难性同 SMPs 问题一样, 要求集合  $L$  和  $X \setminus L$  上的元素分布在计算意义下是不可区分的.  $L$  的密度定义为  $\rho = |L|/|X|$ . 当  $\rho$  是可忽略时,  $U_L \approx_c U_{X \setminus L}$  等价于  $U_L \approx_c U_X$ . 此时,  $U_{X \setminus L}$  和  $U_X$  统计距离接近. 当  $\rho$  已知时,  $U_L \approx_c U_{X \setminus L}$  意味着  $U_L \approx_c U_X$ , 这是因为  $U_X$  可由  $U_L, U_{X \setminus L}$  和  $\rho$  重构出来.

### 注记 5.3

ASMP 问题具有一般性, 可由 DDH、 $d$ -linear、QR 和 DCR 等假设构造. 此外, ASMP 问题可以看作是满足性质 2 的加强子群成员判定问题. 在实际应用中, 性质 2 可以放宽至  $H$  包含一个循环子群.

与 (加强) 子群不可区分问题的比较. 2010 年, Brakerski 和 Goldwasser [BG-CRYPTO-2010] 提出子群不可区分问题 (简称 SIPs 问题). SIPs 问题定义在一个阿贝尔群  $X$  和子群  $L$  上. 此外, SIPs 要求  $X$  同构于两个群的直积, 即  $X \simeq L \times M$  且  $\gcd(\text{ord}(L), \text{ord}(M)) = 1$ . 2014 年, Qin 和 Liu [Qin-PKC-2014] 提出加强的子群不可区分问题 (简称 RSIPs), 进一步要求  $M$  也是一个循环子群. 与 RSIPs 问题相比, ASMPs 问题仅要求商群  $X/L$  是循环的. 因此, ASMP 问题要强于 RSIP 问题, 按理也比 SIP 问题强, 因为 SIP 问题无法由 DDH 和  $d$ -linear 等问题构造. 由此可见, 代数子集成员判定假设要弱于 RSIP 和 SIP 等假设..

下面给出基于 ASMP-HPS 的 ABO-RLF 的构造.

### 构造 5.5

假设 HPS 是一个基于 ASMP 问题的哈希证明系统, 则 ABO-RLF 的具体构造如下:

- $\text{Setup}(\kappa)$ : 运行  $\text{HPS}.\text{Setup}(\kappa)$  生成 HPS 的公开参数  $pp = (\mathsf{H}, SK, PK, X, L, W, \Pi, \alpha)$ , 选择商群  $H$  的一个随机生成元  $aL$ , 输出公开参数  $\hat{pp} = (pp, a)$ .
- $\text{Gen}(\hat{pp}, b^*)$ : 以公开参数  $\hat{pp} = (pp, a)$  和一个有损分支  $b^* \in \mathbb{Z}_p$  为输入, 运行  $(x, w) \leftarrow \text{HPS}.\text{SampYes}(pp)$  抽取  $L$  中的一个随机元素, 计算求值密钥  $ek = -b^*a + x \in X$ .
- $\text{Eval}(ek, b, sk)$ : 以求值密钥  $ek = -b^*a + x$ , 分支  $b$  和  $sk$  为输入, 计算并输出  $\alpha(sk) \parallel \mathsf{H}_{sk}(ek + ba)$ . 该算法定义了  $f_{ek,b}(sk) := \alpha(sk) \parallel \mathsf{H}_{sk}(ek + ba)$ .

### 定理 5.3

假设  $X = \{0, 1\}^n$ . 对于任意  $x \notin L$ , 函数  $f_x(sk) = \alpha(sk) \parallel \mathsf{H}_x(sk)$  是一个  $\nu$ -规则函数. 上述构造是一个基于代数子集成员判定问题的  $(\nu, \log |\text{Img}\alpha|)$ -ABO-RLFs 函数族.

**证明** 根据 ASMP 问题的性质, 当  $b \neq b^*$  时,  $ek + ba = x + (b - b^*)a \notin L$ . 此时,  $f_{ek,b}(\cdot)$  是一个  $\nu$ -规则函数. 当  $b = b^*$  时,  $ek + ba = x + (b - b^*)a = x \in L$ . 此时,  $f_{ek,b}(\cdot)$  是一个有损函数. 在安全性方面, 隐藏有损分支性质源于代数子集成员判定问题的困难性, 即对于任意  $b_0^*, b_1^* \in \mathbb{Z}_p$ ,  $(-b_0^*a + x) \approx_c (-b_1^*a + u) \equiv u \equiv (-b_1^*a + u) \approx_c (-b_1^*a + x)$ , 当  $u \leftarrow X$ . 定理得证!  $\square$

### 5.1.4.4 规则有损函数的应用

有损陷门函数在标准模式下具有求逆陷门和单射性, 因此可以用于构造公钥加密方案、抗碰撞哈希函数等高级密码学原语. 规则有损函数在这些应用方面似乎无用武之地. 然而, 在泄漏密码学领域, 规则有损函数及其重要的应用价值. Chen 等人 [Chen-TCS-2018] 指出规则有损函数可以用于构造抗泄漏单向函数 (Leakage-Resilient OWFs)、抗泄漏消息认证码 (Leakage-Resilient MACs)、抗泄漏和选择密文攻击安全的密钥封装方案 (Leakage-Resilient CCA-secure KEM) 等高级密码学原语. 下面对这些构造分别作一简要介绍.

**抗泄漏单向函数.** 类似有损陷门函数蕴含一个单向陷门函数, 规则有损陷门函数则蕴含一个抗泄漏单向函数, 有以下定理保证.

#### 定理 5.4

假设  $\text{RLF} = (\text{Setup}, \text{GenInj}, \text{GenLossy}, \text{Eval})$  是一个定义在  $\{0,1\}^n$  上的规则有损函数族, 其有损模式的像空间大小最多为  $2^\tau$ . 则  $(\text{Setup}, \text{GenInj}, \text{Eval})$  是一个  $\ell$ -抗泄漏单向函数, 其中  $\ell \leq n - \tau - \omega(\log \kappa)$ .



关于抗泄漏单向函数的形式化定义及安全模型可以参考文献 [Chen-TCS-2018] 的附录 A.1. 下面简要介绍该定理的证明.

**证明** 我们通过游戏序列的方式组织证明. 令  $S_i$  表示事件 “ $\mathcal{A}$  在游戏  $i$  中成功”.

**Game 0.** 该游戏是标准的抗泄漏单向函数游戏, 其中挑战者  $\mathcal{CH}$  与对手  $\mathcal{A}$  按以下方式交互:

1. 初始化:  $\mathcal{CH}$  生成 (标准模式) 求值密钥  $ek \leftarrow \text{GenInj}(\kappa)$ , 随机选择  $x^* \xleftarrow{\text{R}} \{0,1\}^n$  并计算  $y^* \leftarrow f_{ek}(x^*)$ , 然后将  $(ek, y^*)$  发送给对手  $\mathcal{A}$  作为挑战信息.
2. 泄漏询问:  $\mathcal{A}$  可以自适应地进行泄漏询问. 对于任意泄漏询问  $\langle g \rangle$ ,  $\mathcal{CH}$  返回  $g(x^*)$ .
3. 求逆:  $\mathcal{A}$  输出  $x$ , 如果  $x = x^*$  则  $\mathcal{A}$  成功.

根据定义, 则有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = \Pr[S_0]$$

**Game 1.** 该游戏与 Game 0 唯一不同之处在于第 1 步:

1. 初始化:  $\mathcal{CH}$  生成 (有损模式) 求值密钥  $ek \leftarrow \text{GenLossy}(\kappa)$ .

根据两种模式的不可区分性, 则有:

$$|\Pr[S_1] - \Pr[S_0]| \leq \text{negl}(\kappa)$$

下面重点分析游戏 1 中的概率  $\Pr[S_1]$ . 假设  $ek$  是任意一个由  $\text{GenLossy}(\kappa)$  生成的求值密钥. 对于任意对手, 成功猜测  $x^*$  的概率完全由  $x^*$  的平均极小熵. 特别地, 在已知  $y^* \leftarrow f_{ek}(x^*)$  和  $x^*$  的部分泄漏信息条件下,  $x^*$  的平均极小熵为  $2^{-\tilde{H}_\infty(x^*|(y^*, leak))}$ . 由于  $f_{ek}(\cdot)$  的输出结果最多有  $2^\tau$  种可能值, 泄漏量最多为  $2^\ell$ , 则有

$$\tilde{H}_\infty(x^*|(f_{ek}(x^*), leak)) \geq H_\infty(x) - \tau - \ell = n - \tau - \ell$$

由于  $n - \tau - \ell \geq \omega(\log \lambda)$ , 则  $\mathcal{A}(ek, y^*, leak)$  输出  $x^*$  的概率是可忽略的  $\text{negl}(\lambda)$ . 该结论对于由  $\text{GenLossy}$  随机生成的求值密钥  $ek$  也成立. 这就证明了  $\Pr[S_1] = \text{negl}(\kappa)$ , 从而  $\Pr[S_0]$  也是可忽略的. 定理得证!  $\square$

**抗泄漏消息认证码.** 消息认证码是一种重要的密码学原语, 在认证协议、公钥加密方案设计等方面具有重要的应用. 关于抗泄漏消息认证码的形式化定义及其安全模型可以参考文献 [Chen-TCS-2018] 的附录 A.2. 下面介绍如何利用 ABO-RLF 或者 OT-RLF 可以构造抗泄漏消息认证码. 利用 ABO-RLF 构造 MAC 的主要思路将 ABO-RLF 的定义域输入看作 MAC 的密钥, 而将输入的分支看作消息, 其输出作为 MAC 的标签, 如图 ?? 所示.

#### 构造 5.6 (基于 ABO-RLF 的抗泄漏消息认证码)

- $\text{Setup}(\kappa)$ : 运行  $\text{ABORLF}.\text{Setup}(\kappa)$  生成 RLF 的公开参数  $pp = (EK, B, X, Y)$ , 其中  $|X| = 2^n$  和  $B = \{0,1\}^b$ , 运行  $\text{ABORLF}.\text{Gen}(pp, 0^b)$  生成 RLF 的求值密钥  $ek$ , 输出 MAC 的公开参数  $\hat{pp} = (pp, ek)$ . 密

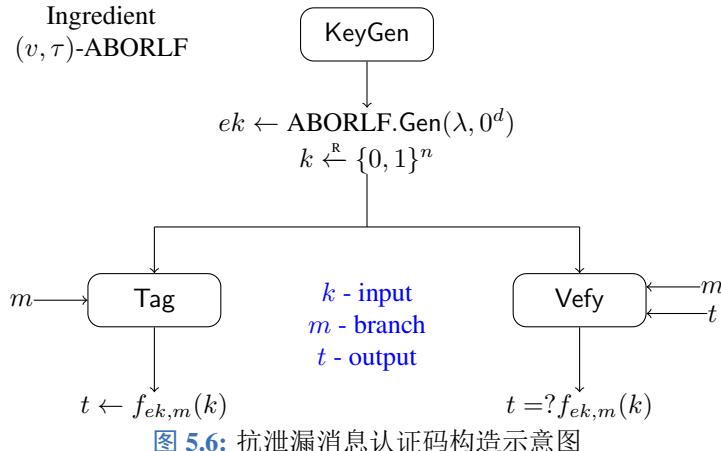


图 5.6: 抗泄漏消息认证码构造示意图

钥空间、消息空间和认证码空间分别定义为  $K = X$ 、 $M = B$  和  $T = Y$ .

- $\text{Gen}(\hat{pp})$ : 随机选择  $k \xleftarrow{R} X$  作为 MAC 的密钥.
- $\text{Tag}(k, m)$ : 以密钥  $k$  和消息  $m$  为输入, 计算  $t \leftarrow f_{ek,m}(k)$ , output  $(m, t)$  作为消息  $m$  的认证码.
- $\text{Vefy}(k, m, t)$ : 如果  $t = f_{ek,m}(k)$  则输出 1, 否则输出 0.



### 定理 5.5

如果 ABORLF 是一个  $(\nu, \tau)$ -ABO-RLFs 函数族, 则构造方案 ?? 是一个  $\ell$ -抗泄漏消息认证码, 其中  $\ell \leq n - \tau - \log \nu - \omega(\log \lambda)$ .



**证明** 下面通过游戏的方式组织证明. 令  $S_i$  表示事件 “ $\mathcal{A}$  在游戏  $i$  中攻击成功”.

**Game 0.** 该游戏是消息认证码的抗泄漏一次强伪造安全性游戏. 挑战者  $\mathcal{CH}$  通过以下方式与敌手  $\mathcal{A}$  交互完成游戏.

1. 承诺和初始化: 敌手  $\mathcal{A}$  在看到公开参数前, 声明一个目标消息  $\langle m^* \rangle$ .  $\mathcal{CH}$  运行  $pp \leftarrow \text{ABORLF}.\text{Setup}(\kappa)$  和  $ek \leftarrow \text{ABORLF}.\text{Gen}(pp, 0^b)$ .  $\mathcal{CH}$  选择  $k \xleftarrow{R} X$ , 计算  $t^* \leftarrow f_{ek,m^*}(k)$ , 再将  $\hat{pp} = (pp, ek)$  和  $t^*$  发送给  $\mathcal{A}$ .
2. 学习阶段:  $\mathcal{A}$  可以自适应地进行泄漏询问. 对于任意泄漏查询  $\langle g \rangle$ , 只要泄漏量不超过  $\ell$ ,  $\mathcal{CH}$  返回  $g(k)$  给  $\mathcal{A}$ .
3. 伪造:  $\mathcal{A}$  输出  $(m, t)$ . 如果  $m \neq m^*$  且  $t = f_{ek,m}(k)$ , 则  $\mathcal{A}$  成功.

根据定义, 则有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = \Pr[S_0]$$

**Game 1.** 该游戏与 Game 0 的唯一区别是第 1 步中,  $\mathcal{CH}$  通过运行  $\text{ABORLF}.\text{Gen}(pp, m^*)$  替代  $\text{ABORLF}.\text{Gen}(pp, b^*)$  生成  $ek$ . 根据 ABO-RLF 隐藏有损分支的性质, Game 0 与 Game 1 在计算上不可区分. 因此有:

$$|\Pr[S_1] - \Pr[S_0]| \leq \text{negl}(\kappa)$$

下面分析事件  $S_1$  的概率  $\Pr[S_1]$ . 由于构造的消息认证码唯一, 事件  $S_1$  实际上等价于  $\mathcal{A}$  输出  $(m, t)$ , 其中  $m \neq m^*$  且  $f_{ek,m}(k) = t$ . 则消息认证码  $t$  的条件熵由  $\mathcal{A}$  的视图  $(pp, ek, m, leak, m^*, t^*)$  决定. 具体为

$$\tilde{H}_\infty(t|view) = \tilde{H}_\infty(t|ek, m, leak, t^*) \quad (5.9)$$

$$\geq \tilde{H}_\infty(t|ek, m) - \ell - \tau \quad (5.10)$$

$$\geq n - \log \nu - \ell - \tau \quad (5.11)$$

在上面的推导过程中, 公式 (??) 依据事实  $t = f_{ek,m}(k)$  由  $ek$ 、 $m$  和  $k$  决定, 且  $k$  与  $m^*$  和  $pp$  独立. 公式 (??) 依据引理 ?? 和泄漏量的上界  $\ell$  比特, 以及认证码  $t^*$  最多有  $2^\tau$  个可能的值. 对于任意  $m \neq m^*$ ,  $f_{ek,m}(\cdot)$  是一个  $\nu$ -到-1 函数. 所以  $H_\infty(f_{ek,m}(k)) \geq H_\infty(k) - \log \nu$ , 从而公式 (??) 成立.

根据参数选择方式, 则有  $n - \log \nu - \ell - \tau \geq \omega(\log \kappa)$ . 因此,  $\Pr[S_1] \leq \text{negl}(\kappa)$  成立.

综上, 定理得证!  $\square$ .

**抗泄漏密钥封装方案.** 类似 Qin 和 Liu 在 2013 年亚密会上提出的抗泄漏公钥加密方案的设计模式, 利用 ABO-RLF 设计 CCA 安全的抗泄漏密钥封装方案也额外需要一个哈希证明系统作为封装密钥的工具. 设计思路如图 ?? 所示. 简单地说, 先利用一个哈希证明系统封装一个(随机)哈希值  $\pi$ , 再利用 ABO-RLF 计算  $\pi$  的知识证明  $t$ , 类似于 Cramer-Shoup 方案的指定验证者零知识证明. 这里的  $t$  也可以看作是密文参数的一个消息认证码. 然而, 我们不能将工具 ABO-RLF 替换为普通的消息认证码. 这是因为  $\pi$  还要用于提取随机值作为封装密钥, 为了实现方案的可证明安全性, 知识证明  $t$  不能泄漏  $\pi$  太多的信息, 这就要求挑战密文中 ABO-RLF 函数是有损的, 而敌手查询的解密密文中 ABO-RLF 又是几乎单射的. 与 Qin-Liu 方案的设计模式相比, 该模式具有的优势是 ABO-RLF 也可以利用哈希证明系统构造, 从而整个密钥封装方案可以基于哈希证明系统实现.

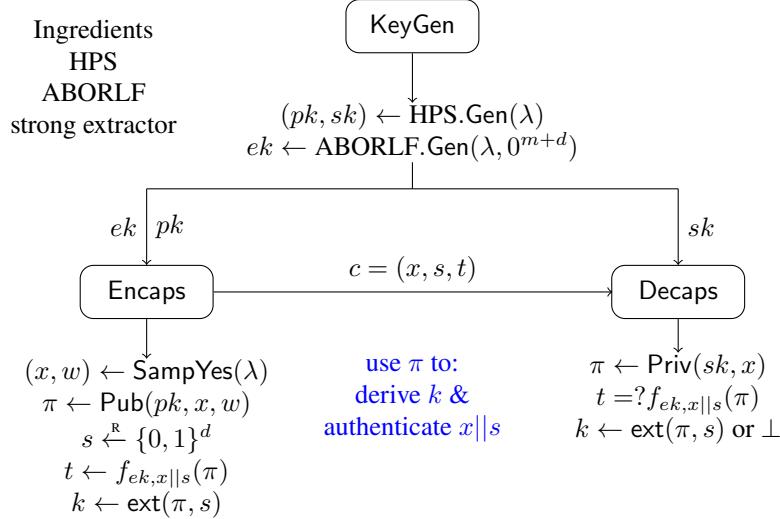


图 5.7: 基于 HPS 和 ABO-RLF 的抗泄漏密钥封装方案设计思路

### 构造 5.7 (基于 HPS 和 ABO-RLF 的抗泄漏密钥封装方案)

- $\text{Setup}(\kappa)$ : 运行  $\text{HPS}.\text{Setup}(\kappa)$  生成 HPS 的公开参数  $pp_1 = (\mathsf{H}, SK, PK, X, L, W, \Pi, \alpha)$ , 其中  $\mathsf{H}$  是  $\epsilon_1$ -universal<sub>1</sub>; 运行  $\text{ABORLF}.\text{Setup}(\kappa)$  生成 ABO-RLF 的公开参数  $pp_2 = (EK, B = X \times \{0, 1\}^d, \Pi, T)$ ; 选择一个平均情况下  $(n - \tau - \ell, \kappa, \epsilon_2)$ -提取器  $\text{ext} : \Pi \times \{0, 1\}^d \rightarrow \{0, 1\}^\kappa$ , 其中  $n = \log 1/\epsilon_1$ ; 输出公开参数  $pp = (pp_1, pp_2)$ .
- $\text{KeyGen}(pp)$ : 首先, 拆分公开参数为  $pp = (pp_1, pp_2)$ ; 然后, 运行  $(pk, sk) \leftarrow \text{HPS}.\text{KeyGen}(pp_1)$  和  $ek \leftarrow \text{ABORLF}.\text{Gen}(pp_2, 0^{m+d})$ , 输出公钥  $\hat{pk} = (pk, ek)$  和私钥  $sk$ .
- $\text{Encaps}(\hat{pk})$ : 以公钥  $\hat{pk} = (pk, ek)$  为输入. 首先, 随机采样  $(x, w) \leftarrow \text{HPS}.\text{SampYes}(pp_1)$ , 计算  $\pi \leftarrow \text{HPS}.\text{Pub}(pk, x, w)$ ; 然后, 选取随机种子  $s \leftarrow \{0, 1\}^d$ , 计算  $t \leftarrow f_{ek,x||s}(\pi)$ ; 输出密文  $c = (x, s, t)$  和封装密钥  $k \leftarrow \text{ext}(\pi, s)$ .
- $\text{Decaps}(sk, c)$ : 以私钥  $sk$  和密文  $c = (x, s, t)$  为输入. 首先, 计算  $\pi \leftarrow \text{HPS}.\text{Priv}(sk, x)$ ; 然后, 判断  $t = f_{ek,x||s}(\pi)$  是否成立. 如果成立, 则输出  $k \leftarrow \text{ext}(\pi, s)$ ; 否则, 输出  $\perp$ .



**定理 5.6**

假设 SMP 问题是困难的, HPS 是一个  $\epsilon_1$ -universal<sub>1</sub> 哈希证明系统, ABORLF 是一个  $(\nu, \tau)$  全除一规则有损函数, ext 是一个平均情况下  $(n - \tau - \ell, \kappa, \epsilon_2)$  强提取器, 则构造方案 ?? 是一个  $\ell$  抗泄漏和选择密文攻击安全的密钥封装方案, 其中  $\ell \leq n - \tau - \kappa - \log \nu - \omega(\log \kappa)$ .



**证明** 在证明分析中, 我们将满足  $x \in L$  的密文  $(x, s, t)$  称之为合法密文 “valid ciphertext”, 而满足  $t = f_{ek, x||s}(\pi)$  的密文称之为合格密文 “well-formed ciphertext”. 显而易见, 一个合法密文不一定是合格的.

下面通过游戏的方式组织证明. 起始游戏定义为 Game 0, 在该游戏中挑战者  $\mathcal{CH}$  执行标准的 LR-CCA 安全性游戏, 即  $k_0^*$  是一个真实密钥而  $k_1^*$  是一个随机密钥, 而在终止游戏中,  $k_0^*$  和  $k_1^*$  都是随机选取的. 令  $S_i$  表示事件 “ $\mathcal{A}$  在游戏 Game  $i$  中成功”.

**Game 0.** 标准的 LR-CCA 安全性游戏.  $\mathcal{CH}$  按以下方式与  $\mathcal{A}$  交互通信:

1. 初始化:  $\mathcal{CH}$  运行  $pp_1 \leftarrow \text{HPS}.\text{Setup}(\kappa)$  和  $pp_2 \leftarrow \text{ABORLF}.\text{Setup}(\lambda)$  分别生成 HPS 和 ABORLF 的公开参数; 运行  $(pk, sk) \leftarrow \text{HPS}.\text{KeyGen}(pp_1)$  和  $ek \leftarrow \text{ABORLF}.\text{Gen}(pp_2, 0^{m+d})$  分别生成 HPS 的公私钥和 ABO-RLF 的求值密钥; 令  $sk$  为密钥封装方案的私钥并将公开参数  $pp = (pp_1, pp_2)$  和公钥  $pk = (pk, ek)$  发送给  $\mathcal{A}$ .
2. 询问阶段 1:  $\mathcal{A}$  可以自适应地进行密钥泄漏查询. 对于任意泄漏询问  $\langle g \rangle$ , 只要泄漏总量小于  $\ell$ , 则  $\mathcal{CH}$  返回  $g(sk)$ .
3. 挑战:  $\mathcal{CH}$  按以下方式处理:
  - (a). 随机选取  $\beta \in \{0, 1\}$ ,  $s^* \xleftarrow{\text{R}} \{0, 1\}^d$ ,  $(x^*, w^*) \leftarrow \text{HPS}.\text{SampYes}(pp_1)$ ;
  - (b). 通过  $\text{HPS}.\text{Pub}(pk, x^*, w^*)$  计算哈希值  $\pi^* \leftarrow \text{H}_{sk}(x^*)$ ;
  - (c). 计算函数值  $t^* \leftarrow f_{ek, x^*||s^*}(\pi^*)$  和封装密钥  $k_0^* \leftarrow \text{ext}(\pi^*, s^*)$ ;
  - (d). 随机选取  $k_1^* \xleftarrow{\text{R}} \{0, 1\}^\kappa$ ;
  - (e). 将挑战密文  $c^* = (x^*, s^*, t^*)$  和  $k_\beta^*$  发送给对手  $\mathcal{A}$ .
4. 询问阶段 2:  $\mathcal{A}$  可以自适应地进行解封装查询. 对于任意解封装询问  $c = (x, s, t)$ , 当  $c \neq c^*$  时,  $\mathcal{CH}$  返回  $\text{KEM}.\text{Decaps}(sk, c)$  给  $\mathcal{A}$ . 具体地, 利用  $\text{HPS}.\text{Priv}(sk, x)$  计算  $\pi \leftarrow \text{H}_{sk}(x)$ . 如果  $t = f_{ek, x||s}(\pi)$ , 输出  $k \leftarrow \text{ext}(\pi, s)$ . 否则, 输出  $\perp$ . 如果询问挑战密文  $c^*$  的解封装, 依据游戏规则, 挑战者直接拒绝回答.
5. 猜测: 最终,  $\mathcal{A}$  输出  $\beta$  的猜测结果  $\beta'$ . 如果  $\beta' = \beta$ , 则  $\mathcal{A}$  成功.

根据定义, 则有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/2|$$

**Game 1.** 与 Game 0 的唯一不同之处在于:  $\mathcal{CH}$  在初始化阶段选择  $(x^*, w^*)$  和  $s^*$ . 该变化仅是概念上的不同. 因此:

$$\Pr[S_1] = \Pr[S_0]$$

**Game 2.** 与 Game 1 的不同之处在于:  $\mathcal{CH}$  将分支参数  $0^{m+d}$  替换为  $x^*||s^*$  以生成 ABO-RLF 的求值密钥  $ek \leftarrow \text{ABORLF}.\text{Gen}(pp_2, \cdot)$ . 根据 ABO-RLF 隐藏有损分支的性质, 则有:

$$|\Pr[S_2] - \Pr[S_1]| \leq \text{negl}(\kappa)$$

**Game 3.** 与 Game 2 的不同之处在于: 在挑战阶段  $\mathcal{CH}$  通过哈希证明系统的私有计算  $\text{HPS}.\text{Priv}(sk, x^*)$  替代公开计算  $\text{HPS}.\text{Pub}(pk, x^*, w^*)$ , 计算哈希值  $\pi^* \leftarrow \text{H}_{sk}(x^*)$ . 依据 HPS 的正确性, 可得:

$$\Pr[S_3] = \Pr[S_2]$$

**Game 4.** 与 Game 3 的不同之处在于:  $\mathcal{CH}$  通过  $\text{HPS}.\text{SampNo}$  替代  $\text{HPS}.\text{SampYes}$  以采样  $x^*$ . 这一变化可以归约到 SMP 问题的困难性上, 即:

$$|\Pr[S_4] - \Pr[S_3]| \leq \text{Adv}_{\mathcal{A}}^{\text{smp}}(\kappa) \leq \text{negl}(\kappa)$$

**Game 5.** 与 Game 4 的不同之处在于: 如果密文  $\langle c = (x, s, t) \rangle$  中  $x \notin L$ , 则  $\mathcal{CH}$  直接决绝回答解封装询问.

令  $E$  表示事件 “在 Game 5 中  $\mathcal{A}$  询问了一个非法但形式合格的解封装密文, 即,  $f_{ek,x||s}(\pi) = t$ , 其中  $\pi = \mathbf{H}_{sk}(x)$  和  $x \notin L \wedge (x, s, t) \neq (x^*, s^*, t^*)$ . 显然, 如果事件  $E$  不发生, 则 Game 4 和 Game 5 是等价的. 根据差分引理, 则有:

$$|\Pr[S_5] - \Pr[S_4]| \leq \Pr[E]$$

**Game 6.** 与 Game 5 不同之处在于:  $\mathcal{CH}$  随机选择  $k_0^* \xleftarrow{\text{R}} \{0, 1\}^\kappa$  替代  $k_0^* \leftarrow \text{ext}(\pi^*, s^*)$ . 显而易见,  $\mathcal{A}$  在 Game 6 中的视图与  $\beta \in \{0, 1\}$  独立无关. 因此,

$$\Pr[S_6] = 1/2$$

下面只需要证明概率  $\Pr[E]$  是可忽略的, 敌手在 Game 5 和 Game 6 之间的视图差别是可忽略的.

### 引理 5.8

概率  $\Pr[E]$  关于安全参数  $\kappa$  是可忽略的.



**证明** 令  $E_i$  表示事件 “ $\mathcal{A}$  的第  $i$  次解封装询问  $c = (x, s, t)$  是非法但合格的密文. 根据  $E$  的定义, 则有  $E = \cup_{1 \leq i \leq Q_d} E_i$ . 接下来, 分析  $\Pr[E_i]$  的上界. 令  $view$  表示敌手在提交第一个解封装查询之前的视图. 显然,  $view = (pk, ek, leak, x^*, s^*, t^*, k_\beta^*)$ .

事件  $E_1$  的概率与哈希证明系统封装的哈希值  $\mathbf{H}_{sk}(x)$  有密切关系. 下面的推导给出了该值的平均极小熵:

$$\tilde{\mathbf{H}}_\infty(\mathbf{H}_{sk}(x)|view, x) = \tilde{\mathbf{H}}_\infty(\mathbf{H}_{sk}(x)|pk, x, leak, t^*, k_\beta^*) \quad (5.12)$$

$$\geq \tilde{\mathbf{H}}_\infty(\mathbf{H}_{sk}(x)|pk, x) - \ell - \tau - \kappa \quad (5.13)$$

$$= n - \ell - \tau - \kappa \quad (5.14)$$

在上述推导过程中, 公式 (??) 依据事实  $\mathbf{H}_{sk}(x)$  由私钥  $sk$  和元素  $x$  完全确定, 而  $sk$  与 AOB-RLF 的求值密钥  $ek$ , 挑战元素  $x^*$  和随机种子  $s^*$  独立无关. 公式 (??) 依据引理 follows from Lemma ?? 和事实泄漏量上界为  $\ell$  比特,  $t^*$  (或  $k_\beta^*$ ) 最多有  $2^\tau$  (或  $2^\kappa$ ) 个可能取值. 公式 (??) 依据 HPS 的  $\epsilon_1$ -universal<sub>1</sub> 性质. 一个合法且合规的密文需满足  $x||s \neq x^*||s^*$ . 由于密文的第三部分元素完全确定了前两部分的元素值, 而分支满足  $x||s \neq x^*||s^*$  的求值密钥  $ek$  确定了一个  $\nu$ -规则函数. 因此, 认证码  $t = f_{ek,x||s}(\mathbf{H}_{sk}(x))$  依然保持了  $\mathbf{H}_{sk}(x)$  的大部分平均极小熵. 结合引理 ?? 和公式 (??), 故有  $\tilde{\mathbf{H}}_\infty(t|view', x) \geq n - \ell - \tau - \kappa - \log \nu$ . 这就证明了  $\Pr[E_1] \leq 2^{\ell+\tau+\kappa+\log \nu}/2^n$ . 由于敌手每次通过不合法解封装查询最多排除  $\nu$  个哈希值  $\mathbf{H}_{sk}(x)$ , 所以  $\Pr[E_i] \leq 2^{\ell+\tau+\kappa+\log \nu}/(2^n - i\nu)$ . 利用联合界性质, 则有:

$$\Pr[E] \leq \sum_{i=1}^{Q_d} \Pr[E_i] \leq \frac{Q_d 2^{\ell+\tau+\kappa+\log \nu}}{2^n - Q_d \nu} \leq \frac{Q_d}{2^{n-\ell-\tau-\kappa-\log \nu} - Q_d}$$

由于  $n - \tau - \ell - \kappa - \log \nu \geq \omega(\log \kappa)$ , 所以该上界关于安全参数  $\kappa$  是可忽略的. This proves the lemma.

### 引理 5.9

敌手在 Game 5 和 Game 6 之间的视图在统计上不可区分.



**证明** 从敌手视角看, 上述两个游戏不可区分的主要原因挑战密文中哈希证明系统封装的哈希值  $\mathbf{H}_{sk}(x^*)$  依然具有较高的条件熵, 从而提取器提取的随机比特串的分布同均匀分布统计距离可忽略. 下面, 我们将 Game 5 中的元素  $k_\beta^*$  记作  $k_{5,\beta}^*$ , 而将 Game 6 中的记作  $k_{6,\beta}^*$ , 将密钥泄漏信息记作  $leak$ . 令  $view' = (pk, ek, leak, x^*, s^*, t^*)$ . 则有:

$$\tilde{\mathbf{H}}_\infty(\mathbf{H}_{sk}(x^*)|view') = \tilde{\mathbf{H}}_\infty(\mathbf{H}_{sk}(x^*)|pk, x^*, leak, t^*) \quad (5.15)$$

$$\geq \tilde{\mathbf{H}}_\infty(\mathbf{H}_{sk}(x^*)|pk, x^*) - \ell - \tau \quad (5.16)$$

$$= n - \ell - \tau \quad (5.17)$$

在上述推导过程中, 公式 (??) 依据事实  $H_{sk}(x^*)$  与  $ek$  和  $s^*$  独立. 公式 (??) 依据事实泄漏量上界为  $\ell, t^*$  最多有  $2^\tau$  个可能取值. 公式 (??) 依据 HPS 的  $\epsilon_1$ -universal<sub>1</sub> 性质.

由于  $k_{5,0}^* \leftarrow \text{ext}(\Lambda_{sk}(x^*), s^*), k_{6,0}^* \xleftarrow{R} K$ ,  $\text{ext}$  是一个平均情况下  $(n-\tau-\ell, \kappa, \epsilon_2)$ -强提取器, 则有  $\Delta[(view', k_{5,0}^*), (view', k_{6,0}^*)] \leq \epsilon_2$ . 根据  $k_{5,\beta}^*$  和  $k_{6,\beta}^*$  的定义, 有  $\Delta[(view', k_{5,\beta}^*), (view', k_{6,\beta}^*)] \leq \epsilon_2/2$ .

值得注意的是, 在 Game 5 和 Game 6 中, 对于非法密文 (即  $x \notin L$ ) 的解封装询问, 挑战者直接返回  $\perp$ , 而对于所有合法密文 ( $x \in L$ ) 的解封装询问, 依据  $H$  的仿射性质, 其结果完全由  $(pk, ek)$  确定. 由此可知, Game 5 中的所有解封装询问结果完全由  $(view', k_{5,\beta}^*)$  的一个函数  $h$  确定, 而 Game 6 中的所有解封装询问完全由同一函数的函数值  $h(view', k_{6,\beta}^*)$  确定. 故手在 Game 5 中的视图记作  $view_5 = (view', k_{5,\beta}^*, h(view', k_{5,\beta}^*))$ , 在 Game 6 中的视图记作  $view_6 = (view', k_{6,\beta}^*, h(view', k_{6,\beta}^*))$ . 则有  $\Delta[view_5, view_6] \leq \epsilon_2/2$ . 引理得证!

综上定理 ?? 得证! □

## 5.1.5 CWZ18 方案

前面构造的抗泄漏密码方案主要利用抗泄漏密码假设或者结合特殊的方案结构, 使得挑战者在模拟密钥泄漏预言机时都可以转化为知道真实私钥的情况. 例如基于哈希证明系统的技术 [NS-CRYPTO-2009], 利用 SMA 问题假设将合法密文转化为非法密文过程中, 挑战者在知道真实密钥的情况下, 依然可以利用 SMA 问题假设将合法子集判断问题元素  $x \in L$  转化为非法元素  $x \in X \setminus L$ . 再结合 Leftover Hash Lemma, 即使在泄漏部分私钥的情况下依然能够提取出伪随机比特串用于掩盖真实消息. 2018 年, Chen 等人 [Chen-ASIACRYPT-2018] 提出一种在计算意义上利用模拟泄漏来构造泄漏预言机的方法. 该方法引入抗泄漏公开可计算伪随机函数 (Leakage-Resilient Publicly Evaluable PseudoRandom Functions, 简称 LR-PEPRFs), 证明了 LR-PEPRFs 直接蕴含了抗泄漏密钥封装方案. 下面, 主要介绍 LR-PEPRFs 的定义、构造及应用.

### 5.1.5.1 LR-PEPRFs 的定义

公开可计算伪随机函数的概念是 Chen 和 Zhang [Chen-SCN-2014] 于 2014 年提出的一种具有特殊性质的伪随机函数, 可以看作是公钥密码学领域与弱伪随机函数 (Weak PRFs) 对应的一种密码学原语. 在 PEPRFs 中, 每个私钥都对应了一个公钥, 以及一个  $\mathcal{NP}$  语言集. 对于语言中的任意元素, 除了利用私钥计算其 PRF 值外, 也可以利用公钥及该元素相关的证据来计算. PEPRFs 可以由特殊假设或更通用假设构造, 例如 (可提取) 哈希证明系统和陷门函数. 接下来, 介绍文献 [Chen-SCN-2014] 中关于 PEPRFs 的标准定义及文献 [Chen-ASIACRYPT-2018] 中关于其抗泄漏性质的安全模型.

#### 定义 5.6 (公开可计算伪随机函数)

令  $L = \{L_{pk}\}_{pk \in PK}$  是一个定义在  $X$  上的  $\mathcal{NP}$  语言集. 一个公开可计算伪随机函数  $F : SK \times X \rightarrow Y \cup \perp$  包含以下三个多项式时间算法:

- $\text{Gen}(\kappa)$ : 以安全参数  $\kappa$  为输入, 输出一个公钥  $pk$  和一个私钥  $sk$ .
- $\text{PrivEval}(sk, x)$ : 以私钥  $sk$  和元素  $x \in X$  为输入, 输出  $y \leftarrow F_{sk}(x) \in Y \cup \perp$ .
- $\text{PubEval}(pk, x, w)$ : 以公钥  $pk$  和元素  $x \in L_{pk}$  及其证据  $w$  为输入, 输出  $y \leftarrow F_{sk}(x) \in Y$ .

在实际应用中, 要求  $L$  可有效随机抽样, 即对于每个公钥  $pk \in PK$ , 存在一个有效抽样算法  $\text{SampRel}$ , 输入公钥  $pk$ , 输出一个随机元素  $x \in L_{pk}$  及一个证据  $w$ .



在 PEPRF 的定义中, PRF 函数的输入  $x$  可能在集合  $L_{pk}$  中, 此时 PRF 值  $F_{sk}(x)$  可能是没有定义的. 此时, 用一个特殊符号  $\perp$  来表示.

抗泄漏伪随机性. 令  $\mathcal{A}$  是一个攻击 PEPRFs 的敌手, 定义如下优势:

$$\text{Adv}_{\mathcal{A}}(\kappa) = \Pr_{\beta' = \beta} \left[ \begin{array}{l} (pk, sk) \leftarrow \text{Gen}(\lambda); \\ \text{state} \leftarrow \mathcal{A}^{\mathcal{O}_{\text{leak}}(\cdot)}(pk); \\ (x^*, w^*) \leftarrow \text{SampRel}(pk); \\ y_0^* \leftarrow F_{sk}(x^*), y_1^* \xleftarrow{\text{R}} Y; \\ \beta \xleftarrow{\text{R}} \{0, 1\}; \\ \beta' \leftarrow \mathcal{A}(pk, x^*, y_\beta^*); \end{array} \right] - \frac{1}{2}.$$

其中,  $\mathcal{O}_{\text{leak}}(\cdot)$  是泄漏预言机, 输入泄漏函数  $f : SK \rightarrow \{0, 1\}^*$ , 返回  $f(sk)$ , 且所有泄漏函数输出的比特长度之和不超过  $\ell$ . 如果对于任意 PPT 敌手, 上述实验中的优势函数是可忽略的, 则该公开可计算伪随机函数 PEPRF 是  $\ell$ -抗泄漏弱伪随机的. Chen 和 Zhang 在文献 [Chen-SCN-2014] 中指出, 由于 PEPRFs 的公开可计算性, 其完全伪随机性是不可能实现的.

### 5.1.5.2 LR-PEPRFs 的构造

抗泄漏 PEPRFs 可以由可穿刺 PEPRFs 构造. 可穿刺 PEPRFs(简称 PPEPRFs)的定义如下:

#### 定义 5.7 (可穿刺公开可计算伪随机函数)

令  $L = \{L_{pk}\}$  是一个定义在  $X$  上的  $\mathcal{NP}$  语言集. 一个可穿刺公开可计算伪随机函数  $F : SK \times X \rightarrow Y \cup \perp$  包含以下多项式时间算法:

- $\text{Gen}(\kappa)$ : 以安全参数  $\kappa$  为输入, 输出一个公钥  $pk$  和一个私钥  $sk$ .
- $\text{PrivEval}(sk, x)$ : 以私钥  $sk$  和元素  $x \in X$  为输入, 输出  $y \leftarrow F_{sk}(x) \in Y \cup \perp$ .
- $\text{Puncture}(sk, x^*)$ : 以私钥  $sk$  和  $x^* \in L_{pk}$  为输入, 输出一个可穿刺密钥  $sk_{x^*}$ .
- $\text{PuncEval}(sk_{x^*}, x)$ : 以一个可穿刺密钥  $sk_{x^*}$  和  $x \neq x^*$  为输入, 输出  $y \leftarrow F_{sk}(x) \in Y \cup \perp$ .
- $\text{PubEval}(pk, x, w)$ : 以公钥  $pk$  和元素  $x \in L_{pk}$  及证据  $w$  为输入, 输出  $y \leftarrow F_{sk}(x) \in Y$ .



PPEPRFs 的安全性要求敌手在知道可穿刺密钥的情况下弱伪随机性依然成立.

弱伪随机性. 令  $\mathcal{A}$  是一个攻击 PPEPRFs 的敌手, 定义下面的优势函数:

$$\text{Adv}_{\mathcal{A}}(\kappa) = \Pr_{\beta' = \beta} \left[ \begin{array}{l} (pk, sk) \leftarrow \text{Gen}(\kappa); \\ (x^*, w^*) \leftarrow \text{SampRel}(pk); \\ sk_{x^*} \leftarrow \text{Puncture}(sk, x^*); \\ y_0^* \leftarrow F_{sk}(x^*), y_1^* \xleftarrow{\text{R}} Y; \\ \beta \xleftarrow{\text{R}} \{0, 1\}; \\ \beta' \leftarrow \mathcal{A}(pk, sk_{x^*}, x^*, y_\beta^*); \end{array} \right] - \frac{1}{2}.$$

如果对于任意 PPT 敌手  $\mathcal{A}$ , 在上述实验中的优势函数都是可忽略的, 则称该 PPEPRF 是弱伪随机的

**基于 PPEPRFs 和  $i\mathcal{O}$  的 LR-PEPRFs.** 令  $F : SK \times X \rightarrow Y \cup \perp$  是一个关于  $L = \{L_{pk}\}_{pk \in PK}$  的可穿刺公开可计算伪随机函数,  $i\mathcal{O}$  是一个不可区分混淆器,  $\text{Ext} : Y \times S \rightarrow Z$  是一个平均情况下  $(n, \epsilon)$ -强提取器. 不失一般性, 可以假设  $Y = \{0, 1\}^\rho$ . 接下来, 介绍一种针对  $\hat{L} = \{\hat{L}_{pk}\}_{pk \in PK}$  的抗泄漏公开可计算伪随机函数  $\hat{F} : \hat{SK} \times \hat{X} \rightarrow Z \cup \perp$ , 其中  $\hat{X} = X \times S$ ,  $\hat{L}_{pk} = \{\hat{x} = (x, s) : x \in L_{pk} \wedge s \in S\}$ . 根据  $\hat{L}$  的定义,  $x \in L_{pk}$  的证据  $w$  也是  $\hat{x} = (x, s) \in \hat{L}_{pk}$  的证据, 其中  $s$  是  $S$  中任意种子.

- $\text{Gen}(\kappa)$ : 运行  $F.\text{Gen}(\kappa)$  以生成  $(pk, sk)$ , 构造  $\hat{sk} \leftarrow i\mathcal{O}(\text{PrivEval})$ , 其中程序  $\text{PrivEval}$  的定义见图 ??; output  $(pk, \hat{sk})$ .
- $\text{PrivEval}(\hat{sk}, \hat{x})$ : 输入  $\hat{sk}$  和  $\hat{x} = (x, s) \in \hat{X}$ , 输出  $\hat{y} \leftarrow \hat{sk}(\hat{x})$ . 这里实际上定义了  $\hat{F}_{\hat{sk}}(\hat{x}) := \text{ext}(F_{sk}(x), s)$ , 其中  $\hat{x} = (x, s)$ .

- $\text{PubEval}(pk, \hat{x}, w)$ : 输出  $pk, \hat{x} = (x, s) \in \hat{L}_{pk}$  和  $\hat{x}$  的证据  $w$ , 计算利用  $F.\text{PubEval}(pk, x, w)$  计算  $y \leftarrow F_{sk}(x)$ ,  
输出  $z \leftarrow \text{ext}(y, s)$ .

<b>PrivEval</b> <b>Constants:</b> PPEPRF secret key $sk$ <b>Input:</b> $\hat{x} = (x, s)$ 1. Output $\text{ext}(F_{sk}(x), s)$ .
---

图 5.8: 程序 PrivEval 的描述.

<b>PrivEval*</b> <b>Constants:</b> PPEPRF punctured secret key $sk_{x^*}$ , $x^*$ and $y^*$ <b>Input:</b> $\hat{x} = (x, s)$ 1. If $x = x^*$ , output $\text{ext}(y^*, s)$ . 2. Else, output $\text{ext}(F_{sk_{x^*}}(x), s)$ .
---

图 5.9: 程序 PrivEval\* 的描述

**定理 5.7**

如果  $F$  是一个安全的可穿刺公开可计算伪随机函数,  $i\mathcal{O}$  是一个不可区分混淆器,  $\text{Ext}$  是一个平均情况下  $(n, \epsilon)$ -强提取器, 则上面构造的公开可计算伪随机函数是  $\ell$ -抗泄漏弱伪随机的, 其中  $\ell \leq \rho - n$ .



**证明** 通过游戏的方式组织证明. 令  $S_i$  表示事件 “ $\mathcal{A}$  在游戏  $i$  中成功”.

**Game 0.** 这是标准的 PEPRFs 抗泄漏弱伪随机游戏. 挑战者  $\mathcal{CH}$  按以下方式同对手  $\mathcal{A}$  交互执行游戏.

1. 初始化:  $\mathcal{CH}$  运行  $(pk, sk) \leftarrow F.\text{Gen}(\kappa)$ , 创建  $\hat{sk} \leftarrow i\mathcal{O}(\text{PrivEval})$ , 然后将  $pk$  发送给  $\mathcal{A}$ .
2. 泄漏询问: 当收到泄漏询问函数  $\langle f \rangle$  时, 只要泄漏总量不超过  $\ell$ ,  $\mathcal{CH}$  将  $f(\hat{sk})$  发回给  $\mathcal{A}$ .
3. 挑战:  $\mathcal{CH}$  随机采样  $(x^*, w^*) \leftarrow \text{SampRel}(pk)$ , 选择  $s^* \xleftarrow{R} S$ , 再利用  $F.\text{PubEval}(pk, x^*, w^*)$  计算  $y^* \leftarrow F_{sk}(x^*)$  和  $z_0^* \leftarrow \text{ext}(y^*, s^*)$ . 接下来, 随机选择  $z_1^* \xleftarrow{R} Z$  和  $\beta \xleftarrow{R} \{0, 1\}$ . 最后,  $\mathcal{CH}$  将  $\hat{x}^* = (x^*, s^*)$  和  $z_\beta^*$  发送给  $\mathcal{A}$ .
4. 猜测:  $\mathcal{A}$  输出一比特  $\beta'$  作为对  $\beta$  的猜测结果. 如果  $\beta' = \beta$ , 则  $\mathcal{A}$  成功.

根据上述定义, 则有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/2|$$

**Game 1.** 该游戏与 Game 0 的不同之处在于:  $\mathcal{CH}$  在初始化阶段就随机采样  $x^*$  和  $w^*$  并计算  $y^* \leftarrow F_{sk}(x^*)$ . 该变化仅是概念上的不同, 因此:

$$\Pr[S_1] = \Pr[S_0]$$

**Game 2.** 该游戏与 Game 1 的不同之处在于:  $\mathcal{CH}$  在初始化阶段同时计算  $sk_{x^*} \leftarrow F.\text{Puncture}(sk, x^*)$ , 创建  $\hat{sk} \leftarrow i\mathcal{O}(\text{PuncPriv})$ . 此处的程序  $\text{PrivEval}^*$  通过常量  $sk_{x^*}, x^*$  和  $y^*$  构建, 见图 ?? 中的定义. 显而易见, 对于所有输入, 程序  $\text{PrivEval}$  和  $\text{PrivEval}^*$  输出结果是一致的. 因此, 可以将这两个游戏之间的区别直接归约到  $i\mathcal{O}$  安全性上, 即:

$$|\Pr[S_2] - \Pr[S_1]| \leq \text{Adv}_{\mathcal{A}}^{i\mathcal{O}} \leq \text{negl}(\kappa)$$

**Game 3.** 该游戏与 Game 2 的不同之处在于:  $\mathcal{CH}$  在初始化阶段随机选择  $y^* \xleftarrow{R} Y$ , 而不再通过  $y^* \leftarrow F_{sk}(x^*)$  计算

生成. 根据 PPEPRF 的弱伪随机性, 这一变化对于任意 PPT 敌手是不可区分的, 即:

$$|\Pr[S_3] - \Pr[S_2]| \leq \text{Adv}_{\mathcal{A}}^{\text{PPEPRF}} \leq \text{negl}(\kappa)$$

**Game 4.** 该游戏与 Game 3 的不同之处在于:  $\mathcal{CH}$  在挑战阶段随机选择  $z_0^* \xleftarrow{R} Z$ , 而不再通过  $z_0^* \leftarrow \text{Ext}(y^*, s^*)$  计算生成.

令  $V$  表示由公钥  $pk, x^*$  和  $s^*$  组成的集合. 在 Game 3 和 Game 4 中,  $y^*$  是从  $Y$  上均匀选取且独立于  $V$ , 因此  $H_\infty(y^*|V) = \rho$ . 注意到  $\mathcal{A}$  最多可以获取  $\hat{k}$  的  $\ell$  比特泄漏量, 记作  $leak$ . 该泄漏量与  $y^*$  有关. 根据引理 ?? 有  $\tilde{H}_\infty(y^*|(V, leak)) \geq H_\infty(y^*|V) - \ell = \rho - \ell$ , 且大于  $n$ . 由于  $\text{Ext}$  是一个平均情况下  $(n, \epsilon)$ -强提取器, 由此可得即时在  $V$  和泄漏信息已知的条件下,  $\text{Ext}(y^*, s^*)$  与  $z_0^* \in Z$  的统计距离不超过  $\epsilon$ . 注意到  $\mathcal{A}$  在 Game 3 和 Game 4 中的视图完全由  $z_0^*, z_1^*, \beta^*, V$  和  $leak$  确定, 而  $z_1^*, \beta^*, V$  和  $leak$  的分布在两个游戏中是一样的. 所以  $\mathcal{A}$  在这两个游戏中的视图差异不超过  $\epsilon/2$ . 因此:

$$|\Pr[S_4] - \Pr[S_3]| \leq \epsilon/2 \leq \text{negl}(\kappa)$$

在 Game 4 中,  $z_0^*$  和  $z_1^*$  都是从  $Z$  中随机选取的. 所以:

$$\Pr[S_4] = 1/2$$

综上, 定理得证!  $\square$

**提高泄露率的构造.** 上面构造 LR-PEPRFs 的泄漏率较低. 下面介绍一种提高泄漏率至最优的方法. 该方法需要下面几个密码学原语: (1) 一个 IND-CPA 安全的对称加密方案 SKE, 其消息空间为  $\{0, 1\}^\rho$ 、密文空间为  $\{0, 1\}^v$ ; (2) 一个  $(v, \tau)$ -有损函数. 具体构造如下:

- **Gen( $\kappa$ ):** 运行  $(pk, sk) \leftarrow F.\text{Gen}(\lambda), h \leftarrow LF.\text{GenInj}(\lambda), k_e \leftarrow SKE.\text{Gen}(\lambda)$ , 创建一个冗余密文  $ct \leftarrow SKE.\text{Enc}(k_e, 0^\rho)$  作为  $\hat{sk}$ , 计算  $\eta^* \leftarrow h(ct)$ , 创建  $C_{\text{eval}} \leftarrow iO(\text{PrivEval})$  (程序 PrivEval 的定义见图 ??,  $\eta^*$  看作是该程序的触发器), 设置  $\hat{pk} = (pk, C_{\text{eval}})$ , 输出  $(\hat{pk}, \hat{sk})$ .
- **PrivEval( $\hat{sk}, \hat{x}$ ):** 输入  $\hat{sk}$  和  $\hat{x} = (x, s) \in \hat{X}$ , 输出  $\hat{y} \leftarrow C_{\text{eval}}(\hat{sk}, \hat{x})$ . 这相当于定了  $\hat{F}_{\hat{sk}}(\hat{x}) := \text{Ext}(F_{sk}(x), s)$ , 其中  $\hat{x} = (x, s)$ .
- **PubEval( $\hat{pk}, \hat{x}, w$ ):** 输入  $\hat{pk} = (pk, C_{\text{eval}}, t), \hat{x} = (x, s) \in \hat{L}_{pk}$  和  $\hat{x}$  的证据  $w$ , 利用  $F.\text{PubEval}(pk, x, w)$  计算  $y \leftarrow F_{sk}(x)$ . 输出  $\hat{y} \leftarrow \text{ext}(y, s)$ .

### PrivEval

**Constants:** PPEPRF secret key  $sk, \eta^*$

**Input:**  $\hat{sk}, \hat{x} = (x, s)$

1. If  $h(\hat{sk}) \neq \eta^*$ , output  $\perp$ .
2. Else, output  $\text{Ext}(F_{sk}(x), s)$ .

图 5.10: 程序 PrivEval 的描述

### 定理 5.8

如果  $F$  是一个安全可穿刺公开可计算伪随机函数,  $iO$  是不可区分混淆器, SKE 是 IND-CPA 安全的对称加密方案, LF 是一个  $(v, \tau)$ -有损函数族, Ext 是平均情况下  $(n, \epsilon)$ -强提取器, 则上述构造的 PEPRF 方案是  $\ell$ -抗泄漏弱伪随机的, 其中  $\ell \leq \rho - n - \tau$ .

**证明** 通过设置合适的参数, 如设置  $v = \rho + o(\rho), n = o(\rho), \tau = o(v)$ , 则有  $|\hat{sk}| = v = \rho + o(\rho), \ell = \rho - o(\rho)$ , 并且泄漏率达到最优.

下面通过游戏的方式组织证明. 令  $S_i$  表示事件 “ $\mathcal{A}$  在游戏  $i$  中成功”.

PrivEval*
<b>Constants:</b> PPEPRF punctured secret key $sk_{x^*}$ , $k_e$ , $x^*$ and $\eta^*$
<b>Input:</b> $\hat{sk}, \hat{x} = (x, s)$
1. If $h(\hat{sk}) \neq \eta^*$ , output $\perp$ .
2. If $x = x^*$ , set $y^* \leftarrow \text{SKE.Dec}(k_e, \hat{sk})$ , output $\text{ext}(y^*, s)$ .
3. Else, output $\text{ext}(F_{sk_{x^*}}(x), s)$ .

图 5.11: 程序 PuncEval 的描述

**Game 0.** 该游戏是标准的 PEPRFs 抗泄漏弱伪随机性游戏.  $\mathcal{CH}$  与  $\mathcal{A}$  按以下方式交互完成游戏.

1. 初始化:  $\mathcal{CH}$  运行  $(pk, sk) \leftarrow F.\text{Gen}(\lambda)$ ,  $h \leftarrow \text{LF.GenInj}(\lambda)$ , 随机选取  $k_e \leftarrow \text{SKE.Gen}(\lambda)$ , 生成一个冗余密文  $ct \leftarrow \text{SKE.Enc}(k_e, 0^\rho)$  作为  $\hat{sk}$ , 计算  $\eta^* \leftarrow h(ct)$ , creates  $C_{\text{eval}} \leftarrow i\mathcal{O}(\text{PrivEval})$ .  $\mathcal{CH}$  设置  $\hat{pk} = (pk, C_{\text{eval}})$  并发送给对手  $\mathcal{A}$ .
2. 泄漏询问: 当收到泄漏询问函数  $\langle f \rangle$  时, 只要泄漏总量不超过  $\ell$ , 则  $\mathcal{CH}$  将  $f(\hat{sk})$  发送给  $\mathcal{A}$ .
3. 挑战:  $\mathcal{CH}$  随机采样  $(x^*, w^*) \leftarrow \text{SampRel}(pk)$ , 选择  $s^* \xleftarrow{\text{R}} S$ , 利用  $F.\text{PubEval}(pk, x^*, w^*)$  计算  $y^* \leftarrow F_{sk}(x^*)$ ,  $z_0^* \leftarrow \text{ext}(y^*, s^*)$ , 随机选择  $z_1^* \xleftarrow{\text{R}} Z$ ,  $\beta \xleftarrow{\text{R}} \{0, 1\}$ . 最后,  $\mathcal{CH}$  将  $\hat{x}^* = (x^*, s^*)$  和  $z_\beta^*$  发送给  $\mathcal{A}$ .
4. 猜测:  $\mathcal{A}$  输出一比特  $\beta'$  作为对  $\beta$  的猜测结果. 如果  $\beta' = \beta$ , 则  $\mathcal{A}$  成功.

根据定义, 则有:

$$\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[S_0] - 1/2|$$

**Game 1.** 该游戏与 Game 0 的不同之处在于:  $\mathcal{CH}$  在初始化阶段就随机选择  $x^*, w^*$  并计算  $y^* \leftarrow F_{sk}(x^*)$ . 该变化仅是概念上的不同, 因此:

$$\Pr[S_1] = \Pr[S_0]$$

**Game 2.** 该游戏与 Game 1 的不同之处在于:  $\mathcal{CH}$  在初始化阶段计算  $ct \leftarrow \text{SKE.Enc}(k_e, y^*)$  而不再是  $ct \leftarrow \text{SKE.Enc}(k_e, 0^\rho)$ . 这一区别可以直接归约到 SKE 的 IND-CPA 安全性上, 所以:

$$|\Pr[S_2] - \Pr[S_1]| \leq \text{Adv}_{\mathcal{A}}^{\text{SKE}}$$

**Game 3.** 该游戏与 Game 2 的不同之处在于:  $\mathcal{CH}$  在初始化阶段同时计算  $sk_{x^*} \leftarrow F.\text{Puncture}(sk, x^*)$  并创建  $C_{\text{eval}} \leftarrow i\mathcal{O}(\text{PrivEval})$  程序  $\text{PrivEval}^*$  基于常量  $(sk_{x^*}, x^*, y^*)$  的定义见图 ??.

根据  $h$  的单射性和 SKE、PPEPRF 的正确性, 对于任意输入, 两个程序  $\text{PrivEval}$  和  $\text{PuncPriv}$  的输出结果是一致的. 因此, 这两个游戏之间的区别可以直接归约到  $i\mathcal{O}$  安全性上, 即:

$$|\Pr[S_3] - \Pr[S_2]| \leq \text{Adv}_{\mathcal{A}}^{i\mathcal{O}}$$

**Game 4.** 该游戏与 3 的不同之处在于: 在初始化阶段,  $\mathcal{CH}$  随机选择  $y^* \xleftarrow{\text{R}} Y$  而不再是通过  $y^* \leftarrow F_{sk}(x^*)$  计算所得.

基于 PPEPRF 的伪随机性假设, 这两个游戏之间的变化对于任意 PPT 敌手是不可区分的, 即:

$$|\Pr[S_4] - \Pr[S_3]| \leq \text{Adv}_{\mathcal{A}}^{\text{PPEPRF}}$$

**Game 5.** 该游戏与 Game 4 的不同之处在于:  $\mathcal{CH}$  利用  $\text{LF.GenLossy}(\lambda)$  选择函数  $h$ , 而不在选择一个单射函数. 这两个游戏之间的区别可以直接归约到有损函数的安全性上, 即:

$$|\Pr[S_5] - \Pr[S_4]| \leq \text{Adv}_{\mathcal{A}}^{\text{LF}}$$

**Game 6.** 该游戏与 Game 5 的区别在于:  $\mathcal{CH}$  在挑战阶段随机选择  $z_0^* \xleftarrow{R} Z$ , 而不再设置  $z_0^* \leftarrow \text{ext}(y^*, s^*)$ .

令  $V$  表示公钥  $\hat{pk} = (pk, C_{\text{eval}})$ ,  $x^*$  和  $s^*$  组成的集合. 在 Game 5 和 Game 6 中,  $y^*$  是从  $Y$  中均匀选取且与  $sk_{x^*}$ ,  $x^*$  和  $s^*$  完全独立, 单射与  $\eta^*$  相关且最多有  $2^\tau$  种取值. 根据引理 ??, 则有  $H_\infty(y^*|V) \geq \rho - \tau$ . 注意到  $\mathcal{A}$  最多可以获取  $\hat{sk}$  的  $\ell$  比特泄漏量, 记作  $leak$ . 该泄漏量与  $y^*$  有关. 根据引理 ?? 可得  $\tilde{H}_\infty(y^*|(V, leak)) \geq H_\infty(y^*|V) - \ell = \rho - \tau - \ell$ , 且该值大于  $n$ . 由于  $\text{Ext}$  是平均情况下  $(n, \epsilon)$ -强提取器, 由此可得即时在  $V$  和泄漏信息已知的条件下,  $\text{Ext}(y^*, s^*)$  与  $z_0^* \in Z$  的统计距离不超过  $\epsilon$ . 注意到  $\mathcal{A}$  在 Game 5 和 Game 6 中的视图完全由  $z_0^*$ ,  $z_1^*$ ,  $\beta^*$ ,  $V$  和  $leak$  确定, 而  $z_1^*$ ,  $z_1^*$ ,  $\beta^*$ ,  $V$  和  $leak$  的分布在两个游戏中是一样的. 所以  $\mathcal{A}$  在这两个游戏中的视图差异不超过  $\epsilon/2$ . 因此:

$$|\Pr[S_6] - \Pr[S_5]| \leq \epsilon/2 \leq \text{negl}(\kappa)$$

在 Game 6 中,  $z_0^*$  和  $z_1^*$  都是从  $Z$  中随机选取的. 所以:

$$\Pr[S_6] = 1/2$$

综上, 定理得证! □

### 5.1.5.3 LR-PEPRFs 的应用

Chen 和 Zhang 在文献 [Chen-SCN-2014] 中指出弱伪随机的公开可计算伪随机函数直接蕴含了 IND-CPA 安全的密钥封装方案. 2018 年, Chen 等人在文献 [Chen-ASIACRYPT-2018] 中指出这一结论在密钥泄漏环境下依然成立. 由此, 利用抗泄漏的公开可计算为随机函数, 按照下面的方式可以构造一个抗泄漏密钥封装方案.

假设  $F : SK \times X \rightarrow Y$  是一个 LR-PEPRF, 其中  $L = \{L_{pk}\}_{pk \in PK}$ ,  $Y$  是一个加法群. KEM 的密钥同 PEPRF 的密钥. 当加密消息  $m \in Y$  时, 随机选择  $x \xleftarrow{R} L_{pk}$  及其证据  $w$ , 计算  $k \leftarrow \text{PubEval}(pk, x, w)$ , 输出密文  $(x, k + m)$ . 解密过程是利用  $\text{PrivEval}(k, x)$  重新计算  $k$ . KEM 方案的 LR-CPA 安全性依赖于 PEPRF 的抗泄漏弱伪随机性. 结合数据封装机制 [CS-EUROCRYPT-2002], 上述 KEM 方案也可以转化为 LR-CPA 安全的公钥加密方案.

#### 构造 5.8 (基于 PEPRFs 的 KEM 方案)

假设  $F : SK \times X \rightarrow Y$  是一个抗泄漏弱随机安全的伪随机函数, 其中  $L = \{L_{pk}\}_{pk \in PK}$ ,  $Y$  是一个加法群. 构造的密钥封装方案如下:

- $\text{Setup}(1^\kappa)$ : 运行  $\text{HPS}.\text{Setup}(1^\kappa)$ , 输出 HPS 的一个实例参数  $\text{params} = (\mathsf{H}, SK, PK, X, L, W, \Pi, \alpha)$ . 选择一个平均情况下  $(\log \Pi - \lambda, \epsilon_2)$ -强提取器  $\text{Ext} : \Pi \times \{0, 1\}^t \rightarrow \{0, 1\}^m$ . 将  $pp = (\text{params}, \text{Ext})$  作为公开参数, 其中  $\{0, 1\}^m$  作为明文空间和  $X \times \{0, 1\}^t \times \{0, 1\}^m$  作为密文空间.
- $\text{KeyGen}(pp)$ : 运行  $(pk, sk) \leftarrow \text{HPS}.\text{KeyGen}(pp)$ , 输出公钥  $pk$  和私钥  $sk$ .
- $\text{Encrypt}(pk, M; r)$ : 以公钥  $pk$ , 明文  $M \in \{0, 1\}^m$  和随机数  $r$  为输入, 执行如下步骤:
  1. 运行  $(x, w) \leftarrow \text{SampR}(r)$  生成随机实例和相应的证据;
  2. 通过  $\text{HPS}.\text{PubEval}(pk, x, w)$  计算实例  $x$  的哈希证明  $\pi \leftarrow \mathsf{H}_{sk}(x)$ ;
  3. 随机选择  $s \xleftarrow{R} \{0, 1\}^t$ , 计算  $\psi = \text{Ext}(\pi, s) \oplus M$ ;
  4. 输出  $(x, s, \psi)$  作为密文  $c$ .
- $\text{Decrypt}(sk, c)$ : 以私钥  $sk$  和密文  $c = (x, s, \psi)$  为输入, 通过  $\text{HPS}.\text{PrivEval}(sk, x)$  计算  $x$  的哈希证明  $\pi \leftarrow \mathsf{H}_{sk}(x)$ , 再恢复明文  $M' := \psi \oplus \text{Ext}(\pi, s)$ .



## 5.2 抗篡改安全

相关密钥攻击 (Related-Key Attacks, RKAs) 最早由 Biham [**Biham-JOC-1994**] 和 Knudsen [**Knudsen-AUSCRYPT-1992**] 提出的。2003 年, Bellare 和 Kohno [**BK-EUROCRYPT-2003**] 给出它的形式化定义。早期设计的密码算法仅能抵抗简单的线性密钥篡改攻击, 例如 Bellare 和 Cash [**BC-CRYPTO-2010**] 提出的基于 DDH 假设的抗相关密钥攻击的伪随机函数 (RKA-PRFs)。2011 年, Bellare 等人 [**BCM-ASIACRYPT-2011**] 给出如何从 RKA-PRFs 和其它非 RKA 安全的密码算法来实现 RKA 安全的密码算法, 包括公钥加密、对称加密、签名和基于身份加密。同年, Ap-plebaum 等人 [**AHI-ITCS-2011**] 提出基于 LPN 和 LWE 假设的抗线性密钥篡改语义安全对称加密方案。2012 年, Wee [**Wee-EUROCRYPT-2012**] 提出利用特殊性质的自适应单向陷门函数构造抗线性篡改的 RKA-CCA 公钥加密方案, 并给出基于因子分解、DBDH 和 LWE 等困难问题的具体实现。

### 5.2.1 安全模型

根据文献 [**BCM-ASIACRYPT-2011**], 一个密码系统通常由系统参数、算法 (程序实现的代码) 和密钥 (公钥/私钥) 三部分组成。公钥/私钥是最有可能受到 RKA 攻击的, 而系统参数和算法假定是不受攻击的。这是因为, 系统参数并不包含用户的密钥信息, 与用户是独立的。它可以在用户密钥选取之前确定并且可以嵌入到算法的实现代码中。令  $\Phi = \{\phi : \mathcal{SK} \rightarrow \mathcal{SK}\}$  是一个从密钥空间  $\mathcal{SK}$  到自身的变换函数族。一个公钥加密方案 PKE 的 RKA-CCA 安全模型的定义如下:

**RKA-CCA 安全性.** 定义公钥加密方案 PKE 的 RKA-CCA 敌手  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  的优势函数如下:

$$\text{Adv}_{\mathcal{A}, \Phi, \text{PKE}}^{\text{RKA-CCA}}(\kappa) = \Pr_{\beta' = \beta} \left[ \begin{array}{l} pp \leftarrow \text{Setup}(1^\kappa); \\ (pk, sk) \leftarrow \text{KeyGen}(pp); \\ (m_0, m_1, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{rka}}(\cdot)}(pp, pk), s.t. |m_0| = |m_1|; \\ \beta \xleftarrow{R} \{0, 1\}; \\ c^* \leftarrow \text{Encrypt}(pk, m_\beta); \\ \beta' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{rka}}(\cdot)}(pp, pk, state, c^*); \end{array} \right] - \frac{1}{2},$$

上述定义中, 敌手  $\mathcal{A}$  在接收到挑战密文前后两阶段都可以访问密钥篡改谕言机同时获得在篡改密钥下的解密结果。具体地, 谕言机  $\mathcal{O}_{\text{rka}}(\cdot)$  的输入为一对篡改函数和密文  $(\phi, c)$ , 其中  $\phi \in \Phi$ , 输出为  $\text{Decrypt}(\phi(sk), c)$ 。在第 2 阶段询问中, 敌手不能进行满足条件  $(\phi(sk), c) = (sk, c^*)$  的询问, 否则敌手直接获取了挑战密文的解密结果, 使安全模型失去了实际意义。如果对于任意的 PPT 敌手  $\mathcal{A}$ , 优势函数  $\text{Adv}_{\mathcal{A}, \Phi, \text{PKE}}^{\text{RKA-CCA}}(\kappa)$  是可忽略的, 则称公钥加密方案 PKE 是  $\Phi$ -RKA-CCA 安全的。

 **笔记** 在 RKA 攻击中,  $\Phi$  称为密钥篡改函数族。如果对于所有密钥  $sk \in \mathcal{SK}$  及所有不同的篡改函数  $\phi, \phi' \in \Phi$  都有  $\phi(sk) \neq \phi'(sk)$ , 则密钥篡改函数族  $\Phi$  称为“claw-free”的。在已有的 RKA 安全加密或其他密码方案中, 大部分方案仅能抵抗这类篡改函数。“Claw-free”篡改函数是一种特殊的函数, 在实际中, 绝大部分篡改攻击函数都是非“claw-free”的。从前面的定义可以看出 RKA-CCA 与 IND-CCA 安全性之间有着密切的联系。在两种模型中, 敌手都可以访问解密服务。不同之处在于 RKA 敌手还可以访问篡改密钥下的解密服务。此外, 只要  $\phi(sk) \neq sk$ , 敌手是可以访问挑战密文的解密服务的。这也是 RKA-CCA 安全性比 IND-CCA 安全性更难实现的原因之一。如果密钥篡改函数仅包含单位函数  $1_\phi$ , 则  $\{1_\phi\}$ -RKA-CCA 等价于 IND-CCA。

### 5.2.2 Wee12 方案

#### 5.2.2.1 自适应单向陷门关系的特殊性质

自适应单向陷门关系 (ATDR) 在构造 IND-CCA 安全公钥加密方面具有强大的优势。而 RKA-CCA 安全的 PKE 本身也是 IND-CCA 安全的, 那么一个自然的问题是自适应单向陷门关系能否用于构造 RKA-CCA 安全的公

钥加密方案, 需要满足哪些特殊的性质, 篡改函数集的形式如何呢. 2012 年, Wee [Wee-EUROCRYPT-2012] 给出了这些问题的答案, 提出一种基于自适应单向陷门关系的 RKA-CCA 安全公钥加密方案的通用构造. 带标签自适应单向陷门关系在随机采样和求逆算法中会额外输入一个标签, 而该标签与自适应单向陷门关系的陷门无关. 一个带标签自适应单向陷门关系  $\text{ATDR} = (\text{Setup}, \text{KeyGen}, \text{Sample}, \text{TdInv})$  需要满足以下两个额外的性质:

- 密钥同态性 ( $\Phi$ -Key Homomorphism): 对于任意  $\phi \in \Phi$  和任意的陷门  $td$ , 标签  $tag$ , 关系值  $y$ , 存在一个 PPT 算法  $T$ , 使得

$$\text{TdInv}(\phi(td), tag, y) = \text{TdInv}(td, tag, T(pp, \phi, tag, y)).$$

- 指纹识别性 ( $\Phi$ -Fingerprinting): 类似于指纹认证, 对于一个固定的关系值 (指纹)  $y^*$ , 对陷门的任何篡改, 通过求逆算法都可以被检测出来. 具体地, 定义敌手  $\mathcal{A}$  的优势函数如下:

$$\text{Adv}_{\mathcal{A}, \Phi, \text{ATDR}}^{\text{FP}}(\kappa) = \Pr \left[ \begin{array}{lcl} & & pp \leftarrow \text{Setup}(1^\lambda); \\ \text{TdInv}(\phi(td), tag^*, y^*) \neq \perp & : & tag^* \leftarrow \mathcal{A}(pp); \\ \wedge \phi \in \Phi \wedge \phi(td) \neq td & & (ek, td) \leftarrow \text{KeyGen}(pp); \\ & & (s^*, y^*) \xleftarrow{\text{R}} \text{Sample}(td, tag^*); \\ & & \phi \leftarrow \mathcal{A}(pp, ek, td, y^*); \end{array} \right]$$

对于任意 PPT 敌手  $\mathcal{A}$ , 如果优势函数  $\text{Adv}_{\mathcal{A}, \Phi, \text{ATDR}}^{\text{FP}}(\kappa)$  都是可忽略的, 则称 ATDR 是  $\Phi$ -Fingerprinting 的.

 **笔记** 上面这两个额外的性质为 RKA-CCA 安全性的证明提供了一种简洁的解决办法. 首先, 密钥同态性实际上提供了一种通过原始求逆预言机  $\text{TdInv}(td, tag, \cdot)$  来回答在篡改密钥  $\phi(td)$  下的求逆询问. 指纹识别性实际上保证了敌手不能查询挑战关系值在原始陷门下的求逆询问. 当用 ATDR 构造 IND-CCA 安全的公钥加密方案时, 这两种额外的性质可以直接用于 RKA-CCA 安全性中, 从而使得构造 IND-CCA 安全的公钥加密方案也是 RKA-CCA 安全的.

 **笔记** 在  $\Phi$ -Fingerprinting 性质中, 敌手是知道陷门  $td$  的. 这一条件在后面的证明中至关重要, 等价于挑战者知自适应单向陷门关系的陷门, 从而可以模拟解密预言机.

### 5.2.2.2 基于自适应单向陷门关系的 RKA-CCA 方案

下面介绍 Wee 的通用构造方案. 该方案还需要一个强不可伪造一次签名方案 (Strong One-Time Signatures) OTS = ( $\text{Setup}, \text{KeyGen}, \text{Sign}, \text{Verify}$ ). 其中,  $\text{Setup}(1^\kappa)$  输出签名方案的系统参数  $pp$ ;  $\text{KeyGen}(pp)$  输出签名公钥  $vk$  和私钥  $sigk$ ;  $\text{Sign}(sigk, M)$  输出消息  $M$  的签名  $\sigma$ ;  $\text{Verify}(vk, M, \sigma)$  输出 1 当且仅当  $\sigma$  是  $M$  的一个合法签名. 一次签名方案的敌手  $\mathcal{A}$  的优势函数定义如下:

$$\text{Adv}_{\mathcal{A}, \text{OTS}}(\kappa) = \Pr \left[ \begin{array}{lcl} & & pp \leftarrow \text{Setup}(1^\lambda); \\ \text{Verify}(vk, M', \sigma') = 1 & : & (vk, sigk) \leftarrow \text{KeyGen}(pp); \\ \wedge (M', \sigma') \neq (M, \sigma) & & M \leftarrow \mathcal{A}(vk); \\ & & \sigma \xleftarrow{\text{R}} \text{Sign}(sigk, M); \\ & & (M', \sigma') \leftarrow \mathcal{A}(\sigma); \end{array} \right]$$

对于任意 PPT 敌手, 如果优势函数  $\text{Adv}_{\mathcal{A}, \text{OTS}}(\kappa)$  是可忽略的, 则称一次签名方案是强不可伪造的.

#### 构造 5.9 (Wee RKA-CCA PKE)

- $\text{Setup}(1^\kappa)$ : 运行  $\text{ATDR}.\text{Setup}(1^\kappa)$  和  $\text{OTS}.\text{Setup}(1^\kappa)$ , 分别输出 ATDR 的系统参数  $pp_1$  和  $pp_2$ . 选择一个伪随机函数  $G : X \rightarrow \{0, 1\}^m$ , 其中  $X$  为 ATDR 的原像空间. 输出公钥加密方案的公开参数  $pp = (pp_1, pp_2, G)$ . 其中  $\{0, 1\}^m$  作为明文空间.
- $\text{KeyGen}(pp)$ : 运行  $(ek, td) \leftarrow \text{ATDR}.\text{KeyGen}(pp)$ , 输出公钥  $pk := ek$  和私钥  $sk := td$ .
- $\text{Encrypt}(pk, M)$ : 以公钥  $pk := ek$  和明文  $M \in \{0, 1\}^m$  为输入, 执行如下步骤:

1. 运行  $(vk, sigk) \leftarrow \text{OTS.KeyGen}(pp_2)$  生成一次签名的公钥和私钥;
  2. 运行  $(x, y) \leftarrow \text{ATDR.Sample}(ek, vk)$  生成一个随机采样  $(x, y)$ ;
  3. 计算  $\psi = G(x) \oplus M$ ;
  4. 运行  $\sigma \leftarrow \text{OTS.Sign}(sigk, y || \psi)$ ;
  5. 输出密文  $C = (vk, \sigma, y, \psi)$ .
- $\text{Decrypt}(sk, C)$ : 以私钥  $sk := td$  和密文  $C = (vk, \sigma, y, \psi)$  为输入, 执行如下步骤:
    1. 验证  $\text{OTS.Verify}(vk, y || \psi, \sigma) = 1$ . 若不成立, 则返回  $\perp$ , 否则执行后续步骤;
    2. 计算  $x \leftarrow \text{ATDR.TdInv}(td, vk, y)$ . 若  $x = \perp$ , 则返回  $\perp$ , 否则执行后续步骤;
    3. 计算  $M' = G(x) \oplus \psi$  并返回明文  $M'$ .



**正确性.** 上述加密方案的正确性可由自适应单向陷门关系的正确性直接推导出来. 下面主要介绍方案的 RKA-CCA 安全性的证明.

### 定理 5.9

如果 ATDR 是一个自适应单向陷门关系族, 且满足  $\Phi$ -密钥同态性和  $\Phi$ -指纹识别性, OTS 是一个强不可伪造一次签名方案, 那么定义 ?? 中的公钥加密方案是  $\Phi$ -RKA-CCA 安全的. 特别地, 对于任意攻击方案  $\Phi$ -RKA-CCA 安全性的敌手  $\mathcal{A}$ , 若  $\mathcal{A}$  询问解密预言机  $\mathcal{O}_{\text{rka}}(\cdot)$  的次数最多为  $Q$  次, 则敌手成功的优势满足如下关系式:

$$\text{Adv}_{\mathcal{A}, \Phi, \text{PKE}}^{\text{RKA-CCA}}(\kappa) \leq \text{Adv}_{\mathcal{B}_1, \text{OTS}}(\kappa) + \text{Adv}_{\mathcal{B}_2, \Phi, \text{ATDR}}^{\text{FP}}(\kappa) + \text{Adv}_{\mathcal{B}_3, \text{ATDR}}^{\text{OW}}(\kappa),$$

其中,  $\mathcal{B}_1$  是攻击一次签名强不可伪造性的敌手,  $\mathcal{B}_2$  是攻击自适应单向陷门关系  $\Phi$ -指纹识别性的敌手,  $\mathcal{B}_3$  是攻击自适应单向陷门关系单向性的敌手.  $\mathcal{B}_1$ 、 $\mathcal{B}_2$  和  $\mathcal{B}_3$  的运行时间与敌手  $\mathcal{A}$  的时间接近或增加与  $Q$  线性增长的多项式时间开销.



**证明** 令  $S_i$  表示敌手在 Game<sub>i</sub> 中的成功事件. 以游戏序列的方式组织证明如下:

Game<sub>0</sub>: 该游戏是标准的 RKA-CCA 游戏, 挑战者  $\mathcal{CH}$  和敌手  $\mathcal{A}$  交互如下:

- 初始化:  $\mathcal{CH}$  运行  $\text{Setup}(1^\kappa)$  生成公开参数  $pp$ , 同时运行  $\text{KeyGen}(pp)$  生成公私钥对  $(pk, sk)$ .  $\mathcal{CH}$  将  $(pp, pk)$  发送给  $\mathcal{A}$ .
- 询问: 对于敌手的任意询问  $(\phi, C)$ ,  $\mathcal{CH}$  首先判断  $\phi \in \Phi$  是否成立, 若不成立则返回  $\perp$ , 否则返回  $\text{Decrypt}(\phi(sk), C)$  的解密结果.
- 挑战:  $\mathcal{A}$  选择  $M_0, M_1 \in \mathbb{G}$  并发送给  $\mathcal{CH}$ .  $\mathcal{CH}$  选择随机比特  $\beta \in \{0, 1\}$ , 作如下计算:
  1. 运行  $(vk^*, sigk^*) \leftarrow \text{OTS.KeyGen}(pp_2)$  生成一次签名的公钥和私钥;
  2. 运行  $(x^*, y^*) \leftarrow \text{ATDR.Sample}(ek, vk^*)$  生成一个随机采样  $(x^*, y^*)$ ;
  3. 计算  $\psi^* = G(x^*) \oplus M_\beta$ ;
  4. 运行  $\sigma^* \leftarrow \text{OTS.Sign}(sigk^*, y^* || \psi^*)$ ;
  5. 输出密文  $C^* = (vk^*, \sigma^*, y^*, \psi^*)$  并发送给  $\mathcal{A}$ .
- 猜测:  $\mathcal{A}$  输出对  $\beta$  的猜测  $\beta'$ .  $\mathcal{A}$  成功当且仅当  $\beta' = \beta$ .

根据定义, 我们有:

$$\text{Adv}_{\mathcal{A}, \Phi, \text{PKE}}^{\text{RKA-CCA}}(\kappa) = |\Pr[S_0] - 1/2|$$

Game<sub>1</sub>: 该游戏与 Game<sub>0</sub> 的唯一不同在于拒绝解密查询的条件. 对于敌手提交的解密查询  $(\phi, C)$ , 其中  $C = (vk, \sigma, y, \psi)$ , 若  $vk = vk^*$ , 则  $\mathcal{CH}$  直接拒绝提供解密服务并返回  $\perp$ . 若  $vk \neq vk^*$ , 则  $\mathcal{CH}$  提供的解密查询服务与 Game<sub>0</sub> 完全一样. 下面分四种情况讨论敌手在两个连续游戏中的视图之间的区别.

- Case 1:  $vk \neq vk^*$ . 在这种情况下, 游戏 Game<sub>0</sub> 与 Game<sub>1</sub> 中的解密服务是完全一样的.
- Case 2:  $vk = vk^*$  且  $(\sigma, y || \psi) = (\sigma^*, y^* || \psi^*)$  且  $\phi(sk) = sk$ . 该情况实际上等价于  $(\phi(sk), C) = (sk, C^*)$ . 根据 RKA-CCA 安全模型的定义, 这种情况在两个游戏中都是不允许进行解密查询的.

- Case 3:  $vk = vk^*$  且  $(\sigma, y||\psi) \neq (\sigma^*, y^*||\psi^*)$ . 根据一次签名的强不可伪造性, 我们可以直接证明  $(y||\psi, \sigma)$  通过验证的概率是可忽略的. 具体地, 我们有

$$\Pr[\text{Verify}(vk, y||\psi, \sigma) = 1] \leq \text{Adv}_{\mathcal{B}_1, \text{OTS}}(\kappa).$$

- Case 4:  $vk = vk^*$  且  $(\sigma, y||\psi) = (\sigma^*, y^*||\psi^*)$  且  $\phi(sk) \neq sk$ . 根据 ATDR 的  $\Phi$ -指纹识别性, 解密服务在计算  $x \leftarrow \text{ATDR.TdInv}(td, vk, y)$  时,  $x \neq \perp$  的概率是可忽略的. 若  $x = \perp$ , 则解密预言机直接返回  $\perp$ . 此时, 两个游戏中解密预言机返回的结果是一样的. 特别地, 我们有

$$\Pr[\text{ATDR.TdInv}(\phi(sk), vk^*, y) \neq \perp] \leq \text{Adv}_{\mathcal{B}_2, \Phi, \text{ATDR}}^{\text{FP}}(\kappa).$$

由上述分析可知, 敌手在两个游戏中的视图区别是可忽略的. 具体地, 我们有

$$|\Pr[S_0] - \Pr[S_1]| \leq \text{Adv}_{\mathcal{B}_1, \text{OTS}}(\kappa) + \text{Adv}_{\mathcal{B}_2, \Phi, \text{ATDR}}^{\text{FP}}(\kappa).$$

**Game<sub>2</sub>**: 该游戏与 Game<sub>1</sub> 的唯一不同在于挑战者利用 ATDR 的自适应性来回答解密询问. 对于敌手提交的解密查询  $(\phi, C)$ , 其中  $C = (vk, \sigma, y, \psi)$ , 解密预言机的工作方式如下:

- 若  $vk = vk^*$  或者  $\text{OTS.Verify}(vk, y||\psi, \sigma) = 0$ , 返回  $\perp$ ;
- 计算  $x := \text{ATDR.TdInv}(td, vk, T(pp, \phi, vk, y))$ . 如果  $x = \perp$ , 返回  $\perp$ ;
- 否则, 计算并返回  $M' = G(x) \oplus \psi$ .

根据  $\Phi$ -密钥同态性, 我们有  $\text{ATDR.TdInv}(td, vk, T(pp, \phi, vk, y)) = \text{ATDR.TdInv}(\phi td, vk, y)$ . 也就是说, 挑战者在 Game<sub>2</sub> 中模拟的解密预言机和 Game<sub>1</sub> 中的是完全一致的. 因此, 我们有

$$\Pr[S_1] = \Pr[S_2].$$

**Game<sub>3</sub>**: 该游戏与 Game<sub>2</sub> 的唯一不同在于挑战密文中  $\psi^*$  的计算方式.  $\mathcal{CH}$  随机选择  $K \xleftarrow{\text{R}} \{0, 1\}^m$ , 计算  $\psi^* = K \oplus M_\beta$ . 根据 ATDR 的自适应单向性及函数  $G$  输出结果的伪随机性, 我们可以证明敌手在这两个连续游戏中的视图区别不超过  $\text{Adv}_{\mathcal{B}_3, \text{ATDR}}^{\text{OW}}(\kappa)$ . 故, 我们有

$$|\Pr[S_2] - \Pr[S_3]| \leq \text{Adv}_{\mathcal{B}_3, \text{ATDR}}^{\text{OW}}(\kappa),$$

其中  $\mathcal{B}_3$  是攻击自适应单向陷门关系的单向性的 PPT 敌手.

在最后一个游戏中, 由于  $\psi^*$  的分布与挑战比特  $\beta$  完全独立不相关, 从而敌手在游戏中的优势为 0, 即

$$\Pr[S_3] = \frac{1}{2}.$$

综上, 定理 ?? 得证. □

### 5.2.3 CQZ+22 方案

上世纪 90 年代, 哥德尔奖得主 Cynthia Dwork 和 Moni Naor 引入的不可延展性是密码学中单向性和伪随机性之外的另一重要性质, 该性质精准刻画了密码组件输入/输出之间的独立性. 已有的研究工作考察了加密、承诺、零知识证明、编码、程序混淆的不可延展性, 然而一直未涉及密码学乃至计算机科学中最基本的函数. 函数的不可延展性与单向性之间存在怎样的关联以及如何构造高效的不可延展函数均是未解的公开问题.

2022 年, Chen 等人 [CQZDC-JOC-2022] 在 PKC 2016 工作 [Chen-PKC-2016] 的基础上进一步完善了不可延展函数的性质及构造, 成功解决了上述问题. 在理论层面, 首次绘制出函数不可延展性与单向性之间的清晰图景, 通过巧妙结合方程求解技巧和变换集代数性质, 建立起不可延展函数与单向函数之间的关联, 并分别在标准模型

和随机预言机模型中给出了通用构造, 解决了美国 Georgia Tech 大学 Alexandra Boldyreva 教授和德国 Bochum 大学 Eike Kiltz 教授等著名密码学家提出的公开问题. 在应用层面, 不仅直接蕴含了密码谜题的高效设计, 还深度揭示了不可延展函数在抗篡改安全中的强力应用: (1) 证明了对于代数诱导的变换集, 抗非平凡拷贝攻击属于密码方案的内蕴性质, 从而直接提升了一大批密码方案的抗相关密钥攻击安全性; (2) 构造出了迄今为止效率和安全均最优的认证密钥导出函数, 提供了将传统安全提升为抗篡改安全的关键技术工具.

### 5.2.3.1 不可延展函数

#### 定义 5.8 (有效计算函数)

一个有效计算函数族  $\mathcal{F}$  包含以下三个概率多项式时间算法 ( $\text{Gen}, \text{Eval}, \text{Vefy}$ ):

- $\text{Gen}(1^\kappa)$ : 输入安全参数  $\kappa$ , 输出一个函数索引  $s$ . 每个函数索引  $s$  定义了一个函数  $f_s : X_s \rightarrow Y_s$ . 该函数可以是确定性的也可以是随机性的.
- $\text{Eval}(s, x)$ : 输入函数索引  $s$  和一个元素  $x \in X_s$ , 输出函数的一个像值  $y \leftarrow f_s(x)$ . 令  $\text{supp}(f_s(x))$  是随机变量  $f_s(x)$  的支撑集. 如果  $f_s$  是确定的, 则  $\text{supp}(f_s(x))$  缩减为包含唯一像值  $f_s(x)$  的集合.
- $\text{Vefy}(s, x, y)$ : 输入函数索引  $s$  和  $(x, y) \in X_s \times Y_s$ , 当  $y \in \text{supp}(f_s(x))$  时, 输出 “1”, 否则, 输出 “0”. 对于确定性函数, 可以直接通过重新计算  $x$  的像值来验证.



**笔记** 上述定义统一了确定函数和随机函数的概念. 对于任意两个不同的原像  $x_1, x_2 \in X_s$ , 如果  $\text{supp}(f(x_1))$  和  $\text{supp}(f(x_2))$  没有交集, 则  $f$  是单射函数.

下面介绍有效计算函数的单向性 (One-Wayness) 和不可延展性 (Non-malleability).

#### 定义 5.9 (单向性和自适应单向性)

定义  $\mathcal{F}$  的单向性敌手的优势函数如下:

$$\text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{ow}}(\kappa) = \Pr \left[ x \in f_s^{-1}(y^*) : \begin{array}{l} s \leftarrow \text{Gen}(1^\kappa); \\ x^* \xleftarrow{\text{R}} X_s, y^* \leftarrow f_s(x^*); \\ x \leftarrow \mathcal{A}(f_s, y^*); \end{array} \right].$$

对于任意 PPT 敌手  $\mathcal{A}$ , 如果优势函数  $\text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{ow}}(\kappa)$  关于安全参数  $\kappa$  是可忽略的, 则  $\mathcal{F}$  是单向的. 如果允许  $\mathcal{A}$  访问除了  $y^*$  之外的任意  $y$  的求逆预言机  $\mathcal{O}_{\text{inv}}$  情况下,  $\mathcal{F}$  仍然是单向的, 则称  $\mathcal{F}$  是自适应单向的.



#### 定义 5.10 (不可延展性和自适应不可延展性)

令  $\Phi$  是一个定义域  $X$  上的变换函数集. 定义  $\mathcal{F}$  的不可延展性敌手的优势函数如下:

$$\text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{nm}}(\kappa) = \Pr \left[ \phi \in \Phi \wedge \text{Vefy}(s, \phi(x^*), y) = 1 : \begin{array}{l} s \leftarrow \text{Gen}(1^\kappa); \\ x^* \xleftarrow{\text{R}} X_s, y^* \leftarrow f_s(x^*); \\ (\phi, y) \neq (\text{id}, y^*) \\ (\phi, y) \leftarrow \mathcal{A}(f_s, y^*); \end{array} \right].$$

对于任意 PPT 敌手  $\mathcal{A}$ , 如果优势函数  $\text{Adv}_{\mathcal{A}, \mathcal{F}}^{\text{nm}}(\kappa)$  关于安全参数  $\kappa$  是可忽略的, 则  $\mathcal{F}$  是  $\Phi$ -不可延展的. 如果允许  $\mathcal{A}$  访问除了  $y^*$  之外的任意  $y$  的求逆预言机  $\mathcal{O}_{\text{inv}}$  情况下,  $\mathcal{F}$  仍然是不可延展的, 则称  $\mathcal{F}$  是自适应  $\Phi$ -不可延展的.



**笔记** 一般来说, 一个函数族的定义域和值域范围依赖函数索引. 为简便起见, 我们可以假设对于所有函数索引  $s$ , 定义域和值域都是不变的. 从而将  $X_s, Y_s$  和  $f_s$  分别简写为  $X, Y$  和  $f$ . 在不可延展函数中, 存在一些不可能转换函数类, 如单位变换  $\text{id}$  和常量变换  $\phi_c$ , 无法实现不可延展性. 敌手可以输出  $(\text{id}, y^*)$  或  $(\phi_c, f(c))$  从而赢得游戏. 因此, 给出一个可能实现不可延展性的变换函数集  $\Pi$  是非常重要的.

**定义 5.11 (一般变换函数集)**

定义满足下面两个性质的变换函数集  $\Phi_{\text{brs}}^{\text{srs}}$ :

- **有界根集合 (Bounded Root Space):** 令  $r(\kappa)$  是安全参数  $\kappa$  的一个变量,  $R_\phi = \{x \in X : \phi(x) = 0\}$ . 如果  $|R_\phi| \leq r(\kappa)$ , 则  $\phi$  最多有  $r(\kappa)$  个根. 如果对于每个  $\phi \in \Phi$  和  $\phi_c \in \text{cf}$ , 变换函数  $\phi' = \phi - \phi_c$  和  $\phi'' = \phi - \text{id}$  都最多有  $r(\kappa)$  个根, 那么称变换函数集  $\Phi$  有  $r(\kappa)$ -有界根集.
- **可采样根集合 (Sampleable Root Space):** 如果存在一个 PPT 算法 SampRS, 输入  $\phi$ , 均匀随机地输出集合  $R_\phi$  中的一个元素, 则称变换函数  $\phi$  有一个可采样根集. 如果对于每个  $\phi \in \Phi$  和  $\phi_c \in \text{cf}$ , 复合函数  $\phi' = \phi - \phi_c$  和  $\phi'' = \phi - \text{id}$  都有可采样根集, 那么称变换函数集  $\Phi$  有可采样根集.



**笔记** 变换函数集  $\Phi_{\text{brs}}^{\text{srs}}$  非常强大, 几乎包含了所有除去单位变换  $\text{id}$  和常量变换  $\text{cf}$  的代数诱导变换函数集, 如线性变换集  $\Phi^{\text{lin}} = \{\phi_a : \phi_a(x) = a + x\}_{a \in \mathbb{G}}$ , 仿射变换集  $\Phi^{\text{aff}} = \{\phi_{a,b} : \phi_{a,b}(x) = ax + b\}_{a,b \in \mathbb{R}}$  和多项式变换集  $\Phi^{\text{poly}(d)} = \{\phi_q : \phi_q(x) = q(x)\}_{q \in \mathbb{F}_d(x)}$ , 其中  $\mathbb{G}$  是一个群,  $\mathbb{R}$  是一个环,  $\mathbb{F}_d(x)$  是有限域  $\mathbb{F}$  上次数不超过  $d$  的多项式集.

**5.2.3.2 不可延展性与单向性之间的关系**

首先, 概括地给出函数的不可延展性与单向性之间的关系, 如图 ?? 所示. 然后, 介绍图中转换关系相关几个重要引理.

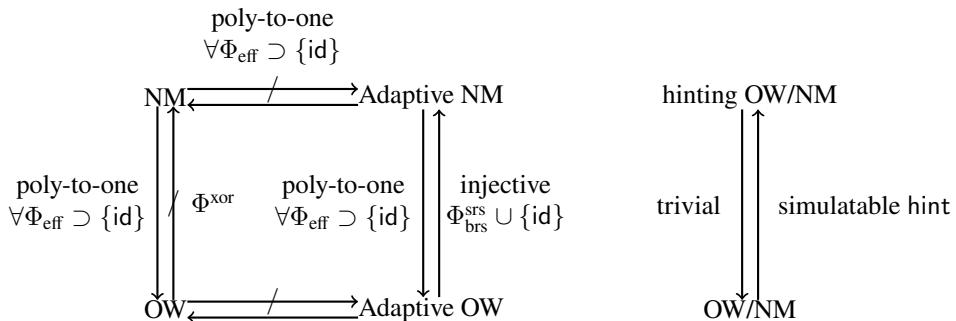


图 5.12: 不可延展函数与单向函数之间的关系

**引理 5.10**

令  $\mathcal{F}$  是一个多对一函数族. 对于任意可实现的变换集  $\Phi$ , 则:  $\Phi$ -不可延展性  $\Rightarrow$  单向性.



**证明** 假设  $\mathcal{A}$  是一个以不可忽略概率攻破  $\mathcal{F}$  单向性的敌手, 我们可以构造另一个算法  $\mathcal{B}$  以不可忽略的概率攻破  $\mathcal{F}$  的不可延展性. 算法  $\mathcal{B}$  模拟敌手  $\mathcal{A}$  在单向性游戏中的挑战者的过程如下:

1. 初始化及挑战: 给定一个函数  $f \leftarrow \mathcal{F}.\text{Gen}(1^\kappa)$  和一个像值  $y^* \leftarrow f(x^*)$ , 其中  $x^* \xleftarrow{R} X$ ,  $\mathcal{B}$  将  $(f, y^*)$  发送给敌手  $\mathcal{A}$ .
2. 攻击: 当敌手  $\mathcal{A}$  输出一个解  $x$  时,  $\mathcal{B}$  随机选择一个变换函数  $\phi \in \Phi \setminus \{\text{id}\}$ , 返回  $(\phi, f(\phi(x)))$  作为对  $\mathcal{F}$  不可延展性的攻击结果.

由于  $\mathcal{F}$  是多对一的, 在  $\mathcal{A}$  攻击成功的条件下, 即  $x \in f^{-1}(y^*)$ , 我们有  $\Pr[x = x^* | y^*] \geq 1/\text{poly}(\kappa)$ , 其中  $x^* \xleftarrow{R} X$ . 这是因为最多有多项式个数  $x$  满足  $f(x) = y^*$ , 并且每个  $x$  在敌手  $\mathcal{A}$  的角度看都是一样的. 因此, 如果  $\mathcal{A}$  能够以不可忽略的概率攻破  $\mathcal{F}$  的单向性, 那么算法  $\mathcal{B}$  同样能够以不可忽略的概率攻破  $\mathcal{F}$  的不可延展性.  $\square$

引理 ?? 反向结论并不成立, 也就是说一个函数满足单向性但并不一定满足不可延展性.

**引理 5.11**

单向性  $\not\Rightarrow \Phi^{\text{xor}}$ -不可延展性.



### 证明

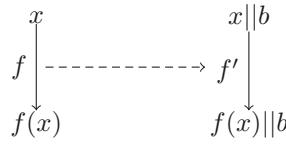


图 5.13: 可延展函数示例

引理 ??可通过反证法证明。我们可以通过一个单向函数  $f$  构造另一个单向函数  $f'$ , 如图 ??所示。显而易见,  $f'$  仍然满足单向性, 但是不满足异或变换集下的不可延展性。□

**笔记** 思考一下, 不可延展性的反例还有哪些? 其实  $\Phi$ -同态的单向函数就一个很好的反例。所谓  $\Phi$  同态性, 对于任意  $\phi \in \Phi$  和任意  $x \in X$ , 则有  $f(\phi(x)) = \phi(f(x))$ 。显而易见,  $f$  是单向的, 但不是  $\Phi$ -不可延展的。

### 引理 5.12

对于  $\mathcal{F}$  上任意可实现的变换集  $\Phi$  of  $\mathcal{F}$ , 自适应  $\Phi$ -不可延展性  $\Rightarrow$  自适应单向性。



**证明** 引理 ??的证明可以直接通过引理 ??的结论推出。□

### 引理 5.13

当  $\mathcal{F}$  是单射函数时, 对于  $\Phi = \Phi_{\text{brs}}^{\text{srs}} \cup \text{id}$ ,  $(q+1)$ -自适应单向性  $\Rightarrow q$ -自适应  $\Phi$ -不可延展性。



**证明** 假设  $\mathcal{A}$  是一个攻击  $\mathcal{F}$  自适应不可延展性的对手。我们可以构建一个攻击  $\mathcal{F}$  自适应单向性的对手  $\mathcal{B}$ .  $\mathcal{B}$  按以下方式模拟  $\mathcal{A}$  在自适应不可延展游戏中的挑战者:

初始化及挑战: 给定  $f \leftarrow \mathcal{F}.\text{Gen}(1^\kappa)$  和一个像值  $y^* \leftarrow f(x^*)$ , 其中  $x^* \xleftarrow{\text{R}} X$ ,  $\mathcal{B}$  将  $(f, y^*)$  作为挑战信息发送给对手  $\mathcal{A}$ .

攻击: 当  $\mathcal{A}$  询问求逆谕言机时,  $\mathcal{B}$  将询问直接发送给自己的挑战者并将返回的结果发送给  $\mathcal{A}$ 。当  $\mathcal{A}$  输出一个攻击结果  $(\phi, y) \neq (\text{id}, y^*)$  时,  $\mathcal{B}$  按下面的方式进行处理:

- Case 1:  $\phi = \text{id} \wedge y \neq y^*$ .  $\mathcal{B}$  询问求逆谕言机  $\mathcal{O}_{\text{inv}}$ , 获取  $y$  的逆  $x$ , 再将  $x$  输出作为  $\mathcal{B}$  的求解结果。
- Case 2:  $\phi \in \Phi_{\text{brs}}^{\text{srs}} \wedge y \neq y^*$ .  $\mathcal{B}$  询问求逆谕言机  $\mathcal{O}_{\text{inv}}$ , 获得  $y$  的逆  $x$ , 再运行  $\text{SampRS}(\phi')$  输出  $\phi'(\alpha) = 0$  的一个随机解, 其中  $\phi'(\alpha) = \phi(\alpha) - x$ 。
- Case 3:  $\phi \in \Phi_{\text{brs}}^{\text{srs}} \wedge y = y^*$ .  $\mathcal{B}$  运行  $\text{SampRS}(\phi'')$  输出  $\phi''(\alpha) = 0$  的一个随机解, 其中  $\phi''(\alpha) = \phi(\alpha) - \alpha$ .

下面分析  $\mathcal{B}$  的策略的正确性。在  $\mathcal{A}$  攻击成功的条件下, 我们有  $\text{Vefy}(f, \phi(x^*), y) = 1$ 。利用  $\mathcal{F}$  的单射性质, 对于情形 1, 我们有  $\text{id}(x^*) = x^* = x$ ; 对于情形 2, 我们有  $\phi(x^*) = x$ , 即,  $x^*$  是  $\phi'(\alpha) = 0$  的一个解; 对于情形 3, 我们有  $\phi(x^*) = x^*$ , 即,  $x^*$  是  $\phi''(\alpha) = 0$  的一个解。将这三种情形结合起来, 利用变换集  $\Phi_{\text{brs}}^{\text{srs}}$  的 BRS&SRS 性质,  $\mathcal{B}$  通过最多  $(q+1)$  次求逆询问输出正确解  $x^*$  的概率为  $1/\text{poly}(\kappa)$ 。因此, 如果  $\mathcal{A}$  攻破  $q$ -自适应不可延展性的概率是不可忽略的, 那么  $\mathcal{B}$  攻破  $(q+1)$ -自适应单向性的概率也是不可忽略的。引理 ??得证。□

### 引理 5.14

当  $\mathcal{F}$  是多对一函数族时, 对于任意可实现的变换集  $\Phi \supset \{\text{id}\}$ ,  $\Phi$ -不可延展性  $\not\Rightarrow$  自适应  $\Phi$ -不可延展性。



**证明** 我们通过寻找反例来证明引理 ??的结论。令  $\mathcal{F} = (\text{Gen}, \text{Eval}, \text{Vefy}, \text{TdInv})$  是一个带陷门的  $\Phi$ -不可延展函数族。如图 ??所示, 我们从  $\mathcal{F}$  构造另一个函数族  $\mathcal{F}'$ , 使得  $\mathcal{F}'$  仍然是  $\Phi$ -不可延展的, 但不是自适应  $\Phi$ -不可延展的。对于任意  $f \in \mathcal{F}$ , 假设  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ , 则  $f' \in \mathcal{F}'$  的定义如下:

$$\begin{aligned} f' : \quad \{0, 1\}^n &\rightarrow \{0, 1\}^{m+1} \\ x &\rightarrow \quad 0 || f(x) \end{aligned}$$

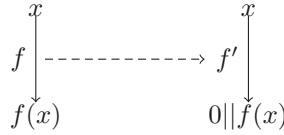


图 5.14: 自适应可延展函数示例

对于任意  $y' = b||y \in \{0, 1\}^{m+1}$ ,  $\mathcal{F}'$  求逆函数的定义如下:

$$f'^{-1}(y') = \begin{cases} f^{-1}(y) & \text{if } b = 0 \\ td & \text{if } b = 1 \end{cases}$$

上面实际上定义了一个“危险的”求逆算法, 对于非法原像  $b||y$ (其中  $b = 1$ ), 求逆算法将直接输出  $f$  的陷门. 因此, 在自适应不可延展游戏中, 敌手可以通过这类“危险的”询问获取求逆陷门从而攻破  $f'$  的不可延展性. 而在不可延展性游戏里, 由于没有提供求逆预言机,  $f'$  依然保持有不可延展性. 综上, 引理 ?? 得证.  $\square$

#### 引理 5.15 (Computationally Simulatable Case)

令  $\mathcal{F}$  是一个单射函数族,  $\mathcal{H}: X \rightarrow K$  是  $\mathcal{F}$  的 hardcore functions. 那么, 对于任意变换集  $\Phi \subseteq \Phi_{\text{brs}}^{\text{srs}} \cup \text{id}$ , 自适应  $\Phi$ -不可延展性蕴含  $\mathcal{H}$ -hinting 自适应  $\Phi$ -不可延展性



### 5.2.3.3 不可延展函数的构造

下面利用自适应单向陷门函数和全除一有损陷门函数分别构造确定性不可延展函数和随机化不可延展函数. 对于确定性不可延展函数, 可以通过自适应单向陷门函数直接构造, 有如下结论:

#### 定理 5.10

令  $\mathcal{F}$  是一个单射的自适应单向陷门函数族,  $\mathcal{H}$  是与之相关的 hardcore functions. 那么  $\mathcal{F}$  是一个自适应  $\mathcal{H}$ -hinting  $\Phi$ -不可延展的, 其中  $\Phi = \Phi_{\text{brs}}^{\text{srs}} \cup \{\text{id}\}$ .



**证明** 根据引理 ??,  $\mathcal{F}$  是自适应  $\Phi$ -不可延展的. 根据引理 ??,  $\mathcal{F}$  也是自适应  $\mathcal{H}$ -hinting 不可延展的. 定理得证!  $\square$

随机化的不可延展函数的构造需要用到两个密码工具: 全除一有损函数 [Qin-PKC-2014](可以看作是不带陷门的全除一有损单向陷门函数) 和一次签名方案. 假设 ABOLF = {Gen, Eval} 是一个  $(X, Z, \tau)$ -全除一有损函数, 分支空间为  $B = \{0, 1\}^d$ . OTS = (Setup, Gen, Sign, Verify) 是一个强不可伪造一次签名方案, 其中验证密钥空间满足  $VK \subseteq B$ , 签名空间为  $\Sigma$ ,  $Y = B \times Z \times \Sigma$ . 令  $n = \log |X|$ ,  $\tau = \log |Z|$ . 下面构造一个从  $X$  到  $Y$  的不可延展函数.

#### 构造 5.10 (随机化的不可延展函数)

- Gen( $1^\kappa$ ): 输入安全参数  $\kappa$ , 输出  $s \leftarrow \text{ABOLF.Gen}(1^\kappa, 0^d)$ .
- Eval( $s, x$ ): 输入函数索引  $s$  和原像  $x \in X$ , 执行以下步骤:
  1. 生成一对一次签名密钥  $(vk, sigk) \leftarrow \text{OTS.Gen}(1^\kappa)$ ;
  2. 计算  $z \leftarrow g_{s, vk}(x)$ ,  $\sigma \leftarrow \text{OTS.Sign}(sigk, z)$ ;
  3. 输出  $y = (vk, z, \sigma)$ .
- Verify( $s, x, y$ ): 输入  $s, x$  和  $y = (vk, z, \sigma)$ , 如果  $z = g_{s, vk}(x) \wedge \text{OTS.Verify}(vk, z, \sigma) = 1$ , 输出“1”, 否则输出“0”.



**定理 5.11**

令  $\mathcal{H} : X \rightarrow K = \{0,1\}^\ell$  是  $\mathcal{F}$  的一个 hardcore 函数族. 则  $\mathcal{F}$  是一个  $\mathcal{H}$ -hinting  $\Phi$ -不可延展函数族, 其中  $\Phi = \Phi^{\text{poly}(d)} \setminus \text{cf}$ ,  $\log d \leq n - \tau - \ell - \omega(\log \lambda)$ .



**证明** 令  $S_i$  表示敌手在  $\mathcal{A}$  在 Game<sub>i</sub> 中成功的事件. 以游戏序列的方式组织证明如下:

Game<sub>0</sub>: 该游戏是标准的 hinting 不可延展性游戏. 挑战者  $\mathcal{CH}$  与敌手  $\mathcal{A}$  按如下方式执行游戏.

1. 初始化:  $\mathcal{CH}$  通过运行  $s \leftarrow \text{ABOLF.Gen}(1^\kappa, 0^d)$  生成  $\mathcal{F}$  的一个随机索引  $s$ , 并选择  $h \xleftarrow{R} \mathcal{H}$ , 将  $(s, h)$  发送给  $\mathcal{A}$ .
2. 挑战:  $\mathcal{CH}$  选择  $x^* \xleftarrow{R} X$ , 生成  $(vk^*, sigk^*) \leftarrow \text{OTS.Gen}(1^\kappa)$ , 计算  $z^* \leftarrow g_{s, vk^*}(x^*)$ ,  $\sigma^* \leftarrow \text{OTS.Sign}(sk^*, z^*)$ , 将  $(y^* = (vk^*, z^*, \sigma^*), h(x^*))$  发送给  $\mathcal{A}$ , 其中  $y^*$  是函数的像值,  $h(x^*)$  是 hint 函数值.
3. 攻击:  $\mathcal{A}$  输出一对元素  $(\phi, y = (vk, z, \sigma))$ . 如果  $\mathcal{F}.Verify(s, \phi(x^*), y) = 1$ , 即  $z = g_{s, vk}(\phi(x^*))$ ,  $\text{OTS.Verify}(vk, z, \sigma) = 1$ , 则  $\mathcal{A}$  成功.

根据定义, 我们有:

$$\text{Adv}_{\mathcal{A}} = \Pr[S_0]$$

Game<sub>1</sub>: 该游戏与 Game<sub>0</sub> 的唯一区别是  $\mathcal{A}$  的攻击结果  $(\phi, y^*)$  成功的条件定义为  $\phi(x^*) = x^* \wedge \phi \in \Phi \setminus \{\text{id}\}$ . 由于  $g_{s, vk^*}(\cdot)$  是一个单射函数,  $z^*$  的值完全决定了它的原像, 所以敌手成功的定义仅是一种概念上的变化. 因此, 我们有

$$\Pr[S_0] = \Pr[S_1].$$

Game<sub>2</sub>: 该游戏与 Game<sub>1</sub> 的唯一区别在于挑战者在初始化阶段就生成  $(vk^*, sigk^*)$  并且  $s$  的生成方式由  $\text{ABOLF.Gen}(1^\lambda, vk^*)$  代替  $\text{ABOLF.Gen}(1^\lambda, 0^d)$ . 如果存在一个敌手在两个游戏中的视图是可以区分的, 那么我们可以构造一个归约算法来攻破 ABOLF 的隐藏分支性质. 因此, 我们有:

$$|\Pr[S_1] - \Pr[S_2]| \leq \text{negl}(\kappa)$$

下面重点分析概率  $\Pr[S_2]$ . 在 Game<sub>2</sub> 中, 当下列条件之一成立, 则  $\mathcal{A}$  的攻击结果  $(\phi, x)$  成功

- $E_1: y = y^* \text{ 且 } \phi(x^*) = x^* \wedge \phi \in \Phi \setminus \{\text{id}\}$ .
- $E_2: vk \neq vk^* \text{ 且 } \text{OTS.Verify}(vk, z, \sigma) = 1 \wedge g_{s, vk}(\phi(x^*)) = z \wedge \phi \in \Phi$ .
- $E_3: vk = vk^* \wedge (z, \sigma) \neq (z^*, \sigma^*) \text{ 且 } \text{OTS.Verify}(vk^*, z, \sigma) = 1 \wedge g_{s, vk^*}(\phi(x^*)) = z \wedge \phi \in \Phi$ .

显然,  $S_2 = E_1 \vee E_2 \vee E_3$ . 下面分析概率  $\Pr[E_i]$  的上界, 其中  $1 \leq i \leq 3$ .

值得注意的是  $\mathcal{A}$  在输出  $(\phi, y)$  前的视图信息是  $(s, h, y^* = (vk^*, z^*, \sigma^*), h(x^*))$ . 我们有:

$$\tilde{H}_\infty(x^*|view) = \tilde{H}_\infty(x^*|(z^*, \sigma^*, h(x^*))) \quad (5.18)$$

$$= \tilde{H}_\infty(x^*|(z^*, h(x^*))) \quad (5.19)$$

$$\geq H_\infty(x^*) - \tau - \ell = n - \tau - \ell \quad (5.20)$$

在上述推导过程中, 公式 (5.18) 源于  $s, h$  和  $vk^*$  与  $x^*$  独立这一事实. 公式 (5.19) 源于  $\sigma^*$  是从  $sk^*$  和  $z^*$  导出且  $sk^*$  与  $x^*$  独立这一事实. 在 Game<sub>2</sub> 中,  $g_{s, vk^*}(\cdot)$  是一个有损函数, 其像空间尺寸最大为  $2^\tau$ . 公式 (5.20) 源于引理 ?? 和  $(z^*, h(x^*))$  最多有  $2^{\tau+\ell}$  可能取值这一事实.

由于  $\tilde{H}_\infty(x^*|view) \geq n - \tau - \ell$  以及变换函数  $\phi \in \Phi \setminus \{\text{id}\}$  的 IOCR 性质 (参见文献 [ChenQZDC22] 的引理 4.2), 我们有  $\Pr[E_1] \leq 1/2^{n-\tau-\ell-\log d}$ .

根据  $\phi \in \Phi$  的 HOE 性质 (参见文献 [ChenQZDC22] 的引理 4.2), 我们有  $\tilde{H}_\infty(\phi(x^*)|view) \geq n - \tau - \ell - \log d$ . 对于所有  $vk \neq vk^*$ ,  $g_{s, vk}(\cdot)$  是一个单射函数, 故  $z = g_{s, vk}(\phi(x^*))$  的平均极小熵与  $\phi(x^*)$  一样. 因此, 我们有  $\Pr[E_2] \leq 1/2^{n-\tau-\ell-\log d}$ .

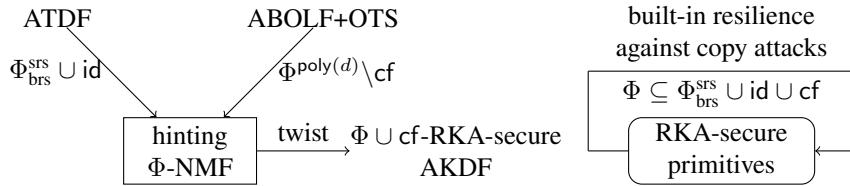


图 5.15: 不可延展函数的构造及应用.

由于选择的参数  $d$  满足  $\omega(\lambda) \leq n - \tau - \ell - \log d$ , 所以我们有  $\Pr[E_1]$  和  $\Pr[E_2]$  关于安全参数  $\kappa$  都是可忽略的. 根据一次签名的强不可伪造性, 我们有  $\Pr[E_3] \leq \text{Adv}_{\mathcal{A}}^{\text{OTS}} \leq \text{negl}(\lambda)$ .

通过上述分析, 可得  $\Pr[S_2]$  关于安全参数  $\kappa$  是可忽略的.

综上, 定理得证! □.

**笔记**  $\mathcal{F}$  的 hardcore 函数族可以是一个定义在  $X$  到  $\{0, 1\}^\ell$  的通用哈希函数族.

### 5.2.3.4 不可延展函数的应用

2015 年, Qin 等人 [Qin-PKC-2015] 将不可延展密钥派生函数 (Non-Malleable Key Derivation Functions, NM-KDFs) [FaustMVW14] 推广为连续不可延展密钥派生函数 (Continuous Non-Malleable KDFs, CNM-KDFs), 并提出一种利用连续不可延展密钥派生函数设计抗相关密钥攻击的密码原语 (包括公钥加密, 数字签名, 基于身份加密等) 的通用模式. 2022 年, Chen 等人 [CQZDC-JOC-2022] 进一步简化了此概念的名称, 称之为抗相关密钥攻击的可认证密钥派生函数 (Authenticated KDFs, 简称 AKDFs), 使其名称更能展现出它的性质, 并且增强了 CNM-KDFs 的安全性. 因此, 以 AKDFs 为跳板, 可以得出设计出性能良好、变换函数集范围广的多种抗相关密钥攻击的密码原语. 图 ??展示了不可延展函数的构造及其在 RKA 安全密码原语方面的应用.

下面, 我们重点回顾 AKDFs 的形式化定义及基于不可延展函数构造的 AKDFs 构造.

#### 定义 5.12 (可认证密钥派生函数)

一个可认证密钥派生函数 AKDFs 包含以下三个多项式时间算法:

- $\text{Setup}(1^\kappa)$ : 输入安全参数  $\kappa$ , 输出系统参数  $pp$ , 并定义了原始密钥空间  $X$ , 认证标签空间  $T$  和派生密钥空间  $K$ .
- $\text{Sample}(pp)$ : 输入公开参数  $pp$ , 选择一个原始密钥  $x \xleftarrow{R} X$ , 计算它的认证标签  $t \in T$ , 输出  $(x, t)$ .
- $\text{Derive}(x, t)$ : 输入原始密钥  $x \in X$  和标签  $t \in T$ , 输出一个派生密钥  $k \in K$  或者一个拒绝符号  $\perp$ , 表示  $t$  不是  $x$  的合法标签.



**相关密钥攻击安全性.** 令  $\Phi$  是一个定义在原始密钥空间  $X$  上的一个变换函数集. 假设  $\Phi$  包含单位变换  $\text{id}$ . 定义 AKDFs 敌手的优势函数如下:

$$\text{Adv}_{\mathcal{A}, \text{AKDF}}^{\text{rka}}(\kappa) = \Pr \left[ \beta' = \beta : \begin{array}{l} pp \leftarrow \text{Setup}(1^\kappa); \\ (x^*, t^*) \leftarrow \text{Sample}(pp); \\ k_0^* \leftarrow \text{Derive}(x^*, t^*), k_1^* \xleftarrow{R} K; \\ \beta \xleftarrow{R} \{0, 1\}; \\ \beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{derive}}^\Phi}(pp, t^*, k_\beta^*); \end{array} \right] - \frac{1}{2}.$$

对于任意 PPT 敌手  $\mathcal{A}$ , 如果优势函数  $\text{Adv}_{\mathcal{A}, \text{AKDF}}^{\text{rka}}(\kappa)$  关于安全参数  $\kappa$  是可忽略的, 那么称 AKDFs 是  $\Phi$ -RKA-安全的. 其中  $\mathcal{O}_{\text{derive}}^\Phi$  是相关密钥派生预言机, 输入  $\langle \phi, t \rangle \neq \langle \text{id}, t^* \rangle$ , 返回  $\text{Derive}(\phi(x^*), t)$ . 在实验中,  $\mathcal{A}$  可以自适应地询问预言机  $\mathcal{O}_{\text{derive}}^\Phi$ . 但是, 敌手不能进行形如将  $\langle \text{id}, t^* \rangle$  的非法询问, 否则敌手必定成功了. 根据  $\phi \in \text{cf}$  还是  $\phi \in \Phi$ , 合法询问  $\langle \phi, t \rangle \neq \langle \text{id}, t^* \rangle$  可以进一步分为常量询问和非常量询问.

在证明一个 AKDFs 方案的 RKA 安全性时, 首要任务在不知道原始密钥  $x^*$  的情况下如何回答敌手的相关密钥派生询问. 一种简单粗暴的方式是与其设法回答敌手的相关密钥派生询问, 不如直接拒绝回答敌手的所有 RKA 询问. 我们希望即时敌手在看到挑战信息  $(x^*, t^*)$  的情况下, 也无法生成一个合法的询问  $\langle \phi, t \rangle$ , 使得  $t$  是  $\phi(x^*)$  的合法认证标签. 因此, 在回答敌手的相关密钥派生询问时, 挑战者(模拟者)直接返回  $\perp$  即可. 下面介绍如何利用不可延展函数巧妙地构造 RKA 安全的可认证密钥派生函数.

### 构造 5.11 (基于不可延展函数的密钥派生函数)

令  $\mathcal{F} = (\text{Gen}, \text{Eval}, \text{Verify})$  是一个  $\mathcal{H}$ -hinting  $\Phi$ -不可延展函数族, 其中  $\mathcal{H} : X \rightarrow K$  是一个 hardcore 函数,  $\Phi$  是一个包含单位变换  $\text{id}$  的变换集.

- $\text{Setup}(1^\kappa)$ : 输入安全参数  $\kappa$ , 运行  $f \leftarrow \mathcal{F}.\text{Gen}(1^\kappa)$ , 选择一个 hardcore 函数  $h \leftarrow \mathcal{H}$ , 输出系统参数  $pp = (f, h)$ .
- $\text{Sample}(pp)$ : 输入公开参数  $pp = (f, h)$ , 随机选择一个原始密钥  $x \xleftarrow{\text{R}} X$ , 计算  $t \leftarrow f(x)$ , 输出  $(x, t)$ .
- $\text{Derive}(x, t)$ : 输入原始密钥  $x$  和标签  $t$ , 如果  $\mathcal{F}.\text{Verify}(f, x, t) = 1$  成立, 则输出  $k \leftarrow h(x)$ , 否则输出  $\perp$ .



### 定理 5.12

如果  $\mathcal{F}$  是  $\mathcal{H}$ -hinting  $\Phi$ -不可延展的, 那么上面构造的 AKDFs 是  $\Phi'$ -RKA 安全的, 其中  $\Phi' = \Phi \cup \text{cf}$ .



**证明** 令  $S_i$  表示事件“敌手在 Game<sub>i</sub> 中成功”. 以游戏序列的方式组织证明如下:

Game<sub>0</sub>: 该游戏是 AKDFs 的标准 RKA 安全性游戏. 挑战者  $\mathcal{CH}$  与敌手  $\mathcal{A}$  按如下方式执行游戏:

1. 初始化:  $\mathcal{CH}$  生成函数索引  $f \leftarrow \mathcal{F}.\text{Gen}(1^\kappa)$  并选择一个相应的 hardcore 函数  $h \leftarrow \mathcal{H}$ , 然后将  $pp = (f, h)$  发送给  $\mathcal{A}$ .
2. 挑战:  $\mathcal{CH}$  随机采样一个原始密钥  $x^* \xleftarrow{\text{R}} X$ , 计算  $t^* \leftarrow f(x^*)$ ,  $k_0^* \leftarrow h(x^*)$ , 选择  $k_1^* \xleftarrow{\text{R}} K$ ,  $\beta \xleftarrow{\text{R}} \{0, 1\}$ , 然后将  $(pp, t^*, k_\beta^*)$  作为挑战信息发送给  $\mathcal{A}$ .
3. 相关密钥派生询问: 当收到合法询问  $\langle \phi, t \rangle \neq \langle \text{id}, t^* \rangle$  时, 如果  $\mathcal{F}.\text{Verify}(f, \phi(x^*), t) = 1$  成立,  $\mathcal{CH}$  返回  $h(\phi(x^*))$ , 否则返回  $\perp$ . (注: 对于常量查询,  $\mathcal{CH}$  不需要利用  $x^*$  就可以进行回答.)
4. 猜测:  $\mathcal{A}$  输出一个猜测比特  $\beta'$ , 如果  $\beta' = \beta$  成立, 则  $\mathcal{A}$  成功.

根据 RKA 安全性的定义, 我们有:

$$\text{Adv}_{\mathcal{A}, \text{AKDF}}^{\text{rka}}(\kappa) = |\Pr[S_0] - 1/2|$$

Game<sub>1</sub>(拒绝所有非常量询问): 该游戏与 Game<sub>0</sub> 的唯一不同之处在于处理非常量询问的方式. 对于所有非常量询问  $\langle \phi, t \rangle$ , 其中  $\phi \in \Phi$ ,  $\mathcal{CH}$  直接返回  $\perp$ . 令  $E$  为事件存在  $\mathcal{A}$  的非常量询问  $\langle \phi, t \rangle$  满足条件  $\mathcal{F}.\text{Verify}(f, \phi(x^*), t) = 1$ . 根据 Game<sub>0</sub> 和 Game<sub>1</sub> 的定义, 如果事件  $E$  发生, 在 Game<sub>1</sub> 中,  $\mathcal{CH}$  直接返回  $\perp$ , 而在 Game<sub>0</sub> 中,  $\mathcal{CH}$  返回  $\phi(x^*)$ . 显而易见, 对于任意 PPT 敌手  $\mathcal{A}$ , 当事件  $E$  未发生时,  $\mathcal{A}$  在 Game<sub>0</sub> 和 Game<sub>1</sub> 中的视图是完全一样的. 根据差分引理, 我们有:

$$|\Pr[S_0] - \Pr[S_1]| \leq \Pr[E]$$

根据下面的引理, 当  $\mathcal{F}$  满足  $\mathcal{H}$ -hinting  $\Phi$ -不可延展性时, 事件  $E$  发生的概率是可忽略的.

### 断言 5.1

$$\Pr[E] \leq \text{poly}(\kappa) \cdot \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\Phi-\text{nm}}(\kappa).$$



**证明** 令  $\mathcal{B}$  是一个攻击  $\mathcal{F}$  的  $\mathcal{H}$ -hinting  $\Phi$ -不可延展性的敌手. 给定挑战信息  $(f, h, y^*, h(x^*))$ , 其中  $f \leftarrow \mathcal{F}.\text{Gen}(1^\kappa)$ ,  $h \leftarrow \mathcal{H}$ ,  $y^* \leftarrow f(x^*)$  并且  $x^* \xleftarrow{\text{R}} X$ ,  $\mathcal{B}$  按以下方式模拟  $\mathcal{A}$  在 Game<sub>1</sub> 中的挑战者: 令  $pp = (f, h)$ ,  $t^* = y^*$ ,  $k_0^* \leftarrow h(x^*)$ , 选择  $k_1^* \xleftarrow{\text{R}} K$ ,  $\beta \xleftarrow{\text{R}} \{0, 1\}$ , 然后将  $(pp, t^*, k_\beta^*)$  发送给  $\mathcal{A}$ . 这里,  $\mathcal{B}$  并不知道  $x^*$  的值. 然而, 这并不影响  $\mathcal{B}$  模拟回答  $\mathcal{A}$  的询问, 因为在 Game<sub>1</sub> 中, 对于所有非常量查询,  $\mathcal{B}$  直接返回  $\perp$ . 由此可知,  $\mathcal{B}$  能完美地模拟  $\mathcal{A}$  在 Game<sub>1</sub> 中的视图环境. 令  $L$  为  $\mathcal{A}$  的所有非常量查询列表. 由于  $\mathcal{A}$  是一个 PPT 敌手, 所以  $|L| \leq \text{poly}(\kappa)$ . 最后,

$\mathcal{B}$  从列表  $L$  中随机选择一组元素  $\langle \phi, t \rangle$  作为攻击  $\mathcal{F} \mathcal{H}$ -hinting  $\Phi$ -不可延展性的结果. 当事件  $E$  发生时,  $\mathcal{B}$  成功的概率至少是  $1/\text{poly}(\kappa)$ . 因此,  $\mathcal{B}$  成功的优势至少为  $\Pr[E]/\text{poly}(\kappa)$ . 如果  $\Pr[E]$  是不可忽略的, 那么  $\mathcal{B}$  的优势也是不可忽略的, 与  $\mathcal{F}$  的不可延展性矛盾. 特别地, 由于  $\mathcal{B}$  攻击  $\mathcal{F}$  不可延展性成功的概率不超过  $\text{Adv}_{\mathcal{A}, \mathcal{F}}^{\Phi-\text{nm}}(\kappa)$ , 所以  $\Pr[E]/\text{poly}(\kappa) \leq \text{Adv}_{\mathcal{A}, \mathcal{F}}^{\Phi-\text{nm}}(\kappa)$ . 引理得证!  $\square$

**Game<sub>2</sub>** ( $k_0^* \xleftarrow{\text{R}} K_1$ ): 该游戏与 Game<sub>1</sub> 的唯一不同之处在于  $\mathcal{CH}$  随机选择  $k_0^* \xleftarrow{\text{R}} K$ , 而不是通过  $k_0^* \leftarrow h(x^*)$  计算而来. 显然, 如果存在一个敌手  $\mathcal{A}$  在 Game<sub>1</sub> 和 Game<sub>2</sub> 中的视图存在差异  $\epsilon(\kappa)$ , 我们可以构造一个归约算法以至少  $\epsilon(\kappa)/2$  的优势攻破 hardcore 函数的伪随机性. 因此, 我们有:

$$|\Pr[S_1] - \Pr[S_2]| \leq \text{negl}(\kappa)$$

在 Game<sub>2</sub> 中,  $k_\beta$  的值与  $\beta$  完全独立. 故,  $\Pr[S_2] = 1/2$ .

综上, 定理得证!  $\square$

## 5.3 消息依赖密钥安全

在公钥加密方案的安全模型中, 加密的消息一般是指明文空间的某一概率分布选择的, 而与加密算法的私钥无关。然而在硬盘加密、匿名证书系统等应用环境下, 加密的消息与私钥有关甚至是私钥本身(如  $f(sk)$ , 其中  $f$  是一个从密钥空间到消息空间的函数.), 传统的安全模型并不能完全满足这类应用的需求。实际上, 早在 1984 年 Goldwasser 和 Micali [GM-JCSS-1984] 提出概率加密方案时, 已经指出当加密的消息与密钥相关时, 无法保证方案的语义安全性。Black 等人 [BRS2002] 将加密这种特别消息下的安全性称之为消息依赖密钥安全性 (Key-Dependent Message Security, 简称 KDM 安全性) 或者是 Camenisch 和 Lysyanskaya [CL-EUROCRYPT-2001] 针对多用户环境下的循环加密安全性 (Circular Security)。KDM 安全性不仅能够解决实际应用中面临的安全问题, 而且还可以用于设计 CCA 安全的公钥加密方案和单向陷门函数 [KMT-JOC-2022]。

在 KDM 安全性中, 不同的应用环境要求被加密的私钥函数  $f$  有所不同。一般情况下, 简单的仿射函数即可满足需求。然而, 即使在这种情况下, 设计 KDM 安全的公钥加密方案也是相当困难的。2008 年, Boneh 等人 [BHHO2008] 利用私钥的密文公开可计算性的思想, 提出了第一个标准模型下基于 DDH 困难问题的循环加密方案。后来, 学者们基于类似思想提出了不同计算假设下的 KDM 安全的加密方案 [ACPS2009; BG2010; MTY2011-EUROCRYPT]。2016 年, Wee 将这些方案的设计思想统一为同态平滑投影哈希技术 [Wee-PKC-2016]。尽管这些方案具有 KDM 安全性, 但是仅能抵抗选择明文攻击。对于选择密文攻击, 由于私钥的密文公开可计算性与解密服务之间是相矛盾的, 因此设计抗选择密文攻击的 KMD 安全加密方案更具有挑战性。一种似乎万能效率不太高的方式就是利用从 CPA 到 CCA 转化的“Naor-Yung 双加密”模式 [CCS2009]。另一种方式就寻找特殊的密码工具实现 KDM-CCA 安全性, 如有损代数过滤器 (Lossy Algebraic Filter) [Hofheinz2013-EUROCRYPT]、辅助输入安全的认证加密 (Authenticated Encryption with Auxiliary-Input, 简称 AIAE) [HLL-ASIACRYPT-2016] 等。

本节内容主要介绍消息依赖密钥安全性的模型, 基于同态平滑投影哈希技术的 KDM-CPA 安全设计方法, 基于辅助输入安全认证加密技术的 KDM-CCA 安全设计方法, 以及具有 KDM-CCA 安全的 Cramer-Shoup 加密方案。

### 5.3.1 安全模型

消息依赖密钥加密可以看作是一种特殊的密钥泄漏函数。在消息依赖密钥安全模型中, 存在一个与密钥相关的函数集合  $\mathcal{F}$  将(一组)密钥映射到消息空间。与密钥泄漏安全模型不同, 消息依赖密钥加密泄漏的不是该密钥的函数值而是它的密文, 比如  $\text{Encrypt}(pk, f(sk))$ 。

在给出消息依赖密钥安全模型的形式化定义之前, 先介绍一个简单的反例说明并不是所有语义安全的加密方案都是 KDM 安全的 [CGH2012]。假设  $\text{PKE} = (\text{Setup}, \text{KeyGen}, \text{Encrypt}, \text{Decrypt})$  是任意一个语义安全的加密方案。在该方案的基础上构造一个新的加密方案  $\text{PKE}' = (\text{Setup}', \text{KeyGen}', \text{Encrypt}', \text{Decrypt}')$ , 其中  $\text{Setup}'$  和  $\text{KeyGen}'$  与原方案一样,  $\text{Encrypt}'$  和  $\text{Decrypt}'$  的定义如下:

$$\text{Encrypt}'(M) = \begin{cases} \text{Encrypt}(M)||0 & \text{if } M \neq sk \\ \text{Encrypt}(M)||1 & \text{if } M = sk \end{cases} \quad \text{Decrypt}'(C||b) = \begin{cases} \text{Decrypt}(C) & \text{if } b = 0 \\ sk & \text{if } b = 1 \end{cases}.$$

显然, 在语义安全性模型中, 消息是从明文空间中公开选取的, 等于私钥  $sk$  的概率是可忽略的, 密文的形式永远是  $\text{Encrypt}(M)||0$ , 所以  $\text{PKE}'$  仍然是语义安全的。但是在消息依赖密钥加密模型中, 消息等于或不等于  $sk$  时的密文形式是可以直接区分的, 所以  $\text{PKE}'$  不具有消息依赖密钥安全性。

#### 定义 5.13 (KDM-CCA 安全性)

对于任意  $n \in \mathbb{N}$ , 令  $\mathcal{F} = \{f : \mathcal{SK}^n \rightarrow \mathcal{M}\}$  是一个从  $n$  维密钥空间到消息空间的 KDM 函数族。定义公钥

加密方案 PKE 的 KDM-CCA 敌手  $\mathcal{A}$  的优势函数如下:

$$\text{Adv}_{\mathcal{A}, \mathcal{F}, \text{PKE}}^{\text{KDM-CCA}}(\kappa) = \left| \Pr \left[ \begin{array}{l} \beta' = \beta : \\ \quad pp \leftarrow \text{Setup}(1^\kappa); \\ \quad (pk_i, sk_i) \leftarrow \text{KeyGen}(pp), \forall i \in [n]; \\ \quad \text{Set } \text{CL} = \overrightarrow{pk} = (pk_1, \dots, pk_n), \overrightarrow{sk} = (sk_1, \dots, sk_n); \\ \quad \beta \xleftarrow{\text{R}} \{0, 1\}; \\ \quad \beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Enc}}(\cdot), \mathcal{O}_{\text{Dec}}(\cdot)}(pp, \overrightarrow{pk}); \end{array} \right] - \frac{1}{2} \right|,$$

其中, 加密谕言机和解密谕言机的定义如下:

- 加密谕言机  $\mathcal{O}_{\text{Enc}}(\cdot)$ : 输入  $(i, f)$ , 其中  $i \in [n]$ ,  $f \in \mathcal{F}$ , 如果  $\beta = 0$ , 返回  $C = \text{Encrypt}(pk_i, f(\overrightarrow{sk}))$ ; 如果  $\beta = 1$ , 返回  $C = \text{Encrypt}(pk_i, 0^{|\mathcal{M}|})$ . 最后, 将  $(i, C)$  添加至密文列表 CL 中.
- 解密谕言机  $\mathcal{O}_{\text{Dec}}(\cdot)$ : 输入  $(i, C)$ , 其中  $i \in [n]$ . 如果  $(i, C) \in \text{CL}$ , 返回  $\perp$ ; 否则, 返回  $\text{Dncrypt}(sk_i, C)$ .

上述定义中, 如果对于任意的 PPT 敌手  $\mathcal{A}$ , 优势函数  $\text{Adv}_{\mathcal{A}, \Phi, \text{PKE}}^{\text{KDM-CCA}}(\kappa)$  是可忽略的, 则称公钥加密方案 PKE 是  $\mathcal{F}$ -KDM-CCA 安全的.



**笔记** KDM-CCA 安全模型说明了敌手在解密服务的帮助下, 也无法区分一组密文是加密的私钥相关函数值还是某一固定消息, 例如  $0^{|\mathcal{M}|}$ . 若不允许敌手进行解密询问, 则上述模型即是 KDM 或 KDM-CPA 全性的定义. 不同类型的函数族  $\mathcal{F}$  对于实现 KDM 安全性的难度是不同的. 若  $\mathcal{F}$  是常数函数族  $\{f_M : \overrightarrow{sk} \rightarrow M\}_{M \in \mathcal{M}}$ , 则 KDM-CPA 安全性等价于传统的语义安全性 (IND-CPA). 而 KDM-CCA 安全性即是传统的 IND-CCA 安全性. 若  $\mathcal{F}$  是选择函数族  $\{f_i : \overrightarrow{sk} \rightarrow sk_i\}$ , 此时的 KDM 安全性也称之为循环加密安全性 (Circular Security).

## 5.3.2 Wee16 方案

### 5.3.2.1 同态平滑仿射哈希

平滑仿射哈希 (Smooth Projective Hashing, 简称 SPH) 是 Cramer 和 Shoup 在 2002 年提出的一种哈希证明系统 [CS-EUROCRYPT-2002]. 令  $\text{HPS} = (\text{Setup}, \text{KeyGen}, \text{PrivEval}, \text{PubEval})$  是一个哈希证明系统. 运行  $\text{Setup}(1^\kappa)$  输出一组公开参数  $pp = (\mathsf{H}, SK, PK, X, L, W, \Pi, \alpha)$ , 其中  $X$  定义了一个元素集合,  $L \subset X$  表示合法元素集合,  $X \setminus L$  表示非法元素集合, 合法元素与非法元素在计算上是不可区分的. 运行  $\text{KeyGen}(pp)$  将输出一对密钥  $(pk, sk)$ , 其中  $sk \xleftarrow{\text{R}} SK$ ,  $pk = \alpha(sk)$  也成为仿射密钥.

**同态平滑仿射哈希:** 是一组定义在  $\{\mathsf{H}_{sk} : X \rightarrow \Pi\}$  上的平滑仿射哈希. 类似平滑仿射哈希, 除了具有私有可计算、公开可计算、平滑性等性质外, 还需要具有同态性. 具体如下:

- 私有可计算性 (Private Evaluation): 对于任意  $x \in X$ , 存在算法  $\text{PrivEval}(sk, x)$ , 输出  $\pi = \mathsf{H}_{sk}(x)$ .
- 公开可计算性 (Public Evaluation): 对于任意  $x \in L$  以及相应的  $w$ , 存在算法  $\text{PubEval}(pk, x, w)$ , 输出  $\pi = \mathsf{H}_{sk}(x)$ .
- 平滑性 (Smoothness):  $\mathsf{H}_{sk}(\cdot)$  在输入  $x \xleftarrow{\text{R}} X$  时的输出与  $\Pi$  上的均匀分布统计接近, 即:

$$(pk, \mathsf{H}_{sk}(x)) \approx_s (pk, \pi)$$

其中  $(pk, sk) \leftarrow \text{KeyGen}(pp)$ ,  $\pi \xleftarrow{\text{R}} \Pi$ .

- 同态性 (Homomorphism): 对于所有  $sk \in SK$  和所有  $x_0, x_1 \in X$ , 我们有  $\mathsf{H}_{sk}(x_0) \cdot \mathsf{H}_{sk}(x_1) = \mathsf{H}_{sk}(x_0 \cdot x_1)$ .

### 5.3.2.2 基于同态平滑仿射哈希的 KDM-CPA 方案

#### 构造 5.12 (基于 HSPH 的 KDM-CPA 方案)

除了使用工具同态平滑投影哈希外, 我们还需要一个从消息空间  $\mathcal{M}$  到哈希值空间  $\Pi$  的一个公开可计算且可逆的映射  $\phi: \mathcal{M} \rightarrow \Pi$ .

- $\text{Setup}(1^\kappa)$ : 运行  $pp \leftarrow \text{HPS}.\text{Setup}(1^\kappa)$ , 输出系统参数  $pp = (\mathsf{H}, SK, PK, X, L, W, \Pi, \alpha)$ .
- $\text{KeyGen}(pp)$ : 运行  $(pk, sk) \leftarrow \text{HPS}.\text{KeyGen}(pp)$ , 输出公钥  $pk$  和私钥  $sk$ , 其中  $sk \xleftarrow{\$} SK$ ,  $pk = \alpha(sk)$ .
- $\text{Encrypt}(pk, M)$ : 以公钥  $pk$  和明文  $M \in \mathcal{M}$  为输入, 执行如下步骤:
  1. 运行  $(x, w) \leftarrow \text{SampR}(r)$  生成随机实例  $x \in L$  及相应的证据  $w$ , 其中  $r$  为采用时使用的随机数;
  2. 通过  $\text{HPS}.\text{PubEval}(pk, x, w)$  计算实例  $x$  的哈希证明  $\pi = \mathsf{H}_{sk}(x)$ ;
  3. 计算  $\psi = \pi \cdot \phi(M)$ ;
  4. 输出密文  $C = (x, \psi)$ .
- $\text{Decrypt}(sk, C)$ : 以私钥  $sk$  和密文  $C = (x, \psi)$  为输入, 计算  $M' = \phi^{-1}(\mathsf{H}_{sk}(x)^{-1} \cdot \psi)$  并返回明文  $M'$ .



**正确性.** 方案的正确性可由仿射哈希的完备性保证. 安全性由如下定理保证.

#### 定理 5.13

假设  $\mathsf{H}_{sk}(\cdot)$  是一个定义在  $X$  上的仿射哈希函数, 满足平滑性和同态性, 并且  $L \subseteq X$  上的 SMP 困难问题成立, 那么构造 ?? 中的 PKE 方案是  $\mathcal{F}$ -KDM-CPA 安全的, 其中  $\mathcal{F} = \{f_{e,k}: sk \rightarrow \phi^{-1}(\mathsf{H}_{sk}(e) \cdot k) \mid e \in X, k \in \Pi\}$ .



定理 ?? 的证明思路主要是将密钥函数值  $f_{x,\pi}(sk)$  的密文转化为函数参数  $(x, \pi)$  的密文, 由此使得 KDM 密文与私钥  $sk$  无关. 转化的技术是仿射哈希的同态性. 即, 将  $f_{x,\pi}(sk)$  的密文:

$$\text{Encrypt}(pk, f_{e,k}(sk)) = (x, \mathsf{H}_{sk}(x) \cdot f_{e,k}(sk))$$

转化为

$$\text{Encrypt}(pk, f_{e,k}(sk)) = (x \cdot e^{-1}, \text{HPS}.\text{PubEval}(pk, x, w) \cdot k). \quad (5.21)$$

从而使得挑战者在不知道私钥  $sk$  的情况下, 也可以回答敌手的 KDM 加密询问. 下面给出详细的证明过程.

**证明** 令  $S_i$  表示敌手在 Game<sub>i</sub> 中的成功事件. 以游戏序列的方式组织证明如下:

Game<sub>0</sub>: 该游戏是标准的 KDM-CPA 游戏, 挑战者  $\mathcal{CH}$  和敌手  $\mathcal{A}$  交互如下:

- 初始化:  $\mathcal{CH}$  运行  $\text{Setup}(1^\kappa)$  生成公开参数  $pp$ , 同时运行  $\text{KeyGen}(pp)$  生成公私钥对  $(pk, sk)$ .  $\mathcal{CH}$  将  $(pp, pk)$  发送给  $\mathcal{A}$ .
- 挑战:  $\mathcal{CH}$  选择随机比特  $\beta \in \{0, 1\}$ .
- 询问: 对于敌手的任意询问  $f_{e,k} \in \mathcal{F}$ ,  $\mathcal{CH}$  作如下计算:
  1. 如果  $\beta = 0$ ,  $\mathcal{CH}$  随机选择  $x \in L$  及相应的证据  $w$ , 计算密文  $C = (x, \psi)$ , 其中

$$\psi = \text{HPS}.\text{PubEval}(pk, x, w) \cdot \phi(f_{e,k}(sk)) = \text{HPS}.\text{PubEval}(pk, x, w) \cdot \mathsf{H}_{sk}(e) \cdot k.$$

2. 如果  $\beta = 1$ ,  $\mathcal{CH}$  随机选择  $x \in L$  及相应的证据  $w$ , 计算密文  $C = (x, \psi)$ , 其中

$$\psi = \text{HPS}.\text{PubEval}(pk, x, w) \cdot \phi(0^{|M|}).$$

3. 最后,  $\mathcal{CH}$  将密文  $C = (x, \psi)$  返回给敌手.

- 猜测:  $\mathcal{A}$  输出对  $\beta$  的猜测  $\beta'$ .  $\mathcal{A}$  成功当且仅当  $\beta' = \beta$ .

根据定义, 我们有:

$$\text{Adv}_{\mathcal{A}, \mathcal{F}, \text{PKE}}^{\text{KDM-CPA}}(\kappa) = |\Pr[S_0] - 1/2|$$

**Game<sub>1</sub>**: 该游戏与 Game<sub>0</sub> 的唯一不同在于  $\beta = 0$  时加密预言机的工作方式. 具体地, 对于敌手的任意加密询问  $f \in \mathcal{F}$ , 当  $\beta = 0$  时,  $\mathcal{CH}$  返回形如公式 ?? 中的密文.

假设  $\mathcal{A}$  询问加密预言机的次数最多为  $Q$  次, 我们可以利用混合游戏的思想在 Game<sub>0</sub> 和 Game<sub>1</sub> 之间定义  $Q - 1$  个混合游戏 Game<sub>0,i</sub>, 其中  $i \in \{1, \dots, Q - 1\}$ . 在 Game<sub>0,i</sub> 中, 当  $\beta = 0$  时,  $\mathcal{A}$  的前  $i$  个询问的密文是公式 ?? 中的形式, 而后  $Q - i$  次询问的密文按正常方式加密得来. 显然, Game<sub>0,0</sub> = Game<sub>0</sub>, Game<sub>0,Q</sub> = Game<sub>1</sub>. 对于任意的  $i$ , 敌手在两个连续游戏中的视图是不可区分的, 即 Game<sub>0,i-1</sub> ≈ Game<sub>0,i</sub>. 这是因为

$$\begin{aligned} & \text{Encrypt}(pk, f_{e,k}(sk)) \\ = & (x, \text{HPS.PubEval}(pk, x, w) \cdot \mathsf{H}_{sk}(e) \cdot k) : (x, w) \leftarrow \text{SampR}(r) \\ = & (x, \mathsf{H}_{sk}(x) \cdot \mathsf{H}_{sk}(e) \cdot k) : \text{仿射性质} \\ \approx_c & (x, \mathsf{H}_{sk}(x) \cdot \mathsf{H}_{sk}(e) \cdot k) : x \xleftarrow{R} X, \text{SMP 问题} \\ = & (x, \mathsf{H}_{sk}(x \cdot e) \cdot k) : x \xleftarrow{R} X, \text{同态性质} \\ = & (x \cdot e^{-1}, \mathsf{H}_{sk}(x) \cdot k) : x \xleftarrow{R} X \\ \approx_c & (x \cdot e^{-1}, \mathsf{H}_{sk}(x) \cdot k) : (x, w) \leftarrow \text{SampR}(r) \\ = & (x \cdot e^{-1}, \text{HPS.PubEval}(pk, x, w) \cdot k) : (x, w) \leftarrow \text{SampR}(r), \text{仿射性质} \end{aligned}$$

特别注意, 在上式的演进过程中, 密钥  $s$  是完全公开的. 因此, 在前  $i$  次询问时,  $\mathcal{CH}$  可以用公钥和 KDM 函数  $f_{e,k}$  计算密文  $(x \cdot e^{-1}, \text{HPS.PubEval}(pk, x, w) \cdot k)$ , 而对于后  $Q - i$  次询问,  $\mathcal{CH}$  可以用私钥  $sk$  和 KDM 函数  $f_{e,k}$  计算密文  $\text{Encrypt}(pk, f_{e,k}(sk))$ .

由此可知,

$$|\Pr[S_1] - \Pr[S_2]| \leq \text{negl}(\kappa).$$

**Game<sub>2</sub>**: 该游戏与 Game<sub>1</sub> 的唯一不同在于  $\beta = 0$  时加密预言机的工作方式. 具体地, 对于敌手的任意加密询问  $f \in \mathcal{F}$ , 当  $\beta = 0$  时,  $\mathcal{CH}$  返回一个随机密文  $(x, \psi)$ , 其中  $x \xleftarrow{R} X, \psi \xleftarrow{R} \Pi$ . 在 Game<sub>1</sub> 中, 由于 KDM 密文可以由公钥  $pk$  和 KDM 函数的参数  $(e, k)$  公开计算, 因此我们只需要证明

$$(x \cdot e^{-1}, \text{HPS.PubEval}(pk, x, w) \cdot k) \approx (x, \psi)$$

其中  $(x, w) \leftarrow \text{SampR}(r), x \xleftarrow{R} X, \psi \xleftarrow{R} \Pi$ . 这是因为

$$\begin{aligned} & (x \cdot e^{-1}, \text{HPS.PubEval}(pk, x, w) \cdot k) : (x, w) \leftarrow \text{SampR}(r) \\ \approx_c & (x \cdot e^{-1}, \mathsf{H}_{sk}(x) \cdot k) : (x, w) \leftarrow \text{SampR}(r), \text{仿射性质} \\ \approx_c & (x \cdot e^{-1}, \mathsf{H}_{sk}(x) \cdot k) : x \xleftarrow{R} X, \text{SMP 性质} \\ \approx_s & (x \cdot e^{-1}, \pi \cdot k) : x \xleftarrow{R} X, \pi \xleftarrow{R} \Pi, \text{平滑性} \\ = & (x, \psi) : x \xleftarrow{R} X, \psi \xleftarrow{R} \Pi \end{aligned}$$

由此可知, 在游戏 Game<sub>2</sub> 中, 当  $\beta = 0$ , 加密预言机返回的都是随机密文, 与 KDM 函数值  $f_{e,k}(sk)$  无关.

**Game<sub>3</sub>**: 该游戏与 Game<sub>2</sub> 的唯一不同在于  $\beta = 1$  时加密预言机的工作方式. 具体地, 对于敌手的任意加密询问  $f \in \mathcal{F}$ , 当  $\beta = 1$  时,  $\mathcal{CH}$  返回一个随机密文  $(x, \psi)$ , 其中  $x \xleftarrow{R} X, \psi \xleftarrow{R} \Pi$ . 我们知道, 当  $\beta = 1$  时, 加密预言机返回的密文形式是  $\text{Encrypt}(pk, 0^{|\mathcal{M}|})$ . 利用仿射哈希的平滑性, 可以直接证明消息  $0^{|\mathcal{M}|}$  的密文与一个随机密文是不可区分的, 即

$$|\Pr[S_2] - \Pr[S_3]| \leq \text{negl}(\kappa).$$

在 Game<sub>3</sub> 中, 不管  $\beta = 0$  还是  $\beta = 1$ , 加密预言机返回的都是一个随机密文, 与挑战比特  $\beta$  完全无关. 由此,

我们可得

$$\Pr[S_3] = 1/2.$$

综上, 定理得证! □

### 5.3.3 HLL16 方案

2015 年, Lu、Li 和 Jia [LLJ2015] 构造了第一个高效的、密文紧凑的 KDM[ $\mathcal{F}_{\text{aff}}$ ]-CCA 安全公钥加密方案, 其密文甚至只包括 7 个群元素. 该方案的安全性同时基于 DDH 和 DCR 这两个假设, 所支持的私钥函数集合为仿射函数集合  $\mathcal{F}_{\text{aff}}$ . 该方案使用了一个很重要的工具, 即 RKA 安全的认证加密方案  $\overline{\text{AE}}$ . 用在 RKA 攻击下同时具有不可区分性 (IND- $\mathcal{F}$ -RKA) 和完整性 (INT- $\mathcal{F}$ -RKA) 的认证加密方案, 构造出了 KDM-CCA 安全的公钥加密方案. LLJ 方案的 KDM[ $\mathcal{F}_{\text{aff}}$ ]-CCA 安全性十分依赖于组件  $\overline{\text{AE}}$  的 INT- $\mathcal{F}_{\text{aff}}$ -RKA 安全性 (参见文献 [LLJ2015] 定义 2). 然而 LLJ 方案中使用的认证加密方案在安全性上存在一定的缺陷, 从而导致 LLJ 方案的 KDM[ $\mathcal{F}_{\text{aff}}$ ]-CCA 安全性证明的失败.

2016 年, Han、Liu 和 Lyu [HLL-ASIACRYPT-2016] 针对仿射函数集合和多项式函数集合, 分别构造高效的、密文紧凑的 KDM[ $\mathcal{F}_{\text{aff}}$ ]-CCA 安全公钥加密方案以及高效的 KDM[ $\mathcal{F}_{\text{poly}}^d$ ]-CCA 安全公钥加密方案. 该方案提出一个全新的密码原语, 即支持辅助输入的认证加密方案 AIAE, 并为其定义了弱 RKA 安全性. 该新原语成为了实现 KDM-CCA 安全公钥加密方案的必要工具.

下面主要介绍 AIAE 的概念、基于 AIAE 的 KDM-CCA 安全公钥加密方案的通用设计方法、支持仿射函数集的 KDM-CCA 安全公钥加密方案和支持多项式函数集的 KDM-CCA 安全公钥加密方案.

#### 5.3.3.1 支持辅助输入的认证加密方案

下面介绍支持辅助输入的认证加密方案 AIAE 的形式化定义及其弱 RKA 安全性. 基于传统的认证加密方案和具有密钥同态性质的带标签哈希证明系统, 给出 AIAE 的通用构造, 并基于 DDH 假设给出 AIAE 的具体实例.

##### 定义 5.14 (支持辅助输入的认证加密方案)

一个支持辅助输入的认证加密方案  $\text{AIAE} = (\text{Setup}, \text{Encrypt}, \text{Decrypt})$  由三个 PPT 算法组成:

- $\text{Setup}(1^\kappa)$ : 输入安全参数  $\kappa$ , 输出一个公开参数  $pp_{\text{AIAE}}$ . 公开参数  $pp_{\text{AIAE}}$  隐含地定义了一个密钥空间  $\mathcal{K}_{\text{AIAE}}$ 、一个消息空间  $\mathcal{M}$  和一个辅助输入空间  $\mathcal{Aux}$ . 为了方便起见, 公开参数  $pp_{\text{AIAE}}$  会默认作为所有算法的一个输入.
- $\text{Encrypt}(k, m, aux)$ : 以密钥  $k \in \mathcal{K}_{\text{AIAE}}$ 、消息  $m \in \mathcal{M}$  和辅助输入  $aux \in \mathcal{Aux}$  为输入, 输出一个密文  $C$ .
- $\text{Decrypt}(k, C, aux)$ : 以密钥  $k \in \mathcal{K}_{\text{AIAE}}$ 、密文  $C$  和辅助输入  $aux \in \mathcal{Aux}$  为输入, 输出一个消息  $m \in \mathcal{M}$  或者一个代表解密失败的符号  $\perp$ .



**正确性.** 对所有  $pp_{\text{AIAE}} \leftarrow \text{Setup}(1^\kappa)$ 、所有密钥  $k \in \mathcal{K}_{\text{AIAE}}$ 、所有消息  $m \in \mathcal{M}$  和所有辅助输入  $aux \in \mathcal{Aux}$ , 满足  $\text{Decrypt}(k, \text{Encrypt}(k, m, aux), aux) = m$ .



**笔记** 认证加密方案是一种私钥加密方案. 辅助输入的认证加密方案可以看作是一种辅助输入的私钥加密方案, 也可以看成是传统认证加密方案的一个扩展. 因为对于所有的系统参数  $pp_{\text{AIAE}}$ , 如果辅助输入空间  $\mathcal{Aux}$  都为空集, 则上述定义就退化为传统的认证加密方案.

令  $\mathcal{F}$  为一族从  $\mathcal{K}_{\text{AIAE}}$  到  $\mathcal{K}_{\text{AIAE}}$  的函数. 我们通过图??中的实验来定义 AIAE 的弱 RKA 安全性. 在 weak-INT- $\mathcal{F}$ -RKA 实验的 FINALIZE 程序中, 有一个用灰色背景标注的特殊规则.

##### 定义 5.15 (AIAE 的弱 $\mathcal{F}$ -RKA 安全性)

一个支持辅助输入的认证加密方案 AIAE 是弱  $\mathcal{F}$ -RKA 安全的, 如果它是 IND- $\mathcal{F}$ -RKA 安全和弱 INT- $\mathcal{F}$ -

RKA 安全的, 即对于任意 PPT 敌手  $\mathcal{A}$ , 满足条件

$$\text{Adv}_{\text{AIAE}, \mathcal{F}, \mathcal{A}}^{\text{ind-rka}}(\kappa) := |\Pr[\text{IND-}\mathcal{F}\text{-RKA}^{\mathcal{A}} \Rightarrow 1] - 1/2| \leq \text{negl}(\kappa),$$

$$\text{Adv}_{\text{AIAE}, \mathcal{F}, \mathcal{A}}^{\text{weak-int-rka}}(\kappa) := \Pr[\text{weak-INT-}\mathcal{F}\text{-RKA}^{\mathcal{A}} \Rightarrow 1] \leq \text{negl}(\kappa),$$

其中 IND- $\mathcal{F}$ -RKA 和 weak-INT- $\mathcal{F}$ -RKA 是图??中描述的实验.



<b>程序 INITIALIZE:</b> $pp_{\text{AIAE}} \leftarrow \text{AIAE}.\text{Setup}(1^\kappa).$ $k \xleftarrow{R} \mathcal{K}_{\text{AIAE}}.$ $\beta \xleftarrow{R} \{0, 1\}.$ // 挑战比特 Return $pp_{\text{AIAE}}.$  <b>程序 ENC(<math>f \in \mathcal{F}, m_0, m_1, aux</math>):</b> 如果 $ m_0  \neq  m_1 $ , Return $\perp$ . $aiae.ct \leftarrow \text{AIAE}.\text{Encrypt}(f(k), m_\beta, aux).$ Return $aiae.ct$ .  <b>程序 FINALIZE(<math>\beta'</math>):</b> Return $(\beta' = \beta).$	<b>程序 INITIALIZE:</b> $pp_{\text{AIAE}} \leftarrow \text{AIAE}.\text{Setup}(1^\kappa)$ , $k \xleftarrow{R} \mathcal{K}_{\text{AIAE}}.$ Return $pp_{\text{AIAE}}.$  <b>程序 ENC(<math>f \in \mathcal{F}, m, aux</math>):</b> $aiae.ct \leftarrow \text{AIAE}.\text{Encrypt}(f(k), m, aux).$ $\mathcal{Q}_{\text{ENC}} := \mathcal{Q}_{\text{ENC}} \cup \{(f, aiae.ct, aux)\}.$ $\mathcal{Q}_{\text{AUXF}} := \mathcal{Q}_{\text{AUXF}} \cup \{(f, aux)\}.$ Return $aiae.ct$ .  <b>程序 FINALIZE(<math>f^* \in \mathcal{F}, aiae.ct^*, aux^*</math>):</b> 如果 $(f^*, aiae.ct^*, aux^*) \in \mathcal{Q}_{\text{ENC}}$ , Return 0. 如果存在 $(f, aux) \in \mathcal{Q}_{\text{AUXF}}$ 使得 $aux = aux^*$ 但 $f \neq f^*$ , Return 0. // 特殊规则 Return $(\text{AIAE}.\text{Decrypt}(f^*(k), aiae.ct^*, aux^*) \neq \perp).$
--	---

图 5.16: 支持辅助输入认证加密方案 AIAE 的 IND- $\mathcal{F}$ -RKA 实验(左图)及 weak-INT- $\mathcal{F}$ -RKA 实验(右图)



**笔记** 辅助输入的认证加密方案的弱 RKA 安全性实际上包含两个方面: 一是敌手在任意篡改密钥  $f(k)$  下两个长度相同消息的密文依然不可区分; 二是敌手在获得若干篡改密钥和若干消息的密文, 依然无法生成一个新消息或新辅助元素的合法密文.

**AIAE 方案的通用构造:** 基于标准的认证加密方案和密钥同态的带标签哈希证明系统可以构造支持辅助输入认证加密方案 AIAE. 使用的组件具体包括:

- 一个传统的(即不支持辅助输入的)认证加密方案, 记作  $\text{AE} = (\text{Setup}, \text{Encrypt}, \text{Decrypt})$ , 其密钥空间为  $\mathcal{K}$ , 消息空间为  $\mathcal{M}$ ;
- 一个具有密钥同态性质的带标签哈希证明系统, 记作  $\text{THPS} = (\text{Setup}, \text{KeyGen}, \text{PrivEval}, \text{PubEval})$ , 其哈希值空间为  $\mathcal{K}$ 、标签空间为  $\mathcal{T}$ ;
- 一个抗碰撞的哈希函数族, 记作  $\mathcal{TCR} = \{\text{TCR} : \{0, 1\}^* \rightarrow \mathcal{T}\}$ .

在上述组件中, 要求 THPS 的哈希值空间与 AE 的密钥空间相匹配, 而 THPS 的标签空间与  $\mathcal{H}$  的值域相匹配.

Han 等人设计的支持辅助输入认证加密方案  $\text{AIAE} = (\text{Setup}, \text{Encrypt}, \text{Decrypt})$  见图??, 其密钥空间为  $\mathcal{K}_{\text{AIAE}} := \mathcal{S}\mathcal{K}$ 、消息空间为  $\mathcal{M}$ 、辅助输入空间为  $\mathcal{Aux} = \{0, 1\}^*$ .

<b>AIAE.Setup(<math>1^\kappa</math>):</b> $pp_{\text{THPS}} \leftarrow \text{THPS}.\text{Setup}(1^\kappa).$ $pp_{\text{AE}} \leftarrow \text{AE}.\text{Setup}(1^\kappa).$ $\text{TCR} \leftarrow \mathcal{TCR}.$ $pp_{\text{AIAE}} := (pp_{\text{THPS}}, pp_{\text{AE}}, \text{TCR}).$ Return $pp_{\text{AIAE}}.$	<b>AIAE.Encrypt(<math>sk, m, aux</math>):</b> $x \xleftarrow{R} L$ 及证据 $w$ . $t \leftarrow \text{TCR}(x, aux) \in \mathcal{T}.$ $k \leftarrow \text{H}_{sk}(x, t) \in \mathcal{K}.$ $\chi \leftarrow \text{AE}.\text{Encrypt}(k, m).$ Return $\langle x, \chi \rangle.$	<b>AIAE.Decrypt(<math>sk, \langle x, \chi \rangle, aux</math>):</b> 如果 $x \notin X$ , Return $\perp$ . $t := \text{TCR}(x, aux) \in \mathcal{T}.$ $k := \text{H}_{sk}(x, t) \in \mathcal{K}.$ $m/\perp \leftarrow \text{AE}.\text{Decrypt}(k, \chi).$ Return $m/\perp.$
--	--	--

图 5.17: 支持辅助输入认证加密方案 AIAE 的通用构造

根据 AE 的正确性, 可以直接验证 AIAE 的正确性. 方案的弱  $\mathcal{F}_{\text{raff}}$ -RKA 安全性由定理 ??保证, 其中  $\mathcal{F}_{\text{raff}}$  是受限仿射函数集合, 定义为:  $\mathcal{F}_{\text{raff}} := \{f_{(a,b)} : \text{sk} \mapsto a \cdot \text{sk} + b \in \mathcal{S}\mathcal{K} \mid a \in \mathbb{Z}_{|\mathcal{K}|}^*, b, f \parallel \in \mathcal{S}\mathcal{K}\}.$

**定理 5.14**

如果 (i)AE 是 OT 安全的; (ii)THPS 是 2-universal、密钥同态的且 SMP 问题困难; (iii)TCR 是抗碰撞的; 则图??中构造的 AIAE 方案是弱  $\mathcal{F}_{\text{raff}}\text{-RKA}$  安全的.



上述定理的证明请参考博士论文 [Han2019] 定理 3.1 的证明.

**AIAE 的实例化方案.** 2015 年, Qin、Liu 和 Chen [QLC-IET-IFS-2015] 基于  $d$ -Linear 假设构造了一个带标签哈希证明系统. 2019 年, Han 将该哈希证明系统进一步扩展为具有密钥同态性质的带标签哈希证明系统 THPS, 并基于 DDH 困难假设证明了 THPS 的相关性质, 如图 ?? 所示. 将该哈希证明系统带入到上述通用构造中, 即可以得到一个基于 DDH 问题的支持辅助输入认证加密方案.

$pp_{\text{THPS}} \leftarrow \text{THPS}.\text{Setup}(1^\kappa)$ : $(N, p, q) \leftarrow \text{GenN}(1^\kappa)$ , 即选取两个 $\kappa$ 比特的安全素数 $p$ 和 $q$ 使得 $2pq + 1$ 也是素数且 $N := pq$ . $\bar{N} := 2N + 1$ . $g_1, g_2 \leftarrow \mathbb{QR}_{\bar{N}}$ . Return $pp_{\text{THPS}} := (N, p, q, \bar{N}, g_1, g_2)$ , 其隐含地定义了 $(\mathcal{H}, \mathcal{SK}, \mathcal{PK}, \mathcal{T}, X, L, W, \Pi, \alpha)$ : $\Pi := \mathbb{QR}_{\bar{N}}$ . $X := \mathbb{QR}_{\bar{N}}^2 \setminus \{(1, 1)\}$ . $L := \{(g_1^w, g_2^w) \mid w \in \mathbb{Z}_N \setminus \{0\}\}$ . $\mathcal{T} := \mathbb{Z}_N$ . $\mathcal{SK} := (\mathbb{Z}_N)^4$ . $\mathcal{PK} := \mathbb{QR}_{\bar{N}}^2$ . 对于 $\text{sk} = (k_1, k_2, k_3, k_4) \in \mathcal{SK}$ , $x = (x_1, x_2) \in X$ , $t \in \mathcal{T}$ , $\mathcal{H}_{\text{sk}}(x, t) := x_1^{k_1+k_3t} x_2^{k_2+k_4t} \in \Pi$ . 对于 $\text{sk} = (k_1, k_2, k_3, k_4) \in \mathcal{SK}$ , $\text{pk} = \alpha(\text{sk}) := (g_1^{k_1} g_2^{k_2}, g_1^{k_3} g_2^{k_4}) \in \mathcal{PK}$ .		
$K \leftarrow \text{THPS}.\text{PubEval}(\text{pk}, x, w, t)$ : $\text{pk} = (h_1, h_2) \in \mathcal{PK}$ . Return $K := h_1^w h_2^{wt}$ .	$K \leftarrow \text{THPS}.\text{PrivEval}(\text{sk}, x, t)$ : $\text{sk} = (k_1, k_2, k_3, k_4) \in \mathcal{SK}$ 和 $x = (x_1, x_2) \in X$ . Return $K := x_1^{k_1+k_3t} x_2^{k_2+k_4t}$ .	

图 5.18: 基于 DDH 假设的密钥同态带标签哈希证明系统  $\text{THPS}_{\text{DDH}}$

### 5.3.3.2 KDM-CCA 方案的通用设计方法

下面介绍一种构造 KDM-CCA 安全公钥加密方案的通用设计方法, 该方法主要使用以下三个组件: 一个密钥封装机制 KEM、一个公钥加密方案 PKE 以及一个支持辅助输入的认证加密方案 AIAE.

方案的通用设计方法见图 ??, 其中:

- KEM 和 PKE 使用相同的公钥和私钥  $(\text{pk}, \text{sk})$ .
- KEM.Enc 为 AIAE 方案封装一个(会话)密钥  $k$ , 同时密钥封装  $\text{kem.ct} = \text{aux}$  是 AIAE.Enc 的辅助输入 aux.
- PKE.Enc 加密消息  $m$  得到 PKE 方案的密文  $\text{pke.ct}$ , 同时  $\text{pke.ct}$  是 AIAE.Enc 所加密的消息.
- AIAE.Enc 使用 KEM 方案所封装的(会话)密钥  $k$  来加密  $\text{pke.ct}$ , 使用  $\text{aux} := \text{kem.ct}$  作为辅助输入, 得到 AIAE 方案的密文  $\text{aiae.ct}$ .

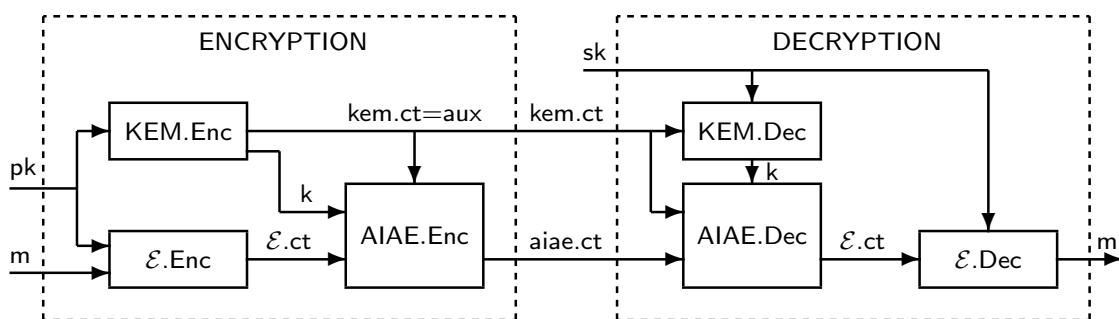


图 5.19: KDM-CCA 安全公钥加密方案的通用设计方法

在上述通用设计方法中, PKE.Encrypt 直接对消息  $m$  进行加密. 特别地, 在  $n$ -KDM[ $\mathcal{F}$ ]-CCA 的实验中, PKE.Encrypt 直接加密依赖密钥消息  $f(\text{sk}_1, \dots, \text{sk}_n)$ , 即  $\text{pke.ct} \leftarrow \text{PKE}.\text{Encrypt}(\text{pk}_j, f(\text{sk}_1, \dots, \text{sk}_n))$ , 其中  $f \in \mathcal{F}$ . 因此, 要求

PKE.Encrypt 算法可以通过计算不可区分的变换,使得 pke.ct 中不含有私钥  $(sk_1, \dots, sk_n)$  的特定信息,从而私钥的一部分熵得到了保留.这样的 PKE.Encrypt 称为针对函数集合  $\mathcal{F}$  的熵过滤器(Entropy Filter).

### 5.3.3.3 实例化方案

使用前一节中的通用设计方法,通过设计对应的组件 KEM 和组件 PKE,可以构造出支持仿射函数集合  $\mathcal{F}_{\text{aff}}$  和多项式函数集合  $\mathcal{F}_{\text{poly}}^d$  的 KDM-CCA 安全公钥加密方案。

**支持仿射函数集的 KDM-CCA 安全公钥加密方案.**令  $\text{AIAE}_{\text{DDH}} = (\text{AIAE}.\text{Setup}, \text{AIAE}.\text{Encrypt}, \text{AIAE}.\text{Decrypt})$  是前面构造的基于 DDH 假设的支持辅助输入认证加密方案,其密钥空间为  $(\mathbb{Z}_N)^4$ 、消息空间为  $\mathcal{M}$ .根据图??中描述的 KDM-CCA 安全公钥加密方案的通用设计方法,还需要构造另外两个组件:

- **KEM:** 针对具体的认证加密方案  $\text{AIAE}_{\text{DDH}}$ ,需要设计一个密钥封装机制 KEM,使得可以为  $\text{AIAE}_{\text{DDH}}$  方案封装一个密钥元组  $(k_1, k_2, k_3, k_4) \in (\mathbb{Z}_N)^4$ .
- **PKE:** 针对仿射函数集合  $\mathcal{F}_{\text{aff}}$ ,需要设计一个公钥加密方案 PKE,使得加密算法 PKE.Encrypt 可以通过计算不可区分的变换,转换成针对仿射函数的熵过滤器.

公钥加密方案  $\text{PKE} = (\text{Setup}, \text{KeyGen}, \text{Encrypt}, \text{Decrypt})$  的具体描述见图??,其中组件 KEM 和 PKE 的算法用灰色背景标注.在初始化算法中,  $pp_{\text{AIAE}}$  中所包含的因子  $p$  和  $q$  没有提供给  $pp'_{\text{AIAE}}$ ,因为  $p$  和  $q$  不会在方案  $\text{AIAE}_{\text{DDH}}$  的加解密算法中使用.

$pp \leftarrow \text{Setup}(1^\kappa):$ $pp_{\text{AIAE}} \leftarrow \text{AIAE}.\text{Setup}(1^\kappa)$ , 其中 $pp_{\text{AIAE}} = (N, p, q, \bar{N}, \bar{g}_1, \bar{g}_2, pp_{\text{AT}}, \text{TCR})$ , $N = pq$ , $\bar{N} = 2N + 1$ , $\bar{g}_1, \bar{g}_2 \in \mathbb{QR}_{\bar{N}}$ . $pp'_{\text{AIAE}} := (N, \bar{N}, \bar{g}_1, \bar{g}_2, pp_{\text{AE}}, \text{TCR})$ . $g_1, g_2, g_3, g_4, g_5 \leftarrow \mathbb{SCR}_{N^s}$ . Return $pp := (pp'_{\text{AIAE}}, g_1, g_2, g_3, g_4, g_5)$ .  $\langle \text{aux}, \text{aiae.ct} \rangle \leftarrow \text{Encrypt}(\text{pk}, m)$ : $m \in [N^{s-1}]$ // $(k, \text{aux}) \leftarrow \text{KEM}.\text{Encrypt}(\text{pk})$ : $k = (k_1, k_2, k_3, k_4) \xleftarrow{\text{R}} (\mathbb{Z}_N)^4$ . $r \leftarrow [\lfloor \frac{N}{4} \rfloor]$ . $(u_1, u_2, u_3, u_4, u_5) := (g_1^r, g_2^r, g_3^r, g_4^r, g_5^r)$ mod $N^2$ . $(e_1, e_2, e_3, e_4) := (h_1 T^{k_1}, h_2 T^{k_2}, h_3 T^{k_3},$ $h_4 T^{k_4})$ mod $N^2$ . $\text{aux} := (u_1, \dots, u_5, e_1, \dots, e_4)$ .  // $\text{pke.ct} \leftarrow \text{PKE}.\text{Enc}(\text{pk}, m)$ : $\tilde{r}_1, \tilde{r}_2, \tilde{r}_3, \tilde{r}_4 \xleftarrow{\text{R}} [\lfloor \frac{N}{4} \rfloor]$ . $(\tilde{u}_1, \tilde{u}_2, \tilde{u}_3, \tilde{u}_4, \tilde{u}_5, \tilde{u}_6, \tilde{u}_7, \tilde{u}_8) := (g_1^{\tilde{r}_1}, g_2^{\tilde{r}_1},$ $g_2^{\tilde{r}_2}, g_3^{\tilde{r}_3}, g_3^{\tilde{r}_3}, g_4^{\tilde{r}_4}, g_4^{\tilde{r}_4}, g_5^{\tilde{r}_4})$ mod $N^s$ . $\tilde{e} := h_1^{\tilde{r}_1} h_2^{\tilde{r}_2} h_3^{\tilde{r}_3} h_4^{\tilde{r}_4} T^m$ mod $N^s$ . $t := g_1^m$ mod $N \in \mathbb{Z}_N$ . $\text{pke.ct} := (\tilde{u}_1, \dots, \tilde{u}_8, \tilde{e}, t)$ . $\text{aiae.ct} \leftarrow \text{AIAE}.\text{Encrypt}(k, \text{pke.ct}, \text{aux})$ . Return $\langle \text{aux}, \text{aiae.ct} \rangle$ .	$(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(pp)$ : $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4 \xleftarrow{\text{R}} [\lfloor \frac{N^2}{4} \rfloor]$ . $(h_1, h_2, h_3, h_4) := (g_1^{-x_1} g_2^{-y_1}, g_2^{-x_2} g_3^{-y_2},$ $g_3^{-x_3} g_4^{-y_3}, g_4^{-x_4} g_5^{-y_4})$ mod $N^s$ . $\text{pk} := (h_1, h_2, h_3, h_4)$ . $\text{sk} := (x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$ . Return $(\text{pk}, \text{sk})$ .  $m/\perp \leftarrow \text{Decrypt}(\text{sk}, \langle \text{aux}, \text{aiae.ct} \rangle)$ : // $k/\perp \leftarrow \text{KEM}.\text{Decrypt}(\text{sk}, \text{aux})$ : Parse $\text{aux} = (u_1, \dots, u_5, e_1, \dots, e_4)$ . If $e_1 u_1^{x_1} u_2^{y_1}, e_2 u_2^{x_2} u_3^{y_2}, e_3 u_3^{x_3} u_4^{y_3},$ $e_4 u_4^{x_4} u_5^{y_4} \in \mathbb{RU}_{N^2}$ $(k_1, k_2, k_3, k_4) := (\text{dlog}_T(e_1 u_1^{x_1} u_2^{y_1}),$ $\text{dlog}_T(e_2 u_2^{x_2} u_3^{y_2}), \text{dlog}_T(e_3 u_3^{x_3} u_4^{y_3}),$ $\text{dlog}_T(e_4 u_4^{x_4} u_5^{y_4}))$ mod $N$ . $k := (k_1, k_2, k_3, k_4)$ . Else, Return $\perp$ .  $\text{pke.ct}/\perp \leftarrow \text{AIAE}.\text{Decrypt}(k, \text{aiae.ct}, \text{aux})$ . // $m/\perp \leftarrow \text{PKE}.\text{Dec}(\text{sk}, \text{pke.ct})$ : Parse $\text{pke.ct} = (\tilde{u}_1, \dots, \tilde{u}_8, \tilde{e}, t)$ . If $\tilde{e} \tilde{u}_1^{x_1} \tilde{u}_2^{y_1} \tilde{u}_3^{x_2} \tilde{u}_4^{y_2} \tilde{u}_5^{x_3} \tilde{u}_6^{y_3} \tilde{u}_7^{x_4} \tilde{u}_8^{y_4} \in \mathbb{RU}_{N^s}$ $m := \text{dlog}_T(\tilde{e} \tilde{u}_1^{x_1} \tilde{u}_2^{y_1} \tilde{u}_3^{x_2} \tilde{u}_4^{y_2} \tilde{u}_5^{x_3} \tilde{u}_6^{y_3}$ $\tilde{u}_7^{x_4} \tilde{u}_8^{y_4})$ mod $N^{s-1}$ . IF $t = g_1^m$ mod $N$ , Return $m$ . Else, Return $\perp$ .
---	--

图 5.20: 支持仿射函数集合的 KDM-CCA 安全公钥加密方案 PKE

上述方案的安全性由以下定理保证.

#### 定理 5.15

如果 (i)  $\text{AIAE}_{\text{DDH}}$  是弱  $\mathcal{F}_{\text{aff}}$ -RKA 安全的; (ii) GenN 算法的 DCR 假设成立; (iii) GenN 算法的 DL 假设成立. 则图??中构造的 PKE 方案是  $n$ -KDM[ $\mathcal{F}_{\text{aff}}$ ]-CCA 安全的,其中  $n \in \mathbb{N}$  表示用户数量、 $\mathcal{F}_{\text{aff}}$  是仿射函数集合.

下面简要描述证明的思路, 详细分析请参考文献 [**HLL-ASIACRYPT-2016**] 中定理 2 的证明.

1. 对于  $n$  个用户的私钥  $\text{sk}_i = (x_{i,j}, y_{i,j})_{j=1}^4$ ,  $i \in [n]$ ,
  - 每个私钥  $\text{sk}_i$ (从信息论意义上) 可以分为两个部分, 即模  $N$  的部分  $(x_{i,j}, y_{i,j})_{j=1}^4 \bmod N$  和模  $\phi(N)/4$  的部分  $(x_{i,j}, y_{i,j})_{j=1}^4 \bmod \phi(N)/4$ ;
  - 每个私钥  $\text{sk}_i$  可以通过在一个固定的基私钥  $(x_j, y_j)_{j=1}^4$  上加随机偏移量  $(\bar{x}_{i,j}, \bar{y}_{i,j})_{j=1}^4$  来生成, 即  $\text{sk}_i = (x_{i,j}, y_{i,j})_{j=1}^4 := (x_j, y_j)_{j=1}^4 + (\bar{x}_{i,j}, \bar{y}_{i,j})_{j=1}^4$ .
2. 每个用户的公钥  $\text{pk}_i = (h_{i,1}, \dots, h_{i,4})$  只会泄露对应私钥  $\text{sk}_i$  模  $\phi(N)/4$  部分的信息.
3. 对于敌手的 ENCRYPT 查询  $(i_\lambda, f_\lambda)$ , 挑战密文可能通过  $\text{pek.ct}$  部分泄露私钥  $\text{sk}_i$  ( $i \in [\ell]$ ) 的信息. 通过计算不可区分的变换, 能够保证 ENCRYPT 查询泄露的  $\text{sk}_i$  的信息是有限的.
  - 根据 IV<sub>d</sub> 假设, 可以改变 ENCRYPT 查询中  $\text{pke.ct}$  的生成方式, 使得  $\text{pke.ct}$  不泄露任何关于  $(x_j, y_j)_{j=1}^4 \bmod N$  的信息, 即不泄露基私钥模  $N$  部分的信息.
  - 根据 IV<sub>d</sub> 假设, 可以改变 ENCRYPT 查询中  $\text{kem.ct} (= \text{aux})$  的生成方式, 使得其封装一个与 AIAE.Encrypt 中使用的密钥不同的密钥. 假设 AIAE.Encrypt 使用的密钥是  $\text{k}_\lambda = (r_\lambda k_j^* + s_{\lambda,j})_{j=1}^4$ , 那么  $\text{kem.ct} (= \text{aux})$  中所封装的密钥则是  $(r_\lambda \cdot (k_j^* - \alpha_j x_j - \alpha_{j+1} y_j) - r_\lambda \cdot (\alpha_j \bar{x}_{i_\lambda,j} + \alpha_{j+1} \bar{y}_{i_\lambda,j}) + s_{\lambda,j})_{j=1}^4 \bmod N$ . 因此,  $(k_1^*, \dots, k_4^*)$  现在被  $(x_j, y_j)_{j=1}^4 \bmod N$  保护了起来.
4. Decrypt 查询也可能会泄露  $(x_j, y_j)_{j=1}^4 \bmod N$  的信息. 通过计算不可区分的变换, 保证 Decrypt 查询可以不使用  $(x_j, y_j)_{j=1}^4 \bmod N$  进行解密. 注意到敌手提交的密文只要满足  $\forall j \in [5], u_j \in \mathbb{SCR}_{N^2}$  和  $\forall j \in [8], \tilde{u}_j \in \mathbb{SCR}_{N^s}$ , 则 Decrypt 可以只用  $\phi(N)$  和私钥  $\text{sk}_i$  模  $\phi(N)/4$  部分的信息来完成解密过程.
  - 如果敌手提交的密文使得  $\exists j \in [5], u_j \notin \mathbb{SCR}_{N^2}$ , 则根据 AIAE<sub>DDH</sub> 的弱 INT- $\mathcal{F}_{\text{raff}}$ -RKA 安全性, 可以证明 AIAE.Decrypt 解密失败并输出  $\perp$ .
  - 如果敌手提交的密文使得  $\exists j \in [8], \tilde{u}_j \notin \mathbb{SCR}_{N^s}$ , 可以证明  $t \neq g_1^m \bmod N$ , 从而 PKE.Decrypt 解密失败并输出  $\perp$ .
5. 因此,  $(x_j, y_j)_{j=1}^4 \bmod N$  和  $(k_1^*, \dots, k_4^*)$  对于敌手而言都是均匀分布的. 此外, AIAE.Encrypt 总是使用  $\text{k}^* = (k_1^*, \dots, k_4^*)$  的受限仿射函数值作为密钥进行加密, 故再根据 AIAE<sub>DDH</sub> 的 IND- $\mathcal{F}_{\text{raff}}$ -RKA 安全性, PKE 的  $n$ -KDM[ $\mathcal{F}_{\text{aff}}$ ]-CCA 安全性得证.

 **笔记** IV<sub>d</sub> 假设可以看作是扩展的 Paillier 密文在加密 0 向量和敌手指定的消息向量是不可区分的. 扩展的 Paillier 加密方案的密文形式是  $g^r(1+N)^m \bmod N^{s+1}$ , 其中  $s \geq 2$ ,  $g$  是  $\mathbb{Z}_{N^{s+1}}^*$  上的  $N^s$  次剩余元素, 具体可参考文献 [**BG2010**].

**支持多项式函数集的 KDM-CCA 安全公钥加密方案.** 针对有界多项式函数集合  $\mathcal{F}_{\text{poly}}^d$  设计 KDM-CCA 安全的公钥加密方案, 其中  $d$  表示多项式函数集合里多项式函数的最大次数,  $d$  可以是安全参数  $\kappa$  的任意多项式, 仍然遵循图 ?? 中描述的通用设计方法, 即通过构造三个组件 KEM、PKE 和 AIAE. 该方案与前面介绍的支持仿射函数集合的 KDM-CCA 安全公钥加密方案(参见图 ??)共享相同的组件 KEM 和组件 AIAE<sub>DDH</sub>. 为了使 KDM-CCA 安全性支持所有有界次数的多项式函数, 需要重新设计组件 PKE, 使其加密算法 PKE.Encrypt 可以通过计算不可区分的变换, 转换成针对多项式函数的熵过滤器. 设计此类 PKE 组件的基本技术是多项式降元技术. Han 等人提出的降元技术可以将 KDM 询问的多项式函数的可能项式复杂度从  $\binom{8n+d}{8n} = \Theta(d^{8n})$  降至  $\binom{8+d}{8} = \Theta(d^8)$ . 由于该方案的描述较复杂, 在此不再赘述. 详细内容可参考文献 [**HLL-ASIACRYPT-2016**] 第 6 节.

### 5.3.4 QLH13 方案

2001 年, Camenisch 和 Lysyanskaya [**CL-EUROCRYPT-2001**] 提出一个实用的匿名证书系统. 在该系统中, 为了阻止用户将自己的密钥分享给他人使用, 作者提出通过循环加密私钥来实现“all-or-nothing sharing”的性质: 一个用户若允许别人使用他的私钥, 则别人可以得到他的所有证书(具体指方案的私钥). 通常,  $n$  个用户的循环加密方案包含以下  $n$  个密文:  $\{\text{Encrypt}(pk_i, sk_{i+1 \bmod n})\}_{i \in [n]}$ . 显而易见, 若知道一个私钥, 通过该组密文可以得到所有用户的私钥.

2013 年, Qin 等人 [**Qin-ACISP-2013**] 指出, 实现“all-or-nothing sharing”性质也可通过如下形式的循环加密实

现, 从而设计了一个具有 KDM-CCA 安全性的 CS 方案. 除了需要对加密的消息重新编码外, 方案的结构与原始 CS 方案完全相同, 具有更好的应用价值.

$$\text{Encrypt}(pk_1, sk_1 - sk_2), \text{Encrypt}(pk_2, sk_2 - sk_3), \dots, \text{Encrypt}(pk_n, sk_n - sk_1).$$

### 5.3.4.1 KDM 函数族定义

令  $L := \text{poly}(\kappa)$  和  $M := \text{poly}(\kappa)$  为两个正整数,  $q$  为一个素数. 令  $\mathcal{X}$  为  $\mathbb{Z}_q$  的一个有限子集. 环  $\mathbb{Z}_q$  上的函数族  $\mathcal{F}_{q,n}^{L,M} : \mathcal{X}^n \rightarrow \mathbb{Z}_q$  定义如下: 对于任意  $f \in \mathcal{F}_{q,n}^{L,M}$ , 函数  $f$  的表达式如下:

$$f(x_1, \dots, x_n) = \sum_{t=1}^L \prod_{i,j \in [n]} \alpha_{i,j,t} (x_i - x_j)^{a_{i,j,t}} \pmod{q},$$

其中  $\alpha_{i,j,t} \in \mathbb{Z}_q$ ,  $\sum_{i,j \in [n]} a_{i,j,t} \leq M$ . 实际上,  $f$  可以看做是一个以  $n^2$  个  $\{x_i - x_j\}_{i,j \in [n]}$  为变量的多变量函数. 令  $\mathcal{S}(n^2)$  表示环  $\mathbb{Z}_q$  上所有具有  $n^2$  个变量的多项式, 则

$$\mathcal{F}_{q,n}^{L,M} = \{g : (x_i - x_j)_{i,j \in [n]} \rightarrow \mathbb{Z}_q \mid g \in \mathcal{S}(n^2), x_i \in \mathcal{X}, i \in [n]\}.$$

显然, 集合  $\mathcal{F}_{q,n}^{L,M}$  包含  $(x_1 - x_2), (x_2 - x_3), \dots, (x_{n-1} - x_n), (x_n - x_1)$  这些特殊的函数. 由此可得

$$\begin{pmatrix} x_1 - x_2 \\ x_2 - x_3 \\ \vdots \\ x_{n-1} - x_n \\ x_n - x_1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & -1 & 0 & \dots & 0 & 0 \\ 0 & 1 & -1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & -1 \\ -1 & 0 & 0 & \dots & 0 & 1 \end{pmatrix}}_A \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

其中矩阵  $A_{n \times n}$  的秩为  $n$ .

对于任意向量  $\vec{b} = (b_1, b_2, \dots, b_n) \in \mathbb{Z}_q^n$  和  $\beta \in \mathbb{Z}_q$ , 函数

$$f(x_1, x_2, \dots, x_n) = (\vec{b} \cdot A \cdot \vec{x}) + \beta$$

$$= (\vec{b} \cdot A \cdot \vec{x}) + \beta = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \beta$$

构成一个仿射函数且属于函数集合  $\mathcal{F}_{q,n}^{L,M}$ .

令  $\mathcal{I} = \{\vec{b} \cdot A \cdot (x_1, x_2, \dots, x_n)^T + \beta \mid \vec{b} \in \mathbb{Z}_q^n, \beta \in \mathbb{Z}_q\}$ ,  $\Gamma$  为所有从  $\mathcal{X}^n$  到  $\mathbb{Z}_q$  的仿射函数集合. 则  $\mathcal{I} = \mathcal{F}_{q,n}^{L,M} \cap \Gamma$ . 因此, 新定义的函数集合不仅包含了部分仿射函数, 而且当多项式的次数大于 1 时, 该函数集合还包含了许多不属于仿射函数的一般函数, 见图 ??.

当考虑依赖密钥消息加密安全性时, 集合  $\mathcal{X}$  替换为私钥空间  $\mathcal{K}$ , 而函数族定义为  $\mathcal{F}_{q,n}^{L,M} = \{g : (sk_i - sk_j)_{i,j \in [n]} \rightarrow \mathbb{Z}_q \mid g \in \mathcal{S}(n^2), sk_i \in \mathcal{K}, i \in [n]\}$ . 当计算某一元素的函数值时, 默认假设  $\mathcal{K} \subseteq \mathbb{Z}_q$ .

如果  $\mathcal{K} \not\subseteq \mathbb{Z}_q$ , 我们假设存在一个有效的单射函数将  $\mathcal{K}$  映射到  $\mathbb{Z}_q$  的一个向量空间上. 所有私钥  $sk_i$  被替换为  $\mathbb{Z}_q$  上的某个向量.

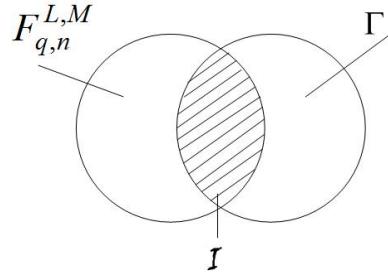


图 5.21: 函数族之间的关系

### 5.3.4.2 KDM-CCA 安全的 Cramer-Shoup 方案

Cramer-Shoup 方案 [CramerS03] 的消息空间是一个阶为素数  $q$  的有限循环群  $\mathbb{G}$ , 而私钥空间为  $\mathbb{Z}_q^6$ . 为了把私钥作为消息进行加密, 必须将方案的密钥  $sk = (x_1, x_2, x_3, x_4, x_5, x_6) \in \mathbb{Z}_q^6$  编码为群  $\mathbb{G}$  上的六个元素. 假设存在一个有效的编码算法  $\text{Encode} : \mathbb{Z}_q \rightarrow \mathbb{G}$  和解码算法  $\text{Decode} : \mathbb{G} \rightarrow \mathbb{Z}_q$  并且对于所有  $x \in \mathbb{Z}_q$  满足  $\text{Decode}(\text{Encode}(x)) = x$ . 下面, 给出这种编码算法的一种具体构造.

基于上述消息编码的 Tailored Cramer-Shoup 加密方案  $\text{TCS} = (\text{Setup}, \text{KeyGen}, \text{Encrypt}, \text{Decrypt})$  如构造 ?? 所示.

#### 构造 5.13 (Tailored Cramer-Shoup 加密方案)

- $\text{Setup}(1^\kappa)$ : 选择一个阶为素数  $q$  的有限循环群  $\mathbb{G}$ , 两个随机生成元  $g, \hat{g} \xleftarrow{\text{R}} \mathbb{G}$  及一个目标抗碰撞哈希函数  $H : \mathbb{G}^3 \rightarrow \mathbb{Z}_q$ . 返回系统参数  $pp = (q, \mathbb{G}, g, \hat{g}, H)$ .
- $\text{KeyGen}(pp)$ : 随机选择  $(x_1, \dots, x_6) \xleftarrow{\text{R}} \mathbb{Z}_q^6$  并计算

$$h_1 = g^{x_1} \hat{g}^{x_2} \quad h_2 = g^{x_3} \hat{g}^{x_4} \quad h_3 = g^{x_5} \hat{g}^{x_6}.$$

返回公钥  $pk = (h_i)_{i \in [3]}$  和私钥  $sk = (x_i)_{i \in [6]}$ .

- $\text{Encrypt}(pk, m)$ : 将公钥  $pk$  展开为  $(h_1, h_2, h_3)$ . 对于消息  $m \in \mathbb{Z}_q$ , 随机选择  $r \xleftarrow{\text{R}} \mathbb{Z}_q$  并计算

$$u = g^r, \quad \hat{u} = \hat{g}^r, \quad c = h_1^r \cdot \text{Encode}(m), \quad v = (h_2^t h_3)^r$$

其中  $t = H(u, \hat{u}, c)$ . 返回密文  $C = (u, \hat{u}, c, v)$ .

- $\text{Decrypt}(sk, C)$ : 首先, 将私钥  $sk$  展开为  $(x_i)_{i=1}^6$ , 密文  $C$  展开为  $(u, \hat{u}, c, v)$ . 计算  $t = H(u, \hat{u}, c)$  并验证  $u^{x_3 t + x_5} \cdot \hat{u}^{x_4 t + x_6} \stackrel{?}{=} v$  是否成立. 若不成立, 返回  $\perp$ ; 否则返回  $\text{Decode}(c/u^{x_1} \hat{u}^{x_2})$ .



在 Tailored Cramer-Shoup 加密方案中, 消息编码算法和解码算法的具体构造如下:

- 系统参数  $\text{Setup}(1^\kappa)$ : 选择一个安全素数  $p = 2q + 1$ , 即  $p$  和  $q$  都为素数. 令  $\mathbb{G} = \mathbb{QR}_p$  表示  $\mathbb{Z}_p^*$  的二次剩余群. 选择两个随机生成元  $g, \hat{g} \in \mathbb{QR}_p$ . 则系统参数为  $pp = (p, q, \mathbb{QR}_p, g, \hat{g})$ .
- 编码算法  $\text{Encode}(x)$ : 令  $\left(\frac{x}{p}\right)$  表示 Legendre 符号. 则编码算法  $\text{Encode} : \mathbb{Z}_q \rightarrow \mathbb{QR}_p$  定义为:

$$\text{Encode}(x) := \begin{cases} x, & \text{if } \left(\frac{x}{p}\right) = 1 \\ p - x, & \text{if } \left(\frac{x}{p}\right) = -1 \end{cases}$$

- 解码算法  $\text{Decode}(y)$ : 解码算法  $\text{Decode} : \mathbb{QR}_p \rightarrow \mathbb{Z}_q$  定义为:

$$\text{Decode}(y) := \begin{cases} y, & \text{if } 1 \leq y \leq q \\ p - y, & \text{if } q < y \leq p - 1 \end{cases}$$

在上述编码/解码算法中, 对于每个  $x \in \mathbb{Z}_q$ , 如果  $\left(\frac{x}{p}\right) = 1$ , 则  $x \in \mathbb{QR}_p$ ; 如果  $\left(\frac{x}{p}\right) = -1$ , 则  $\left(\frac{p-x}{p}\right) = 1$ . 因此,  $p-x \in \mathbb{QR}_p$ . 对于不同的  $x_1, x_2 \in \mathbb{Z}_q$ , 可知  $\text{Encode}(x_1) \neq \text{Encode}(x_2)$ . 故,  $\text{Encode}$  是一个双射. 类似地, 可以验证  $\text{Decode}$  是相应的逆映射.

**正确性.** 对于消息  $m$  的密文  $C = (u, \hat{u}, c, v)$ , 我们有

$$\begin{cases} u^{x_3 t + x_5} \cdot \hat{u}^{x_4 t + x_6} = (g^{x_3} \hat{g}^{x_4})^{tr} \cdot (g^{x_5} \hat{g}^{x_6})^r = (h_2^t h_3)^r = v \\ \text{Decode}\left(\frac{c}{u^{x_1} \hat{u}^{x_2}}\right) = \text{Decode}\left(\frac{h_1^r \cdot \text{Encode}(m)}{h_1^r}\right) = \text{Decode}(\text{Encode}(m)) = m \end{cases}.$$

由此说明 Tailored Cramer-Shoup 加密方案是正确的.

**KDM-CCA 安全性.** 方案的安全性由下面的定理保证:

### 定理 5.16

对于任意正整数  $n \geq 2$ ,  $L$  和  $M$ , 实例化的 Cramer-Shoup 方案相对于函数集  $\mathcal{F}_{q,n}^{L,M}$  是 KDM-CCA 安全的. 具体可得:

$$\text{Adv}_{\text{TCS}, \mathcal{A}}^{\text{kdm-cca}}(\kappa) \leq 2nQ_e \cdot \left( \text{Adv}_{\mathbb{QR}_p, \mathcal{D}}^{\text{ddh}}(\kappa) + \text{Adv}_{\mathcal{H}, \mathcal{B}}^{\text{tcr}}(\kappa) + \frac{(Q_d + 4)}{q} \right)$$

这里我们假设  $\mathcal{A}$  最多询问  $Q_e$  次加密服务和  $Q_d$  次解密服务.



**证明** 我们仅给出定理的证明思路, 具体证明可参考文献 [Qin-ACISP-2013].

由于 Tailored CS 加密方案与原始的 CS 加密方案仅在消息的形式上不同, 因此该方案仍然是 IND-CCA 安全的. 要证明它的 KDM-CCA 安全性, 我们只需要给出模拟私钥函数  $f(sk)$  加密谕言机的方法. 假设  $n$  个用户的私钥分别为  $sk_i = (x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}, x_{i,5}, x_{i,6})$  ( $i = 1, 2, \dots, n$ ). 则方案将私钥变换为消息的 KDM 函数族  $\mathcal{F}_{q,n}^{L,M}$  的具体形式如下:

$$\begin{aligned} f(sk_1, \dots, sk_n) &= f((x_{i,1}, x_{i,2}, x_{i,3}, x_{i,4}, x_{i,5}, x_{i,6})_{i=1}^n) \\ &= \sum_{t=1}^L \prod_{i,j \in [n], s \in [6]} \alpha_{i,j,t} (x_{i,s} - x_{j,s})^{a_{i,j,t}} \pmod{q}, \end{aligned}$$

其中  $\alpha_{i,j,t} \in \mathbb{Z}_q$ ,  $a_{i,j,t} \in \mathbb{N}$ .

我们知道, 公钥和私钥的形式分别为  $pk = (h_i)_{i=1}^3 = (g^{x_{2i-1}} \hat{g}^{x_{2i}})_{i=1}^3$  和  $sk = (x_i)_{i=1}^6$ , 其中  $g, \hat{g}$  是随机选取的生成元,  $(x_i)_{i=1}^6$  是从  $\mathbb{Z}_q$  中独立选取的. 当使用公钥  $pk$  加密消息  $m \in \mathbb{Z}_q$  时, 先选取随机数  $r \in \mathbb{Z}_q$ , 再计算密文  $C = (u, \hat{u}, c, v) = (g^r, \hat{g}^r, h_1^r \cdot \text{Encode}(m), h_2^r h_3^r)$ , 其中  $t = \mathcal{H}(u, \hat{u}, c)$ . 而在证明过程中, 模拟者可以通过一个公钥/私钥对  $(pk^*, sk^*)$  来生成其他  $n$  个用户的公钥. 首先, 模拟者随机选取差值  $\widehat{sk}_i \in \mathbb{Z}_q^6$ , 并利用这些差值和公钥  $pk^*$  构造新的公钥  $pk_i$  且相应的私钥为  $sk_i = sk^* + \widehat{sk}_i$ . 模拟者虽然不知道  $pk_i$  对应的真正私钥, 但是对于任意两个用户的私钥之差是可以计算出来的. 因此, 模拟者可以完美地计算任意 KDM 函数  $f \in \mathcal{F}_{q,n}^{L,M}$  的值  $f(sk_1, \dots, sk_n)$ , 从而完美模拟 KDM 加密谕言机. 对于敌手的解密询问, 假设密文  $C = (u, \hat{u}, c, v)$  是使用公钥  $pk_i$  加密某一消息  $m$  得到的. 那么, 模拟者可以构造一个使用公钥  $pk^*$  加密另一消息  $m^*$  的密文  $C^* = (u, \hat{u}, c, v^*)$ , 并且利用  $m^*$  恢复真正的消息  $m$ . 由此可知, 模拟者可以利用自己的挑战者提供的解密服务(私钥  $sk^*$  对应的解密谕言机)来获得消息  $m^*$ , 从而恢复出原始消息. 这说明 Tailored CS 加密方案的 KDM-CCA 安全性可以归约到自身的 IND-CCA 安全性上.

## 第六章 公钥加密的功能性扩展

### 内容提要

- 确定性公钥加密
- 可搜索公钥加密
- 代理重加密【待补充】
- 门限公钥加密【待补充】

## 6.1 确定性公钥加密

确定性公钥加密由 Bellare, Boldyreva 和 O’Neill [BBO2007] 于 2007 年提出, 是一种加密算法确定, 每个明文对应一个密文的公钥加密方案. 经典的(教科书式)RSA 加密方案就是一种确定性公钥加密方案. 确定性加密方案无法实现标准的语义安全性, 但是在加密数据检索等方面具有广泛的应用. 正因如此, 确定性加密得到了广泛而深入的研究, 包括确定性加密的安全模型 [BFO2008; BS-JOC-2014], 确定性加密的方案构造 [Wee-EUROCRYPT-2012; FOR2012; FOR-JOC-2015], 确定性加密的功能推广 [XXZ2012; ZFLW-IWSEC-2017] 等.

### 6.1.1 安全模型

同传统的公钥加密方案, 一个确定性公钥加密方案 DPKE 包含参数生成算法  $\text{Setup}$ , 密钥生成算法  $\text{KeyGen}$ , 加密算法  $\text{Encrypt}$  和解密算法  $\text{Decrypt}$  四个算法. 明文空间一般要求具有较高的最小熵, 否则敌手可以根据明文分布推出密文所对应的明文. 2011 年, Brakerski 和 Segev [BS2011] 提出确定性公钥加密的辅助输入安全模型. 在该模型中, 除了明文分布信息外, 敌手还拥有被加密明文的额外信息, 但是从该额外信息恢复明文是困难的. 为了刻画敌手的这一攻击能力, 我们定义消息空间  $\mathcal{M}$  上的一组难于求逆辅助输入 (Hard-to Invert Auxiliary Inputs) 函数:  $\mathcal{F} = \{f_\kappa\}$ . 对于任意 PPT 敌手  $\mathcal{A}$ , 如果下面的不等式成立, 其中  $x \xleftarrow{\text{R}} \mathcal{M}$ ,

$$\Pr[\mathcal{A}(1^\kappa, f_\kappa(x)) = x] \leq \delta$$

则称一个可有效计算的函数族  $\mathcal{F} = \{f_\kappa\}$  关于可有效采样(消息)分布  $\mathcal{M}$  是  $\delta$ -难于求逆的.

#### 定义 6.1 (辅助输入安全性)

令  $\mathcal{F} = \{f : \mathcal{M} \rightarrow \{0,1\}^*\}$  是一个从消息空间到任意长度空间的难于求逆辅助输入函数族. 定义确定性公钥加密方案 DPKE 的辅助输入敌手  $\mathcal{A}$  的优势函数如下:

$$\text{Adv}_{\mathcal{A}, \mathcal{F}, \text{DPKE}}^{\text{PrivInd}}(\kappa) = \Pr \left[ \beta' = \beta : \begin{array}{l} pp \leftarrow \text{Setup}(1^\kappa); \\ (pk, sk) \leftarrow \text{KeyGen}(pp); \\ (M_0, M_1) \xleftarrow{\text{R}} \mathcal{M}; \\ \beta \xleftarrow{\text{R}} \{0,1\}; \\ C \leftarrow \text{Encrypt}(pk, M_\beta); \\ \beta' \leftarrow \mathcal{A}(pk, C, f(M_0), f(M_1)) \end{array} \right] - \frac{1}{2}.$$

如果对于任意的 PPT 敌手  $\mathcal{A}$ , 优势函数  $\text{Adv}_{\mathcal{A}, \mathcal{F}, \text{DPKE}}^{\text{PrivInd}}(\kappa)$  关于安全参数  $\kappa$  是可忽略的, 则称确定性公钥加密方案 DPKE 是  $(\mathcal{F}, \mathcal{M})$ -PrivInd 安全的.



### 6.1.2 基于对偶仿射哈希的 DPKE 方案

#### 6.1.2.1 对偶仿射哈希

对偶仿射哈希 (Dual Projective Hashing) 是 2012 年 Wee [Wee-EUROCRYPT-2012] 提出的一种新型密码学原语. 它类似于 Cramer 和 Shoup 提出的仿射哈希, 不同之处是对于非法元素, 仿射哈希的输出结果是随机的(平滑性), 而对偶仿射哈希要求输出结果能够唯一确定哈希密钥, 且存在一个陷门有效恢复哈希密钥. 下面给出对偶仿射哈希的定义, 其中部分符号无特殊说明外, 与第四章哈希证明系统定义中的符号含义一致.

#### 定义 6.2 (对偶仿射哈希)

一个对偶仿射哈希 DPH 包含以下五个算法:

- $\text{Setup}(1^\kappa)$ : 以安全参数  $\kappa$  为输入, 输出公开参数  $pp = (\mathsf{H}, SK, PK, X, L, W, \Pi, \alpha)$ , 其中  $\mathsf{H} : X \times$

$SK \rightarrow \Pi$  是一个由集合元素  $u \in X$  索引的一族带密钥哈希函数, 记作  $H_u(sk)$ <sup>a</sup>,  $L$  是定义在  $X$  上的  $\mathcal{NP}$  语言,  $W$  是对应的证据集合,  $\alpha(\cdot)$  是从私钥集合  $SK$  到公钥集合  $PK$  的投射函数. 我们将  $L$  称之为 Yes 实例,  $X \setminus L$  称之为 No 实例. 特别地, 存在两个抽样算法:  $\text{SampleYes}(pp)$  输出一对随机 Yes 实例  $(u, w)$ , 其中  $u$  在  $L$  上均匀分布,  $w$  是相应的证据;  $\text{SampleNo}(pp)$  输出一对随机 No 实例  $(u, td)$ , 其中  $u$  在  $X \setminus L$  上均匀分布,  $td$  是相应的(求逆)陷门. 此外, 对偶仿射哈希要求 Type 2 类型的子集成员判定问题是困难的.

- $\text{KeyGen}(pp)$ : 以公开参数  $pp$  为输入, 随机采样  $sk \xleftarrow{R} SK$ , 计算  $pk \leftarrow \alpha(sk)$ , 输出  $(pk, sk)$ .
- $\text{PrivEval}(sk, u)$ : 以私钥  $sk$  和  $u \in X$  为输入, 输出  $\pi = H_u(sk)$ .
- $\text{PubEval}(pk, u, w)$ : 以公钥  $pk$ ,  $u \in L$  及相应的  $w$  为输入, 输出  $\pi = H_u(sk)$ , 其中  $pk = \alpha(sk)$ .
- $\text{TdInv}(td, pk, \pi)$ : 对于任意  $(u, td) \leftarrow \text{SampleNo}(pp)$  和任意  $sk \in SK$ , 若  $pk = \alpha(sk)$ ,  $\pi = H_u(sk)$ , 该算法输出  $sk \in SK$ , 使得  $\alpha(sk) = pk$ ,  $H_u(sk) = \pi$ .

<sup>a</sup>不同于哈希证明系统, 这里用集合  $X$  中的元素作为哈希函数的索引, 而将私钥空间中的元素作为哈希函数的输入.



### 6.1.2.2 DPKE 方案的构造

#### 构造 6.1 (基于对偶仿射哈希的 DPKE 方案)

基于一个对偶仿射哈希  $\text{DPH} = (\text{Setup}, \text{KeyGen}, \text{PrivEval}, \text{PubEval}, \text{TdInv})$ , 构造的确定性公钥加密方案如下:

- $\text{Setup}(1^k)$ : 运行  $pp \leftarrow \text{DPH}.\text{Setup}(1^k)$ , 输出系统参数  $pp = (\mathsf{H}, SK, PK, X, L, W, \Pi, \alpha)$ .
- $\text{KeyGen}(pp)$ : 运行  $(u, td) \leftarrow \text{DPH}.\text{SampleNo}(pp)$ , 输出公钥  $pk = u$  和私钥  $sk = td$ .
- $\text{Encrypt}(pk, M)$ : 以公钥  $pk = u$  和明文  $M \in SK$  为输入, 输出密文  $C = \alpha(M) || H_u(M)$ .
- $\text{Decrypt}(sk, C)$ : 以私钥  $sk = td$  和密文  $C = y_0 || y_1$  为输入, 输出明文  $M' = \text{TdInv}(td, y_0, y_1)$ .



**笔记** 在第 4.4.3 节中, 利用仿射哈希证明系统构造公钥加密或密钥封装方案时, 仿射哈希的公钥和私钥分别作为加密或密钥封装方案的公钥和私钥, 而随机抽样元素则作为密文的一部分. 在构造确定性加密方案时, 仿射哈希的公钥作为密文, 而随机抽样的元素则作为公钥, 用法刚好相反, 这也许作者将其称为对偶仿射哈希的原因.

在分析方案 ?? 的安全性之前, 先回顾一下可重构提取器 (Reconstructive Extractor) 的概念 [Trevisan-ACM-2001].

#### 定义 6.3 (可重构提取器)

一个可重构提取器  $(\epsilon, \delta)$ -可重构提取器包含如下两个函数 ( $\text{Ext}, \text{Rec}$ ):

- $\text{Ext} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \Sigma$  是一个提取算法.
- $\text{Rec}(1^n, 1/\epsilon)$ : 对于任意  $x \in \{0, 1\}^n$  和任意函数  $D$  满足

$$|\Pr[\mathcal{D}(r, \text{Ext}(x, r)) = 1] - \Pr[\mathcal{D}(r, \text{Ext}(x, \sigma)) = 1]| \geq \epsilon$$

其中  $r \xleftarrow{R} \{0, 1\}^d$ ,  $\sigma \xleftarrow{R} \Sigma$ , 则以  $(1^n, 1/\epsilon)$  为输入, 在运行时间  $\text{poly}(n, 1/\epsilon, \log |\Sigma|)$  内,  $\text{Rec}$  输出  $x \in \{0, 1\}^n$  的概率至少为  $\delta$ , i.e.,

$$\Pr[\text{Rec}^{\mathcal{D}}(1^n, 1/\epsilon) = x] \geq \delta.$$



根据文献 [Trevisan-ACM-2001], 任意  $(\epsilon, \delta)$ -可重构提取器也是一个极小熵为  $1/\delta$  的强提取器. 类似地, 对于  $\delta \cdot \text{negl}(\cdot)$ -难于求逆辅助输入, 任意  $(\epsilon, \delta)$ -可重构提取器的输出结果是伪随机的.

**可重构提取器的构造.** 事实证明, 随机线性函数不仅是一个好的随机数提取器, 也是一个好的可重构提取器.

**引理 6.1 ([BrakerskiG10])**

令  $q$  是一个素数. 则函数  $\text{Ext} : \{0, 1\}^n \times \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q$ , 定义为  $(\mathbf{x}, \mathbf{a}) \mapsto \mathbf{x}^\top \vec{a}$ , 是一个  $(\epsilon, \frac{\epsilon^3}{512 \cdot n \cdot q^2})$ -可重构提取器. 

根据引理 ??,  $\text{Ext}$  将  $(x_1, \dots, x_n)$  和  $(a_1, \dots, a_n)$  映射为  $a_1 x_1 + \dots + a_n x_n \pmod{q}$ . 此外, 该引理可以推广到以下情况:

- $\mathbb{G}$  是一个阶为素数  $q$  的循环群,  $g$  是生成元,  $\text{Ext} : \{0, 1\}^n \times \mathbb{G}^n \rightarrow \mathbb{G}$  定义为  $(\mathbf{x}, g^\mathbf{a}) \mapsto g^{\mathbf{x}^\top \mathbf{a}}$ .

方案 ??的安全性由下面的定理保证.

**定理 6.1**

如果  $(x, pp) \mapsto \alpha(pp, x)$  是一个  $(\epsilon, \delta)$ -可重构提取器, 子集成员判定问题是困难的, 则构造方案 ??是  $(\mathcal{F}, \mathcal{M})$ -PrivSind 安全的. 

**证明** 令  $S_i$  表示对手在 Game<sub>i</sub> 中的成功事件. 以游戏序列的方式组织证明如下:

Game<sub>0</sub>: 该游戏是标准的 PrivInd 游戏, 挑战者  $\mathcal{CH}$  和对手  $\mathcal{A}$  交互如下:

- 初始化:  $\mathcal{CH}$  运行  $\text{Setup}(1^\kappa)$  生成公开参数  $pp$ , 同时运行  $\text{KeyGen}(pp)$  生成公私钥对  $(pk, sk) = (u, td)$ .  $\mathcal{CH}$  将  $(pp, u)$  发送给  $\mathcal{A}$ .
- 挑战:  $\mathcal{CH}$  选择两个随机消息  $(M_0, M_1) \xleftarrow{R} \mathcal{M}$  和随机比特  $\beta \in \{0, 1\}$ , 将  $C = \alpha(M_\beta) || \mathsf{H}_u(M_\beta) \leftarrow \text{Encrypt}(pk, M_\beta)$ ,  $f(M_0)$  和  $f(M_1)$  发送给  $\mathcal{A}$ .
- 猜测:  $\mathcal{A}$  输出对  $\beta$  的猜测  $\beta'$ .  $\mathcal{A}$  成功当且仅当  $\beta' = \beta$ .

根据定义, 我们有:

$$\mathsf{Adv}_{\mathcal{A}, \mathcal{F}, \text{DPKE}}^{\text{PrivInd}}(\kappa) = |\Pr[S_0] - 1/2|$$

Game<sub>1</sub>: 该游戏与 Game<sub>0</sub> 的唯一不同在于公钥  $pk$  的选择方式. 将  $(u, td) \leftarrow \text{SamplNo}(pp)$  替换为  $(u, w) \leftarrow \text{SamplYes}(pp)$ , i.e., 公钥  $u$  从 Yes 实例集合中选择. 根据子集成员判定问题困难性, 我们有

$$|\Pr[S_0] - \Pr[S_1]| \leq \mathsf{Adv}_{\mathcal{A}, \mathcal{F}, \text{SMP}}(\kappa)$$

Game<sub>2</sub>: 该游戏与 Game<sub>1</sub> 的唯一不同在于密文  $\mathsf{H}_u(M_\beta)$  的计算方式变为  $\text{PubEval}(\alpha(pp, M_\beta), u, w)$ . 根据仿射哈希的性质, Game<sub>1</sub> 和 Game<sub>2</sub> 的分布是完全一样的. 因此, 我们有

$$\Pr[S_1] = \Pr[S_2]$$

Game<sub>3</sub>: 该游戏与 Game<sub>2</sub> 的唯一不同在于密文  $\alpha(pp, M_\beta)$  的计算方式变为随机选择  $\sigma \xleftarrow{R} \Sigma$ . 具体地, 密文从  $\alpha(pp, M_\beta) || \text{PubEval}(\alpha(pp, M_\beta), u, w)$  变为  $\sigma || \text{PubEval}(\sigma, u, w)$ . 可以证明, 如果对手  $\mathcal{A}$  在 Game<sub>2</sub> 和 Game<sub>3</sub> 中的优势相差至多  $2\epsilon$ . 否则, 我们可以利用  $\mathcal{A}$  构造一个区分器  $\mathcal{D}$ , 使得

$$|\Pr[\mathcal{D}(pp, \alpha(pp, M), f(M)) = 1] - \Pr[\mathcal{D}(pp, \sigma, f(M)) = 1]| \geq \epsilon$$

其中  $pp \leftarrow \text{DPH.Setup}$ ,  $M \xleftarrow{R} \mathcal{M}$ ,  $\sigma \xleftarrow{R} \Sigma$ . 区分器  $\mathcal{D}$  模拟  $\mathcal{A}$  的视图方式如下: 首先,  $\mathcal{D}$  随机选择  $(u, w) \leftarrow \text{SamplYes}(pp)$ , 将  $u$  作为公钥发送给  $\mathcal{A}$ . 其次,  $\mathcal{D}$  随机选择  $\beta \xleftarrow{R} \{0, 1\}$ , 将自己的挑战信息  $M$  作为  $M_\beta$ , 并从空间  $\mathcal{M}$  中随机选择一个消息作为  $M_{1-\beta}$ . 接下来,  $\mathcal{D}$  利用公钥  $u$  及其证据  $w$  计算  $y_1 = \text{PubEval}(y_0, u, w)$  并将密文  $y_0 || y_1$  发送给  $\mathcal{A}$ . 最后, 根据对手  $\mathcal{A}$  的输出结果  $\beta'$ , 如果  $\beta = \beta'$ ,  $\mathcal{D}$  输出 1, 否则输出 0. 显然, 若  $y_0 = \alpha(pp, M)$ , 则  $\mathcal{D}$  完美地模拟了  $\mathcal{A}$  在 Game<sub>2</sub> 中的视图; 若  $y_0 = \sigma$ , 则  $\mathcal{D}$  完美地模拟了  $\mathcal{A}$  在 Game<sub>3</sub> 中的视图. 由于这两种情况的概率各为  $1/2$ , 所以  $\mathcal{D}$  的区分概率至少为  $\epsilon$ . 由此, 我们可以利用  $\text{Rec}^{\mathcal{D}}$  以概率  $\delta$  从  $f(M)$  中恢复  $M$ . 也就是说,

我们能够以至少  $\epsilon \cdot \delta$  的概率在分布  $\mathcal{M}$  上求函数  $f$  的逆. 这与假设相矛盾. 由此可得

$$|\Pr[S_2] - \Pr[S_3]| \leq 2\epsilon$$

在 Game<sub>3</sub> 中, 由于  $\mathcal{A}$  的视图与挑战比特  $\beta$  完全独立不相关, 所以  $\Pr[S_3] = \frac{1}{2}$ .

综上, 定理得证! □

### 6.1.3 基于 DLIN 的实例化方案

下面介绍一种基于 DLIN 问题的对偶仿射哈希的具体实现, 结合通用构造方案 ??, 可以得到一个基于 DLIN 假设的确定性公钥加密方案 [BS2011].

令  $\mathbb{G}$  是一个阶为素数  $q$  的循环群,  $g$  是它的一个生成元.  $d$ -DLIN 假设描述如下: 给定  $g_1, \dots, g_{d+1}, g_1^{r_1}, \dots, g_d^{r_d}$ , 其中  $g_1, \dots, g_{d+1} \xleftarrow{\text{R}} \mathbb{G}$ ,  $r_1, \dots, r_d \xleftarrow{\text{R}} \mathbb{Z}_q$ , 则  $g_{d+1}^{r_1+\dots+r_d}$  依然是伪随机的, 与  $\mathbb{G}$  中的随机元素在计算上不可区分. 显然, 经典的 DDH 假设可以看作是  $d=1$  的 DLIN 假设.

#### 构造 6.2 (对偶仿射哈希)

一个对偶仿射哈希 DPH 包含以下五个算法:

- $\text{Setup}(1^\kappa)$ : 公开参数定义为  $pp = (\mathbb{G}, g^{\mathbf{P}})$ , 其中  $\mathbf{P} \xleftarrow{\text{R}} \mathbb{Z}_q^{d \times m}$ . 除此之外, 公开参数还定义了私钥空间  $SK = \{0, 1\}^m$ , 公钥空间  $PK = \mathbb{G}^m$ , Yes 实例集合  $L = \{g^{\mathbf{WP}} : \mathbf{W} \in \mathbb{Z}_q^{m \times d}\}$  和 No 实例集合  $X \setminus L = \{g^{\mathbb{A}} : \mathbb{A} \in \mathbb{Z}_q^{m \times m} \text{ 满秩}\}$ <sup>a</sup>, 证据空间  $W = \mathbb{Z}_q^{m \times d}$ , 哈希值空间  $\Pi = \mathbb{G}^m$ . 对于任意  $\mathbf{x} \in SK$  和任意  $\mathbf{U} \in X$ , 哈希函数  $H$  和仿射映射  $\alpha(pp, \cdot)$  的定义分别如下:

$$H_{\mathbf{U}}(\mathbf{x}) = \mathbf{U}^{\mathbf{x}} \in \mathbb{G}^m \text{ and } \alpha(pp, \mathbf{x}) = g^{\mathbf{Px}}$$

采样算法 SampleYes 和 SampleNo 分别定义如下:

$$(u = g^{\mathbf{WP}}, w = \mathbf{W}) \leftarrow \text{SampleYes}(pp) \text{ and } (u = g^{\mathbb{A}}, td = \mathbf{W}^{-1}) \leftarrow \text{SampleNo}(pp)$$

其中  $\mathbf{W} \xleftarrow{\text{R}} \mathbb{Z}_q^{m \times d}$ ,  $\mathbb{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{m \times m}$ <sup>b</sup>.

- $\text{KeyGen}(pp)$ : 以公开参数  $pp$  为输入, 随机选择  $sk = \mathbf{x} \xleftarrow{\text{R}} SK$ , 计算  $pk \leftarrow \alpha(pp, sk) = g^{\mathbf{Px}}$ .
- $\text{PrivEval}(sk, u)$ : 以私钥  $sk = \mathbf{x} \in \{0, 1\}^m$  和  $u = \mathbf{U} \in \mathbb{G}^{m \times m}$  为输入, 输出  $\pi = H_u(sk) = \mathbf{U}^{\mathbf{x}}$ .
- $\text{PubEval}(pk, u, w)$ : 以公钥  $pk = g^{\mathbf{Px}}$ ,  $u = g^{\mathbf{WP}} \in L$  及相应的  $w = \mathbf{W}$  为输入, 输出  $\pi = H_u(sk) = g^{\mathbf{W} \cdot \mathbf{Px}}$ .
- $\text{TdInv}(td, pk, \pi)$ : 对于任意  $(u = g^{\mathbb{A}}, td = \mathbf{A}^{-1}) \leftarrow \text{SampleNo}(pp)$  和任意  $sk = \mathbf{x} \in SK$ , 由于

$$\pi = H_u(sk) = g^{\mathbf{Ax}}$$

且  $\mathbf{x} \in \{0, 1\}^m$ , 所以, 已知  $\mathbf{A}^{-1}$  和  $\pi$ , 可以计算出  $g^{\mathbf{x}}$ , 从而得到  $\mathbf{x}$ .

<sup>a</sup>根据 [BonehHHO08], 在 DLIN 假设下集合  $L$  与  $X \setminus L$  中的元素是不可区分的.

<sup>b</sup>从集合  $\mathbb{Z}_q^{m \times m}$  中随机选择一个矩阵  $\mathbf{A}$ , 以压倒性地概率  $\mathbf{A}$  是满秩的.



## 6.2 可搜索公钥加密

网络技术的发展使得个人以及企业的数据规模迅速膨胀,海量的数据资源受限于硬件设备而不能妥善保存。随着云存储技术的逐渐成熟,许多大型互联网公司开始搭建大容量的云存储服务设施,为企业及个人的数据存储提供支持。越来越多的用户选择将本地数据上传至云存储服务器以便减轻本地数据的存储和管理开销。由于云存储服务器的提供商并不是一个完全可信的实体,黑客针对云存储服务器的攻击也层出不穷,用户在接受云服务的同时面临数据泄密的风险 [KMAF-ISTAS-2021]。因此,企业及个人的隐私数据不能以明文形式存储在云服务器上,用户在数据上传之前会对本地数据进行加密处理,确保云端数据即使在遭受恶意攻击或不可信云存储服务器主动泄密数据的情况下仍能维护其安全性。传统的数据加密技术能保证数据的安全特性,然而这种技术对于云上数据的保护阻碍数据检索的高效性。云服务器将数据提供者的密文数据存储起来,当数据使用者检索需求消息时,只能将所有密文从云端下载、解密之后才能得到自己需要的信息。这种检索方式的效率极低且容易对网络资源造成巨大的浪费,无法满足数据使用者检索隐私数据时的效率需求。

为了在保证隐私数据安全的同时解决数据检索的效率问题,可搜索加密 (Searchable Encryption, SE) 这一概念应运而生。用户在本地提取明文数据中的关键词信息构造关键字密文索引,云服务器存储这些密文索引,具备检索能力的用户再根据其所要检索的关键词信息生成检索令牌发送至云服务器,云服务器通过其检索匹配算法对其所寻求的密文信息进行搜索并返回结果。如此,便可在不需要解密云端密文的情况下完成对需求数据的高效率检索。根据加密密钥是否可公开,可搜索加密可以划分为可搜索对称加密 (Searchable Symmetric-key Encryption, SSE) [SWP-SP-2000] 和可搜索公钥加密 (Public-key Encryption with Keyword Search, PEKS) [BDOP2004]

### 6.2.1 PEKS 方案

#### 6.2.1.1 形式化定义

可搜索公钥加密技术的产生可以追溯到最初的不可信任邮件服务器路由问题。在该系统中, Email 用户可以利用自己的私钥生成一个关键词  $w$  的检索令牌  $T_w$  发送给 Email 服务器,使得服务器能够返回所有包含关键词  $w$  的邮件,而服务器不会得到密文关键词的其他信息。由于该系统利用 Email 用户的公钥加密关键词, Boneh 等人将其称为可搜索公钥加密。可搜索公钥加密的通用系统模型如图 ??所示。

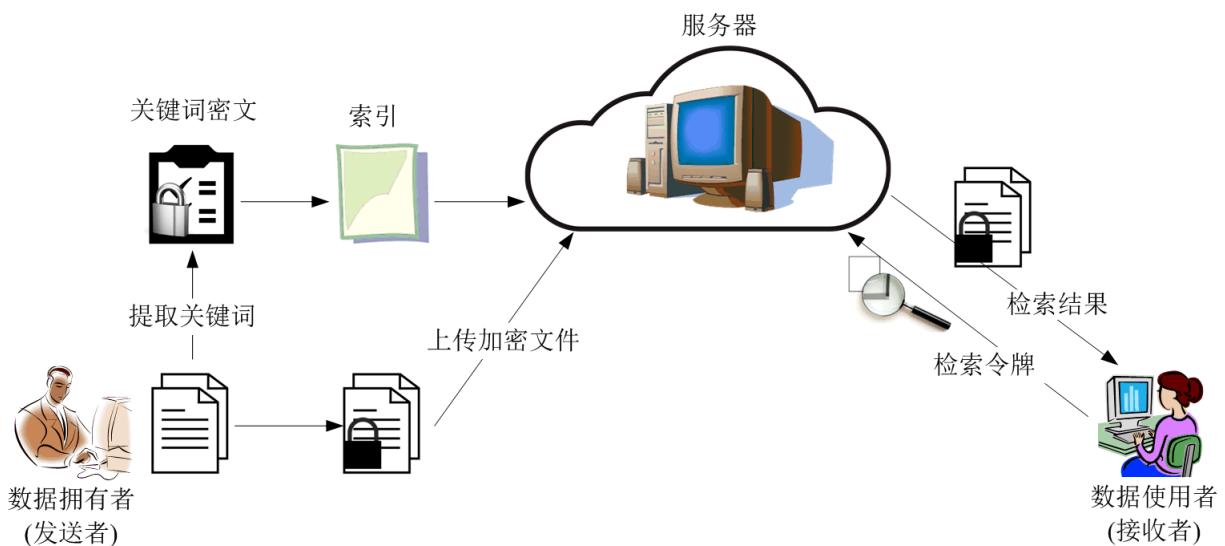


图 6.1: 可搜索公钥加密系统模型

**定义 6.4 (可搜索公钥加密)**

一个可搜索公钥加密方案由以下五个多项式时间算法组成:

- $\text{Setup}(1^\kappa)$ : 系统参数生成算法以安全参数  $1^\kappa$  为输入, 输出系统公开参数  $pp$ , 其中  $pp$  包含了用户的公钥空间  $\mathcal{PK}$ 、私钥空间  $\mathcal{SK}$ 、关键词空间  $\mathcal{W}$ 、密文空间  $\mathcal{C}$  和检索令牌空间  $\mathcal{TK}$  的描述. 类似公钥加密方案, 该算法由可信第三方生成并公开, 系统中的所有用户共享, 所有算法均将  $pp$  作为输入的一部分.
- $\text{KeyGen}(pp)$ : 密钥生成算法以公开参数  $pp$  为输入, 输出一对公/私钥对  $(pk, sk)$ , 其中公钥公开, 私钥秘密保存.
- $\text{Encrypt}(pk, w)$ : 加密算法以公钥  $pk \in \mathcal{PK}$  和关键词  $w \in \mathcal{W}$  为输入, 输出关键词  $w$  的一个可搜索密文  $C_w \in \mathcal{C}$ .
- $\text{TokenGen}(sk, w)$ : 检索令牌生成算法以私钥  $sk \in \mathcal{SK}$  和关键词  $w \in \mathcal{W}$  为输入, 输出关键词  $w$  的一个检索令牌  $T_w$ .
- $\text{Test}(T_{w'}, C_w)$ : 检索算法以关键词  $w'$  的检索令牌  $T_{w'}$  和关键词  $w$  的密文  $C_w$  为输入, 如果  $w = w'$ , 则输出 1, 否则, 输出 0.



**正确性.** 该性质保证了可搜索公钥加密的密文可检索功能, 即利用私钥可以生成关键词的检索令牌并检索出所有包含匹配关键词的密文. 正式地, 对于任意关键词  $w \in \mathcal{W}$ , 有:

$$\Pr[\text{Test}(T_w, \text{Encrypt}(pk, w)) = 1] = 1 - \text{negl}(\kappa) \quad (6.1)$$

公式 ?? 的概率建立在生成系统参数  $pp \leftarrow \text{Setup}(1^\kappa)$ 、公/私钥  $(pk, sk) \leftarrow \text{KeyGen}(pp)$ 、检索令牌  $T_w \leftarrow \text{TokenGen}(sk, w)$  和关键词密文  $C_w \leftarrow \text{Encrypt}(pk, w)$  的随机带上. 如果上述概率严格等于 1, 则称可搜索公钥加密方案满足完美正确性.

**一致性.** 该性质保证了可搜索公钥加密的检索错误率, 即检索令牌仅能与所有包含匹配关键词的密文通过检索算法. 也就是说, 对于任意关键词  $w, w' \in \mathcal{W}$  且  $w \neq w'$ , 有:

$$\Pr[\text{Test}(T_{w'}, \text{Encrypt}(pk, w)) = 1] = \text{negl}(\kappa) \quad (6.2)$$

与公钥加密方案不同, 可搜索公钥加密方案不仅需要满足正确性. 还要满足一致性. Abdalla 等人 [AbdallaBCKLMNPS08] 研究了可搜索公钥加密的完美一致性、统计一致性和计算一致性. 一般地, 我们仅考虑计算一致性. 在许多 PEKS 方案中, 由于正确性蕴含了一致性, 因此也忽略对 PEKS 方案一致性的分析. 下面介绍一致性的两种形式化定义: 弱一致性和强一致性.

**定义 6.5 (一致性)**

可搜索公钥加密方案的一致性通过下面的挑战者与敌手之间的游戏来描述.

- **初始化:**  $\mathcal{CH}$  运行  $\text{Setup}(1^\kappa)$  生成系统参数  $pp$ , 运行算法  $\text{KeyGen}(pp)$  生成用户公钥和私钥  $(pk, sk)$ , 并将系统参数  $pp$  和公钥  $pk$  发送给敌手  $\mathcal{A}$ .
- **阶段 1:**  $\mathcal{A}$  可以适应性地进行检索令牌询问  $w \in \mathcal{W}$ ,  $\mathcal{CH}$  计算检索令牌  $T_w \leftarrow \text{TokenGen}(sk, w)$  并返回给  $\mathcal{A}$ .
- **输出 (弱一致性):**  $\mathcal{A}$  输出两个不同的关键词  $w$  和  $w'$ .  $\mathcal{CH}$  计算  $w$  的 PEKS 密文  $C_w \leftarrow \text{Encrypt}(pk, w)$  和  $w'$  的检索令牌  $T_{w'} \leftarrow \text{TokenGen}(sk, w')$ . 如果  $\text{Test}(T_{w'}, C_w) = 1$ , 则  $\mathcal{A}$  成功.
- **输出 (强一致性):**  $\mathcal{A}$  输出两个不同的关键词  $w$  和  $w'$ , 以及关键词  $w$  的 PEKS 密文  $C_w$ .  $\mathcal{CH}$  计算  $w'$  的检索令牌  $T_{w'} \leftarrow \text{TokenGen}(sk, w')$ . 如果  $\text{Test}(T_{w'}, C_w) = 1$ , 则  $\mathcal{A}$  成功.

在上述游戏中, 令  $\mathcal{A}$  成功的事件记作  $\text{SuccA}$ ,  $\mathcal{A}$  的优势定义为  $\text{Adv}_{\mathcal{A}} = \Pr[\text{SuccA}]$ . 如果对于任意  $t$  时间敌手, 询问最多  $q_w$  次检索令牌, 成功的优势最多为  $\epsilon$ , 则称该 PEKS 方案是  $(t, q_w, \epsilon)$ -弱一致的 (或强一致的).



### 6.2.1.2 安全模型

可搜索公钥加密的语义安全性是为了防止敌手(恶意存储服务器)从关键词密文  $\text{PEKS}(pk, w)$  中得到  $w$  的任何额外信息,除非敌手获取了  $w$  的检索令牌.此外,敌手可以自适应地获取其他关键词  $w'$  的检索令牌  $T_{w'}$ .下面通过两个关键词密文的不可区分性来描述可搜索加密的语义安全性,即自适应选择关键词攻击下的密文不可区分安全性,简称 CI-CKA 安全性.

#### 定义 6.6 (CI-CKA 安全性)

定义可搜索公钥加密方案敌手  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  的优势函数如下:

$$\text{Adv}_{\mathcal{A}, \text{PEKS}}^{\text{CI-CKA}}(\kappa) = \Pr \left[ \beta' = \beta : \begin{array}{l} pp \leftarrow \text{Setup}(1^\kappa); \\ (pk, sk) \leftarrow \text{KeyGen}(pp); \\ (w_0, w_1, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{TokenGen}}(\cdot)}(pp, pk); \\ \beta \xleftarrow{R} \{0, 1\}; \\ C^* \leftarrow \text{Encrypt}(pk, w_\beta); \\ \beta' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{TokenGen}}(\cdot)}(pp, pk, state, C^*) \end{array} \right] - \frac{1}{2},$$

在上述定义中,  $\mathcal{O}_{\text{TokenGen}}(\cdot)$  表示检索令牌预言机,其在接收到关键词  $w$  的询问后,输出  $\text{TokenGen}(sk, w)$ ,但是要求  $w \neq w_0, w_1$ .如果任意的 PPT 敌手  $\mathcal{A}$  在上述定义中的优势函数均为可忽略函数,则称可搜索公钥加密方案是 CI-CKA 安全的.



### 6.2.1.3 PEKS 与 IBE 之间的关系

基于身份加密 (Identity-Based Encryption, 简称 IBE) 是一种能够以用户任意身份(如 Email 地址、姓名、身份证号等)作为加密公钥的新型公钥加密技术,能够简化传统公钥加密技术中的密钥管理复杂性.下面给出IBE的形式化定义,其安全模型如适应性选择身份和明文攻击/密文攻击下的密文不可区分性 (IND-ID-CPA/CCA) 参见 [BF-SIAM-2003].

#### 定义 6.7 (基于身份加密)

一个基于身份加密方案 IBE 包含以下五多项式时间算法:

- $\text{Setup}(1^\kappa)$ : 系统参数生成算法以安全参数  $1^\kappa$  为输入,输出系统公开参数  $pp$ ,其中  $pp$  定义了系统的主公钥空间  $\mathcal{MPK}$ 、主私钥空间  $\mathcal{MSK}$ 、用户身份空间  $\mathcal{ID}$ 、私钥空间  $\mathcal{SK}$ 、消息空间  $\mathcal{M}$  和密文空间  $\mathcal{C}$ .
- $\text{KeyGen}(pp)$ : 主密钥生成算法以公开参数  $pp$  为输入,输出一主公/私钥对  $(mpk, msk)$ ,其中主公钥  $mpk$  公开,主私钥  $msk$  秘密保存.
- $\text{Extract}(msk, id)$ : 用户密钥生成算法以主私钥  $msk$  和用户身份  $id \in \mathcal{ID}$  为输入,输出用户私钥  $sk_{id}$ .
- $\text{Encrypt}(mpk, id, M)$ : 加密算法以主公钥  $mpk$ 、用户身份  $id$  和消息  $M \in \mathcal{M}$  为输入,输出消息  $M$  在身份  $id$  下加密的一个密文  $C \in \mathcal{C}$ .
- $\text{Decrypt}(sk_{id}, C)$ : 解密算法以私钥  $sk_{id}$  和密文  $C$  为输入,输出一个消息  $M'$  或  $\perp$ ,表示解密失败.



PEKS 和 IBE 两种密码原语之间似乎有着天然的联系,可以相互转化.图 ?? 给出了二者参数空间以及算法之间的匹配关系. Boneh 等人指出构造一个安全的 PEKS 方案似乎比构造一个 IBE 方案更困难,这是因为任意一个 PEKS 方案隐含了一个 IBE 方案,见构造 ?? .然而,反之未必成立.

#### 构造 6.3 (从 PEKS 到 IBE 的转化)

假设  $\text{PEKS} = (\text{Setup}, \text{KeyGen}, \text{Encrypt}, \text{TokenGen}, \text{Test})$  是一个可搜索公钥加密方案,下面构造一个消息空间为  $\{0, 1\}$  的基于身份加密方案  $\text{IBE} = (\text{Setup}, \text{KeyGen}, \text{Extract}, \text{Encrypt}, \text{Decrypt})$ .

参数对应关系		算法对应关系	
$\text{PEKS}.\mathcal{PK}$	$\text{IBE}.\mathcal{MPK}$	$\text{PEKS}.\text{Setup}$	$\text{IBE}.\text{Setup}$
$\text{PEKS}.\mathcal{SK}$	$\text{IBE}.\mathcal{MSK}$	$\text{PEKS}.\text{KeyGen}$	$\text{IBE}.\text{KeyGen}$
$\text{PEKS}.\mathcal{T}\mathcal{K}$	$\text{IBE}.\mathcal{ID}$	$\text{PEKS}.\text{TokenGen}$	$\text{IBE}.\text{Extract}$
$\text{PEKS}.\mathcal{W}$	$\text{IBE}.\mathcal{M}$	$\text{PEKS}.\text{Encrypt}$	$\text{IBE}.\text{Encrypt}$
$\text{PEKS}.\mathcal{C}$	$\text{IBE}.\mathcal{C}$	$\text{PEKS}.\text{Test}$	$\text{IBE}.\text{Decrypt}$

图 6.2: PEKS 与 IBE 之间的关系

- $\text{IBE}.\text{Setup}(1^\kappa)$ : 运行  $pp_{\text{PEKS}} \leftarrow \text{PEKS}.\text{Setup}$ , 将 PEKS 的系统参数  $pp_{\text{PEKS}}$  作为 IBE 的系统参数  $pp$ .
- $\text{KeyGen}(pp)$ : 运行  $(pk_{\text{PEKSS}}, sk_{\text{PEKS}}) \leftarrow \text{PEKS}.\text{KeyGen}(pp_{\text{PEKS}})$ , 将 PEKS 的用户公钥  $pk_{\text{PEKSS}}$  和私钥  $sk_{\text{PEKSS}}$  分别作为 IBE 的主公钥  $mpk$  和主私钥  $msk$ .
- $\text{Extract}(msk, id)$ : 对于任意用户身份  $id \in \{0, 1\}^*$ , 运行  $T_b \leftarrow \text{PEKS}.\text{TokenGen}(sk_{\text{PEKS}}, id||b)$  两次, 其中  $b = 0, 1$ . 将检索令牌  $T_0$  和  $T_1$  作为用户  $id$  的私钥, 即  $sk_{id} = (T_0, T_1)$ .
- $\text{Encrypt}(mpk, id, M)$ : 对于消息  $M \in \{0, 1\}$ , 运行  $C_{\text{PEKS}} \leftarrow \text{PEKS}.\text{Encrypt}(pk_{\text{PEKS}}, id||M)$ . 将 PEKS 的密文  $C_{\text{PEKS}}$  作为 IBE 密文  $C$ .
- $\text{Decrypt}(sk_{id}, C)$ : 输入用户私钥  $sk_{id} = (T_0, T_1)$  和密文  $C = C_{\text{PEKS}}$ , 如果  $\text{PEKS}.\text{Test}(T_0, C_{\text{PEKS}}) = 1$ , 则输出 0; 如果  $\text{PEKS}.\text{Test}(T_1, C_{\text{PEKS}}) = 1$ , 则输出 1.



上述构造方案的安全性由下面的引理保证:

### 引理 6.2

如果 PEKS 是一个满足自适应选择关键词攻击下密文不可区分 (CI-CKA) 安全性的可搜索公钥加密方案, 则构造方案 ?? 是一个满足自适应选择身份和密文攻击下密文不区分 (IND-ID-CCA) 安全性的基于身份加密方案.



**笔记** 在不考虑安全性的情况下, 利用一个 IBE 方案按照图 ?? 的方式是可以构造一个满足正确性的 PEKS 方案. 将一个固定消息  $0^{|\mathcal{M}|}$  的 IBE 密文  $\text{IBE}.\text{Encrypt}(mpk, w, 0^{|\mathcal{M}|})$  作为关键词  $w$  的 PEKS 密文. 检索匹配算法只需要利用  $w$  对应的标识密钥解密该密文, 如果解密出的结果与固定消息  $0^{|\mathcal{M}|}$  一致, 则检索成功. 然而, IBE 的加密算法并不要求身份标识是保密的, 也就是说 IBE 密文可能会泄漏身份的信息. 此外, IBE 解密算法不一定满足一致性, 利用不同身份标识的用户私钥可能解密出正确的结果. 2005 年, Abdalla 等人 [Abdalla2005] 指出, 解决这两个问题可以选择一个匿名的基于身份加密方案并将固定消息  $0^{|\mathcal{M}|}$  替换为随机消息  $R$ , 将  $\text{IBE}.\text{Encrypt}(mpk, w, R)$  和  $R$  同时作为 PEKS 的密文. 在匿名的基于身份加密方案中, 由于密文不会泄漏身份的信息, 故 PEKS 密文不会泄漏关键词的信息. 又由于加密的是随机消息, 一个不匹配的检索令牌 (用户的标识密钥) 解密出的消息与  $R$  一致的可能性是可以忽略的.



#### 6.2.1.4 方案构造

下面介绍 Boneh 等人在 2004 年提出的一个可搜索公钥加密方案, 记作 BDOP-PEKS 方案.

### 构造 6.4 (BDOP-PEKS 方案)

- $\text{Setup}(1^\kappa)$ : 选择一个双线性配对群  $(e, g, q, \mathbb{G}, \mathbb{G}_T)$ , 其中  $\mathbb{G}$  和  $\mathbb{G}_T$  是两个阶为素数  $q$  的群,  $g$  是群  $\mathbb{G}$  的一个生成元,  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  是两个群之间的双线性映射. 此外, 选择两个哈希函数  $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}$  和  $H_2 : \mathbb{G}_T \rightarrow \{0, 1\}^{\log q}$ . 输出系统参数  $pp = (e, g, q, \mathbb{G}, \mathbb{G}_T, H_1, H_2)$ .
- $\text{KeyGen}(pp)$ : 随机选择  $\alpha \xleftarrow{R} \mathbb{Z}_q$ , 计算  $h = g^\alpha$ , 输出公钥  $pk = h$  和私钥  $sk = \alpha$ .
- $\text{Encrypt}(pk, w)$ : 对于任意关键词  $w \in \{0, 1\}^*$ , 随机选择  $r \xleftarrow{R} \mathbb{Z}_q$ , 计算  $t = e(H_1(w), h^r)$ , 输出密文  $C = (g^r, H_2(t))$ .

- $\text{TokenGen}(sk, w)$ : 对于任意关键词  $w \in \{0, 1\}^*$ , 输出检索令牌  $T_w = H_1(w)^\alpha$ .
- $\text{Test}(T_w, C)$ : 对于密文  $C = (A, B)$  和检索令牌  $T_w$ , 判断等式  $H_2(e(T_w, A)) = B$  是否成立. 如果成立, 则输出 1, 否则输出 0.



**笔记** 事实上, BDOP-PEKS 方案是在 Boneh 和 Franklin 的基于身份加密方案(记作 BF-IBE 方案) [BF-SIAM-2003] 基础上设计的. 由于 BF-IBE 方案满足身份匿名性, 利用前面讨论的从匿名 IBE 到 PEKS 的转化思路, 设计出 BDOP-PEKS 方案是比较自然的.

BDOP-PEKS 方案的安全性基于 BDH (Bilinear Diffie-Hellman) 问题的困难性. 双线性配对上的 BDH 问题描述如下: 给定一个双线性配对群  $(e, g, q, \mathbb{G}, \mathbb{G}_T)$ , 输入  $g, g^a, g^b, g^c \in \mathbb{G}$ , 计算  $e(g, g)^{abc}$ . BDOP-PEKS 方案的安全性由下面的定理 ?? 保证.

### 定理 6.2

假设双线性配对上的 BDH 问题是困难的, 则在随机预言机模型下 BDOP-PEKS 方案是 CI-CKA 安全的.



**证明** 以归约的方式证明上述定理, 具体过程如下:

令  $\mathcal{A}$  是一个以  $\epsilon$  优势攻击 BDOP-PEKS 方案 CI-CKA 安全性的敌手,  $g, u_1 = g^\alpha, u_2 = g^\beta, u_3 = g^\gamma \in \mathbb{G}$  是双线性配对  $(e, g, q, \mathbb{G}, \mathbb{G}_T)$  上的一个 BDH 问题实例. 若  $\mathcal{A}$  成功的概率不可忽略, 归约证明的目标是构造一个算法  $\mathcal{B}$ , 以 BDH 问题实例为输入, 借助敌手  $\mathcal{A}$  的能力以不可忽略的概率解决该 BDH 问题实例, 从而推出矛盾. 假设  $\mathcal{A}$  询问哈希函数  $H_2$  的次数最多为  $q_{H_2}$ , 询问检索令牌的次数最多为  $q_T$ , 则  $\mathcal{B}$  按如下方式模拟模拟  $\mathcal{A}$  在游戏中的视图环境, 即原始游戏中挑战者的行为.

初始化:  $\mathcal{B}$  根据双线性配对的参数  $(e, g, q, \mathbb{G}, \mathbb{G}_T)$  选择两个哈希函数  $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}$  和  $H_2 : \mathbb{G}_T \rightarrow \{0, 1\}^{\log q}$ , 并令系统参数为  $pp = (e, g, q, \mathbb{G}, \mathbb{G}_T, H_1, H_2)$ , 用户的公钥为  $pk = u_1$ .  $\mathcal{B}$  将系统参数  $pp$  和公钥  $pk$  发送给  $\mathcal{A}$ . 显而易见, 这里隐含地选择了用户私钥为挑战 BDH 问题实例中的  $\alpha$ . 尽管  $\mathcal{B}$  不知道  $\alpha$  的具体值(也不能知道该秘密, 否则就没法进行归约证明了), 但是  $\alpha$  是随机选择的, 所以  $\mathcal{B}$  模拟的系统参数和用户公钥与实际游戏环境是一致的.

询问阶段 1: 在真实游戏中, 挑战者只需要回答敌手的检索令牌询问和挑战询问, 而任意元素的哈希值计算是公开的. 为了使算法  $\mathcal{B}$  能够模拟  $\mathcal{A}$  的视图环境, 需要将哈希函数  $H_1$  和  $H_2$  看作随机预言机, 即在随机预言机模型中模拟敌手的各类预言机查询结果. 具体如下:

- $H_1$  和  $H_2$  询问: 在任何时候, 敌手  $\mathcal{A}$  都可以询问随机预言机  $H_1$  或  $H_2$ . 对于  $H_1$  哈希询问,  $\mathcal{B}$  维护一个形如  $\langle w_j, h_j, a_j, c_j \rangle$  且初始化为空的  $H_1$ -列表. 当  $\mathcal{A}$  询问  $w_i \in \{0, 1\}^*$  的  $H_1$  哈希值时, 算法  $\mathcal{B}$  按如下方式进行回答:

1. 如果  $w_i$  已经在  $H_1$ -列表的元素  $\langle w_j, h_j, a_j, c_j \rangle$  中, 则算法  $\mathcal{B}$  返回  $H_1(w_i) = h_i \in \mathbb{G}$ .
2. 否则,  $\mathcal{B}$  随机选择一比特  $c_i \in \{0, 1\}$ , 使得  $\Pr[c_i = 0] = 1/(q_T + 1)$ .
3.  $\mathcal{B}$  随机选择  $a_i \in \mathbb{Z}_q$ , 并计算

$$h_i = \begin{cases} u_2 \cdot g^{a_i}, & \text{If } c_i = 0 \\ g^{a_i}, & \text{If } c_i = 1 \end{cases}$$

4.  $\mathcal{B}$  将元素  $\langle w_j, h_j, a_j, c_j \rangle$  添加到  $H_1$ -列表中并将哈希值  $H_1(w_i) = h_i$  返回给  $\mathcal{A}$ . 显而易见, 不论随机比特  $c_i$  取值如何, 哈希值  $h_i$  都是群  $\mathbb{G}$  中的一个随机元素且与  $\mathcal{A}$  当前的视图独立无关. 这与  $H_1$  是一个随机预言机的假设一致.

类似地,  $\mathcal{A}$  可以在任何时候询问  $H_2$  的哈希值. 当  $\mathcal{B}$  维护一个形如  $\langle t_i, V_i \rangle$  且初始化为空的  $H_2$ -列表. 当  $\mathcal{A}$  询问  $t_i$  的  $H_2$  哈希值时, 如果  $t_i$  在  $H_2$ -列表的元素  $\langle t_i, V_i \rangle$  中, 则  $\mathcal{B}$  返回  $V_i$ ; 否则,  $\mathcal{B}$  随机选择  $V_i \in \{0, 1\}^{\log q}$ , 将  $\langle t_i, V_i \rangle$  添加到  $H_2$ -列表中, 并将哈希值  $H_2(t_i) = V_i$  返回给  $\mathcal{A}$ .

- 检索令牌询问: 当  $\mathcal{A}$  询问关键词  $w_i$  的检索令牌时,  $\mathcal{B}$  按下面的方式进行回答:

1.  $\mathcal{B}$  通过  $H_1$  哈希询问方式获取  $w_i$  的哈希值  $h_i \in \mathbb{G}$ , 即  $H_1(w_i) = h_i$ . 令  $\langle w_j, h_j, a_j, c_j \rangle$  是  $H_1$ -列表中的相应元素. 如果  $c_i = 0$ , 则  $\mathcal{B}$  模拟失败并终止游戏.
2. 否则,  $c_i = 1$  且  $h_i = g^{a_i} \in \mathbb{G}$ .  $\mathcal{B}$  计算  $T_i = u_1^{a_i}$ . 注意到  $H_1(w_i) = g^{a_i}$  且  $u_1 = g^\alpha$ , 所以  $T_i = H_1(w_i)^\alpha$  是  $w_i$  的一个正确的检索令牌.  $\mathcal{B}$  将  $T_i$  发送给  $\mathcal{A}$ .

挑战: 当阶段 1 询问结束时,  $\mathcal{A}$  选择两个挑战关键词  $w_0, w_1 \in \{0, 1\}^*$  发送给  $\mathcal{B}$ . 算法  $\mathcal{B}$  按下面的方式生成挑战 PEKS 密文:

1.  $\mathcal{B}$  通过两次  $H_1$  哈希询问获取  $w_0$  和  $w_1$  的哈希值  $h_0, h_1 \in \mathbb{G}$  且满足  $H_1(w_0) = h_0$  和  $H_1(w_1) = h_1$ . 假设  $\langle w_0, h_0, a_0, c_0 \rangle$  和  $\langle w_1, h_1, a_1, c_1 \rangle$  分别是相应的  $H_1$ -列表中的元素. 如果  $c_0 = 1$  且  $c_1 = 1$ , 则  $\mathcal{B}$  模拟失败并终止游戏.
2. 否则,  $c_0$  和  $c_1$  中至少有一个等于 0.  $\mathcal{B}$  随机选择  $b \in \{0, 1\}$  使得  $c_b = 0$ .
3. 算法  $\mathcal{B}$  随机选择  $J \in \{0, 1\}^{\log q}$  并将  $C^* = (u_3, J)$  作为挑战密文返回给  $\mathcal{A}$ .

值得注意的是, 挑战密文隐含地定义了  $H_2(e(H_1(w_b), u_1^\gamma)) = J$ . 由此可知

$$J = H_2(e(H_1(w_b), u_1^\gamma)) = H_2(e(u_2 g^{a_b}, g^{\alpha\gamma})) = H_2(e(g, g)^{\alpha\gamma}(\beta + a_b))$$

是  $w_b$  的一个合法密文.

询问阶段 2:  $\mathcal{A}$  可以继续进行哈希询问和关键词的检索令牌询问, 但是不允许询问挑战关键词的检索令牌.

输出: 最终,  $\mathcal{A}$  将输出一猜测比特  $b' \in \{0, 1\}$ .  $\mathcal{B}$  从  $H_2$ -列表中随机选择一个元素  $\langle t, v \rangle$ , 计算  $T = t/e(u_1, u_3)^{a_b}$  作为 BDH 问题解  $e(g, g)^{\alpha\beta\gamma}$  的一个猜测结果, 其中  $a_b$  是挑战阶段使用的元素. 如果  $\mathcal{A}$  询问过  $H_2$  的哈希值  $H_2(e(H_1(w_0), u_1^\gamma))$  或  $H_2(e(H_1(w_1), u_1^\gamma))$ , 那么,  $H_2$ -列表以  $1/2$  的概率包含一个元素  $\langle t, v \rangle$ , 其中  $t = H_2(e(H_1(w_b), u_1^\gamma)) = H_2(e(g, g)^{\alpha\gamma}(\beta + a_b))$ . 因此,  $T = t/e(u_1, u_3)^{a_b} = e(g, g)^{\alpha\beta\gamma}$ .

至此, 完成了算法  $\mathcal{B}$  的描述. 下面的主要任务是分析  $\mathcal{B}$  正确输出 BDH 问题实例解  $e(g, g)^{\alpha\beta\gamma}$  的概率  $\epsilon'$ . 首先, 我们分析  $\mathcal{B}$  在模拟游戏中不终止的概率. 定义以下两个事件:

- $E_1$ : 表示事件  $\mathcal{B}$  在回答  $\mathcal{A}$  的检索令牌询问时不终止游戏.
- $E_2$ : 表示事件  $\mathcal{B}$  在挑战阶段不终止游戏.

上述两个事件的概率下界由下面的断言保证.

### 断言 6.1

$\mathcal{B}$  在回答  $\mathcal{A}$  的所有检索令牌查询结果时不终止游戏的概率至少为  $1/e$ , 即  $\Pr[E_1] \geq 1/e$ .



**证明** 假设  $w_i$  是  $\mathcal{A}$  的第  $i$  次询问检索令牌的关键词. 在  $i$  次询问中,  $\mathcal{B}$  终止游戏的条件是  $w_i$  相应的  $H_1$ -列表元素  $\langle w_i, h_i, a_i, c_i \rangle$  中,  $c_i = 0$ . 尽管哈希值  $H_1(w_i)$  的生成方式与  $c_i$  有关, 但是  $H_1(w_i)$  的分布与  $c_i = 0$  还是  $c_i = 1$  不相关. 根据  $c_i$  的分布, 可知  $\mathcal{B}$  在回答本次询问过程中终止游戏的概率最多为  $\Pr[c_i = 0] = 1/(q_T + 1)$ . 由于  $\mathcal{A}$  进行检索令牌查询的次数最多为  $q_T$ , 所以  $\mathcal{B}$  在所有检索令牌询问中都不终止游戏的概率至少为  $(1 - 1/(q_T + 1))^{q_T} \geq 1/e$ .

断言 ?? 证毕! □

### 断言 6.2

$\mathcal{B}$  在挑战密文生成阶段时不终止游戏的概率至少为  $1/q_T$ , 即  $\Pr[E_2] \geq 1/q_T$ .



**证明** 在挑战阶段,  $\mathcal{B}$  终止游戏的条件是挑战关键词  $w_0$  和  $w_1$  相应的  $H_1$ -列表元素  $\langle w_0, h_0, a_0, c_0 \rangle$  和  $\langle w_1, h_1, a_1, c_1 \rangle$  中,  $c_0 = c_1 = 1$ . 由于  $\mathcal{A}$  不允许询问  $w_0$  和  $w_1$  的检索令牌, 所以  $c_0$  和  $c_1$  的值独立于  $\mathcal{A}$  的当前视图, 且  $c_0$  和  $c_1$  的取值是相互独立的. 根据  $c_i$  的分布, 可知  $\Pr[c_0 = 1] = \Pr[c_1 = 1] = 1 - 1/(q_T + 1)$ , 由此可知  $\Pr[c_0 = c_1 = 1] = (1 - 1/(q_T + 1))^2 \leq 1 - 1/q_T$ . 所以  $\mathcal{B}$  在挑战密文生成阶段不终止游戏的概率至少为  $1/q_T$ .

断言 ?? 证毕! □

由于  $\mathcal{A}$  是不允许询问挑战关键词  $w_0$  和  $w_1$  的检索令牌, 所以两个事件  $E_1$  和  $E_2$  是相互独立的. 因此,  $\Pr[E_1 \wedge E_2] \geq 1/(eq_T)$ .

最后, 我们分析  $\mathcal{A}$  询问哈希值  $H_2(e(H_1(w_b, u_1^\gamma)))$  的概率下界, 由以下断言保证,

### 断言 6.3

假设在真实游戏中, 给定  $\mathcal{A}$  系统参数  $pp$ , 公钥  $pk = u_1$ . 当询问挑战关键词  $w_0$  和  $w_1$  的密文时, 返回给  $\mathcal{A}$  的结果是  $C^* = (u_3 = \gamma, J)$ . 则  $\mathcal{A}$  在真实游戏中询问  $H_2$  的哈希值  $H_2(e(H_1(w_0, u_1^\gamma)))$  或  $H_2(e(H_1(w_1, u_1^\gamma)))$  的概率至少为  $2\epsilon$ .



**证明** 令  $E_3$  表示事件 “ $\mathcal{A}$  在真实游戏中询问了哈希值  $H_2(e(H_1(w_0, u_1^\gamma)))$  或  $H_2(e(H_1(w_1, u_1^\gamma)))$ ”. 显然, 若事件  $E_3$  未发生, 在随机预言机模型下, 挑战密文中的元素  $J$  完全独立于  $\mathcal{A}$  的当前视图, 从而挑战阶段  $b \in \{0, 1\}$  的取值与  $\mathcal{A}$  的视图独立. 所以,  $\Pr[b = b' | \neg E_3] = 1/2$ . 根据假设, 敌手  $\mathcal{A}$  在真实游戏中成功的优势至少为  $|\Pr[b = b'] - 1/2| \geq \epsilon$ . 又由于

$$\begin{aligned} \Pr[b = b'] &= \Pr[b = b' | E_3] \Pr[E_3] + \Pr[b = b' | \neg E_3] \Pr[\neg E_3] \\ &\leq \Pr[E_3] + \Pr[b = b' | \neg E_3] \Pr[\neg E_3] \\ &= \Pr[E_3] + \frac{1}{2} \Pr[\neg E_3] \\ &= \frac{1}{2} + \frac{1}{2} \Pr[E_3] \end{aligned}$$

$$\begin{aligned} \Pr[b = b'] &= \Pr[b = b' | E_3] \Pr[E_3] + \Pr[b = b' | \neg E_3] \Pr[\neg E_3] \\ &\geq \Pr[b = b' | \neg E_3] \Pr[\neg E_3] \\ &= \frac{1}{2} \Pr[\neg E_3] \\ &= \frac{1}{2} - \frac{1}{2} \Pr[E_3] \end{aligned}$$

所以  $\epsilon \leq |\Pr[b = b'] - 1/2| \leq \frac{1}{2} \Pr[E_3]$ . 由此可得  $\Pr[E_3] \geq 2\epsilon$ .

断言 ?? 证毕!

□

若  $\mathcal{B}$  不终止游戏, 根据算法  $\mathcal{B}$  的描述, 则  $\mathcal{B}$  模拟的游戏环境与真实游戏环境是完全一样的. 根据断言 ??, 至模拟游戏结束,  $\mathcal{A}$  询问哈希值  $H_2(e(H_1(w_0, u_1^\gamma)))$  或  $H_2(e(H_1(w_1, u_1^\gamma)))$  的概率至少为  $2\epsilon$ . 由于  $b$  是独立于  $w_0$  和  $w_1$  随机选取的, 所以  $\mathcal{A}$  询问哈希值  $H_2(e(H_1(w_b, u_1^\gamma)))$  的概率至少为  $\epsilon$ . 因此, 以至少  $\epsilon$  的概率,  $H_2$ -列表中存在形如  $\langle e(H_1(w_b, u_1^\gamma), \cdot) \rangle$  的元素. 如果  $\mathcal{B}$  不终止游戏, 则  $\mathcal{B}$  正确选取到元素  $\langle e(H_1(w_b, u_1^\gamma), \cdot) \rangle$  的概率至少为  $\epsilon/q_T$ . 结合  $\mathcal{B}$  不终止游戏的概率至少为  $1/(eq_T)$ , 所以  $\mathcal{B}$  成功解决 BDH 问题的概率至少为  $\epsilon/(eq_T q_{H_2})$ .

定理 ?? 证毕!

□

## 6.2.2 PKE-PEKS 方案

整合的公钥加密和可搜索公钥加密 (Integrated PKE and PEKS, 简称 PKE-PEKS) 解决了 PEKS 方案缺少对消息加密的不足之处. 本节重点介绍 PKE-PEKS 的形式化定义、安全模型及通用构造方法.

### 6.2.2.1 形式化定义

参考文献 [BSS2006; ZI2007; Chen-DCC-2016], 我们回顾 PKE-PEKS 的定义.

**定义 6.8**

一个 PKE-PEKS 方案包含以下六个 PPT 算法:

- $\text{Setup}(1^\kappa)$ : 系统参数生成算法以安全参数  $1^\kappa$  为输入, 输出系统公开参数  $pp$ . 公开参数定义了消息空间  $\mathcal{M}$ 、关键词空间  $\mathcal{W}$  和密文空间  $\mathcal{C}$ . 该算法由可信第三方生成并公开, 系统中的所有用户共享, 所有算法均将  $pp$  作为输入的一部分.
- $\text{KeyGen}(pp)$ : 密钥生成算法以公开参数  $pp$  为输入, 输出一对公/私钥对  $(pk, sk)$ , 其中公钥公开, 私钥秘密保存.
- $\text{Encrypt}(pk, m, w)$ : 加密算法以公钥  $pk$ 、消息  $m \in \mathcal{M}$  和关键词  $w \in \mathcal{W}$  为输入, 输出一个 PKE-PEKS 密文  $C$ .
- $\text{Decrypt}(sk, C)$ : 解密算法以私钥  $sk$  和一个 PKE-PEKS 密文  $C \in \mathcal{C}$  为输入, 输出一个明文  $m \in \mathcal{M}$  或者一个特殊符号  $\perp$  表示  $C$  是一个非法密文.
- $\text{TokenGen}(sk, w)$ : 检索令牌生成算法以私钥  $sk$  和关键词  $w \in \mathcal{W}$  为输入, 输出关键词  $w$  的一个检索令牌  $T_w$ .
- $\text{Test}(T_{w'}, C)$ : 检索算法以关键词  $w'$  的检索令牌  $T_{w'}$  和关键词  $w$  的密文  $C$  为输入, 如果  $w = w'$ , 则输出 1; 否则, 输出 0.



**正确性.** 对于任意系统参数  $pp \leftarrow \text{Setup}(1^\kappa)$ , 任意公/私钥对  $(pk, sk) \leftarrow \text{KeyGen}(pp)$ , 任意消息  $m \in \mathcal{M}$ , 任意关键词  $w \in \mathcal{W}$  和任意检索令牌  $T_w \leftarrow \text{TokenGen}(sk, w)$ , 我们有

$$\text{Decrypt}(sk, \text{Encrypt}(pk, m, w)) = m \text{ 且 } \text{Test}(T_w, \text{Encrypt}(pk, m, w)) = 1.$$

**一致性.** 除了正确性, 类似 PEKS, 我们需要考虑 PKE-PEKS 的一致性. 通俗地讲, 如果对于任意  $m \in \mathcal{M}$  和  $w \neq w'$ , 我们有  $\text{Test}(T_{w'}, \text{Encrypt}(pk, m, w)) = 0$ , 则称 PKE-PEKS 方案满足一致性. PKE-PEKS 的一致性的形式化定义可以参考 PEKS 的一致性来定义.

### 6.2.2.2 安全模型

一个安全的 PKE-PEKS 方案不仅需要保障数据隐私还要保障关键词隐私, 分别通过下面两个安全模型来刻画.

**定义 6.9 (DT-Priv 安全性)**

定义 PKE-PEKS 方案的数据隐私敌手  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  的优势函数如下:

$$\text{Adv}_{\mathcal{A}, \text{PKE-PEKS}}^{\text{DT-Priv}}(\kappa) = \Pr \left[ \beta' = \beta : \begin{array}{l} pp \leftarrow \text{Setup}(1^\kappa); \\ (pk, sk) \leftarrow \text{KeyGen}(pp); \\ (M_0^*, M_1^*, w^*, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{Decrypt}}(\cdot), \mathcal{O}_{\text{TokenGen}}(\cdot), \mathcal{O}_{\text{Test}}(\cdot)}(pp, pk); \\ \beta \xleftarrow{\text{R}} \{0, 1\}; \\ C^* \leftarrow \text{Encrypt}(pk, M_\beta^*, w^*); \\ \beta' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{Decrypt}}(\cdot), \mathcal{O}_{\text{TokenGen}}(\cdot), \mathcal{O}_{\text{Test}}(\cdot)}(pp, pk, state, C^*) \end{array} \right] - \frac{1}{2},$$

其中, 谎言机的定义分别如下:

- $\mathcal{O}_{\text{Decrypt}}(\cdot)$  表示解密谎言机, 其在接收到密文  $C$  的询问后, 输出  $M \leftarrow \text{Decrypt}(sk, C)$ .
- $\mathcal{O}_{\text{TokenGen}}(\cdot)$  表示检索令牌谎言机, 其在接收到关键词  $w$  的询问后, 输出  $T_w \leftarrow \text{TokenGen}(sk, w)$ .
- $\mathcal{O}_{\text{Test}}(\cdot)$  表示检索测试谎言机, 其在接收到关键词  $w$  和密文  $C$  的询问后, 输出  $\text{Test}(T_w, C)$ , 其中  $T_w \leftarrow \text{TokenGen}(sk, w)$ .

在猜测阶段, 敌手不能访问挑战密文  $C^*$  的解密询问, 而对于检索令牌询问和检索测试询问没有任何限制. 在上述定义中, 对于任意 PPT 敌手  $\mathcal{A}$ , 若优势函数均为可忽略函数, 则称 PKE-PEKS 方案满足数据隐私安

全性, 简称 DT-Priv 安全性.



### 定义 6.10 (KW-Priv 安全性)

定义 PKE-PEKS 方案的关键词隐私敌手  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  的优势函数如下:

$$\text{Adv}_{\mathcal{A}, \text{PKE-PEKS}}^{\text{KW-Priv}}(\kappa) = \Pr_{\beta' = \beta : \begin{array}{l} \beta \xleftarrow{\text{R}} \{0, 1\}; \\ C^* \leftarrow \text{Encrypt}(pk, M^*, w_{\beta}^*); \\ \beta' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{Decrypt}}(\cdot), \mathcal{O}_{\text{TokenGen}}(\cdot), \mathcal{O}_{\text{Test}}(\cdot)}(pp, pk, state, C^*) \end{array}} \left[ \begin{array}{l} pp \leftarrow \text{Setup}(1^\kappa); \\ (pk, sk) \leftarrow \text{KeyGen}(pp); \\ (w_0^*, w_1^*, M^*, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{Decrypt}}(\cdot), \mathcal{O}_{\text{TokenGen}}(\cdot), \mathcal{O}_{\text{Test}}(\cdot)}(pp, pk); \end{array} \right] - \frac{1}{2},$$

其中, 谎言机的定义同 DT-Priv 安全性中的定义. 在任意阶段, 敌手都不能访问挑战关键词  $w_0^*$  和  $w_1^*$  的检索令牌谎言机, 也不能访问  $(C^*, w_0^*)$  和  $(C^*, w_1^*)$  的检索测试谎言机. 而对于解密询问没有任何限制. 在上述定义中, 对于任意 PPT 敌手  $\mathcal{A}$ , 若优势函数均为可忽略函数, 则称 PKE-PEKS 方案满足关键词隐私安全性, 简称 KW-Priv 安全性.



### 定义 6.11 (Jointly CCA 安全性)

对于任意 PPT 敌手  $\mathcal{A}$ , 如果两个游戏中的优势函数  $\text{Adv}_{\mathcal{A}, \text{PKE-PEKS}}^{\text{DT-Priv}}(\kappa)$  和  $\text{Adv}_{\mathcal{A}, \text{PKE-PEKS}}^{\text{KW-Priv}}(\kappa)$  都是可忽略的, 则称 PKE-PEKS 满足联合选择密文攻击安全性 (Jointly CCA 安全性).



### 6.2.2.3 方案构造

本节介绍两种通用构造: 基于 HIBE 的构造和基于 IBE 的构造.

**基于 HIBE 的 PKE-PKES 方案.** 基于一个两层的 HIBE 方案 HIBE=(Setup, Extract, Derive, Encrypt, Decrypt) 外加一个一次签名方案 OTS=(Setup, KeyGen, Sign, Verify), 可以构造一个 CCA 安全的 PKE-PEKS 方案. 不失一般性, 这里假设 HIBE 的消息空间是  $\{0, 1\}^n$ , 每一层的身份空间是  $\{0, 1\}^*$ , 签名方案的验证密钥空间是  $\{0, 1\}^n$ , 其中  $n = n(\kappa)$ .

### 构造 6.5 (基于 HIBE 的 PKE-PEKS 方案)

- **Setup( $\kappa$ ):** 运行  $pp_{\text{HIBE}} \leftarrow \text{HIBE}.\text{Setup}(\kappa)$  和  $pp_{\text{OTS}} \leftarrow \text{OTS}.\text{Setup}(\kappa)$ , 输出系统参数  $pp = (pp_{\text{HIBE}}, pp_{\text{OTS}})$ .
- **KeyGen( $\kappa$ ):** 运行  $(mpk, msk) \leftarrow \text{HIBE}.\text{KeyGen}(pp_{\text{HIBE}})$ , 输出公钥和私钥  $(pk, sk) \leftarrow (mpk, msk)$ .
- **Encrypt( $pk, m, w$ ):** 输入公钥  $pk$ 、消息  $m$  和关键词  $w$ , 执行以下步骤:
  1. 运行  $(vk, sigk) \leftarrow \text{OTS}.\text{KeyGen}(pp_{\text{OTS}})$ .
  2. 用层次一身份  $0||vk$  加密消息  $m$ ,  $c \leftarrow \text{HIBE}.\text{Encrypt}(pk, 0||vk, m)$ .
  3. 用层次二身份  $(1||w, vk)$  加密验证公钥  $vk$ ,  $s \leftarrow \text{HIBE}.\text{Encrypt}(pk, (1||w, vk), vk)$ .
  4. 计算  $\sigma \leftarrow \text{OTS}.\text{Sign}(sigk, c||s)$ , 输出密文  $u = (vk, c, s, \sigma)$ .
- **Decrypt( $sk, u$ ):** 输入私钥  $sk$  和密文  $u$ , 执行以下步骤:
  1. 将密文  $u$  拆分为  $(vk, c, s, \sigma)$ .
  2. 如果  $\text{OTS}.\text{Verify}(vk, c||s, \sigma) = 1$ , 计算  $dk \leftarrow \text{HIBE}.\text{Extract}(sk, 0||vk)$ ,  
输出  $m \leftarrow \text{HIBE}.\text{Decrypt}(dk, c)$ .  
否则输出  $\perp$ .
- **TokenGen( $sk, w$ ):** 输入私钥  $sk$  和关键词  $w$ , 计算  $t_1 \leftarrow \text{HIBE}.\text{Extract}(sk, 1||w)$  和  $t_2 \leftarrow w$ , 输出检索

令牌  $T_w = (t_1, t_2)$ .

- **Test**( $T_w, u$ ): 输入检索令牌  $T_w$  和密文  $u$ , 执行以下步骤:
  1. 将  $T_w$  拆分为  $(t_1, t_2)$ ,  $u$  拆分为  $(vk, c, s, \sigma)$ .
  2. 如果  $\text{OTS.Verify}(vk, c||s, \sigma) = 1$ , 计算  $dk \leftarrow \text{HIBE.Derive}(t_1, (1||t_2, vk))$ ,  
如果  $vk = \text{HIBE.Decrypt}(dk, s)$ , 则输出 1; 否则, 输出 0.
  - 否则, 输出 0.



**正确性.** 方案的正确性可由基于的 HIBE 的正确性直接验证. 方案的安全性由下面的定理保证.

### 定理 6.3

若 HIBE 方案满足 IND-HIBE-CPA 安全性且第一层次身份满足 ANO-HIBE-CPA 匿名性, 一次签名 OTS 满足 sEUF-CMA 安全性, 则构造方案 ?? 是一个 jointly CCA-安全的 PKE-PEKS 方案.



**证明** 如果  $\text{OTS.Verify}(vk, c||s, \sigma) = 1$ , 则称  $u = (vk, c, s, \sigma)$  是一个合法的 PKE-PEKS 密文. 令  $u^* = (vk^*, c^*, s^*, \sigma^*)$  表示敌手  $\mathcal{A}$  收到的挑战 PKE-PEKS 密文. 定理 ?? 的证明由以下两个引理组成.

### 引理 6.3

假设 HIBE 是  $(t_1, q_{k_1}, \epsilon_1)$ -IND-HIBE-CPA 安全的, OTS 是  $(t_3, 1, \epsilon_3)$ -sEUF-CMA 安全的. 则 PKE-PEKS 方案具有  $(t, q_w, q_t, q_d, \epsilon)$ -DT-Priv 安全性, 其中

$$\begin{aligned} \epsilon &\leq \epsilon_1 + \frac{1}{2}\epsilon_3, q_w + q_t + q_d \leq q_{k_1}, \\ t &\leq \min\{t_3 - t_b - q_w t_k - (q_t + q_d)(t_k + t_v + t_d) - 2t_e, t_1 - t_g - (q_t + q_d)(t_v + t_d) - t_e - t_s\}. \end{aligned}$$



**证明** 假设  $\mathcal{A}$  是一个以时间  $t$  和优势  $\text{Adv}_{\mathcal{A}}$  攻击 PKE-PEKS 方案的 DT-Priv 安全性的敌手. 令  $\text{Forge}$  表示事件 “ $\mathcal{A}$  提交了一个形式为  $(vk^*, c, s, \sigma)$  的合法密文到解密预言机”(这里假设  $vk^*$  在游戏开始之前就已确定.).  $\text{Succ}_{\mathcal{A}}$  表示事件 “敌手  $\mathcal{A}$  在游戏中成功”. 我们证明下面的两个断言:

### 断言 6.4

$$\Pr[\text{Forge}] \leq \epsilon_3.$$



### 断言 6.5

$$|\Pr[\text{Succ}_{\mathcal{A}} \wedge \neg \text{Forge}] + \frac{1}{2} \Pr[\text{Forge}] - \frac{1}{2}| \leq \epsilon_1.$$



断言 ?? 的证明. 我们利用敌手  $\mathcal{A}$  构造一个伪造算法  $\mathcal{F}$  攻击 OTS 的 sEUF-CMA 安全性.  $\mathcal{F}$  按照下面的方式模拟  $\mathcal{A}$  在 DT-Priv 游戏中的挑战者行为:

- 初始化: 输入安全参数  $\kappa$  和一次签名的验证公钥  $vk^*$  (由  $\text{OTS.KeyGen}(pp_{\text{OTS}})$  生成, 其中  $pp_{\text{OTS}} \leftarrow \text{OTS.KeyGen}(\kappa)$ ),  $\mathcal{F}$  运行  $\text{HIBE.Setup}(\kappa)$  获取 HIBE 的系统参数  $pp_{\text{HIBE}}$ , 运行  $\text{HIBE.KeyGen}(pp_{\text{HIBE}})$  获取 HIBE 的主公钥和主私钥  $(mpk, msk)$  并将其作为 PKE-PEKS 的公钥和私钥  $(pk, sk)$ .  $\mathcal{F}$  将系统参数  $pp = (pp_{\text{HIBE}}, pp_{\text{OTS}})$  和公钥  $pk$  发送给  $\mathcal{A}(pk)$ .
- 询问阶段 1: 由于  $\mathcal{F}$  知道 PKE-PEKS 的私钥  $sk$ , 所以  $\mathcal{F}$  可以回答敌手的检索令牌查询、匹配测试询问和解密询问. 如果  $\mathcal{A}$  在该阶段提交了一个合法密文  $(vk^*, c, s, \sigma)$  到解密预言机, 则  $\mathcal{F}$  输出  $(c||s, \sigma)$  作为自己的伪造结果并终止游戏.
- 挑战: 当  $\mathcal{A}$  输出两个挑战消息  $m_0^*$  和  $m_1^*$ , 以及一个挑战关键词  $w^*$  时,  $\mathcal{F}$  按以下方式处理: 选择一个随机比特  $b$ , 计算  $c^* \leftarrow \text{HIBE.Encrypt}(pk, 0||vk^*, m_b^*), s^* \leftarrow \text{HIBE.Encrypt}(pk, (1||w^*, vk^*), vk^*)$ , 并通过询问自己的一次签名预言机获取消息  $c^*||s^*$  的一个  $\sigma^*$ . 最后,  $\mathcal{F}$  发送挑战密文  $(vk^*, c^*, s^*, \sigma^*)$  给敌手  $\mathcal{A}$ .

- 询问阶段 2: 如果  $\mathcal{A}$  在该阶段询问了一个合法的解密查询  $(vk^*, c, s, \sigma)$ , 其中  $(c, s, \sigma) \neq (c^*, s^*, \sigma^*)$ , 则  $\mathcal{F}$  直接输出  $(c||s, \sigma)$  作为伪造的签名.
- 猜测: 最终,  $\mathcal{A}$  将输出一个猜测比特  $b'$  作为对  $b$  的猜测结果.

显而易见,  $\mathcal{F}$  模拟的上述游戏环境与敌手  $\mathcal{A}$  在真实 DT-Priv 游戏中的视图是完全一样的并且  $\mathcal{F}$  的成功概率与  $\Pr[\text{Forge}]$  相同. 这与 OTS 的安全性相矛盾. 断言 ??证毕!  $\square$

断言 ??的证明. 我们利用敌手  $\mathcal{A}$  构造一个区分算法  $\mathcal{D}$  攻击 HIBE 的选择身份 IND-HIBE-CPA 安全性.  $\mathcal{D}$  按照下面的方式模拟  $\mathcal{A}$  在 DT-Priv 游戏中的挑战者行为:

- 初始化: 输入 HIBE 的公开参数  $pp_{\text{HIBE}}$ ,  $\mathcal{D}$  运行  $\text{OTS}.\text{Setup}(\kappa)$  生成 OTS 的公开参数  $pp_{\text{OTS}}$ , 再运行  $\text{OTS}.\text{KeyGen}(pp_{\text{OTS}})$  生成  $(vk^*, sigk^*)$ . 接下来, 选择一个身份  $id^* = 0||vk^*$  并发送给  $\mathcal{D}$  的挑战者(即, HIBE 方案的挑战者)作为目标身份, 并获取 HIBE 方案的主公钥  $mpk$ .  $\mathcal{D}$  将 PKE-PEKS 的公钥设置为  $mp = mpk$  并发送给攻击者  $\mathcal{A}$ .
- 询问阶段 1: 当收到敌手  $\mathcal{A}$  的检索令牌、检索测试和解密查询时,  $\mathcal{D}$  按以下方式回答:
  - 检索令牌询问  $\langle w \rangle$ :  $\mathcal{D}$  查询第一层次身份  $\langle 1||w \rangle$  的 HIBE 密钥, 将其作为关键词  $w$  的检索令牌返回给  $\mathcal{A}$ .
  - 检索测试询问  $\langle u, w \rangle$ :  $\mathcal{D}$  首先按照查询检索令牌的方式获取  $w$  的检索令牌  $T_w$ , 然后运行  $\text{Test}(T_w, u)$ , 将结果返回给  $\mathcal{A}$ .
  - 解密询问  $\langle u \rangle$ :  $\mathcal{D}$  将  $u$  拆分为  $(vk, c, s, \sigma)$ . 如果  $\text{OTS}.\text{Verify}(vk, c||s, \sigma) = 0$ , 则  $\mathcal{D}$  拒绝解密, 返回  $\perp$ . 否则,  $\mathcal{D}$  按以下方式处理:
    - 情形 1:  $vk = vk^*$ . 此时, 事件  $\text{Forge}$  发生,  $\mathcal{D}$  终止游戏并输出一个随机比特.
    - 情形 2:  $vk \neq vk^*$ :  $\mathcal{D}$  先通过 HIBE 的密钥询问, 获取身份  $0||vk$  对应的解密密钥  $dk$ , 再计算  $\text{HIBE}.\text{Decrypt}(dk, c)$  并将结果返回给  $\mathcal{A}$ .
- 挑战: 当  $\mathcal{A}$  输出两个挑战消息  $m_0^*$  和  $m_1^*$  及一个挑战关键词  $w^*$  时,  $\mathcal{D}$  按以下方式处理:  $\mathcal{D}$  将  $m_0^*$  和  $m_1^*$  发送给 HIBE 挑战者, 并获取挑战密文  $c^* \leftarrow \text{HIBE}.\text{Encrypt}(pk, 0||vk^*, m_b^*)$ , 其中  $b$  是  $\mathcal{D}$  的挑战者随机选择的. 接下来,  $\mathcal{D}$  计算  $s^* \leftarrow \text{HIBE}.\text{Encrypt}(pk, (1||w^*, vk^*), vk^*), \sigma^* \leftarrow \text{OTS}.\text{Sign}(sigk^*, c^*||s^*)$ . 最后,  $\mathcal{D}$  将  $u^* = (vk^*, c^*, s^*, \sigma^*)$  发送给  $\mathcal{A}$  作为挑战密文.
- 询问阶段 2:  $\mathcal{A}$  可以自适应地进行更多的解密查询、检索令牌查询和检索测试查询. 由于 HIBE 挑战者允许  $\mathcal{D}$  询问身份标识为  $\langle 1||w \rangle$  的用户密钥, 所以  $\mathcal{D}$  可以回答  $\mathcal{A}$  的所有检索令牌询问和检索测试询问. 对于解密查询,  $\mathcal{D}$  的回答方式询问同阶段 1, 但是对于挑战密文  $\langle u^* \rangle$ ,  $\mathcal{D}$  直接返回  $\perp$ .
- 猜测: 最终,  $\mathcal{A}$  输出一比特  $b'$  作为对  $b$  的猜测结果.  $\mathcal{D}$  将  $b'$  作为自己的输出结果返回给 HIBE 挑战者.

在上述模拟游戏中,  $\mathcal{D}$  攻击 HIBE 的选择身份 IND-HIBE-CPA 安全性的密钥询问都是合法有效的. 若事件  $\text{Forge}$  未发生, 则  $\mathcal{D}$  完美地模拟了  $\mathcal{A}$  在 DT-Priv 游戏中的环境. 令  $\text{SuccD}$  表示事件 “ $\mathcal{D}$  在选择身份 IND-HIBE-CPA 实验中输出正确的猜测比特”. 显而易见:

$$|\Pr[\text{SuccD}] - \frac{1}{2}| = |\Pr[\text{SuccA} \wedge \neg\text{Forge}] + \frac{1}{2}\Pr[\text{Forge}] - \frac{1}{2}|.$$

由 HIBE 的安全性可得断言 ??成立.  $\square$

基于断言 ??和断言 ??, 我们可以得到  $\epsilon$  的具体上界, 即

$$\begin{aligned} \epsilon &= |\Pr[\text{SuccA}] - \frac{1}{2}| \leq |\Pr[\text{SuccA} \wedge \text{Forge}] - \frac{1}{2}\Pr[\text{Forge}]| + |\Pr[\text{SuccA} \wedge \neg\text{Forge}] + \frac{1}{2}\Pr[\text{Forge}] - \frac{1}{2}| \\ &\leq \frac{1}{2}\Pr[\text{Forge}] + |\Pr[\text{SuccA} \wedge \neg\text{Forge}] + \frac{1}{2}\Pr[\text{Forge}] - \frac{1}{2}| \\ &\leq \frac{1}{2}\epsilon_3 + \epsilon_1. \end{aligned}$$

对于事件复杂性, 可以推导出算法  $\mathcal{F}$  运行的事件最多为  $t + t_b + q_w t_k + (q_t + q_d)(t_k + t_v + t_d) + 2t_e$ , 算法  $\mathcal{D}$  运行的时间最多为  $t + t_g + (q_t + q_d)(t_v + t_d) + t_e + t_s$ . 此外,  $\mathcal{D}$  最多询问  $q_w + q_t + q_d$  次 HIBE 用户密钥. 引理 ??得证!  $\square$

**引理 6.4**

假设 HIBE 对于层次一身份是  $(t_2, q_{k_2}, \epsilon_2)$ -ANO-HIBE-CPA 匿名安全的, OTS 是  $(t_3, 1, \epsilon_3)$ -sEUF-CMA 安全的. 则 PKE-PEKS 方案具有  $(t, q_w, q_t, q_d, \epsilon)$ -KW-Priv 安全性, 其中

$$\begin{aligned}\epsilon &\leq \epsilon_2 + \frac{1}{2}\epsilon_3, q_w + q_t + q_d \leq q_{k_2}, \\ t &\leq \min\{t_3 - t_b - q_w t_k - (q_t + q_d)(t_k + t_v + t_d) - 2t_e, t_2 - t_g - (q_t + q_d)(t_v + t_d) - t_e - t_s\}.\end{aligned}$$



**证明** 假设  $\mathcal{A}$  是一个以时间  $t$  和优势  $\text{Adv}_{\mathcal{A}}$  攻击 PKE-PEKS 关键词隐私安全性的敌手  $\mathcal{A}$ . 令  $\text{Forge}$  表示事件 “ $\mathcal{A}$  在询问阶段 2 提交了一个检索测试询问  $\langle u, w \rangle$ , 其中  $u = (vk^*, c, s, \sigma)$  是一个合法的 PKE-PEKS 密文,  $w$  是  $w_0^*$  或  $w_1^*$ ”. 我们证明下面的两个断言:

**断言 6.6**

$$\Pr[\text{Forge}] \leq \epsilon_3.$$

**断言 6.7**

$$|\Pr[\text{SuccA} \wedge \neg \text{Forge}] + \frac{1}{2} \Pr[\text{Forge}] - \frac{1}{2}| \leq \epsilon_2.$$



断言 ?? 的证明. 我们利用敌手  $\mathcal{A}$  构造一个伪造算法  $\mathcal{F}$  攻击 OTS 的 sEUF-CMA 安全性.  $\mathcal{F}$  按照下面的方式模拟  $\mathcal{A}$  在 KW-Priv 游戏中的挑战者行为:

- 初始化: 输入安全参数  $\kappa$  和一次签名的验证公钥  $vk^*$  (由  $\text{OTS.KeyGen}(pp_{\text{OTS}})$  生成, 其中  $pp_{\text{OTS}} \leftarrow \text{OTS.KeyGen}(\kappa)$ ),  $\mathcal{F}$  运行  $\text{HIBE.Setup}(\kappa)$  获取 HIBE 的系统参数  $pp_{\text{HIBE}}$ , 运行  $\text{HIBE.KeyGen}(pp_{\text{HIBE}})$  获取 HIBE 的主公钥和主私钥  $(mpk, msk)$  并将其作为 PKE-PEKS 的公钥和私钥  $(pk, sk)$ .  $\mathcal{F}$  将公开参数  $pp = (pp_{\text{HIBE}}, pp_{\text{OTS}})$  和公钥  $pk$  发送给  $\mathcal{A}(pk)$ .
- 询问阶段 1: 由于  $\mathcal{F}$  知道 PKE-PEKS 的私钥  $sk$ , 所以  $\mathcal{F}$  可以回答敌手的检索令牌询问、检索测试询问和解密询问.
- 挑战: 当  $\mathcal{A}$  输出两个挑战关键词  $w_0^*$  和  $w_1^*$ , 以及一个挑战消息  $m^*$  时,  $\mathcal{F}$  按以下方式处理: 选择一个随机比特  $b$ , 计算  $c^* \leftarrow \text{HIBE.Encrypt}(pk, 0||vk^*, m^*)$ ,  $s^* \leftarrow \text{HIBE.Encrypt}(pk, (1||w_b^*, vk^*), vk^*)$ , 并通过询问自己的一次签名预言机获取消息  $c^*||s^*$  的一个签名  $\sigma^*$ . 最后,  $\mathcal{F}$  将挑战密文  $(vk^*, c^*, s^*, \sigma^*)$  发送给敌手  $\mathcal{A}$ .
- 询问阶段 2: 如果  $\mathcal{A}$  提交了一个合法的检索测试询问  $\langle u, w \rangle$ , 其中  $u = (vk^*, c, s, \sigma)$ ,  $w$  等于  $w_0^*$  或  $w_1^*$ , 由于  $(c, s, \sigma) \neq (c^*, s^*, \sigma^*)$ ,  $\mathcal{F}$  直接返回  $(c||s, \sigma)$  作为伪造的签名.
- 猜测: 最终,  $\mathcal{A}$  将输出一个猜测比特  $b'$  作为对  $b$  的猜测结果.

显而易见,  $\mathcal{F}$  模拟的上述游戏环境与敌手  $\mathcal{A}$  在真实 KW-Priv 游戏中的视图是完全一样的并且  $\mathcal{F}$  的成功概率与  $\Pr[\text{Forge}]$  相同. 由此推出与 OTS 的安全性相矛盾, 从而断言 ?? 得证!  $\square$

断言 ?? 的证明. 我们利用  $\mathcal{A}$  构造一个区分算法  $\mathcal{D}$  以攻击 HIBE 在第一层次身份上的 ANO-HIBE-CPA 匿名性.  $\mathcal{D}$  按照下面的方式模拟  $\mathcal{A}$  在 KW-Priv 游戏中的挑战者行为:

- 初始化: 输出 HIBE 的公开参数  $pp_{\text{HIBE}}$  和主公钥  $mpk$ ,  $\mathcal{D}$  运行  $\text{OTS.Setup}(\kappa)$  生成 OTS 的公开参数  $pp_{\text{OTS}}$ . 接下来,  $\mathcal{D}$  将公开参数  $pp = (pp_{\text{HIBE}}, pp_{\text{OTS}})$  以及 PKE-PEKS 的公钥  $pk = mpk$  发送给敌手  $\mathcal{A}$ .
- 询问阶段 1: 当收到  $\mathcal{A}$  的检索令牌询问、匹配测试询问和解密询问时,  $\mathcal{D}$  按以下方式回答:
  - 检索令牌询问  $\langle w \rangle$ :  $\mathcal{D}$  向自己的挑战查询身份  $\langle 1||w \rangle$  的 HIBE 密钥, 将查询结果及关键词  $w$  发送给  $\mathcal{A}$ .
  - 匹配测试询问  $\langle u, w \rangle$ :  $\mathcal{D}$  先按照查询检索令牌的方式获取  $w$  的检索令牌  $T_w$ , 然后运行  $\text{Test}(T_w, u)$ , 将结果返回给  $\mathcal{A}$ .
  - 解密询问  $\langle u \rangle$ :  $\mathcal{D}$  将  $u$  拆分为  $(vk, c, s, \sigma)$ . 如果  $\text{OT.Verify}(vk, c||s, \sigma) = 0$ , 则  $\mathcal{D}$  拒绝解密并返回  $\perp$ . 否则,  $\mathcal{D}$  通过询问 HIBE 挑战者获取身份标识为  $0||vk$  的 HIBE 用户密钥, 然后将解密结果  $\text{HIBE.Decrypt}(dk, c)$  返回给敌手  $\mathcal{A}$ .
- 挑战: 当  $\mathcal{A}$  输出一个挑战消息  $m^*$  及两个挑战关键词  $w_0^*$  和  $w_1^*$  时,  $\mathcal{D}$  按以下方式处理:

1. 运行  $(vk^*, sk_\sigma^*) \leftarrow \text{OTS.KeyGen}(\kappa)$ , 令  $id^* = 0||vk^*$ .
  2. 计算消息  $m^*$  的密文  $c^* \leftarrow \text{HIBE.Encrypt}(pk, id^*, m^*)$ .
  3. 将  $vk^*$  作为消息同两个挑战身份标识  $(1||w_0^*, vk^*)$  和  $(1||w_1^*, vk^*)$  发送给 HIBE 挑战者, 从而得到消息  $vk^*$  在身份  $(1||w_b^*, vk^*)$  下的密文  $s^*$ , 其中  $b$  是  $\mathcal{D}$  的挑战者随机选取的比特.
  4. 计算签名  $\sigma^* \leftarrow \text{OTS.Sign}(sk_\sigma^*, c^*||s^*)$ .
  5. 将  $u^* = (vk^*, c^*, s^*, \sigma^*)$  作为挑战密文发送给对手  $\mathcal{A}$ .
- 询问阶段 2:  $\mathcal{A}$  可以继续自适应地询问检索令牌、匹配测试和解密,  $\mathcal{D}$  按以下方式回答:
    - 检索令牌询问  $\langle w \rangle$ : 只要  $w \neq w_0^*, w_1^*$ ,  $\mathcal{D}$  就可以利用 HIBE 挑战者查询身份标识为  $1||w$  的用户密钥, 并将该密钥及  $w$  作为检索令牌发送给  $\mathcal{A}$ .
    - 匹配测试询问  $\langle u, w \rangle$ : 若询问  $\langle u^*, w_0^* \rangle$  或  $\langle u^*, w_1^* \rangle$  的匹配测试, 根据 KW-Priv 游戏规则,  $\mathcal{D}$  将拒绝回答. 否则,  $\mathcal{D}$  将  $u$  拆分为  $(vk, c, s, \sigma)$ , 首先验证  $\text{OTS.Verify}(vk, c||s, \sigma) = 1$  是否成立. 如果不成立, 则  $\mathcal{D}$  返回 0. 如果成立并且  $w$  不等于  $w_0^*$  或  $w_1^*$ , 则  $\mathcal{D}$  按照检索令牌查询方式获取  $w$  的检索令牌  $T_w$ , 并将计算结果  $\text{Test}(T_w, u)$  返回给  $\mathcal{A}$ . 否则,  $\mathcal{D}$  按以下方式处理:
      - 情形 1:  $vk = vk^*$ . 此时, 事件 **Forge** 发生 (此时,  $w$  等于  $w_0^*$  或  $w_1^*$ . 对于一个合法的询问, 必然有  $u \neq u^*$ ), 则  $\mathcal{D}$  终止游戏并返回一个随机比特.
      - 情形 2:  $vk \neq vk^*$ .  $\mathcal{D}$  利用 HIBE 挑战者获取身份标识为  $(1||w, vk)$  的用户密钥  $dk$ , 若  $vk = \text{HIBE.Decrypt}(dk, s)$  则返回 1, 否则返回 0.
    - 解密询问  $\langle u \rangle$ :  $\mathcal{D}$  按照询问阶段 1 中的方式进行回答  $\mathcal{A}$  的解密查询. 由于 HIBE 挑战者允许  $\mathcal{D}$  询问所有形如  $\langle 0||vk \rangle$  的身份密钥, 所以  $\mathcal{D}$  可以正确地回答所有解密询问.
  - 猜测. 最终,  $\mathcal{A}$  输出一比特  $b'$  作为对  $b$  的猜测结果,  $\mathcal{D}$  将  $b'$  返回给 HIBE 挑战者, 作为自己的猜测结果.

在上述模拟游戏中,  $\mathcal{D}$  攻击 HIBE 的 ANO-HIBE-CPA 匿名性的策略是合法的. 进一步, 若事件 **Forge** 未发生, 则  $\mathcal{D}$  完美地模拟了  $\mathcal{A}$  在 KW-Priv 游戏中的环境. 令 **SuccD** 表示事件 “ $\mathcal{D}$  在 ANO-HIBE-CPA 实验中输出正确的猜测比特”. 显而易见:

$$|\Pr[\text{SuccD}] - \frac{1}{2}| = |\Pr[\text{SuccA} \wedge \neg\text{Forge}] + \frac{1}{2}\Pr[\text{Forge}] - \frac{1}{2}|.$$

由 HIBE 的匿名性, 断言 ?? 成立. □

基于断言 ?? 和断言 ??, 我们可以得到  $\epsilon$  的上界, 即

$$\begin{aligned} \epsilon = |\Pr[\text{SuccA}] - \frac{1}{2}| &\leq |\Pr[\text{SuccA} \wedge \text{Forge}] - \frac{1}{2}\Pr[\text{Forge}]| + |\Pr[\text{SuccA} \wedge \neg\text{Forge}] + \frac{1}{2}\Pr[\text{Forge}] - \frac{1}{2}| \\ &\leq \frac{1}{2}\Pr[\text{Forge}] + |\Pr[\text{SuccA} \wedge \neg\text{Forge}] + \frac{1}{2}\Pr[\text{Forge}] - \frac{1}{2}| \\ &\leq \frac{1}{2}\epsilon_3 + \epsilon_2 \end{aligned}$$

对于事件复杂性, 可以推导出算法  $\mathcal{F}$  运行的时间最多为  $t + t_b + q_w t_k + (q_t + q_d)(t_k + t_v + t_d) + 2t_e$ , 算法  $\mathcal{D}$  运行的时间最多为  $t + t_g + (q_t + q_d)(t_v + t_d) + t_e + t_s$ . 此外,  $\mathcal{D}$  最多询问  $q_w + q_t + q_d$  次 HIBE 用户密钥. 引理 ?? 得证! 通过引理 ?? 和引理 ??, 定理 ?? 得证! □

**基于 IBE 的 PKE-PKES 方案.** 基于一个 IBE 方案  $\text{IBE}=(\text{Setup}, \text{Extract}, \text{Derive}, \text{Encrypt}, \text{Decrypt})$  和一个一次签名方案  $\text{OTS}=(\text{Setup}, \text{KeyGen}, \text{Sign}, \text{Verify})$ , 可以构造一个 CCA 安全的 PKE-PEKS 方案. 不失一般性, 这里假设 IBE 的消息空间是  $\{0, 1\}^n$ , 身份空间是  $\{0, 1\}^*$ , 签名方案的验证密钥空间是  $\{0, 1\}^n$ , 其中  $n = n(\kappa)$ .

#### 构造 6.6 (基于 IBE 的 PKE-PEKS 方案)

- $\text{Setup}(\kappa)$ : 运行  $pp_{\text{IBE}} \leftarrow \text{IBE}.\text{Setup}(\kappa)$  和  $pp_{\text{OTS}} \leftarrow \text{OTS}.\text{Setup}(\kappa)$ , 输出系统参数  $pp = (pp_{\text{IBE}}, pp_{\text{OTS}})$ .
- $\text{KeyGen}(\kappa)$ : 运行  $(mpk, msk) \leftarrow \text{IBE}.\text{KeyGen}(pp_{\text{IBE}})$ , 输出公钥和私钥  $(pk, sk) \leftarrow (mpk, msk)$ .
- $\text{Encrypt}(pk, m, w)$ : 输入公钥  $pk$ 、消息  $m$  和关键词  $w$ , 执行以下步骤:
  1. 运行  $(vk, sigk) \leftarrow \text{OTS.KeyGen}(pp_{\text{OTS}})$ .
  2. 用身份  $0||vk$  加密消息  $m$ ,  $c \leftarrow \text{IBE}.\text{Encrypt}(pk, 0||vk, m)$ .

3. 用身份  $1||w$  加密验证公钥  $vk$ ,  $s \leftarrow \text{IBE}.\text{Encrypt}(pk, 1||w, vk)$ .
  4. 计算  $\sigma \leftarrow \text{OTS}.\text{Sign}(sigk, c||s)$ , 输出密文  $u = (vk, c, s, \sigma)$ .
- $\text{Decrypt}(sk, u)$ : 输入私钥  $sk$  和密文  $u$ , 执行以下步骤:
    1. 将密文  $u$  拆分为  $(vk, c, s, \sigma)$ .
    2. 如果  $\text{OTS}.\text{Verify}(vk, c||s, \sigma) = 1$ , 计算  $dk \leftarrow \text{IBE}.\text{Extract}(sk, 0||vk)$ ,  
输出  $m \leftarrow \text{IBE}.\text{Decrypt}(dk, c)$ .  
否则输出  $1$ .
  - $\text{TokenGen}(sk, w)$ : 输入私钥  $sk$  和关键词  $w$ , 计算  $T_w \leftarrow \text{IBE}.\text{Extract}(sk, 1||w)$ , 输出检索令牌  $T_w$ .
  - $\text{Test}(T_w, u)$ : 输入检索令牌  $T_w$  和密文  $u$ , 执行以下步骤:
    1. 将  $u$  拆分为  $(vk, c, s, \sigma)$ .
    2. 如果  $\text{OTS}.\text{Verify}(vk, c||s, \sigma) = 1$ ,  
如果  $vk = \text{IBE}.\text{Decrypt}(T_w, s)$ , 则输出  $1$ ; 否则, 输出  $0$ .  
否则, 输出  $0$ .



**正确性.** 方案的正确性可由 IBE 方案的正确性直接验证. 方案的安全性由下面的定理保证.

#### 定理 6.4

若 IBE 方案满足选择身份 IND-IBE-CPA 安全性且满足 ANO-IBE-CCA 身份匿名性及弱健壮性, 一次签名 OTS 满足 sEUF-CMA 安全性, 则构造方案 ?? 是一个 jointly CCA 安全的 PKE-PEKS 方案.



**证明** 如果  $\text{OTS}.\text{Verify}(vk, c||s, \sigma) = 1$ , 则称  $u = (vk, c, s, \sigma)$  是一个合法的 PKE-PEKS 密文. 令  $u^* = (vk^*, c^*, s^*, \sigma^*)$  表示敌手  $\mathcal{A}$  收到的挑战 PKE-PEKS 密文. 定理 ?? 的证明由以下两个引理组成.

#### 引理 6.5

假设 IBE 是  $(t_1, q_{k_1}, \epsilon_1)$ -IND-IBE-CPA 安全的, OTS 是  $(t_3, 1, \epsilon_3)$ -sEUF-CMA 安全的. 则 PKE-PEKS 方案具有  $(t, q_w, q_t, q_d, \epsilon)$ -DT-Priv 安全性, 其中

$$\begin{aligned}\epsilon &\leq \epsilon_1 + \frac{1}{2}\epsilon_3, q_w + q_t + q_d \leq q_{k_1}, \\ t &\leq \min\{t_3 - t_b - q_w t_k - (q_t + q_d)(t_k + t_v + t_d) - 2t_e, t_1 - t_g - (q_t + q_d)(t_v + t_d) - t_e - t_s\}.\end{aligned}$$



**证明** 假设  $\mathcal{A}$  是一个以时间  $t$  和优势  $\text{Adv}_{\mathcal{A}}$  攻击 PKE-PEKS 方案的 DT-Priv 安全性的敌手. 令  $\text{Forge}$  表示事件 “ $\mathcal{A}$  提交了一个形式为  $(vk^*, c, s, \sigma)$  的合法密文到解密预言机”(这里假设  $vk^*$  在游戏开始之前就已确定).  $\text{Succ}_{\mathcal{A}}$  表示事件 “敌手  $\mathcal{A}$  在游戏中成功”. 我们证明下面的两个断言:

#### 断言 6.8

$$\Pr[\text{Forge}] \leq \epsilon_3.$$



#### 断言 6.9

$$|\Pr[\text{Succ}_{\mathcal{A}} \wedge \neg \text{Forge}] + \frac{1}{2} \Pr[\text{Forge}] - \frac{1}{2}| \leq \epsilon_1.$$



**断言 ?? 的证明.** 我们利用敌手  $\mathcal{A}$  构造一个伪造算法  $\mathcal{F}$  攻击 OTS 的 sEUF-CMA 安全性.  $\mathcal{F}$  按照下面的方式模拟  $\mathcal{A}$  在 DT-Priv 游戏中的挑战者行为:

- 初始化: 输入安全参数  $\kappa$  和一次签名的验证公钥  $vk^*$  (由  $\text{OTS}.\text{KeyGen}(ppots)$  生成, 其中  $ppots \leftarrow \text{OTS}.\text{KeyGen}(\kappa)$ ),  $\mathcal{F}$  运行  $\text{IBE}.\text{Setup}(\kappa)$  获取 IBE 的公开参数  $ppIBE$ , 运行  $\text{IBE}.\text{KeyGen}(ppIBE)$  获取 IBE 的主公钥和主私钥  $(mpk, msk)$  并将其作为 PKE-PEKS 的公钥和私钥  $(pk, sk)$ .  $\mathcal{F}$  将系统参数  $pp = (ppIBE, ppots)$  和公钥  $pk$  发送给  $\mathcal{A}(pk)$ .

- 询问阶段 1: 由于  $\mathcal{F}$  知道 PKE-PEKS 的私钥  $sk$ , 所以  $\mathcal{F}$  可以回答敌手的检索令牌查询、匹配测试询问和解密询问. 如果  $\mathcal{A}$  在该阶段提交了一个合法密文  $(vk^*, c, s, \sigma)$  到解密预言机, 则  $\mathcal{F}$  输出  $(c||s, \sigma)$  作为自己的伪造结果并终止游戏.
- 挑战: 当  $\mathcal{A}$  输出两个挑战消息  $m_0^*$  和  $m_1^*$ , 以及一个挑战关键词  $w^*$  时,  $\mathcal{F}$  按以下方式处理: 选择一个随机比特  $b$ , 计算  $c^* \leftarrow \text{H}(\text{IBE}.\text{Encrypt}(pk, 0||vk^*, m_b^*))$ ,  $s^* \leftarrow \text{IBE}.\text{Encrypt}(pk, 1||w^*, vk^*)$ , 并通过询问自己的一次签名预言机获取消息  $c^*||s^*$  的一个  $\sigma^*$ . 最后,  $\mathcal{F}$  发送挑战密文  $(vk^*, c^*, s^*, \sigma^*)$  给敌手  $\mathcal{A}$ .
- 询问阶段 2: 如果  $\mathcal{A}$  在该阶段询问了一个合法的解密查询  $(vk^*, c, s, \sigma)$ , 其中  $(c, s, \sigma) \neq (c^*, s^*, \sigma^*)$ , 则  $\mathcal{F}$  直接输出  $(c||s, \sigma)$  作为伪造的签名.
- 猜测: 最终,  $\mathcal{A}$  将输出一个猜测比特  $b'$  作为对  $b$  的猜测结果.

显而易见,  $\mathcal{F}$  模拟的上述游戏环境与敌手  $\mathcal{A}$  在真实 DT-Priv 游戏中的视图是完全一样的并且  $\mathcal{F}$  的成功概率与  $\Pr[\text{Forge}]$  相同. 这与 OTS 的安全性相矛盾. 断言 ?? 证毕!  $\square$

断言 ?? 的证明. 我们利用敌手  $\mathcal{A}$  构造一个区分算法  $\mathcal{D}$  攻击 IBE 的选择身份 IND-IBE-CPA 安全性.  $\mathcal{D}$  按照下面的方式模拟  $\mathcal{A}$  在 DT-Priv 游戏中的挑战者行为:

- 初始化: 输入 IBE 的公开参数  $pp_{\text{IBE}}$ ,  $\mathcal{D}$  运行  $\text{OTS}.\text{Setup}(\kappa)$  生成 OTS 的公开参数  $pp_{\text{OTS}}$ , 再运行  $\text{OTS}.\text{KeyGen}(pp_{\text{OTS}})$  生成  $(vk^*, sigk^*)$ . 接下来, 选择一个身份  $id^* = 0||vk^*$  并发送给  $\mathcal{D}$  的挑战者(即, IBE 方案的挑战者)作为目标身份, 并获取 IBE 方案的主公钥  $mpk$ .  $\mathcal{D}$  将 PKE-PEKS 的公钥设置为  $mp = mpk$  并发送给攻击者  $\mathcal{A}$ .
- 询问阶段 1: 当收到敌手  $\mathcal{A}$  的检索令牌、检索测试和解密查询时,  $\mathcal{D}$  按以下方式回答:
  - 检索令牌询问  $\langle w \rangle$ :  $\mathcal{D}$  查询身份  $\langle 1||w \rangle$  的 IBE 密钥, 将其作为关键词  $w$  的检索令牌返回给  $\mathcal{A}$ .
  - 匹配测试询问  $\langle u, w \rangle$ :  $\mathcal{D}$  首先按照查询检索令牌的方式获取  $w$  的检索令牌  $T_w$ , 然后运行  $\text{Test}(T_w, u)$ , 将结果返回给  $\mathcal{A}$ .
  - 解密询问  $\langle u \rangle$ :  $\mathcal{D}$  将  $u$  拆分为  $(vk, c, s, \sigma)$ . 如果  $\text{OTS}.\text{Verify}(vk, c||s, \sigma) = 0$ , 则  $\mathcal{D}$  拒绝解密, 返回  $\perp$ . 否则,  $\mathcal{D}$  按以下方式处理:
    - 情形 1:  $vk = vk^*$ . 此时, 事件  $\text{Forge}$  发生,  $\mathcal{D}$  终止游戏并输出一个随机比特.
    - 情形 2:  $vk \neq vk^*$ :  $\mathcal{D}$  先通过 IBE 的密钥询问, 获取身份  $0||vk$  对应的解密密钥  $dk$ , 再计算  $\text{IBE}.\text{Decrypt}(dk, c)$  并将结果返回给  $\mathcal{A}$ .
- 挑战: 当  $\mathcal{A}$  输出两个挑战消息  $m_0^*$  和  $m_1^*$  及一个挑战关键词  $w^*$  时,  $\mathcal{D}$  按以下方式处理:  $\mathcal{D}$  将  $m_0^*$  和  $m_1^*$  发送给 IBE 挑战者, 并获取挑战密文  $c^* \leftarrow \text{IBE}.\text{Encrypt}(pk, 0||vk^*, m_b^*)$ , 其中  $b$  是  $\mathcal{D}$  的挑战者随机选择的. 接下来,  $\mathcal{D}$  计算  $s^* \leftarrow \text{IBE}.\text{Encrypt}(pk, 1||w^*, vk^*)$ ,  $\sigma^* \leftarrow \text{OTS}.\text{Sign}(sigk^*, c^*||s^*)$ . 最后,  $\mathcal{D}$  将  $u^* = (vk^*, c^*, s^*, \sigma^*)$  发送给  $\mathcal{A}$  作为挑战密文.
- 询问阶段 2:  $\mathcal{A}$  可以自适应地进行更多的解密查询、检索令牌查询和匹配测试查询. 由于 IBE 挑战者允许  $\mathcal{D}$  询问身份标识为  $\langle 1||w \rangle$  的用户密钥, 所以  $\mathcal{D}$  可以回答  $\mathcal{A}$  的所有检索令牌查询和匹配测试询问. 对于解密查询,  $\mathcal{D}$  的回答方式询问同阶段 1, 但是对于挑战密文  $\langle u^* \rangle$ ,  $\mathcal{D}$  直接返回  $\perp$ .
- 猜测: 最终,  $\mathcal{A}$  输出一比特  $b'$  作为对  $b$  的猜测结果.  $\mathcal{D}$  将  $b'$  作为自己的输出结果返回给 HIBE 挑战者.

在上述模拟游戏中,  $\mathcal{D}$  攻击 IBE 的选择身份 IND-IBE-CPA 安全性的密钥询问都是合法有效的. 若事件  $\text{Forge}$  未发生, 则  $\mathcal{D}$  完美地模拟了  $\mathcal{A}$  在 DT-Priv 游戏中的环境. 令  $\text{SuccD}$  表示事件 “ $\mathcal{D}$  在选择身份 IND-HIBE-CPA 实验中输出正确的猜测比特”. 显而易见:

$$\left| \Pr[\text{SuccD}] - \frac{1}{2} \right| = \left| \Pr[\text{SuccA} \wedge \neg \text{Forge}] + \frac{1}{2} \Pr[\text{Forge}] - \frac{1}{2} \right|.$$

由 IBE 的安全性可得断言 ?? 成立.  $\square$

基于断言 ?? 和断言 ??, 我们可以得到  $\epsilon$  的具体上界, 即

$$\begin{aligned} \epsilon = \left| \Pr[\text{SuccA}] - \frac{1}{2} \right| &\leq \left| \Pr[\text{SuccA} \wedge \text{Forge}] - \frac{1}{2} \Pr[\text{Forge}] \right| + \left| \Pr[\text{SuccA} \wedge \neg \text{Forge}] + \frac{1}{2} \Pr[\text{Forge}] - \frac{1}{2} \right| \\ &\leq \frac{1}{2} \Pr[\text{Forge}] + \left| \Pr[\text{SuccA} \wedge \neg \text{Forge}] + \frac{1}{2} \Pr[\text{Forge}] - \frac{1}{2} \right| \\ &\leq \frac{1}{2} \epsilon_3 + \epsilon_1. \end{aligned}$$

对于事件复杂性, 可以推导出算法  $\mathcal{F}$  运行的事件最多为  $t + t_b + q_w t_k + (q_t + q_d)(t_k + t_v + t_d) + 2t_e$ , 算法  $\mathcal{D}$  运行的时间最多为  $t + t_g + (q_t + q_d)(t_v + t_d) + t_e + t_s$ . 此外,  $\mathcal{D}$  最多询问  $q_w + q_t + q_d$  次 IBE 用户密钥. 引理 ??得证!  $\square$

### 引理 6.6

假设 IBE 是  $(t_2, q_{k_2}, q_{d_2} \epsilon_2)$ ANO-IBE-CPA 匿名安全的和  $(t_4, q_{k_4}, \epsilon_4)$  弱健壮性的, OTS 是  $(t_3, 1, \epsilon_3)$ -sEUF-CMA 安全的. 则 PKE-PEKS 方案具有  $(t, q_w, q_t, q_d, \epsilon)$ -KW-Priv 安全性, 其中

$$\begin{aligned}\epsilon &\leq \epsilon_2 + \epsilon_4 + \frac{1}{2}\epsilon_3, q_w \leq q_{k_2}, q_t + q_d \leq q_{d_2}, q_w + q_t + q_d \leq q_{k_4}, \\ t &\leq \min\{t_3 - t_b - q_w t_k - (q_t + q_d)(t_k + t_v + t_d) - 2t_e, t_4 - t_g, t_2 - (q_w + q_t)t_k\}.\end{aligned}$$



**证明** 假设  $\mathcal{A}$  是一个以时间  $t$  和优势  $\text{Adv}_{\mathcal{A}}$  攻击 PKE-PEKS 关键词隐私安全性的对手  $\mathcal{A}$ . 令  $\text{Forge}$  表示事件 “ $\mathcal{A}$  在询问阶段 2 提交了一个检索测试询问  $\langle u, w \rangle$ , 其中  $u = (vk^*, c, s, \sigma)$  是一个合法的 PKE-PEKS 密文,  $w$  是  $w_0^*$  或  $w_1^*$ ”. 令  $\text{Break}$  表示事件 “密文  $s^*$  在身份  $1||w_b^*$  下的解密结果不等于  $\perp$ ”. 我们证明下面的三个断言:

### 断言 6.10

$$\Pr[\text{Forge}] \leq \epsilon_3.$$



### 断言 6.11

$$\Pr[\text{Break}] \leq \epsilon_4.$$



### 断言 6.12

$$|\Pr[\text{SuccA} \wedge \overline{\text{Forge}} \vee \text{Break}] + \frac{1}{2}\Pr[\text{Forge}] - \frac{1}{2}| \leq \epsilon_2.$$



断言 ??的证明. 我们利用对手  $\mathcal{A}$  构造一个伪造算法  $\mathcal{F}$  攻击 OTS 的 sEUF-CMA 安全性.  $\mathcal{F}$  按照下面的方式模拟  $\mathcal{A}$  在 KW-Priv 游戏中的挑战者行为:

- 初始化: 输入安全参数  $\kappa$  和一次签名的验证公钥  $vk^*$  (由  $\text{OTS.KeyGen}(ppots)$  生成, 其中  $ppots \leftarrow \text{OTS.KeyGen}(\kappa)$ ),  $\mathcal{F}$  运行  $\text{IBE.Setup}(\kappa)$  获取 IBE 的公开参数  $pp_{\text{IBE}}$ , 运行  $\text{IBE.KeyGen}(pp_{\text{IBE}})$  获取 IBE 的主公钥和主私钥  $(mpk, msk)$  并将其作为 PKE-PEKS 的公钥和私钥  $(pk, sk)$ .  $\mathcal{F}$  将公开参数  $pp = (pp_{\text{IBE}}, ppots)$  和公钥  $pk$  发送给  $\mathcal{A}(pk)$ .
- 询问阶段 1: 由于  $\mathcal{F}$  知道 PKE-PEKS 的私钥  $sk$ , 所以  $\mathcal{F}$  可以回答对手的检索令牌询问、匹配测试询问和解密询问.
- 挑战: 当  $\mathcal{A}$  输出两个挑战关键词  $w_0^*$  和  $w_1^*$ , 以及一个挑战消息  $m^*$  时,  $\mathcal{F}$  按以下方式处理: 选择一个随机比特  $b$ , 计算  $c^* \leftarrow \text{IBE.Encrypt}(pk, 0||vk^*, m^*)$ ,  $s^* \leftarrow \text{IBE.Encrypt}(pk, 1||w_b^*, vk^*)$ , 并通过询问自己的一次签名预言机获取消息  $c^*||s^*$  的一个签名  $\sigma^*$ . 最后,  $\mathcal{F}$  将挑战密文  $(vk^*, c^*, s^*, \sigma^*)$  发送给对手  $\mathcal{A}$ .
- 询问阶段 2: 如果  $\mathcal{A}$  提交了一个合法的匹配测试询问  $\langle u, w \rangle$ , 其中  $u = (vk^*, c, s, \sigma)$ ,  $w$  等于  $w_0^*$  或  $w_1^*$ , 由于  $(c, s, \sigma) \neq (c^*, s^*, \sigma^*)$ ,  $\mathcal{F}$  直接将  $(c||s, \sigma)$  作为伪造的签名输出.
- 猜测: 最终,  $\mathcal{A}$  将输出一个猜测比特  $b'$  作为对  $b$  的猜测结果.

显而易见,  $\mathcal{F}$  模拟的上述游戏环境与对手  $\mathcal{A}$  在真实 KW-Priv 游戏中的视图是完全一样的并且  $\mathcal{F}$  的成功概率与  $\Pr[\text{Forge}]$  相同. 由此推出与 OTS 的安全性相矛盾, 从而断言 ??得证!  $\square$

断言 ??的证明. 我们利用  $\mathcal{A}$  构造一个算法  $\mathcal{B}$  攻击 IBE 方案的弱健壮性.  $\mathcal{B}$  按以下方式模拟  $\mathcal{A}$  在 PKE-PEKS 的 KW-Priv 游戏中的挑战者: 输入安全参数  $\kappa$ , IBE 的公开参数  $pp_{\text{IBE}}$  和主公钥  $mpk$ ,  $\mathcal{B}$  运行  $\text{OTS.Setup}(\kappa)$  生成一次签名的公开参数  $ppots$ ,  $\mathcal{B}$  将 PKE-PEKS 的公开参数  $pp = (pp_{\text{IBE}}, ppots)$  和公钥  $pk = mpk$  发送给  $\mathcal{A}(pk)$ . 由于  $\mathcal{B}$  可以询问 IBE 挑战者形如  $1||w$  和  $0||vk$  的身份密钥, 所以  $\mathcal{B}$  可以回答  $\mathcal{A}$  的检索令牌询问、匹配测试询问和解密询问. 当  $\mathcal{A}$  输出一个挑战消息  $m^*$  及两个挑战关键词  $w_0^*$  和  $w_1^*$  时,  $\mathcal{B}$  运行  $\text{OTS.KeyGen}(ppots)$  生成签名方案的一对公钥和私钥  $(vk^*, sigk^*)$ , 选择一个随机比特  $b$ , 将两个身份标识  $1||w_b^*$  和  $1||w_b^*$  及消息  $vk^*$  发送给 IBE 挑战. 假

设  $s^* \leftarrow \text{IBE}.\text{Encrypt}(pk, 1||w_b^*, vk^*)$  是  $\mathcal{B}$  的挑战者生成的密文.  $\mathcal{B}$  将其作为 PKE-PEKS 密文的一部分. 因此,  $\mathcal{B}$  在 IBE 的 WROB 实验中成功的概率恰好是  $\Pr[\text{Break}]$ . 由 IBE 的弱健壮性可知断言 ?? 成立.  $\square$

断言 ?? 的证明. 我们利用  $\mathcal{A}$  构造一个区分算法  $\mathcal{D}$  以攻击 IBE 的 ANO-IBE-CCA 匿名性.  $\mathcal{D}$  按照下面的方式模拟  $\mathcal{A}$  在 KW-Priv 游戏中的挑战者行为:

- 初始化: 输出 IBE 的公开参数  $pp_{\text{IBE}}$  和主公钥  $mpk, \mathcal{D}, \mathcal{D}$  运行  $\text{OTS}.\text{Setup}(\kappa)$  生成 OTS 的公开参数  $pp_{\text{OTS}}$ . 接下来,  $\mathcal{D}$  将公开参数  $pp = (pp_{\text{IBE}}, pp_{\text{OTS}})$  以及 PKE-PEKS 的公钥  $pk = mpk$  发送给对手  $\mathcal{A}$ .
- 询问阶段 1: 当收到  $\mathcal{A}$  的检索令牌询问、匹配测试询问和解密询问时,  $\mathcal{D}$  按以下方式回答:
  - 检索令牌询问  $\langle w \rangle$ :  $\mathcal{D}$  向 IBE 挑战者查询身份  $\langle 1||w \rangle$  的用户密钥, 将查询结果发送给  $\mathcal{A}$ .
  - 匹配测试询问  $\langle u, w \rangle$ :  $\mathcal{D}$  将  $u$  拆分为  $(vk, c, s, \sigma)$ . 如果  $\text{OTS}.\text{Verify}(vk, c||s, \sigma) = 0$ ,  $\mathcal{D}$  输出 0. 否则,  $\mathcal{D}$  向 IBE 挑战者查询在身份  $\langle 1||w, s \rangle$  下的解密结果. 如果解密结果等于  $vk$ , 则  $\mathcal{D}$  返回 1, 否则返回 0.
  - 解密询问  $\langle u \rangle$ :  $\mathcal{D}$  将  $u$  拆分为  $(vk, c, s, \sigma)$ . 如果  $\text{OTS}.\text{Verify}(vk, c||s, \sigma) = 0$ , 则  $\mathcal{D}$  拒绝解密并返回 ⊥. 否则,  $\mathcal{D}$  询问 IBE 挑战者的解密询问  $\langle 0||vk, c \rangle$ , 并将结果返回给对手  $\mathcal{A}$ .
- 挑战: 当  $\mathcal{A}$  输出一个挑战消息  $m^*$  及两个挑战关键词  $w_0^*$  和  $w_1^*$  时,  $\mathcal{D}$  按以下方式处理:
  1. 运行  $(vk^*, sk_\sigma^*) \leftarrow \text{OTS}.\text{KeyGen}(\kappa)$ .
  2. 计算消息  $m^*$  的密文  $c^* \leftarrow \text{IBE}.\text{Encrypt}(pk, 0||vk^*, m^*)$ .
  3. 将  $vk^*$  作为消息同两个挑战身份标识  $1||w_0^*$  和  $1||w_1^*$  发送给 IBE 挑战者, 从而得到消息  $vk^*$  在身份  $1||w_b^*$  下的密文  $s^*$ , 其中  $b$  是  $\mathcal{D}$  的挑战者随机选取的比特.
  4. 计算签名  $\sigma^* \leftarrow \text{OTS}.\text{Sign}(sk_\sigma^*, c^*||s^*)$ .
  5. 将  $u^* = (vk^*, c^*, s^*, \sigma^*)$  作为挑战密文发送给对手  $\mathcal{A}$ .
- 询问阶段 2:  $\mathcal{A}$  可以继续自适应地询问检索令牌、匹配测试和解密,  $\mathcal{D}$  按以下方式回答:
  - 检索令牌询问  $\langle w \rangle$ : 只要  $w \neq w_0^*, w_1^*$ ,  $\mathcal{D}$  就可以利用 IBE 挑战者查询身份标识为  $1||w$  的用户密钥, 并将该密钥作为检索令牌发送给  $\mathcal{A}$ .
  - 匹配测试询问  $\langle u, w \rangle$ : 若询问  $\langle u^*, w_0^* \rangle$  或  $\langle u^*, w_1^* \rangle$  的匹配测试, 根据 KW-Priv 游戏规则,  $\mathcal{D}$  将拒绝回答. 否则,  $\mathcal{D}$  将  $u$  拆分为  $(vk, c, s, \sigma)$ , 首先验证  $\text{OTS}.\text{Verify}(vk, c||s, \sigma) = 1$  是否成立. 如果不成立, 则  $\mathcal{D}$  返回 0. 如果成立并且  $w$  不等于  $w_0^*$  或  $w_1^*$ , 则  $\mathcal{D}$  查询 IBE 挑战者的解密询问  $\langle 1||w, s \rangle$ , 如果解密结果等于  $vk$ , 则  $\mathcal{D}$  返回 1; 如果不等于  $vk$ , 则返回 0. 否则,  $\mathcal{D}$  按以下方式处理:
    - 情形 1:  $vk = vk^*$ . 此时, 事件 Forge 发生 ( $w$  等于  $w_0^*$  或  $w_1^*$ . 对于一个合法的询问, 必然有  $u \neq u^*$ ), 则  $\mathcal{D}$  终止游戏并返回一个随机比特.
    - 情形 2:  $vk \neq vk^*$ . 如果  $s \neq s^*$ , 则  $\mathcal{D}$  利用 IBE 挑战者获取  $\langle 1||w, s \rangle$  的解密结果, 若解密结果等于  $vk$ , 则返回 1, 否则返回 0. 如果  $s = s^*$ ,  $\mathcal{D}$  返回 0.
  - 解密询问  $\langle u \rangle$ :  $\mathcal{D}$  按照询问阶段 1 中的方式进行回答  $\mathcal{A}$  的解密查询. 由于 IBE 挑战者允许  $\mathcal{D}$  询问所有形如  $\langle 0||vk, c \rangle$  的解密查询, 所以  $\mathcal{D}$  可以正确地回答所有解密询问.
- 猜测. 最终,  $\mathcal{A}$  输出一比特  $b'$  作为对  $b$  的猜测结果,  $\mathcal{D}$  将  $b'$  返回给 HIBE 挑战者, 作为自己的猜测结果.

在上述模拟游戏中,  $\mathcal{D}$  攻击 IBE 的 ANO-IBE-CCA 匿名性的策略是合法的. 进一步, 若事件 Forge 和 Break 都未发生, 则  $\mathcal{D}$  完美地模拟了  $\mathcal{A}$  在 KW-Priv 游戏中的环境. 令  $\text{SuccD}$  表示事件 “ $\mathcal{D}$  在 ANO-IBE-CCA 实验中输出正确的猜测比特”. 显而易见:

$$|\Pr[\text{SuccD} - \frac{1}{2}]| = |\Pr[\text{SuccA} \wedge \overline{\text{Forge} \vee \text{Break}}] + \frac{1}{2}\Pr[\text{Forge}] - \frac{1}{2}|.$$

由 IBE 的匿名性, 断言 ?? 成立.  $\square$

根据断言 ??、断言 ?? 和断言 ??, 我们可以得到  $\epsilon$  的上界, 即

$$\begin{aligned} |\Pr[\text{SuccA}] - \frac{1}{2}| &\leq |\Pr[\text{SuccA} \wedge \text{Forge} \vee \text{Break}] - \frac{1}{2}\Pr[\text{Forge}]| + |\Pr[\text{SuccA} \wedge \overline{\text{Forge} \vee \text{Break}}] + \frac{1}{2}\Pr[\text{Forge}] - \frac{1}{2}| \\ &\leq \frac{1}{2}\Pr[\text{Forge}] + \Pr[\text{Break}] + |\Pr[\text{SuccA} \wedge \overline{\text{Forge} \vee \text{Break}}] + \frac{1}{2}\Pr[\text{Forge}] - \frac{1}{2}| \\ &\leq \frac{1}{2}\epsilon_3 + \epsilon_4 + \epsilon_2 \end{aligned}$$

对于事件复杂性,可以推导出算法  $\mathcal{F}$  运行的时间最多为  $t + t_b + q_w t_k + (q_t + q_d)(t_k + t_v + t_d) + 2t_e$ , 算法  $\mathcal{B}$  运行时间最多为  $t + t_g$ , 算法  $\mathcal{D}$  运行的时间最多为  $t + (q_w + q_t)t_k$ . 此外,  $\mathcal{B}$  最多询问  $q_w + q_t + q_d$  次 IBE 的解密密钥,  $\mathcal{D}$  最多询问  $q_w$  IBE 解密密钥和  $q_t + q_d$  次 IBE 密文解密. 引理 ??得证!

通过引理 ??和引理 ??, 定理 ??得证! □

### 6.2.3 PAEKS 方案

Boneh 等人 [BDOP2004] 提出的 PEKS 安全模型仅能保护密文中关键词的隐私, 无法保障检索陷门中的关键词隐私. 事实上, 无论是 PEKS 还是 PKE-PEKS 或者其他可搜索公钥加密方案, 如果关键词加密算法是公开可计算的, 则敌手在获得一个检索令牌时可能获取检索令牌中的关键词信息. 这是因为攻击者可以猜测一个关键词并生成该关键词的密文, 然后利用匹配检索算法判断该密文与获取的检索令牌是否匹配, 从而获取检索令牌中的关键词信息. 该攻击通常称为关键词猜测攻击 (Keyword Guessing Attacks, 简称 KGA). 如果关键词空间较小该攻击非常有效, 例如韦氏字典中仅包含大约  $22500 \approx 2^{18}$  个关键词. 攻击者从一个检索令牌中获取关键词信息的概率至少为  $1/2^{18}$ .

Boneh 等人在提出 PEKS 概念时已指出检索令牌需要安全信道传输, 以防止检索令牌被恶意用户获取并利用. 目前, 抵抗关键词猜测攻击的技术主要有以下几种:

- 扩大关键词空间技术.** 2009 年, Tang 等人 [TC-EuroPKI-2009] 首次提出基于关键词注册的 PEKS 方案. 该方案的基本思想是引入一个关键词注册服务器, 用户在进行关键词加密或生成检索令牌前, 需要利用安全信道将该关键词发送给注册服务器, 注册服务器利用自己的密钥将关键词映射成一个新的 (无语义的) 关键词并通过安全信道传送给用户, 如图 ??所示, 将原始关键词  $W$  映射到  $W' = H(K, W)$ , 其中  $H$  是一个哈希函数. 为了减少安全通信代价, 还可以将密钥  $k$  替换为关键词的盲签名, 不仅能够隐藏注册关键词的信息还可以在公开信道上传输. 此时只需要将注册服务器替换为一个关键词匿名签名服务器 [CMYGHWW-TIFS-2016]. 该类技术一般需要注册服务器保持在线, 所以注册服务器的可靠性和安全性对系统的影响非常大. 此外, 用户需要经常与注册服务器进行交互式通信, 在一定程度上也增加了用户的计算和通信开销.

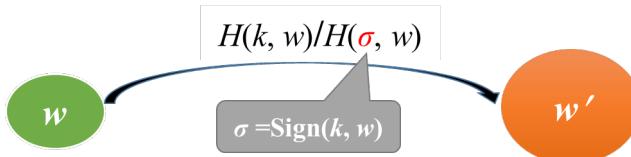


图 6.3: 扩大关键词空间技术

- 指定验证者技术.** 2008 年, Baek 等人 [BSS2008] 提出无安全信道可搜索公钥加密方案的概念, 也称为指定验证者的可搜索公钥加密方案 (Designated server Public-key Encryption with Keyword Search, 简称 dPEKS). 其目的在于去掉用户和服务器之间的安全信道, 提高方案效率. 在 dPEKS 中, 关键词密文由接收者和指定检索服务器的公钥联合加密而来, 只有指定的服务器才可以利用检索令牌进行密文检索. 然而, 该方案后来被发现仍然存在安全缺陷, 并不能抵抗离线关键词猜测攻击 [RPSL2009]. 事实上, 该技术本身存在一定的安全隐患, 这是因为敌手在加密关键词时可以不使用指定服务器的公钥或者选择一个自己生成的公钥, 从而使得该敌手可以进行检索匹配操作. 因此, 许多方案后来发现并不安全 [YPHG-IJCM-2013; NKE2018]
- 指定发送者技术.** 2017 年, Huang 等人 [HL-InfSci-2017] 提出一种称作可搜索公钥认证加密的概念 (Public-key Authenticated Encryption with Keyword Search, 简称 PAEKS), 如图 ??. 在加密关键词时, 通过引入发送者的私钥使得检索令牌仅能用于检索指定发送者的关键词密文, 从而使关键词密文和检索令牌同时满足不可区分性. 当前, 该思想已被推广到构造无证书、基于身份等环境下的可搜索公钥加密方案 [HMZKL-IEEE-TII-2018; LLYSTH-ProvSec-2019; CWZH-IEEE-CC-2022].

下面重点介绍 PAEKS 的概念、安全模型、方案构造及存在的问题.

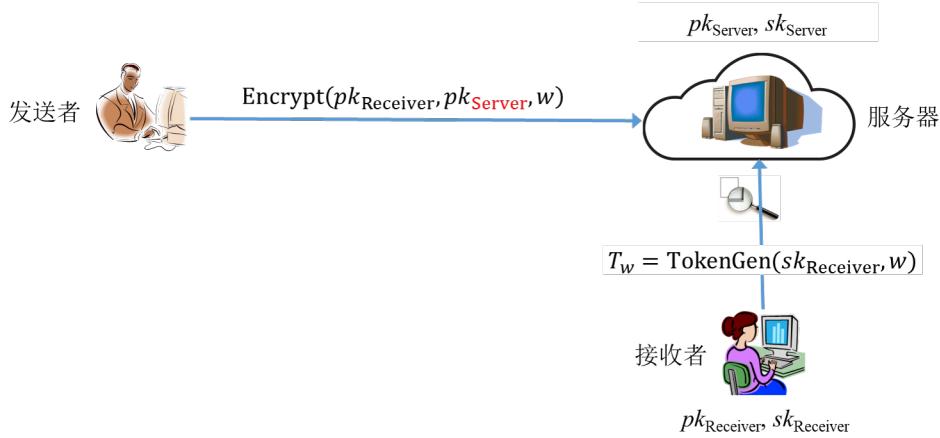


图 6.4: 指定验证者技术

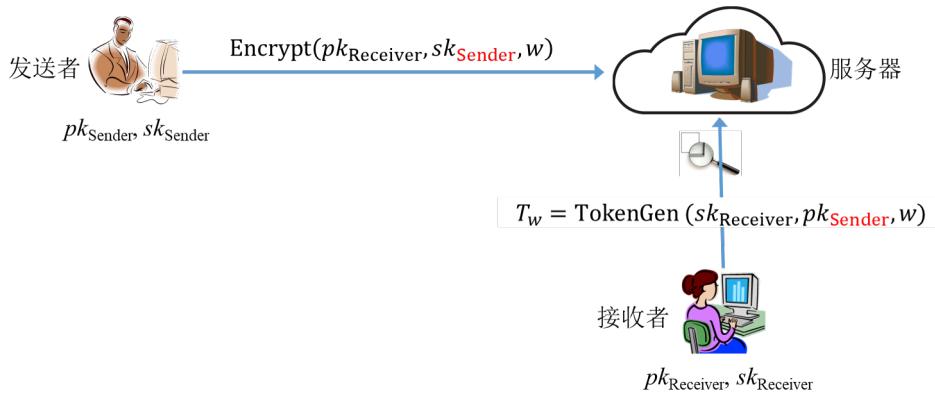


图 6.5: 指定发送者技术

### 6.2.3.1 形式化定义

#### 定义 6.12 (可搜索公钥认证加密)

一个可搜索公钥认证加密方案 PAEKS 由以下六个多项式时间算法组成:

- $\text{Setup}(1^\kappa)$ : 系统参数生成算法以安全参数  $1^\kappa$  为输入, 输出系统公开参数  $pp$ , 其中  $pp$  包含了用户的公钥空间  $\mathcal{PK}$ 、私钥空间  $\mathcal{SK}$ 、关键词空间  $\mathcal{W}$ 、密文空间  $\mathcal{C}$  和检索令牌空间  $\mathcal{TK}$  的描述. 类似公钥加密方案, 该算法由可信第三方生成并公开, 系统中的所有用户共享, 所有算法均将  $pp$  作为输入的一部分.
- $\text{KeyGen}_S(pp)$ : 数据发送者密钥生成算法以公开参数  $pp$  为输入, 输出一对公/私钥对  $(pk_S, sk_S)$ , 其中公钥  $pk_S$  公开, 私钥  $sk_S$  秘密保存.
- $\text{KeyGen}_R(pp)$ : 数据接收者密钥生成算法以公开参数  $pp$  为输入, 输出一对公/私钥对  $(pk_R, sk_R)$ , 其中公钥  $pk_R$  公开, 私钥  $sk_R$  秘密保存.
- $\text{Encrypt}(sk_S, pk_R, w)$ : 关键词加密算法以发送者私钥  $sk_S$ 、接收者公钥  $pk_R$  和关键词  $w \in \mathcal{W}$  为输入, 输出关键词  $w$  的一个可搜索密文  $C_w \in \mathcal{C}$ .
- $\text{TokenGen}(sk_R, pk_S, w)$ : 检索令牌生成算法以接收者私钥  $sk_R$ 、发送者公钥  $pk_S$  和关键词  $w \in \mathcal{W}$  为输入, 输出关键词  $w$  的一个检索令牌  $T_w$ .
- $\text{Test}(pk_S, pk_R, T_{w'}, C_w)$ : 检索算法以发送者公钥  $pk_S$ 、接收者公钥  $pk_R$ 、关键词  $w'$  的检索令牌  $T_{w'}$  和关键词  $w$  的密文  $C_w$  为输入, 如果  $w = w'$ , 则输出 1, 否则, 输出 0.



**正确性和一致性.** 类似 PEKS, PAEKS 的正确性保证了关键词密文的可检索功能, 而一致性降低了检索的错误率. 具体地, 对于任意密钥对  $(pk_R, sk_R)$  和  $(pk_S, sk_S)$ , 任意两个关键词  $w$  和  $w'$ , 令  $C \leftarrow \text{Encrypt}(pk_R, sk_S, w)$ ,

$T_{w'} \leftarrow \text{TokenGen}(sk_R, pk_S, w')$ . 如果  $w = w'$ , 则  $\Pr[\text{Test}(pk_R, pk_S, C, T_{w'}) = 1] = 1 - \text{negl}(\kappa)$ ; 如果  $w \neq w'$ , 则  $\Pr[\text{Test}(pk_R, pk_S, C, T_{w'}) = 0] = 1 - \text{negl}(\kappa)$ .

 **笔记** 如果去掉 PAEKS 方案中数据发送者的密钥生成算法, 从而将数据发送者密钥从加密算法和检索令牌生成算法参数列表中去除, 则上述定义退化为标准的 PEKS 方案的定义.

### 6.2.3.2 安全模型

PAEKS 方案的安全模型包含两个方面: 关键词不可区分性 (Cipher-keyword Indistinguishability, 简称 CI 安全性) 和检索令牌不可区分性 (Trapdoor Indistinguishability, 简称 TI 安全性), 分别保障关键词密文和检索令牌的隐私. 令  $(pk_S, sk_S)$  和  $(pk_R, sk_R)$  分别是一组受攻击的数据发送者和攻击者. 在这两种模型中, 敌手具有下面两种攻击能力:

- **选择密文攻击 (Chosen Keyword to Cipher-keyword, 简称 CKC 攻击):** 在 CKC 攻击中, 敌手拥有获取任意关键词密文的能力, 即敌手可以选择一个关键词  $w$  和指定的任意接收者公钥  $pk$ , 获取该关键词相应的密文. 具体地, 敌手具有访问选择关键词密文谕言机  $\mathcal{O}_{\text{Encrypt}}(sk_S, \cdot, \cdot)$  的能力. 敌手可以自适应地选择一个关键词和一个接收者公钥  $pk$ , 通过该谕言机获取关键词密文  $C_w = \text{Encrypt}(sk_S, pk, w)$ .
- **选择检索令牌攻击 (Chosen Keyword to Trapdoor, 简称 CKT 攻击):** 在 CKT 攻击中, 敌手拥有获取任意关键词检索令牌的能力, 即敌手可以选择一个关键词  $w$  和指定的任意发送者的公钥  $pk$ , 获取该关键词相应的检索令牌. 类似地, 敌手这一能力通过一个检索令牌生成谕言机  $\mathcal{O}_{\text{TokenGen}}(sk_R, \cdot, \cdot)$  来刻画; 敌手可以自适应地选择一个关键词  $w$  和一个发送者公钥  $pk$ , 通过该谕言机获取关键词检索令牌  $T_w = \text{TokenGen}(sk_R, pk, w)$ .

令  $w_0^*$  和  $w_1^*$  是敌手选择的两个挑战关键词, 则敌手在访问上面两个谕言机时必须有所限制, 否则从理论上无法保障任何安全性. 例如在 CI 安全模型中, 敌手会收到某一个挑战关键词的密文  $C_{w_b^*}$ . 显而易见, 敌手不能访问挑战关键词的检索令牌. 否则, 敌手可以通过检索匹配算法直接打破 CI 安全性. 除了这种平凡攻击外, 有些 CI 安全模型还限制敌手访问挑战关键词的密文. 如果不限制敌手访问挑战关键词的密文, 这种选择关键词密文攻击也称为完全选择关键词密文攻击 (Fully CKC attacks).

#### 定义 6.13 (关键词密文不可区分安全性)

定义可搜索公钥认证加密方案敌手  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  的优势函数如下:

$$\text{Adv}_{\mathcal{A}, \text{PAEKS}}^{\text{CI}}(\kappa) = \Pr \left[ \beta' = \beta : \begin{array}{l} pp \leftarrow \text{Setup}(1^\kappa); \\ (pk_S, sk_S) \leftarrow \text{KeyGen}_S(pp); \\ (pk_R, sk_R) \leftarrow \text{KeyGen}_R(pp); \\ (w_0, w_1, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{Encrypt}}(sk_S, \cdot, \cdot), \mathcal{O}_{\text{TokenGen}}(sk_R, \cdot, \cdot)}(pp, pk_S, pk_R); \\ \beta \xleftarrow{R} \{0, 1\}; \\ C^* \leftarrow \text{Encrypt}(sk_S, pk_R, w_\beta); \\ \beta' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{Encrypt}}(sk_S, \cdot, \cdot), \mathcal{O}_{\text{TokenGen}}(sk_R, \cdot, \cdot)}(pp, pk_S, pk_R, state, C^*) \end{array} \right] - \frac{1}{2},$$

在上述定义中, 敌手可以提交任意形如  $(pk, w)$  的询问到关键词密文谕言机, 但是不能提交形如  $(pk_R, w_b^*)$  的询问到检索令牌谕言机. 如果任意的 PPT 敌手  $\mathcal{A}$  在上述定义中的优势函数均为可忽略函数, 则称可搜索公钥认证加密方案 PAEKS 是 (fully) CI 安全的.



**多关键词密文不可区分性.** 由于 PAEKS 的加密算法并不是完全公开可计算的, 需要知道数据发送者的私钥才能计算. 因此, PAEKS 加密算法可以看成是一个对称加密算法. 众所周知, 在对称加密算法中, 若攻击者是一个自适应选择明文攻击敌手, 则单密文不可区分蕴含多密文不可区分. 这里的自适应选择明文攻击敌手是允许攻击者选择挑战消息并获取相应的加密密文. 对于窃听攻击者, 该结论不一定成立, 详细内容可以参考文献 [KL-Cryptography-2007] 定理 3.24. 正因为如此, 有必要定义 PAEKS 的多关键词密文不可区分性 [Qin-Information-Science-2020]. 多

关键词密文安全性游戏的定义同定义??, 不同之处在于挑战阶段, 敌手提交两组关键词  $(w_{0,1}^*, w_{0,2}^*, \dots, w_{0,n}^*)$  和  $(w_{1,1}^*, w_{1,2}^*, \dots, w_{1,n}^*)$ , 而挑战者随机选择一组关键词进行加密, 从而有  $n$  个挑战关键词密文.

类似对称加密方案, 如果 PAEKS 敌手也是自适应的, 能够选择并获取关键词的密文查询, 则完全关键词密文不可区分性 (fully CI-security) 蕴含了完全多密文不可区分性 (fully MCI-security), 即下面的定理成立:

### 定理 6.5

如果一个 PAEKS 方案是完全关键词密文不可区别的, 则该方案也是完全多关键词密文不可区分的.



**笔记** 早期的一些 PAEKS 方案的 CI 安全模型并不允许敌手查询挑战关键词的密文, 如 [HL-InfSci-2017; NE-IET-InfS-2019]. 此时, 单关键词密文不可区分未必蕴含多关键词密文不可区分. 的确如此, Qin 等人在文献 [Qin-Information-Science-2020; QCZZ-ProvSec-2021] 中指出早期的方案在多关键词密文不可区分安全模型中并不安全.

**笔记** 在定义 ?? 中, 敌手允许访问非挑战接收者公钥加密的密文或者是检索非挑战发送者公钥加密密文的检索令牌, 这种情景也称为多用户环境. 多用户环境下的 PAEKS 框架如图 ???. 在早期的 PAEKS 安全模型中, 如 [HL-InfSci-2017], 并未考虑多用户环境. 2019 年, Noroozi 和 Eslami [NE-IET-InfS-2019] 指出单用户环境下的 PAEKS 方案在多用户环境下并不一定安全. 事实上, 早期的 PAEKS 方案在多用户环境下无法保障关键词密文的不可区分性.

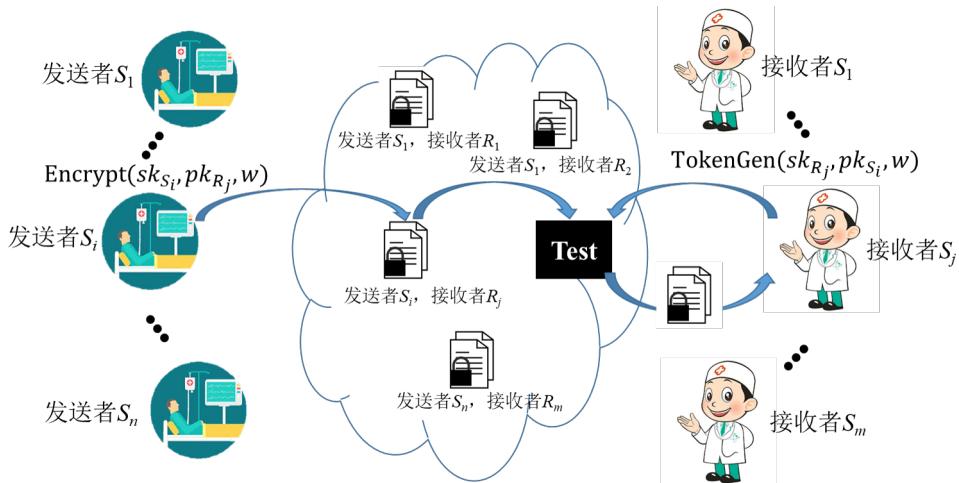


图 6.6: 多用户环境下的 PAEKS 模型

下面将已有的几种针对关键词密文不可区分性的代表性 PAEKS 安全模型总结见表 [table:ch6-PAEKS-CI], 其中  $b \in \{0, 1\}$ ,  $i \in \{1, \dots, n\}$ , 符号 “ $\star$ ” 表示任意公钥或关键词. 从比较结果可以看出, 通过是否允许敌手访问其他用户公钥下的关键词密文或者关键词检索令牌, 是否允许访问挑战关键词的密文, 不同安全模型达到的应用环境有所不同. 在这四种模型中, QCZ+21 方案对敌手访问两个预言机的限制最少, 安全性最高.

表 6.1: PAEKS 方案的密文不可区分安全模型对比

模型	密文不可区分安全性		适用环境
	关键词密文查询预言机	检索令牌查询预言机	
HL17 [HL-InfSci-2017]	$pk = pk_R \wedge w \neq w_b^*$	$pk = pk_S \wedge w \neq w_b^*$	单用户、单密文
NE19 [NE-IET-InfS-2019]	$(pk, w) \neq (pk_R, w_b^*)$	$(pk, w) \neq (pk_S, w_b^*)$	多用户、单密文
QCH+20 [Qin-Information-Science-2020]	$pk = pk_R \wedge w \neq w_{b,i}^*$	$pk = pk_S \wedge w \neq w_{b,i}^*$	单用户、多密文
QCZ+21 [QCZZ-ProvSec-2021]	$(pk, w) = (\star, \star)$	$(pk, w) \neq (pk_S, w_{b,i}^*)$	多用户、多密文

检索令牌不可区分性的定义类似关键词密文不可区分性, 不同之处在于挑战信息是一个检索令牌, 而敌手可以提交任意形式的公钥/关键词对  $(pk, w)$  到检索令牌预言机, 但是不能询问挑战公钥/关键词  $(pk_R, w_b^*)$  的关键词密文. 否则, 敌手通过检索匹配算法直接打破方案的安全性. 定义 ?? 给出了 (fully) TI 安全性的形式化定义.

**定义 6.14 (检索令牌不可区分安全性)**

定义可搜索公钥认证加密方案敌手  $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$  的优势函数如下:

$$\text{Adv}_{\mathcal{A}, \text{PAEKS}}^{\text{TI}}(\kappa) = \Pr \left[ \beta' = \beta : \begin{array}{l} pp \leftarrow \text{Setup}(1^\kappa); \\ (pk_S, sk_S) \leftarrow \text{KeyGen}_S(pp); \\ (pk_R, sk_R) \leftarrow \text{KeyGen}_R(pp); \\ (w_0, w_1, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\text{Encrypt}}(sk_S, \cdot, \cdot), \mathcal{O}_{\text{TokenGen}}(sk_R, \cdot, \cdot)}(pp, pk_S, pk_R); \\ \beta \xleftarrow{\text{R}} \{0, 1\}; \\ T^* \leftarrow \text{TokenGen}(sk_R, pk_S, w_\beta); \\ \beta' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\text{Encrypt}}(sk_S, \cdot, \cdot), \mathcal{O}_{\text{TokenGen}}(sk_R, \cdot, \cdot)}(pp, pk_S, pk_R, state, C^*) \end{array} \right] - \frac{1}{2},$$

在上述定义中, 敌手可以提交任意形如  $(pk, w)$  的询问到检索令牌谕言机, 但是不能提交形如  $(pk_R, w_b^*)$  的询问到关键词密文谕言机. 如果任意的 PPT 敌手  $\mathcal{A}$  在上述定义中的优势函数均为可忽略函数, 则称可搜索公钥认证加密方案 PAEKS 是 (fully) TI 安全的.



**笔记** 上面定义的完全检索令牌不可区分安全性适用于多用户、多检索令牌不可区分的应用环境. 在实际应用中, 一个 PAEKS 方案在实现多关键词密文不可区分性的同时, 可能无法同时满足多检索令牌的不可区分性. 此时, 要求敌手不能询问挑战关键词的检索令牌查询. 代表性的几个 TI 安全模型 [HL-InfSci-2017; NE-IET-InfS-2019; QCZZ-ProvSec-2021] 都有这种限制. 此外, 文献 [HL-InfSci-2017] 定义的安全模型仅适用于单用户环境. 能否同时实现多关键词密文和多检索令牌的完全安全性值得进一步研究.

### 6.2.3.3 方案构造

下面介绍 Qin 等人 [QCZZ-ProvSec-2021] 提出的一种 PAEKS 方案. 该方案也是第一个在多用户环境下满足(完全)关键词密文不可区分安全性和(非完全)检索令牌不可区分安全性的可搜索公钥加密方案.

**构造 6.7 (QCZ+21-PAEKS 方案)**

- **Setup( $1^\kappa$ )**: 选择一个双线性配对  $e : G \times G \rightarrow G_T$ , 其中  $G$  和  $G_T$  是两个阶为素数  $p$  的循环群,  $g$  是  $G$  的一个随机生成元. 接下来, 选择三个哈希函数  $H_1 : \{0, 1\}^* \rightarrow G$ ,  $H_2 : G \rightarrow \{0, 1\}^{\log p}$  和  $H_3 : G \rightarrow \{0, 1\}^{hLen}$ , 其中  $hLen$  是密码哈希函数如 SHA-1 输出的长度. 输出公开参数为  $pp = (G, G_T, p, g, e, H_1, H_2, H_3)$ .
- **KeyGen<sub>S</sub>( $pp$ )**: 随机选择一个随机元素  $u \in \mathbb{Z}_p$ , 计算并输出数据发送者的公钥  $pk_S = g^u$  和私钥  $sk_S = u$ .
- **KeyGen<sub>R</sub>( $pp$ )**: 随机选择两个随机元素  $x, v \in \mathbb{Z}_p$ , 计算并输出数据接收者的公钥  $pk_R = (g^x, g^v)$  和私钥  $sk_R = (x, v)$ .
- **Encrypt( $sk_S, pk_R, w$ )**: 数据发送者选择一个随机元素  $r \in \mathbb{Z}_p$ , 计算  $A = g^r$ ,  $B = H_2(e(h^r, g^x))$ , 其中  $h = H_1(w || pk_S || pk_R || k)$ ,  $k = H_3(g^{uv})$ . 输出关键词  $w$  的密文  $C_w = (A, B)$ .
- **TokenGen( $sk_R, pk_S, w$ )**: 数据接收者计算并输出关键词  $w$  的检索令牌  $T_w = h^x$ , 其中  $h = H_1(w || pk_S || pk_R || k)$ ,  $k = H_3(g^{uv})$ .
- **0/1  $\leftarrow$  Test( $pk_S, pk_R, T_w, C_{w'}$ )**: 对于关键词  $w'$  的密文  $C_{w'} = (A, B)$ , 关键词  $w$  的检索令牌  $T_w$ , 检索服务器判断  $H_2(e(T_w, A)) \stackrel{?}{=} B$  是否成立. 若成立, 则输出 1; 否则, 输出 0.



方案的正确性可以直接得到验证, 方案的安全性分别由定理 ?? 和定理 ?? 保证. 在分析方案的安全性之前, 先介绍安全性证明依赖的两个困难问题: BDH 问题和 ODH 问题. 其中, BDH 问题是标准的计算双线性配对 Diffie-Hellman 问题, 可参考前面的介绍. 下面主要介绍 ODH 问题.

**ODH 问题.** 该问题包含判定性 ODH 问题 (Decisional Oracle Diffie-Hellman problems, 简称 DODH 问题) [ABR2001]

和计算性 ODH 问题 (Computational Oracle Diffie-Hellman problems, 简称 CODH 问题) [QCZZ-ProvSec-2021].

令  $G$  是一个素数阶循环群,  $p$  是群的阶,  $g$  是群的一个随机生成元. 给定  $g^u, g^v$  和谕言机  $\mathcal{O}_v(\cdot)$ , 其中谕言机输入  $X \in G$  输出  $\mathcal{H}_v(X) = H(X^v)$ , 则 DODH 问题的目标是区分  $H(g^{uv})$  和一个随机比特串  $k \in \{0, 1\}^{hLen}$ , 其中  $H$  是一个密码学哈希函数, 值域为  $\{0, 1\}^{hLen}$ . Abdalla、Bellare 和 Rogaway 认为只要敌手不询问谕言机  $\mathcal{H}_v$  在元素  $g^u$  上的值, 则 DODH 是困难的.

#### 定义 6.15 (DODH 假设 [AbdallaBR01])

令  $G$  是一个阶为素数  $p$  的循环群,  $g$  是一个随机生成元,  $H : \{0, 1\}^* \rightarrow \{0, 1\}^{hLen}$  是一个密码学哈希函数. 对于任意一个 PPT 敌手  $\mathcal{A}$ , 如果下面的优势函数关于安全参数  $\kappa$  是可忽略的, 则称 DODH 问题的困难性假设成立.

$$\text{Adv}_{\mathcal{A}}^{\text{DODH}}(\kappa) = \left| \Pr[\mathcal{A}^{\mathcal{O}_v(\cdot)}(g^u, g^v, H(g^{uv})) = 1] - \Pr[\mathcal{A}^{\mathcal{H}_v(\cdot)}(g^u, g^v, k) = 1] \right|$$

其中  $u, v \leftarrow \mathbb{Z}_p$ ,  $k \leftarrow \{0, 1\}^{hLen}$  和  $\mathcal{O}_v(X) = H(X^v)$ .  $\mathcal{A}$  不能查询  $g^u$  的结果  $\mathcal{H}_v(g^u)$ .



与区分  $H(g^{uv})$  和一个随机比特串相反, Qin 等人提出的 CODH 问题目标是计算哈希值  $H(g^{uv})$ . 给定  $g^u$  和  $g^v$ , 在 CODH 问题中, 敌手除了可以查询谕言机  $\mathcal{O}_v(X) = H(X^v)$ , 还可以查询谕言机  $\mathcal{O}_u(X) = H(X^u)$ . 只要敌手不查询谕言结果  $\mathcal{O}_u(g^v)$  或  $\mathcal{O}_v(g^u)$ , 则 CODH 问题被认为也是困难的.

#### 定义 6.16 (CODH 假设 [QinCZZ21])

令  $G$  是一个阶为素数  $p$  的循环群,  $g$  是一个随机生成元,  $H : \{0, 1\}^* \rightarrow \{0, 1\}^{hLen}$  是一个密码学哈希函数. 对于任意一个 PPT 敌手  $\mathcal{A}$ , 如果下面的优势函数关于安全参数  $\kappa$  是可忽略的, 则称 CODH 问题的困难性假设成立.

$$\text{Adv}_{\mathcal{A}}^{\text{CODH}}(\kappa) = \Pr[\mathcal{A}^{\mathcal{O}_u(\cdot), \mathcal{O}_v(\cdot)}(g^u, g^v, H) = H(g^{uv})]$$

其中  $u, v \leftarrow \mathbb{Z}_p$ ,  $\mathcal{O}_u(X) = H(X^u)$  和  $\mathcal{H}_v(X) = H(X^v)$ .  $\mathcal{A}$  不能查询  $g^v$  和  $g^u$  的谕言结果  $\mathcal{O}_u(g^v)$  和  $\mathcal{O}_v(g^u)$ .



**笔记** 一般来地, 一个问题的计算性版本肯定比判定性版本要困难, 至少不会更容易, 例如 DDH 问题与 CDH 问题, 或者 DBDH 问题与 CBDH 问题, 等等. 对于 DODH 问题与 CODH 问题, 细心的读者可能会发现, CODH 问题允许敌手访问的谕言机要比 DODH 问题允许敌手访问的谕言机多. 因此, CODH 问题比 DODH 问题困难的结论并不是那么直接. 尽管如此, Qin 等人证明 CODH 问题并不比 DODH 问题容易解决, 见定理 ???. 因此, 若 DODH 问题是困难的, 则 CODH 问题一定是困难的.

#### 定理 6.6

令  $G$  是一个阶为素数  $p$  的循环群,  $g$  是一个随机生成元  $G$ ,  $H$  一个随机选取的哈希函数. 则有

$$\text{Adv}_{\mathcal{A}}^{\text{CODH}}(\kappa) \leq \text{Adv}_{\mathcal{B}}^{\text{DODH}}(\kappa) + \frac{1}{2^{hLen}}.$$



#### 定理 6.7

如果 BDH 困难性假设成立, 则 PAEKS 方案 ?? 在随机谕言机模型下满足完全关键词密文不可区分性. 具体地, 如果存在一个敌手  $\mathcal{A}$  能够以  $\epsilon$  的优势攻击 PAEKS 方案的完全关键词密文不可区分性, 则可以构造另一个算法  $\mathcal{B}$  以  $\epsilon'$  的概率求解一个 BDH 问题实例的解, 其中  $\epsilon' \geq \frac{\epsilon}{e \cdot Q_{H_2} \cdot (1+Q_T)}$ ,  $e$  是自然对数,  $Q_{H_2}$  和  $Q_T$  分别是访问  $H_2$  谕言机和检索令牌谕言的最大次数.



**证明** [定理 ?? 的证明] 通过归约的方式组织证明. 下面描述如何利用 (算法)  $\mathcal{A}$  作为子程序, 构造一个攻击 BDH 问题的算法  $\mathcal{B}$ . 令  $G$  是阶为素数  $p$  的双线性配对群,  $e : G \times G \rightarrow G_T$  是双线性映射.  $\mathcal{B}$  的输入是一个 BDH 问题实

例  $(g, X = g^x, Y = g^y, Z = g^z) \in G^4$ , 目标是计算  $T = e(g, g)^{xyz}$ .  $\mathcal{B}$  按以下方式模拟  $\mathcal{A}$  在 CI 游戏中的视图.

初始化.  $\mathcal{B}$  选择三个哈希函数  $H_1, H_2, H_3$ , 设置公开参数为  $pp = (G, G_T, p, g, e, H_1, H_2, H_3)$ . 接下来, 选择一个随机元素  $u \in \mathbb{Z}_p$ , 设置  $pk_S^* = g^u$  作为数据发送者的公钥,  $sk_S^* = u$  作为相应的私钥. 类似地, 选择一个随机元素  $v \in \mathbb{Z}_p$ , 设置  $pk_R^* = (X, g^v)$  作为接收者的公钥,  $sk_R^* = (x, v)$  作为相应的私钥, 其中  $x$  对于  $\mathcal{B}$  是已知的. 将公钥  $pk_S^*$  和  $pk_R^*$ , 以及公开参数  $pp$  发送给对手  $\mathcal{A}$ .

哈希询问. 在证明中,  $H_3$  看作是标准的密码哈希函数, 输入  $x$ ,  $\mathcal{A}$  和  $\mathcal{B}$  都可以自行计算相应的哈希值  $H_3(x)$ . 而  $H_1$  和  $H_2$  被看作是随机预言机, 工作过程如下:

- $H_1$ -询问:  $\mathcal{B}$  维护一个列表  $\langle I_i, a_i, c_i, h_i \rangle$ , 称为  $H_1$ -列表, 其中  $I_i = w_i || pk_S || pk_R || k_i$ . 当  $\mathcal{A}$  提交询问  $I_i = w_i || pk_S || pk_R || k_i$ ,  $\mathcal{B}$  按以下方式回答:

- $\mathcal{B}$  检查  $H_1$  列表中是否存在包含  $I_i$  的元素. 如果存在, 则返回相应的  $h_i$  给  $\mathcal{A}$ ; 否则, 选择一个随机比特  $c_i \in \{0, 1\}$  满足  $\Pr[c_i = 0] = \frac{1}{Q_{H_1}}$ .
- 选择一个随机元素  $a_i \in \mathbb{Z}_p$ . 如果  $c_i = 0$ , 设置  $h_i = Y \cdot g^{a_i}$ ; 如果  $c_i = 1$ , 设置  $h_i = g^{a_i}$ .
- $\mathcal{B}$  将  $h_i$  返回给  $\mathcal{A}$  作为  $I_i$  的哈希值  $H_1(I_i)$ , 并将  $\langle I_i, a_i, c_i, h_i \rangle$  添加到  $H_1$ -列表中.

- $H_2$ -询问:  $\mathcal{B}$  维护一个形如  $\langle t_i, V_i \rangle$  的  $H_2$ -列表. 当  $\mathcal{A}$  提交询问  $t_i \in G_T$  时,  $\mathcal{B}$  首先检查列表中是否存在元素  $t_i$ . 如果存在, 则  $\mathcal{B}$  返回相应的值  $V_i$ . 否则,  $\mathcal{B}$  选择一个随机值  $V_i \in \{0, 1\}^{\log p}$ . 最后,  $\mathcal{B}$  将  $V_i$  作为哈希值  $H_2(t_i)$  返回给  $\mathcal{A}$ , 并将  $\langle t_i, V_i \rangle$  添加到  $H_2$ -列表中.

关键词密文询问. 当  $\mathcal{A}$  提交关键词密文询问  $(pk_R = (pk_{R,1}, pk_{R,2}), w_i) \in G^2 \times \{0, 1\}^*$  时,  $\mathcal{B}$  首先计算  $k_i = H_3(pk_{R,2}^u)$ . 然后, 通过  $H_1$ -哈希询问的方式获取  $I_i = w_i || pk_S^* || pk_R || k_i$  的哈希值  $H_1(I_i) = h_i$ . 接下来, 选择一个随机元素  $r \in \mathbb{Z}_p$ , 计算  $A = g^r$  和  $t_i = e(h_i, pk_{R,1})^r$ .  $\mathcal{B}$  通过  $H_2$ -哈希询问的方式获取  $V_i$  的哈希值  $H_2(t_i) = V_i$ . 最后,  $\mathcal{B}$  设置  $B = V_i$ , 并将关键词密文  $C_{w_i} = (A, B)$  发送给  $\mathcal{A}$ .

检索令牌询问. 当  $\mathcal{A}$  提交检索令牌询问  $(pk_S, w_i) \in G \times \{0, 1\}^*$  时,  $\mathcal{B}$  首先计算  $k_i = H_3(pk_S^v)$ , 通过  $H_1$ -哈希询问的方式获取  $I_i = w_i || pk_S || pk_R^* || k_i$  的哈希值  $H_1(I_i) = h_i$ , 及相应的元素组  $\langle I_i, a_i, c_i, h_i \rangle$ . 如果  $c_i = 0$ ,  $\mathcal{B}$  终止游戏. 否则,  $h_i = g^{a_i}$ , 从而  $\mathcal{B}$  可以计算  $T_{w_i} = X^{a_i} (= H_1(I_i)^x)$ . 最终,  $\mathcal{B}$  将检索令牌  $T_{w_i}$  发送给  $\mathcal{A}$ .

挑战. 当  $\mathcal{A}$  提交两个挑战关键词  $w_0^*$  和  $w_1^*$  时,  $\mathcal{B}$  首先选择一个随机比特  $b \in \{0, 1\}$ . 然后, 计算  $k^* = H_3(g^{uv})$ , 通过  $H_1$ -哈希询问的方式获取  $I^* = w_b^* || pk_S^* || pk_R^* || k^*$  的哈希值  $H_1(I^*) = h^*$ , 及相应的元素组  $\langle I^*, a^*, c^*, h^* \rangle$ . 如果  $c^* = 1$ ,  $\mathcal{B}$  终止游戏. 否则,  $c = 0$  且  $h^* = Y \cdot g^{a^*}$ .  $\mathcal{B}$  选择一个随机元素  $V^* \in \{0, 1\}^{\log p}$ , 将挑战关键词密文  $C^* = (Z, V^*)$  返回给  $\mathcal{A}$ . 这隐含地定义了  $H_2(t^*) = V^*$  和  $t^* = e(H_1(I^*), X)^z = e(g^y \cdot g^{a^*}, g^x)^z = e(g, g)^{xz(y+a^*)}$ .

更多询问.  $\mathcal{A}$  可以继续进行关键词密文和检索陷门询问, 所受限制是  $\mathcal{A}$  不能询问检索令牌谕言机关于  $(pk_S^*, w_0^*)$  和  $(pk_S^*, w_1^*)$  的检索陷门.

猜测. 最终,  $\mathcal{A}$  输出一个比特  $b'$  作为对挑战关键词密文中  $C^*$  随机比特  $b$  的猜测结果. 同时,  $\mathcal{B}$  从  $H_2$ -列表中选择一个随机元素  $\langle t, V \rangle$ , 输出  $t/e(X, Z)^{a^*}$  作为对  $e(g, g)^{xyz}$  的猜测结果.

至此, 算法  $\mathcal{B}$  的描述完毕. 下面, 分析  $\mathcal{B}$  的成功概率. 首先, 我们解释为什么  $\mathcal{B}$  在上面的游戏中可能成功. 令  $I_0 = w_0^* || pk_S^* || pk_R^* || k^*$  和  $I_1 = w_1^* || pk_S^* || pk_R^* || k^*$ , 其中  $k^* = H_3(g^{uv})$ . 令  $F$  表示事件 “ $\mathcal{A}$  在游戏中查询了哈希值  $H_2(e(H_1(I_0), X)^z)$  或者  $H_2(e(H_1(I_1), X)^z)$ ”. 接下来, 我们证明下面的引理.

### 引理 6.7

如果对手  $\mathcal{A}$  以优势  $\epsilon$  攻破 PAEKS 方案的完全关键词密文不可区分性, 则有  $\Pr[F] \geq 2\epsilon/(e \cdot (1 + Q_T))$ .



**证明** 令  $E_1$  和  $E_2$  分别表示事件 “ $\mathcal{B}$  在回答检索令牌询问阶段不终止游戏” 和 “ $\mathcal{B}$  在回答挑战关键词密文询问阶段不终止游戏”. 对于  $\mathcal{A}$  的第  $i$  次检索令牌查询  $(pk_s, w_i)$ , 存在元素组  $\langle I_i, a_i, c_i, h_i \rangle$  满足  $I_i = w_i || pk_S || pk_R^* || H_3(pk_S^v)$ . 不管  $c_i = 0$  还是  $c_i = 1$ ,  $h_i$  都具有相同的分布且与  $\mathcal{A}$  的视图独立, 所以  $\mathcal{B}$  在回答每次检索令牌询问时终止游戏的概率最多为  $1/(1 + Q_T)$ . 由于  $\mathcal{A}$  最多查询  $Q_T$  次检索陷门, 所以有  $\Pr[E_1] = \left(1 - \frac{1}{1+Q_T}\right)^{Q_T} \geq \frac{1}{e}$ .

类似地, 在查询挑战关键词密文之前,  $\mathcal{A}$  的视图与  $c^*$  独立, 所以  $\Pr[E_2] = \Pr[c^* = 0] = \frac{1}{1+Q_T}$ .

由于  $\mathcal{A}$  被禁止查询  $(pk_S^*, w_0^*)$  和  $(pk_S^*, w_1^*)$  的检索令牌, 所以事件  $E_1$  和  $E_2$  相互独立. 故有  $\Pr[E_1 \wedge E_2] \geq \frac{1}{e \cdot (1 + Q_T)}$ .

令  $E_3$  表示事件“在  $\mathcal{B}$  不终止游戏的情况下,  $\mathcal{A}$  询问哈希值  $H_2(e(H_1(I_0^*), X)^z)$  或  $H_2(e(H_1(I_1^*), X)^z)$ ”. 显然, 如果  $E_3$  从不发生, 则  $\mathcal{A}$  猜测  $b$  的优势为零. 由于  $\Pr[b' = b] = \Pr[b' = b|E_3] \cdot \Pr[E_3] + \Pr[b' = b|\overline{E}_3] \cdot \Pr[\overline{E}_3]$ , 则有

$$\begin{aligned} \Pr[b' = b|\overline{E}_3] \cdot \Pr[\overline{E}_3] &\leq \Pr[b' = b] \leq \Pr[E_3] + \frac{1}{2} \cdot \Pr[\overline{E}_3] \\ \Rightarrow \quad \frac{1}{2} - \frac{1}{2} \cdot \Pr[E_3] &\leq \Pr[b' = b] \leq \frac{1}{2} + \frac{1}{2} \cdot \Pr[E_3] \\ \Rightarrow \quad \left| \Pr[b' = b] - \frac{1}{2} \right| &\leq \frac{1}{2} \cdot \Pr[E_3]. \end{aligned}$$

由此可得,  $\Pr[E_3] \geq 2 \cdot \epsilon$ . 当  $\mathcal{B}$  不终止游戏时, 上述游戏完美地模拟了完全关键词密文不可区分安全性游戏, 则有  $\Pr[F|(E_1 \wedge E_2)] = \Pr[E_3]$ . 因此,

$$\begin{aligned} \Pr[F] &= \Pr[F|(E_1 \wedge E_2)] \cdot \Pr[E_1 \wedge E_2] + \Pr[F|\overline{E}_1 \wedge \overline{E}_2] \cdot \Pr[\overline{E}_1 \wedge \overline{E}_2] \\ &\geq \Pr[E_3] \Pr[E_1 \wedge E_2] \geq \frac{2\epsilon}{e(1 + Q_T)}. \end{aligned}$$

引理 ?? 证毕! □

注意到事件  $F$  的发生意味着  $H_2$  列表包含满足  $t = e(H_1(I_b), X)^z$  的元素组  $\langle t, V \rangle$  的概率至少为  $1/2$ . 由于,

$$t = e(H_1(I_b), X)^z = e(H_1(I^*), X)^z = e(Y \cdot g^{a^*}, X)^z,$$

所以  $e(g, g)^{xyz} = t/e(X, Z)^{a^*}$ .

又由于  $\mathcal{B}$  选择到正确元素组  $\langle t, V \rangle$  的概率是  $1/Q_{H_2}$ , 所以  $\mathcal{B}$  成功解决 BDH 问题的概率至少为  $\Pr[F]/(2 \cdot Q_{H_2})$ , 即,

$$\epsilon' \geq \frac{\Pr[F]}{2 \cdot Q_{H_2}} \geq \frac{\epsilon}{e \cdot Q_{H_2} \cdot (1 + Q_T)}.$$

定理 ?? 证毕! □

### 定理 6.8

如果 CODH 困难性假设成立, 则 PAEKS 方案 ?? 在随机预言机模型下满足检索令牌不可区分性. 具体地, 如果存在一个敌手  $\mathcal{A}$  能够以  $\epsilon$  的优势攻击 PAEKS 方案的检索令牌不可区分性, 则可以构造另一个算法  $\mathcal{B}$  以  $\epsilon'$  的概率求解一个 CODH 问题实例的解, 其中  $\epsilon' \geq \frac{\epsilon}{Q_{H_1}}$ ,  $Q_{H_1}$  是访问  $H_1$  预言机的最大次数.



**证明** 下面通过归约的方式组织证明. 首先描述如何利用 (算法)  $\mathcal{A}$  作为子程序, 构造一个攻击 CODH 问题的算法  $\mathcal{B}$ . 令  $G$  是阶为素数  $p$  的双线性配对群,  $e : G \times G \rightarrow G_T$  是双线性映射.  $\mathcal{B}$  的输入是一个 CODH 问题实例  $(g, U = g^u, V = g^v, H_3)$  及两个预言机  $\mathcal{O}_u(X) = H_3(X^u)$  和  $\mathcal{O}_v(Y) = H_3(Y^v)$ , 目标是计算  $H_3(g^{uv})$ .  $\mathcal{B}$  按以下方式模拟  $\mathcal{A}$  在 TI 游戏中的视图.

初始化.  $\mathcal{B}$  选择两个哈希函数  $H_1$  和  $H_2$ , 设置公开参数为  $pp = (G, G_T, p, g, e, H_1, H_2, H_3)$ . 然后, 设置  $pk_S^* = U$  作为数据发送者的公钥,  $sk_S^* = u$  作为相应的私钥 (这里的  $u$  对于  $\mathcal{B}$  未知).  $\mathcal{B}$  选择一个随机元素  $x \in \mathbb{Z}_p$ , 设置  $pk_R^* = (g^x, V)$  作为数据接收者的公钥,  $sk_R^* = (x, v)$  作为相应的私钥 (这里的  $x$  对于  $\mathcal{B}$  已知).  $\mathcal{B}$  将公钥  $pk_S^*$  和  $pk_R^*$ , 以及公开参数  $pp$  发送给  $\mathcal{A}$ .

哈希询问. 在证明中, 哈希函数  $H_2$  和  $H_3$  看作是标准的密码哈希函数, 而  $H_1$  看作是随机预言机, 工作如下:

- **$H_1$ -询问:**  $\mathcal{B}$  维护一个形如  $\langle I_i, h_i \rangle$  的  $H_1$  列表, 其中  $I_i = w_i || pk_S || pk_R || k_i$ . 当  $\mathcal{A}$  提交查询  $I_i = w_i || pk_S || pk_R || k_i$  时,  $\mathcal{B}$  随机选择  $h_i \in G$  作为回答结果, 并将  $\langle I_i, h_i \rangle$  添加到  $H_1$  列表中.

$\mathcal{B}$  自己也可能查询特殊元素  $I_i$  的  $H_1$  谎言机, 其中  $k_i$  用特殊符号 “ $\star$ ” 代替, 表示 DH 问题的未知解  $H_3(g^{uv})$ .

此时,  $\mathcal{B}$  选择一个随机元素  $h_i \in G$  并设置  $H_1(I_i) = h_i$ .

关键词密文询问. 当  $\mathcal{A}$  询问  $(pk_R = (pk_{R,1}, pk_{R,2}), w_i) \in G^2 \times \{0,1\}^*$  的关键词密文时, 如果  $pk_{R,2} \neq V$ ,  $\mathcal{B}$  通过查询谎言机  $\mathcal{O}_u(pk_{R,2})$  以获取  $k_i = H_3(pk_{R,2}^u)$ . 否则,  $\mathcal{B}$  设置  $k_i = \star$ . 然后,  $\mathcal{B}$  通过查询谎言机  $H_1$  以获取  $I_i = w_i || pk_S^* || pk_R || k_i$  的哈希值  $H_1(I_i) = h_i$ . 接下来, 选择一个随机元素  $r \in \mathbb{Z}_p$ , 计算  $A = g^r$  和  $B = H_2(e(h_i, pk_{R,1})^r)$ . 最后,  $\mathcal{B}$  将密文  $C_{w_i} = (A, B)$  返回给  $\mathcal{A}$ .

检索令牌询问. 当  $\mathcal{A}$  询问  $(pk_S, w_i) \in G \times \{0,1\}^*$  的检索令牌时, 如果  $pk_S \neq U$ ,  $\mathcal{B}$  通过查询谎言机  $\mathcal{O}_v(pk_S)$  以获取  $k_i = H_3(pk_S^v)$ . 否则,  $\mathcal{B}$  设置  $k_i = \star$ . 然后,  $\mathcal{B}$  通过查询谎言机  $H_1$  以获取  $I_i = w_i || pk_S || pk_R^* || k_i$  的哈希值  $H_1(I_i) = h_i$ .  $\mathcal{B}$  计算检索令牌  $T_{w_i} = h_i^x (= H_1(I_i)^x)$  并返回给对手  $\mathcal{A}$ .

挑战. 当  $\mathcal{A}$  提交两个挑战关键词  $w_0^*$  和  $w_1^*$  时,  $\mathcal{B}$  首先挑选一个随机比特  $b \in \{0,1\}$  并通过查询谎言机  $H_1$  以获取  $I^* = w_b^* || pk_S^* || pk_R^* || \star$  的哈希值  $H_1(I^*) = h^*$ .  $\mathcal{B}$  计算挑战检索令牌  $T_{w_b^*} = (h^*)^x$  并返回给对手  $\mathcal{A}$ .

更多询问.  $\mathcal{A}$  可以继续进行关键词密文和检索令牌询问, 所受限制是  $\mathcal{A}$  不能询问  $(pk_R^*, w_0^*)$  和  $(pk_R^*, w_1^*)$  的关键词密文以及  $(pk_S^*, w_0^*)$  和  $(pk_S^*, w_1^*)$  的检索令牌.

猜测. 最终,  $\mathcal{A}$  输出一个比特  $b'$  作为对挑战检索令牌  $T_{w_b^*}$  中随机比特  $b$  的猜测结果. 同时,  $\mathcal{B}$  从(除去特殊形式询问的) $H_1$  列表中随机选择一个元素组  $\langle I = w || pk_S || pk_R || k, h \rangle$  将其中的  $k$  作为 CODH 问题的解  $H_3(g^{uv})$ .

至此, 算法  $\mathcal{B}$  模拟的 TI 游戏描述完毕. 下面分析  $\mathcal{B}$  成功的概率.

令  $I_0 = w_0^* || pk_S^* || pk_R^* || k^*$  和  $I_1 = w_1^* || pk_S^* || pk_R^* || k^*$ , 其中  $k^* = H_3(g^{uv})$ . 由于  $\mathcal{A}$  在询问挑战检索令牌前, 不能查询关键词密文  $\text{Encrypt}_{sk_S^*}(pk_R^*, w_i^*)$  和检索令牌  $\text{TokenGen}_{sk_R^*}(pk_S^*, w_i^*)$  ( $i = 0, 1$ ), 所以哈希值  $H_1(I_i)$  与  $\mathcal{A}$  的视图独立. 此外, 不管  $I^* = I_0$  还是  $I^* = I_1$ , 相应的哈希值具有相同的分布. 因此, 如果  $\mathcal{A}$  从未询问  $H_1(I_0)$  或  $H_1(I_1)$ , 则对手区分挑战检索令牌的优势为零. 令  $E$  表示事件 “ $\mathcal{A}$  查询过  $H_1(I_0)$  或  $H_1(I_1)$ ”. 下面证明, 如果  $\mathcal{A}$  以不可忽略的优势  $\epsilon$  区分挑战检索令牌, 则事件  $E$  发生的概率也是不可忽略的. 这是因为,

$$\Pr[b' = b] = \Pr[b' = b | E] \cdot \Pr[E] + \Pr[b' = b | \bar{E}] \cdot \Pr[\bar{E}]$$

且

$$\begin{aligned} & \Pr[b' = b | \bar{E}] \cdot \Pr[\bar{E}] \leq \Pr[b' = b] \leq \Pr[E] + \frac{1}{2} \cdot \Pr[\bar{E}] \\ \Rightarrow & \frac{1}{2} - \frac{1}{2} \cdot \Pr[E] \leq \Pr[b' = b] \leq \frac{1}{2} + \frac{1}{2} \cdot \Pr[E] \\ \Rightarrow & \left| \Pr[b' = b] - \frac{1}{2} \right| \leq \frac{1}{2} \cdot \Pr[E]. \end{aligned}$$

从而可得  $\Pr[E] \geq 2\epsilon$ .

$\mathcal{B}$  均匀且随机选择挑战比特  $b$ . 如果事件  $E$  发生, 则哈希列表  $H_1$  至少以  $1/2$  概率存在形如  $I_b = w_b^* || pk_S^* || pk_R^* || H_3(g^{uv})$  的哈希询问. 因此,  $\mathcal{B}$  从  $H_1$  列表中随机选取到元素组  $\langle I_b, h^* \rangle$  的概率至少为  $1/Q_{H_1}$ . 综合上述条件,  $\mathcal{B}$  找到 CODH 问题解的概率至少为  $\epsilon' \geq \frac{\epsilon}{Q_{H_1}}$ . 定理 ?? 证毕!  $\square$

### 6.2.3.4 应用

Boneh 等人 [BDOP2004] 提出可搜索公钥加密的一个应用场景是加密邮件路由过滤. 假设 Alice 希望她的邮件网关能够将一些包含关键词 “urgent” 的邮件及时推送到她的手机上, 而其他邮件可以暂时存放在台式电脑中. 她可以生成关键 “urgent”的一个检索令牌并嵌入在邮件网关中. 当其他用户如 Bob, 想发送一封邮件给 Alice 时, 他可以利用 Alice 的公钥将邮件内容及关键词加密, 再将加密后的邮件发送给 Alice. 当 Alice 的邮件网关收到该加密邮件时, 可以利用检索匹配算法检测邮件内容是否包含关键词 “urgent”. 如果匹配成功, 则邮件网关将该邮件路由到 Alice 的手机上. 否则, 将该邮件路由到 Alice 的台式电脑中.

可搜索公钥加密的一个优点是邮件内容保密的同时, 邮件网关可以利用指定关键词的检索令牌过滤出相关的加密邮件. 可搜索公钥认证加密具有类似的功能, 同时还可以保障检索令牌的隐私. 然而, 对于不同发送者发送的加密邮件, 即使是检索同一关键词, 邮件接收者要分别生成针对不同用户的检索令牌, 应用环境有所受限. 尽管如此, PAEKS 可应用于公司等固定对象的邮件分类服务, 如图 ?? 所示. 许多公司会不定时发送一些通知邮件, 同时员工也会收到大量的重要或非重要的邮件. 利用 PAEKS 技术, 每个员工可以在邮件网关处指定一个检索令牌, 从而可以对公司发送的邮件进行分类处理.

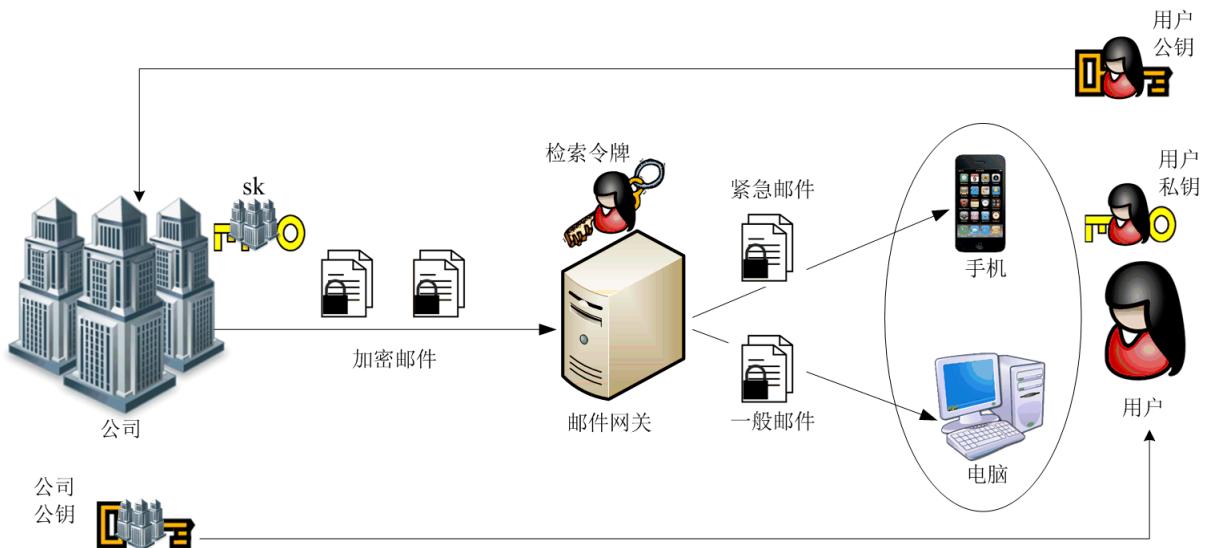


图 6.7: 可搜索公钥认证加密的应用

除了上述应用外, 可搜索(认证)公钥加密还可以应用于加密电子医疗记录系统 [GY-JMS-2015; LL-CC-2019], 外包云数据加密存储和检索 [GYZHLQH-TDSC-2021], 等等.

## 第七章 标准化及工程实践

### 内容提要

□ 公钥加密的标准化工作简介

□ 公钥加密的工程实践经验

## 7.1 标准化与工程实践

纸上得来终觉浅, 绝知此事要躬行.

— 宋·陆游《冬夜读书示子聿》

### 7.1.1 公钥加密的标准化

标准化对密码技术的实际落地应用具有重要意义, 否则, 即使是同一密码方案/协议也可能由于参数选取、接口设计缺乏统一的规范而无法互联互通. 以下首先简要介绍与密码领域相关较为密切的国内外标准化组织.

#### 7.1.1.1 国内外标准化组织简介

##### 1. 国际标准化组织与国际电工委员会

国际标准化组织 (ISO) 与国际电工委员会 (IEC) 联合成立了名为 ISO/IEC JTC 1 的委员会, 重点关注信息技术领域的标准化, 联合制定了一系列 ISO/IEC 标准, 其中的 ISO/IEC 18033 系列标准规定了加密算法、密码协议和密钥管理技术. ISO/IEC 标准通常由世界各地的成员国共同参与制定, 涉及多轮草案和投票, 标准化过程严格, 需经过彻底的审查和意见反馈与修订. 正因如此, ISO/IEC 标准具有广泛的国际认可度, 在实施全球化技术和安全政策方面均具有强大的影响力, 可确保来自不同供应商的产品和服务可以安全有效地协同工作. 符合 ISO/IEC 标准的密码产品通常质量和安全方面具有较高的置信度, 这对于金融交易、医疗保健和国家安全等关键应用至关重要.

##### 2. 互联网工程任务组

互联网工程任务组 (IETF) 是一个开放的标准组织, 负责开发和推广互联网标准, 特别是维护 TCP/IP 协议族的标准. 与 ISO/IEC 不同, 它不依赖于任何特定国家或管理机构, 没有正式的会员资格或会员资格要求. IETF 将其技术文档发布为征求意见稿 RFC(Requests for Comments), IETF 制定的 RFC 全方位涵盖了计算机网络体系, 在安全性与隐私方面, IETF 制定的技术标准和实践文档致力于抵御已知和新出现的威胁, 为互联网的安全和隐私提供了重要的基础要素. IETF 针对安全方面正在进行的一些工作包括: 最新版本的传输层安全协议 TLS 1.3、自动证书管理环境协议 (最近发布为 RFC 8555) 和消息传递分层安全协议等. IETF 标准具有高度的包容性, 任何人均可参与到标准制定的过程中, 且标准制定更多的基于实施和部署规范方面的实际经验, 强调实用性和执行性. IETF 标准在万物互联互通中起到了至关重要的作用, 标准化的通信协议确保不同的系统可以无缝地协同工作, SSL/TLS 等就是 IETF 标准的典范工作.

##### 3. 美国电气电子工程师学会

美国电气电子工程师学会 (Institute of Electrical and Electronics Engineers, IEEE) 的标准化组织为推出了公钥密码学标准 IEEE P1363. 该标准包括传统公钥密码学 (IEEE Std 1363-2000 and 1363a-2004)、格基公钥密码学 (IEEE Std 1363.1-2008)、口令基公钥密码学 (IEEE Std 1363.2-2008)、使用双线性映射的公钥密码学 (IEEE Std 1363.3-2013).

##### 4. 中国国家标准局

中国国家标准局现称为中国国家标准化管理委员会 (SAC), 是中国国务院直属的政府机构, 它负责起草和管理国家标准, 并代表中国加入 ISO/IEC 等国际标准组织. 中国国家标准局一直积极制定信息安全国家标准, 包括密码算法和协议. SAC 的标准对国内行业具有重大影响, 并且经常被用作中国境内法规的基础. 随着中国在全球贸易中的地位不断提升, SAC 标准在国际上的影响力也越来越高.

##### 5. 美国国家标准局

美国国家标准与技术研究院 (National Institute of Standards, NIST) 成立于 1901 年, 现隶属于美国商务部. NIST 是美国最古老的物理科学实验室之一, 成立之初的目的是消除当时美国工业在测量基础设施方面的短板. 当前, 从智能电网和电子健康记录到原子钟、先进纳米材料和计算机芯片, 无数产品和服务都在某种程度上依赖于 NIST 提供的技术、测量和标准. NIST 致力于制定与信息技术各个方面相关的标准和指南, 还专门为联邦机构和广大公众制定密码标准和指南, 包括哈希算法、随机数生成算法、加密方案、签名方案

和后量子密码方案等。尽管 NIST 是美国机构,但具有国际影响力,其标准和指南不仅被美国联邦机构广泛采用,还被私营部门组织和全球其他政府广泛采用。

## 6. 美国国家标准学会

美国国家标准学会 (American National Standards Institute, ANSI) 成立于 1918 年,是美国非盈利民间标准化团体。作为自愿性标准体系中的协调中心,ANSI 的主要职能是协调国内各机构团体的标准化活动、审核批准美国国家标准、代表美国参加国际标准化活动、提供标准信息咨询服务等。在密码学领域,该组织制定了基于椭圆曲线的公钥密码学标准 ANSI X9.63。

## 7. RSA 公司

1990 年起, RSA 公司发布了一系列公钥密码技术标准 PKCS(Public Key Cryptography Standards),旨在推广公司拥有专利的密码算法,如 RSA 加密算法与签名、Schnorr 签名等。尽管 PKCS 系列不是工业标准,但其中的部分算法已经在纳入若干标准化组织(如 IETF 和 PKIX 工作组)的正式标准进程中。

### 7.1.1.2 公钥加密标准方案

选择密文安全常简称为 CCA 安全,自上世纪 90 年代起即成为公钥加密的事实标准 (de facto standard),正因如此,绝大多数标准化组织制定的公钥加密标准均具备选择密文安全。以下首先介绍基于数论类假设的公钥加密标准方案。

- 基于整数分解类困难问题的公钥加密方案

PKCS#1 [PKCS1] 是 PKCS 系列标准中最早也应用最广泛的一个,制定了 RSA 加密和签名标准,最新的版本号为 v2.2。PKCS#1 中定义了 RSA 公钥和私钥应如何表示和存储,规定了基本的 RSA 操作,包括加密和解密,签名和验证。特别的,标准中为 RSA 加密方案引入填充机制 OAEP(Optimal Asymmetric Encryption Padding),得到可证明 IND-CCA 安全的 RSA-OAEP,解决了早期版本中存在的安全问题,如针对 PKCS#1 v1.5 填充的自适应选择明文攻击和 Bleichenbacher 攻击。

- 基于离散对数类困难问题的公钥加密方案

离散对数类困难问题根据代数结构的不同,划分为数域和椭圆曲线两个子类,在同样的安全级别下,后者的参数规模更为紧致,因此构建于其上的密码方案相比前者具有显著的性能优势,但是由于数学结构复杂,工程实践的难度也更大。DHIES (Diffie-Hellman Integrated Encryption Scheme, DHIES) 是 DHAES [ABR-ePrint-1999] 的标准化方案,采用混合加密方式,密钥封装机制基于数域循环群上的 ElGamal PKE 和哈希函数构造,数据封装机制基于消息验证码和堆成加密方案构造。DHIES 整体方案在随机预言机模型中基于 CDH 假设具备可证明的选择密文安全。ECIES (Elliptic Curve Encryption Scheme, ECIES) 是 DHIES 在椭圆曲线循环群上的对于版本。DHIES 和 ECIES 被纳入 IEEE 1363a、ANSI X9.63 和 ISO/IEC 18033-2 标准。ECIES 还被椭圆曲线密码标准组 (Standards for Efficient Cryptography Group, SECG) 纳入到椭圆曲线密码学标准 SEC 1 [SEC-1] 中。

NIST 在联邦信息处理标准 (Federal Information Processing Standards Publication) FIPS 186-5 [FIPS-186-5]、SECG 在 SEC 2 [SEC-2] 和 ECC Brainpool 在 RFC 5639 [RFC-5639] 中分别给出了推荐的椭圆曲线参数选择。中国密码管理局为满足国内电子认证服务系统等应用需求,于 2010 年 12 月 17 日发布了《SM2 椭圆曲线公钥密码算法》 [SM2],2016 年成为中国国家密码标准。SM2 标准中包括推荐椭圆曲线参数和包括公钥加密方案在内的各种类型公钥密码方案。

- 基于格类困难问题的公钥加密方案

Shor 算法的出现意味着在后量子时代基于数论类困难问题的密码方案将不再安全,因此设计能够抵抗量子攻击的密码方案成为当前密码学的前沿热点,其中格基方案是抗量子安全密码学中的主流。NIST 自 2016 年开始了后量子密码学标准方案的征集。经过最新一轮的评审,NIST 于 2023 年 8 月 24 号发布了 3 个 FIPS 草案拟定了抗量子密码系列方案,其中 FIPS 203 [FIPS-203] 定义了基于 LWE 困难问题的公钥加密方案 CRYSTALS-KYBER [Kyber-EUROSP-2018]。

如前所述,绝大多数标准中的公钥加密方案都满足 IND-CCA 安全。然而,IND-CCA 安全与同态性无法共存,

在分布式计算环境和大数据应用等密态数据的可操作性比机密性保护更重要的场景中, 迫切需要标准化的同态公钥加密方案.

- 部分同态加密方案标准

ISO/IEC 18033-6 [**ISO/IEC-18033-6**] 标准中定义了 Exponential ElGamal 和 Paillier [**Paillier-EUROCRYPT-1999**] 两个加法同态加密方案.

- 全同态加密方案标准

全同态加密尚处于飞速发展阶段, 然而工业界的应用需求更为迫切. 2017 年, 来自 IBM、Microsoft、Intel 和 NIST 和其它开放组织的研究人员共同成立了全同态标准化联盟 (Homomorphic Encryption Standardization Consortium), 并发布了同态加密标准文档 [**HE-Standard**]. 该文档涵盖了适用于整数运算的 Brakerski-Gentry-Vaikuntanathan (BGV) [**BGV-TOCT-2014**] 和 Brakerski/Fan-Vercauteren (BFV) [**Brakerski-CRYPTO-2012; FV-ePrint-2012**]、适用于浮点数运算的 Cheon-Kim-Kim-Song (CKKS) [**CKKS-ASIACRYPT-2017**] 以及适用于 Boolean 电路求值的 Ducas-Micciancio (FHEW) [**DM-EUROCRYPT-2015**] 和 Chillotti-Gama-Georgieva-Izabachene (TFHE) [**CGGI-JoC-2020**]. 该文档尽管不是官方标准, 但基本可以看成事实上的标准.

## 7.1.2 公钥加密的工程实践

实现密码算法对程序员的素质要求较高, 既需要专业的密码知识以确保实现的忠实性和安全性, 也需要精湛的编程技术以确保实现的效率. 在一般情况下, 不建议非专业程序员自行从底层起构建密码算法, 如此不仅可省去重复制造轮子的无用功, 更能避免造出方形轮子的错误.

### 7.1.2.1 重要方案的优秀开源实现

工程实践中经常需要使用已有的公钥加密方案, 以下推荐部分常用方案的优秀开源实现供一线程序员按图索骥.

#### 标准公钥加密方案

- RSA-OAEP: OpenSSL 库 [**OpenSSL**] 中提供了 C 语言版本的实现.
- Paillier: mpc4j 库 [**mpc4j**] 提供了 Java 语言的实现.
- ElGamal: Kunlun 库 [**libKunlun**] 中给出了 ElGamal PKE 及其多个衍生方案的 C++ 实现, 同时给出了配套的零知识证明实现, 可直接部署应用于密态计算场景.

#### 属性加密

- FAME(Fast Attribute-based Message Encryption) [**AC-CCS-2017**]: 首个基于标准假设完全安全的密文策略和密钥策略 ABE 方案 (对策略类型或属性没有任何限制), 构建于 Type-III 双线性映射上. 相应的开源实现可参考: <https://github.com/sagrawal87/ABE>

全同态加密: Microsoft 的 SEAL (Simple Encrypted Arithmetic Library) 库 [**SEAL**] 给出了 BGV、BFV 和 CKKS 方案的优秀实现, PALISADE 的后继者 OpenFHE [**OpenFHE**] 则包含了所有主流全同态加密方案的实现.

### 7.1.2.2 重要的开源密码库

下面的内容适用于程序员在实现自研公钥加密方案时, 为如何选择合适的密码算法库做出参考.

表 7.1: 常用开源密码算法库

库名	编程语言	支持算子类型				易用性	实时性	国密算法支持
		对称密码	大整数运算	椭圆曲线	双线性映射			
OpenSSL	C	✓	✓	✓	✗	★★★★★	★★★★★	SM2/SM3/SM4
tongsuo	C	✓	✓	✓	✗	★★★★★	★★★★★	SM2/SM3/SM4
gmSSL	C	✓	✓	✓	✗	★★★	★★★	SM2/SM3/SM4/SM9/ZUC
mcl	C/C++	✓	✓	✓	✓	★★★	★★★★★	—
MIRACL	C/C++	✓	✓	✓	✓	★★★★★	★★★	—
NTL	C++	✗	✓	✗	✗	★★★★★	★★★★★	—
Bouncy Castle	Java/C#	✓	✓	✓	✓	★★★★★	★★★★★	SM2/SM3/SM4
Crypto++	C++	✓	✓	✓	✓	★	★★★★	SM3/SM4
Botan	C++	✓	✓	✓	✗	★★★★★	★★★★★	SM2/SM3/SM4
libsodium	C	✗	✓	✓	✗	★★★	★★	—
libgcrypt	C	✗	✓	✓	✗	★★★★	★★★★	SM2/SM3/SM4

### 7.1.3 密码学的工程实践经验

请伟嘉在此处补充: 其实工程实践经验应该和密码学中的对象关联不强, 适用于公钥加密的经验应该同样适用于其他种类方案.β

## 后记

终于糊弄完了.