密码学报 ISSN 2095-7025 CN 10-1195/TN
Journal of Cryptologic Research, 2019, 6(5): 541–560

E-mail: jcr@cacrnet.org.cn
http://www.jcr.cacrnet.org.cn
Tel/Fax: +86-10-82789618

混淆电路与不可区分混淆*

张 正[1,2], 张方国[1,2]

1. 中山大学 数据科学与计算机学院, 广州 510006
2. 广东省信息安全技术重点实验室, 广州 510006

通信作者: 张方国, E-mail: isszhfg@mail.sysu.edu.cn

摘 要: 混淆电路 (garbled circuits) 和混淆 (obfuscation) 是密码研究领域中两个非常重要的工具. 混淆电路是由 Yao 在安全两方计算问题中提出的一种构造方法发展而来, 混淆则是从代码混淆发展而来. 混淆电路和混淆均要求对电路进行 "加密" 使得电路内容不可读且保留其功能性. 由于直接对电路操作, 混淆电路和混淆相当于对函数本身进行 "保护". 这一特点使得他们在密码学领域以及现实社会均具有广泛的应用. 但是, 如此相似的两个工具在通用构造上以及相互构造中却呈现出完全相反的难度. 混淆电路的实现仅需要 AES 等简单的加密方案, 而目前大多数的通用混淆即不可区分混淆的实现则是依赖于多线性映射的构造. 也就是说, 现有的混淆电路的实现效率远远高于不可区分混淆的实现效率. 这一实现效率上的巨大差异使得混淆电路和混淆的区别和联系成为我们关注的问题. 本文对混淆电路和不可区分混淆进行介绍, 包括语义定义、安全定义、常见的构造方案, 以及应用场景. 并对两者在不同的安全级别间的区别与联系进行分析说明, 探索两者之间互相构造的可能性. 通过这些分析和探索, 我们希望为接下来两个工具的构造和优化提供新的思路.

关键词: 布尔电路; 混淆电路; 混淆; 不可区分混淆; 可重用混淆电路

中图分类号: TP309.7 **文献标识码:** A DOI: 10.13868/j.cnki.jcr.000321

中文引用格式: 张正, 张方国. 混淆电路与不可区分混淆[J]. 密码学报, 2019, 6(5): 541–560.

英文引用格式: ZHANG Z, ZHANG F G. Garbled circuits and indistinguishability obfuscation[J]. Journal of Cryptologic Research, 2019, 6(5): 541–560.

Garbled Circuits and Indistinguishability Obfuscation

ZHANG Zheng[1,2], ZHANG Fang-Guo[1,2]

1. School of Data and Computer Science, Sun Yat-Sen University, Guangzhou 510006, China
2. Guangdong Key Laboratory of Information Security, Guangzhou 510006, China

Corresponding author: ZHANG Fang-Guo, E-mail: isszhfg@mail.sysu.edu.cn

Abstract: Garbled circuits and obfuscation are two creative tools in the field of cryptography. Garbled circuits begin with Yao's construction of two-party secure computation, while obfuscation comes from code obfuscation. They both aim to "encrypt" the circuits while preserving functionality, which makes them widely used in the field of cryptography and the real society. However, these tools present completely opposite difficulties in general construction. The implementation of the garbled circuits

* 基金项目: 国家自然科学基金 (61672550); 国家重点研发计划 (2017YFB0802500)
Foundation: National Natural Science Foundation of China (61672550); National Key Research and Development Program of China (2017YFB0802500)
收稿日期: 2018-12-26 定稿日期: 2019-05-06

only requires symmetric encryption schemes such as AES, while the most common construction for general obfuscation is dependent on the multilinear maps. In other words, the efficiency of the garbled circuits is much higher than that of general obfuscation. Therefore, the distinction and connection between these tools are not clear. This paper introduces garbled circuits and obfuscation, including syntax, security definitions, constructions and applications, analyzes the differences and relations between them and explores the possibility of mutual construction. This paper aims to provide some new ideas for the construction and optimization of these two tools.

Key words: boolean circuits; garbled circuits; obfuscation; indistinguishability obfuscation; reusable garbled circuits

1 引言

随着对称密码和公钥密码的不断发展, 仅仅对消息的加密已经不能满足人类社会的安全需求. 云计算、协同计算等新型计算模型的出现, 也使得对函数或电路的加密变得越来越重要. 在交给代理商进行计算之前, 用户希望不仅仅对输入加密, 对函数本身也进行加密, 从而达到保护函数 "内部信息" 的目的. 例如若医院将疾病的诊断外包, 那么不仅病人的信息涉及隐私, 疾病的诊断方法也是医院需要保护的信息. 因此, 对于函数或电路的加密成为一个新的研究热点, 许多相关的密码学方案也被提出. 其中, 包括混淆、函数加密 (functional encryption)、混淆电路 (又称混乱电路, 加密电路等)、随机编码 (randomized encoding) 等等. 其中, 有两个相近的概念——混淆和混淆电路.

混淆是由程序混淆 [1] 发展而来. 学者们最初为了保护代码安全而提出混淆的概念. 也就是说, 混淆的目的是在保持程序功能的前提下, 使得程序内容 "不可读". 电路混淆或者说函数混淆起始于 Hada 的研究 [2]. Hada 提出的混淆要求混淆后的程序相当于一个黑盒预言机, 除非函数是可学习的, 否则这种混淆的构造是不存在的. 在 2001 年, Barak 等人 [3] 第一次提出了电路混淆的形式化定义, 并给出了可证明的安全定义——虚拟黑盒安全 (virtual black box). 这种安全定义除了电路输出其他信息都不泄露. 但是, Barak 等人在文章中表示, 虚拟黑盒混淆不具有通用构造方案, 即存在一类函数无法达到虚拟黑盒混淆安全. 同篇文章中提出了一种弱化的混淆方案, 称为不可区分混淆 (indistinguishability obfuscation). 不可区分混淆是指两个规模相同、功能相同的电路在混淆后不可区分, 而且保留其功能性. Garg 在 2016 年 [4] 提出第一个利用多线性映射构造通用不可区分混淆的方案. 之后, Bitansky 等人 [5] 和 Ananth 等人 [6] 又分别使用公钥和私钥函数加密构造不可区分混淆. Lin 等人 [7–10] 在此基础上, 利用低维的多线性映射 (最低完成维度为 3 的) 构造函数加密从而完成不可区分混淆的构造. 但是现有文章中构造的多线性映射方案均存在不同程度的安全性问题, 以及效率低下的缺点, 利用安全的密码原语构造高效的通用不可区分混淆方案依然是很多学者的研究重点.

混淆电路最初是由 Yao 在 1986 年 [11] 为解决安全两方计算而提出的一种电路加密技巧. 之后, 由 Goldreich 等人 [12] 将此方案中两个参与者扩展为多个. 但是, 混淆电路一直是以协议中一种处理技巧进行描述, 没有独立的定义和证明. 直到 2008 年, Lindell 等人 [13] 在对 Yao 的方案进行安全证明中第一次提出了混淆电路方案, 将混淆电路提取成一个独立的方案. 之后, 2012 年, Bellare 等人 [14] 给出混淆电路具体的形式化定义, 以及隐私性、健忘性、可靠性等安全定义. 此项工作奠定了混淆电路做为一项独立的密码方案的基础. Hemenway 等人 [15] 在 2016 年提出自适应安全的混淆电路方案. 但是, 这些混淆电路的方案均有一个缺点——不可重用. 可重用的混淆电路是否存在这个问题困扰了学者三十多年, 直到 2013 年, Goldwasser 等人 [16] 利用全同态加密以及基于属性的加密方案构造了第一个可重用的混淆电路方案. 之后, Wang 等人 [17] 又对此进行改进, 舍弃原方案的安全性等级, 利用随机线性码构造的混淆电路替换 Goldwasser 方案中的全同态加密. 然而这些可重用混淆电路方案都不高效, 是否存在更高效的可重用的混淆电路依然是人们关注的问题. 除此之外, 可重用混淆电路是否可以构造混淆也是一个待探索的问题.

从表面来看, 不可区分混淆和混淆电路存在很多相似的地方. 比如说, 在功能上两者都是对电路进行 "加密" 以保护电路安全性为目的, 都要求 "加密" 后的电路保留其功能性. 而在安全定义上, 不可区分混淆的不可区分性和混淆电路基于不可区分安全相似: 均要求功能相同、规模相同的电路在 "加密" 后不可

区分.

但是, 从已有的构造方案来看, 混淆电路仅仅需要对称加密方案和伪随机函数即可完成构造, 而不可区分混淆需要用到多线性映射这种效率不高且安全性不稳定的工具. 根据已有的实现方案对比, 不可区分混淆的方案在对于规模不超过 100 的电路进行混淆需要的时间在小时级别 [18–20], 而混淆电路抵抗半诚实的敌手可以达到每个门的平均加密时间为 10 微秒 [21], 抵抗恶意的敌手可以达到每个门的平均加密时间为 50 毫秒 [22]. 也就是说, 现有的构造方案, 混淆电路的效率远远高于不可区分混淆.

这两者之间的效率差距如此之大, 使得我们开始思考不可区分混淆和混淆电路的区别到底在哪里? 是否能够利用现有的混淆电路去构造不可区分混淆? 如果不能, 屏障又在哪里? 本文即是以这几个问题为出发点, 对不可区分混淆和混淆电路进行研究. 通过对定义、安全要求、应用场景的分析, 得到两者之间的区别与联系. 并试图通过相互构造, 为进一步的研究提供思路.

本文主要综述基于混淆电路和混淆两个密码体制, 探索两个密码体制之间的区别与联系. 本文分为两部分. 第 2–4 节为第一部分, 首先介绍布尔电路, 接下来讲混淆电路和混淆的语义定义、安全定义、构造方案、应用等; 第 5–6 节为第二部分, 主要讲混淆电路与混淆的区别与联系. 首先探索混淆电路与混淆的区别, 从设计初衷、定义、安全要求、构造难度等方面分析, 接下来探索混淆电路和混淆的相互构造并分析难点所在, 为接下来的研究提供新思路.

2 布尔电路简介

计算模型一般分为两类, 一致性模型 (uniform models) 和非一致性模型 (non-uniform models). 一致性模型包括单带图灵机 (single-tape turing machines)、多带图灵机 (multi-tape turing machines) 等, 而非一致性模型包括布尔电路 (boolean circuits)、接受建议的机器 (machines that take advice). 其中布尔电路是最为人们熟知的非一致性模型. 布尔电路通过一些逻辑门操作完成函数的计算功能, 因此混淆和混淆电路对布尔电路的 "加密" 即隐藏电路的门操作种类甚至电路逻辑图. 本节将对布尔电路这一计算模型进行简单的介绍, 包括电路的定义与性质、函数与电路之间的转换、通用电路等.

2.1 电路的定义与性质

布尔电路作为一种非一致性模型, 能够为不同输入规模的同一个函数设计不同的电路. 而一个具体的布尔电路是一个有向无环图, 其利用标准布尔操作与 ($\wedge$)、或 ($\vee$)、非 ($\neg$) 作用到输入的二进制串上产生输出结果. 而对于与和或操作, 要求点的入度为 2, 对于非操作要求点的入度为 1. 其具体定义如下:

定义 1 (布尔电路 [23]) 对任意 $n \in \mathbb{N}$, 一个 n 输入单输出的布尔电路是具有 n 个源顶点和 m 个汇顶点的有向图. 其中:

- 源顶点也称输入顶点, 指的是入度为 0 的顶点. 汇顶点也称输出顶点, 指的是出度为 0 的顶点.
- 每个非源顶点称为一个逻辑门, 并用逻辑操作 $\wedge$ (与)、$\vee$ (或)、$\neg$ (非) 中的一个操作进行标记.
- 顶点的扇入度指的是进入该顶点的边的条数. 标记为 $\wedge$ 和 $\vee$ 的顶点的扇入度等于 2, 而标记为 $\neg$ 的顶点的扇入度等于 1.
- 布尔电路 C 的规模 (或大小) 是 C 中顶点的个数, 记为 $|C|$.
- 如果 C 是一个布尔线路而 $x \in \{0,1\}^n$ 是它的一个输入, 则将 C 在 x 上的输出记为 $C(x)$, $C(x)$ 可以自然的定义如下. 形式上, 我们为 C 中每个顶点 v 定义输出值 $\mathrm{val}(v)$: 如果 v 是第 i 个源顶点, 则定义 $\mathrm{val}(v) = x_i$; 否则, $\mathrm{val}(v)$ 递归的定义为将顶点 v 上的逻辑操作作用于 v 的所有入边关联的顶点的输出值上得到的结果. $C(x)$ 是 C 的输出顶点的值.

在实际构造过程中, 由于非门可以和与门、或门进行合并. 例如电路 $\neg x_1 \wedge x_2$ 可以看作一个门电路 $g(x_1, x_2) = \neg x_1 \wedge x_2$. 如果将电路中的非门均进行此修改, 那么布尔电路中所有逻辑门将被看作一个二元函数 $g : \{0,1\} \times \{0,1\} \to \{0,1\}$. 统一的结构便于设计加密方案, 因此若不做特殊说明, 文章中出现的逻辑门不再局限于与门、或门, 而是看作一个二元函数. 由于输出长度为 m 的电路可以拆分为 m 个子电路. 因此, 在构造方案中一般考虑输出长度为 1 的电路.

布尔电路一旦选定, 有向无环图的结构也被确定下来. 常见的可以使用一个六元组 $f = (n, m, q, A, B, G)$ 表示一个布尔电路 [14]. 其中 $n \geqslant 2$ 表示输入长度, $m \geqslant 1$ 表示输出长度, $q \geqslant 1$ 表示门的个数. 输入顶点有 n 个, 每个门对应一个顶点, 则图中顶点共有 $n+q$ 个. 将输出均用门顶点表示, 因此有向无环图中共有 $r = n+q$ 条线, 令输入 (顶点) 标记为 $\text{Inputs} = \{1, \cdots, n\}$, 线 (边) 标记为 $\text{Wires} = \{1, \cdots, n+q\}$, 输出 (边) 标记为 $\text{OutputWires} = \{n+q-m+1, \cdots, n+q\}$, 门 (顶点) 标记为 $\text{Gates} = \{n+1, \cdots, n+q\}$. A、B、G 均为函数. $A : \text{Gate} \to \text{Wires} \setminus \text{OutputWires}$ 输出每个门的第一条输入线标号. $B : \text{Gate} \to \text{Wires} \setminus \text{OutputWires}$ 输出每个门的第二条输入线标号. $G : \text{Gates} \times \{0,1\}^2 \to \{0,1\}$ 表示门计算, 即表示门的种类.

布尔电路的规模是指其逻辑门的个数, 记作 $|C|$. 另外一个衡量布尔电路计算规模的参数是深度. 布尔电路的深度是指从输入顶点到输出顶点最长一条路的长度, 记作 $\text{depth}(C)$. 电路规模衡量一个电路的大小, 电路深度衡量的是一次计算需要的时间.

2.2 电路与函数

作为非一致性模型, 单个电路很难衡量一个问题的复杂度. 因此, 将不同输入长度的电路放入一个集合中, 以集合为单位即可描述一个任意输入长度的问题. 将不同输入长度的电路放入一个集合中, 这样的形式称为电路簇. 电路簇的形式化定义如下:

定义 2 (电路簇 [24]) 设 $T : \mathbb{N} \to \mathbb{N}$ 是一个函数, $T(n)$ 规模的电路簇是一系列布尔电路 $\{C_n\}_{n\in\mathbb{N}}$, 其中 C_n 有 n 个输入位和一个输出位, 并且其规模 $|C_n| \leqslant T(n)$ 对任意 n 成立.

将同一问题的不同输入规模的电路放入集合也可构造一个电路簇, 这样的电路簇 $\{C_n\}_{n\in\mathbb{N}}$ 可以计算一个函数 $f : \{0,1\}^* \to \{0,1\}^*$. 若函数 f 输入 x, 从电路簇中选取电路 $C_{|x|}$ 用来计算 $C_{|x|}(x)$. 因此, 利用电路簇的规模复杂度可以度量函数的复杂度. 电路簇的规模复杂度定义为函数 $s : \mathbb{N} \to \mathbb{N}$, 其中 $s(n)$ 是电路 C_n 的规模. 函数 f 的电路复杂度 (circuit complexity), 记作 s_f, 是指能计算 f 的最小规模电路簇的规模复杂度. 其形式化定义如下:

定义 3 (电路复杂度 [24]) 函数 $f : \{0,1\}^* \to \{0,1\}^*$ 的电路复杂度定义为函数 $s_f : \mathbb{N} \to \mathbb{N}$, 其中 $s_f(n)$ 表示将 f 限定为 n 比特输入串时, 能计算 f 的最小电路.

根据电路深度和电路规模, 可以将电路簇分为不同的 NC^d 类. 最常用的为 NC^0 和 NC^1 两类, 即深度为常数规模和对数规模的电路. Barrington [25] 提出任意一个深度为 d 的电路都可以转化为一系列大小为 5×5、数量为 4^d 的矩阵相乘形式的矩阵分支程序. 这种方法称为 Barrington 定理. 根据此定理, NC^1 级别的电路可以转化为多项式规模的矩阵分支程序, 将电路的 "加密" 转化为对矩阵的 "加密".

定义 4 (NC 类 [24]) 对于任意 d, 如果存在判定问题 L 的电路簇 $\{C_n\}$ 使得 C_n 的规模为 $\text{poly}(n)$ 且深度为 $\mathcal{O}(\log^d n)$, 则称 L 属于 NC^d 类.

电路簇使得布尔电路对于输入规模的限制放开, 成为了联系一致性模型和非一致性模型的纽带. 如果给定 n, 可以在 $\text{poly}(n)$ 时间内构造出电路 C_n, 那么此多项式规模的电路簇 $\{C_n\}_{n\in\mathbb{N}}$ 被称为是一致的. 一个函数可以由一致的多项式规模电路簇计算, 那么它可以由多项式时间算法计算. 这个算法首先构造出一个电路, 然后对给定输入计算电路的输出. 因此, 多项式规模电路簇可以求解的判定问题属于 $\mathcal{P}/\text{poly}$.

2.3 通用电路

将不同输入规模的电路放入集合中称为电路簇, 那么将相同规模的电路整合为一个电路称为通用电路. 一个通用电路可以计算规模相同的任意一个电路, 通过增加对电路的描述作为输入使得通用电路的输出为不同电路的计算结果. 通用电路与通用图灵机相似, 将函数转化为数据作为程序的输入. 因此, 通用电路的深度问题与图灵机的时空转换问题相关 [26].

若将一个函数的表述加到电路的输入中, 使得电路的输出结果等于函数的计算结果, 这样的电路称为通用电路 (universal circuits). 通用电路可以计算规模相同的任意一个电路. 即对任意一个通用电路 $\text{UC}(x, p)$, 输入任意一个规模为 n 的电路 C 的描述 p 以及此电路的输入 x, 得到 $\text{UC}(x, p) = C(x)$.

valiant 在 1976 年 [26] 提出了一种有效的通用电路构造方案. 有效的是指通用电路的规模和深度是多项式级别扩展的. 而 valiant 的方案对电路规模为 k 的电路构造的通用电路规模为 $\mathcal{O}(k \log k)$, 深度为 $\mathcal{O}(k)$. Cook 等人在 1985 年 [27] 提出了深度是线性的、规模为 $\mathcal{O}(\frac{k^3 d}{\log k})$ 的通用电路构造方案. 之后又不

断有学者提出优化的构造方案 [28–31]. 通用电路可以有效的隐藏计算电路的拓扑结构, 因此在很多密码方案中均有应用.

3 混淆电路

混淆电路对于安全多方计算的意义重大, 而 Yao 构造安全多方计算常用的混淆电路一直被看作一种构造方法. 直到 2012 年, 混淆电路才作为一个独立的密码方案出现, 随之给出其语义定义、安全定义等. 本节将对混淆电路做出详细地介绍. 首先 3.1 节整理对混淆电路的语义定义, 以及选择性安全、适应性安全、基于模拟安全、基于不可区分安全等不同安全级别的安全定义. 此外简单介绍一些混淆电路的性质或扩展. 接下来在 3.2 节说明经典的 Yao 构造混淆电路方案以及一些优化方案. 最后将简述混淆电路的不同应用场景.

3.1 语义定义及安全定义

混淆电路起源于 Yao 在 1982 年和 1986 年 [11,32] 提出的关于安全两方计算协议的构造之中. 在文章中, Yao 构造一个安全两方计算协议来解决著名的大富翁问题, 即有两个大富翁 Alice 和 Bob, 他们想要在不泄露自己具体财富值的情况下得知谁更富有. 这个问题相当于对一个比较两输入大小的电路进行加密, 使得每个人都可以得到最终结果并且无法获取他人的输入. 随后在 1987 年, Glodreich 等人 [12] 具体描述了基于 Yao 的方法构造安全两方计算协议. 而 Beaver 等人 [33] 则在 1990 年首次提出混淆电路的定义. 在 1999 年, Naor 等人 [34] 利用伪随机函数构造混淆电路并应用到隐私保护的拍卖过程中.

2008 年, Lindell 等人 [13] 在证明 Yao 的两方安全计算协议中, 利用形式化的语言描述了混淆电路的构造过程. 2012 年, Bellare 等人 [14] 首次给出严谨的形式化定义和安全性定义. Bellare 等人提出的形式化定义中, 一个混淆电路方案由五个子函数组成 $\mathrm{GC} = (\mathrm{Gb}, \mathrm{En}, \mathrm{Ev}, \mathrm{De}, \mathrm{ev})$, 分别用做混淆电路、加密输入、计算输出、解密输出以及函数计算. 除了混淆电路函数 Gb 和加密输入函数 En 外, 计算函数 Ev 和解密函数 De 经常看作一个函数使用, 而 ev 函数是原始函数在未加密状态下的计算过程. 因此, 学者们普遍使用的混淆电路的定义是简化后的三元组函数 $\mathcal{G} = (\mathrm{Gb}, \mathrm{En}, \mathrm{Ev})$.

定义 5 (混淆电路) 一个混淆电路方案可以看作一个三元组算法 $\mathcal{G} = (\mathrm{Gb}, \mathrm{En}, \mathrm{Ev})$, 每个算法的功能如下:

- $\mathrm{Gb}(1^\lambda, C) \to (F, k)$. 此算法用来对电路 C 进行加密. 输入函数 C 和安全参数 k, 输出 (F, k) 表示加密后的电路 F 以及一个密钥 k.
- $\mathrm{En}(k, x) \to X$. 此算法用来加密输入. 输入函数 C 的输入 x 以及加密密钥 k, 加密得到输入的密文 X.
- $\mathrm{Ev}(F, X) \to Y$. 此算法用来在密文状态下计算电路 C. 输入加密后的函数 F 以及加密后的输入 X, 得到输出 $Y = f(x)$.

此定义要求满足正确性, 即对于任意电路 C 和输入 x, 都有

$$\Pr\left[C(x) = \mathrm{Ev}(F, X) | \mathrm{Gb}(1^\lambda, C) \to (F, k), \mathrm{En}(k, x) \to X\right] = 1 - \mathrm{negl}(k)$$

混淆电路的安全定义分为四种: 基于模拟的选择性安全, 基于不可区分的选择性安全, 基于模拟的自适应安全和基于不可区分的自适应安全 [15]. 基于模拟和基于不可区分是指在敌手区分加密函数的能力, 而选择性安全和自适应安全则是要求敌手是否可以根据加密后电路选取输入. 四种安全定义分别对应一个游戏 SimSel, IndSel, SimAda, IndAda 如图 1 所示. 四种安全的成立要求在对应游戏中, 敌手区分的能力优势可忽略, 即 $\Pr[b' = b] - \frac{1}{2} \leqslant \mathrm{negl}(\lambda)$.

除此之外, 混淆电路还有一些其他相关的性质特点.

首先是关于函数 f 的辅助函数——侧信息函数. 加密的过程中, 虽然电路和输入在符合安全定义情况下都得到了不同程度的保护, 但是不可避免地还是会产生一些关于明文的额外信息的泄露. 我们称这种关于明文的额外信息为侧信息 [14]. 侧信息函数 $\Phi(f)$ 就是用来表示对函数 f 的侧信息泄露. 常用的侧信息

SimSel game:	**IndSel game**:
1. 敌手选择电路 C 以及输入 x, 发送给挑战者; 2. 挑战者随机选取 $b \in \{0,1\}$; 3. 若 $b=0$, 挑战者计算 $\mathrm{Gb}(1^\lambda, C) \to (F,k), \mathrm{En}(k,x) \to X$; 若 $b=1$, 计算 $\mathrm{Sim}C(1^{\lambda,C}) \to (F,k)$, $\mathrm{SimIn}(k, C(x)) \to X$; 4. 挑战者发送 (F,X) 给敌手, 敌手猜测 b'.	1. 敌手选择一对规模相同、功能相同的电路 C_0, C_1 以及对应的输入 x_0, x_1, 满足 $C_0(x_0) = C_1(x_1)$, 并发送给挑战者; 2. 挑战者随机选取 $b \in \{0,1\}$; 3. 挑战者计算 $\mathrm{Gb}\left(1^\lambda, C_b\right) \to (F,k)$, $\mathrm{En}(k, x_b) \to X$; 4. 挑战者发送 (F,X) 给敌手, 敌手猜测 b'.
SimAda game:	**IndAda game**:
1. 敌手选择电路 C 发送给挑战者; 2. 挑战者随机选取 $b \in \{0,1\}$; 3. 若 $b=0$, 挑战者计算 $\mathrm{Gb}(1^\lambda, C) \to (F,k)$; 若 $b=1$, 计算 $\mathrm{Sim}C(1^{\lambda,C}) \to (F,k)$; 4. 挑战者发送 F 给敌手, 敌手选取 x 发送给挑战者; 5. 若 $b=0$, 挑战者计算 $\mathrm{En}(k,x) \to X$; 若 $b=1$, 计算 $\mathrm{SimIn}(k, C(x)) \to X$; 6. 挑战者发送 X 给敌手, 敌手猜测 b'.	1. 敌手选择一对规模相同、功能相同的电路 C_0, C_1 发送给挑战者; 2. 挑战者随机选取 $b \in \{0,1\}$, 计算 $\mathrm{Gb}\left(1^\lambda, C_b\right) \to (F,k)$; 3. 挑战者发送 F 给敌手, 敌手选取 x_0, x_1, 满足 $C_0(x_0) = C_1(x_1)$, 发送给挑战者; 4. 挑战者计算 $\mathrm{En}(k, x_b) \to X$; 5. 挑战者发送 X 给敌手, 敌手猜测 b'.

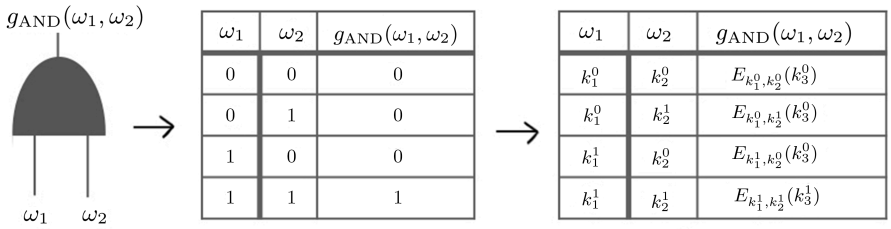
图 1 SimSel, IndSel, SimAda, IndAda 游戏过程
Figure 1 Game of SimSel, IndSel, SimAda and IndAda

函数有三种, 分别为 Φ_{size}, Φ_{topo}, Φ_{circ}. 其中 Φ_{size} 表示泄露函数 f 的电路大小, 包括输入输出长度、门的个数等信息. Φ_{topo} 表示泄露函数 f 的电路拓扑图, 相当于敌手可获取到将输入和输出以及所有门看作图的顶点, 电路的连接线看作图的边所构成的有向无环图. Φ_{circ} 则表示泄露函数 f 的整个电路图, 即仅仅保护输入而电路是公开的. 侧信息函数有利于将加密后所隐藏的信息和泄露的信息形式化的表示, 从而更加清晰地表现了加密函数功能性和信息隐藏的能力. 一般来说, 对于侧信息为 Φ_{topo} 的混淆电路方案可借助于通用电路转化为侧信息为 Φ_{size} 的混淆电路方案.

由于传统的基于 Yao 的加密函数方案都有一个通病, 即混淆电路的一次性. 若多次使用不同的输入计算混淆电路, 那么隐藏的电路信息和输入信息将被泄露. 因此, 有学者提出可重用混淆电路 [16], 即对加密后的电路 F, 可以使用多个不同的输入计算 $\mathrm{En}(k,x) \to X$, 且不泄露电路和输入的信息. 此外, 如果加密输入 X 的每个比特仅与输入 x 的一个比特相关, 那么我们称此混淆电路是投射的 [35].

3.2 常见构造方法

2008 年, Lindell 等人 [13] 在证明 Yao 的安全两方计算协议中, 描述了混淆电路的构造过程. 完整的构造是从加密一个门开始到加密整个电路. 假设 g 是布尔电路 C 的一个门, 那么 g 可以表示为一个函数 $g: \{0,1\} \times \{0,1\} \to \{0,1\}$. 其中两个输入线标记为 ω_1, ω_2, 输出线标记为 ω_3. 随机生成六个密钥 k_1^0, k_1^1, k_2^0, k_2^1, k_3^0, $k_3^1 \in \{0,1\}^\lambda$ 分别表示三条线的输入为 0 和 1 两种情况. 接下来, 门 g 将会利用一个对称加密方案 E 产生四个密文 $c_{0,0}$, $c_{0,1}$, $c_{1,0}$, $c_{1,1}$. 具体构造方法如图 2 所示.



ω_1	ω_2	$g_{\mathrm{AND}}(\omega_1,\omega_2)$
0	0	0
0	1	0
1	0	0
1	1	1

ω_1	ω_2	$g_{\mathrm{AND}}(\omega_1,\omega_2)$
k_1^0	k_2^0	$E_{k_1^0,k_2^0}(k_3^0)$
k_1^0	k_2^1	$E_{k_1^0,k_2^1}(k_3^0)$
k_1^1	k_2^0	$E_{k_1^1,k_2^0}(k_3^0)$
k_1^1	k_2^1	$E_{k_1^1,k_2^1}(k_3^1)$

图 2 Yao 的混淆电路构造方案
Figure 2 Construction of garbled circuits from Yao

将四个密文进行一个随机置换即得到门 g 的加密表 c_0, c_1, c_2, c_3. 在计算门 g 的时候, 利用对应的密钥 k_1^x 和 k_2^y 解密 c_0, c_1, c_2, c_3, 如果某个解密结果不是 $\perp$, 那么得到的结果即为 $k_3^{g(x,y)}$. 有了对一个门的

加密方案之后, 对整个电路进行类似的加密方法, 即可得到整个加密方案. 在 Bellare 等人给出混淆电路的形式化定义后, 对于此种构造方法可利用双密钥加密方案加密, 即可实现基于安全的密码原语构造混淆电路的工作. 这些构造均为选择性安全的. 2016 年, Hemenway 等人 [15] 提出了基于模拟自适应安全的混淆电路, 通过对输入密钥进行对称加密, 完成了由选择性安全到自适应安全的转化. Jafargholi 等人 [36] 在此基础上构造了基于不可区分的自适应安全混淆电路.

四个密文若按照真值表的顺序公布可能会泄露信息. 为了安全着想, 需要将四个密文 c_0, c_1, c_2, c_3 进行随机置换. Beaver 等人 [33] 提出一种 Point-and-permute 技术来优化这一过程. 在每条线的密文 k_i^b 中都设置一个随机的置换比特. k_i^0 和 k_i^1 的置换比特相反且没有对应关系, 即 k_i^b 中置换比特与 b 无关. 而对于四个密文则可以用两个密钥的置换比特进行标记, 这样既不泄露信息而且在解密时仅需解密对应置换比特的密文. 这样的话, 对称加密方案也可替换为一个简单的一次性加密方案 $E_{k_1^x,k_2^y}(k_3^{g(x,y)}) = H(g; k_1^x \parallel k_2^y) \oplus k_3^{g(x,y)}$, 其中 H 是一个带种子的伪随机函数. Naor 等人 [34] 在此基础上提出若适当的选择 k_3^b 可以使得置换比特对应的第一个密文变为全为零的字符串, 这样只需传输三个密文即可.

除此之外, Kolesnikov 和 Schneider 在 2008 年 [37] 提出了 Free-XOR 技术. Free-XOR 技术是选定一个共同的秘密值 $\Delta \in \{0,1\}^\lambda$, 此秘密值全局唯一. 对于每条线的标签 k_i^0 和 k_i^1 的选择需满足 $k_i^0 \oplus k_i^1 = \Delta$, 即 $k_i^b = k_i^0 \oplus b\Delta$. 这种构造方式的优点是对于异或门计算的简化. 若电路中存在异或门 $g = (\omega_1, \omega_2, \omega_3)$, 那么无需再进行加密计算 $E_{k_1^x,k_2^y}(k_3^{g(x,y)})$, 仅需直接计算 $k_3^{g(x,y)} = k_1^x \oplus k_2^y = (k_1^0 \oplus k_2^0) \oplus (x \oplus y)\Delta$ 即可. 因此, 混淆电路仅需对非异或门仅需对称加密即可, 实现了效率上的提高. 这项技术还被 Ball 等人 [38] 应用在算术电路中, 使得在 $\mathbb{Z}_m$ 上加法和数乘的计算优化. 类似于 Free-XOR 技术, 首先选择一个公共秘密值 $\Delta_m \in \mathbb{Z}_m^\lambda$, 将每条线的标签设为 $\boldsymbol{k}_i^b = \boldsymbol{k}_i^0 + b\Delta_m$, 其中 $b \in \mathbb{Z}_m$, $\boldsymbol{k}_i^b$ 是 $\mathbb{Z}_m$ 上的向量. 因此, 对于加法计算满足 $\boldsymbol{k}_1^x + \boldsymbol{k}_2^y = (\boldsymbol{k}_1^0 + \boldsymbol{k}_2^0) + (x+y)\Delta_m$, 对于数乘计算满足 $c\boldsymbol{k}_1^x = c\boldsymbol{k}_1^0 + (cx)\Delta_m$, 使得在算术电路中加法门和数乘门的计算简化.

而 Zahur 等人在 2015 年的 EUROCRYPT 会议上提出了一种名为半门的设计思路 [39], 此方法在 Free-XOR 的基础上减少了与门的密文数量. 假设电路中存在一个与门 $\omega_3 = \omega_1 \wedge \omega_2$, 且利用 Free-XOR 技术给出每条线的标签为 $k_i^0, k_i^1 = k_i^0 \oplus \Delta$ $(i = 1, 2, 3)$. 若按照原来的方案需要至少 3 个密文的传输, 而半门的处理可以使密文数量减少到 2 个. 具体方法如下: 在加密过程中, 生成者随机选择一个比特 r, 那么有 $\omega_3 = \omega_{31} \oplus \omega_{32} = (\omega_1 \wedge r) \oplus (\omega_1 \wedge (r \oplus \omega_2))$. 首先来看 $\omega_{31} = \omega_1 \wedge r$, 生成者已知 r 的取值, 可生成密文 $H(k_1^0) \oplus k_{31}^0$ 和 $H(k_1^1) \oplus k_{31}^0 \oplus r\Delta$. 接下来考虑 $\omega_{32} = \omega_1 \wedge (r \oplus \omega_2)$, 生成者可以公布 $r \oplus \omega_2$ 的标签 $(r \oplus 0, k_2^{r \oplus 0})$ 和 $(r \oplus 1, k_2^{r \oplus 1})$ 给对方 (并不泄露 k_2^b 和 b 之间的关系). 此时计算方可得到对应的 $r \oplus \omega_2$ 的值, 因此生成者产生两个密文 $H(k_2^{r \oplus 0}) \oplus k_{32}^0$ 和 $H(k_2^{r \oplus 1}) \oplus k_{32}^0 \oplus k_2^0$. 当 $r \oplus \omega_2$ 为 0 时, 计算方可根据第一个密文得到 k_{32}^0, 当 $r \oplus \omega_2$ 为 1 时, 计算方可根据第二个密文得到 $k_{32}^0 \oplus k_2^0$, 再与实际输入的 k_2^b 进行异或操作即可得到 k_{32}^b. 由于此方案基于 Free-XOR, 因此使用四个密文计算 ω_{31} 和 ω_{32} 再进行异或操作也可完成对与门 $\omega_3 = \omega_1 \wedge \omega_2$ 的加密, 再利用 Naor 等人的方法 [34] 对每对密文中第一个密文转化为全零, 那么最终只需要传输两个密文即可计算一个与门. 此方法大大提高了与门的加密速度和计算速度.

3.3 应用

混淆电路从 1986 年发展至今, 依然有大量学者为之努力和探索, 可见混淆电路强大的实用性和对密码学的重要性. 2006 年, Malkhi 等人 [21] 根据 Yao 的方案构造了一个安全两方计算系统——Fairplay. 作者对系统进行了测试, 64 比特输入的大富翁游戏需要 254 个门计算, 一次计算需要大约 4 s 时间. 从 Fairplay 中看到混淆电路的计算时间较短, 效率较高, 可应用在很多现实场景中. 此小节将对混淆电路的应用进行综述. 混淆电路起源于安全两方计算, 之后被推广到安全多方计算中去. 除了在安全计算中的应用, 混淆电路还在一次性编程 (one-time program) [40]、可验证计算、同态计算、函数加密甚至不可区分混淆的构造中均有应用.

安全两方计算 安全两方计算分为两种——SFE (secure function evaluation) 和 PFE (private function evaluation). SFE 是指 Alice 和 Bob 两个人均知道函数的功能, 而各自持有一部分输入, 要求在计算过程中不泄露自己持有的输入. PFE 是指 Alice 和 Bob 一方持有函数, 一方持有输入, 要求在计算过程中不泄露自己持有的输入或函数. 由于通用函数的存在, 使得 SFE 协议中构造一个通用电路 UC 是两方均知道的, 再将持有函数的一方的函数描述作为通用电路的输入进行计算. 因此, 对于 PFE 的计算可以归结

到 SFE 的计算上. 最早的 SFE 方案是 Yao 在 1986 年提出的 [11], 利用混淆电路和不经意传输 (oblivious transfer) [41] 完成交互轮数为 4 且敌手是诚实但好奇的安全两方计算要求. 不经意传输是指 A 有一对消息想要传递给 B 其中一个, 但是 B 不希望 A 知道他获取的是哪一个, 而 A 希望 B 仅获取一个消息. 不经意传输和混淆电路构成现有安全两方计算的两大基础工具. SFE 通过混淆电路实现对计算过程的保密而通过不经意传输保证输入的隐私性. Goldwasser 等人 [12] 利用零知识证明系统将 Yao 的方案扩展为可抵抗恶意的敌手. 但是这种转化是低效率的, 因此学者们 [38,42–44] 对利用混淆电路构造更高效的抵抗恶意敌手安全两方计算进行探索. 还有学者为了追求高效, 定义相对弱的安全两方计算协议 [35,45,46]. 除此之外, Wang 等人 [47,48] 提出了带认证的混淆电路方案用以构造高效的抵抗恶意敌手的安全两方计算方案.

其他应用 安全多方计算是安全两方计算的扩展, 因此混淆电路在安全多方计算中也有广泛的应用 [49]. 如: Patra 等人 [50] 构造的安全三方计算, 以及一系列利用混淆电路作为基础工具构造的安全多方计算方案 [44,51–53]. Gennaro 等人 [54] 将混淆电路应用在可验证计算中, 而 Gentry 等人 [55] 则利用混淆电路进行同态计算. 而在公钥加密方案中的 KDM 安全 [56,57] 上, 混淆电路也能助一臂之力. 除了在理论密码学中的应用之外, 混淆电路在现实生活中也有非常广泛的应用. 首先混淆电路可以用在安全文本处理 [58] 中, 其次在隐私保护拍卖 [34] 和隐私信用调查 [59] 中也使用到混淆电路. 特别地, 混淆电路还可以用在隐私医疗诊断 [60,61] 中, 保护病人的隐私.

混淆电路由于其结构简单、快速高效的特点, 不仅在密码学理论中有广泛应用, 还能在现实社会中有效保护个人隐私. 混淆电路的强大功能性和广泛的应用场景引起了学者们的关注, 越来越多的学者为了构造更加高效、更加安全、功能更加强大的混淆电路而开展研究工作.

4 混淆

混淆作为一个强大的密码学工具, 其安全有效的构造一直困扰着密码界. 在通用虚拟黑盒混淆被证明不存在之后, 学者们致力于通用不可区分混淆的构造. 但是, 目前通用不可区分混淆的构造依然面临着巨大问题. 本节将对混淆进行详细的介绍. 首先 4.1 节将概述包括虚拟黑盒混淆和不可区分混淆在内混淆的语义定义和安全定义. 此外还定义一些其他的混淆概念, 包括非一致输入混淆、简洁的混淆等. 4.2 节简述现有的不可区分混淆的通用构造方案, 主要分为用多线性映射直接构造和用函数加密构造两种方法. 最后, 介绍混淆的应用.

4.1 语义定义及安全定义

混淆, 即将任意一个电路在保持其功能的前提下进行 “加密”, 使得混淆后的电路 “不可读”, 达到保护电路内部信息的目的. 也就是说, 混淆相当于将程序或者函数放入一个盒子中, 人们可以 “使用” 这个盒子, 但是不能 “打开” 盒子. 2001 年, Barak 等人 [3] 首次给出了具有虚拟黑盒安全混淆的形式化定义. 虚拟黑盒混淆除了功能性和多项式规模扩展的要求之外, 还要求除了输出其他信息都不泄露. 也就是说, 调用混淆后的程序和使用预言机的效果相同. 这是一个基于断言的安全性极强的定义.

定义 6 (电路混淆) 一个概率多项式时间算法 $\mathcal{O}$ 可以看做是一个电路混淆, 需要满足以下三个条件:

- (功能性) 对任意一个电路 C, $\mathcal{O}(C)$ 表示的电路与电路 C 的功能相同.
- (多项式效率) 对任意一个电路 C, 存在多项式 $\mathcal{P}$ 且满足 $|\mathcal{O}(C)| \leqslant \mathcal{P}(|C|)$.
- (“虚拟黑盒” 特性) 对任意一个概率多项式时间敌手 $\mathcal{A}$, 存在一个概率多项式时间模拟器 $\mathcal{S}$ 和一个可忽略函数 α, 使得对任意一个电路 C 满足

$$|\Pr[\mathcal{A}(\mathcal{O}(C)) = 1] - \Pr[\mathcal{S}^{C}(1^{|C|}) = 1]| \leqslant \alpha(|C|)$$

但是, 在 Barak [3] 和 Goldwasser 的文章中都有指出, 存在一类函数无法达到虚拟黑盒混淆. 也就是说, 不存在通用的虚拟黑盒混淆方案. 为了构造通用的混淆方案, 学者们试图对虚拟黑盒混淆的安全要求放宽, 使得通用构造也保持具有实际意义的安全性. 2007 年, Hohenberger 等人 [62] 提出了一个平均情况虚拟黑盒混淆的定义. 而 Döttling 等人在 2011 年 [63] 提出使用可信硬件中存储全同态加密算法的解密

密钥来构造基于硬件的程序混淆. 但是, 不基于硬件假设的情况下, 直到 2013 年才提出第一个通用混淆的方法.

Garg 等人 [4] 在 2013 年提出了一个弱化的安全性定义——不可区分混淆, 并且给出了第一个通用不可区分混淆的构造方案. 不可区分混淆相较虚拟黑盒混淆, 仅要求对两个功能相同, 规模相同的电路在混淆后不可区分. 不可区分混淆虽然比虚拟黑盒混淆安全性较弱, 但是具有通用构造. 通用不可区分混淆的出现, 使得密码界许多待解决的问题出现了新的解决方案. 因此, 不可区分混淆的强大引得学者们频频回顾.

定义 7 (不可区分混淆) 一个一致的概率多项式算法 $i\mathcal{O}$ 被称为一个对电路簇 $\{\mathcal{C}_\lambda\}$ 的不可区分混淆, 需要满足以下两个条件:

- 对任意安全参数 $\lambda \in \mathbb{N}$, 任意电路 $C \in \mathcal{C}_\lambda$ 以及任意输入 x, 有

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

- 对任意概率多项式区分器 D, 存在一个可忽略函数 α, 满足要求: 对任意安全参数 $\lambda \in \mathbb{N}$, 任意电路对 $C_0, C_1 \in \mathcal{C}_\lambda$, 且满足 $C_0(x) = C_1(x)$, 那么有

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

在构造混淆的过程中, 也构造了一些具有特殊性质的混淆. 非一致输入混淆 (differing-inputs obfuscation) [64] 是不可区分混淆的一种变形, 此定义不再要求两个电路功能相同, 允许两个电路存在相同输入且输出不同, 但是找到这个输入的概率可忽略. 非延展混淆 (non-malleable obfuscation) [65] 是指从一个函数混淆结果不可以得到关于另一个相关函数的混淆. 简洁的 (succinct) 混淆 [66] 是指混淆后的电路规模与原电路规模无关, 仅与输入长度、安全参数等相关.

除了对电路的混淆, 有学者对于其他计算模型的混淆也展开了研究, 包括对图灵机的混淆, 对 RAM 的混淆等等. 对于图灵机的混淆, 由于 Pippenger 等人 [67] 证明任意一个图灵机都可转化为一个电路, 因此对图灵机的混淆将归结到对电路的混淆. 但是 Pippenger 等人的转换方法使得电路规模不仅与图灵机的规模相关, 还与图灵机的计算时间和内存占用相关, 近期的文章 [68–70] 着重于研究构造混淆电路规模仅与图灵机规模相关的方案, 构造常数级乘法界定的图灵机混淆方案, 即混淆后的电路规模仅与原始电路规模线性相关.

4.2 常见构造方法

现有的对于任意 $\mathcal{P}$/poly 电路的通用不可区分混淆的构造方法主要分为两种. 一种是由 Garg 等人 [3] 在 2013 年提出的, 使用多线性映射和分支程序对 NC^1 电路完成不可区分混淆, 再利用自举技术构造对任意多项式级别电路的混淆. 另一种方法是 Bitansky 等人 [5] 和 Ananth 等人 [6] 分别提出的利用公钥的和私钥的函数加密构造对任意多项式级别电路的通用不可区分混淆.

第一种构造方法如图 3 所示, 第一步利用多线性映射等基础工具构造对 NC^1 电路的不可区分混淆. 首先, Barrington [25] 提出任意一个深度为 d 的电路都可以转化为一系列大小为 5×5、数量为 4^d 的矩阵相乘形式的矩阵分支程序. 这种方法称为 Barrington 定理. 使用此方法将 NC^1 电路转化成矩阵相乘的形式, 并对矩阵进行填充随机数和 Kilian 的随机矩阵相乘 [71] 的方法, 实现分支程序随机化. 为了抵抗部分输入攻击, 再进行乘法绑定 (multiply bunding). 最后, 为了抵抗其他非线性攻击手段, 利用多线性映射或者由多线性映射构造的分级编码程序就行编码, 即完成了对任意 NC^1 电路的不可区分混淆方案.

得到 NC^1 电路的不可区分混淆方案后, 再利用自举技术构造对任意多项式级别电路的不可区分混淆. 自举技术如图 4 所示, 利用对全同态加密技术的解密算法进行 NC^1 级别的混淆从而完成对任意多项式电路的混淆. 具体来说, 首先利用全同态加密构造两对公私钥, 并分别对任意多项式规模电路 C 进行加密, 得到 $(g_1, \mathrm{PK}^1, g_2, \mathrm{PK}^2)$. 再构造一个 NC^1 级别的电路 $P^{\mathrm{SK}^2, g_1, g_2}$. 此电路的功能为: 输入密文与非交互式承诺不可区分证明. 如果对此承诺验证通过, 则用 SK^1 解密密文, 否则终止. 对此电路进行 NC^1 级别的混淆得到最终的对任意多项式规模电路 C 的不可区分混淆:

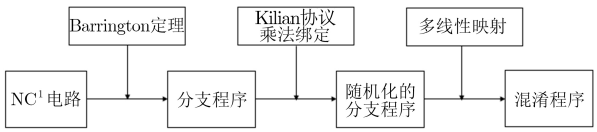


图 3 Garg 提出的通用混淆构造方案

Figure 3 Construction of $i\mathcal{O}$ from Garg

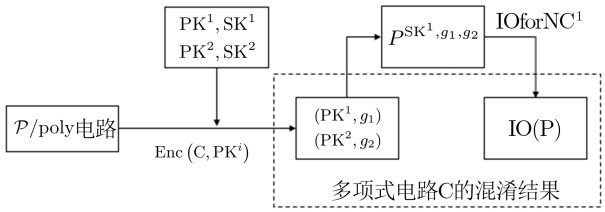


图 4 $\mathcal{P}$/poly 电路混淆过程

Figure 4 Construction of $\mathcal{P}$/poly circuits' obfuscation

另一种方法则是利用简洁的或者紧致的函数加密方案完成对任意多项式电路的通用不可区分混淆的构造. 简洁的是指函数加密中加密电路的大小是 $\mathrm{poly}(n,\lambda)$, 即关于输入电路的规模 n 和安全参数 λ 的多项式规模扩展. Bitansky 和 Vaikuntanathan [5] 利用对称加密方案和简洁的公钥函数加密方案按混淆电路的输入比特数递归定义了对任意多项式规模电路的不可区分混淆构造. 对输入长度为 i 的电路 C_i, 首先利用两次对称加密方案加密电路 C_i 得到 CT^0 和 CT^1, 并构造新函数:

$$f_i(x_i,\mathrm{SK},\beta)=U(\mathrm{Sym.Dec}(\mathrm{SK},\mathrm{CT}_i^{\beta}),x_i)$$

此函数表示先将加密后的电路 CT_i^{β} 进行解密, 再利用通用电路 U 计算 $\mathrm{CT}_i^{\beta}(x)$. 将此函数利用函数加密的密钥生成函数 FE.KeyGen 加密得到 FSK_i. 接下来构造输入为 $i-1$ 比特的函数 $E_{i-1}^0(\boldsymbol{x}_{i-1})$:

$$E_{i-1}^0(\boldsymbol{x}_{i-1})=\{\mathrm{FE.Enc}(\mathrm{PK}_i,((\boldsymbol{x}_{i-1},x_i),\mathrm{SK}_i^0,0));\mathrm{PRF}_{K^i}(\boldsymbol{x}_{i-1},x_i)\}_{x_i\in\{0,1\}}$$

通过将输入 x_i 的最后一位拆分为 0 和 1 两种情况完成递归定义需要接下来混淆的输入长度为 $i-1$ 比特的函数. 这种构造方法是递归定义的, 因此要求函数加密是简洁的.

而 Ananth 和 Jain [6] 则是利用紧致的公钥函数加密方案构造对任意多项式电路的不可区分混淆方案. 紧致的 (compact) 是指函数加密的加密函数的运行时间必须是关于安全参数和输入长度多项式时间. 第一步利用归纳的思想由 c 元输入的函数加密方案 (FE_c) 构造 $c+1$ 元输入的函数加密方案 (FE_{c+1}). 利用一个紧致的公钥函数加密 FE 的 KeyGen 函数加密 $c+1$ 元函数 f, 生成 $\mathrm{FE}_{c+1}.\mathrm{sk}_f$. 并构造电路 G 如图 5 所示. 若 FE_{c+1}.Enc 函数接收到输入为 $(\mathrm{msk},x,1)$ 时, 利用 FE_c.KeyGen 函数加密电路 G 得到 $\mathrm{FE}_c.\mathrm{sk}_G$; 若 FE_{c+1}.Enc 函数接收到输入为 (msk,x,i) 且 $2\leqslant i\leqslant c+1$ 时, 利用 c 元函数加密的 FE_c.Enc 函数加密 $(x,x,1,\tau,i)$ 即可. 由此可构造任意元输入的函数加密方案, 再通过 Goldwasser 等人 [72] 的方法构造不可区分混淆. 首先通过以输入比特长度为单位递归构造多输入函数加密方案. 多输入的函数加密方案如果满足输入为 $n+1$ 元, 那么对输入为 n 比特的电路 C 混淆, 只需构造通用电路 $U(x_1,\cdots,x_n,C)=C(x_1,\cdots,x_n)$ 并将 $x_1,\cdots,x_n$ 和 C 当作多输入函数加密的 $n+1$ 元输入进行加密.

由于紧致的和简洁的函数加密方案的构造都需要用到多线性映射. 其他不可区分混淆的构造方案也均需要多线性映射的帮助. 因此目前通用不可区分混淆的构造依然离不开多线性映射. 而多线性映射在 2003 年由 Boneh 和 Silverberg 提出 [73]. 之后相继提出的多线性映射方案 GGH13 [74]、CLT13 [75]、GGH15 [76] 等, 不仅效率不高, 安全性还受到了不同程度的攻击 [77–81]. 因此是否存在安全高效的多线性映射方案还不得而知 [82,83]. 除此之外, 利用 Garg 等人的方法需要的多线性映射维度为多项式级别的, 降低多线性映射维度也成为学者优化不可区分混淆构造的一个方向. 2015 年, Zimmerman 提出对算术电路

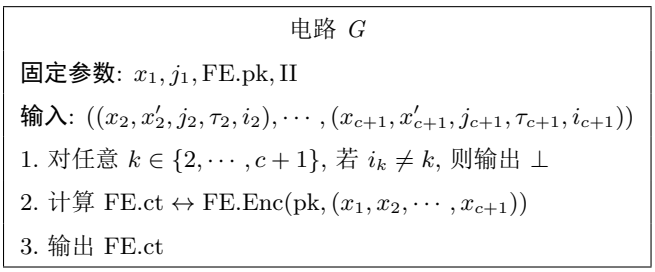


图 5 电路 G 的具体内容
Figure 5 Details of circuit G

直接进行混淆 [66], 从而避免了分支程序带来的多线性映射维度的增加, 使得多线性映射的操作数量从深度 d 指数级别降为多项式级别. 2017 年, Lin 等人 [7–10] 将不可区分混淆的构造需要的多线性映射维度从多项式降到常数阶, 之后通过构造特别的伪随机生成器使得多线性映射维度降低到 3. 寻找高效安全的三线性映射去构造不可区分混淆称为新的问题. 三线性映射甚至多线性映射的发展也直接影响到不可区分混淆的发展. 因此, 越来越多的学者试图跨越从三线性映射到双线性映射的鸿沟或者抛弃之前的构造方法, 利用已有的安全的密码原语, 寻求更有效可行的构造方案.

4.3 应用

混淆之所以受到如此广泛的关注和研究, 是因为混淆其强大的功能性. 混淆不仅仅是实现了保护函数 "内容" 的目的, 还解决了许多困扰密码学界多年的问题. 本节将对不可区分混淆在密码学中的应用进行介绍, 包括对于函数加密方案, 多方密钥协商方案, 自线性对方案等的构造.

对通用电路构造安全的函数加密方案一直是一个未解决困难问题, 大多数的构造仅能实现内积加密函数这样简单电路的函数加密方案. 但是, Garg 在提出第一个通用不可区分混淆方案的文章中就提出了利用不可区分混淆构造对于任意多项式规模电路的函数加密方案. 由于函数加密要求对函数输入的保护, 此方案需要构造一个特殊函数 $P^{f,\mathrm{sk}_1,\mathrm{CRS}}(e_1, e_2, \pi) = f(x)$ 用以解密加密后的输入并进行计算. 此函数将计算函数 f, 一个公钥加密方案的私钥 sk_1 以及零知识证明的参数 CRS 作为固定参数, 输入使用公钥 pk_1 加密输入 x 的密文 e_1 和公钥 pk_2 加密输入 x 的密文 e_2 以及一个零知识证明 π, 此证明表明 e_1 和 e_2 是合法构造的. 当函数 P 得到输入后, 先验证证明 π 是否有效, 若有效则使用私钥 sk_1 解密 e_1 得到输入 x, 最后计算 $f(x)$. 将函数 P 混淆后的程序作为函数加密函数 $\mathrm{KeyGen}(\mathrm{msk}, f)$ 的输出 sk_f, 而函数加密函数 $\mathrm{Enc}(\mathrm{mpk}, x)$ 则使用公钥加密 x 得到 e_1 和 e_2 并构造其零知识证明 π 作为输出 $c = (e_1, e_2, \pi)$. 这样就保证了函数加密需要的对输入加密的安全要求. 因此, 使用不可区分混淆可以构造选择性不可区分安全的函数加密方案.

非交互无需可信第三方的多方密钥协商方案也是密码学的一个难题. 2014 年 Boneh 等人 [84] 提出了使用混淆构造无需可信第三方非交互的多方密钥协商方案, 并且此方案每个成员公布的信息大小与总成员数量无关. 同样地, 此构造也依赖于构造一个特殊的函数 P_E. 此函数内置一个伪随机函数 PRF, 在输入成员集合 S, 成员 i 公布的信息 pk_i (一个签名方案的公钥) 以及一个签名 σ 之后, 首先验证签名正确性, 若验证通过, 那么输出 $\mathrm{PRF}(S, \{\mathrm{pk}_j\})$ 作为协商的密钥. 每个成员 i 除了要生成自己的签名私钥 sk_i 和公布的签名公钥 pk_i 之外, 还需要公布对函数 P_E 的混淆程序 $i\mathcal{O}(P_E)$. 在计算协商密钥时, 成员 j 使用自己的私钥 sk_j 对 S 进行签名 σ_i, 选取同一个 $i\mathcal{O}(P_E)$ (一般选择最小号成员生成的混淆程序) 计算得到协商密钥. 这样就完成了无需可信第三方的多方密钥协商方案的构造.

此外, Boneh 在此文章中还利用类似的技术实现了支持分布式初始化的广播加密方案, 以及抗勾结的叛徒追踪系统等方案的构造. 2014 年, Sahai 等人 [85] 利用混淆构造了可否认加密方案 (deniable encryption). 同年, Garg 等人 [86] 利用不可区分混淆构造两轮的安全多方计算模型. 从这些应用可以看出, 不可区分混淆在解决一些问题的关键在于构造一个定制的函数, 将需要隐藏的信息作为函数的固定参数或者利用函数判定某些性质, 再完成计算. 将此定制的函数进行混淆, 达到隐藏信息和打包的作用.

除了以上提到的应用方法之外, 2016 年, Yamakawa 等人 [87] 利用不可区分混淆构造自线性对, 从而构造安全的多线性映射. 与其他应用不同的是, 自线性对的构造利用的是不可区分混淆中两个功能相同、

规模相同的函数在混淆后不可区分的关系. 利用 Blum 整数 N 构造一个群 Q_N^+, 在未知整数 N 分解的情况下无法求取群的阶从而无法有效计算映射函数 $e(g^x, g^y) \to g^{cxy}$. 因此构造一对功能相同的函数 τ_y 和 τ'_y, 其中 $\tau_y \leftarrow e(\cdot, y)$ 而 $\tau'_y \leftarrow e(\cdot, y + \text{order})$. 由于 order 未知因此 τ'_y 可以保护 y 不被泄露, 而 τ_y 则在 order 未知的情况下可有效构造. 借助于不可区分混淆使得 $i\mathcal{O}(\tau_y)$ 和 $i\mathcal{O}(\tau'_y)$ 不可区分, 方案即可使用 $i\mathcal{O}(\tau_y)$ 既保证 y 的安全又能够有效构造. 借助于不可区分混淆, Yamakawa 等人解决了自线性对的构造问题.

可以看出, 不可区分混淆在密码学中应用广泛. 借助于其不可区分性以及对函数整体的打包和保护, 不可区分混淆解决了很多密码学中困扰许久的问题.

5 混淆电路与混淆的关系

从前两节的介绍中可以看到, 不可区分混淆和混淆电路在安全定义上有很多相似之处. 除此之外, 两者在密码学领域都占有重要地位, 具有强大的功能, 为其他密码方案的构造提供了有效的工具. 但是, 从构造中我们也发现, 两种方案的构造方式天差地别, 并且效率也相差甚远. 因此, 我们在本节着重分析两个密码工具的区别与联系, 探究其中的差距到底在哪里. 在 5.1 节, 我们分析传统 Yao 方式的混淆电路与通用混淆即不可区分混淆的关系. 接下来, 我们将在 5.2 节探索可重用混淆电路与不可区分混淆的相互构造.

5.1 混淆电路与混淆的区别

在之前的介绍中, 不可区分混淆和混淆电路有着很多相似之处, 在构造的过程中, 都需要对函数进行“加密”, 使得敌手无法从加密后的函数中获取自己想要的信息. 而在安全性定义上, 不可区分混淆的不可区分性和加密函数基于不可区分的安全性 (包括选择性和自适应的), 都是要求对两个功能相同、规模相同的函数任意选取一个进行混淆或加密, 再让敌手进行猜测区分. 不可区分混淆和混淆电路从表面上看似乎可以看作一个密码工具, 但是在现实的研究中又具有很大的差别. 到底不可区分混淆和混淆电路的区别在哪里呢?

接下来我们将从设计初衷、定义、安全要求、构造难度四个方面进行对比. 目的是分析不可区分混淆和传统混淆电路的根本区别, 为将来的密码研究探索新思路.

设计初衷 从设计初衷上来说, 混淆最初是由程序混淆、代码混淆发展而来, 混淆电路则是从安全多方计算所衍生出的一个密码方案. 混淆最初的目的是保护程序和代码内容的安全. 经过近十年的演变, 不可区分混淆的主要作用也是保护电路或者函数的内部信息不被泄露. 对比混淆电路, 他最初的目的是保护输入信息, 由于输入信息加密而衍生的需要在加密条件下进行电路计算. 在混淆电路的三十年发展中, 其主要作用是为了保护每一位参与者的输入安全, 使参与者之间的输入互不可知. 因此, 可以明显看出不可区分混淆必须加密和保护的是电路本身的安全和隐私, 而混淆电路必须加密和保护的是输入信息的安全.

定义 从定义上, 不可区分混淆只有两个函数 Obf 和 Eval, 分别用来混淆程序和计算输出. 而混淆电路则由三个函数组成 Gb、En、Ev, 分别拥有加密电路、加密输入、计算输出的功能. 函数中加密电路函数 Gb 可以看作和不可区分混淆的混淆函数 Obf 相似, 计算函数 Ev 和不可区分混淆的计算函数 Eval 相似. 剩下的函数是用来加密输入. 也就是说, 从定义上看到, 加密函数比不可区分函数在加密电路和保证加密电路的功能之外, 还增加了对输入的加密.

安全要求 从安全要求上, 由于传统的 Yao 的构造是选择性安全的构造, 因此我们将不可区分混淆和混淆电路基于不可区分选择性安全都统一为图 6 所示的游戏进行比较. 不可区分混淆的不可区分性要求在此游戏后, 敌手猜测成功的优势是一个可忽略函数, 混淆电路的基于不可区分的隐私性同样的要求在此游戏后, 敌手猜测成功的优势是一个可忽略函数.

对比两个游戏的过程, 在电路的选择、计算、交互上是相同的. 两个游戏均是由敌手先选择两个功能相同、规模相同的电路 C_0 和 C_1, 并将这两个电路一起发送给挑战者. 挑战者在收到两个电路之后, 先随机生成一个比特 $b \in \{0, 1\}$, 并对对应的电路 C_b 进行操作 (不可区分混淆运行混淆函数 Obf, 混淆电路运行加密函数 Gb), 在加密或者混淆完成后将结果返回给敌手. 敌手收到之后可以进行电路的计算 (不可区分混淆运行计算函数 Eval, 加密函数运行计算函数 Ev), 并猜测挑战者选取的比特 b. 如果敌手不能以超过 $\frac{1}{2}$ 的概率猜测正确, 则说明此方案是安全的.

不可区分混淆的不可区分性	加密电路基于不可区分的选择性安全
1. 敌手选择一对规模相同、功能相同的电路 C_0, C_1 发送给挑战者; 2. 挑战者随机选取 $b \in \{0,1\}$; 3. 挑战者计算 $i\mathcal{O}(C_b) \to P$; 4. 挑战者将 P 发送给敌手; 5. 敌手可以对任意输入 x, 计算 $P(x)$; 6. 敌手猜测 b'.	1. 敌手选择一对规模相同、功能相同的电路 C_0, C_1, 以及对应的输入 x_0, x_1, 满足 $C_0(x_0) = C_1(x_1)$, 并发送给挑战者; 2. 挑战者随机选取 $b \in \{0,1\}$; 3. 挑战者计算 $\mathrm{Gb}(1^\lambda_b) \to (F,k), \mathrm{En}(k,x_b) \to X$; 4. 挑战者发送 (F,X) 给敌手, 敌手猜测 b'.

图 6 不可区分混淆和加密电路的安全定义
Figure 6 Security definition of $i\mathcal{O}$ and garbled circuits

但是可以看到在两个游戏中对电路输入的操作是不同的. 不可区分混淆的输入是在挑战者返回混淆后的程序之后, 由敌手任意次选择输入并带入电路进行计算. 而 Yao 的混淆电路的输入则是要求敌手必须在发送两个电路 C_0 和 C_1 之前选择好并发送给挑战者, 由挑战者统一进行加密返回给敌手. 在此方面上, 混淆更接近适应性安全的混淆电路. 此外, 敌手只能使用发送给挑战者的输入对加密后的电路进行计算, 不能再选择新的输入. 也就是说, 不可区分混淆完成的混淆程序在敌手任意多次选择输入并计算之后, 仍然无法区分. 混淆电路加密后的电路则是限制在一次输入计算过程中不可区. 如果对加密后的电路进行多次不同输入的计算, 我们可以通过局部输入攻击 (partial evaluation attacks) [4] 进行区分.

构造难度 从构造难度上, 不可区分混淆需要多线性映射或者分级编码系统构造, 混淆电路需要双密钥密码进行构造. 2017 年, Halevi 等人 [18] 实现了基于 GGH15 多线性映射的不可区分混淆方案. 在文章中, 分支函数长度为 20 时, 混淆时间已经超过四万秒. 虽然在之后 Lin 等人 [7–10] 的工作使得不可区分混淆方案可以基于三线性映射构造, 但是一方面还没有相关实现展示其具体效率, 另一方面没有突破多线性映射的禁锢. 由于多线性映射至今无法确定是否存在安全的构造, 现有的构造都收到不同程度的攻击并且效率极低, 因此不可区分混淆的实现效率也很低. 而在 2004 年 Dahlia 等人 [88] 基于 Yao 的混淆电路实现的安全多方计算平台 Fairplay 上, 64 比特输入、电路规模为 256 的大富翁游戏仅需要 4 秒时间. 在使用 Free-XOR、半门、row-reduction 的操作后, 混淆电路的效率进一步得到提升. 可以看到混淆电路比起不可区分混淆来看更加高效.

综上所述, 我们可以看到不可区分混淆和混淆电路两个概念还是存在相当大的区别. 不可区分混淆构造要求更高, 效率较低, 但是对电路的安全性保护更加全面. 混淆电路的构造要求低, 效率较高, 对输入也有不可区分混淆没有的加密机制, 但是对电路本身的安全性保护较低, 在实际使用过程中受限, 混淆电路不可重复使用.

5.2 混淆电路与混淆的联系

本节将探讨混淆电路与混淆之间的联系, 试图通过混淆构造可重用混淆电路, 并探索构造混淆所需的混淆电路的安全等级. 需要说明的是, 我们期望构造的混淆电路和混淆均为通用构造, 特殊函数的构造暂不考虑. 因此, 我们探索的是混淆电路与通用混淆即不可区分混淆之间的关系. 通过相互构造, 期望发现其相互关系, 为混淆电路和混淆的发展提供思路.

由 5.1 节的讨论可知混淆电路与混淆最大的障碍在于可重用性, 也就是说对于同一个混淆电路, 可以实现多组不同输入的计算并保证其安全性. Goldwasser 等人 [16] 给出第一个可重用混淆电路的构造. 可重用混淆电路是在原有的混淆电路方案中加入可重用, 即在加密函数 En 中, 要求可以选择不同的输入进行加密. 同时在使用同一个加密后的电路 F 时不泄露电路和输入信息. 其形式化定义如下:

定义 8 (可重用混淆电路) 一个混淆电路方案 $\mathrm{uGC} = (\mathrm{uGC.Gb}, \mathrm{uGC.En}, \mathrm{uGC.Ev})$ 被称为可重用混淆电路, 那么:

- $\mathrm{uGC.Gb}(1^\lambda, C) \to (F, \mathrm{sk})$. 此算法用于对输入的电路 C, 在安全参数 λ 下加密得到加密后的电路 F 以及一个密钥 sk.
- $\mathrm{uGC.En}(\mathrm{sk}, x) \to X$. 此算法利用密钥 sk 将电路 C 的一个输入 x 进行加密, 得到 X. 对于同一

个 C 加密后得到 F 和 sk, 可以利用 sk 构造多组输入 x 的加密输入 X, 称为可重用的.

- uGC.Ev$(F, X) \to Y$. 此算法对加密后的电路 F 以及输入 X 进行计算, 得到 Y, 且 Y 的值等于 $C(x)$.

Goldwasser 等人 [16] 使用全同态加密和基于属性的加密方案构造了可重用混淆电路. 文章中使用的基于属性的加密方案 ABE = (ABE.Setup, ABE.KeyGen, ABE.Enc, ABE.Dec) 与一般的 ABE 方案不同. 普通的 ABE 方案, 在解密过程断言为真则解密, 断言为假则停机. 而 Goldwasser 所使用的 ABE 是加密两个消息 m_0 和 m_1, 若断言为真则解密 m_1, 反之则解密 m_0. 使用这种 ABE 方案和 Yao 的混淆电路方案 GC 结合, 即可解密 Yao 输入密钥对的其中一个. 而断言则是使用全同态加密方案 FHE = (FHE.KeyGen, FHE.Enc, FHE.Eval, FHE.Dec) 的计算函数判定. 为保证函数安全性, 额外选取一个对称加密方案 Sym = (Sym.KeyGen, Sym.Enc, Sym.Dec) 将函数加密即可. 具体构造方法如算法 1 所示.

算法 1 可重用混淆电路构造过程

uGC.Gb$(\mathbf{1^\lambda}, \boldsymbol{C})$:

1. $(\text{fmpk}_i, \text{fmsk}_i) \leftarrow \text{ABE.Setup}(1^\lambda)$ for $i \in [k]$;
2. $\text{sk} \leftarrow \text{Sym.KeyGen}(1^\lambda)$;
3. $E \leftarrow \text{Sym.Enc}(\text{sk}, C)$;
4. 构造电路 U_E. U_E 输入密钥 sk 和值 x, 首先计算 $C \leftarrow \text{Sym.Dec}(\text{sk}, E)$, 然后计算并返回 $C(x)$;
5. 定义 n 表示电路 U_E 的输入长度;
6. 定义 $\text{FHE.Eval}^i_{U_E}(\text{hpk}, \psi_1, \cdots, \psi_n)$ 表示 U_E 在 $(\psi_1, \cdots, \psi_n)$ 输入下同态计算得到密文的第 i 比特;
7. 定义 k 表示 FHE 的一个密文长度, 即 $k = |\psi|$;
8. $\text{fsk}_i \leftarrow \text{ABE.KeyGen}(\text{fmsk}_i, \text{FHE.Eval}^i_f)$ for $i \in [k]$;
9. 返回 (Γ, gsk), 其中 $\Gamma = (\{\text{fsk}_i\}_{i\in[k]})$, $\text{gsk} = (\{\text{fmpk}_i\}_{i\in[k]}, \text{sk})$uGC.En$(\text{gsk}, \boldsymbol{x})$:
1. 拆分 $\text{gsk} = (\{\text{fmpk}_i\}_{i\in[k]}, \text{sk})$, 那么电路 U_E 的输入为 $x' = (x'_1, \cdots, x'_n) = (\text{sk}||x)$;
2. $(\text{hpk},\text{hsk}) \leftarrow \text{FHE.KeyGen}(1^\lambda)$;
3. $\psi_i \leftarrow \text{FHE.Enc}(\text{hpk}, x'_i)$ for $i \in [n]$;
4. $(\Gamma', \{L^0_i, L^1_i\}^k_{i=1}) \leftarrow \text{Gb.Gb}(1^\lambda, \text{FHE.Dec}(\text{hsk}, \cdot))$;
5. $c_i \leftarrow \text{ABE.Enc}(\text{fmpk}_i, (\text{hpk}, \{\psi_j\}_{j\in[n]}), L^0_i, L^1_i)$ for $i \in [k]$;
6. 返回 $c = (c_1, \cdots, c_k, \Gamma')$uGC.Ev$(\boldsymbol{\Gamma}, \boldsymbol{c})$:
1. $L^{d_i}_i \leftarrow \text{ABE.Dec}(\text{fsk}_i, c_i)$ for $i \in [k]$;
2. $Y = \text{Gb.Eval}(\Gamma', L^{d_1}_1, \cdots, L^{d_k}_k) = \text{FHE.Dec}(\text{hsk}, d_1, \cdots, d_k)$;
3. 返回 Y.

若我们希望找到通用混淆和可重用混淆电路的关系, 就需要对现有的可重用混淆电路进行改造. 首先, 目前存在的通用混淆方案为不可区分混淆方案. 不可区分混淆在两个功能相同规模相同的电路对 C_0, C_1 满足: $\Pr[b' = b | b' \leftarrow \mathcal{A}(i\mathcal{O}.\text{obf}(C_b))] - \frac{1}{2} \leqslant \text{negl}(\lambda)$. 因此, 我们需要定义基于不可区分安全性的可重用混淆电路. 可重用混淆电路的基于不可区分的自适应安全定义在一个游戏 uAdaInd. 游戏过程为: 敌手任意的选择一对规模相同、功能相同的电路 C_0, C_1, 将此电路对发送给挑战者. 挑战者随机选取 $b \in \{0, 1\}$, 计算 $\text{uGC.Gb}(1^\lambda, C_b) \to (F_b, \text{sk})$, 得到后将 F_b 发送给敌手. 敌手根据 F_b, 任意的循环以下步骤: 选择一个输入 x, 询问挑战者, 挑战者计算 $\text{uGC.En}(\text{sk}, x) \to X$ 并返回 X 给敌手, 计算 $\text{uGC.Ev}(F_b, X) \to Y$. 最后敌手猜测 b'. 若 $\Pr[b' = b] - \frac{1}{2} \leqslant \text{negl}(\lambda)$, 则称此混淆电路方案满足基于不可区分的自适应安全.

有了可重用混淆电路的定义和构造, 我们尝试使用不可区分混淆构造满足可重用性的混淆电路方案. 假设存在一个不可区分混淆 $i\mathcal{O} = (i\mathcal{O}.\text{obf}, i\mathcal{O}.\text{Eval})$, 以及一个对称加密方案 Sym = (Sym.KeyGen, Sym.Enc, Sym.Dec), 构造一个可重用混淆电路方案 uGC:

- $\text{uGC.Gb}(1^\lambda, C)$: 计算 $\text{sk} \leftarrow \text{Sym.KeyGen}(1^\lambda)$. 构造电路 $C_{\text{sk},C}(X)$. 电路 $C_{\text{sk},C}(X)$ 首先计算 $x \leftarrow \text{Sym.Dec}(\text{sk}, X)$, 然后计算并返回 $C(x)$. 将电路 $C_{\text{sk},C}(X)$ 混淆得到 $\Gamma \leftarrow i\mathcal{O}.\text{obf}(1^\lambda, C_{\text{sk},C})$. 返回 (Γ, sk).
- $\text{uGC.En}(\text{sk}, x)$: 计算 $X \leftarrow \text{Sym.Enc}(\text{sk}, x)$. 返回 X.
- $\text{uGC.Ev}(\Gamma, X)$: 计算 $Y \leftarrow i\mathcal{O}.\text{Eval}(\Gamma, X)$. 返回 Y.

如此构造的可重用混淆电路方案是满足基于不可区分性的自适应性安全的. 即可构造由基于不可区

分性自适应安全的可重用混淆电路到不可区分混淆的规约. 对任意一个不可区分混淆的挑战者, 敌手首先选择两个功能相同、规模相同的电路 C_0 和 C_1, 计算 $\mathrm{sk} \leftarrow \mathrm{Sym.KeyGen}(1^\lambda)$. 接下来, 利用电路 C_0, C_1 以及 sk 构造两个新的功能相同且规模相同的电路 $C_{\mathrm{sk},C_0}(X)$, $C_{\mathrm{sk},C_1}(X)$. 敌手将 $C_{\mathrm{sk},C_0}(X)$, $C_{\mathrm{sk},C_1}(X)$ 作为挑战电路对发送给不可区分混淆的挑战者得到一个回答 $i\mathcal{O}(C_{\mathrm{sk},C_b}(X))$. 此回答也是对于基于不可区分性的自适应安全的可重用混淆电路方案的挑战密文. 若对此密文可区分那么对不可区分混淆的密文也可区分, 即此可重用混淆电路方案是满足基于不可区分性的自适应性安全的.

接下来, 我们将探索使用混淆电路构造不可区分混淆的方法. 从 3.1 节可以看到, 传统的混淆电路是选择性安全的, 且是 "一次性" 的. 因此, 学者们构造了可重用的混淆电路. 那么, 可重用混淆电路与不可区分混淆又是否相等呢?

首先回顾一下 Goldwasser 等人 [16] 提出的可重用混淆电路的安全定义. 可重用混淆电路的安全定义是基于模拟的, 称为可重用的输入和电路隐私性 (input and circuit privacy with reusability). 其具体定义如下.

定义 9 (可重用的输入和电路隐私性) 令 uGC 是电路簇 $\{C_n\}_{n\in\mathbb{N}}$ 的可重用混淆电路方案. 对于一对概率多项式时间算法 $\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)$ 以及一个概率多项式时间模拟器 $\mathcal{S}=(\mathcal{S}_1,\mathcal{S}_2)$ 来说, 考虑图 7 的两个游戏:

$\mathrm{Exp}^{\mathrm{real}}_{\mathrm{uGC},\mathcal{A},\mathcal{S}}(1^k)$	$\mathrm{Exp}^{\mathrm{ideal}}_{\mathrm{uGC},\mathcal{A},\mathcal{S}}(1^k)$
1. $(C,\mathrm{state}_A)\leftarrow\mathcal{A}_1(1^k)$	1. $(C,\mathrm{state}_A)\leftarrow\mathcal{A}_1(1^k)$
2. $(\mathrm{gsk},\Gamma)\leftarrow\mathrm{uGC}(1^k,C)$	2. $(\Gamma',\mathrm{state}_{\mathcal{S}})\leftarrow\mathcal{S}_1(1^k,1^{\lvert C\rvert})$
3. $\alpha\leftarrow\mathcal{A}_2^{\mathrm{uGCEn}(\mathrm{gsk},\cdot)}(C,\Gamma,\mathrm{state}_A)$	3. $\alpha\leftarrow\mathcal{A}_2^{O(\cdot,C)[t[\mathrm{state}_\delta]]}(C,\Gamma',\mathrm{state}_A)$
4. Output α	4. Output α

图 7 可重用输入和电路隐私性游戏

Figure 7 Game of input and circuit privacy with reusability

其中 $O(\cdot,C)[[\mathrm{state}_S]]$ 是一个运行 $\mathcal{S}_2$ 的预言机, 输入 x, $C(x)$, $1^{|x|}$ 以及 S 的状态, 返回 $\mathcal{S}_2$ 的输出结果. 如果在这两个游戏中, $\{\mathrm{Exp}^{\mathrm{real}}_{\mathrm{uGC},\mathcal{A}}(1^k)\}_{k\in\mathbb{N}}$ 与 $\{\mathrm{Exp}^{\mathrm{ideal}}_{\mathrm{uGC},\mathcal{A},\mathcal{S}}(1^k)\}_{k\in\mathbb{N}}$ 两个分布是计算不可区分的, 那么称此可重用混淆电路方案满足可重用的输入和电路隐私性.

从安全定义来看, 可重用混淆电路的安全性是基于模拟的自适应安全. 类比混淆来说, 相当于虚拟黑盒混淆的安全定义. 但是由于其加密函数 En 是使用私钥 sk 进行计算的, 和不需要限制使用明文进行计算的混淆不同. 而 Goldwasser 等人的文章中 [17] 有这样一段话: "加密输入需要私钥是必须的. 如果可以公开加密, 那么敌手的能力将和传统的混淆相同. 而这种混淆已经被证明不存在通用构造. " 这段话从侧面印证了如果在此安全定义下构造不需要私钥的可重用混淆电路方案, 就相当于完成了通用虚拟黑盒混淆的构造.

由于基于模拟的自适应安全也可满足此定义下的基于不可区分的自适应安全, 所以现有的基于模拟的自适应安全的可重用混淆电路均满足基于不可区分的自适应安全. 但是, 是否存在不需要私钥的可重用混淆电路方案暂时是一个未解之谜. 即敌手在获取到 F_b 后可自行计算函数, 无需再向挑战者获取输入 x 的加密密文 X.

假设存在不需要私钥的可重用混淆电路方案, 那么构造不可区分混淆将是简单的. 假设存在一个满足基于不可区分自适应性安全的不需要私钥的混淆电路方案 uGC. 构造一个不可区分混淆方案如下:

- $i\mathcal{O}.\mathrm{obf}(1^\lambda,C)$: 计算 $\mathrm{uGC.Gb}(1^\lambda,C)\rightarrow(F,\mathrm{pk})$, 返回 $O=(F,\mathrm{pk})$.
- $i\mathcal{O}.\mathrm{Eval}(x,O)$: 分离 $O=(F,\mathrm{pk})$, 计算 $\mathrm{uGC.En}(\mathrm{pk},x)\rightarrow X$, $\mathrm{uGC.Ev}(F,X)\rightarrow Y$, 返回 Y.

综上可知, 利用不可区分混淆构造基于不可区分的自适应安全的可重用混淆电路方案是简单的, 而利用基于不可区分的自适应安全的可重用混淆电路方案构造不可区分混淆则是未知的. 通过探索可知, 构造不可区分混淆相当于构造一个不需要私钥的基于不可区分自适应性安全的混淆电路方案. 因此, 利用混淆电路方案构造混淆也许是一个新的发展方向.

6 总结

混淆电路与混淆作为对函数本身的保护是密码研究领域两个非常重要的工具. 混淆电路在安全多方计算、同态计算、可验证计算等方面均有应用, 而混淆也在可否认加密、广播加密、自线性映射等方面均有应用. 他们不仅在密码学理论研究中起到重要作用, 在现实生活中也有广泛的应用空间. 由于混淆电路和混淆均是对电路的保护, 将来可能应用到人类社会的创作、发明、作品等技术保护. 但是, 混淆电路和混淆都还存在很多问题值得被探索和研究.

首先, 混淆电路特别是可重用混淆电路方案的有效实现暂时缺失. 可重用混淆电路相较于不可重用的方案来说大大提高了构造的复杂度, 在现实应用过程中也降低了构造速率. 但是现有的相关可重用混淆电路的研究依然很少, 仅仅停留在理论分析上. 目前可重用混淆电路在摆脱了同态加密的重负后, 使用的方案仅基于 LWE、随机线性码以及对称加密方案. 但是安全性也大大降低. 可重用混淆电路的效率以及安全参数的选择依然是一个待探索的领域. 此外, 是否存在更加有效的可重用混淆电路的构造方案或优化方案也是一个待探索的方向. 又或者, 是否存在效率接近不可重用的混淆电路而重用次数有限次的混淆电路方案? 关于可重用混淆电路, 依然等待着学者们的探索与发现.

其次, 混淆的有效构造依然困扰着密码界. 传统的基于多线性映射的混淆由于多线性映射的不安全性导致构造的混淆方案不安全, 是否存在安全有效的多线性映射方案依然是迷. 而现有的 GGH13、CTL13 以及 GGH15 均被攻破, 构造多线性映射的问题停滞不前. 对于使用较为成熟的双线性对方案构造混淆, 近两年成为学者们努力的目标, 现有的研究显示有完成的可能性. 但是这种构造方式的效率和参数也将成为下一个关心的问题, 以及这些构造是否存在可攻击的漏洞也未可知. 对于不可区分混淆的构造和实现依然值得深入研究.

最后, 关于混淆电路和混淆之间的关系, 通过本文的探索, 问题纠结在于是否存在不需要私钥的基于不可区分的自适应性安全可重用混淆电路方案. 是否可以通过现有的基于模拟的可重用混淆电路方案进行修改, 得到基于不可区分的方案并解放私钥. 又或者对现有的可重用混淆电路作为基础, 结合密码学其他工具解放私钥是否可以完成混淆的构造? 而这种构造方式相较于现有的混淆构造是否能够提高效率? 此外, 关于可重用混淆电路的构造, 可否利用对具体函数的虚拟黑盒混淆进行构造. 由于现有的对具体函数的混淆效率相较通用混淆较高, 因此尝试使用对具体函数的混淆来构造可重用混淆电路或许可行.

虽然混淆电路和混淆还有很长的路要走, 但是这两个密码方案已经显示其强大的功能. 相信经过无数学者的努力和奋斗, 终有一天我们会突破这些难题, 让混淆电路和混淆为人类社会安全添砖加瓦.

References

[1] GU W W, HUANG G F, LIAO M D. The application of obfuscation in cryptographic protocols[J]. Netinfo Security, 2017, (9): 81–84. [DOI: 10.3969/j.issn.1671-1122.2017.09.019]
顾微微, 黄桂芳, 廖茂东. 代码混淆在密码协议中的应用 [J]. 信息网络安全, 2017, (9): 81–84. [DOI: 10.3969/j.issn.1671-1122.2017.09.019]

[2] HADA S. Zero-knowledge and code obfuscation[C]. In: Advances in Cryptology—ASIACRYPT 2000. Springer Berlin Heidelberg, 2000: 443–457. [DOI: 10.1007/3-540-44448-3_34]

[3] BARAK B. GOLDREICH O, IMPAGLIAZZO R, et al. On the (im)possibility of obfuscating programs[C]. In: Advances in Cryptology—CRYPTO 2001. Springer Berlin Heidelberg, 2001: 1–18. [DOI: 10.1007/3-540-44647-8_1]

[4] GARG S, GENTRY C, HALEVI S, et al. Candidate indistinguishability obfuscation and functional encryption for all circuits[J]. SIAM Journal on Computing, 2016, 45(3): 882–929. [DOI: 10.1137/14095772X]

[5] BITANSKY N, VAIKUNTANATHAN V. Indistinguishability obfuscation from functional encryption[C]. In: Proceedings of 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS). IEEE, 2015: 171–190. [DOI: 10.1109/FOCS.2015.20]

[6] ANANTH P, JAIN A. Indistinguishability obfuscation from compact functional encryption[C]. In: Advances in Cryptology—CRYPTO 2015, Part I. Springer Berlin Heidelberg, 2015: 308–326. [DOI: 10.1007/978-3-662-47989-6_15]

[7] LIN H J. Indistinguishability obfuscation from constant-degree graded encoding schemes[C]. In: Advances in Cryptology—EUROCRYPT 2016, Part I. Springer Berlin Heidelberg, 2016: 28–57. [DOI: 10.1007/978-3-662-

49890-3_2]
[8] LIN H, VAIKUNTANATHAN V. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings[C]. In: Proceedings of 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS). IEEE, 2016: 11–20. [DOI: 10.1109/FOCS.2016.11]
[9] LIN H J. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs[C]. In: Advances in Cryptology—CRYPTO 2017, Part I. Springer Cham, 2017: 599–629. [DOI: 10.1007/978-3-319-63688-7_20]
[10] LIN H, TESSARO S. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs[C]. In: Advances in Cryptology—CRYPTO 2017, Part I. Springer Cham, 2017: 630–660. [DOI: 10.1007/978-3-319-63688-7_21]
[11] YAO A C C. How to generate and exchange secrets[C]. In: Proceedings of 1986 IEEE 27th Annual Symposium on Foundations of Computer Science. IEEE, 1986: 162–167. [DOI: 10.1109/SFCS.1986.25]
[12] GOLDREICH O, MICALI S, WIGDERSON A. How to play any mental game[C]. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing. ACM, 1987: 218–229. [DOI: 10.1145/28395.28420]
[13] LINDELL Y, PINKAS B. A proof of security of Yao's protocol for two-party computation[J]. Journal of Cryptology, 2009, 22(2): 161–188. [DOI: 10.1007/s00145-008-9036-8]
[14] BELLARE M, HOANG V T, ROGAWAY P. Foundations of garbled circuits[C]. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security. ACM, 2012: 784–796. [DOI: 10.1145/2382196.2382279]
[15] HEMENWAY B, JAFARGHOLI Z, OSTROVSKY R, et al. Adaptively secure garbled circuits from one-way functions[C]. In: Advances in Cryptology—CRYPTO 2016, Part III. Springer Berlin Heidelberg, 2016: 149–178. [DOI: 10.1007/978-3-662-53015-3_6]
[16] GOLDWASSER S, KALAI Y, POPA R A, et al. Reusable garbled circuits and succinct functional encryption[C]. In: Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing. ACM, 2013: 555–564. [DOI: 10.1145/2488608.2488678]
[17] WANG Y, MALLUHI Q M, KHAN K M D. Garbled computation in cloud[J]. Future Generation Computer Systems, 2016, 62: 54–65. [DOI: 10.1016/j.future.2015.11.004]
[18] HALEVI S, HALEVI T, SHOUP V, et al. Implementing BP-obfuscation using graph-induced encoding[C]. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017: 783–798. [DOI: 10.1145/3133956.3133976]
[19] APON D, HUANG Y, KATZ J, et al. Implementing cryptographic program obfuscation[J]. IACR Cryptology ePrint Archive, 2014: 2014/779.
[20] ZHANG Z, ZHANG F, ZHANG H. Implementing indistinguishability obfuscation using GGH15[C]. In: Information Security and Cryptology—INSCRYPT 2017. Springer Cham, 2017: 27–43. [DOI: 10.1007/978-3-319-75160-3_4]
[21] HUANG Y, EVANS D, KATZ J, et al. Faster secure two-party computation using garbled circuits[C]. In: Proceedings of 20th USENIX Security Symposium. USENIX, 2011: 331–335.
[22] PINKAS B, SCHNEIDER T, SMART N P, et al. Secure two-party computation is practical[C]. In: Advances in Cryptology—ASIACRYPT 2009. Springer Berlin Heidelberg, 2009: 250–267. [DOI: 10.1007/978-3-642-10366-7_15]
[23] ARORA S, BARAK B. Computational Complexity: A Modern Approach[M]. Cambridge University Press, 2009.
[24] GOLDREICH O. Computational complexity: A conceptual perspective[J]. In: ACM SIGACT News, 2008, 39(3): 35–39. [DOI: 10.1145/1412700.1412710]
[25] BARRINGTON D A. Bounded-width polynomial-size branching programs recognize exactly those languages in NC1[J]. Journal of Computer and System Sciences, 1989, 38(1): 150–164. [DOI: 10.1016/0022-0000(89)90037-8]
[26] VALIANT L G. Universal circuits (preliminary report). In: Proceedings of ACM Symposium on Theory of Computing (STOC 1976). ACM, 1976: 196–203. [DOI: 10.1145/800113.803649]
[27] COOK S A, HOOVER H J. A depth-universal circuit[J]. SIAM Journal on Computing, 1985, 14(4): 833–839. [DOI: 10.1137/0214058]
[28] GÜNTHER D, KISS Á, SCHNEIDER T. More efficient universal circuit constructions. In: Advances in Cryptology—ASIACRYPT 2017, Part II. Springer Cham, 2017: 443–470. [DOI: 10.1007/978-3-319-70697-9_16]
[29] KISS Á, SCHNEIDER T. Valiant's universal circuit is practical[C]. In: Advances in Cryptology—EUROCRYPT 2016, Part I. Springer Berlin Heidelberg, 2016: 699–728. [DOI: 10.1007/978-3-662-49890-3_27]
[30] KOLESNIKOV V, SCHNEIDER T. A practical universal circuit construction and secure evaluation of private functions[C]. In: Financial Cryptography and Data Security—FC 2008. Springer Berlin Heidelberg, 2008: 83–97. [DOI: 10.1007/978-3-540-85230-8_7]
[31] YU Y, LEIWO J, PREMKUMAR B. Hiding circuit topology from unbounded reverse engineers[C]. In: Information

Security and Privacy—ACISP 2006. Springer Berlin Heidelberg, 2006: 171–182. [DOI: 10.1007/11780656_15]

[32] YAO A C. Protocols for secure computations[C]. In: Proceedings of 23rd Annual Symposium on Foundations of Computer Science (SFCS 1982). IEEE, 1982: 160–164. [DOI: 10.1109/SFCS.1982.38]

[33] BEAVER D, MICALI S, ROGAWAY P. The round complexity of secure protocols[C]. In: Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing. ACM, 1990: 503–513. [DOI: 10.1145/100216.100287]

[34] NAOR M, PINKAS B, SUMNER R. Privacy preserving auctions and mechanism design[C]. In: Proceedings of the 1st ACM Conference on Electronic Commerce. ACM, 1999: 129–139. [DOI: 10.1145/336992.337028]

[35] AUMANN Y, LINDELL Y. Security against covert adversaries: Efficient protocols for realistic adversaries[C]. In: Theory of Cryptography Conference—TCC 2007. Springer Berlin Heidelberg, 2007: 137–156. [DOI: 10.1007/978-3-540-70936-7_8]

[36] JAFARGHOLI Z, SCAFURO A, WICHS D. Adaptively indistinguishable garbled circuits[C]. In: Theory of Cryptography Conference—TCC 2017, Part II. Springer Cham, 2017: 40–71. [DOI: 10.1007/978-3-319-70503-3_2]

[37] KOLESNIKOV V, SCHNEIDER T. Improved garbled circuit: Free XOR gates and applications[C]. In: Automata, Languages and Programming—ICALP 2008, Part II. Springer Berlin Heidelberg, 2008: 486–498. [DOI: 10.1007/978-3-540-70583-3_40]

[38] BALL M, MALKIN T, ROSULEK M. Garbling gadgets for Boolean and arithmetic circuits[C]. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016: 565–577. [DOI: 10.1145/2976749.2978410]

[39] ZAHUR S, ROSULEK M, EVANS D. Two halves make a whole[C]. In: Advances in Cryptology—EUROCRYPT 2015, Part II. Springer Berlin Heidelberg, 2015: 220–250. [DOI: 10.1007/978-3-662-46803-6_8]

[40] GOLDWASSER S, KALAI Y T, ROTHBLUM G N. Delegating computation: Interactive proofs for muggles[C]. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. ACM, 2008: 113–122. [DOI: 0.1145/1374376.1374396]

[41] WU P F, SHEN Q N, QIN J, et al. Survey of oblivious RAM[J]. Journal of Software, 2018, 29(9): 2753–2777. [DOI: 10.13328/j.cnki.jos.005591]
吴鹏飞, 沈晴霓, 秦嘉, 等. 不经意随机访问机研究综述 [J]. 软件学报, 2018, 29(9): 2753–2777. [DOI: 10.13328/j.cnki.jos.005591]

[42] LINDELL Y, PINKAS B. An efficient protocol for secure two-party computation in the presence of malicious adversaries[C]. In: Advances in Cryptology—EUROCRYPT 2007. Springer Berlin Heidelberg, 2007: 52–78. [DOI: 10.1007/978-3-540-72540-4_4]

[43] MOHASSEL P, ROSULEK M. Non-interactive secure 2PC in the offline/online and batch settings[C]. In: Advances in Cryptology—EUROCRYPT 2017, Part III. Springer Cham, 2017: 425–455. [DOI: 10.1007/978-3-319-56617-7_15]

[44] LINDELL Y. Fast cut-and-choose-based protocols for malicious and covert adversaries[C]. In: Advances in Cryptology—CRYPTO 2013, Springer Berlin Heidelberg, 2013: 1–17. [DOI: 10.1007/978-3-642-40084-1_1]

[45] MOHASSEL P, FRANKLIN M. Efficiency tradeoffs for malicious two-party computation[C]. In: Public Key Cryptography—PKC 2006. Springer Berlin Heidelberg, 2006: 458–473. [DOI: 10.1007/11745853_30]

[46] GOYAL V, MOHASSEL P, SMITH A. Efficient two party and multi party computation against covert adversaries[C]. In: Advances in Cryptology—EUROCRYPT 2008. Springer Berlin Heidelberg, 2008: 289–306. [DOI: 10.1007/978-3-540-78967-3_17]

[47] WANG X, RANELLUCCI S, KATZ J. Authenticated garbling and efficient maliciously secure two-party computation[C]. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017: 21–37. [DOI: 10.1145/3133956.3134053]

[48] KATZ J, RANELLUCCI S, ROSULEK M, et al. Optimizing authenticated garbling for faster secure two-party computation[C]. In: Advances in Cryptology—CRYPTO 2018, Part III. Springer Cham, 2018: 365–391. [DOI: 10.1007/978-3-319-96878-0_13]

[49] XU B, PENG C G, GU C X. Secure multiparty computation protocol with fairness[J]. Computer Engineering, 2012, (7): 116–118. [DOI: 10.3969/j.issn.1000-3428.2012.07.038]
徐滨, 彭长根, 顾崇旭. 公平的安全多方计算协议 [J]. 计算机工程, 2012, (7): 116–118. [DOI: 10.3969/j.issn.1000-3428.2012.07.038]

[50] PATRA A, RAVI D. On the exact round complexity of secure three-party computation[C]. In: Advances in Cryptology—CRYPTO 2018, Part II. Springer Cham, 2018: 425–458. [DOI: 10.1007/978-3-319-96881-0_15]

[51] KAMARA S, MOHASSEL P, RAYKOVA M. Outsourcing multi-party computation[J]. IACR Cryptology ePrint Archive, 2011: 2011/272.

[52] GOLDREICH O. Secure multi-party computation[J]. Manuscript. Preliminary version, 1998: 78.
[53] HASTINGS M, HEMENWAY B, NOBLE D, et al. SoK: General purpose compilers for secure multi-party computation[C]. In: Proceedings of 2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019, Vol. 1: 479–496. [DOI: 10.1109/SP.2019.00028]
[54] GENNARO R, GENTRY C, PARNO B. Non-interactive verifiable computing: Outsourcing computation to untrusted workers[C]. In: Advances in Cryptology—CRYPTO 2010. Springer Berlin Heidelberg, 2010: 465–482. [DOI: 10.1007/978-3-642-14623-7_25]
[55] GENTRY C, HALEVI S, VAIKUNTANATHAN V. A simple BGN-type cryptosystem from LWE[C]. In: Advances in Cryptology—EUROCRYPT 2010. Springer Berlin Heidelberg, 2010: 506–522. [DOI: 10.1007/978-3-642-13190-5_26]
[56] BARAK B, HAITNER I, HOFHEINZ D, et al. Bounded key-dependent message security[C]. In: Advances in Cryptology—EUROCRYPT 2010. Springer Berlin Heidelberg, 2010: 423–444. [DOI: 10.1007/978-3-642-13190-5_22]
[57] APPLEBAUM B. Key-dependent message security: Generic amplification and completeness[C]. In: Advances in Cryptology—EUROCRYPT 2011. Springer Berlin Heidelberg, 2011: 527–546. [DOI: 10.1007/978-3-642-20465-4_29]
[58] KATZ J, MALKA L. Secure text processing with applications to private DNA matching[C]. In: Proceedings of the 17th ACM Conference on Computer and Communications Security. ACM, 2010: 485–492. [DOI: 10.1145/1866307.1866361]
[59] PAUS A, SADEGHI A R, SCHNEIDER T. Practical secure evaluation of semi-private functions[C]. In: Applied Cryptography and Network Security—ACNS 2009. Springer Berlin Heidelberg, 2009: 89–106. [DOI: 10.1007/978-3-642-01957-9_6]
[60] BARNI M, FAILLA P, KOLESNIKOV V, et al. Secure evaluation of private linear branching programs with medical applications[C]. In: Computer Security—ESORICS 2009. Springer Berlin Heidelberg, 2009: 424–439. [DOI: 10.1007/978-3-642-04444-1_26]
[61] JAGADEESH K A, WU D J, BIRGMEIER J A, et al. Deriving genomic diagnoses without revealing patient genomes[J]. Science, 2017, 357(6352): 692–695. [DOI: 10.1126/science.aam9710]
[62] HOHENBERGER S, ROTHBLUM G N, VAIKUNTANATHAN V. Securely obfuscating re-encryption[C]. In: Theory of Cryptography Conference—TCC 2007. Springer Berlin Heidelberg, 2007: 233–252. [DOI: 10.1007/978-3-540-70936-7_13]
[63] DÖTTLING N, MIE T, MÜLLER-QUADE J, et al. Basing obfuscation on simple tamper-proof hardware assumptions[J]. IACR Cryptology ePrint Archive, 2011: 2011/675.
[64] ANANTH P, BONEH D, GARG S, et al. Differing-inputs obfuscation and applications[J]. IACR Cryptology ePrint Archive, 2013: 2013/689.
[65] CANETTI R, VARIA M. Non-malleable obfuscation[C]. In: Theory of Cryptography Conference—TCC 2009. Springer Berlin Heidelberg, 2009: 73–90. [DOI: 10.1007/978-3-642-00457-5_6]
[66] ZIMMERMAN J. How to obfuscate programs directly[C]. In: Advances in Cryptology—EUROCRYPT 2015, Part II. Springer Berlin Heidelberg, 2015: 439–467. [DOI: 10.1007/978-3-662-46803-6_15]
[67] PIPPENGER N, FISCHER M J. Relations among complexity measures[J]. Journal of the ACM (JACM), 1979, 26(2): 361–381. [DOI: 10.1145/322123.322138]
[68] ANANTH P, JAIN A, SAHAI A. Indistinguishability obfuscation for Turing machines: Constant overhead and amortization[C]. In: Advances in Cryptology—CRYPTO 2017, Part II. Springer Cham, 2017: 252–279. [DOI: 10.1007/978-3-319-63715-0_9]
[69] AGRAWAL S, MAITRA M. FE and IO for Turing machines from minimal assumptions[C]. In: Theory of Cryptography Conference—TCC 2018, Part II. Springer Cham, 2018: 473–512. [DOI: 10.1007/978-3-030-03810-6_18]
[70] GARG S, SRINIVASAN A. A simple construction of iO for Turing machines[C]. In: Theory of Cryptography Conference—TCC 2018, Part II. Springer Cham, 2018: 425–454. [DOI: 10.1007/978-3-030-03810-6_16]
[71] KILIAN J. Founding cryptography on oblivious transfer[C]. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. ACM, 1988: 20–31. [DOI: 10.1145/62212.62215]
[72] GOLDWASSER S, GORDON S D, GOYAL V, et al. Multi-input functional encryption[C]. In: Advances in Cryptology—EUROCRYPT 2014. Springer Berlin Heidelberg, 2014: 578–602. [DOI: 10.1007/978-3-642-55220-5_32]
[73] BONEH D, SILVERBERG A. Applications of multilinear forms to cryptography[J]. Contemporary Mathematics, 2003, 324(1): 71–90. [DOI: 10.1090/conm/324/05731]
[74] GARG S, GENTRY C, HALEVI S. Candidate multilinear maps from ideal lattices[C]. In: Advances in

Cryptology—EUROCRYPT 2013. Springer Berlin Heidelberg, 2013: 1–17. [DOI: 10.1007/978-3-642-38348-9_1]

[75] CORON J S, LEPOINT T, TIBOUCHI M. Practical multilinear maps over the integers[C]. In: Advances in Cryptology—CRYPTO 2013. Springer Berlin Heidelberg, 2013: 476–493. [DOI: 10.1007/978-3-642-40041-4_26]

[76] GENTRY C, GORBUNOV S, HALEVI S. Graph-induced multilinear maps from lattices[C]. In: Theory of Cryptography Conference—TCC 2015. Springer Berlin Heidelberg, 2015: 498–527. [DOI: 10.1007/978-3-662-46497-7_20]

[77] LEE H T, SEO J H. Security analysis of multilinear maps over the integers[C]. In: Advances in Cryptology—CRYPTO 2014, Part I. Springer Berlin Heidelberg, 2014: 224–240. [DOI: 10.1007/978-3-662-44371-2_13]

[78] CHEON J H, HAN K, LEE C, et al. Cryptanalysis of the multilinear map over the integers[C]. In: Advances in Cryptology—EUROCRYPT 2015, Part I. Springer Berlin Heidelberg, 2015: 3–12. [DOI: 10.1007/978-3-662-46800-5_1]

[79] GENTRY C, HALEVI S, MAJI H K, et al. Zeroizing without zeroes: Cryptanalyzing multilinear maps without encodings of zero[J]. IACR Cryptology ePrint Archive, 2014: 2014/929.

[80] HU Y, JIA H. Cryptanalysis of GGH map[C]. In: Advances in Cryptology—EUROCRYPT 2016, Part I. Springer Berlin Heidelberg, 2016: 537–565. [DOI: 10.1007/978-3-662-49890-3_21]

[81] CORON J S, LEE M S, LEPOINT T, et al. Cryptanalysis of GGH15 multilinear maps[C]. In: Advances in Cryptology—CRYPTO 2016, Part II. Springer Berlin Heidelberg, 2016: 607–628. [DOI: 10.1007/978-3-662-53008-5_21]

[82] ZHANG F G. From bilinear pairings to multilinear maps[J]. Journal of Cryptologic Research, 2016, 3(3): 211–228. [DOI: 10.13868/j.cnki.jcr.000122]
张方国. 从双线性对到多线性映射 [J]. 密码学报, 2016, 3(3): 211–228. [DOI: 10.13868/j.cnki.jcr.000122]

[83] CHENG R, ZHANG F G. An overview on the secure program obfuscation[J]. Netinfo Security. 2014, (8): 6–16. [DOI: 10.3969/j.issn.1671-1122.2014.08.002]
成荣, 张方国. 安全的程序混淆研究综述 [J]. 信息网络安全, 2014, (8): 6–16. [DOI: 10.3969/j.issn.1671-1122.2014.08.002]

[84] BONEH D, ZHANDRY M. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation[J]. Algorithmica, 2017, 79(4): 1233–1285. [DOI: 10.1007/s00453-016-0242-8]

[85] SAHAI A, WATERS B. How to use indistinguishability obfuscation: Deniable encryption, and more[C]. In: Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing. ACM, 2014: 475–484. [DOI: 10.1145/2591796.2591825]

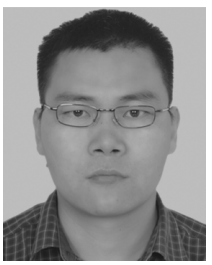
[86] GARG S, GENTRY C, HALEVI S, et al. Two-round secure MPC from indistinguishability obfuscation[C]. In: Theory of Cryptography Conference—TCC 2014. Springer Berlin Heidelberg, 2014: 74–94. [DOI: 10.1007/978-3-642-54242-8_4]

[87] YAMAKAWA T, YAMADA S, HANAOKA G, et al. Self-bilinear map on unknown order groups from indistinguishability obfuscation and its applications[J]. In: Advances in Cryptology—CRYPTO 2014, Part II. Springer Berlin Heidelberg. 2014: 90–107. [DOI: 10.1007/978-3-662-44381-1_6]

[88] MALKHI D, NISAN N, PINKAS B, et al. Fairplay—A secure two-party computation system[C]. In: Proceedings of the 13th USENIX Security Symposium. USENIX, 2004, 13.

作者信息

张正 (1994–), 河南平顶山人, 博士生在读. 主要研究领域为混淆的实现与优化.
zhangzh65@mail2.sysu.edu.cn



张方国 (1972–), 山东淄博人, 博士, 教授, 博士生导师, 中国密码学会常务理事. 主要研究领域为密码学理论及其应用, 特别是椭圆曲线密码体制、安全多方计算、可证明安全等.
isszhfg@mail.sysu.edu.cn