山 东 大 学

# 博 士 学 位 论 文 开 题 报 告

## 题 目：安全多方计算及其应用

培 养 单 位 网络空间安全学院（研究院）

学科（专业） 网络空间安全

导 师 陈宇

研 究 生 涂彬彬

学 号 202020881

开题报告日期 2021.11.16

研究生院制

二〇一五年六月

# 目录

# 1 研究背景和意义

安全多方计算（secure multi-party computation，MPC）旨在解决一组互不信任的参与方之间保护隐私的协同计算问题，在整个计算过程中，除了由输入输出可以推测出的信息外，没有更多额外的信息泄漏。安全多方计算的安全性可保证数据拥有方的隐私信息安全，安全多方计算的功能性可以为数据需求方提供数据之间的共享流通和计算。因此，安全多方计算能在数据拥有方和需求方之间，实现数据的可信互联互通，打破“数据孤岛”限制，解决数据安全性、隐私性问题，大幅降低数据信息交易成本，提升数据使用价值，为数据拥有方和需求方提供有效的对接渠道，构建互惠互利的交互服务网络。

门限密码系统是安全多方计算技术的关键性应用。对于密码系统而言，密钥是实现密码功能操作的关键。然而，传统公钥密码的密钥唯一，密钥的安全性完全依靠用户的秘密保存，一旦用户的密钥丢失，不仅难以恢复，还容易形成单点故障问题导致密码系统无法正常运行，甚至恶意敌手截获密钥后，可任意窃取用户隐私信息或者伪造用户身份。门限密码系统完美体现了“不要把所有的鸡蛋放在一个篮子里”的思想，将传统公钥密码的集中化权限进行分散，保证了多用户共同分享密码操作权限，并以一种多方安全计算的方式进行密码操作，使得多个用户分布式协同完成密码操作。门限密码系统可避免传统公钥密码由于密钥唯一而导致的单点故障问题，对于提升密码系统的健壮性和安全性等有着重要作用。在已有的研究中，理论上存在通用的方法解决一般性的安全多方计算问题，即对现有的密码系统采用通用的安全多方计算技术进行分布式扩展，构建安全的门限密码系统，但是 Goldreich [1] 指出直接应用现有的通用方法来解决具体的问题是低效且不切合实际的。传统密码系统的分布式研究需要兼顾密码系统的应用环境、实现功能、效率、安全性等方面，需要定制的特殊的安全多方计算协议。

目前，安全多方计算研究在基础理论和实际应用上已经取得了较多的成果，在门限密码系统的应用也日渐丰富。但是，随着网络技术的不断发展，特别是云计算、物联网、人工智能、区块链等分布式场景对于隐私保护和数据安全需求的不断提升。无论是在基础理论研究方面，还是在应用协议研究方面，安全多方计算都有待进一步的研究和探讨。例如，通用的安全多方计算协议的计算效率和通信效率需要继续优化；实用的安全模型和安全性分析方法还有待深入研究；应用问题的多样性，特别是对于具体的门限密码系统需要设计更多的特定安全协议来完成具体计算；网络技术的发展也使得安全多方计算不断地面临着新的应用需求。

综上所述，安全多方计算能够在数据不离开数据持有节点的前提下，完成数

据的分析、处理和结果发布，能提供数据访问权限控制和数据交换的一致性保障，使得在确保参与各方利益前提下的协作计算成为可能，其研究对于打破“数据孤岛”，促进分布式网络环境下的信息交流与数据共享产生巨大的推动。此外，安全多方计算技术能对传统密码系统进行分布式扩展，细化密码操作的访问控制权限，防止单点故障问题，提升密码系统的健壮性和安全性，适用于当前分布式应用系统的应用需求。由于网络环境的复杂性和应用问题的多样性，安全多方计算研究尚不充分，较多问题有待进一步的研究和探讨。因此，对于基础安全多方计算协议展开研究，提升应用效率，加强安全性，同时针对特定的应用问题，特别是针对标准密码算法的进行分布式设计，满足门限密码系统的应用需求，不仅具有较重要的理论意义，而且对于我国经济和社会领域具有广泛的应用前景。

## 2 国内外的研究现状及分析

1982 年，Yao [2] 首次提出安全多方计算的思想，用“百万富翁问题”的实例描述了两方安全计算问题，即两个百万富翁在不泄露自己财产数额的前提下，如何比较出谁更富有？该问题可以抽象为：拥有秘密输入的双方，如何在保证输入的隐秘性和结果正确性的同时完成数值大小比较的计算任务。随后，Goldreich 等 [3] 将两方安全计算概念推广到多方安全计算，彻底打开了安全多方计算研究的大门。目前对于安全多方计算的研究较为广泛，本文主要针对理论优化和应用创新两个方面进行研究：一方面是探索安全多方计算基础理论优化，包括对安全多方计算基础协议的构建和完善，以及对设计通用安全多方计算协议的方法论的研究；另一方面的研究是将安全多方计算与实际应用问题相结合，为具体的分布式应用设计兼顾效率与安全的协议，特别是安全多方计算技术在门限密码系统中的应用，比如对现有标准密码算法的门限化设计，满足当前分布式应用系统对于门限密码的迫切需求。

### 2.1 MPC 协议的研究现状

安全多方计算的基础理论是安全多方计算应用协议研究的理论基石，具体包括协议的存在性、协议的计算模型，协议安全性、协议公平性以及协议构造方法等方面的内容。1987 年 Goldreich 等 [3] 提出一种通用安全计算协议的构造方法，攻克了安全多方计算的存在性难题，为安全多方计算后期研究工作奠定了坚实的基础。1998 年，Goldreich [1] 对安全多方计算的概念、计算模型、半诚实模型下的安全性定义以及通用协议的描述进行全面概括。

目前，通用安全多方计算协议设计主要基于乱码电路、不经意传输、秘密分享、同态加密等技术构造。Yao [2] 首次提出乱码电路技术并结合不经意传输，给

出了第一个通用的两方安全计算协议——Yao 协议。该方法将要计算的函数转换成一个布尔电路 C，其中布尔电路可分解为 AND、OR 等基础的门电路，然后依次实现基础门电路就获得了计算函数的最终结果。基于 Yao 乱码电路的通用安全两方计算协议的通信轮数是常数，但是协议的通信量较大，与电路的 AND/OR 门数线性相关，而且扩展到多方场景较为复杂。后续有大量的工作对原协议的通信复杂度进行优化，如 point-and-permute [4]，free XOR [5]，GRR3/2 [6, 7]，half gates [8]，slicing and dicing [9] 等，其中 point-and-permute [4] 技术通过加入置换比特来确定解密位置，避免了原协议中加入较长的校验比特和置换操作；free XOR [5] 技术通过加入全局随机变量，实现了本地运算异或门的功能，即计算异或门的计算参数不需要额外的传输；GRR3 [6] 技术通过改变原协议的加密形式，可减少每个门 1/4 的通信量；GRR2 [7] 技术通过引入 Shamir 的秘密分享，可减少每个门 1/2 的通信量，但是该技术不能和 free XOR 兼容；half gates [8] 技术能将传统的 AND 门转化为特殊的两个半门，可减少每个门 1/2 的通信量，同时该技术能与 free XOR 技术兼容；slicing and dicing [9] 技术通过对每个门输入进行拆分处理，能减少每个门 5/8 的通信量，同时该技术能与 free XOR 技术兼容。

通用安全多方计算协议的另一条主流构造方法基于秘密分享 [3, 10, 11, 12]。Goldreich 等 [3] 基于秘密分享和不经意传输技术给出了通用的两方安全计算协议——GMW 协议。该协议通过对每个秘密输入进行异或分享，然后逐层的对电路求值，其中异或门的计算各参与方只需要本地计算，AND 门需要各参与方使用不经意传输实现。最后通过恢复输出导线的值获得最终的函数计算结果。该技术较容易扩展到多方场景，但是要求各参与方之间进行多轮交互，通信轮数与电路的层数相关，通信复杂度较高。Ben-Or 等 [10] 采用 Shamir 的秘密分享 [13] 给出了另一个通用的安全多方计算——BGW 协议。该协议通过对每个秘密输入进行秘密分享，巧妙的利用 Shamir 秘密分享方案的同态性质，各个参与方可计算各门的输出分享，最后通过秘密分享的恢复算法组合出最终的函数计算结果。与 GMW 协议 [3] 不同的是，该协议采用的 Shamir 秘密分享拥有天然的同态性质，在计算乘法门时不需要使用不经意传输技术，直接通过对分享进行乘积运算，对应的新的多项式的首项即为乘积结果。该协议适用于对域上的包含加法、乘法、标量乘法门的算术电路求值，计算效率较高且容易扩展到多方场景，但是通信轮数与电路层数相关，在计算乘法门时会导致导致多项式次数扩张，要求更多的参与方进行降次。随后，Beaver [11] 提出了“乘法三元组”的概念，可将安全多方计算协议划分为（参与方输入未知时）预处理阶段和（参与方选择好输入时）在线阶段，在预处理阶段各参与方生成一些关联性的随机量，在线阶段使用预存的随机量实现高效的在线函数计算。该方法极大的减少了 BGW 协议中的

通信复杂度，避免了计算乘法门时的多项式扩张问题。

## 2.2 门限密码的研究现状

门限密码可看作关于密码功能操作 (解密/签名) 的特殊的多方安全计算，多个用户秘密地分享密钥信息，在进行解密/签名操作时，用户可使用自己的私钥，通过多方安全计算的模式多方协作解密密文/签名消息，对于解决单点故障问题，以及分布式环境下多用户的数据安全、隐私保护和身份认证有着重要的应用意义。虽然理论上可对现有的密码系统采用通用的安全多方计算技术进行分布式扩展，构建安全的门限密码系统，但是 Goldreich [1] 指出直接应用现有的通用方法来解决具体的问题是低效和不切实际的。因此，针对具体的门限密码系统的设计需要兼顾实现功能、应用效率、安全性等方面，构造特定的安全多方计算协议。1979 年，Shamir 和 Blakley 分别独立地提出了秘密分享方案，门限的思想便深入人心。随后，Desmedt 和 Frankel 等 [14, 15] 在研究面向群组的密码 (group oriented cryptography) 时，引入了门限密码的概念，彻底地打开了门限密码研究的大门。目前，本文主要研究安全多方计算技术在标准密码算法门限化的应用：比如门限 ECDSA 和门限 SM2 等。

**门限 ECDSA 的研究现状.** 上世纪九十年代，随着数字签名标准 (digital signature algorithm，DSA) 的提出和广泛应用。特别是，近些年密码货币的爆发式发展，基于椭圆曲线的数字签名标准 (ECDSA) 在比特币等密码货币中承担重要应用。而在其中签名密钥的丢失也意味着具体经济的损失。由于比特币系统中使用了多重签名的解决方案，并非门限签名，限制了系统的灵活性，难以支持复杂的过程结构。因此，对 DSA 进行门限化具有重要的应用意义。

在门限 DSA 的研究探索中，由于 ECDSA 签名的非线性结构，对其门限化需要所有签名参与方通过交互运算共同协商随机数，并计算随机数的逆和随机数与签名私钥的乘积，导致对 ECDSA 算法门限化非常困难。1996 年，Gennaro 等人 [16] 给出了交互式门限签名的形式化定义，并基于联合随机秘密分享方案 (joint random secret sharing，JRSS) 和联合零秘密分享方案 (joint zero secret sharing，JZSS)，并设计了计算乘积和求逆的多方安全计算方法，对 DSA 进行门限化，构造了全分布式的门限数字签名标准。具体而言，根据 JRSS 技术协商出 DSA 的签名验证公钥和签名私钥，并分别保留私钥分享。签名阶段，各参与方使用 JRSS 协商随机数 k 并各自保留随机数分享，使用多方安全求逆运算计 $k^{-1}$，使用多方安全乘积运算计算签名私钥与随机数 $k$ 的组合，各参与方可获得签名值的分享。最后各参与方公布签名分享，达到门限值即可组合出最终签名。但是该方案在组合签名阶段，各用户需要计算私钥与随机数的乘积与随机数的

求逆，导致了多项式的次数扩张。具体而言，当门限值是 $t$ 时，至少需要 $2t+1$ 个用户才能组合出签名。因此，该方案并非门限最优化的，而且通信消耗较大，应用并不方便。

2016 年，Gennaro 等人 [17] 针对门限 ECDSA 无法达到门限最优的问题，基于 Paillier [18] 的同态加密方案的门限版本配合陷门承诺技术，构造了门限最优的门限 ECDSA。具体而言，该方案采用了门限同态加密技术，各参与方可以使用同态加密的性质对于明文进行密态操作，在保证各方隐私信息安全的情况下，完成复杂的签名运算。但是该方案依旧要求各用户之间进行大量的交互运算，导致通信消耗较高，而且在恶意模型下签名算法需要复杂的零知识证明技术。随后，在 CCS 2018 上，Gennaro 和 Goldfeder [19] 对上述方案进行优化，使用了特殊的多方安全计算技术，避免了使用复杂的零知识证明方法，设计了较为简单的门限 ECDSA 方案。但是该方法依旧需要 Paillier 的同态加密，而目前并没有针对 Paillier 加密方案的高效分布式密钥生成算法。针对该问题，Lindell 等人 [20] 使用指数 ElGamal 的加法同态算法代 Paillier 的加法同态加密算法对门限 ECDSA 分布式密钥生成算法和签名算法进一步优化。

**门限 SM2 的研究现状.** 2010 年，我国国家密码管理局发布了 SM2 椭圆曲线公钥密码算法，对我国商用密码产品和信息安全体系建设具有重大意义。然而当时对于 SM2 的门限化设计工作研究较少，限制了 SM2 算法在分布式系统中的应用推广。尚铭等人 [21] 基于秘密分享的思想对 SM2 签名算法进行门限化，并分别针对存在可信中心和不存在可信中心的情况下，设计门限了 SM2 签名算法，填补了 SM2 算法在多方合作应用环境下的空缺。但是，该方案无法避免多项式扩展问题，难以达到最优门限，在门限值为 $t$ ($t$ 个人可恢复密钥)，总用户数量要求 $n \geq 2t-1$，即至少需要 $2t-1$ 个参与者集合才可以生成一个有效的签名。2020 年, Zhang 等 [22] 提出了基于同态加密的 SM2 两方协同签名算法，并给出了可证明安全分析，但是该方法对签名私钥采用乘积形式分享，多方扩展较为复杂。随后，涂彬彬等 [23] 归纳了三种安全多方计算方法, 分别是基于门限加法同态加密算法、基于多个加法同态加密算法和基于单一加法同态加密算法的安全多方计算协议，并基于安全多方计算技术设计了门限 SM2 签名算法。该门限 SM2 可以达到门限最优，即对于门限值 $t$，总用户数量要求 $n \geq t$，只需要 $t$ 个参与者集合就可以生成一个有效的签名。

# 3 前期的理论研究与试验论证工作的结果

## 3.1 MPC 前期理论研究

目前，本文主要对 Yao 协议和 BGW 协议等通用 MPC 协议的效率优化进行了前期理论研究。

### 3.1.1 Yao 协议

Yao 乱码电路 [2] 包含两个参与方：参与方 A 是乱码电路生成方，参与方 B 是计算方。两方计算任意函数 $f$，需要将函数转化成布尔电路。令参与方 A 的输入是 $x \in \{0,1\}^n$，参与方 B 的输入是 $y \in \{0,1\}^n$。令 $C$ 是由函数 $f$ 转化的布尔电路，以 $x, y \in \{0,1\}^n$，输出 $C(x,y) \in \{0,1\}^n$ (假设输入长度，输出长度和安全参数的长度都是 $n$)。令 $g$： $\{0,1\} \times \{0,1\} \to \{0,1\}$ 是布尔电路 $C$ 的一个门，令 $g$ 的两个输入导线为 $w_1, w_2$，输出导线为 $w_3$。

乱码电路生成方 A 计算乱码电路 $g$ 的过程：A 独立运行密钥生成算法 $\mathsf{Gen}(1^\lambda)$ 获得密钥 $k_1^0, k_1^1, k_2^0, k_2^1, k_3^0, k_3^1$，根据 $g$ 计算如下密文并执行置换操作。

$$c_{0,0} = \mathsf{Enc}_{k_1^0}(\mathsf{Enc}_{k_2^0}(k_3^{g(0,0)}))$$

$$c_{0,1} = \mathsf{Enc}_{k_1^0}(\mathsf{Enc}_{k_2^1}(k_3^{g(0,1)}))$$

$$c_{1,0} = \mathsf{Enc}_{k_1^1}(\mathsf{Enc}_{k_2^0}(k_3^{g(1,0)}))$$

$$c_{1,1} = \mathsf{Enc}_{k_1^1}(\mathsf{Enc}_{k_2^1}(k_3^{g(1,1)}))$$

乱码电路计算方 B 获得输入导线对应的两个密钥 $(k_1^a, k_2^b), a, b \in \{0,1\}$ 和上述置换后的密文表，计算乱码电路 $g$ 的过程：B 只能正确解密上述四个密文中的 $c_{a,b}$ 获得 $k_3^{g(a,b)} = \mathsf{Dec}_{k_2^b}(\mathsf{Dec}_{k_1^a}(c_{a,b}))$，其中 $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$，表示 IND-CPA 安全的对称加密。

基于 Yao 乱码电路的安全两方计算协议：

1) A 对布尔电路 $C$ 的所有门进行计算，获得乱码电路 $G(C)$，并将 $G(C)$ 发送给 B；

2) 令 $w_1, \cdots, w_n$ 表示 $x$ 对应的输入导线，令 $w_{n+2}, \cdots, w_{2n}$ 表示 $y$ 对应的输入导线；

   (a) A 将 $w_1, \cdots, w_n$ 导线对应的密钥 $k_1^{x_1}, \cdots, k_n^{x_n}$ 发送给 B；

   (b) 对于 $w_{n+2}, \cdots, w_{2n}$ 导线对应的密钥，A 和 B 执行 2 选 1 的不经意传输协议进行选择，其中 $i \in [n]$，A 输入为 $(k_{n+i}^0, k_{n+i}^1)$，B 的输入为 $y_i$；

3) B 获得乱码电路 $G(C)$ 和 $2n$ 个输入密钥，B 可以计算电路获得函数值 $f(x,y)$，并将 $f(x,y)$ 发送给 A。

**乱码电路行减少优化方法.** Pinkas 等 [7] 提出乱码电路行减少 (Garbled Row Reduction, GRR2) 优化技术，使用秘密分享技术对 Yao 协议进行优化，将每个门的四个密文减少至两个密文。以 AND 门为例，具体的优化思路如下：令 $g$ 的两个输入导线表示为 $(a,b)$，输出导线表示为 $c$；A 生成密钥 $k_a^0, k_a^1, k_b^0, k_b^1, k_c^0, k_c^1$，计算 $k^1 = H(k_a^0||k_b^0), k^2 = H(k_a^0||k_b^1), k^3 = H(k_a^1||k_b^0), k^4 = H(k_a^1||k_b^1)$，以 $p(1) = k^1, p(2) = k^2, p(3) = k^3$ 为多项式值计算二次多项式 $p(\cdot)$，并计算二次多项式 $q(\cdot)$ 使得 $q(4) = k^4, q(5) = p(5), q(6) = p(6)$，然后，令 $k_c^0 = p(0), k_c^1 = q(0)$。因此，每个门只需要两个多项式值 $(p(5), p(6))$。计算方 B 在获得两个输入导线的密钥后能计算额外的一个多项式值，结合每个门公布的两个值可计算出输出导线的密钥，即对应多项式的首项。

因为 GRR2 优化技术中每个门的输出密钥 $k_c^0, k_c^1$ 被多项式限定，无法满足 free XOR 优化技术中的密钥结构，所以该技术无法和 free XOR 技术 [5] 兼容，导致 XOR 门依旧需要传输密文。因此，探索新型的秘密分享技术，优化现有 GRR2 技术，进一步减小每个门的通信量且兼容 free XOR 技术对于提升 MPC 协议效率尤其关键。

#### 3.1.2 BGW 协议

BGW 协议可以用于对域上包含加法、乘法、标量乘法门的算数电路求值。该协议巧妙地利用了 Shamir 秘密分享对各参与方的输入进行秘密分享，并直接在秘密分享上实现函数计算。假定参与方 $P_i$ 的秘密输入 $x$，令 $[x]$ 表示各个参与方持有的 Shamir 秘密份额，即 $P_i$ 选择首项为 $x$ 的 $t-1$ 次随机多项式 $p(\cdot)$，其他参与方的秘密份额为 $p(j), j \in [n]$。其中，$t$ 为秘密分享的门限值，即任意 $t-1$ 个秘密份额都不会泄漏 $x$ 的任何信息。

BGW 协议计算过程为：对于算数电路的每一条导线 $w$，各个参与方都持有导线值 $x_w$ 所对应的秘密 $[x_w]$。

1) 加法门计算：对于输入导线为 $a,b$ 对应输入值为 $x,y$，输出导线为 $c$ 的加法门，各参与方的目的是计算 $z = x + y$ 的秘密份额。各参与方分别持有输入导线的秘密份额 $[x_a]$ 和 $[y_b]$，只需要本地计算输出导线的秘密份额 $[z_c] = [x_a] + [y_b]$；

2) 乘法门计算：对于输入导线为 $a,b$ 对应输入值为 $x,y$，输出导线为 $c$ 的乘法门，各参与方的目的是计算 $z = x \cdot y$ 的秘密份额。各参与方分别持有输入导线的秘密份额 $[x_a]$ 和 $[y_b]$，本地计算秘密份额的乘积；然后，$2t-1$ 个参

与方协同计算多项式降次过程，将 $2t-2$ 次的多项式降到 $t-1$，且该多项式的首项是 $z$，每个参与方获得 $[z_c]$ 的秘密份额；

BGW 协议所有门计算完成后，各参与方会拥有输出导线的秘密份额，然后基于 Shamir 秘密分享的恢复算法计算函数值。但是，该协议每个乘法门需要各参与方之间进行交互，协议的通信轮数和电路层数相关。此外，每次乘法门计算时，会导致多项式次数扩张，需要额外的多项式降次，增加协议的通信复杂度。针对多项式扩张的问题，Beaver [11] 提出了“Beaver 乘法三元组”的优化方法。

**Beaver 三元组优化.** Beaver 三元组 (或称乘法三元组) 指的是秘密份额三元组 $[a], [b], [c]$，其中 $a$ 和 $b$ 是从某个适当的域中选择出的随机数，且 $c=ab$。在线阶段中，每对一个乘法门计算都需要“消耗”一个 Beaver 三元组。考虑一个输入导线为 $x, y$ 的乘法门。各个参与方持有秘密份额 $[x]$ 和 $[y]$。为应用 Beaver 三元组 $[a], [b], [c]$ 计算 $[xy]$，参与方执行下述步骤:

1) 各个参与方在本地计算 $[x-a]$ 并公布，即所有参与方向其他参与方告知自己持有的秘密份额 $[x-a]$。因此，所有参与方可获得 $d=x-a$。虽然 $d$ 的取值依赖于秘密值 $x$，但由于秘密值 $x$ 被随机值 $a$ 所掩盖，$d$ 不会泄漏 $x$ 的任何信息。

2) 各个参与方在本地计算 $[y-b]$ 并公布，即所有参与方向其他参与方告知自己持有的秘密份额 $[y-b]$。因此，所有参与方可获得 $e=y-b$。同理，虽然 $e$ 的取值依赖于秘密值 $y$，但由于秘密值 $y$ 被随机值 $b$ 所掩盖，$e$ 不会泄漏 $y$ 的任何信息。

3) 根据 $xy=(x-a+a)(y-b+b)=(d+a)(e+b)=de+db+ae+c$. 各个参与方可通过 $d$ 和 $e$，以及秘密份额 $[a], [b], [c]$, 在本地计算 $xy$ 的秘密份额 $[xy]=de+d[b]+e[a]+[c]$.

应用 Beaver 三元组，只需要公开两个参数即可通过本地计算完成乘法门的求值，避免了计算乘法门时的多项式扩张，极大的减小了各参与方之间的通信量，在线阶段非常高效。目前有多种方法用于离线阶段生成 Beaver 三元组，例如同态加密、不经意传输，或者以随机数作为输入直接执行 BGW 乘法协议。然而，各种方法各有缺点，并不完美：基于同态加密技术计算复杂度较高，基于不经意传输技术通信复杂度较高，通过 BGW 协议乘法协议会导致多项式扩张的问题，难以实现最优门限。因此，探索高效生产 Beaver 三元组的方法对于提升 MPC 协议效率尤其重要。

## 3.2 门限密码前期理论研究

目前，我们主要研究了 MPC 技术在标准密码算法门限化上的应用，特别是 ECDSA 和 SM2 签名算法，以及其相关扩展的门限化设计。

### 3.2.1 ECDSA 签名及其门限化

**ECDSA 签名.** 令 $\mathbb{G}$ 为椭圆曲线上的群, $G$ 为群 $\mathbb{G}$ 中阶为 $q$ 的基点. 随机选择签名私钥 $x \leftarrow \mathbb{Z}_q$, 计算验证公钥 $X = xG$. 关于消息 $m \in \{0,1\}^*$ 的 ECDSA 签名步骤如下:

1) 计算 $e = H(m)$.

2) 选择随机数 $k \leftarrow \mathbb{Z}_q$, 计算 $(r_x, r_y) = kG$.

3) 计算 $r = r_x \bmod q$, 如果 $r = 0$ 返回步骤 2.

4) 计算 $s' = k^{-1}(e + rx) \bmod q$, $s = \min\{q - s', s'\}$.

5) 输出签名值 $\sigma = (r, s)$.

ECDSA 签名 $\sigma = (r, s)$ 的验证步骤如下:

1) 计算 $e = H(m)$, $(r'_x, r'_y) = s^{-1} \cdot (m' \cdot G + r \cdot Q)$.

2) 如果 $r = r'_x \bmod q$, 输出 1, 否则输出 0.

**ECDSA 签名门限化.** ECDSA 签名的特殊结构导致构造门限 ECDSA 具有如下困难：(1) 各个参与方需要分享签名私钥和签名所用的随机数；(2) 在签名过程中，需要计算随机数逆的秘密份额以及随机数逆和签名私钥的乘积。在安全多方计算中计算乘积相对较为复杂，所以门限 ECDSA 构造也较为复杂。目前构造门限 ECDSA 签名主要基于秘密分享、同态加密、不经意传输等技术。然而各种构造方法并不完美，其中，基于秘密分享构造的门限 ECDSA 签名难以避免多项式扩展的问题，无法达到门限最优。基于同态加密构造的门限 ECDSA 签名，在签名过程中需要复杂的加解密运算，恶意模型下还需要昂贵的零知识证明技术，计算复杂度较高。基于不经意传输的门限 ECDSA 签名需要大量的预计算过程，而且通信复杂度较高。

ECDSA 签名在各中应用系统中有着重要应用，对其的门限化设计可满足分布式应用需求，特别是在区块链、密码货币中防止单点故障，保证私钥安全有着关键作用。然而，现有的门限 ECDSA 签名并不完美，在计算复杂度和通信复杂度很难找到均衡。因此，结合 ECDSA 签名结构和安全多方计算技术，探索基于 ECDSA 的新型门限签名尤为重要。

**ECDSA 适配器签名门限化.** 适配器签名是传统签名的扩展，能同时实现签名授权和证据提取两种功能，具有减少链上操作, 提高交易的可替代性和突破区块链脚本语言限制等多种优势, 在许多区块链应用中都显示出了重要的应用价值, 如支付通道网络 [24] 和原子交换协议 [25] 等。Aumayr 等人 [24] 给出适配器签名的形式化定义，并基于 ECDSA 签名给出首个可证明安全的适配器签名。但是该适配器签名在预签名过程中需要零知识证明保证预签名操作的正确执行。该证明以签名随机数作为证据难以离线操作，导致预签名操作较为复杂，也限制了 ECDSA 适配器门限化设计。因为在门限化过程中，各个参与方持有的是随机数的秘密分享，预签名中的零知识证明需要各方协调生产，难以实现。因此，目前门限 ECDSA 适配器签名并没有具体的构造方案。因此，基于安全多方计算技术，结合 ECDSA 适配器签名结构，探索安全高效且易于门限化的 ECDSA 适配器签名尤为重要。

#### 3.2.2 SM2 签名及其门限化

**SM2 签名.** 令 $\mathbb{G}$ 为椭圆曲线上的群, $G$ 为群 $\mathbb{G}$ 中阶为 $n$ 的基点. 随机选择签名私钥 $x \in [1, n-2]$, 计算验证公钥 $X = xG$. 关于消息 $m \in \{0,1\}^*$ 的 SM2 签名步骤如下:

1) 计算 $e = H(m)$.

2) 选择随机数 $k \in [1, n-1]$, 计算 $(r_x, r_y) = kG$.

3) 计算 $r = r_x + e \bmod n$, 如果 $r = 0$ 或者 $r + k = n$ 则返回步骤 2.

4) 计算 $s = (1+x)^{-1}(k - rx) \bmod n$, 如果 $s = 0$ 则返回步骤 2.

5) 输出签名值 $\sigma = (r, s)$.

SM2 签名 $\sigma = (r, s)$ 的验证步骤如下:

1) 如果 $r \notin [1, n-1]$ 或者 $s \notin [1, n-1]$, 输出 0 并退出.

2) 计算 $e = H(m)$.

3) 计算 $t = (r + s) \bmod n$, 如果 $t = 0$, 输出 0 并退出.

4) 计算 $(r'_x, r'_y) = sG + tX$.

5) 计算 $r' = r'_x + e \bmod n$, 如果 $r' = r$ 则输出 1 否则输出 0.

SM2 签名算法是我国自主研发的签名算法, 具有安全性高, 运算速度快, 签名尺寸小等优势. SM2 签名算法的安全性分析已有较多工作, 具体可参考 [26, 27].

**SM2 签名门限化.** 目前研究 SM2 签名的门限化工作较少，严重影响 SM2 签名算法在分布式应用场景下的推广应用。尚铭等人 [21] 基于 Shamir 的秘密分享技术，给出了门限 SM2 算法，但是该方案难以避免多项式扩展问题，无法达到门限最优。在门限值为 $t$ 时，总用户数量要求 $n \geq 2t-1$，即至少需要 $2t-1$ 个参与者集合才可以生成一个有效的签名。随后，Zhang 等 [22] 基于同态加密给出 SM2 两方协同签名算法，但是该方案采用乘积的形式分享签名私钥，多方扩展较为困难。涂彬彬等 [23] 归纳了三种安全多方计算方法，设计了门限化的 SM2 签名算法。该方案可达到最优门限，各参与方的交互轮数较少，但是只实现了在半诚实模型下可证明安全。因此，构造交互轮数低，计算效率高的门限 SM2 签名算法，同时探索 SM2 签名的功能扩展，比如构造基于 SM2 适配器签名，SM2 签密等并结合安全多方计算技术进行门限化，对于 SM2 标准签名算法的推广和应用具有重大意义。

# 4 主要研究内容、实施方案及其可行性论证

## 4.1 主要研究内容

(1) 探索高效且安全性更强的 MPC 协议

1) 安全模型研究：调研现有 MPC 的安全性模型，比如半诚实模型、恶意模型、静态模型、自适应模型等，研究各种安全模型下 MPC 的存在性问题，以及具体的 MPC 协议构造技术。探索在各种安全模型下是否存在新型的高效 MPC 协议构造，丰富现有的通用 MPC 协议构造模式。同时，结合实际应用和具体攻击模式，探索更贴近现实的安全性模型，并在该模型下试图给出具体的 MPC 协议的构造。

2) 效率优化研究：梳理现有的通用 MPC 协议设计路线，比如：基于 Yao 乱码电路的 MPC (Yao-协议)、基于不经意传输的 MPC (GMW-协议)，以及基于秘密分享的 MPC (BGW-协议) 等。深入学习不同构造路线下的效率优化技术，比如 Yao-协议的优化技术：point-and-permute [4]，free XOR [5]，GRR3/2 [6, 7]，fleXOR [28]，half gates [8]，slicing and dicing [9] 等；GMW-协议的优化技术：OT extension [29]，low-depth circuits [30] 等；BGW-协议的优化技术：online/offline，Beaver 乘法组 [11] 等。进一步研究现有优化技术的困难点，比如：GRR2 优化技术 [7] 受限于使用的秘密分享份额尺寸，同时难以兼容 free XOR [5]；Beaver 乘法组需要消耗大量的随机值，对于随机值的生成效率要求较高；试图突破现有优化技术的桎梏，探索更适用的通用 MPC 效率优化技术，实现更加高

效的通用 MPC 协议。

(2) 探索丰富的安全多方计算应用

1) 门限密码应用：在物联网、云计算、区块链等分布式应用场景下，传统公钥密码应用具有局限性，比如：功能性上难满足具体的应用需求，实际应用效率应用系统难以承受，安全性上难以达到安全需求等。针对新型分布式应用场景下对于分布式密码系统的迫切需求，探索当前应用系统对于密码功能的需求，通过深挖安全多方计算技术的应用模式，结合分布式应用需求，根据具体的密码算法，探索安全高效的门限密码构造。

2) 标准算法扩展及其门限化：标准密码算法 ECDSA，SM2，SM9 等在各中应用场景下应用广泛，但是对于新型的分布式应用场景存在迫切的门限化构造需求。特别是，目前基于 ECDSA，SM2，SM9 的门限签名需要较为复杂的 MPC 技术，实现恶意模型下的安全性更需要昂贵的零知识证明技术。对此，我们整理当前安全多方计算在标准算法门限化的具体应用，提炼核心构造技术，结合具体的算法结构和实际应用需求，安全高效实现标准算法的门限化。此外，贴合具体应用需求，对于传统密码算法，特别是标准密码算法进行功能扩展，可实现安全高效的密码应用，比如：适配器签名 (adaptor signature, AS) [25, 24, 31]，也称为无脚本脚本 (scriptless script) 可看作是数字签名对困难关系的扩展，联合了签名授权和证据提取两种功能，是目前解决区块链应用 (如密码货币) 中扩展性差、吞吐量低等问题的重要密码技术。签密方案能同时实现加密和签名，在具体应用中比单独实现加密和签名更加高效。因此，我们跟踪研究新型应用和具体功能，针对具体需求对相关算法进行功能扩展，并基于安全多方计算技术进行分布式设计，解决分布式应用场景下的密码算法应用需求。比如：我们针对现有适配器签名存在的问题，一方面探索门限适配器签名的形式化定义，为后续方案设计提供安全性分析基础，另一方面，我们对现有 ECDSA 的适配器签名进行效率优化，并采用安全多方计算技术进行分布式设计，同时探索基于 SM2 构造适配器签名算法的构造方法和门限化的设计技术。

### 4.2 实施方案及其可行性论证

目前，在 MPC 协议效率优化方面，针对 Yao 协议的优化技术限制，我们试图突破 GRR2 技术难以兼容 free XOR 的问题，并探索 BGW 协议中高效批量生成 Beaver 三元组的方法，初步给出优化 GRR2 技术和高效生成 Beaver 三元

组的设计思想。在安全多方计算应用方面，我们结合具体的应用需求，扩展的现有密码算法的应用功能，对现有标准密码算法进行分布式门限化的设计，已构造了基于 ECDSA 的两方签名算法、ECDSA 适配器签名及其两方协议扩展、高效的 SM2 适配器签名算法，以及 SM2 签密算法及其两方协议扩展。后续工作：一方面根据 MPC 协议优化思路设计优化方案并给出具体的实现和可证明安全，另一方面主要研究如何将已构造的两方协议扩展到更加通用的门限版本，同时跟进新型密码应用，探索更加丰富的安全多方计算的应用。

#### 4.2.1 Yao 协议的优化思路

Pinkas 等 [7] 提出的乱码电路行减少 (Garbled Row Reduction, GRR2) 技术，将原 Yao 协议的每个门的通信量减少 1/2，但是该技术与 free XOR 技术并不兼容。我们通过深入学习 GRR2 优化技术，发现其本质上是基于 Shamir 的秘密分享：将每个门的输出导线看作是两个秘密，根据不同的导线输入，计算方可计算出不同的秘密代表不同的导线输出。

以 AND 门为例：乱码电路生成方 A 根据输入导线密钥的四种组合形式 $(k_a^0, k_b^0)$, $(k_a^0, k_b^1)$, $(k_a^1, k_b^0)$, $(k_a^1, k_b^1)$，前三种组合对应输出导线密钥 $k_c^0$，后一种对应输出导线密钥 $k_c^1$。A 将前三种组合看作秘密分享的秘密份额，并生成多项式 $p(\cdot)$，并定义 $k_c^0$ 为多项式 $p(\cdot)$ 的首项；然后根据多项式 $p(\cdot)$ 计算另一个多项式的两个值，并结合 $(k_a^1, k_b^1)$ 对应的点，生成新的多项式 $q(\cdot)$，并定义 $k_c^1$ 为多项式 $q(\cdot)$ 的首项；最后公布两个多项式重合的两个值作为计算参数；乱码电路计算方 B 在获得一组密钥之后可以计算出一个多项式的值，结合两个计算参数，可以恢复出对应的输出导线的密钥，完成 AND 的计算。

GRR2 技术每个门只需要公布两个多项式的值作为计算参数，将 Yao 协议每个门需要公布 4 个密文降低到 2 个，但是前三种密钥组合限定了多项式 $p$，两个计算参数和第四种密钥组合限定了多项式 $q$，导致输出导线对应的密钥难以满足 free XOR 技术的密钥结构。

根据上述分析，我们后续探索具有不同尺寸秘密份额的秘密分享技术。该技术可优化公布的秘密份额，减少现有 Yao 协议每个门需要公布的密文尺寸。初步分析，我们可以考虑引入加密机制，比如对于 Shamir 的秘密分享，秘密尺寸和秘密份额尺寸相同。如果秘密的尺寸为 $|S|$，每个参与方的秘密份额尺寸也是 $|S|$。引入加密机制后，可以对秘密进行加密，然后对密钥进行秘密分享，如此每个参与方的秘密份额尺寸就变成了密钥尺寸 (一般 128 比特)。如果 $|S|$ 远远大于 128 比特，可极大的减少秘密份额尺寸。此外，在减少秘密份额尺寸的同时，可考虑增加多项式次数，放开 GRR2 优化技术对于输出导线的密钥的限制，可以兼容 free XOR 优化技术。

#### 4.2.2 Beaver 乘法组的批量生成

Beaver 乘法组可避免传统 BGW 协议乘法门多项式降次的过程是高效优化 BGW 协议的关键性技术。然而，为了保证协议安全，每个乘法组只能使用一次，在线阶段会消耗大量的乘法组。因此，高效的生成 Beaver 乘法组尤为重要。目前主要基于同态加密技术生成乘法组的计算复杂度较高，基于不经意传输技术生成乘法组的通信复杂度较高。

针对上述问题，我们初步考虑 Beaver 乘法组扩展技术，基于少量乘法组高效批量扩展出大量的乘法组，如此离线阶段只需要生成少量的乘法组即可满足在线阶段乘法组的大量需求。以两方场景为例，参与方 A 和参与方 B 分别拥有乘法组 $(a_1, b_1, c_1)$ 和 $(a_2, b_2, c_2)$，满足 $c = c_1 + c_2 = ab = (a_1 + a_2)(b_1 + b_2)$，试图生成多个乘法组。具体思想如下：

1) 参与方 A 选择随机一次多项式 $f_1, g_1$，选择首项为 0 的随机二次多项式 $h_1$，计算 $f_1(1)$, $f_1(2)$, $f_1(3)$, $g_1(1)$, $g_1(2)$, $g_1(3)$, $h_1(1)$, $h_1(2)$, $h_1(3)$，将 $f_1(2)$, $g_1(2)$, $h_1(3)$, $d_1 = f_1(3) - a_1$, $e_1 = g_1(3) - b_1$ 发送给 B；

2) 参与方 B 选择随机一次多项式 $f_2, g_2$，选择首项为 0 的随机二次多项式 $h_2$，计算 $f_2(1)$, $f_2(2)$, $f_2(3)$, $g_2(1)$, $g_2(2)$, $g_2(3)$, $h_2(1)$, $h_2(2)$, $h_2(3)$，将 $f_2(1)$, $g_2(1)$, $h_2(3)$, $d_1 = f_2(3) - a_2$, $e_2 = g_2(3) - b_2$ 发送给 A；

随后，A 本地计算 $f(1)g(1) = (f_1(1) + f_2(1))(g_1(1) + g_2(1))$, $d = d_1 + d_2 = f(3) - a$, $e = e_1 + e_2 = g(3) - b$, $u_A = u_1(f(1)g(1) + h_1(1)) + u_2h_1(2) + u_3(de + db_1 + ea_1 + c_1 + h_2(3))$；B 本地计算 $f(2)g(2) = (f_1(2) + f_2(2))(g_1(2) + g_2(2))$, $d = d_1 + d_2 = f(3) - a$, $e = e_1 + e_2 = g(3) - b$, $u_B = u_1h_2(1) + u_2(f(2)g(2) + h_2(2)) + u_3(db_2 + ea_2 + c_2 + h_1(3))$；

根据上述构造可生成新的乘法组 $u_A + u_B = f(0)g(0) = (f_1(0) + f_2(0))(g_1(0) + g_2(0))$，其中参与方 A 拥有 $(f_1(0), g_1(0), u_A)$，参与方 B 拥有 $(f_2(0), g_2(0), u_B)$；基于该构造方法参与方 A 和 B 可以使用相同的预存乘法组，每次独立随机的选择多项式，并行计算出多个乘法组。

安全性分析的关键在于每个乘法组都是使用相同的预存乘法组 $(a_1, b_1, c_1)$ 和 $(a_2, b_2, c_2)$ 是否会泄漏后续生成的乘法组？一方面，这里的预存乘法组每次都使用不同的多项式值 $f_i(3), g_i(3)$ 隐藏 (类似一次一密，其中预存乘法组相当于明文，$f_i(3), g_i(3)$ 相当于使用一次的密钥)，且 A 和 B 选用新的二次多项式进一步隐藏 A 和 B 计算的秘密份额。

#### 4.2.3 基于 ECDSA 的门限签名算法

目前基于 ECDSA 构造的两方签名算法需要复杂的安全多方计算技术，比如同态加密、不经意传输、秘密分享等 [32, 33, 34]，同时需要昂贵的零知识证明技术抵抗恶意的敌手。我们发现 ECDSA 的签名结构具有可组合性质，即签名相同消息的两个 ECDSA 签名值可以组合成新的签名。基于这个观察，我们首次提出组合 ECDSA 算法，并基于 ECDSA 安全性给出了组合 ECDSA 签名的安全性证明。同时，我们对组合 ECDSA 签名算法进行两方分布式设计，构造安全高效的两方签名协议，并基于 ECDSA 给出可证明安全。

**组合 ECDSA 算法.** 组合 ECDSA 算法包含三个算法：密钥生成算法 Gen，签名算法 Sign 和签名验证算法 Vrfy，具体如下：

- $\mathsf{Gen}(pp) \to (x, G)$：密钥生成算法输入公开参数 $pp$，均匀随机选择 $x_1, x_2 \leftarrow \mathbb{Z}_q$，计算 $Q_1 = x_1 G, Q_2 = x_2 G$, $Q_{\text{add}} = Q_1 + Q_2$, $Q_{\text{mul}} = x_1 x_2 G$。其中，签名私钥是 $x = (x_1, x_2)$, 签名验证公钥是 $Q = (Q_{\text{add}}, Q_{\text{mul}})$;
- $\mathsf{Sign}(x, m) \to \sigma$：签名算法输入签名私钥 $x$ 和消息 $m$，计算消息的哈希值 $H(m)$，选择随机数 $k \leftarrow \mathbb{Z}_q$，计算 $R = kG = (r_x, r_y), r = r_x \bmod q$，如果 $r = 0$ 则重新选择随机数 $k$，最后计算 $s' = k^{-1}(H(m) + rx) \bmod q$, $s = \min\{s', q - s'\}$, 输出签名值 $\sigma = (r, s)$;
- $\mathsf{Vrfy}(Q, m, \sigma) \to 1/0$：签名验证算法输入签名验证公钥 $Q$，消息 $m$，签名值 $\sigma$，计算消息的哈希值 $H(m)$，计算 $(r_x, r_y) = s^{-1}(H(m)^2 G + rH(m)Q_{\text{add}} + r^2 Q_{\text{mul}})$，如果 $r = r_x \bmod q$，则输出 1，否则输出 0。

**定理 1** *如果 ECDSA 是 EUF-CMA 安全的，则组合 ECDSA 也是 EUF-CMA 安全的.*

**基于组合 ECDSA 的两方签名.** 基于组合 ECDSA 的两方签名协议包含分布式密钥生成协议 DGen，分布式签名协议 DSign 和签名验证算法 Vrfy。其中，签名验证算法和组合 ECDSA 的验证算法相同。令两方参与方分别是 A 和 B。分布式密钥生成协议 DGen 和分布式签名协议 DSign 具体如下：

- **分布式密钥生成协议输入公共参数** $\mathsf{DGen}(pp)$**.**
  1) 参与方 A 首先运行 $\text{ECDSA.}\mathsf{Gen}(pp) \to (x_1, Q_1 = x_1 G)$，并发送 $(\mathsf{prove}, 1, Q_1, x_1)$ 给 $\mathcal{F}_{\text{zk}}^{R_{\text{DL}}}$;
  2) 参与方 B 验证 $\mathcal{F}_{\text{zk}}^{R_{\text{DL}}}$ 的消息 $(\mathsf{prove}, 1, Q_1)$ 并接受，否则拒绝，然后运行 $\text{ECDSA.}\mathsf{Gen}(pp) \to (x_2, Q_2 = x_2 G)$，并发送 $(\mathsf{prove}, 2, Q_2, x_2)$ 给 $\mathcal{F}_{\text{zk}}^{R_{\text{DL}}}$;

3) A 验证 $\mathcal{F}_{\text{zk}}^{R_{\text{DL}}}$ 的消息 $(\mathsf{proof}, 2, Q_2)$ 并接受，否则拒绝；

输出 a) A 计算 $Q_{\text{add}} = Q_1 + Q_2, Q_{\text{mul}} = x_1 Q_2, Q = (Q_{\text{add}}, Q_{\text{mul}})$，存储 $(x_1, Q)$；

b) B 计算 $Q_{\text{add}} = Q_1 + Q_2, Q_{\text{mul}} = x_2 Q_1, Q = (Q_{\text{add}}, Q_{\text{mul}})$，存储 $(x_2, Q)$；

- 分布式签名协议 $\mathsf{DSign}(sid, m)$：

1) A 选择随机数 $k_1 \leftarrow \mathbb{Z}_q$，并计算 $R_1 = k_1 G$，然后 A 发送 $(\mathsf{com\text{-}prove}, sid||1, R_1, k_1)$ 给 $\mathcal{F}_{\text{com-zk}}^{R_{\text{DL}}}$；

2) B 接收 $\mathcal{F}_{\text{com-zk}}^{R_{\text{DL}}}$ 的消息 $(\mathsf{proof\text{-}receipt}, sid||1)$，选择随机数 $k_2 \leftarrow \mathbb{Z}_q$，并计算 $R_2 = k_2 G$，发送 $(\mathsf{com\text{-}prove}, sid||2, R_2, k_2)$ 给 $\mathcal{F}_{\text{zk}}^{R_{\text{DL}}}$；

3) A 验证 $\mathcal{F}_{\text{zk}}^{R_{\text{DL}}}$ 的消息 $(\mathsf{proof}, sid||2, R_2)$ 并接受，否则拒绝，然后发送 $(\mathsf{decom\text{-}proof}, sid||1)$ 给 $\mathcal{F}_{\text{com-zk}}$；

4) B 验证 $\mathcal{F}_{\text{zk}}^{R_{\text{DL}}}$ 的消息 $(\mathsf{decom\text{-}proof}, sid||1, R_1)$ 并接受，否则拒绝，然后计算 $R = k_2 R_1 = (r_x, r_y)$；在接收到待签名的消息 $m$ 后，B 计算 $r = r_x \bmod q$，$s_2 = k_2^{-1}\,(r\ x_2 + H(m)) \bmod q$，并将 $s_2$ 发送给 A；

5) A 计算 $R = k_1 R_2 = (r_x, r_y)$，计算 $r = r_x \bmod q$，$s' = k_1^{-1}\,(r\ x_1 + H(m))\ s_2 \bmod q$，$s = \min\{s', q - s'\}$，A 验证签名 $(r, s)$，如果正确，则输出该签名，否则拒绝。

**定理 2** *如果 ECDSA 是 EUF-CMA 安全的，以及两方签名协议中使用的零知识证明和承诺方案也是安全的，则上述构造的两方签名协议也是安全的。*

**实验分析.** 我们基于 OpenSSL 库，使用 C++ 语言，实现了 ECDSA，组合 ECDSA，以及基于 ECDSA 的两方签名协议。实验结果表明我们的两方签名协议的签名算法和 ECDSA 的算法耗时相近，各项操作的耗时都是微秒级的，非常高效。实验结果如图 1 所示。

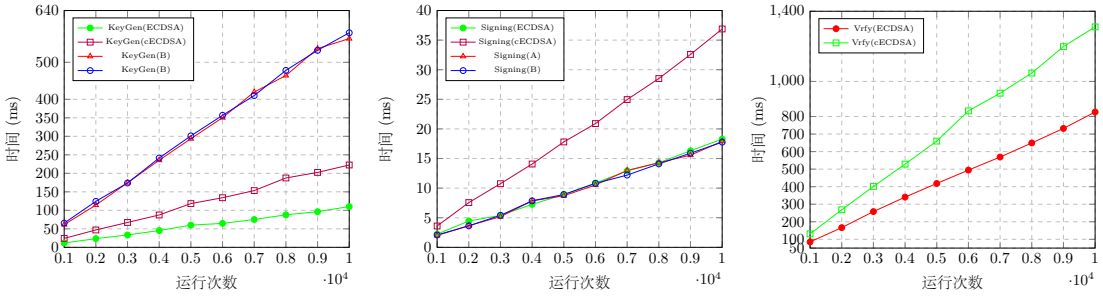


图 1: 效率对比 (左) 密钥生成，(中) 签名算法，(右) 验证算法

**门限 ECDSA 签名.** 上述构造的两方签名是一种特殊的 $(2, 2)$-门限签名，想要直接扩展到通用的门限场景较为复杂，直观原因在于组合 ECDSA 签名的结构

可看作两个 ECDSA 的乘积，两方组合容易，拆分也容易，在两方场景下较为高效，但是如果扩展到三个甚至更多参与方时，多个 ECDSA 签名的乘积聚合相对复杂，特别是各个参与方的密钥量和参与方数量指数级增加，签名验证也较为复杂。因此，后续工作可继续探索新型安全多方计算技术，将上述两方签名扩展到通用门限版本。

#### 4.2.4 门限 ECDSA 适配器签名

现有的 ECDSA 适配器签名计算预签名值时的零知识证明和签名使用的随机数相关，难以离线执行，导致预签名过程效率较低。现有的两方适配器签名的形式化定义满足公钥可聚合的性质，该性质与较多的高效两方 ECDSA 签名算法不符合。因此，我们设计支持离线证明的 ECDSA 适配器签名，且给出满足交互式密钥生成的两方适配器签名的形式化定义，然后给出两方 ECDSA 适配器签名，并给出可证明安全。

**ECDSA 适配器签名.** 关于困难关系 $R$ 的 ECDSA 适配器签名包含预签名算法 $\mathsf{pSign}$，预签名验证算法 $\mathsf{pVrfy}$，适配算法 $\mathsf{Adapt}$ 和提取算法 $\mathsf{Ext}$。令困难关系 $R = I_Y = (Y, \pi), y | Y = yG \wedge \mathsf{V}(I_Y) = 1$ 和 $R_Z = \{(I_Z = (G, Q, Y, Z), x) | Q = xG \wedge Z = xY\}$，令 $\mathsf{P}$ 和 $\mathsf{V}$ 分别表示证明和验证算法。

- $\mathsf{pSign}(Q, x, m, I_Y) \to \hat{\sigma}$: 预签名算法输入公钥 $Q$ 和签名私钥 $x$，消息 $m$，以及困难实例 $I_Y$，计算 $Z = xY$，运行证明算法 $\pi_Z \leftarrow \mathsf{P}(I_Z = (G, Q, Y, Z), x)$，并选择随机数 $k \leftarrow \mathbb{Z}_q$，计算 $kY = (r_x, r_y)$，$r = r_x \bmod q$，$\hat{s} = k^{-1}(H(m) + rx) \bmod q$，输出预签名值 $\hat{\sigma} = (r, \hat{s}, Z, \pi_Z)$；
- $\mathsf{pVrfy}(Q, m, I_Y, \hat{\sigma}) \to 1/0$: 预签名验证算法输入公钥 $Q$，消息 $m$ 和困难实例 $I_Y$ 和预签名值 $\hat{\sigma}$，首先验证 $Z$ 结构的证明 $\pi_Z$，如果 $\mathsf{V}(I_Z, \pi_Z) \to 0$，输出 0，否则计算 $\hat{s}^{-1}(H(m)Y + rZ) = (r_x, r_y)$，如果 $r = r_x \bmod q$ 则输出 1，否则输出 0;
- $\mathsf{Adapt}(y, \hat{\sigma}) \to \sigma$: 适配器算法输入证据 $y$ 和预签名值 $\sigma$，计算 $s = \hat{s}/y \bmod q$，并输出签名值 $\sigma = (r, s)$；
- $\mathsf{Ext}(\sigma, \hat{\sigma}, I_Y) \to y$: 提取算法输入签名值 $\sigma$，预签名值 $\hat{\sigma}$ 和困难实例 $I_Y$，计算 $y = \hat{s}/s \bmod q$，如果 $(I_Y, y) \in R$，则输出 $y$，否则输出 $\perp$；

**定理 3** *如果 ECDSA 是安全的且 $R$ 是困难关系，则 ECDSA 适配器签名是安全的。*

**支持离线证明的 ECDSA 适配器签名.** 令 $\mathsf{P}$ 和 $\mathsf{V}$ 分别表示证明和验证算法，令困难关系 $R_Z = \{(I_Z = (G, Q, Y, Z, \pi_Y, \pi_Z), y) | Y = yG \wedge Z = yQ \wedge \mathsf{V}(Y,$

$\pi_Y) = 1 \wedge \mathsf{V}(G, Y, Q, Z, \pi_Z) = 1\}$，$\pi_Y$ 表示存在证据 $y$ 使得 $Y = yG$ 成立，$\pi_Z$ 表示存在证据 $y$ 使得 $Y = yG$ 且 $Z = yQ$ 成立。

- $\mathsf{pSign}(Q, x, m, I_Z) \to \hat{\sigma}$: 预签名算法输入公钥 $Q$ 和签名私钥 $x$，消息 $m$，以及困难实例 $I_Z$，如果 $\mathsf{V}(I_Z) \to 0$，输出 0，否则选择随机数 $k \leftarrow \mathbb{Z}_q$，计算 $kY = (r_x, r_y)$，$r = r_x \bmod q$，$\hat{s} = k^{-1}(H(m) + rx) \bmod q$，输出预签名值 $\hat{\sigma} = (r, \hat{s})$;

- $\mathsf{pVrfy}(Q, m, I_Z, \hat{\sigma}) \to 1/0$: 预签名验证算法输入公钥 $Q$，消息 $m$ 和困难实例 $I_Z$ 和预签名值 $\hat{\sigma}$，首先验证 $Z$ 结构的证明 $\pi_Z$，如果 $\mathsf{V}(I_Z) \to 0$，输出 0，否则计算 $\hat{s}^{-1}\ (H(m)Y + r\ Z) = (r_x, r_y)$，如果 $r = r_x \bmod q$ 则输出 1，否则输出 0;

- $\mathsf{Adapt}(y, \hat{\sigma}) \to \sigma$: 适配器算法输入证据 $y$ 和预签名值 $\hat{\sigma}$,计算 $s = \hat{s}/y \bmod q$，并输出签名值 $\sigma = (r, s)$;

- $\mathsf{Ext}(\sigma, \hat{\sigma}, I_Z) \to y$: 提取算法输入签名值 $\sigma$，预签名值 $\hat{\sigma}$ 和困难实例 $I_Z$，计算 $y = \hat{s}/s \bmod q$，如果 $(I_Z, y) \in R$，则输出 $y$，否则输出 $\perp$;

**支持分布式密钥生成的适配器签名.** 关于困难关系 $R$ 和两方签名算法的两方适配器签名包含交互式预签名协议 IpSign，预签名验证算法 pVrfy，适配算法 Adapt 和提取算法 Ext。其中，两方签名算法包含交互式密钥生成算法 IGen 和交互式签名算法 ISign，以及签名验证算法 Vrfy。

- $\mathsf{IpSign}(sk_0, sk_1, m, Y) \to \hat{\sigma}$: 交互式预签名协议输入签名消息和困难实例 $Y$，以及参与方 A 的私钥 $sk_0$ 和参与方 B 的私钥 $sk_1$，输出预签名值 $\hat{\sigma}$;

- pVrfy，Adapt 和 Ext 是非交互的和适配器签名中的对应算法相同。

两方适配器签名的正确性和可适配性类似适配器签名，除了使用 IGen 和 IpSign 代替 Gen 和 pSign 算法

不可伪造性的定义类似两方签名，除了敌手在签名伪造实验中可以访问额外的预签名预言机 $\Pi^b_{\mathrm{IpSign}}$ 。具体而言，在签名伪造实验 $\mathrm{aSigForge}^b_{\mathcal{A},\Pi,\sum}(\lambda)$ 中，敌手 $\mathcal{A}$ 可以访问交互式密钥生成预言机 $\Pi^b_{\mathrm{IGen}}$，交互式签名预言机 $\Pi^b_{\mathrm{ISign}\langle sk_{1-b},\cdot\rangle}$ 和交互式预签名预言机 $\Pi^b_{\mathrm{IpSign}\langle sk_{1-b},\rangle}$。上述预言机模拟诚实参与方操作。其中，交互式密钥生成预言机和交互式签名预言机的定义和两方签名的不可伪造实验中的定义相同。令交互式密钥生成预言机，交互式签名预言机和交互式预签名预言机初始化机器 $M$，$M_{sid}$，$M_{\langle pS,sid\rangle}$，其中 $M_{sid}$，$M_{\langle pS,sid\rangle}$ 拥有 $M$ 的状态。

定义交互式预签名预言机如下:

1) 接收到 $(sid, m, Y)$，检测是首次接收到会话符号 $sid$，则初始化新的机器 $M_{\langle pS,sid\rangle}$。该机器拥有 $M$ 在交互式密钥生成结束时存储的签名私钥分享 $sk_{(1-b)}$ 和其他状态，输入消息 $m$，以及困难实例 $Y$，并模拟诚实的参与方执行交互式签名协议；

2) 接收到 $(sid, m, Y)$，检测不是首次接收到会话符号 $sid$ 时，预言机操作机器 $M_{\langle pS,sid\rangle}$ 进行交互，如果机器 $M_{\langle pS,sid\rangle}$ 结束，则输出机器 $M_{\langle pS,sid\rangle}$ 的输出。

**定义 1** *支持交互式密钥生成的两方适配器签名满足不可伪造性，如果对任意的 PPT 敌手，存在可忽略函数* $\mathsf{negl}$，$b \in \{0,1\}$，*满足*

$$\Pr\ [aSigForge^{b}_{\mathcal{A},\Pi,\sum}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

其中签名伪造实验 $\mathrm{aSigForge}^{b}_{\mathcal{A},\Pi,\sum}$ 定义如下：

$\mathrm{aSigForge}^{b}_{\mathcal{A},\Pi_{\mathrm{R},\sum_2}}(\lambda)$

$(vk, sk_b) \leftarrow \mathcal{A}^{\Pi^b_{\mathrm{IKG}}}(1^\lambda)$

$m^* \leftarrow \mathcal{A}^{\Pi^b_{\mathrm{ISign}_{\langle sk_{1-b},\cdot\rangle}},\Pi^b_{\mathrm{IpSign}_{\langle sk_{1-b},\cdot\rangle}}}(vk, sk_b)$

$(Y, y) \leftarrow \mathsf{GenR}(1^\lambda)$

$\hat{\sigma^*} \leftarrow \Pi^{\mathcal{A}}_{\mathrm{IpSign}_{\langle sk_{1-b},\cdot\rangle}}(m^*, Y)$

$\sigma^* \leftarrow \mathcal{A}^{\Pi^b_{\mathrm{ISign}_{\langle sk_{1-b},\cdot\rangle}},\Pi^b_{\mathrm{IpSign}_{\langle sk_{1-b},\cdot\rangle}}}(\hat{\sigma^*}, Y)$

return $m^* \notin \mathcal{Q} \wedge \mathsf{Vrfy}_{vk}(m^*, \sigma^*) = 1$

$\Pi^b_{\mathrm{IKG}}(\cdot,\cdot)$

$((vk, sk_b); (vk, sk_{1-b})) \leftarrow \Pi^b_{\mathrm{IKG}}(\cdot,\cdot)$

$\Pi^b_{\mathrm{ISign}_{\langle sk_{1-b},\cdot\rangle}}(sid, m)$

$\sigma \leftarrow \Pi^b_{\mathrm{ISign}_{\langle sk_{1-b},\cdot\rangle}}(sid, m)$

$\mathcal{Q} = \mathcal{Q} \cup \{m\}$

$\Pi^b_{\mathrm{IpSign}_{\langle sk_{1-b},\cdot\rangle}}(sid, m, Y)$

$\hat{\sigma} \leftarrow \Pi^b_{\mathrm{IpSign}_{\langle sk_{1-b},\cdot\rangle}}(sid, m, Y)$

$\mathcal{Q} = \mathcal{Q} \cup \{m\}$

**定义 2** *支持交互式密钥生成的两方适配器签名满足证据可提取性，如果对任意的 PPT 敌手，存在可忽略函数* $\mathsf{negl}$，$b \in \{0,1\}$，*满足*

$$\Pr\ [aWitExt^{b}_{\mathcal{A},\Pi_{\mathrm{R},\sum_2}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

其中证据提取实验 $\mathrm{aWitExt}^{b}_{\mathcal{A},\Pi_{\mathrm{R},\sum_2}}$ 定义如下：

$\mathrm{aWitExt}^{b}_{\mathcal{A},\Pi_{\mathsf{R}},\Sigma_2}(\lambda)$

$(vk, sk_b) \leftarrow \mathcal{A}^{\Pi^b_{\mathrm{IKG}}}(1^\lambda)$

$(m^*, Y^*) \leftarrow \mathcal{A}^{\Pi^b_{\mathrm{ISign}\langle sk_{1-b},\cdot\rangle}, \Pi^b_{\mathrm{IpSign}\langle sk_{1-b},\cdot\rangle}}(vk, sk_b)$

$\hat{\sigma^*} \leftarrow \Pi^{\mathcal{A}}_{\mathrm{IpSign}\langle sk_{1-b},\cdot\rangle}(m^*, Y)$

$\sigma^* \leftarrow \mathcal{A}^{\Pi^b_{\mathrm{ISign}\langle sk_{1-b},\cdot\rangle}, \Pi^b_{\mathrm{IpSign}\langle sk_{1-b},\cdot\rangle}}(\hat{\sigma^*}, Y)$

$y' = \mathsf{Ext}(\sigma^*, \hat{\sigma^*}, Y^*)$

return $m^* \notin \mathcal{Q} \wedge \mathsf{Vrfy}_{vk}(m^*, \sigma^*) = 1 \wedge (Y^*, y') \notin \mathsf{R}$

$\Pi^b_{\mathrm{IKG}}(\cdot,\cdot)$

$((vk, sk_b); (vk, sk_{1-b})) \leftarrow \Pi^b_{\mathrm{IKG}}(\cdot,\cdot)$

$\Pi^b_{\mathrm{ISign}\langle sk_{1-b},\cdot\rangle}(sid, m)$

$\sigma \leftarrow \Pi^b_{\mathrm{ISign}\langle sk_{1-b},\cdot\rangle}(sid, m)$

$\mathcal{Q} = \mathcal{Q} \cup \{m\}$

$\Pi^b_{\mathrm{IpSign}\langle sk_{1-b},\cdot\rangle}(sid, m, Y)$

$\hat{\sigma} \leftarrow \Pi^b_{\mathrm{IpSign}\langle sk_{1-b},\cdot\rangle}(sid, m, Y)$

$\mathcal{Q} = \mathcal{Q} \cup \{m\}$

**定理 4** 支持分布式密钥生成的两方适配器签名算法是安全的，如果该算法满足两方预签名值的可适配性，两方适配器签名的选择消息不可伪造性，以及两方适配器签名的证据可提取性。

**两方 ECDSA 适配器签名.** 两方 ECDSA 适配器签名协议主要包含交互式密钥生成算法 IGen 和交互式预签名算法 IpSign，其中交互式密钥生成算法 IGen 和两方 ECDSA 签名的密钥生成算法相同，预签名验证算法、适配算法和提取算法和 ECDSA 适配签名相同。协议描述如下：

- $\mathsf{IGen}(pp) \to ((sk_0, vk), (sk_1, vk))$:

  1) 参与方 A 选择私钥 $x_0 \leftarrow \mathbb{Z}_q$，计算 $Q_0 = x_0 G$，然后 A 发送 (com-prove, 0, $Q_0$, $x_0$) 给 $\mathcal{F}^{R_{\mathrm{DL}}}_{\text{com-zk}}$；
  2) 参与方 B 接收 $\mathcal{F}^{R_{\mathrm{DL}}}_{\text{com-zk}}$ 的消息 $(\mathsf{proof\text{-}receipt}, 0)$，选择私钥 $x_1 \leftarrow \mathbb{Z}_q$，计算 $Q_1 = x_1 G$，然后 A 发送 $(\mathsf{prove}, 1, Q_1, x_1)$ 给 $\mathcal{F}^{R_{\mathrm{DL}}}_{\mathrm{zk}}$；
  3) A 接收 $\mathcal{F}^{R_{\mathrm{DL}}}_{\mathrm{zk}}$ 的消息 $(\mathsf{proof}, 1, Q_1)$ 并验证，如果接受，则计算 $Q = x_1 Q_2$，并运行 Paillier 加密的密钥生成算法 $\mathsf{Gen}_{\mathrm{enc}}(1^\lambda) \to (pk_{\mathrm{enc}}, sk_{\mathrm{dec}})$，其中，$pk_{\mathrm{enc}} = N = p_1 p_2$，加密 $x_0$ 获得 $c_{\mathrm{key}} \leftarrow \mathsf{Enc}(pk_{\mathrm{enc}}, x_0)$，发送 $(\mathsf{decom\text{-}proof}, 0)$ 给 $\mathcal{F}^{R_{\mathrm{DL}}}_{\text{com-zk}}$，发送 $(\mathsf{prove}, 0, N, (p_1, p_2))$ 给 $\mathcal{F}^{R_{\mathrm{P}}}_{\mathrm{zk}}$，发送 $(\mathsf{proof}, c_{\mathrm{key}})$ 给 B；本地存储 $(sk_0, vk) = ((x_0, sk_{\mathrm{dec}}), (Q, c_{\mathrm{key}}))$;
  4) B 接收 $\mathcal{F}^{R_{\mathrm{DL}}}_{\text{com-zk}}$ 的消息 $(\mathsf{decom\text{-}proof}, 0, Q_0)$，接收 $\mathcal{F}^{R_{\mathrm{P}}}_{\mathrm{zk}}$ 的消息 $(\mathsf{proof}, 0, N)$，接收 A 的消息 $(proof, c_{\mathrm{key}})$，B 验证接收的消息，如果通过则计算 $Q = x_2 Q_1$，并本地存储 $(sk_1, vk) = (x_1, (Q, c_{\mathrm{key}}, pk_{\mathrm{enc}}))$;

- $\mathsf{IpSign}(sk_0, sk_1, m, I_Y) \to \hat{\sigma}$:

  1) 参与方 A 选择随机数 $k_0 \leftarrow \mathbb{Z}_q$，计算 $R_0 = k_0 Y$，然后 A 发送 (com-prove, $sid||0$, $R_0$, $k_0$) 给 $\mathcal{F}^{R_{\mathrm{DL}}}_{\text{com-zk}}$；

2) 参与方 B 接收 $\mathcal{F}_{\text{com-zk}}^{R_{\text{DL}}}$ 的消息 $(\mathsf{proof\text{-}receipt}, sid||0)$，然后选择随机数 $k_1 \leftarrow \mathbb{Z}_q$，计算 $R_1 = k_1 Y$，并发送消息 $(\mathsf{prove}, sid||1, R_1, k_1)$ 给 $\mathcal{F}_{\text{zk}}^{R_{\text{DL}}}$；

3) A 接收 $\mathcal{F}_{\text{zk}}^{R_{\text{DL}}}$ 的消息 $(\mathsf{proof}, sid||1, R_1)$，如果验证通过，则计算 $R = k_0 R_1 = (r_x, r_y)$，$r = r_x \bmod q$，发送 $(\mathsf{decom\text{-}proof}, sid||0)$ 给 $\mathcal{F}_{\text{com-zk}}^{R_{\text{DL}}}$；

4) B 接收 $\mathcal{F}_{\text{com-zk}}^{R_{\text{DL}}}$ 的消息 $(\mathsf{decom\text{-}proof}, sid||0, R_0)$，验证通过则计算 $m' = H(m)$，$R = k_1 R_0 = (r_x, r_y)$，$r = r_x \bmod q$，随机选择 $\rho \leftarrow \mathbb{Z}_{q^2}$，计算 $c_1 = \mathsf{Enc}(pk_{\text{enc}}, \rho q + k_1^{-1} m' \bmod q)$，$v = k_1^{-1} r x_1 \bmod q$，计算 $c_2 = v \odot c_{\text{key}}$，$c_3 = c_1 \oplus c_2$，并发送 $c_3$ 给 A；

5) A 接收到消息 $c_3$，解密 $s' = \mathsf{Dec}(sk_{\text{dec}}, c_3)$，$s'' = k_0^{-1} s' \bmod q$，$s = \min\{s'', q - s''\}$，即预签名值 $\sigma = (r, s)$，验证预签名的正确性，如果通过则输出预签名值 $\sigma = (r, s)$。

**实验分析.** 我们基于 OpenSSL 库，使用 C++ 语言，实现了 ECDSA 适配器签名，我们的 ECDSA 适配器签名，以及支持离线证明的 ECDSA 适配器签名方案。实验结果表明，支持离线证明的 ECDSA 适配器签名的在线预签名效率比现有的 ECDSA 适配器签名快了近 2.5 倍，运行在线预签名操作 1000 次，三个方案的预签名操作的平均每次耗时分别为 174.75$\mu s$, 198.91$\mu s$ 和 74.74$\mu s$。实验结果如图 3 所示。

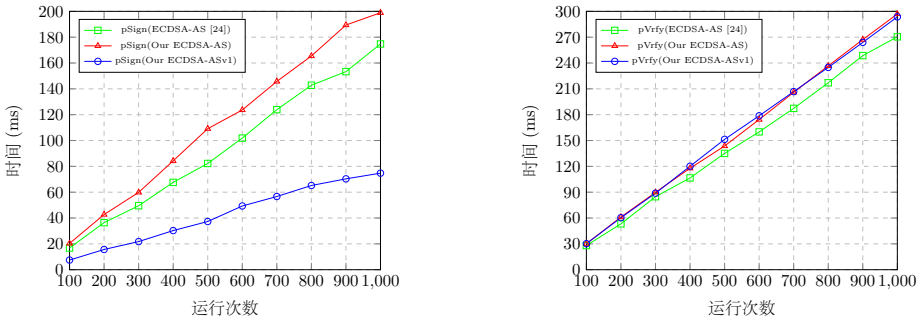


图 2: 效率对比 (左) 预签名算法，(右) 预签名验证算法

**门限 ECDSA 适配器签名.** 目前我们只给出了两方场景下的 ECDSA 适配器签名构造，缺少更加通用的门限 ECDSA 适配器签名方案的研究。一方面，目前并没有门限适配器签名的形式化定义；另一方面，ECDSA 签名结构不具有线性组合的性质，难以使用线性秘密分享直接扩展，需要使用较为复杂的安全多方计算技术构造，比如同态加密和不经意传输等。因此，后续工作中，我们可以借助现有门限 ECDSA 签名的形式化定义，根据适配器签名的性质，探索门限适配器签名的形式化定义，同时构造高效的门限 ECDSA 适配器签名并给出安全性证明。

#### 4.2.5 门限 SM2 适配器签名

**SM2 适配器签名.** SM2 适配器签名是 SM2 签名和困难关系的扩展, 主要包含了预签名算法 pSign、预签名验证算法 pVrfy、适配算法 Adapt、证据提取算法 Ext, 以及 SM2 的密钥生成算法 Gen、签名算法 Sign 和签名验证算法 Vrfy. 令 $R = \{I_Y = (Y, \pi), y | Y = yG \wedge \mathsf{V}(I_Y) = 1\}$ 和 $R_Z = \{(I_Z = (G, X, Y, Z), x) | X = xG \wedge Z - Y = xY\}$, P 和 V 分别表示零知识证明系统中的证明和验证算法。基于 SM2 的适配器签名算法 SM2-ASv1 构造如下:

- $\mathsf{pSign}((X, x), m, I_Y) \to \hat{\sigma}$. 预签名算法输入公私钥对 $(X, x)$, 消息值 $m$ 和困难实例 $I_Y = (Y, \pi_Y)$, 计算 $Z = (x+1)Y$, 运行 $\mathsf{P}(I_Z = (G, X, Y, Z), x) \to \pi_Z$, 计算 $e = H(m)$, 选择随机数 $k \leftarrow \mathbb{Z}_n$, 并计算 $\hat{K} = kG + Z$, $r_x = f(\hat{K})$, $r = e + r_x$, $\hat{s} = (1+x)^{-1}(k - rx) \bmod n$, 最后输出预签名值 $\hat{\sigma} = (r, \hat{s}, Z, \pi_Z)$.

- $\mathsf{pVrfy}(X, m, I_Y, \hat{\sigma}) \to 0/1$. 预签名验证算法输入公钥 $X$, 消息 $m$, 困难实例 $I_Y = (Y, \pi_Y)$, 以及预签名值 $\hat{\sigma} = (r, \hat{s}, Z, \pi_Z)$, 运行 $\mathsf{V}(I_Z, \pi_Z) \to 0$, 则输出 0, 否则计算 $e = H(m)$, $\hat{K} = (\hat{s} + r)X + \hat{s}G + Z$, $r' = f(\hat{K}) + e \bmod n$. 如果 $r' = r$, 该算法输出 1, 否则输出 0.

- $\mathsf{Adapt}(\hat{\sigma}, y) \to \sigma$. 适配算法输入预签名值 $\hat{\sigma}$ 和证据 $y$, 计算 $s = \hat{s} + y \bmod n$, 并输出签名值 $\sigma = (r, s)$.

- $\mathsf{Ext}(\sigma, \hat{\sigma}, I_Y) \to y$. 证据提取算法输入签名值 $\sigma$, 预签名值 $\hat{\sigma}$ 和困难实例 $I_Y = (Y, \pi_Y)$, 计算 $y = s - \hat{s} \bmod n$. 如果 $(Y, y) \in R$, 输出 $y$, 否则输出 $\perp$.

**定理 5** *如果 SM2 签名 $\sum_{SM2}$ 满足 SUF−CMA, 且 $R$ 是一个困难关系, 则上述方案 $\Pi_{R,\sum_{SM2}}$ 在随机预言机模型下是安全的.*

**支持离线证明的 SM2 适配器签名.** 我们对 SM2 适配器签名 SM2-ASv1 进一步优化, 基于离线证明技术设计更加高效的 SM2 适配器签名。根据 $Z = (x+1)Y = y(X + G)$, 可分别以 $x$ 和 $y$ 为证据证明 $Z$ 的结构，其中，以 $x$ 为证据，该证明只能由预签名方生成；以 $y$ 为证据，该证明可由困难关系生成方生成。

根据 SM2-ASv1，该方案主要包含了两对困难关系 $((Y, \pi_Y), y)$ 和 $((G, X, Y, Z), x)$ 的零知识证明，其中 $((Y, \pi_Y), y)$ 的证明可由困难关系生成方离线生成并证明，后者在预签名算法阶段以预签名私钥 $x$ 为证据，证明离散对数相等的困难关系 $\{((G, X, Y, Z), x) | \exists\ x \in \mathbb{Z}_n, \text{ s.t. } X = xG \wedge Z - Y = xY\}$。根据 SM2 签名算法特点，$((G, X, Y, Z), x)$ 的证明过程与预签名过程中的随机数选择以及签名消息是独立的，该部分可由预签名方在离线阶段执行。离线证明构造较为直观，即预签名方在接收 (或生成) 困难关系 $(Y, \pi_Y)$ 后，先离线生成 $Z$ 并进行

离线证明，后续对消息 $m$ 进行预签名时，再选择随机数并生成预签名值。因此，该方案可根据 SM2-ASv1 直接得到。同时，该过程依旧保持困难关系 $((Y,\pi_Y),y)$ 的自证明结构，并不影响 SM2 适配器签名算法的安全性。

本节主要构造由困难关系生成方以 $y$ 为证据离线证明 $Z$ 结构的 SM2 适配器签名方案 SM2-ASv2。令离散对数相等困难关系为 $R=\{(I=(G,I_Y=(Y,\pi_Y),X,Z,\pi),y)|\exists\ y\in\mathbb{Z}_n,\ \text{s.t. } Y=yG\wedge Z-Y=yX\wedge \mathsf{V}(I)\to 1\}$，其中 $\mathsf{P}$ 和 $\mathsf{V}$ 分别表示零知识证明中的证明和验证算法。基于 SM2 的适配器签名算法构造如下:

- $\mathsf{pSign}((X,x),m,I)\to\hat{\sigma}$. 预签名算法输入公私钥对 ($X$,$x$), 消息值 $m$ 和困难实例 $I$=($G$, $I_Y$, $X$, $Z$, $\pi$), 可离线验证 $I$. 如果 $\mathsf{V}(I)\to 0$, 输出 $\perp$, 否则计算 $e=H(m)$, 选择随机数 $k\leftarrow\mathbb{Z}_n$, 并计算 $\hat{K}=kG+Z$, $r_x=f(\hat{K})$, $r=e+r_x$, $\hat{s}=(1+x)^{-1}(k-rx) \bmod n$, 最后输出预签名值 $\hat{\sigma}=(r,\hat{s})$.

- $\mathsf{pVrfy}(X,m,I,\hat{\sigma})\to 0/1$. 预签名验证算法输入公钥 $X$, 消息 $m$, 困难实例 $I=(G,I_Y,X,Z,\pi)$, 以及预签名值 $\hat{\sigma}=(r,\hat{s})$, 可离线验证 $I$, 如果 $\mathsf{V}(I)\to 0$, 输出 $\perp$, 否则计算 $e=H(m)$, $\hat{K}=(\hat{s}+r)X+\hat{s}G+Z$, $r'=f(\hat{K})+e \bmod n$. 如果 $r'=r$, 该算法输出 1, 否则输出 0.

- $\mathsf{Adapt}(\hat{\sigma},y)\to\sigma$. 适配算法输入预签名值 $\hat{\sigma}=(r,\hat{s})$ 和证据 $y$, 计算 $s=\hat{s}+y \bmod n$, 并输出签名值 $\sigma=(r,s)$.

- $\mathsf{Ext}(\sigma,\hat{\sigma},I)\to y$. 证据提取算法输入签名值 $\sigma$, 预签名值 $\hat{\sigma}$ 和困难实例 $I$, 计算 $y=s-\hat{s} \bmod n$. 如果 $(I,y)\in R$, 输出 $y$, 否则输出 $\perp$.

高效的 SM2 适配器签名 SM2-ASv2 的困难实例 $I$ 包含 $Y=yG$ 和 $Z=y(X+G)$，皆以 $y$ 为证据，具体的零知识证明可由困难关系生成方生成。因此，预签名方接收到该困难实例 $I$ 可直接离线验证其正确性，随后需要对消息进行预签名时，根据离线验证的 $Y,Z$，并选择随机数对消息进行预签名。该方法避免了在线预签名阶段中计算 $Z$ 的过程，并可通过零知识证明其结构，与在线 SM2-ASv1 和现有的 ECDSA 适配器签名相比较，SM2-ASv2 在线预签名的计算效率更高。由于 $Z$ 的零知识证明加入到离线证明的困难关系中，预签名中不需要过多的包含 $Z$ 的信息。因此，SM2-ASv2 的签名值更小。同样，根据对 SM2-ASv2 的分析，离线证明过程依旧保证自证明结构 $((Y,\pi_Y),y)$，不影响适配器签名的安全性。

**实验分析.** 我们基于 OpenSSL 库，使用 C++ 语言，实现了别实现了 ECDSA 适配器签名 (ECDSA-AS)，SM2 适配器签名 (SM2-ASv1) 和支持离线证明的 SM2 适配器签名 (SM2-ASv2)。我们对比了三种方案在线预签名操作的计算效

率以及预签名验证算法的计算效率。我们分别运行各方案的程序 1000 次, 发现 ECDSA-AS, SM2-ASv1 和 SM2-ASv2 的在线预签名操作单次平均耗时分别为 166.54$\mu s$，176.27$\mu s$，37.14$\mu s$；预签名验证操作单次平均耗时分别为 252.86$\mu s$，238.46$\mu s$，234.80$\mu s$。实验结果符合理论分析，SM2-ASv2 的在线预签名耗时仅为 ECDSA-AS 的 1/4。

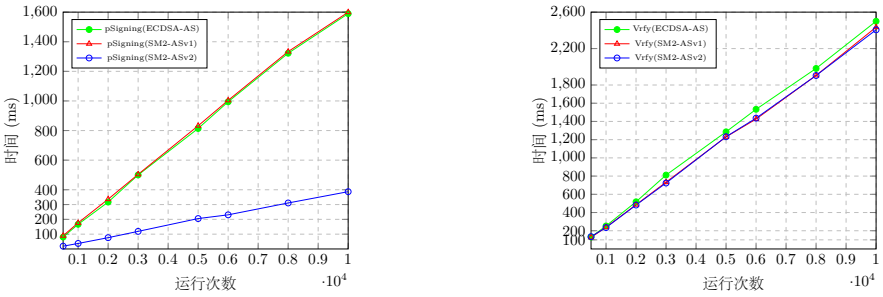


图 3: 效率对比 (左) 预签名算法和 (右) 验证算法

**门限 SM2 适配器签名.** 目前只给出了 SM2 适配器签名，缺少门限 SM2 适配器签名方案的研究。后续工作中，我们可基于两方 ECDSA 适配器签名协议的设计方法，首先对现有的 SM2 适配器签名进行两方扩展，并基于支持交互式密钥生成的两方适配器签名的形式化定义给出可证明安全。然后，在两方的基础上，探索更加广泛的门限 SM2 适配器签名的构造。

#### 4.2.6 门限 SM2 签密方案

**SM2 签密方案.** 根据签密的模块化构造，我们分别采用椭圆曲线上的 Pedersen 承诺 [35]、SM2 签名算法和 SM2 加密算法实例化上述通用构造，获得了具有消息同态计算功能的 SM2 签密方案，主要包含参数建立、密钥生成算法、承诺算法、签密算法、验证算法、解密算法和打开算法。具体构造如下：

参数建立阶段：椭圆曲线系统参数包括有限域 $\mathbb{F}_q$ 的规模 $q$，定义椭圆曲线 $E(\mathbb{F}_q)$ 的方程的两个元素 $a, b \in \mathbb{F}_q$，$E(\mathbb{F}_q)$ 上的基点 $G = (x_G, y_G)$，其中 $x_G$ 和 $y_G$ 是 $\mathbb{F}_q$ 中的两个元素，$G$ 的阶为 $n$。

- $\mathsf{Gen}(1^\lambda) \to (pp = (Q, vk, pk); sp = (sk, dk))$：密钥生成算法输入安全参数，SM2 椭圆曲线系统参数，运行 SM2 签名的密钥生成算法，输出验证公钥和签名私钥 $(vk, sk)$，运行 SM2 加密的密钥生成算法，输出加密公钥和解密私钥 $(pk, dk)$，选择随机数 $v \in [1, n-1]$，计算承诺方案参数 $Q = vG$; 然后输出算法公开参数：承诺方案参数 $Q$、签名验证公钥 $vk$ 和加密公钥 $pk$; 秘密参数：签名私钥 $sk$ 和解密私钥 $dk$;

- $\mathsf{Commit}(Q, m) \to (c, d)$：承诺算法输入承诺方案参数 $Q$ 和消息 $m$；选择随机数 $k \in [1, n-1]$，计算承诺值 $c = m \cdot G + k \cdot Q$；然后输出：承诺值 $c$ 和承诺打开参数 $d = (k, m)$；

- $\mathsf{Sign\text{-}Enc}(pk, sk, c, d) \to (\sigma, ct)$：签密算法输入：加密公钥 $pk$、签名私钥 $sk$、承诺值 $c$ 和承诺打开参数 $d$；运行 SM2 签名算法和 SM2 加密算法，计算 $\sigma = \mathrm{SM2.}\mathsf{Sign}(sk, c)$ 和 $ct = \mathrm{SM2.}\mathsf{Enc}(pk, d)$；输出签密值 $sc = (\sigma, ct)$；

- $\mathsf{Vrfy}(vk, c, \sigma) \to 1/0$：验证算法输入：验证公钥 $vk$、承诺值 $c$ 和签名值 $\sigma$；运行 SM2 签名算法，计算 $\mathrm{SM2.}\mathsf{Vrfy}(vk, \sigma, c) \to 0/1$，输出 1 表示验证正确，0 表示验证错误；

- $\mathsf{Dec}(dk, ct) \to d$：解密算法输入：解密私钥 $dk$ 和密文 $ct$，运行 SM2 解密算法，计算 $d = \mathrm{SM2.}\mathsf{Dec}(dk, ct)$；输出：承诺打开参数 $d$；

- $\mathsf{Open}(d, c) \to m/$ ：打开算法输入：承诺打开参数 $d = (k, m)$，承诺值 $c$，以及承诺方案参数 $G, Q$；判断 $c = m \cdot G + k \cdot Q$ 是否成立，成立则输出消息 $m$，否则输出 ；

**正确性.** 上述构造满足签密的正确性，具体描述如下：对于任意消息 $m$ 的承诺 $(c, d) \leftarrow \mathsf{Commit}(m)$，进行签密获得 $\sigma = \mathrm{SM2.}\mathsf{Sign}(sk, c)$ 和 $ct = \mathrm{SM2.}\mathsf{Enc}(pk, d)$，使用 SM2 的签名验证算法可正确验证的承诺值,$\mathrm{SM2.}\mathsf{Verify}(vk, \sigma, c) \to 1$，通过 SM2 解密算法解密出打开参数,可正确打开消息值 $m \leftarrow \mathsf{Open}(c, \mathrm{SM2.}\mathsf{Dec}(dk, ct))$。

**加法同态性质.** 我们采用了椭圆曲线上的 Pedersen 承诺方案 [35] 实例化该加法同态承诺，即：

- 消息 $m_1$ 的承诺值 $c_1 = m_1 \cdot G + k_1 \cdot Q, d_1 = (k_1, m_1)$; 签密计算为：$\mathsf{Sign\text{-}Enc}(pk,$ $sk,$ $c_1,$ $d_1) \to (\sigma_1,$ $ct_1)$

- 消息 $m_2$ 的承诺值 $c_2 = m_2 \cdot G + k_2 \cdot Q, d_2 = (k_2, m_2)$; 签密计算为：$\mathsf{Sign\text{-}Enc}(pk,$ $sk,$ $c_2,$ $d_2) \to (\sigma_2,$ $ct_2)$

  签密的同态运算如下：

1) 同态运算：$c_{1,2} = c_1 + c_2 = (m_1 + m_2) \cdot G + (k_1 + k_2) \cdot Q$,$d_{1,2} = (k_1 + k_2, m_1 + m_2)$

   解签密过程运行如下：

1) 分别验证签密的正确性：$\mathsf{Verify}(vk, c_1, \sigma_1) \to 1$,$\mathsf{Verify}(vk, c_2, \sigma_2) \to 1$;

2) 分别运行解密算法：$\mathsf{Dec}(dk, ct_1) \to d_1$, $\mathsf{Dec}(dk, ct_2) \to d_2$;

3) 承诺打开算法：计算 $c'_{1,2} = (m_1 + m_2) \cdot G + (k_1 + k_2) \cdot Q$, 验证 $c'_{1,2} = c_{1,2}$ 是否成立？成立则输出 $m_1 + m_2$，否则输出 $\perp$.

**安全性分析.** 基于 SM2 的签密方案是对签密通用构造框架的实例化，根据 An J.H. 等 [36, 37] 的分析，采用安全的底层组件构造，可保证整体的签密方案的安全性。本方案采用了具有信息论安全的隐藏性和计算安全的绑定性的承诺方案、安全的 SM2 签名算法和 SM2 加密算法作为模块化构造组件，可实现整体基于 SM2 的签密方案的安全性：

- 签密的**机密性**由承诺方案的信息论安全的隐藏性和 SM2 加密算法的机密性保证；
- 签密的**完整性、认证**和**不可否认性**由承诺方案的计算安全的绑定性和 SM2 签名算法的完整性、认证和不可否认性保证。

**SM2 签密的两方签名扩展.** 两方交互式 SM2 签名适用于两个终端或者云加端的签名场景，对 SM2 的签名私钥进行了分布式的安全防护，即只获得单一终端的签名私钥无法进行签名。本方案基于两方交互式 SM2 签名算法，嵌入 SM2 签密方案，可获得两方交互式 SM2 签密方案，提升了签密方案中签名私钥的安全性。具体场景描述：终端 A(主) 和 B(副) 分别掌握部分签名私钥，并联合执行签密，其中终端 A 负责承诺和加解密过程，签名过程需要 A 和 B 协同完成。

- 两方签名密钥生成阶段
  1) A 选择随机数 $d_1$ 作为部分签名私钥, 计算 $Q_1 = d_1^{-1} \cdot G$, 并将 $Q_1$ 发送给 B;
  2) B 选择随机数 $d_2$ 作为部分签名私钥, 计算签名验证公钥为 $vk = d_2^{-1} \cdot Q_1 - G$, 并公开公钥 $vk$;
- 两方交互式签名阶段
  1) A 输入承诺值 $c$, 以 $c$ 为签名消息，按照标准 SM2 签名中的方式计算消息摘要 $e = H(Z, c)$；选择随机数 $k_1$, 计算 $G_1 = d^{-1}k_1 \cdot G$, 并将 $e, G_1$ 发送给 B；
  2) B 选择随机数 $k_2$, 计算 $(x_1, y_1) = d_2^{-1}k_2 \cdot G + d_2^{-1} \cdot G_1$; 计算 $r = (x_1 + e) \bmod n$, 若 $r = 0$, 则重新选择随机数 $k_2$ 计算，否则计算 $s_1 = d_2 r + k_2$ 发送给 A;
  3) A 计算 $s = d_1 s_1 + k_1 - r \bmod n$, 若 $s \neq 0$, 且 $s \neq r$, 则输出完整签名 $(s, r)$;

**实验分析.** 我们使用 C++ 语言实现了 SM2 签密方案的承诺、加密、签名、解密、验证、以及打开算法，并循环测试了 1000 次，获得各个操作的平均效率均是毫秒级，分别是承诺：1.18ms/次；加密：2.94ms/次；签名：1.48ms/次；验签：1.63ms/次；解密：1.54ms/次；承诺打开：0.75ms/次。

表 1: SM2 签密各算法的运行时间

| 操作 | 每次耗时 (ms) |
|---|---|
| 承诺 | 1.18 |
| 加密 | 2.94 |
| 签名 | 1.48 |
| 验签 | 1.63 |
| 解密 | 1.54 |
| 打开 | 0.75 |

**门限 SM2 签密.** 目前只给出了支持两方签名的 SM2 签密方案，门限 SM2 签密方案依旧缺少相关研究。后续工作中，我们首先研究门限签密的形式化定义，同时探索安全高效的门限 SM2 签名和加密方案，并以此为基础组件将现有的 SM2 签密方案扩展到门限版本。

## 5 论文进度安排和预期目标

- 2020.9——2021.11 跟踪调研安全多方计算的研究现状，以及安全多方计算技术在门限密码领域的应用情况。探索安全多方计算和门限密码存在的问题，积极思考并给出初步的解决方案。完成论文“ECDSA 适配器签名及其两方扩展”并投稿，初步完成“SM2 适配器签名及其应用”和“基于 ECDSA 的高效两方签名协议”论文的撰写。
- 2021.12——2022.08 深入学习现有安全多方计算协议的各类安全模型和存在性情况，以及各类通用安全多方计算协议的构造技术。熟练掌握安全多方计算协议的构造方法，以及相关的优化技术，并结合安全多方计算在门限密码中的应用，在安全模型、功能扩展，以及效率优化等方面给出更加安全实用的解决方案。初步给出对 GRR2 技术改进并对 Yao 协议优化实现，形成论文；在 MPC 应用方面，对已经构造的两方签名和适配器签名，探索更加通用的门限化构造，并发表论文。
- 2022.09——2023.01 跟进现有通用安全多方计算协议的效率优化技术，并积极探索新型的优化技术，在现有协议基础上给出更加高效的通用安全多方计算协议。聚焦安全多方计算的应用场景，构造更贴合具体应用的门限密码方案。构造出高效批量生产 Beaver 乘法组的方法，提升 BGW-协议的高效实现，并给出可证明安全，形成论文并发表。

- 2023.02——2023.12 积极探索新型的通用安全多方计算构造方法，同时研究不同应用场景下的敌手攻击形式，结合实际应用，探索更强的安全性模型，并在该模型下给出具体的安全多方计算协议的构造，发表论文。
- 2024.01——2024.06 撰写毕业论文，通过毕业答辩。

## 6 论文预期创新点

(1) 本文探索不同尺寸秘密份额的秘密分享技术，优化现有乱码电路行减少 (GRR2) 技术，突破现有 GRR2 无法和 free XOR 兼容的问题，并在此基础上高效实现 Yao 乱码电路，优化 Yao-协议通用 MPC 协议。

(2) 本文探索 BGW 类协议的效率优化技术，试图构造 Beaver 乘法组扩展协议，基于少量乘法组生成大量乘法组，避免现有基于同态加密生成乘法组的计算复杂度较高，以及基于不经意传输生成乘法组的通信复杂度较高的问题，并在此基础上高效实现 BGW-协议通用 MPC 协议。

(3) 本文基于 ECDSA 的两方签名算法避免使用复杂的安全多方计算技术和昂贵的零知识证明技术，相较于现有基于 ECDSA 的两方签名更加高效。本文实现的两方签名算法在签名过程中，各参与方的签名操作和 ECDSA 的签名操作类似，具体实现方案对于 ECDSA 签名变动较小，可重用较多的原始 ECDSA 算法的软硬件实现，对于存在的 ECDSA 应用进行两方升级扩展比较友好。

(4) 本文首次提出了支持交互式密钥生成的适配器签名的形式化定义,可为交互式密钥生成的适配器签名的安全性分析提供研究基础。本文实现的 ECDSA 适配器签名算法与现有 ECDSA 适配器签名相比，预签名阶段的零知识证明可以离线执行，保证在线预签名阶段更加高效。此外，我们对 ECDSA 适配器签名进行分布式设计，构造了高效的两方 ECDSA 适配器签名方案，并给出了可证明安全性分析。

(5) 本文首次基于国密 SM2 签名构造了适配器签名方案, 并基于 SM2 的安全性给出其在随机预言机模型下的安全性证明. 随后, 根据 SM2 签名的结构特点, 采用离线证明技术进行优化, 获得了可支持离线证明的 SM2 适配器签名方案，并基于适配器签名的现有应用, 分别给出 SM2 适配器签名在区块链上的两个具体应用：原子交换协议和支付通道网络, 为后续 SM2 签名算法的应用推广提供参考。

(6) 本文通过采用成熟的签密通用构造框架，基于 SM2 标准算法构造了安全高效的 SM2 签密方案。该方案通过模块化构造，可实现丰富的功能性扩展，比如：消息的同态运算性质，以及无证书、交互式、批量验签等丰富的应用功能，同时该方案可实现 SM2 签名和加密算法的并行执行，极大的提升了签密的应用效率 (maxcost 签名，cost 加密 <cost 签名 +cost 加密)。

## 7 课题研究的条件、经费支持及导师组人员配备情况

山东大学网络空间安全学院（研究院），老师们拥有雄厚的知识基础，优秀的科研攻关能力，实验室配备了先进的计算机设备。项目课题：实验室承担了“973”项目、国家自然科学基金重点项目、国家杰出青年基金项目课题，提供了良好的学习科研环境与财力支持。实验室为科研的顺利进行提供了强有力的保障。

## 8 可能的困难与解决途径

(1) 研究前期很有可能遇到的困难是对晦涩难懂的学术论文的理解，因为知识储备的不足，思维方式的偏差，对论文的理解很有可能有偏差和误解。解决途径：加强交流，与老师和同学们多沟通交流，共同探讨对同一个问题、同一个方法的理解和想法；

(2) 阅读相关文献，通常研究人员会对一个问题有多方面的研究，相似或者互补，而论文中的参考文献正为我们提供了了解相关领域最直接的指导，在费解论文要点时，对相关文献的阅读往往能为我们提供新的视角，从而更全面地掌握相关技术与方法。

(3) 后面很有可能遇到的是创新工作的困难，相信厚积薄发，认真钻研前人经验很重要，但也要养成独立思考问题的习惯。具体问题具体分析，方法不太可能天然普适，这就需要我们依据具体算法先做其特性的研究。

## 参考文献

[1] Goldreich, O.: The Foundations of Cryptography - Volume 2: Basic Applications. Cambridge University Press (2004), http://www.wisdom.weizmann.ac.il/%7Eoded/foc-vol2.html

[2] Yao, A.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois,

USA, 3-5 November 1982. pp. 160–164. IEEE Computer Society (1982), https://doi.org/10.1109/SFCS.1982.38

[3] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A.V. (ed.) Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA. pp. 218–229. ACM (1987), https://doi.org/10.1145/28395.28420

[4] Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA. pp. 503–513. ACM (1990), https://doi.org/10.1145/100216.100287

[5] Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations. Lecture Notes in Computer Science, vol. 5126, pp. 486–498. Springer (2008), https://doi.org/10.1007/978-3-540-70583-3_40

[6] Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Feldman, S.I., Wellman, M.P. (eds.) Proceedings of the First ACM Conference on Electronic Commerce (EC-99), Denver, CO, USA, November 3-5, 1999. pp. 129–139. ACM (1999), https://doi.org/10.1145/336992.337028

[7] Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5912, pp. 250–267. Springer (2009), https://doi.org/10.1007/978-3-642-10366-7_15

[8] Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole - reducing data transfer in garbled circuits using half gates. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques,

Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 220–250. Springer (2015), https://doi.org/10.1007/978-3-662-46803-6_8

[9] Rosulek, M., Roy, L.: Three halves make a whole? beating the half-gates lower bound for garbled circuits. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12825, pp. 94–124. Springer (2021), https://doi.org/10.1007/978-3-030-84242-0_5

[10] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Simon, J. (ed.) Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. pp. 1–10. ACM (1988), https://doi.org/10.1145/62212.62213

[11] Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Feigenbaum, J. (ed.) Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings. Lecture Notes in Computer Science, vol. 576, pp. 420–432. Springer (1991), https://doi.org/10.1007/3-540-46766-1_34

[12] Lindell, Y.: Secure multiparty computation. Commun. ACM 64(1), 86–96 (2021), https://doi.org/10.1145/3387108

[13] Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979), http://doi.acm.org/10.1145/359168.359176

[14] Desmedt, Y.: Society and group oriented cryptography: A new concept. In: Pomerance, C. (ed.) Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings. Lecture Notes in Computer Science, vol. 293, pp. 120–127. Springer (1987), https://doi.org/10.1007/3-540-48184-2_8

[15] Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceed-

ings. Lecture Notes in Computer Science, vol. 435, pp. 307–315. Springer (1989), https://doi.org/10.1007/0-387-34805-0_28

[16] Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust threshold DSS signatures. In: Maurer, U.M. (ed.) Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding. Lecture Notes in Computer Science, vol. 1070, pp. 354–371. Springer (1996), https://doi.org/10.1007/3-540-68339-9_31

[17] Gennaro, R., Goldfeder, S., Narayanan, A.: Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In: Manulis, M., Sadeghi, A., Schneider, S.A. (eds.) Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings. Lecture Notes in Computer Science, vol. 9696, pp. 156–174. Springer (2016), https://doi.org/10.1007/978-3-319-39555-5_9

[18] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding. Lecture Notes in Computer Science, vol. 1592, pp. 223–238. Springer (1999), https://doi.org/10.1007/3-540-48910-X_16

[19] Gennaro, R., Goldfeder, S.: Fast multiparty threshold ECDSA with fast trustless setup. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018. pp. 1179–1194. ACM (2018), https://doi.org/10.1145/3243734.3243859

[20] Lindell, Y., Nof, A.: Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018. pp. 1837–1854. ACM (2018), https://doi.org/10.1145/3243734.3243788

[21] Ming Shang, Yuan Ma, J.L., Jing, J.: A threshold scheme for sm2 elliptic curve cryptographic algorithm. pp. 155–166 (2014)

[22] Zhang, Y., He, D., Zhang, M., Choo, K.R.: A provable-secure and practical two-party distributed signing protocol for SM2 signature algorithm. Frontiers Comput. Sci. 14(3), 143803 (2020), https://doi.org/10.1007/s11704-018-8106-9

[23] Binbin Tu, X.W., Zhang, L.: Two distributed applications of sm2 and sm9. pp. 826–838 (2020)

[24] Aumayr, L., Ersoy, O., Erwig, A., Faust, S., Hostáková, K., Maffei, M., Moreno-Sanchez, P., Riahi, S.: Generalized bitcoin-compatible channels. IACR Cryptol. ePrint Arch. p. 476 (2020), https://eprint.iacr.org/2020/476

[25] Poelstra, A.: Lightning in scriptless scripts. mimblewimble team mailing list https://lists.launchpad.net/mimblewimble/msg00086.html

[26] Zhang, Z., Yang, K., Zhang, J., Chen, C.: Security of the SM2 signature scheme against generalized key substitution attacks. In: Chen, L., Matsuo, S. (eds.) Security Standardisation Research - Second International Conference, SSR 2015, Tokyo, Japan, December 15-16, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9497, pp. 140–153. Springer (2015), https://doi.org/10.1007/978-3-319-27152-1_7

[27] 国家密码管理局: Gm/t 0003-2012 sm2 椭圆曲线公钥密码算法. http://www.gmbz.org.cn/main/viewfile/20180108015515787986.html

[28] Kolesnikov, V., Mohassel, P., Rosulek, M.: Flexor: Flexible garbling for XOR gates that beats free-xor. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II. Lecture Notes in Computer Science, vol. 8617, pp. 440–457. Springer (2014), https://doi.org/10.1007/978-3-662-44381-1_25

[29] Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2729, pp. 145–161. Springer (2003), https://doi.org/10.1007/978-3-540-45146-4_9

[30] Schneider, T., Zohner, M.: GMW vs. yao? efficient secure two-party computation with low depth circuits. In: Sadeghi, A. (ed.) Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7859, pp. 275–292. Springer (2013), https://doi.org/10.1007/978-3-642-39884-1_23

[31] Esgin, M.F., Ersoy, O., Erkin, Z.: Post-quantum adaptor signatures and payment channel networks. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12309, pp. 378–397. Springer (2020), https://doi.org/10.1007/978-3-030-59013-0_19

[32] Lindell, Y.: Fast secure two-party ECDSA signing. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10402, pp. 613–644. Springer (2017), https://doi.org/10.1007/978-3-319-63715-0_21

[33] Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Two-party ECDSA from hash proof systems and efficient instantiations. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11694, pp. 191–221. Springer (2019), https://doi.org/10.1007/978-3-030-26954-8_7

[34] Doerner, J., Kondi, Y., Lee, E., Shelat, A.: Threshold ECDSA from ECDSA assumptions: The multiparty case. In: 2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019. pp. 1051–1066. IEEE (2019), https://doi.org/10.1109/SP.2019.00024

[35] Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings. Lecture Notes in Computer

Science, vol. 576, pp. 129–140. Springer (1991), https://doi.org/10.1007/3-540-46766-1_9

[36] An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: Knudsen, L.R. (ed.) Advances in Cryptology - EUROCRYPT 2002,. Lecture Notes in Computer Science, vol. 2332, pp. 83–107. Springer (2002)

[37] El Aimani, L.: Generic constructions for verifiable signcryption. In: Information Security and Cryptology - ICISC 2011. pp. 204–218. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)