

分类号 _____
U D C _____

密 级 _____
编 号 10486

武汉大学

硕 士 学 位 论 文

门限 SM2 签名方案研究

研 究 生 姓 名: 梁慧强

学 号: 2020202010068

指导教师姓名、职称: 陈建华 教授

专 业 名 称: 应用数学

研 究 方 向: 密码学

二〇二三年五月

The Research of Threshold SM2 Signature Scheme

Candidate: LIANG HUIQIANG

Student Number: 2020202010068

Supervisor: PROF. CHEN JIANHUA

Major: Applied Mathematics

Speciality: Cryptography




School of Mathematics and Statistics

WUHAN UNIVERSITY

May, 2023

论 文 原 创 性 声 明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的研究成果。除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

学位论文作者 (签名): 

2023 年 5 月 18 日

摘 要

公钥密码学有效的解决了密钥分配和身份认证两个问题, 由此对应了两种基本的密码协议: 密钥协商和数字签名. 数字签名能够进行鉴别身份、判断文件真伪、防止网络传输消息的伪造、抵赖和篡改, 具有和手写签名一样的法律效力. 随着互联网的不断发展, 多阶段深层次的密钥管理和身份认证成为信息化社会的研究重点. 而现有的签名方案无法在分布式设备上的部署, 另外私钥的安全性和访问结构的多样化也促进了门限签名的研究和发展.

门限签名是一种特殊的签名方案, 在 n 个参与方中分享私钥, 对于任意包含 t 个参与方的子集都能构建合法签名, 但是少于 t 个参与方则不能得到任何信息. 门限签名能有效的对私钥进行保护, 同时具有灵活的访问结构. 考虑到门限 ECDSA 签名和门限 Schnorr 签名的进展, 作为我国数字签名标准的 SM2 签名的门限形式仍局限于两方或者基于诚实多数的安全假设, 没有有效的解决方案.

为了使 SM2 签名适用于灵活的实际场景, 推动 SM2 签名在区块链、隐私计算中的应用. 本文基于半同态加密和零知识证明, 设计了一个具有非交互式的门限 SM2 签名方案. 方案的非交互性体现在输入消息后只需要一轮通信即可得到签名. 同时, 预签名阶段只需要 2 轮通信. 此外, 在恶意多数模型下达到可识别身份中止, 即如果签名过程失败, 可以找到导致失败的参与方. 兼容标准的 SM2 签名, 允许任意门限 $t \leq n$, 并且增加了密钥更新策略. 性能分析显示该方案预签名阶段的计算量和通信量与参与方数量呈线性增长关系, 理论上是同类门限 ECDSA 签名的计算量的 $1/3$.

关键词: 门限签名, SM2 算法, Ideal/Real 模型, 可身份识别中止

ABSTRACT

Public-key cryptography effectively solves the two problems of key distribution and identity authentication, and thus corresponds to two basic cryptographic protocols: key exchange and digital signature. Digital signature can identify identity, determine the authenticity of document, and the prevention of forgery, denial, and tampering of network transmission message, which have the same legal effect as handwritten signature. With the continuous development of the Internet, multi-stage key management and identity authentication have become the research focus of the information society. The existing signature scheme cannot be deployed on distributed devices, and the security of private key and the diversification of access structure also promote the research and development of threshold signature.

Threshold signature is a special signature scheme, in which private key are shared among n parties, and a legal signature can be constructed for any subset containing t parties, but less than t parties then no information can be obtained. The threshold signature can effectively protect the private key and has a flexible access structure. Considering the progress of the threshold ECDSA signature and threshold Schnorr signature, the threshold form of SM2 signature, which is the digital signature standard in China, is still limited to two-party or security assumption based on an honest majority, there is no efficient solution.

To make SM2 signature suitable for flexible practical scenario, and promote the application of SM2 signature in blockchain and privacy computing. Based on additive homomorphic encryption and zero-knowledge proof, this paper designs a non-interactive threshold SM2 signature scheme. The scheme's non-interactive nature is reflected in that only one round of communication is required to obtain the signature after the message input. At the same time, the pre-signing step only needs 2 rounds of communication. In addition, it is security with identity abort under the malicious majority

adversary model, that is, if the signature process fails, the party that caused the failure can be found. Compatible with the plain SM2 signature, allowing arbitrary threshold $t \leq n$, and adding a key refresh strategy. Performance analysis shows the amount of computation and communication in the pre-signing step of the scheme increases linearly with the number of participants, which is theoretically $1/3$ of the computation amount of the same threshold ECDSA signature.

Key words: Threshold signature, SM2 algorithm, Ideal/Real model, Identifiable abort

目 录

摘要	I
ABSTRACT	II
1 绪论	1
1.1 研究背景及意义	1
1.2 国内外研究现状	2
1.3 论文主要工作	5
1.4 论文的结构	6
2 基础知识	7
2.1 符号定义	7
2.2 椭圆曲线	7
2.3 哈希函数	9
2.4 SM2 数字签名	10
2.5 Paillier 加密	11
2.6 零知识证明	12
2.7 理想 (Ideal)/现实 (Real) 模型	13
3 可识别身份中止的门限 SM2 签名	16
3.1 问题引述	16
3.2 调整的 SM2 签名算法	16
3.3 理想门限签名函数定义	17
3.4 门限 SM2 签名的基本结构	18
3.5 门限 SM2 签名主协议	19
3.6 安全性分析	26
3.7 性能分析	34
4 总结与展望	36
参考文献	37

5 附录	42
致谢	47

插图

2.1 实数域上的椭圆曲线加法示例	8
3.1 SM2 门限签名结构	19
3.2 门限 SM2 签名主协议	20

表格

2.1	几种签名算法	11
3.1	调整后的 SM2 公私钥对	16
3.2	协议的总计算量和通信量	35
3.3	Canetti 等人 [1] 方案预签名阶段的计算量和通信量	35
3.4	实际运行时间	35
5.1	零知识证明的计算和通信花销	46

符号对照表

符号	符号名称
κ, λ	安全参数
p, q	大素数
F_q	阶为 q 的有限域
$\phi(\cdot)$	欧拉函数, $\phi(n)$ 代表小于或者等于 n 的正整数中与 n 互质的数的个数
sk	用户的私钥
pk	用户的公钥
G	椭圆曲线点群的生成元
q	生成元 G 的阶
$H(\cdot)$	哈希函数
ID	用户的身份
$ENTL$	用户身份 ID 的长度
Z	SM2 签名的附加信息
m, msg	待签名或者待加密的消息
t	门限值
$[\cdot]$	加法共享符号, 满足 $\sum[x] = \sum x_i = x$
$ENC(\cdot), DEC(\cdot)$	Paillier 加密, 解密操作
$E(\cdot), D(\cdot)$	半同态加密, 解密操作
aux	广播消息中的辅助信息
Rel	一种 NP 关系
sid, tig	广播消息中的会话标识符
ZK_{Rel}	用于证明 Rel 关系的零知识证明操作

1 绪论

1.1 研究背景及意义

公钥密码学的思想源远流长, 从 Shannon 在 1949 年将信息论 [2] 引入到密码学中, 公钥加密就出现在一些文字材料中但没有被系统的阐述和发展. 直到 Merkle 为了解决传统密码学在加密消息时需要以秘密的形式传输密钥的问题, 在 1974 年 [3] 提出公钥交换的概念, 无需使用安全通道, 在公开信道上选择出共同的密钥用于后续通信, 极大地便利了安全密钥分配问题, 理论上能抵抗任意的窃听敌手. 随后 Diffie 和 Hellman 在 1976 年 [4] 首次发表公钥加密系统, 主要包括两个方面, 基于离散对数难题的密钥交换 (Diffie-Hellman-Merkle Key Exchange) 方法和基于单向函数的数字签名, 基于离散对数难题的密钥交换方法受到了业界的广泛关注, 由数学难题的困难性, 使得公钥加密系统有了坚实的理论基础, 同时, Rivest, Shamir 和 Adleman 在 1977 年 [5] 基于大整数因数分解困难问题, 提出了 RSA 算法用于公钥加密和数字签名, RSA 算法的安全性由 RSA 问题的困难性保证, 也是现在广为流传的公钥算法之一. 由此, 公钥密码学得到了奠基和广泛应用, 也有效的解决了两个问题: 一个是密钥分配、另一个是身份认证. 分别对应两种基本密码协议: 密钥协商 (密钥交换) 和数字签名. 密钥协商在公开信道上建立共享密钥, 随后可以用此密钥传输对称加密密钥进行后续通信. 数字签名用于身份鉴别、文件真伪、防止网络传输消息的伪造、抵赖和篡改.

数字签名作为公钥密码学的重要组成部分, 伴随着公钥密码学一同出现, 也是身份认证的主要方式. 在数字签名系统中, 由发送方和接收方组成, 发送方有自己的私钥, 接收方有发送方的公钥. 发送方用自己的私钥对消息进行签名生成签名信息, 接收方使用发送方的公钥对其消息的签名信息进行正确性验证. 一旦验证通过, 就代表消息具有完整性, 且发送方不能否认所签名的消息. 数字签名具有和手写签名一样的法律效力, 已经广泛服务于人们的生产生活中, 当前普遍认可的数字签名有椭圆曲线数字签名标准 (ECDSA)、EdDSA、BLS、Schnorr 签名、RSA 签名以及我国数字签名标准 (SM2) 等等. 常在电子邮件、电子合同、远程金融交易以及区块链中使用.

随着互联网和计算机制造业的不断发展, 电子设备也由大到小、由少变多、计算能力弱到强、由中心式到分散协作. 物联网、大数据、移动计算和云计算等工作模式也被提出和应用. 社会的信息化程度逐步提高, 人们的生活方式发生了很大的改变. 海量的数据被收集、存储、传播、并且在本地或者云上进行加工处理, 为整个社会的发展提供了极大的便利. 新的概念和新的模式不断的出现和发展, 公钥密码学也展示出很强的适应性, 多阶段深层次的密钥管理和身份认证也成为信息化社会的重要领域和安全屏障, 延伸出了许多具有独特形式的应用, 如隐私计算、安全多方计算和

区块链技术等等,不断的为社会发展增添新动力.例如,从中本聪在 2009 年提出比特币以来,区块链技术和其他加密货币得到了快速发展,在加密货币中,交易信息主要由资金拥有者的数字签名和下一个资金拥有者的公钥组成.但是对于用户而言,私钥是唯一凭证,如果用户私钥遗失或被盗取将产生不可逆的经济损失,比特币首先采用的是多重签名来解决这个问题,但多重签名的灵活性有限,不支持任意的和复杂的访问结构.门限签名具有比多重签名更通用的性质,无需计算签名的数量,即可得到共享的签名用于验证身份.在比特币中,将有效的减少交易费用和时间,这也催生了业界对门限签名的研究和发展,不同数字签名的门限形式也逐步从理论走向实践.

同时,门限签名由 Desmedt 和 Frankel [6,7] 引入,允许多个参与方有共享数字签名的能力,在 n 方参与者中分享私钥 x ,对于任意的包含 t 方的子集都能构建以私钥 x 的合法签名,但是少于 t 方不能得到任何信息.一方面,作为群体身份认证的重要手段,门限签名将单一用户的数字签名推广到多方,同一个群体或者组织可以使用门限数字签名作为群体的标准数字签名用于外界验证,为深层次的身份认证提供了理论基础和实践经验,另一方面,门限签名允许多个参与方持有独自的密钥碎片,具有分布式属性,使用不同的设备存储密钥碎片,即使一部分设备不能使用或者丢失也不影响整体密钥的安全性,使得个人密钥管理成为可能.

作为我国的数字签名标准,SM2 数字签名在 2010 年提出,2016 年作为标准发布 (GB/T 32918.2-2016) [8],2017 年在德国柏林 ISO/IEC 信息安全分技术委员会 (SC27) 会议中入选为国际标准 (ISO/IEC 14888-3:2018),后由 [9] 在 2018 年发布. SM2 数字签名基于椭圆曲线密码学 (Elliptic Curve Cryptography, ECC),其安全性建立在椭圆曲线离散对数难题 (Elliptic Curve Discrete Logarithm Problem, ECDLP),与 RSA 相比,使用较小的密钥即可获得相同的安全强度,并且与 ECDSA 有相当的计算速度和安全性.面对区块链技术、安全多方计算、隐私计算的不断发展,推广 SM2 数字签名的门限形式使其适应多方计算和群体认证,有助于我国商用密码产品在金融交易和医疗领域中的应用,并且作为我国第一个发布的公钥密码算法标准,SM2 签名的推广对我国商用密码产品和国家信息安全体系有建设性作用.

1.2 国内外研究现状

门限签名作为密钥原语在许多场景中都有广泛应用,由标准数字签名 (RSA、ElGamal、Schnorr、ECDSA、SM2 等等) 推广而来,能有效的保护私钥和提供多方认证.例如在 Schnorr 签名中,公私钥对为 $(X = xG, x)$,消息为 m ,签名选择的随机数是 k ,其签名由两部分组成 (r, s) ,其中 $c = H(X, r, m)$, $r = kG$, $s = k + cx$. 由于 (r, s) 的线性性质,所以当 n 个参与方分别持有私钥 x_1, x_2, \dots, x_n ,通过密钥聚合得到主公钥 $X = x_1G + x_2G + \dots + x_nG$,然后分别选择随机数 k_1, k_2, \dots, k_n ,通过聚合得到共同随机数 $r = k_1G + k_2G + \dots + k_nG$,然后每一方对消息 m 计算出 $(r, s_i = k_i + H(X, r, m)x_i)$,将 s_i 线性相加即得到关于公钥 X 和消息 m 的 n 方多重签名.另外,通过 Shamir 秘密共享 (Secret Sharing) 或者其他形式的秘密共享技术可以将 (n, n) 多重签名变为

(t, n) 的门限签名.

但是这种方法会受到流氓密钥攻击 (Vogue-key Attack), 即存在恶意方 P_j 选择特定的公钥 Y , 当收到私钥碎片 $x_1G, \dots, x_{j-1}G, x_{j+1}G, \dots, x_nG$, 然后通过计算 $X_j = Y - x_1G - \dots - x_{j-1}G - x_{j+1}G - \dots - x_nG$ 广播将主公钥替换为 Y , 干扰合法签名的产生. 为了抵抗流氓密钥攻击, Bellare 和 Neven [10] 提出依赖签名者的挑战 $c_i = a_i \cdot H_{sig}(\hat{X}, r, m)$ 计算部分签名 $s_i = k_i + c_i x_i$, 其中 $a_i = H_{agg}(L, X_i)$, \hat{X} 是聚合公钥, $\hat{X} = \sum_{i=1}^n a_i X_i$, $L = \{X_1, \dots, X_n\}$. 此时的验证方程为 $sG = r + \sum_{i=1}^n c_i X_i = r + H_{sig}(\hat{X}, r, m) \hat{X}$ 仍与标准的 Schnorr 签名一致.

在签名系统中, 除了要避免对公钥 X 的攻击, 也需要对随机数 r 进行保护. 在聚合随机数过程中, 也会受到相似的流氓随机数攻击. 通常来说, 至少需要一轮广播用于聚合随机数和公钥, 然后一轮广播签名碎片 s_i . Bellare 和 Neven [10] 通过增加一轮广播使用承诺函数以避免流氓随机数攻击. 同时, 两轮的 Schnorr 多重签名会到并发攻击, Drijvers 等人 [11] 提出在并发情况下, 可以调用 Wagner 算法解决广义生日问题 (Generalized Birthday Problem) [12] 用于伪造签名, 由于 Wagner 算法的亚指数速度使得 $p \approx 2^{256}$ 的椭圆曲线在 $k_{max} = 128$ 的查询下降低到 2^{39} 次操作, 使得绝大多数两轮的 Schnorr 多重签名都不再安全.

为了避免并发攻击且达到两轮的最优通信轮数, Nick 等人在 [13] 提出 Musig-DN, 使用零知识证明避免并发攻击. 但是零知识证明的代价高昂, [14] 中提出 MuSig2, 平衡通信轮数和计算量, 达到通信轮数两轮且不需要零知识证明的并发安全, 其关键在于设置全局相关的 r 使得 Wagner 算法攻击不可行. 另外 Alper 和 Burdges [15] 利用随机数线性组合提出两轮通信的 Schnorr 多重签名. Komlo 和 Goldberg [16] 提出一种门限 Schnorr 签名称为 FROST, 将随机数 k_i 设置为 $d_i + e_i \cdot \rho_i$, 其中 d_i, e_i 是随机选取, $\rho_i = H(l, m, \{i, D_i, E_i\}_{i \in S})$, $D_i = d_i G$, $E_i = e_i G$, 这样在预计算中提前广播 D_i, E_i 达到承诺的作用, 有效的阻止流氓随机数攻击, 且具有并发安全.

Schnorr 签名以简洁方便的批量认证, 密钥聚合和门限形式, 已经集成到比特币中 (BIP340), 并且广泛应用在各个领域. 与 Schnorr 签名不同的是, SM2 和 ECDSA 一直抵制有效的门限签名, 因为其签名 (r, s) 分别为 $(x_1, y_1) = kG, r = H(m) + x_1, s = (1 + x)^{-1}(k - rx)$ 和 $r = kG, s = k^{-1}(H(m) + xr)$. 在计算 s 时, 求逆的运算破坏了 SM2 和 ECDSA 的线性结构, 不能进行简单线性组合, 需要有合适的技术将其逆运算转换为线性结构, 然后进行门限签名.

为了解决这个问题, Mackenzi 和 Reiter [17] 使用加同态加密 (Additive Homomorphic Encryption, AHE) 在诚实多数假设下生成两方签名. 然后, Gennaro 等人 [18] 再次使用 AHE 技术提出多方门限 ECDSA. 随后, Lindell [19] 优化了两方 ECDSA 的计算量, 只需要执行少量的 AHE 运算. 另外, Doerner 等人 [20] 将不经意传输 (Oblivious Transfer, OT) 引入到两方 ECDSA. 相比于两方门限签名, 尤其是不具备线性结构的 ECDSA 和 SM2, 还会有比其门限签名更简洁的形式, 王婧等人 [21] 提出用 Beaver 三元组 (Beaver Triple) 设计两方协同 ECDSA, 通过预先计算 Beaver 三元组, 避免繁重的 AHE 运算. 但是以上的方法都难以推广到多方门限签名, 仍然要对逆运算进行处理.

Gennaro 和 Goldfeder 在 [22] 中将门限签名扩展到 n 方到 n 方. 他们借鉴了安全多方计算中的 SPDZ 方法 [23], 主要思想如下: 首先是将两个数的乘积变为两个数的和 (加法共享), 这里使用了 Paillier 加密, 两个参与方 Alice 和 Bob 分别持有 a 和 b , Alice 计算 $c_A = E(a)$ 发送给 Bob. Bob 随机选择 β , 计算 $c_B = b \cdot E(a) + E(-\beta)$ 发送给 Alice. Alice 解密 $\alpha = D(c_B)$, 这时 $\alpha + \beta = ab$ 且 Alice 和 Bob 分别持有 α 和 β . 因为计算加法共享 ab 而没有揭露秘密信息 a 和 b , 这时 n 个参与方分别持有私钥 x_1, x_2, \dots, x_n 和随机数 k_1, k_2, \dots, k_n , 那么

$$\begin{aligned} & (x_1 + x_2 + \dots + x_n)(k_1 + k_2 + \dots + k_n) \\ &= x_1 k_1 + x_1 k_2 + \dots + x_1 k_n + \dots + x_n k_1 + x_n k_2 + \dots + x_n k_n \\ &= x_1 k_1 + \alpha_{1,2} + \beta_{1,2} + \dots + \alpha_{1,n} + \beta_{1,n} + \dots + \alpha_{n,1} + \beta_{n,1} + \alpha_{n,2} + \beta_{n,2} + \dots + x_n k_n \\ &= x_1 k_1 + \sum_{j=1, j \neq 1}^n (\alpha_{1,j} + \beta_{j,1}) + x_2 k_2 + \sum_{j=1, j \neq 2}^n (\alpha_{2,j} + \beta_{j,2}) + \dots + x_n k_n + \sum_{j=1, j \neq n}^n (\alpha_{n,j} + \beta_{j,n}) \end{aligned}$$

其中 $w_i = x_i k_i + \sum_{j=i, j \neq i}^n (\alpha_{i,j} + \beta_{j,i})$ 为第 i 个参与方独有, $\sum w_i = \sum x_i \cdot \sum k_i$ 这样我们就将乘积变为了线性结构.

因此 Gennaro 和 Goldfeder [22] 将 ECDSA 的签名变为 $r = k^{-1}G, s = H(m)k + rxk$, 在 r 的计算中, 随机选取 k_i, γ_i , 然后计算出 $\gamma_i G$ 广播得到 $\Gamma = \sum \gamma_i G$, 计算 $r_i = u_i \Gamma$ 其中 $\sum u_i = \sum k_i \cdot \sum \gamma_i$, 这时 $\sum r_i = k^{-1}G = r$. 然后计算 w_i 满足 $\sum u_i = \sum x_i \cdot \sum k_i$, 这时 $s_i = H(m)k_i + r w_i, s = \sum s_i$. 同时使用零知识证明和非延展模糊承诺 (Non-Malleable Equivocal Commitments) 抵抗恶意敌手. 同时 Lindell 等人 [24] 也用了相同的思想推广门限 ECDSA.

集中在加法共享技术, 使用 AHE 加密、不经意传输、Shamir 秘密共享 (Shamir Secret Share, SSS)、Beaver 三元组和混淆电路 (Garbled Circuit, GC) 等都可以有效的实现加法共享. 随后, Pettit 等人 [25] 使用 Shamir 秘密共享来实现快速易于实施的多方情况, 但是基于多项式的 Shamir 秘密共享处理乘法变加法时, n 阶多项式变为了 $2n$ 阶多项式, 所以执行 n 方签名时至少需要 $2n$ 个参与方, 并且需要基于诚实多数的假设. Canetti 等人 [1] 基于恶意多数的假设将门限 ECDSA 推广到通用可组合框架下 (Universally Composable Framework, UC Framework) 安全, 可识别身份中止的安全目标, 这项工作基于 Gennaro 等人 [22] 和 Lindell 等人 [24]. 使用了 Paillier 加密完成加法共享, 零知识证明来识别恶意参与方, 并且设置预签名阶段使得参与方确认待签名消息后只需要一轮广播即可完成签名, 定期密钥刷新, 能承担适应性恶意敌手, 具有良好的通信能力和健壮性, 并在通用组模型 (Generic Group Model, GGM) 和随机预言机 (Random Oracle, RO) 下证明了预签名步骤的安全性.

对于门限签名来说, 不仅仅是完成其门限形式, 还需要抵抗各种各样的攻击, 正如前面所述的 Schnorr 门限签名, 需要避免对流氓密钥攻击, 例如采用承诺函数抵抗流氓密钥攻击, 尽管承诺函数的形式会有所不同, 但应对方法是一致的: 迫使参与方在参与计算前, 做出对应承诺, 在参与计算后验证承诺, 避免恶意参与方根据接受到的信息重新选取相应值, 以达到抵抗恶意参与方的目的. 另

一方面, 对于恶意参与方, 在特定的计算步骤和参数选取, 还要使用零知识证明, 检测参与方是否按照协议规范执行.

门限 ECDSA 由理论变为实践, 又促进了其相关理论的发展. Groth 和 Shoup [26] 对 ECDSA 的加法密钥派生 (Additive Key Derivation) 和预签名 (Presignatures) 进行了分析, 表明在使用广义生日攻击的思想 [12] 下, 结合加法密钥派生和预签名的 ECDSA 存在比标准 ECDSA 平方根攻击要快的立方根攻击. 另外给出了重随机预签名 (Re-randomized Presignatures) 和同态密钥派生 (Homogeneous Key Derivation) 两种缓解措施. 同时, Abram 等人 [27] 提出一个通信成本低的方案, 但使用伪随机相关生成器提出了一个不健壮 (需要可信的种子分发中心). Harry 等人 [28] 提出了高效通信的方案, 用更低的计算成本去识别恶意方但牺牲了通信轮数. Kondi 等人 [29], Groth 和 Shoup [30] 等人也从不同角度提出相应的门限 ECDSA 签名方案.

与门限 ECDSA 的突破相比, Shang 等人 [31] 基于秘密共享提出了诚实多数的门限 SM2 方案. Zhang 等人 [32] 提出了基于 Paillier 的两方门限 SM2 方案, 并使用理想零知识证明函数防止恶意方. Liu 等人 [33] 基于双线性对, 提出了基于身份的多方签名. 林等人 [34] 设计了一个两方协同 SM2 数字签名. 以及其他一些学者基于身份, 无证书系统, 结合代理签名, 盲签名, 环签名的多方签名. 综上所述, 门限 Schnorr 签名和门限 ECDSA 签名都有了相应的解决方案, SM2 签名作为我国商用密码标准, 其各种变体和应用也受到了广泛的研究和讨论, 但是对于高效的门限 SM2 签名仍缺乏完整的理论分析.

1.3 论文主要工作

本文以门限签名为起点, 对各种密码原语进行介绍和分析, 针对门限 SM2 签名的方案的效率和安全性理论分析不足, 构建了多方门限 SM2 签名并给出安全性证明. 本文的主要工作如下:

(1) 针对门限 SM2 签名在分布式场景应用的局限性, 本文调整了标准 SM2 签名算法, 在不改变 SM2 的验证过程的情况下, 修改公私钥对以适应门限形式, 并与标准的 SM2 签名兼容.

(2) 针对门限 SM2 签名方案的计算效率和安全性, 本文提出一个非交互式的门限 SM2 签名, 将 SM2 签名步骤分为预签名阶段和签名阶段, 其中签名阶段只需要一轮通信以达到了非交互式, 预签名阶段只需要进行一次 n 方加法共享, 相比于 ECDSA 在预签名阶段的两次 n 方加法共享, 门限 SM2 签名的预签名阶段的总通信量和计算量相当于同级别 ECDSA 的 $1/3$. 并且基于恶意多数模型下证明该方案在任意门限下具有可识别身份中止的安全性.

(3) 针对门限 SM2 签名方案的不同阶段具有相互独立的特性, 本文对所有的子协议独立设计, 只要满足相应功能均可替换和修改对应模块. 同时重新调用子协议 MultiAdd step 和 VSS step 以增加密钥更新策略.

综上所述, 本文方案实现了去中心化、可检测恶意方、具有良好的计算性能, 为深层次的密钥管理和身份认证提供了一种适用方法.

1.4 论文的结构

全文可分为四个章节:

第一章: 绪论. 本章主要讲述了门限签名的研究背景及意义, 介绍了门限 Schnorr 签名, 门限 ECDSA 签名和门限 SM2 签名的研究现状, 概括了本文的主要工作.

第二章: 基础知识. 本章主要介绍了 SM2 门限签名涉及到的相关密码原语, 包括椭圆曲线基本知识, 哈希函数, SM2 数字签名, Paillier 加密, 零知识证明和基于模拟的 Ideal/Real 安全模型.

第三章: 可识别身份中止的门限 SM2 签名. 本章首先介绍了门限 SM2 签名目前存在的问题, 并提出一种调整方式使其适应门限形式. 其次给出了理想门限签名函数. 然后对本文的门限 SM2 签名方案的基本结构做概述, 给出 SM2 门限签名主协议和各个子协议. 最后在 Ideal/Real 模式下给出安全性证明, 和实际性能分析.

第四章: 总结与展望. 本章总结了论文的研究工作, 同时分析了研究方案的通用方法和未来的研究方向.

2 基础知识

2.1 符号定义

首先给出本文通用的符号定义,

定义 2.1.1 (多项式时间, PPT) 如果存在一个多项式 $p(\cdot)$, 对于任意输入 $x \in \{0,1\}^*$, 算法 F 在 $p(|x|)$ 步内终止, 则称算法 F 在多项式时间内运行.

定义 2.1.2 (NP 关系, Rel) 给定一个问题 P , 和一个解 ans , 存在一个算法可以在多项式时间内验证 $P(ans) == 1$, 但不一定存在一个算法可以在多项式时间内计算出 ans^* , 满足 $P(ans^*) == 1$.

定义 2.1.3 (可忽略函数) 如果对任意多项式 $p(\cdot)$, 都存在整数 N 使得对所有 $n > N$ 都有 $f(n) < \frac{1}{p(n)}$, 那么称 f 是一个可忽略函数.

定义 2.1.4 (可忽略优势) 一个事件发生的范围是 n , 发生的概率 $\leq \frac{1}{n}$, 那么称这个事件具有可忽略优势.

定义 2.1.5 (显著优势) 如果一个事件发生的概率是不可忽略优势, 那么称这个事件具有显著优势.

2.2 椭圆曲线

自 1985 年 Koblitz [35] 和 Miller [36] 分别独立提出基于椭圆曲线的公钥密码学. 椭圆曲线就被视为 RSA 算法的继承者, 随着计算能力的提高和因子分解性能不断提高, RSA 需要更多位数的密钥提供安全保障, 椭圆曲线使用更短的密钥长度就可以获得相同的安全级别. 下面给出有限域和椭圆曲线定义.

定义 2.2.1 (有限域) 如果一个域 F 只有 q 有限个元素, 那么我们称之为有限域 F_q 或者伽罗瓦域 (Galois Field) $GF(q)$, 称有限域 F_q 的阶为 q .

定理 2.2.1 (有限域的阶) 有限域 F_q 的阶 $q = p^n$, 其中 p 是素数是 F_q 的特征, n 是 F_q 在其素子域的次数.

本文只考虑两种类型的有限域, q 为一个奇素数的素域 F_p , 其元素可以用整数 $\{0, 1, 2, \dots, p-1\}$ 表示, $q = 2^n$ 的二元扩域 F_{2^n} , 其元素可以用长度为 n 的比特串表示 [37].

定义 2.2.2 (素域上的椭圆曲线) 在特征不为 2 或 3 的域 F_p 上的椭圆曲线 $E(F_p)$ 是一些点集 $\{O\} \cup (x, y) \in F_p \times F_p$, 其中 O 是无穷远点, 满足等式:

$$y^2 = x^3 + ax + b, a, b \in F_p, (4a^3 + 27b^2) \bmod p \neq 0. \quad (2.1)$$

定义 2.2.3 (二元扩域上的椭圆曲线) 在二元扩域 F_{2^n} 上的椭圆曲线 $E(F_{2^n})$ 是一些点集 $\{O\} \cup (x, y) \in F_{2^n} \times F_{2^n}$, 其中 O 是无穷远点, 满足等式:

$$y^2 + xy = x^3 + ax^2 + b, a, b \in F_{2^n}, b \neq 0. \quad (2.2)$$

素域上的椭圆曲线的加法运算和实数域上的几何描述的代数结构类似. 通过将定义 2.2.2 上 a, b, x, y 限制在实数域 R 上, 即是实数域上短 weierstrass 形式的椭圆曲线, 其加法运算的几何描述为“切弦法”. 对于曲线上的两点 $P = (x_1, y_1)$ 和 $Q = (x_2, y_2)$ (如果 P 和 Q 重合, 取经过点 P 的切线与椭圆曲线相交于 $R' = (x_3, -y_3)$) 经过两点的直线与椭圆曲线相交于 $R' = (x_3, -y_3)$, 过点 R' 做 y 轴的平行线, 与椭圆曲线相交于 $R = (x_3, y_3)$, 规定 $P + Q = R$, 如果 P 和 Q 相加为无穷远点 O , 记为 $P + Q = O$. 二元扩域上的加法运算相似但不尽相同, 具体可以参看 [37].

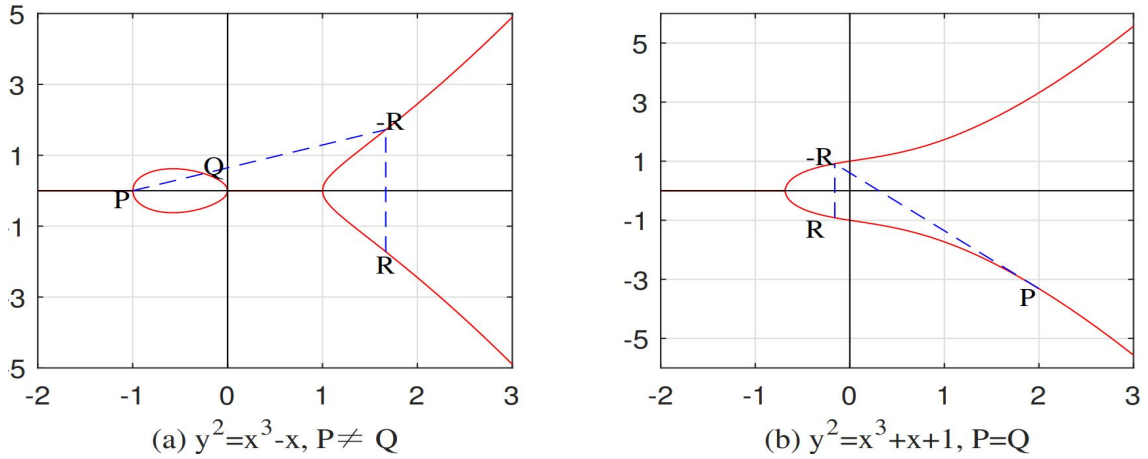


图 2.1: 实数域上的椭圆曲线加法示例

定义 2.2.4 (素域上的椭圆曲线加法法则) 满足定义 2.2.2 的椭圆曲线 $E(F_p)$ 的加法运算定义如下:

1. 单位元: 无穷远点 O , 对于任意的点 $P \in E(F_p)$ 都有 $P + O = O + P = P$.
2. 逆元: 对于任意点 $P = (x_1, y_1) \in E(F_p) \setminus \{O\}$, 其逆元为 $-P = (x_1, -y_1)$.
3. 加法运算: 对于任意的两个点 $P = (x_1, y_1), Q = (x_2, y_2) \in E(F_p)$, 则 $R = P + Q = (x_3, y_3)$:

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases} \quad (2.3)$$

其中

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, P \neq Q \\ \frac{3x_1^2 + a}{2y_1}, P = Q \end{cases} \quad (2.4)$$

素域上的椭圆曲线 $E(F_p)$ 中的所有点组成的集合在素域上的椭圆曲线加法法则下为一个阿贝尔群, 且记 $nP = \overbrace{P + P + \dots + P}^n$ 为椭圆曲线标量乘法.

这里我们给出一个实例 $y^2 = x^3 + x + 1 \pmod{7}$, 其中 $(4a^3 + 27b^2) \pmod{p} = 3 \neq 0 \pmod{p}$, 所以椭圆曲线的点为 $(0, 1), (0, 6), (2, 2), (2, 5)$ 和无穷远点 O , 且令 $P = (0, 1)$, 按照加法运算, $2P = (2, 5), 3P = (2, 2), 4P = (0, 6), 5P = O$. 常见的椭圆曲线有 Secp256k1, Sm2p256v1, Curve25519.

定义 2.2.5 (椭圆曲线的阶) 称有限域 F_p 上的椭圆曲线 $E(F_p)$ 中所有点包括无穷远点的个数为椭圆曲线的阶, 记为 $\#E(F_p) = q$.

定理 2.2.2 (Hasse 定理 [38]) 关于椭圆曲线 $E(F_p)$ 的阶 $\#E(F_p)$, 满足以下不等式:

$$|\#E(F_p) - (p + 1)| \leq 2\sqrt{p} \quad (2.5)$$

定义 2.2.6 (椭圆曲线离散对数问题, ECDLP) 在椭圆曲线 $E(F_q)$ 中, 给定点 P, Q , 要求计算出 $k \in F_q$, 满足 $Q = kP$.

椭圆曲线能够应用在公钥密码系统, 归功于两个性质, 一个是椭圆曲线的点加运算构成阿贝尔群, 另一个是椭圆曲线的标量乘法构成一个单向函数. 阿贝尔群使得可以运行密码系统, 标量乘法的单向函数提供安全保障.

椭圆曲线离散对数问题的难度和选定椭圆曲线的阶直接相关, Koblitz [35] 实验表明有限域上的椭圆曲线离散对数问题比经典的离散对数问题要难. 可以用于构建密码系统. 在求解椭圆曲线离散对数问题上, 有许多传统计算机上的经典算法, 例如: Shanks 的大步小步算法 (Baby Step Giant Step, BSGS), Pollard's Rho 算法, 目前最好算法仍需要全指数时间复杂度 [39]. 通常假设椭圆曲线离散对数问题是很难的, 即没有多项式时间内算法能够解决. 由此得到一个可用于公钥系统构建的单向函数.

2.3 哈希函数

哈希函数 (Hash Function) 是将任意长度的数据映射为固定长度字符串的函数, 哈希函数可以表示为 $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$. 满足以下性质:

1. 可计算性: 对于任意的输入 $m \in \{0, 1\}^*$, 都存在一个多项式时间算法计算出 $H(m) \in \{0, 1\}^n$.
2. 单向性: 对于给定的哈希值 $H(m)$, 没有多项式时间算法可以计算出 m .
3. 抗弱碰撞性: 给定一个消息 $m \in \{0, 1\}^*$ 的哈希值 $H(m)$, 不存在多项式时间算法找到 $m' \neq m$ 满足 $H(m') = H(m)$.

4. 抗强碰撞性: 不存在多项式时间算法找到两个消息 m', m , 使得 $H(m') = H(m)$, 且 $m' \neq m$.
其中性质 3 和性质 4 满足一个即可.

哈希函数可以将消息或者数据压缩为固定长度的摘要, 并且以可忽略的概率找到原像和出现哈希碰撞, 主要用于保障数据的完整性. 哈希函数的结构通常是迭代哈希函数, 或者是 MD 结构, 由 Merkle [40] 和 Damgard [41] 提出, 通过重复使用压缩函数以到达上述性质, 如果压缩函数具有抗碰撞性, 那么迭代哈希函数就具有抗碰撞性. 不考虑哈希函数的数学结构, 只考虑哈希值的范围 n 的长度, 因为生日悖论, 所以大约 \sqrt{n} 次选择后, 重复的概率就会超过 0.5. 但是当 n 足够长, 如 512 位时, 找到强碰撞的代价依然很高.

在本文中, 我们用 Hash 函数作为安全承诺函数, 即假设 Hash 函数是随机预言机 (Random Oracle), 随机预言机是一个密码学概念, 可以理解为一个理论的黑箱, 对于任意输入, 都输出一个真正均匀随机的输出, 有以下性质:

1. 可计算性: 任意的输入, 都存在一个多项式时间算法计算出预言值.
2. 一致性: 相同输入具有相同的输出.
3. 均匀分布性: 随机预言机的输出在值空间中是均匀分布的, 无碰撞.

哈希函数易满足随机预言机的性质 1 和性质 2. 如果一个方案在随机预言机模型下证明安全, 即这个方案中使用到哈希函数假设为随机预言机, 即敌手不利用哈希函数的弱点来攻击这个方案.

2.4 SM2 数字签名

数字签名是由一个签名者对消息 (数据) 生成数字签名, 并有验证者验证签名的合法性. 每一个签名者都有一对公私钥, 用私钥签名, 验证者用签名者的公钥验证. SM2 签名算法由国家密码管理局 [8] 于 2010 年发布, 其签名方案简要描述如下.

1. 密钥生成算法 $ID_A.SM2.KeyGen(\lambda)$: 对于给定的安全参数 λ , 根据 [8] 规定的 SM2 椭圆曲线, 选取椭圆曲线方程参数 $a, b, G = (G_x, G_y)$, SM3 哈希函数和随机数发生器. 随机选取一个整数 $sk = x$, 计算 $pk = X = xG = (x_A, y_A)$ 作为签名者 ID_A 的公私钥对. 计算 ID_A 的附加信息 $Z_A = H_{256}(ENTL_A || ID_A || a || b || x_G || y_G || x_A || y_A)$, 其中 $ENTL_A$ 是 ID_A 的长度信息.
2. 私钥签名算法 $ID_A.sk.SM2.Sign(m)$: 对于待签名消息 m . 随机选择 $k \in [1, q - 1]$ 并计算 $(x_1, y_1) = kG$, $e = H_v(Z_A || m)$. 计算 $r = (e + x_1) \bmod q$, 若 $r == 0, q - k$, 重新选取 k , $s = (1 + x)^{-1}(k - rx) \bmod q$, 若 $s == 0$, 否则重新选取 k . 输出签名 $\sigma = (r, s)$.
3. 公钥验证算法 $ID_A.pk.SM2.Verify(m', \sigma')$: 对于待验证的 $m', \sigma' = (r', s')$. 验证过程是若 $r', s' == 0$, 验证失败, 否则计算 $e' = H_v(Z_A || m')$, $t = (r' + s') \bmod q$. 若 $t == 0$ 验证失败, 否则 $(x'_1, y'_1) = s'G + tX$, $R = (e' + x'_1) \bmod q$, 检查 $R == r'$ 是否为真. 若为真, 验证通过, 输出 1, 否则验证失败, 输出 0.

表 2.1: 几种签名算法

Sym.	Schnorr	ECDSA	SM2
sk	x	x	x
pk	$X = xG$	$X = xG$	$X = xG$
R	$(x_1, y_1) = kG$	$(x_1, y_1) = kG$	$(x_1, y_1) = kG$
e	$H(m R X)$	$H(m)$	$H(H_{256}(X) m)$
r	$x_1 \bmod q$	$x_1 \bmod q$	$e + x_1 \bmod q$
s	$k + rx \bmod q$	$k^{-1}(e + xr) \bmod q$	$(1 + x)^{-1}(k - rx) \bmod q$

这里, 表2.1 给出了 Schnorr 签名, ECDSA 和 SM2 签名. 其中 q 是生成元 G 的阶, $H : \{0, 1\}^* \rightarrow \{0, 1\}^q$ 代表安全的哈希函数, sk 是签名者私钥, pk 是签名者公钥, $k \in [1, q - 1]$ 是签名者选取的随机数, 签名为 (r, s) .

签名方案应满足存在不可伪造性, 即敌手无法伪造任何消息的签名. 通常使用敌手 (Adversary) 和挑战者 (Challenger) 之间的游戏 (Game) 来定义的. Game 游戏的描述如下:

初始化: 挑战者执行密钥生成算法, 生成密钥对 (sk, pk) 并发送 pk 给敌手, 保留 sk .

查询: 敌手选取消息 m_i 进行签名查询, 挑战者回复消息 m_i 的签名 σ_i , 并记录.

伪造: 敌手输出一个消息 m^* 的签名 σ_{m^*} , 如果满足:

1. σ_{m^*} 是消息 m^* 的合法签名.
2. 消息 m^* 的签名没有被查询过.

则敌手在游戏中获胜, 敌手赢得游戏的优势 ϵ 就是伪造一个合法签名的概率.

2.5 Paillier 加密

Paillier 概率加密算法由 Paillier [42] 在 1999 年提出, 能够支持加同态运算. 本文中的 Paillier 加密定义如下:

1. 密钥生成算法 $\text{Paillier.KeyGen}(\kappa)$: 在安全参数 κ 为输入下, 生成两个长度为 $\kappa/2$ 的素数 $sk = (p, q)$ 作为私钥, 设置 $pk = N = pq$ 作为公钥. 输出 (N, p, q) .
2. 公钥加密算法 $\text{pk.Paillier.Enc}(m)$: 输入消息 $m \in Z_N$, 随机选择 $r \in Z_N^*$. 输出 c , 这里

$$c = \text{ENC}_{pk}(m, r) = (1 + N)^m r^N \bmod N^2 \quad (2.6)$$

3. 私钥解密算法 $\text{sk.Paillier.Dec}(c)$: 输入密文 $c \in Z_{N^2}$, 计算 $\mu = \phi(N)^{-1} \bmod N$. 输出 M' , 这里

$$m' = \text{DEC}_{sk}(c) = \left(\frac{c^{\phi(N)} \bmod N^2 - 1}{N} \right) \mu \bmod N \quad (2.7)$$

4. 加同态性质: 给定两个密文 $c_1 = ENC_{pk}(m_1) \in Z_{N^2}, c_2 = ENC_{pk}(m_2) \in Z_{N^2}, k \in Z_N$, 满足

$$DEC_{sk}(c_1 \times c_2 \bmod N^2) = m_1 + m_2 \bmod N \quad (2.8)$$

$$DEC_{sk}(k \cdot c_1 \bmod N^2) = km_1 \bmod N \quad (2.9)$$

定理 2.5.1 (Paillier [42]) 如果合数剩余问题是困难的, 那么 Paillier 加密方案是不可区分的。

不可区分性的 Game 游戏的描述如下:

初始化: 挑战者执行密钥生成算法, 生成密钥对 (sk, pk) 并发送 pk 给敌手, 保留 sk .

查询 1: 敌手任意访问 $pk.Enc(\cdot)$, 随机选取相同长度的 m_0, m_1 并将其发送给挑战者。

查询 2: 挑战者随机选取 $b \leftarrow \{0, 1\}$, 将密文 $c_b = pk.Enc(m_b)$ 发送给敌手, 敌手任意访问 $pk.Enc(\cdot)$.

判断: 敌手输出一个消息 b' , 如果满足: $b' == b$.

则敌手在游戏中获胜, 敌手赢得游戏的优势 ϵ 就是可区分的概率。

2.6 零知识证明

零知识证明是一个交互式证明系统 [43], 由两部分组成, 证明者和验证者。证明者在不透露有关证人 (witness) 的任何信息下, 说服验证者一个陈述 (statement) 是真的。除非有可忽略的错误概率, 验证者只有在这个陈述是真的情况下才会被说服。没有证人的证明者不能说服验证者。且验证者除了陈述是真的以外, 不能学到关于证人的任何信息。陈述是一个 NP 关系。

定义 2.6.1 (零知识证明) $\Sigma\tau$ 协议方案是一个关系 Rel 的概率多项式时间算法, 其由元组 $(G_0, P_1, P_2, V_1, V_2)$ 组成, 其中 G_0 是生成算法, P_i 表示证明者的第 i 步, V_i 表示验证者的第 i 步。

- G_0 生成 τ 作为 (P_1, P_2, V_1, V_2) 的辅助输入。
- P_1 输入 $\kappa = |x|$ 和随机输入 k , 然后输出 R 。
- V_1 选择一个随机变量 e 作为输出。
- P_2 有一个证据 w , 然后输入 (x, w, k, e) 并输出 v 。
- V_2 输入 (x, R, e, v) , 输出一个硬币 $b \in \{0, 1\}$, 如果 $(x, w) \in Rel, b = 1$, 否则 $b = 0$ 。

安全性质:

1. 正确性 (Completeness): 如果 $(x, w) \in Rel$, 对于任意的 $e \leftarrow V_1$ 和 $R \leftarrow P_1(k; 1^\kappa)$, 以及 $v \leftarrow P_2(x, w, k, e)$, 都有显著优势使得 $V_2(x, R, e, v) == 1$ 。
2. 完备性 (Soundness): 如果任意的 $w, (x, w) \notin Rel$, 对于任意的 $P^*, e \leftarrow V_1$, 任意的 $R, v = P_2(x, w, k, e)$ 都有显著优势 $V_2(x, R, e, v) == 0$ 。

或者特殊完备性 (Special Soundness): 存在一个提取器 E^* , 对于任意的 $(\tau \leftarrow G_0), x$ 和 P^* : 如

果 $(R, e, v), (R, e', v') \leftarrow P^*(x, w, \tau)$, 其中 $V_2(x, R, e, v) == V_2(x, R, e', v') == 1$ 但是 $e \neq e'$, 那么就有显著优势使得 $w' \leftarrow E^*(x, R, e, e', v, v')$ 且 $(x, w') \in Rel$.

3. 诚实验证者零知识 (Honest Verifier Zero Knowledge, HVZK): 如果有一个模拟器 S^* , 其中 $(R, e, v) \leftarrow S^*(x)$ 其对于任意的 x 都有显著优势使得 $V_2(x, R, e, v) == 1$, 并且下面两个分布是统计不可区分的.

对于 $(x, w) \in Rel : (R, e, v)$, 其中 $e \leftarrow V_1, R \leftarrow P_1(x, w, k), v = P_2(x, w, k, e)$.

(R, e, v) , 其中 $e \leftarrow V_1, (R, v) \leftarrow S^*(x, e)$.

步骤 V_1 下, 验证者只输出了随机随机数 e , 假设哈希函数为随机预言机, 则可以将其替换为 $H(\cdot)$. 因此, Σ_τ 协议在假设随机预言机的情况下, 变为了非交互式零知识证明协议.

定义 2.6.2 (非交互式零知识证明) 随机预言机 $H : \{0, 1\}^* \rightarrow \{0, 1\}^h$, 辅助输入信息 aux .

- 输入 $(prove, P_1, P_2, aux, x; w)$;

随机选取 k , 计算 $R = P_1(k; 1^\kappa)$, $e = H(aux, x, R)$ 和 $v = P_2(x, w, k, e)$, 输出 $\psi = (R, e, v)$.

- 输入 $(verify, V_2, aux, x, \psi)$;

如果 $V_2(x, R, e', v) == 1$, 输出 1, 否则输出 0, 其中 $e' = H(aux, x, R)$.

2.7 理想 (Ideal)/现实 (Real) 模型

本文方案的安全目标是在恶意模型下可识别身份中止, 这里我们引用 Cohen 等人 [44] 的简要叙述, 更多细节可以查阅 Goldreich 的《Foundations of cryptography》[45]. 非正式的说, 如果一个恶意攻击者在现实世界中攻击协议不能获取比在理想世界中攻击协议更多的信息, 那么这个协议至少在恶意模型下与理想函数 F 一样安全.

定义 2.7.1 (n 方函数) n 方函数是在多项式时间内将 n 个输入向量映射到 n 个输出向量的随机过程. 给定一个 n 方函数 $f : (\{0, 1\}^*)^n \rightarrow (\{0, 1\}^*)^n$, $f_i(\mathbf{x})$ 代表第 i 位的输出, 即 $f_i(\mathbf{x}) = f(\mathbf{x})_i$, 如果每一位的输出值是相同的, 称为函数 f 的公共输出, 否则为私有输出.

在现实世界中, n 方协议 $\Pi = (P_1, \dots, P_n)$ 是一个 n 元组的概率多项式时间交互图灵机. P_i 输入 $x_i \in \{0, 1\}^*$ 和随机数 $r_i \in \{0, 1\}^*$, 输入长度为安全参数 κ . 敌手 A 是另一个交互式图灵机, 它以包含损害方的身份及其私有输入 (可能还有附加输入) 的输入开始执行. 双方在同步的网络中执行协议. 执行按轮 (round) 进行, 每轮由一个发送阶段 (各方从这轮发送信息) 和一个接收阶段 (各方从其他各方接收消息) 组成.

各轮可以在每一轮中通过广播频道或使用完全连接的点对点网络进行通信. 在认证信道, 敌手 A 能读取两个诚实方之间发送的消息, 但不能修改.

在协议执行的整个过程中所有诚实方都遵循规定的协议的指示, 而损害方则从敌手那里接受指示. 敌手被称为恶意的, 这意味着它可以指示损坏方以任意方式偏离协议. 在协议执行结束时,

诚实方输出协议规定的输出, 损坏方不输出任何内容, 敌手输出以损坏方视图为输入的任意视图函数. 协议执行中一方的视图由其输入, 随机比特 b 和执行过程中所接收到的信息组成.

定义 2.7.2 (现实视角) 设 $\Pi = (P_1, \dots, P_n)$ 是 n 方协议, $I \subseteq [n]$ 表示敌手 A 控制的损害方集合. 在 (A, I) 与 Π 在联合输入, 附加输入 z , 安全参数 κ , 和 $\mathbf{x} = (x_1, x_2, \dots, x_n)$ 下, $REAL_{\Pi, I, A(z)}(\mathbf{x}, \kappa)$ 定义为 Π 和 $A(z)$ 按上述过程的交互式信息, 包括敌手 A 为每一个损害方 P_i 计算的信息和 Π 每一个诚实方 P_j 计算的信息.

可识别身份中止的理想计算: n 方函数 f , 参与方 P_1, \dots, P_n 输入 $\mathbf{x} = (x_1, x_2, \dots, x_n)$, 并且存在理想敌手 A 控制损害方集合 $I \subseteq [n]$, 通过以下步骤进行.

向可信方发送输入: 诚实的参与方 P_i 将 x_i 的输入发送给可信方. 而对于被损害的参与方, 敌手 A 将其任意输入发送给可信方, 并将 x'_i 设为 P_i 的实际输入.

提前中止: 敌手 A 选择一个被损害的参与方 $i^* \in I$, 并向可信方发送消息 (abort, i^*) 以中止计算, 此时可信方将 (abort, i^*) 发送给所有参与方并中止计算.

可信方回复敌手 A : 可信方计算 $(y_1, \dots, y_n) = f(x'_1, \dots, x'_n)$ 并将 y_i 发送给每个被损害的参与方 P_i , 即 $i \in I$.

延迟中止: 敌手 A 选择一个被损害的参与方 $i^* \in I$, 并向可信方发送消息 (abort, i^*) 以中止计算, 此时可信方将 (abort, i^*) 发送给所有参与方并停止计算. 否则, 继续.

可信方回复诚实方: 可信方向每个参与方 P_i , 其中 $i \notin I$, 发送 y_i .

输出: 诚实方始终输出从可信方接收到的消息, 而损害方不输出任何内容. 敌手 A 输出一个由从可信方接收到的辅助输入和损害方的消息组成的任意函数值.

定义 2.7.3 (理想视角) N -方函数 $f : (\{0, 1\}^*)^n \rightarrow (\{0, 1\}^*)^n$, 并且 $I \subseteq [n]$ 表示被损害的参与方的集合. 在 (A, I) 与 f 的联合输入, 附加输入 z , 安全参数 κ 和 $\mathbf{x} = (x_1, x_2, \dots, x_n)$ 下, $IDEAL_{f, I, A(z)}^{id-\text{abort}}(\mathbf{x}, \kappa)$ 定义为 P_1, P_2, \dots, P_n 和 $A(z)$ 是从上述过程中得到的输出.

定义 2.7.4 (模拟安全) N -方函数 $f : (\{0, 1\}^*)^n \rightarrow (\{0, 1\}^*)^n$, Π 是一个实现 f 的概率多项式时间协议. 协议 $\Pi(\delta, l)$ 具有可识别身份中止的计算 f , 即如果对于每个概率多项式时间的现实模型对手 A , 存在一个 (期望的) 多项式时间理想模型对手 S , 使得在最多 l 个参与方的每个子集 $I \subseteq [n]$ 上都成立:

$$REAL_{\Pi, I, A(z)}(\mathbf{x}, \kappa) \equiv_c^\delta IDEAL_{\Pi, I, S(z)}^{id-\text{abort}}(\mathbf{x}, \kappa) \quad (2.10)$$

如果 δ 可忽略的, 那么我们就说在计算安全的假设下, Π 协议可以在 l -安全可识别身份中止的情况下计算 f .

定理 2.7.1 (次序组合定理 (Sequential Composition Theorem) [46]) 设 $l < n, m \in N, f_1, \dots, f_m, g$ 为 n 方函数. 我们称 Π 为 n 方协议在 (f_1, \dots, f_m) 混合模型下 l 安全的实现 g , 其中 n 方协议

ρ_1, \dots, ρ_m 分别 l 安全的实现 f_1, \dots, f_m , 并且每轮最多只调用一个函数. 那么 Π 就是在 ρ_1, \dots, ρ_m 下 n 方协议 l 安全地实现 g .

3 可识别身份中止的门限 SM2 签名

3.1 问题引述

SM2 的签名形式使得其分布式方案具有这种各样的局限性, 以往的方案或者基于两方, 不能直接推广到多方、或者基于多项式的秘密共享, 使得参与方扩大到 $2n$ 方、或者利用同态加密, 但未考虑恶意参与方的可身份识别的中止. 在实际部署中缺乏灵活性和健壮性. 一个健壮的门限 SM2 签名需要满足对恶意用户达到可身份识别中止. 灵活性是指满足任意的门限 $t < n$, 并且有密钥更新策略, 达到相对高效的实际运行速度等等. 本文基于这些目标尽可能的设计实用的 SM2 门限签名方案.

3.2 调整的 SM2 签名算法

首先对 SM2 签名的公私钥进行了修改以适应其门限形式. 由 x 和 xG 变为 x 和 $(x^{-1} - 1)G$. 然后签名方程中 r 没有改变, $s = x(k + r) - r \bmod q$, 如表3.1所示.

表 3.1: 调整后的 SM2 公私钥对

Sym.	SM2	Modified SM2
sk	x	x
pk	xG	$(x^{-1} - 1)G$
x_1	$(x_1, y_1) = kG$	$(x_1, y_1) = kG$
e	$H_v(Z_A m)$	$H_v(Z_A m)$
r	$e + x_1 \bmod q$	$e + x_1 \bmod q$
s	$(1 + x)^{-1}(k - rx) \bmod q$	$x(k + r) - r \bmod q$

如果 q 是素数, 那么 $x_{msm2} = (1 + x_{sm2})^{-1} \bmod q$ 是从 $[1, q - 2]$ 到 $[2, q - 1]$ 的双射. 对 x_{sm2} 保密等价于对 x_{msm2} 保密, 使用 x_{msm2} 作为”私钥”不影响 SM2 签名算法的安全性. 并且这些修改不改变 SM2 签名的验证过程.

$$\begin{aligned} (x_1, y_1) &= sG + tpk \\ &= sG + (r + s)pk \end{aligned}$$

$$\begin{aligned} (x'_1, y'_1) &= sG + tpk \\ &= sG + (r + s)pk \end{aligned}$$

$$\begin{aligned}
 &= sG + sxG + rxG &= sG + s(x^{-1} - 1)G + r(x^{-1} - 1)G \\
 &= (1+x)^{-1}(k - rx)(G + xG) + rxG &= (x(k+r) - r)x^{-1}G + r(x^{-1} - 1)G \\
 &= (1+x)^{-1}(k - rx)(1+x)G + rxG &= (k+r)G - rx^{-1}G + rx^{-1}G - rG \\
 &= (k - rx)G + rxG &= (k+r)G - rG \\
 &= kG &= kG
 \end{aligned}$$

3.3 理想门限签名函数定义

下面我们给出理想门限签名函数 F , 可以理解为存在一个可信参与方 F 为每一个参与方分发签名碎片, 提供预言机查询和记录相关信息.

理想门限签名函数 F

包含四个阶段: 密钥生成、签名、验证和密钥更新.

密钥生成: 当从参与方 P_i 接收到 $(keygen, tig, t, i)$,

1. 解释 $tig = (\mathbf{P}, params)$, 其中 $\mathbf{P} = (P_1, P_2, \dots, P_n)$.
 - 如果 $P_i \in \mathbf{P}$, 向 S 发送 $(keygen, tig, t, i)$ 并记录 $(keygen, tig, t, P_i)$.
 - 否则, 忽略此消息.
2. 当每一个 $P_j \in \mathbf{P}$ 都记录了 $(keygen, tig, t, P_j)$, 向 S 发送 $(publickey, tig, t)$ 并做:
 - (a) 当从 S 接收到 $(publickey, tig, t, pk)$, 记录 $(publickey, tig, t, pk)$.
 - (b) 当从 P_i 接收到 $(publickey, tig, t)$.
 - 如果 $(publickey, tig, t, pk)$ 已记录, 输出 $(publickey, tig, t, pk)$.
 - 否则, 忽略此消息.

签名: 初始化 $\mathbf{Q} = \emptyset$, 当从参与方 P_i 接收到 $(sign, sid = (tig, \dots), msg)$, 将 P_i 增加到 \mathbf{Q} .

1. 向 S 发送 $(sign, sid = (tig, \dots), msg)$, 并记录 $(sign, sid, msg, i)$.
2. 当从 S 接收到 $(sign, sid = (tig, \dots), msg, j)$, 如果 P_j 没有被损害, 记录 $(sign, sid, msg, j)$, 否则, 忽略此消息.
3. 当 $|\mathbf{Q}| > t$, 向 S 发送 $(sign, sid, msg)$, 并做:
 - (a) 当从 S 接收到 $(signature, sid, msg, \sigma, U)$, 这里 $U \in \mathbf{P}$:
 - 如果 $(sid, msg, \sigma, 0)$ 已记录, 输出 error.
 - 否则, 如果 (msg, σ) 是公钥 pk 的合法签名, 记录 $(sid, msg, \sigma, 1)$.
 - 否则, 如果 U 被损害, 记录 U 为恶意参与方.
 - 否则记录第一个被损害方为恶意方.
 - (b) 当从 $P_i \in \mathbf{P}$ 接收到 $(signature, sid, msg)$:
 - 如果 $(sid, msg, \sigma, 1)$ 已记录, 向 P_i 发送 $(signature, sid, msg, \sigma)$,

- 否则, 输出恶意方的身份.

验证: 当从任何参与方 P^* 接收到 $(verify, sid, msg, \sigma, pk)$ 时, 其中 pk 是公钥.

1. 向 S 发送 $(verify, sid, msg, \sigma, pk)$ 并做:
 - 如果 (msg, σ, b') 已记录, 设置 $b = b'$.
 - 否则, 如果 P^* 没有对 msg 签名, 设置 $b = 0$.
 - 否则, 如果 (msg, σ) 是 pk 的合法签名, 设置 $b = 1$.
 - 否则, 设置 $b = 0$.

记录 (msg, σ, b) 并向 P^* 发送 $(isture, sid, msg, \sigma, b)$.

密钥更新: 初始化 $\mathbf{Q} = \emptyset$; 当从 P_i 接收到 key-refresh, 将其发送给 S , 将 P_i 增加到 \mathbf{Q} 并做:

1. 当 $|\mathbf{Q}| \geq t$,
 - 刷新各方的公钥、私钥碎片、Paillier 公私钥或门限 t .

理想函数 F 包含四个部分: 密钥生成、签名、验证和密钥更新. 密钥生成向每一个参与方分配共同的公钥. 签名过程中, 只要 t 方准备签名, 即为参与方分配签名 (r, s) . 验证过程中, 对 (msg, σ) 的签名进行记录和验证. 如果 (msg, σ) 已经被记录, 返回已记录的状态 b , 否则, 如果 (msg, σ) 未被记录且 msg 没有过签名, 返回 0 表示对未签名的消息进行验证时返回验证失败 (存在不可伪造性). 如果 (msg, σ) 签过名, 且是合法签名返回验证成功, 不是合法签名返回验证失败. 密钥更新则是附加属性, 提供不同的密钥更新策略.

3.4 门限 SM2 签名的基本结构

从图 3.1 可以看出, SM2 签名主要由 (Key-gen, Sign, Verify, Key-refresh) 组成, 其中 Key-refresh 是自适应的, 不包含在原始的 SM2 签名中. 在门限 SM2 签名中, 将 Key-gen 分为两个部分 Paillier-keygen 和 SM2Ts-keygen, 唯一的区别是用 Paillier-keygen 生成合法的 Paillier 公私钥对, 而 SM2Ts-keygen 用于生成 SM2 的公私钥对. 然后将 Sign 分为 Pre-signing 和 Signing, 因为 Sign 主要是两个步骤, 一个是随机数的产生, 另一个是签名的产生, 两者互相独立并且有先后次序, 在门限签名中, 随机数的产生需要共同产生, 所以我们将 Pre-signing 用于生成随机数和相关签名信息的准备, 而在 Signing 中, 只等待消息 msg 的确认, 即可计算出签名. 另外 SM2 和门限 SM2 签名的 Verify 是一样的, 保证其通用性. 最后提供可选择的 Key-refresh 用于密钥更新, 其中 Global key-refresh 和 Threshold key-refresh 分别用于全局密钥更新和门限值更新.

每一个子协议都包含更基本的协议, 例如在 Paillier-keygen 中, 想要安全的生成 Paillier 公私钥对, 需要调用 ZKprn, ZKmod, ZKfac 等零知识证明分别用于证明 Ring-Pedersen 参数, 合法的 Paillier 公钥和无小因子的 Paillier 私钥.

在 SM2Ts-keygen 中, 包含了两个子协议 MultiAdd 和 VSS, 正如前面的预备知识中所示, MultiAdd 是参与方多次调用 MtAwc, 另外假设哈希函数为随机预言机做理想承诺函数 (Commit-

ment), 抵抗流氓密钥攻击, 剩下的零知识证明 $ZKlog^*$, $ZKenc^*$, $ZKlog$ 用于验证计算步骤以阻止恶意方偏离协议. Pre-signing 也调用了 MultiAdd 用于共享私钥和随机数的乘积.

SM2	Threshold SM2	Subprotocol	Subsub-rotocol
Key-gen	Paillier-keygen		ZKprm
			ZKmod
			ZKfac
	SM2Ts-keygen	Commitment (secure Hash)	
		MultiAdd	$ZKlog^*$
		VSS	$ZKenc^*$
		ZKlog	$ZKlog$
Sign	Pre-signing	MultiAdd	$ZKlog^*$
	Signing		$ZKenc^*$
Verify	Verify		
Key-refresh	Key-refresh	Global key-refresh	VSS
			$ZKlog$
		Threshold key-refresh	VSS

图 3.1: SM2 门限签名结构

3.5 门限 SM2 签名主协议

接下来, 本文给出门限 SM2 签名方案, 简称为 SMT, 分为 9 部分, 其中:

门限 SM2 签名主协议: 如图3.2为 SM2 门限签名主协议, 调用其他协议完成门限签名的密钥生成, 预签名, 签名和密钥更新.

具体的 8 个子协议如下, 有 Paillier-keygen step, SM2Ts-keygen step, Pre-signing step, Signing step, MultiAdd- $(aux, A_i, B_i, a_i, b_i)$ step, VSS- (aux, X_i, x_i, t, n) step, Global- (n, n) key-refresh step, Threshold- (t', n) key-refresh step.

Paillier-keygen step: 每个参与方生成 Paillier 公私钥并广播 Paillier 公钥.

SM2-keygen step: 每个参与方生成自己的私钥, 并生成“公钥” $x^{-1}G - G$ 广播. 同时, 每一方存储自己的“私钥” $[x]$ 即 x_i 和 (t, n) -加法分享 y_i . 更详细的说: 每一个参与方选取随机数 $[\gamma]$ 和 $[x]$, 调用 VSS step 得到 (t, n) -加法分享 y_i , 调用 MultiAdd step 计算 $[\gamma x]$, 然后恢复 γx , 计算 $(\gamma x)^{-1}[\gamma G]$ 作为 $[x^{-1}G]$, 然后恢复 $x^{-1}G$ 并计算 $X = x^{-1}G - G$.

Pre-signing step: t 方选取随机数 $[k]$, 计算 $[kG]$, 并调用 MultiAdd step 计算 kG 和 $[kx]$.

Signing step: 在每一方接收到要签名消息 msg 后, 使用 Pre-signing step 的 kG 和 $[kx]$ 计算其部分签名 $(r, [s])$, 然后广播. 这里 $[s] = [kx] + r[x]$. 每一个参与方得到 (r, s) 验证 $(r, s - r \bmod q)$

是否是 msg 的合法签名.

方案中有两个子协议和 MultiAdd step 和 VSS step, 其中:

MultiAdd step: 多个参与方调用 MtAwc [22], 即每个参与方输入自己的私有值 a_i, b_i 得到输出 c_i . 其中 $c_1 + c_2 + \dots + c_n = (a_1 + a_2 + \dots + a_n)(b_1 + b_2 + \dots + b_n)$. 这时 c_1, c_2, \dots, c_n 可以作为 ab 的加法共享 [ab].

VSS step: 每个参与方输入自己的身份 d_i , 私有值 x_i 和门限 t , 其中 $|\mathbf{P}| = n$. 得到输出 (t, n) -加法分享 y_i 和 $X_i = x_i G$.

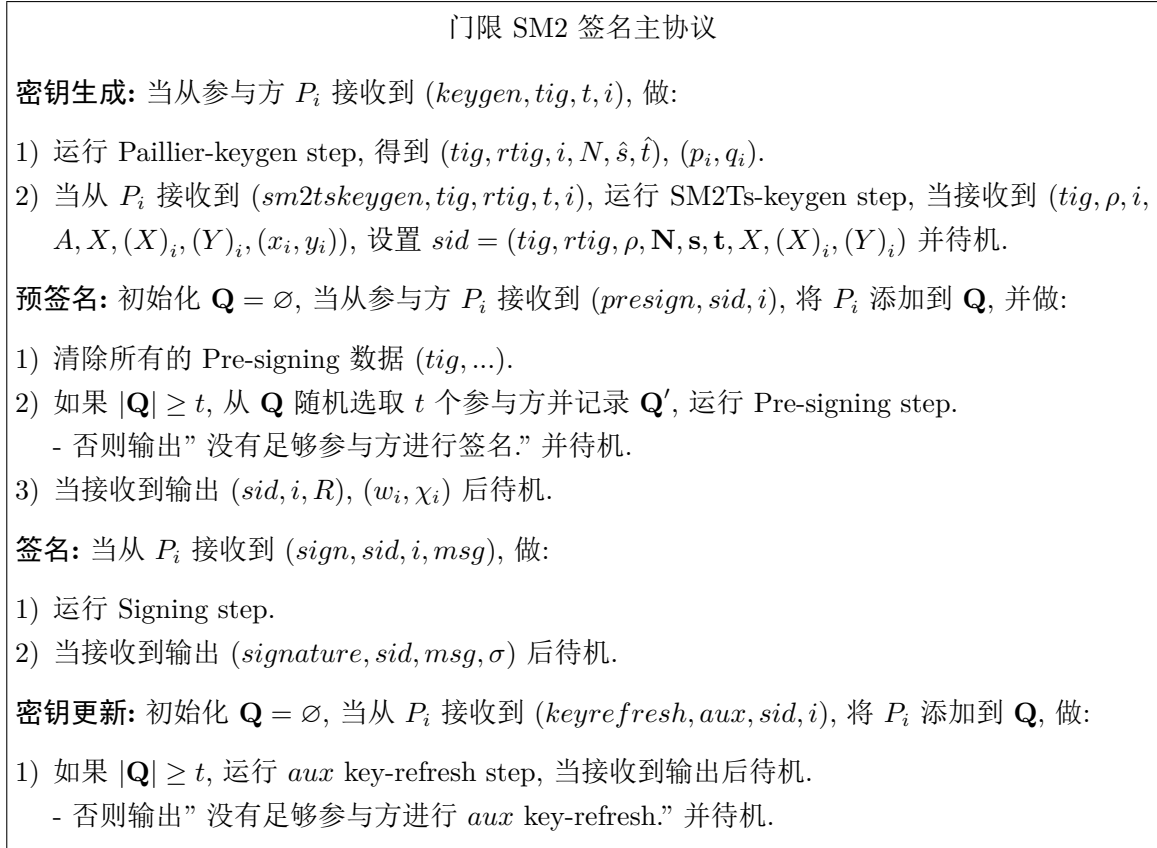


图 3.2: 门限 SM2 签名主协议

除此之外, 本文还增加了密钥更新策略, 有两个选项:

Global key-refresh step: 在主公钥不变的情况下, 每个参与方更新私钥碎片 x_i .

Threshold key-refresh step: 每个参与方更新门限 t 为 t' .

Paillier-keygen step

Round 1.

当从 P_i 接收到 $(keygen, tig, m, i)$, 解释 $tig = (\mathbf{P}, \kappa, a, b, n, G, ID_P, ENTL_P)$, 这里 ID_P 是 $(d_1 || d_2 || \dots || d_n)$, $d_i \in Z_n^*$ 是 P_i 的唯一身份符号, 做:

- 随机选取 $rtig_i = \{0, 1\}^\kappa$, 两个 4κ -比特的素数 (p_i, q_i) , 设置 $N_i = p_i q_i$.

- 随机选取 $r \leftarrow Z_{N_i}^*, \nu \leftarrow Z_{\phi(N_i)}$, 设置 $\hat{t}_i = r^2 \bmod N_i$, $\hat{s}_i = \hat{t}_i^\nu \bmod N_i$.
- 向 ZK_{prm} 发送 $(prove, ZK_{prm}, (tig, i), (N_i, \hat{s}_i, \hat{t}_i); \nu)$,
- 向 ZK_{mod} 发送 $(prove, ZK_{mod}, (tig, i), N_i; (p_i, q_i))$,
- 向 ZK_{fac} 发送 $(prove, ZK_{fac}, (tig, i), N_i; (p_i, q_i))$.
- 向所有的 P_j 发送 $(tig, i, rtig_i, N_i, \hat{t}_i, \hat{s}_i)$.

Output:

1. 当从 ZK_{prm} 接收到 $(verify, ZK_{prm}, (tig, j), N_j, \hat{s}_j, \hat{t}_j, \beta_j)$,
 - 从 ZK_{mod} 接收到 $(verify, ZK_{mod}, (tig, j), N_j, \beta'_j)$,
 - 从 ZK_{fac} 接收到 $(verify, ZK_{fac}, (tig, j), N_j, \hat{\beta}'_j)$.
- 如果有 $\beta_j == 0$ 或者 $\beta'_j == 0$ 或者 $\hat{\beta}'_j == 0$, 报告失败方并中止.
2. 当从 P_j 接收到 $(tig, j, rtig_j, N_j, \hat{t}_j, \hat{s}_j, \hat{\psi}_j, \psi_j, \phi_j)$, 验证 $N_j \geq 2^{8\kappa}$.
 - 设置 $rtig = rtig_1 + rtig_2 + \dots + rtig_n$,
- 输出 $(tig, rtig, i, \mathbf{N} = (N_j), \mathbf{s} = (\hat{s}_j), \mathbf{t} = (\hat{t}_j))$.

Errors. 如果验证过程失败, 报告失败方并中止.

Stored state. 存储 $(tig, rtig, \mathbf{N}, \mathbf{s}, \mathbf{t})$ 和 (p_i, q_i) .

SM2Ts-keygen step

P_i 的 stored state 为 $N_i = p_i q_i, p_i, q_i$.

Round 1.

- 当从 P_i 接收到 $(sm2tskeygen, tig, rtig, m, i)$, 解释 $tig = (\mathbf{P}, params)$, 做:
- 随机选取 $x_i, \gamma_i \leftarrow Z_N^*$, 设置 $X_i = x_i G, \Gamma_i = \gamma_i G$.
 - 随机选取 $\rho_i, u_i \leftarrow \{0, 1\}^\kappa$, 设置 $V_i = H(tig, rtig, i, X_i, \Gamma_i, \rho_i, u_i)$.
- 向所有的 P_j 发送 $(tig, rtig, i, V_i)$.

Round 2.

当接收到所有的 (tig, j, V_j) , 向所有的 P_j 发送 $(tig, i, X_i, \Gamma_i, \rho_i, u_i)$.

Round 3.

1. 当接收到所有的 $(tig, j, X_j, \Gamma_j, \rho_j, u_j)$, 做:
 - 验证 $H(tig, rtig, j, X_j, \Gamma_j, \rho_j, u_j) = V_j$.
2. 当所有 P_j 的验证都通过, 设置 $\rho = \rho_1 + \rho_2 + \dots + \rho_n$, 做:
 - 运行 $VSS(tig, rtig, \rho, X_i, x_i, t, n)$ step,

当接收到输出 (tig, i, A, Y_i) , 且 stored state 为 $(A, Y_i), (x_i, y_i)$.

 - 运行 $MultiAdd(tig, rtig, \rho, X_i, \Gamma_i, x_i, \gamma_i)$ step,

当接收到输出 (tig, i, Γ) , 并且 stored state 为 $(x_i, \gamma_i, \delta_i)$.

 - 设置 $\Delta_i = x_i \Gamma$. 向 ZK_{log} 发送 $(prove, ZK_{log}, (tig, rtig, \rho, i), (Y_i, G); y_i)$,
 - 向 ZK_{log} 发送 $(prove, ZK_{log}, (tig, rtig, \rho, i), (\Delta_i, \Gamma); x_i)$.

向所有的 P_j 发送 $(tig, i, Y_i, \Delta_i, \psi_i, \psi'_i, \delta_i)$.

Output:

1. 当从 ZK_{log} 接收到 $(verify, ZK_{log}, (tig, rtig, \rho, j), (Y_j, G), \beta_j)$,
- 从 ZK_{log} 接收到 $(verify, ZK_{log}, (tig, rtig, \rho, j), (\Delta_j, \Gamma), \beta'_j)$.

如果有 $\beta_j = 0$ 或者 $\beta'_j = 0$, 报告失败方并中止.

2. 当接收到所有的 $(tig, j, Y_j, \Delta_j, \psi_j, \psi'_j, \delta_j)$, 做:

- 设置 $\delta = \sum \delta_j \bmod n$, 验证 $\delta G = \sum \Delta_i$, 如果验证过程失败, 运行 SM2Ts checker.
- 设置 $X = \delta^{-1} \Gamma - G$, 输出 (tig, i, X) .

清除了 stored state 的所有数据.

Errors. 如果验证过程失败, 报告失败方并中止

Stored state. 存储 $(tig, A, X, (X)_i, (Y)_i), (p_i, q_i, x_i, y_i)$.

SM2Ts checker: 检查 MultiAdd step 数据.

1. 向 ZK_{enc*} 发送 $(prove, ZK_{enc*}, (tig, rtig, \rho, i), E_{j,i}, G_j, D_{j,i}, B_i); (b_i, \beta_{i,j}, f_{i,j}, g_{i,j}))$, $j \neq i$.
2. 计算 $U_i = ENC_i(a_i b_i)$ 并向 ZK_{mul} 发送 $(prove, ZK_{mul}, (tig, rtig, \rho, i), (G_i, F_i, U_i); a_i, \dots)$.
3. 向 ZK_{enc} 发送 $(prove, ZK_{enc}, (tig, rtig, \rho, i), U_i \times \prod_{j \neq i} (E_{i,j} \times D_{j,i}); b_i a_i + \sum_{j \neq i} (\alpha_{i,j} + \beta_{i,j}))$.

Pre-signing step

P_i 的 stored state 为 $N_i = p_i q_i, p_i, q_i, y_i$.

Round 1.

当从 P_i 接收到 $(presign, sid, i)$, $sid = (\mathbf{P}, \kappa, a, b, n, G, ID_P, ENTL_P, rtig, X, \mathbf{N}, \mathbf{s}, \mathbf{t})$, 做:

1. 随机选取 $k_i \leftarrow Z_N^*$, 设置 $\Lambda_i = k_i G$.
- 计算 $w_i = \lambda_i y_i$, $W_i = w_i G$ 其中 $\lambda_i = \prod_{k=0, k \neq i}^{t-1} (d_i - d_k)^{-1} (-d_k)$.
- 验证 $A = \sum_{j=1}^t W_j$, 如果验证过程失败, 运行 presigning checker.
2. 运行 MultiAdd- $(sid, W_i, \Lambda_i, w_i, k_i)$ step.
- 当接收到输出 (sid, i, R) , 且 stored state 为 (w_i, k_i, χ_i) . 输出 (sid, i, R) .

清除了 stored data 的所有数据.

Errors. 如果验证过程失败, 报告失败方并中止.

Stored state. 存储 $(sid, R), (p_i, q_i, x_i, y_i, w_i, \chi_i)$.

Presigning check: 检查 VSS step 的数据,

1. 检查 $Y_i = \sum_{j=1}^n (X_j + \sum_{k=1}^m (d_i)^k A_{j,k}), i = 1, \dots, n$.
2. 计算 $C_i = \sum_{j=1}^n C_{j,i}$, 向 ZK_{log*} 发送 $(prove, ZK_{log*}, sid, (C_i, Y_i, G); \dots)$.
3. 计算 $Y_{i,j} = X_i + \sum_{k=1}^{m-1} (d_j)^k A_{i,k}$, 向 ZK_{log*} 发送 $(prove, ZK_{log*}, sid, (C_{i,j}, Y_{i,j}, G); \dots)$.

Signing step

Round 1.

当从 P_i 接收到 $(sign, sid, i, msg)$, 如果 (sid, i, R) 和 (w_i, χ_i) 已记录, 做:

- 计算 $Z = H(ID_P || ENTL_P || \mathbf{P} || a || b || x_G || y_G || x_X || y_X)$. $e = H(Z || msg)$,
 - 计算 $r = (R_{x-axis} + e) \bmod q$, $s_i = \chi_i + w_i r \bmod q$.
- 向所有的 P_j 发送 (sid, i, s_i) .

Output:

- 当接收到所有的 (sid, j, s_j) , 设置 $s = \sum s_j - r \bmod q$, 做:
- 验证 $\sigma = (r, s)$ 是 msg 和 X 的合法签名, 如果验证过程失败, 运行 Signing checker.
- 输出 $(signature, sid, m, \sigma)$, 清除 (sid, i, w_i, χ_i) .

Errors. 如果验证过程失败, 报告失败方并中止.

Signing checker: 检查 MultiAdd step 数据.

1. 向 ZK_{enc*} 发送 $(prove, ZK_{enc*}, sid, E_{j,i}, G_j, D_{j,i}, B_i); (b_i, \beta_{i,j}, f_{i,j}, g_{i,j})$, $j \neq i$.
2. 计算 $U_i = ENC_i(a_i b_i)$ 并向 ZK_{mul} 发送 $(prove, ZK_{mul}, sid, (G_i, F_i, U_i); a_i, \dots)$.
3. 向 ZK_{enc} 发送 $(prove, ZK_{enc}, sid, U_i \times \prod_{j \neq i} (E_{i,j} \times D_{j,i}) \times r \cdot F_i; b_i a_i + \sum_{j \neq i} (\alpha_{i,j} + \beta_{i,j}) + w_i r)$.

MultiAdd- $(aux, A_i, B_i, a_i, b_i)$ step

P_i 的 stored state 为 $N_i = p_i q_i, p_i, q_i$.

Round 1.

- 当从 P_i 接收到 $(multiadd, aux, i)$, 解释 $aux = (P, params)$, 做:
- 随机选择 $v_i \leftarrow Z_{N_i}^*$, 设置 $G_i = ENC_i(a_i, v_i)$.
- 向 ZK_{log*} 发送 $(prove, ZK_{log*}, (aux, i), (G_i, A_i, G); a_i, v_i)$.
- 向所有的 P_j 发送 (aux, i, A_i, B_i, G_i) , 如果 A_i 和 B_i 已经发送过, 忽略 A_i 和 B_i .

Round 2.

1. 当从 ZK_{log*} 接收到 $(verify, ZK_{log*}, (aux, j), (G_j, A_j, G), \beta_j)$.
- 如果有 $\beta_j == 0$, 报告失败方并中止.
2. 当从 P_j 接收到 (aux, j, A_j, B_j, G_j) , 设置 $B = \sum B_i$.
- 对每一个 $j \neq i$ 随机选取 $f_{i,j}, g_{i,j} \leftarrow Z_{N_j}, \beta_{i,j} \leftarrow J$.
 - 计算 $E_{j,i} = b_i \cdot G_j \times ENC_j(-\beta_{i,j}, f_{i,j})$, $D_{j,i} = ENC_i(\beta_{i,j}, g_{i,j})$.
- 向 ZK_{enc*} 发送 $(prove, ZK_{enc*}, (aux, i), (E_{j,i}, G_j, D_{j,i}, B_i); (b_i, \beta_{i,j}, f_{i,j}, g_{i,j}))$.
- 向 P_j 发送 $(aux, i, E_{j,i}, D_{j,i})$.

Output:

1. 当从 ZK_{enc*} 接收到 $(verify, ZK_{enc*}, (aux, i), (E_{i,j}, G_i, D_{i,j}, B_j), \beta_{i,j})$.
- 如果有 $\beta_{i,j} == 0$, 报告失败方并中止.
2. 当接收到所有 $P_j \in P$ 的 $(aux, j, E_{i,j}, D_{i,j})$, 对每一个 $j \neq i$, 设置 $\alpha_{i,j} = DEC_i(E_{i,j})$.
- 计算 $\delta_i = b_i a_i + \sum_{j \neq i} (\alpha_{i,j} + \beta_{i,j}) \bmod n$.
- 输出 (aux, i, B) .

Errors. 如果验证过程失败, 报告失败方并中止

Stored state. 存储 $(aux, \mathbf{N}, \mathbf{s}, \mathbf{t}), (p_i, q_i, a_i, b_i, \delta_i)$.

VSS- (aux, X_i, x_i, t, n) step

Round 1.

当从 P_i 接收到 $(vsstsgen, aux, t, i)$, 解释 $aux = (ID_P, params)$, 做:

- 随机选取 $a_{i,1}, \dots, a_{i,t-1} \leftarrow Z_N^*$, 设置 $A_{i,1} = a_{i,1}G, \dots, A_{i,t-1} = a_{i,t-1}G$.
 - 计算 $y_{i,j} = x_i + \sum_{k=1}^{t-1} (d_j)^k a_{i,k}, j = 1, \dots, n, C_{i,j} = ENC_j(y_{i,j}), j = 1, \dots, n$.
- 向 ZK_{log} 发送 $(prove, ZK_{log}, (tig, i), (X_i, G); x_i)$.

向所有的 P_j 发送 $(aux, i, A_{i,1}, \dots, A_{i,t-1}, C_{i,j})$.

Output:

1. 当从 ZK_{log} 接收到 $(verify, ZK_{log}, (tig, j), (X_j, G), \beta_j)$.
如果有 $\beta_j == 0$, 报告失败方并中止.
2. 当从 P_j 接收到 $(aux, j, A_{j,1}, \dots, A_{j,t-1}, C_{j,i})$. 当所有 P_j 的验证都通过, 设置 $A = \sum X_i$.
- 计算 $y'_{j,i} = DEC_i(C_{j,i}), j = 1, \dots, n$, 验证 $y'_{j,i}G = X_j + \sum_{k=1}^{t-1} (d_i)^k A_{j,k}, j = 1, \dots, n, j \neq i$.
- 计算 $y_i = \sum_{j=1}^n y'_{j,i}, Y_i = y_i G$,
输出 (aux, i, A, Y_i) .

Errors. 如果验证过程失败, 报告失败方并中止.

Stored state. 存储 $(A, Y_i), (x_i, y_i)$.

Global- (n, n) key-refresh step

P_i 的 stored state 为 X_i, x_i .

Round 1.

当从 P_i 接收到 $(keyrefresh, n, n, sid, i)$, 解释 $sid = (\mathbf{P}, \kappa, a, b, n, G, ID_P, ENTL_P, rtig, X, \mathbf{N}, \mathbf{s}, \mathbf{t})$,
运行 VSS- (sid, X_i, x_i, n, n) step.

- 当接收到输出 (sid, i, A, Y_i) , 且 stored state 为 $(A, Y_i), (x_i, y_i)$.

向 ZK_{log} 发送 $(prove, ZK_{log}, (tig, rtig, \rho, i), (Y_i, G); y_i)$.

向所有的 P_j 发送 (sid, i, Y_i, ψ_i) .

Output:

1. 当从 ZK_{log} 接收到 $(verify, ZK_{log}, (tig, rtig, \rho, j), (Y_j, G), \beta_j)$.
如果有 $\beta_j == 0$, 报告失败方并中止.
2. 当接收到所有的 (sid, j, Y_j, ψ_j) , 验证 $A = \sum_{j=1}^n \lambda_j Y_j, \lambda_i = \prod_{k=0, k \neq i}^{n-1} (d_i - d_k)^{-1} (-d_k)$.
如果验证过程失败, 运行 Global key-refresh checker.
- 计算 $x_i = \lambda_i y_i, X_i = \lambda_i Y_i$.
输出 (sid, i, X_i) , 清除除了 stored state 的所有数据.

Errors. 如果验证过程失败, 报告失败方并中止.

Stored state. 存储 (sid, X_i) , 和 (p_i, q_i, x_i) .

Global key-refresh checker: 检查 VSS step 数据,

1. 检查 $Y_i = \sum_{j=1}^n (X_j + \sum_{k=1}^m (d_i)^k A_{j,k}), i = 1, \dots, n$.
2. 计算 $C_i = \sum_{j=1}^n C_{j,i}$, 向 ZK_{log*} 发送 $(prove, ZK_{log*}, (C_i, Y_i, G); \dots)$.
3. 计算 $Y_{i,j} = X_i + \sum_{k=1}^{m-1} (d_j)^k A_{i,k}$, 向 ZK_{log*} 发送 $(prove, ZK_{log*}, (C_{i,j}, Y_{i,j}, G); \dots)$.

Threshold- (t', n) key-refresh step

P_i 的 store state 为 X_i, x_i .

Round 1.

当从 P_i 接收到 $(keyrefresh, t', n, sid, i)$, 解释 $sid = (\mathbf{P}, \kappa, a, b, n, G, ID_P, ENTL_P, rtig, X, \mathbf{N}, \mathbf{s}, \mathbf{t})$.

- 运行 $VSS(sid, X_i, x_i, t', n)$ step, 当接收到输出 (sid, i, A, Y_i) , 和 stored state $(A, Y_i), (x_i, y_i)$.

输出 (sid, i, Y_i) , 清除除了 stored state 的所有数据.

Errors. 如果验证过程失败, 报告失败方并中止.

Stored state. 存储 $(sid, X_i, Y_i), (p_i, q_i, x_i, y_i)$.

协议按轮 (Round) 执行, 在经过身份验证的信道中通信. 通信模式有两种点对点通信和广播模式, 向 P_j 发送 (aux, i, \dots) 代表 P_i 到 P_j 的点对点通信网络, 向所有的 P_j 发送 (aux, i, \dots) 代表向 P_i 到 P 中所有参与方的广播模式.

当每一方在 SM2Ts-keygen step Round 1 中的随机选取公钥碎片 X_i 时, 它需要承诺 (Commitment) 加法共享以防止恶意参与方替换公钥. 同样, 在 Pre-signing step Round 1 中的随机数 R 的生成也使用了 Paillier 承诺, 即使用 Paillier 密文来保证随机数不被替换, 这可以在 MultiAdd- $(aux, A_i, B_i, a_i, b_i)$ step Round 1 的 ZK_{log*} 中找到.

在识别恶意方过程中, 主要思想是每一方的每一步都进行了零知识证明. 例如 aG 需要广播, 则 aG 的零知识证明也进行了广播, 接收方可以验证发送方有秘密 a , 且广播消息为 aG . 因此, 如果敌手控制的损害方试图替换公钥, 他无法计算出他当前的私钥, 因为他没有其他参与方的私钥, 无法发送正确的 ZK_{log} .

公钥验证和签名验证出现错误时并不容易识别损害方, 为此我们设置了额外的检查. 在 VSS 步中, 需要额外执行 ZK_{log*} 来识别损害方发送的消息. 在 MultiAdd 步中, 需要执行额外的 ZK_{log*}, ZK_{mul} 和 ZK_{enc} , 因为只有 δ_i 的计算过程没有被零知识证明.

密钥刷新步骤有不同的选项: 全局密钥刷新 (Global- (n, n) key-refresh step) 允许每一方在主公钥不变下刷新他们的秘密值 x_i , 门限密钥刷新 (Threshold- (t', n) key-refresh step) 是让每一方改变他们的门限值 t' .

上面的 key-refresh 策略是 (t, n) , 本文协议能改变参与方集合 $|P|$ 的大小 n , 即删除和添加参与方. 事实上, 只需要声明的一方 Q 无效, 很容易删除参与方. 然而在不改变公钥的情况下添加一方并不容易, 需要对所有参与方的公钥碎片进行刷新, 协议兼容方案 [47, 48], 可以安全有效的添加参与方.

我们在这里总结需要被证明的 NP 关系. 具体协议见附录.

$$prm = \{(N, \hat{s}, \hat{t}; \nu) | \hat{s} = \hat{t}^\nu \bmod N\} \quad (3.1)$$

$$mod = \{(N; p, q) | N = pq, \gcd(N, \phi(N)) = 1, p, q = 3 \bmod 4\} \quad (3.2)$$

$$fac = \{(l, N_0; p, q) | p, q < \pm 2^l \sqrt{N_0}\} \quad (3.3)$$

$$enc = \{(l, N_0, K; k, \rho) | k \in \pm 2^l, K = (1 + N_0)^k \rho^{N_0} \bmod N_0^2\} \quad (3.4)$$

$$log = \{(X, Y; x) | X = xY\} \quad (3.5)$$

$$enc* = \left\{ \left(\begin{array}{l} l, l', N_0, N_1, \\ D, C, Y, X; \\ x, y, \rho_x, \rho_y \end{array} \right) \middle| \begin{array}{l} x \in \pm 2^l, y \in \pm 2^{l'}, X = xG, \\ Y = (1 + N_1)^y \rho_y^{N_1} \bmod N_1^2, \\ D = C^y (1 + N_0)^x \rho_x^{N_0} \bmod N_0^2 \end{array} \right\} \quad (3.6)$$

$$log* = \{(l, N_0, C, X, G; x, \rho) | x \in \pm 2^l, X = xG, C = (1 + N_0)^x \rho^{N_0} \bmod N_0^2\} \quad (3.7)$$

$$mul = \{(N, E, D, C; x, \rho_x, \rho_y) | (1 + N)^x \rho_x^N = E, C = D^x \rho_y^N \bmod N^2\} \quad (3.8)$$

3.6 安全性分析

本文在 Ideal/Real 模式证明协议实现了理想的门限签名功能 F . 根据我们的安全模型中的理想函数 F 和理想随机预言机, 如果门限 SM2 协议没有实现理想函数 F , 则敌手 A 以显著优势区分 Paillier 密文或伪造增强 SM2 签名.

首先简单总结一下证明思路, 如果敌手 A 能区别现实协议和理想门限签名函数 F , 那么存在一个模拟器 S , 通过调用敌手 A 区分 Paillier 密文或伪造增强 SM2 签名. 重点是归约过程, 类似于游戏系列 (Game series) 证明.

- Game 0: 模拟器 S 实例 SM2 的 (Q_A, d_A) 和辅助信息 $Q'_A = (1 + d_A)^{-1}G$, 倒带敌手 A , 并将公钥设置为 Q_A .
 - 提取损害方的 Paillier 密钥.
 - 不解密诚实方的 Paillier 密文, 而是解密损害方的相关消息.
 - 调用诚实方的零知识证明来生成有效证明, 对随机预言机进行编程.

- Game 1: 模拟器 S 随机选择一个诚实的一方, 这被认为是特殊的.
- 每当诚实方指示损害方使用特殊方 Paillier 密钥加密一个值时, 模拟器发送随机加密 0 的值.
- Game 2: 使用 SM2 oracle 模拟不知道特殊参与方秘密的情况.

在 Game 0 即是模拟器 S 调用敌手 A 与门限 SM2 协议交互. Game 0 和 Game 1 在 Paillier 加密下是计算下是不可区分的. 因为如果模拟器 S 发送随机加密 0 的值, 对手的视图是无法区分的. Game 1 和 Game 2 是完美安全的, Game 2 相当于调用对手 A 伪造 SM2 签名, 如果不能伪造, 由理想门限签名函数定义, 敌手 A 的视图与理想门限签名函数交互一致.

定理 3.6.1 假设强 RSA, Paillier 加密方案的不可区分和增强 SM2 签名的存在不可伪造性, 协议在理想零知识函数 ZK_{Rel} 下实现了理想功能 F .

证明 由引理 3.6.1, 引理 3.6.2 可知. □

引理 3.6.1 如果敌手 A 使用附加信息 $Q'_A = (1+x)^{-1}G$ 输出伪造的增强 SM2 签名, 那么敌手 A 不使用附加信息就可以输出伪造增强 SM2 签名.

证明 由于离散对数困难性假设, Q'_A 不能向敌手提供有关于 x 的任何信息. □

注: SM2 存在不可伪造蕴含了离散对数难题假设.

引理 3.6.2 假设强 RSA, 如果有敌手 A 可以在执行协议时为先前未签名的消息伪造 SM2 签名, 则存在算法 R1 和 R2 访问敌手 A 与理想零知识函数 ZK_{Rel} 黑盒, 至少有一个是正确的,

1. R1 以大于 $1/2$ 的显著概率赢得 Paillier 不可区分性.
2. R2 以不可忽略的概率赢得增强 SM2 存在不可伪造游戏.

证明 设 A 为执行协议时可以伪造未签名消息的敌手, $T \in poly$ 表示伪造发生前辅助信息操作的上限. $N_1, \dots, N_T, (Q_A, d_A)$, 和 Q'_A , 表示根据协议选取的 Paillier 公钥, SM2 密钥对和辅助信息. 设 R1 为 Paillier-distinguisher, R2 为 SM2-forgery, 考虑以下三个实验:

实验 A. R1 使用参数 (Q'_A, Q_A, d_A) 和 $(N_k, c_k), k = 1, \dots, T$, 其中 $c_k = ENC_{N^k}(1)$.

实验 B. R1 使用参数 (Q'_A, Q_A, d_A) 和 $(N_k, c_k), k = 1, \dots, T$, 其中 $c_k = ENC_{N^k}(0)$.

实验 C. R2 使用参数 (Q'_A, Q_A) 调用敌手 A .

声明 3.6.1 假设强 RSA, 如果对手 A 在时间 τ 内以 α 的概率在协议中输出一个伪造签名, 那么 R1 可以在实验 A 中伪造签名的概率在时间 $n \log(n)\tau$ 内至少为 α^2 .

时间 $n \log(n)\tau$ 因为倒带步骤, α 减少到 α^2 因为它需要倒带零知识证明.

声明 3.6.2 假设 Paillier 加密方案的不可区分性, 如果敌手 A 在实验 A 中以概率 α 在时间 τ 输出伪造签名, 则 R1 可以在实验 B 中时间 τ 以概率 α 伪造签名.

声明 3.6.3 如果敌手 A 在时间 τ 中以概率 α 在实验 B 中输出伪造签名, 则 R2 在实验 C 在时间 τ 内以概率为 $\alpha + \text{negl}(\kappa)$ 输出伪造签名.

接下来描述了 R1 和 R2 的模拟和归约过程, 首先为每个阶段设置了一个独立的模拟器, 每个模拟器通过证明-验证列表 L 查询理想零知识函数. 例如, 在 R1 中, 归约过程为模拟器提供了一个挑战 (N, c) . 同理, 在 R2 中的密钥生成阶段, 归约过程为模拟器提供挑战 (SM2 公钥 Q_A 和 Q'_A) 试图赢得存在不可伪造游戏.

• Paillier-distinguisher R1

R1 以 $T \in \text{poly}(\kappa)$ 作为查询数量的上界, Paillier 公钥和密文分别为 $N_1, \dots, N_T, c_1, \dots, c_T$ 和 SM2 密钥 (Q_A, d_A) 和 $Q'_A = (1 + d_A)^{-1}G$. 设置 ctr 为计数器变量, 初始化 $ctr = 0$. 设 L 表示由模拟器 S 存储在内存中并初始化为空集的零知识证明列表. 算法 R1 通过以下步骤与对手 A 交互. 零知识证明查询 (Zero knowledge query)

当从敌手 A 接收到 $(\text{prove}, ZK_{Rel}, aux', x; w)$ 时, 执行:

1. 如果 $aux' \neq aux$, 向 ZK_{Rel} 发送 $(\text{prove}, ZK_{Rel}, x; w)$.
- 当接收到输出 $(\text{verify}, ZK_{Rel}, x, \beta)$, 返回 $(\text{verify}, ZK_{Rel}, aux', x, \beta)$.
2. 否则, 如果倒带步骤生成 $Rel = fac$, 则:
- 编写零知识程序, 提取 p, q , 即 $N = pq$. 将相关元组添加到 L .
3. 否则
(a) 如果 $(ZK_{Rel}, x, \beta) \in L$, 返回 $(\text{verify}, ZK_{Rel}, aux', x, \beta)$.
(b) 否则返回 $(\text{verify}, ZK_{Rel}, aux', x, \beta)$, $\beta = Rel(x; w)$, 并且将 (ZK_{Rel}, x, β) 添加到 L .

密钥生成 (Key-gen).

模拟器 S 在每个 P_i 的磁带上写入 $(\text{keygen}, \text{tig} = (\mathbf{P}, \text{params}, i))$ (包括被破坏方 $\mathbf{C} \subsetneq \mathbf{P}$). 根据给定分布随机 $N_0 = p_0q_0$ 设置 $aux_0 = N_0$, 然后加上 $ctr := ctr + 1$ 并设置 $aux = N_{ctr}$, 调用 $S_1(\text{tig}, C, L, aux)$ 得到输出, 然后输出 $b, L, \text{tig}, \text{rtig}, \{N_j, \hat{s}_j, \hat{t}_j\}_{j \in P}$, 检索 $\{(p_j, q_j)\}_{j \neq b}$.

然后模拟器 S 将 $(\text{sm2tskeygen}, \text{tig}, \text{rtig}, t, i)$ 写入 P_i 的磁带 (包括损坏方 $\mathbf{C} \subsetneq \mathbf{P}$).

随机选取 $\omega \leftarrow F_n$, 调用 $S_2(\text{tig}, C, L, Q'_A, Q_A, \omega)$ 得到输出.

输出 $b, L, \rho, \{x_k, y_k\}_{k \neq b}$, 设 $x_b = (1 + d_A)^{-1} - \sum_{j \neq b} x_j \bmod n$.

预签名 (Pre-signing).

模拟器 S 将 $(\text{presign}, \text{sid}, i)$ 写入每个 P_i 的磁带 (包括损坏的一方 $\mathbf{C} \subsetneq \mathbf{P}$).

在 P 中随机选取 t 方作为 \mathbf{Q}' . 如果 $P_b \notin \mathbf{Q}'$, 则倒带直到包含 P_b .

随机选取 $k_b \leftarrow F_n$, 设 $y^b = \{y_i\}_{i \neq b}$, $y_b = \lambda_b^{-1}((1 + d_A)^{-1} - \sum_{j=1, j \neq b}^n \lambda_j y_j)$, $aux = (c, k_b, y_b)$.

(a) 使用相关输入调用 S_3 .

签名 (Signing).

模拟器 S 在 $P_i \in \mathbf{Q}'$ 的磁带上写入 $(\text{sign}, \text{sid}, i, \text{msg})$.

(a) 检索 $R, (k_i, \chi_i)_{i \notin C}$, 设 $r = (R_{x\text{-axis}} + e) \bmod n$, $s_i = \chi_i + x_i r \bmod n$.

(b) 向敌手 A 发送 $\{(\text{sid}, i, s_i)\}_{i \notin C}$.

错误 (Errors).

1. 如果 SM2Ts checker 运行.
 - 对于 $P_i \in H$, 根据协议将相关信息提交给敌手 A .
 - 对于 P_b , 设置 $\hat{U}_b = ENC_b(0)$, 模拟 S_{mul}, S_{enc} 消息 $(verify, ZK_{mul}, (tig, rtig, \rho, b), (G_b, F_b, \hat{U}_b), 1)$, $(verify, ZK_{enc}, (tig, rtig, \rho, i), U_i \times \prod_{j \neq i} (E_{i,j} \times D_{j,i}), 1)$, 将相关元组添加到 L .
2. 如果 Presigning checker 运行.
 - 对于 $P_i \in H$, 根据协议将相关信息提交给敌手 A .
 - 对于 P_b , 根据协议计算 Y_b, C_b , 模拟 S_{log*} 消息 $(verify, ZK_{log*}, sid, (Y_b, C_b, G), 1)$.
 - 根据协议计算 $Y_{b,j}, C_{b,j}$, 模拟 S_{log*} 消息 $(verify, ZK_{log*}, sid, (Y_{b,j}, C_{b,j}, G), 1)$. 将相关元组添加到 L .
3. 如果 Signing step 失败.
 - 对于 $P_i \in H$, 根据协议将相关信息提交给敌手 A .
 - 对于 P_b , 设置 $\hat{U}_b = x_b \cdot G_b \times ENC_b(0)$, 模拟 S_{mul}, S_{enc} 消息 $(verify, ZK_{mul}, sid, (G_i, F_i, U_i), 1)$, $(verify, ZK_{enc}, sid, U_i \times \prod_{j \neq i} (E_{i,j} \times D_{j,i}) \times r \cdot F_i, 1)$, 将相关元组添加到 L .

• SM2 forger R2

SM2 由对手 A 和 SM2 签名与公钥 Q_A 和 $Q'_A = (1 + d_A)^{-1}G$. 设 L 表示由模拟器 S 存储在内存中并初始化为空集的证明列表. 算法 R2 通过以下步骤与对手 A 交互.

零知识证明查询 (Zero knowledge query).

与 Paillier-distinguisher R1 零知识查询一致.

密钥生成 (Key-gen).

模拟器 S 在每个 P_i 的磁带上写入 $(keygen, tig = (\mathbf{P}, params, i))$ (包括损坏方 $\mathbf{C} \subsetneq \mathbf{P}$).

调用 $S_1(tig, \mathbf{C}, L, \perp)$ 得到输出, 然后输出 $b, L, tig, rtig, \{N_j, \hat{s}_j, \hat{t}_j\}_{j \in P}$, 检索 $\{(p_j, q_j)\}_{j \in P}$.

然后模拟器 S 将 $(sm2tskeygen, tig, rtig, i)$ 写入每个 P_i 的磁带 (包括损坏的一方 $\mathbf{C} \subsetneq \mathbf{P}$).

随机选择 $\omega \leftarrow F_n$, 调用 $S_2(tig, \mathbf{C}, L, Q'_A, Q_A, \omega)$ 得到输出.

- 然后输出 $b, L, tig, rtig, \rho, \{x_k\}_{k \neq b}, \mathbf{X} = (X_1, \dots, X_n)$. 设置 $x_b = x - \sum_{j \neq b} x_j \bmod n$.

预签名 (Pre-signing).

模拟器 S 将 $(presign, sid, i)$ 写入每个 P_i 的磁带 (包括损坏的一方 $\mathbf{C} \subsetneq \mathbf{P}$).

在 \mathbf{P} 中随机选取 t 方作为 S' . 如果 $P_b \notin S'$, 则倒带直到包含 P_b .

设置 $y^{\setminus b} = \{y_i\}_{i \neq b}$,

(a) 询问 SM2 随机预言机得到消息 msg 和点 $R = sG + (r+s)Q_A$ 的签名 (r, s) , 设置 $aux = R$.

(b) 使用相关输入调用 S_3 .

签名 (Signing).

模拟器 S 在 $P_i \in \mathbf{Q}'$ 的磁带上写入 $(sign, sid, i, msg)$.

- 检索 (sid, i, η_0, η_1) 和 $(sid, i, R, k_i, \chi_i)_{i \in H}$.

(a) 对于每一个 $P_i \in H$, 按协议规定计算 s_i , 并向敌手 A 发送 (sid, i, s_i) .

(b) 对于 P_b , 设置 $s_b = s + r - \eta_0 r - \eta_1 \bmod n$, 并向敌手 A 发送 (sid, b, s_b) .

错误 (Errors).

1. 如果 SM2Ts checker 运行, 和 R1 的 SMTs-keygen 的 SM2Ts checker 一致.
2. 如果 Presigning checker 运行, 和 R1 的 Pre-signing 的 Presigning checker 一致.
3. 如果 Signing step 失败.

- 对于 $P_i \in H$, 根据协议将相关信息提交给敌手 A .

- 对于 P_b , 设置 $\hat{U}_b = ENC_b(0)$, 模拟 S_{mul} , S_{enc} 消息 $(verify, ZK_{mul}, sid, (G_i, F_i, U_i), 1)$, $(verify, ZK_{enc}, sid, U_i \times \prod_{j \neq i} (E_{i,j} \times D_{j,i}) \times r \cdot F_i, 1)$, 将相关元组添加到 L .

下面是每个阶段的模拟器, 其中 S_{Rel} 代表 ZK_{Rel} 的模拟器, $Rel \in \{prm, mod, fac, log, log*, enc, enc*, mul\}$.

• **Paillier-keygen** 模拟器 S_1

$S_1(tig, \mathbf{C}, L, aux)$, $tig = (\mathbf{P}, params)$, 一个身份信息 tig , 一组损害参与方 $\mathbf{C} \subset \mathbf{P}$, 用于证明-验证的列表 L , 辅助输入 $aux = \perp$ 或 $aux = N_*$.

Round 1.

随机选取 $P_b \leftarrow \mathbf{P} \setminus \mathbf{C}$, 设 $\mathbf{H} = \mathbf{P} \setminus \mathbf{C} \cup \{P_b\}$.

Case 1: $aux = N_*$.

对于每个 $P_i \in H$, 按照协议执行并提交 $(tig, i, rtig_i, N_i, \hat{t}_i, \hat{s}_i)$ 给对手 A .

对于 P_b , 随机选择一个 $rtig_b \leftarrow \{0, 1\}^\kappa$.

- 设置 $N_b = N_*$, 模拟 S_{mod} 消息 $(verify, ZK_{mod}, (tig, i), N_b, 1)$, 模拟 S_{fac} 消息 $(verify, ZK_{fac}, (tig, i), N_b, 1)$, 其他消息按协议执行.

将相关的元组添加到 L , 提交给对手 $(tig, b, rtig_b, N_b, \hat{t}_b, \hat{s}_b)$ A .

Case 2: $aux = \perp$.

对于每个 $P_i \notin C$, 根据协议执行并提交 $(tig, i, rtig_i, N_i, \hat{t}_i, \hat{s}_i)$ 给对手 A .

Output: 当接收到 A 向 ZK_{prm} 发送的 $(prove, ZK_{prm}, (tig, i), (N_i, \hat{s}_i, \hat{t}_i); \nu)$.

A 向 ZK_{mod} 发送的 $(prove, ZK_{mod}, (tig, i), N_i; (p_i, q_i))$.

A 向 ZK_{fac} 发送到 $(prove, ZK_{fac}, (tig, i), N_i; (p_i, q_i))$.

- 检查它们是否正确. 如果不是, 则将失败方发送到 ZK_{prm} 或 ZK_{mod} 或 ZK_{fac} , 然后中止.

当从敌手 A 接收到所有的 $(tig, j, rtig_j, N_j, \hat{t}_j, \hat{s}_j)$ 时.

按照协议执行, 然后输出 $b, L, tig, rtig$ 和 $\{N_j, \hat{s}_j, \hat{t}_j\}_{j \in P}$.

• **SM2Ts-keygen** 模拟器 S_2

$S_2(tig, \mathbf{C}, L, Q'_A, Q_A, \omega)$, $tig = (\mathbf{P}, params)$, 用于证明-验证的列表 L 和一组损害方 $\mathbf{C} \subsetneq \mathbf{P}$, 初始化 $ext = 0$.

Round 1.

- 对于每个 $P_i \in \mathbf{H}$, 根据协议采样 V_i , 并将 (tig, i, V_i) 提交给敌手 A .

- 对于 P_b , 如果 $ext = 0$, 根据协议设置 X_b, Γ_b .

否则设置 $X_b = Q'_A - \sum_{j \neq b} X_j$, $\Gamma_b = \omega(Q_A + G) - \sum_{j \neq b} \Gamma_j$.

根据协议对 V_b 进行采样, 并将 (tig, i, V_b) 提交给敌手 A .

Round 2.

当从敌手 A 接收到 (tig, j, V_j) 时, 提交 $(tig, i, X_i, \Gamma_i, \rho_i, u_i)$, $P_i \notin \mathbf{C}$ 给敌手 A .

Round 3.

当从敌手 A 接收到 $(tig, j, X_j, \Gamma_j, \rho_j, u_j)$ 时, 检索 $\{\gamma_j\}_{j \in \mathbf{C}}$.

1. 如果 $ext = 0$, 按照协议执行然后提交 $(tig, i, Y_i, \Delta_i, \delta_i)$, $P_i \notin \mathbf{C}$ 给敌手 A .

2. 否则, 验证 $H(\dots) = V_j$ 并根据协议计算 ρ .

- 调用 $Svss(tig, rtig, \rho, \mathbf{C}, L, (X)_j)$, 存储状态为 $\{x_j, y_j\}_{j \neq b}$ 并输出 $L, A, (Y_j)_{j \in P}$.

- 调用 $Smultiadd(tig, rtig, \rho, \mathbf{C}, L, (X)_j, (\Gamma)_j, \perp)$, 存储状态为 $\{x_j, \gamma_j\}_{j \neq b}, (\gamma_b, \delta_b)$ 和输出 L, Γ .

设 $\Delta_b = \omega G - \sum_{i \neq b} x_i \Gamma$, 模拟 S_{log} 消息 $(verify, ZK_{log}, (tig, rtig, \rho, i), (Y_i, G), 1)$,

模拟 S_{log} 消息 $(prove, ZK_{log}, (tig, rtig, \rho, i), (\Delta_i, \Gamma), 1)$.

向敌手 A 提交 $(tig, i, Y_i, \Delta_i, \delta_i)$, $P_i \notin \mathbf{C}$, 将相关的元组添加到 L .

Output:

- 当接收到从 A 发送到 ZK_{log} 的 $(prove, ZK_{log}, (tig, rtig, \rho, i), (Y_i, G); y_i)$,

$(prove, ZK_{log}, (tig, rtig, \rho, i), (\Delta_i, \Gamma); x_i)$. 检查它们是否正确.

如果不是, 则将失败的一方发送到 ZK_{log} , 然后中止.

- 当接收到敌手 A 的 $(tig, j, Y_j, \Delta_j, \delta_j)$ 时, 根据协议执行, 得到 X .

1. $ext = ext + 1$, 如果 $ext = 1$, 回到 round 1, 删除从 round 1 的 S_2 中添加到 L 的元组.

2. 否则, 提取 $\{x_j, y_j\}_{j \notin \mathbf{C}}$, 输出 $L, P_b \in \mathbf{P}$, $\{x_j, y_j\}_{j \neq b}$.

• Pre-signing 模拟器 S_3

$S_3(sid, \mathbf{C}, L, b, y^{\setminus b}, aux)$, 输入 $sid = (tig, rtig, \rho, \mathbf{N}, \mathbf{s}, \mathbf{t}, X, (X)_i, (Y)_i)$, 一组损害方 $\mathbf{C} \subsetneq \mathbf{P}$,

用于证明-验证的列表 L , 以及辅助输入 $aux = R$, 或 $aux = (c^{ctr}, k_b, x_b)$.

Round 1.

Case 1: 如果 $aux = (c^{ctr}, k_b, x_b)$.

- 对于 $P_i \in H$ 按照协议选取 Λ_i .

- 对于 P_b , 设置 $\Lambda_b = k_b G$,

调用 $Smultiadd(sid, \mathbf{C}, L, (W)_j, (\Lambda)_i, aux)$, 存储状态为 $\{k_i, \chi_i\}_{i \neq b}, \chi_b$, 输出 L, R .

存储 $(w_i, \chi_i)_{j \notin \mathbf{C}}$, 输出 R .

Case 2: 如果 $aux = R$, 初始化 $ext = 0$, 做:

如果 $ext = 0$ 根据协议输出 R ,

否则, 设置 $\Lambda_b = R - \sum_{i \neq b} \Lambda_i$, 计算 w_i 并验证 A 是否根据协议执行, 调用 $Smultiadd(sid, \mathbf{C}, L, (W)_j, (\Lambda)_i, aux)$, 存储状态为 $\{k_i, \chi_i\}_{i \neq b}, \eta_0, \eta_1$, 输出 L, R .

1. $ext = ext + 1$, 如果 $ext = 1$, 回到 round 1, 删去 S_3 中 round 1 增加到 L 的元组.

2. 否则, 存储 $(\eta_0, \eta_1), (w_i, \chi_i)_{i \in b}$ 输出 R .

• **MultiAdd** 模拟器 $S_{multiadd}$

$S_{multiadd}(aux, \mathbf{C}, L, (A)_j, (B)_j, aux')$, $aux = sid$ 或者 $(tig, rtig, \rho)$, 一组损害方 $\mathbf{C} \subsetneq \mathbf{P}$, 一个用于证明-验证的表 L 和 $aux' = \perp, R$ 或者 (c, k_b, w_b) .

设置 case 1 为 $aux' = \perp$, case 2 为 $aux' = (c, a_b, b_b)$, case 3 为 $aux' = R$.

Round 1. 每一个参与方 $P_i \in \mathbf{H}$, 按照协议执行并提交 (aux, i, A_i, B_i, G_i) 给敌手 A , 对于 P_b :

Case 1: 设置 $G_b = ENC_b(0)$.

Case 2: 设置 $G_b = a_b \cdot c$.

Case 3: 设置 $G_b = ENC_b(0)$.

模拟 S_{log*} 的消息 $(verify, ZK_{log*}, \dots, (G_b, A_b, G), 1)$, 并将 (aux, b, A_b, B_b, G_b) 提交给敌手 A . 将相关元组添加到 L .

Round 2.

当接收到 $(prove, ZK_{log*}, (aux, i), (G_i, A_i, G); a_i, v_i)$ A 发送到 ZK_{log*} . 检查它们是否正确. 如果不是, 则将失败的一方发送到 ZK_{log*} 并中止.

当从敌手 A 接收到 (aux, j, A_j, B_j, G_j) , 设置 $B = \sum B_i$, 抽取 $\{a_j, b_j\}_{j \in C}$.

- 对于 $P_i \in H$, 按照协议执行并提交 $(aux, i, E_{j,i}, D_{j,i}, \hat{\psi}_{j,i})$ 给 A .

- 对于 P_b , 随机选取 $\{\alpha_{j,b} \leftarrow J\}_{j \neq b}$.

Case 1: 设置 $E_{j,b} = ENC_j(\alpha_{j,b})$, $D_{j,b} = ENC_b(0)$.

Case 2: 设置 $E_{j,b} = ENC_j(\alpha_{j,b})$, $D_{j,b} = (a_j b_b - \alpha_{j,b}) \cdot c$.

Case 3: 设置 $E_{j,b} = ENC_j(\alpha_{j,b})$, $D_{j,b} = ENC_b(0)$.

对于每个 $j \neq b$ 模拟 S_{enc*} 的消息 $(verify, ZK_{enc*}, (aux, i), (E_{j,i}, G_j, D_{j,i}, B_i), 1)$.

提交 $(aux, b, E_{j,b}, D_{j,b})$ 给敌手 A , 将相关元组添加到 L .

Output.

当接收到敌手 A 发送给 ZK_{enc*} 的 $(prove, ZK_{enc*}, \dots, (E_{j,i}, G_j, D_{j,i}, B_i); (b_i, \beta_{i,j}, f_{i,j}, g_{i,j}))$. 检查它们是否正确. 如果不是, 则将失败方发送到 ZK_{enc*} 并中止.

当从敌手 A 接受到 $(aux, j, E_{i,j}, D_{i,j}, \hat{\psi}_{i,j})$, 抽取 $\{\alpha_{j,b}, \beta_{j,b}\}_{j \neq b}$.

Case 1: 设置 $\delta_b = \omega - \sum_{j, i \neq b} a_i b_j - \sum_{j \neq b} (\alpha_{j,b} + \beta_{j,b})$.

Case 2: 设置 $\delta_b = b_b a_b + \sum_{j \neq b} (b_j a_b - \alpha_{j,b}) + (b_b a_j - \beta_{j,b})$.

Case 3: 设置 $\eta_0 = \sum_{j \neq b} a_j$, $\eta_1 = \sum_{j, i \neq b} a_i b_j + \sum_{j \neq b} (\alpha_{j,b} + \beta_{j,b})$.

存储 $\{a_i, b_i\}_{i \neq b}$, δ_b 或者 (η_0, η_1) , 提交 (aux, i, B) 给敌手 A .

• **VSS** 模拟器 S_{vss}

$S_{vss}(aux, \mathbf{C}, L, (X)_j, aux')$, $tig = (P, params)$, 一组恶意方 $\mathbf{C} \subsetneq \mathbf{P}$, 用于证明-验证的表 L , $aux = (tig, rtig, \rho)$ or $aux = sid$.

Round 1.

1. 对于 $P_i \in \mathbf{H}$, 按照协议执行并提交 $(aux, i, A_{i,1}, \dots, A_{i,t-1}, C_{i,j})$ 给敌手 A .
2. 对于 P_b , 随机选取 $y_{b,1}, \dots, y_{b,n} \leftarrow Z_N^*$, 设置 $Y_{b,j} = y_{b,j}G$, $C_{b,j} = ENC_j(y_{b,j}), j = 1, \dots, n$. 获取 $P_j \in \mathbf{C}$ 的标识 d_{j*} , 其中 $|\mathbf{C}| < t$. 然后随机选择 $P_j \notin \mathbf{C}$ 直到 $t-1$ 方.

计算

$$\begin{bmatrix} A_{b,1*} \\ \vdots \\ A_{b,t-1*} \end{bmatrix} = \begin{bmatrix} d_{1*} & \dots & d_{t-1*}^{t-1} \\ \vdots & \ddots & \\ d_{t-1*} & & d_{t-1*}^{t-1} \end{bmatrix}^{-1} \times \begin{bmatrix} Y_{b,1*} - X_b \\ \vdots \\ Y_{b,t-1*} - X_b \end{bmatrix}$$

模拟 S_{log} 消息 $(verify, ZK_{log}, aux, (X_b, G), 1)$.

将 $(aux, i, A_{b,1}, \dots, A_{b,t-1}, C_{b,j})$ 提交给对手 A , 将相关元组添加到 L .

Output.

当接收到 $(prove, ZK_{log}, aux, (X_i, G); x_i)$ 时 A 发送给 ZK_{log} . 检查它们是否正确. 如果不是, 则将失败的一方发送到 ZK_{log} 并中止. 当从敌手 A 接收到 $(aux, j, A_{j,1}, \dots, A_{j,t-1}, C_{j,i})$.

- 设置 $A = \sum X_j$, $Y_b = A + \sum_{k=1}^{t-1} (d_b)^k \sum_{j=1}^n A_{j,k}$, 存储 $\{x_i, y_i\}_{j \neq b}$, 按照协议执行其他数据.

提交 (aux, i, A, Y_i) 给敌手 A . □

下面给出强 RSA 的定义,

定义 3.6.1 (强 RSA 假设) 对于一个 RSA 模 N_0 , 给定 c , 任意多项式时间算法都以可忽略优势输出 m, e , 满足 $m^e = c$.

强 RSA 假设主要在于 Ring-Pedersen 参数在使用中的安全性.

由理想函数到标准的 SM2 签名, 理想门限签名函数的定义满足增强 SM2 签名形式, 下面给出增强 SM2 不可伪造证明. 由 SM2 不可伪造性可得到增强 SM2 签名不可伪造.

定义 3.6.2 (存在不可伪造 (Existential Unforgeability)) 如果一个签名方案 $(KeyGen, sk.Sign, pk.Verify)$ 是存在不可伪造的, 那么对于任意的敌手 A 在多项式时间内 $n \in PPT$, 都有可忽略的概率 $v(\kappa)$ 使得 $Pr[EU(A, n, 1^\kappa) = 1] \leq v(\kappa)$

存在不可伪造实验 $EU(A, n, 1^\kappa)$

1. 生成公私钥对 $(pk, sk) \leftarrow KeyGen(1^\kappa)$, 初始化 $(m_0, \sigma_0) = \emptyset$.
2. 对于 $i = 1, \dots, n(\kappa)$
 - 选择 $M_i \leftarrow A^H(1^\kappa, pk, m_0, \sigma_0, \dots, m_{i-1}, \sigma_{i-1})$
 - 计算 $\sigma_i = sk.Sign(m_i)$, 并记录.
3. A^H 输出 (m, σ) .
4. 输出: $EU(A, n, 1^\kappa) = 1$ 如果 $pk.Verify(m, \sigma) = 1$ 且 $m \notin \{m_1, \dots, m_{n(\kappa)}\}$, 否则为 0.

定理 3.6.2 (增强 SM2 存在不可伪造) 如果 SM2 是存在不可伪造, 那么对于任意的多项式敌手 A 在多项式时间 $t \in PPT$ 和参与方 $n = |P| \in O(1)$, 都有可忽略函数 $v(\kappa)$ 使得 $Pr[EnhSM2(A, n, t, \kappa) = 1] \in v(\kappa)$.

增强 SM2 不可伪造实验 $EnhSM2(A, H, n, t, 1^\kappa)$

1. 生成公私钥对 $(pk, sk) \leftarrow KeyGen(1^\kappa)$, 初始化 $(\vec{R}_0, \vec{m}_0, \vec{\sigma}_0) = \emptyset$.
2. 对于 $i = 1, \dots, t$, 随机 $\vec{R}_i = \{R_{i,j} = k_{i,j}G\}_{j \leq n}$
 - 对于 $j = 1, \dots, n$
 - 选择 $m_{i,j} \leftarrow A^H(G, \vec{R}_0, \vec{m}_0, \vec{\sigma}_0, \dots, \vec{R}_{i-1}, \vec{m}_{i-1}, \vec{\sigma}_{i-1}, m_{i,<j}, \sigma_{i,<j})$
 - 以 $k_{i,j}$ 为随机数, 计算 $\sigma_{i,j} = sk.Sign(m_{i,j})$.
- 设置 $\vec{m}_i = \{m_{i,j}\}_j$, $\vec{\sigma}_i = \{\sigma_{i,j}\}_j$, 并记录.
3. A^H 输出 (m, σ) .
4. 输出: $EnhSM2(A, H, n, t, 1^\kappa) = 1$ 如果 $pk.Verify(m, \sigma) = 1$ 且 $m \notin \{m_{i,j}\}_{i,j}$, 否则为 0.

证明 假设敌手对每个签名做了 $q \in PPT$ 次预言机查询. 那么任意 EnhSM2 敌手以显著优势赢得增强 SM2 不可伪造实验都有 SM2 敌手以相同的优势赢得 SM2 存在不可伪造实验在不超过 $tq^n \log(q) \in PPT$ 次查询下. 定义 SM2 敌手为一个可以黑盒调用 EnhSM2 敌手的算法, 算法如下: 随机选择 q 个消息 $\{M'_i\}_{i=1, \dots, q}$. 随机选择大小为 n 的集合 $I^* \subset \{1, \dots, q\}$, 并对每一个 $m'_{i^*} \in I^*$ 调用 SM2 预言机得到 $(R_{i^*}, H(m'_{i^*}), \sigma_{i^*})$, 做:

1. 将 $\{R_{i^*}\}_{i^* \in I^*}$ 提交给 EnhSM2 敌手.
2. 对于 $i = 1, \dots, q$, 每一次 EnhSM2 敌手预言机查询 m_i , 提交 $H(m'_i)$.
3. 当敌手 EnhSM2 预言机查询 m_{j^*} , 做:
 - 如果 $j^* \neq i^*$, 倒带敌手并重复.
 - 否则提交 σ

因为 $Pr[\forall i^*, i^* = j^*] = \frac{1}{C_q^n \cdot n!} \in O(1/q^n)$, 每一个 j^* 在 $q^n \log(q)$ 次尝试下猜对的概率接近 1. \square

3.7 性能分析

与同类门限 ECDSA [17–19, 22, 25, 49, 50] 相比, 本文使用零知识证明来保证协议的可识别身份中止, 提高了安全目标, 但增加了计算量. 其中 Lindell 的方案 [24] 没有 Pre-signing step, 在表 3.2 中本文的方案 Signing step 包含 Pre-signing step、Keygen step 包含 Paillier-keygen step. 其中 G, N, N^2 分别代表在椭圆曲线点群和 Z_N, Z_N^2 下计算指数幂. k, μ 分别代表椭圆曲线点元素的长度和 Paillier 密文的长度.

与 Canetti 的门限 ECDSA 方案相比. 本文的方案也实现了非交互式, 可识别身份的中止和基于模拟的安全性. 并且 Pre-signing 只有 2 轮且只调用一次 MultiAdd, 而 Canetti 的方案 Pre-signing 3 轮且至少需要调用两次 MultiAdd. 因此, 本文的方案在 Pre-signing 将计算量和通信量减少了近 2/3, 如表 3.3.

本文协议在 Radeon Vega Mobile Gfx 2.30 GHz 的 AMD Ryzen 7 3750H 的配置下使用 Golang 语言实现, 其中 $\kappa_{sm2} = 256, \kappa_{paillier} = 2048$. 在 Paillier-keygen 中使用并行计算来加速协议执行,

表 3.2: 协议的总计算量和通信量

子协议	轮数	计算量	通信量
Paillier-keygen	1	$422N$	$n(2k + 334\mu)$
SM2Ts-keygen	6	$8N + (14+t)G + 4N^2 + n(17N + 8G + 14N^2)$	$9k + 7\mu + n(18k + 13\mu)$
Pre-signing	2	$8N + 4G + 4N^2 + t(17N + 4G + 12N^2)$	$9k + 7\mu + t(17k + 12\mu)$
Signing	1	0	nk
MultiAdd	2	$8N + 3G + 4N^2 + n(16N + 3G + 12N^2)$	$9k + 7\mu + n(17k + 12\mu)$
VSS	1	$tG + n(N + 5G + 2N^2)$	$(t + 2)k + n\mu$

表 3.3: Canetti 等人 [1] 方案预签名阶段的计算量和通信量

Pre-signing	轮	计算量	通信量
三轮方案	3	$n(56N + 12G + 33N^2)$	$n(57k + 54\mu)$
六轮方案	6	$n(49N + 29G + 33N^2)$	$n(67k + 51\mu)$
轻量级	7	$n(38N + 26G + 19N^2)$	$n(67k + 30\mu)$
本文的方案	2	$8N + 4G + 4N^2 + t(17N + 4G + 12N^2)$	$9k + 7\mu + t(17k + 12\mu)$

在 Paillier 加解密中使用中国剩余定理优化, 其他均未优化. Paillier-keygen 和 SM2Ts-keygen 都与参与方集合的大小 n 相关, Pre-signing 只与参与方门限 t 相关, 这里取两者相等. 本文强调各方是由单线程模拟的, 在真实的分布式环境中, 每一方只需要 (当前运行时间/门限值). 且适当优化可以提高更多效率.

表 3.4: 实际运行时间

门限值/秒	Paillier-keygen	SM2Ts-keygen	Pre-signing
2	11.70	2.92	2.89
3	19.13	7.61	7.45
4	28.16	14.75	14.25
5	36.52	24.10	23.13
6	45.06	35.70	34.50
7	56.29	49.61	47.70
8	62.98	65.85	63.41

4 总结与展望

门限签名具有分布均匀, 容错率高的特点. 本文实现了多方门限 SM2 签名, 可应用于分布式密钥管理、智能合约、数字货币 [51–53].

本文协议可以在恶意多数情况下实现可识别身份中止的安全性, 并且具有非常低的耦合性, 适合进一步扩展和维护. 因此它可以用作加密货币中多重签名的替代方案. 此外, 方案还增加了非交互属性, 具体来说, 本协议可以达到以下的目标:

1. 在多个参与方的基础上实现门限 SM2 签名, 使用可验证的秘密共享达到任意门限, 使用 Paillier 加密完成加法共享, 保护各个参与方的私钥安全. 将每一个计算步骤模块化, 添加非交互性以提高计算效率.

2. 实现可识别身份中止的安全目标, 使用零知识证明检测恶意参与方, 并设置密钥更新和参与方更新策略, 在基于模拟下证明安全.

通俗的说, 只要有合适的技术实现两个数的乘积变成一个加法共享, 都可以用来完成 SM2 或者 ECDSA 的门限签名, 并且根据使用场景的不同平衡通信量和计算量, 预处理阶段或者是否存在可信的密钥分配中心. 安全目标的不同决定是诚实多数、基于恶意参与方、基于模拟或者通用可组合框架下安全的可识别身份中止. 更广泛的说, 门限签名是安全多方计算 (Secure Multi-party Computation, MPC, SMPC) [54] 的一种应用实例, 并在全同态加密 (Fully Homomorphic Encryption, FHE) 的发展下有更简洁的解决方式, 但效率有一定的损失, 亦可发展为基于格 (Lattice) [55–57] 的抗量子签名. 对门限 SM2 签名的研究有助于我们熟悉密码原语的使用 [44, 45], 其模块化构建方法也对未来应用型的密码方案提供参考 [47, 51, 52, 58].

下一步, 我们将深化学习可证明安全理论, 完善多方 SM2 签名的证明过程, 寻找抗量子的分布式签名 [58, 59] 和零知识证明, 保证我们的分布式签名方案在量子世界的安全性.

参考文献

- [1] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. Uc non-interactive, proactive, threshold ecdsa with identifiable aborts. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, CCS '20*, pages 1769–1787, 2020.
- [2] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, 1949.
- [3] Ralph C. Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978.
- [4] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
- [5] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
- [6] Yvo Desmedt. Society and group oriented cryptography: a new concept. In *Advances in Cryptology, CRYPTO '87*, pages 120–127, 1988.
- [7] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *Advances in Cryptology, CRYPTO '89*, pages 307–315, 1990.
- [8] 陈建华, 祝跃飞, 叶顶峰, 胡磊, 裴定一, 彭国华, 张亚娟, 张振峰. *GB/T 32918.2-2016*, chapter 信息安全技术 SM2 椭圆曲线公钥密码算法第 2 部分: 数字签名算法 [S], pages 1–10. 中华人民共和国国家质量监督检验检疫总局、中国国家标准化管理委员会, 北京, 2016.
- [9] ISO/IEC. *ISO/IEC 14888-3:2018*, chapter IT security techniques - Digital signatures with appendix - Part 3: Discrete logarithm based mechanisms, pages 39–41. International Organization for Standardization/International Electrotechnical Commission, Switzerland, 2018.
- [10] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06*, pages 390–399, 2006.
- [11] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy (SP), SP '19*, pages 1084–1101, 2019.
- [12] David Wagner. A generalized birthday problem. In *Advances in Cryptology, CRYPTO '02*,

- pages 288–304, 2002.
- [13] Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. Musig-dn: Schnorr multi-signatures with verifiably deterministic nonces. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, pages 1717–1731, 2020.
 - [14] Jonas Nick, Tim Ruffing, and Yannick Seurin. Musig2: Simple two-round schnorr multi-signatures. In *Advances in Cryptology*, CRYPTO '21, pages 189–221, 2021.
 - [15] Handan Kılınç Alper and Jeffrey Burdges. Two-round trip schnorr multi-signatures via de-linearized witnesses. In *Advances in Cryptology*, CRYPTO '21, pages 157–188, 2021.
 - [16] Chelsea Komlo and Ian Goldberg. Frost: Flexible round-optimized schnorr threshold signatures. In *Selected Areas in Cryptography*, SAC '20, pages 34–65, 2021.
 - [17] Philip MacKenzie and Michael K. Reiter. Two-party generation of dsa signatures. In *Advances in Cryptology*, CRYPTO '01, pages 137–154, 2001.
 - [18] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security. In *Applied Cryptography and Network Security*, ACNS '16, pages 156–174, 2016.
 - [19] Yehuda Lindell. Fast secure two-party ecdsa signing. *Journal of Cryptology*, 34(4):1–38, 2021.
 - [20] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Secure two-party threshold ecdsa from ecdsa assumptions. In *2018 IEEE Symposium on Security and Privacy (SP)*, S&P '18, pages 980–997, 2018.
 - [21] Wang Jing, Wu Libing, Min Luo, and Debiao He. Secure and efficient two-party ecdsa signature scheme. *Journal on Communications*, 42(2):12–25, 2021.
 - [22] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ecdsa with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 1179–1194, 2018.
 - [23] Ivan Damgård, Marcel Keller, Enrique Larraia, Christian Miles, and Nigel P. Smart. Implementing aes via an actively/covertly secure dishonest-majority mpc protocol. In *Security and Cryptography for Networks*, SCN '12, pages 241–263, 2012.
 - [24] Yehuda Lindell and Ariel Nof. Fast secure multiparty ecdsa with practical distributed key generation and applications to cryptocurrency custody. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 1837–1854, 2018.
 - [25] Michaella Pettit. Efficient threshold-optimal ecdsa. In *Cryptology and Network Security*, CANS '21, pages 116–135, 2021.
 - [26] Jens Groth and Victor Shoup. On the security of ecdsa with additive key derivation

- and presignatures. In *Advances in Cryptology*, EUROCRYPT '22, pages 365–396, 2022.
- [27] Damiano Abram, Ariel Nof, Claudio Orlandi, Peter Scholl, and Omer Shlomovits. Low-bandwidth threshold ecdsa via pseudorandom correlation generators. In *2022 IEEE Symposium on Security and Privacy (SP)*, SP '22, pages 2554–2572, 2022.
- [28] Harry WH Wong, Jack PK Ma, Hoover HF Yin, and Sherman SM Chow. Real threshold ecdsa. *Network and Distributed System Security*, 1(1):1–18, 2023.
- [29] Yashvanth Kondi, Bernardo Magri, Claudio Orlandi, and Omer Shlomovits. Refresh when you wake up: Proactive threshold wallets with offline devices. In *2021 IEEE Symposium on Security and Privacy (SP)*, SP '21, pages 608–625, 2021.
- [30] Jens Groth and Victor Shoup. Design and analysis of a distributed ecdsa signing service. *Cryptology ePrint Archive, Report 2022/506*, 2022.
- [31] Ming Shang, Yuan Ma, Jingqing Lin, and Jiwu Jing. A threshold scheme for sm2 elliptic curve cryptographic algorithm. *Journal of Cryptologic Research*, 1(2):155–166, 2014.
- [32] Yudi Zhang, Debiao He, Mingwu Zhang, and Kim-Kwang Raymond Choo. A provable-secure and practical two-party distributed signing protocol for sm2 signature algorithm. *Frontiers of Computer Science*, 14(3):1–14, 2020.
- [33] Yang Liu, Qi Feng, Cong Peng, Min Luo, and Debiao He. Asymmetric secure multi-party signing protocol for the identity-based signature scheme in the ieee p1363 standard for public key cryptography. In *Emerging Information Security and Applications*, EISA '22, pages 1–20, 2022.
- [34] 林璟锵, 马原, 荆继武, 王琼霄, 蔡权伟, 王雷. 适用于云计算的基于 SM2 算法的签名及解密方法和系统. *CN104243456A[P]*. 2014.
- [35] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.
- [36] Victor S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology*, CRYPTO '85, pages 417–426, 1986.
- [37] 陈建华, 祝跃飞, 叶顶峰, 胡磊, 裴定一, 彭国华, 张亚娟, 张振峰. *GB/T 32918.1-2016*, chapter 信息安全技术 SM2 椭圆曲线公钥密码算法第 1 部分: 总则 [S], pages 1–42. 中华人民共和国国家质量监督检验检疫总局、中国国家标准化管理委员会, 北京, 2016.
- [38] Helmut Hasse. 47. *Zur Theorie der abstrakten elliptischen Funktionenkörper. I. Die Struktur der Gruppe der Divisorenklassen endlicher Ordnung*, pages 223–230. De Gruyter, Berlin, New York, 1975.
- [39] Neal Koblitz, Alfred Menezes, and Scott Vanstone. The state of elliptic curve cryptography. *Designs, Codes and Cryptography*, 19(20):173–193, 2000.
- [40] Ralph C. Merkle. One way hash functions and des. In *Advances in Cryptology*, CRYPTO'

- 89, pages 428–446, 1990.
- [41] Ivan Bjerre Damgård. A design principle for hash functions. In *Advances in Cryptology, CRYPTO' 89*, pages 416–427, 1990.
 - [42] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology, EUROCRYPT '99*, pages 223–238, 1999.
 - [43] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology, CRYPTO '94*, pages 174–187, 1994.
 - [44] Ran Cohen, Iftach Haitner, Eran Omri, and Lior Rotem. From fairness to full security in multiparty computation. *Journal of Cryptology*, 35(1):1–70, 2021.
 - [45] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*, pages 100–112. Cambridge university press, Cambridge, 2009.
 - [46] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
 - [47] Xiao Li and Mingxing He. A protocol of member-join in a secret sharing scheme. In *Information Security Practice and Experience, ISPEC '06*, pages 134–141, 2006.
 - [48] Jia Yu, Fanyu Kong, Rong Hao, and Xuliang Li. How to publicly verifiably expand a member without changing old shares in a secret sharing scheme. In *Intelligence and Security Informatics, ISI '08*, pages 138–148, 2008.
 - [49] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Two-party ecdsa from hash proof systems and efficient instantiations. In *Advances in Cryptology, CRYPTO '19*, pages 191–221, 2019.
 - [50] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Threshold ecdsa from ecdsa assumptions: The multiparty case. In *2019 IEEE Symposium on Security and Privacy (SP), S&P '19*, pages 1051–1066, 2019.
 - [51] Chenyu Wang, Ding Wang, Yi Tu, Guoai Xu, and Huaxiong Wang. Understanding node capture attacks in user authentication schemes for wireless sensor networks. *IEEE Transactions on Dependable and Secure Computing*, 19(1):507–523, 2022.
 - [52] Shuming Qiu, Ding Wang, Guoai Xu, and Saru Kumari. Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for mobile lightweight devices. *IEEE Transactions on Dependable and Secure Computing*, 19(2):1338–1351, 2022.
 - [53] Zengpeng Li, Ding Wang, and Eduardo Morais. Quantum-safe round-optimal password authentication for mobile devices. *IEEE Transactions on Dependable and Secure Computing*, 19(3):1885–1899, 2022.

- [54] Constantin Blokh, Nikolaos Makriyannis, and Udi Peled. Efficient asymmetric threshold ecdsa for mpc-based cold storage. *Cryptology ePrint Archive, Report 2022/1296*, 2022.
- [55] Shweta Agrawal, Damien Stehle, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. *Cryptology ePrint Archive, Report 2022/634*, 2022.
- [56] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. Musig-l: Lattice-based multi-signature with single-round online phase. In *Advances in Cryptology, CRYPTO '22*, pages 276–305, 2022.
- [57] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *Journal of Cryptology*, 35(2):1–14, 2022.
- [58] Qingxuan Wang, Ding Wang, Chi Cheng, and Debiao He. Quantum2fa: Efficient quantum-resistance two-factor authentication scheme for mobile devices. *IEEE Transactions on Dependable and Secure Computing*, 20(1):193–208, 2021.
- [59] Huiqiang Liang and Jianhua Chen. Non-interactive sm2 threshold signature scheme with identifiable abort. *Frontiers of Computer Science*, 18(1):1–14, 2024.

5 附录

零知识证明实例

- **Ring-Pedersen 参数** ZK_{prm} [1]

附加输入 (N, \hat{s}, \hat{t}) .

证明者有秘密值 ν , 向验证者证明 $\hat{s} = \hat{t}^\nu \bmod N$.

1. 证明者随机选取 $\{a_i \leftarrow Z_{\phi(N)}\}_{i \in [m]}$, 计算 $A_i = \hat{t}^{a_i} \bmod N$, 向验证者发送 A_i .
2. 验证者回复 $\{e_i \leftarrow \{0, 1\}\}_{i \in [m]}$.
3. 证明者计算 $\{z_i = a_i + e_i \nu \bmod \phi(N)\}_{i \in [m]}$, 向验证者发送 z_i .

验证: 接收验证当且仅当对任意的 $i \in [m]$, 满足 $\hat{t}^{z_i} = A_i \hat{s}^{e_i} \bmod N$.

- **Paillier-Blum 模** ZK_{mod}

公共输入 N .

证明者有秘密值 p, q , 向验证者证明 $N = pq$, $\gcd(N, \phi(N)) = 1$, $p, q = 3 \bmod 4$.

1. 证明者随机选择雅克比符号为 -1 的 $w \leftarrow Z_N$ 并发送给验证者.
 2. 验证者发送 $\{y_i \leftarrow Z_N\}_{i \in [m]}$
 3. 对所有的 $i \in [m]$, 计算 $x_i = \sqrt[4]{v_i} \bmod N$, $z_i = y_i^{N-1 \bmod \phi(N)}$.
- 对于所有的 $a_i, b_i \leftarrow \{0, 1\}$, $v_i = (-1)^{a_i} w^{b_i} y_i$.

向验证者发送 $\{(x_i, a_i, b_i), z_i\}_{i \in [m]}$.

验证: 接受验证当且仅当:

- N 是奇合成数.
- 对所有的 $i \in [m]$, 满足 $z_i^N = y_i \bmod N$.
- 对所有的 $i \in [m]$, 满足 $x_i^4 = (-1)^{a_i} w^{b_i} y_i \bmod N$ and $a_i, b_i \leftarrow \{0, 1\}$.

- **无小因子** ZK_{fac} .

附加输入 Ring-Pederson 参数 $\hat{s}, \hat{t} \in Z_{\hat{N}}^*$ 和一个小安全素数 \bar{N} .

公共输入一个 RSA 模 N_0 .

证明者有秘密值 p, q , 向验证者证明 $p, q < \pm 2^l \sqrt{N_0}$.

1. 证明者随机选取

$$\begin{cases} \alpha, \beta \leftarrow \pm 2^{l+\epsilon} \sqrt{N_0}, \mu, \nu \leftarrow \pm 2^l \bar{N} \\ \omega \leftarrow \pm 2^l N_0 \hat{N}, r \leftarrow \pm 2^{l+\epsilon} N_0 \hat{N}, \\ x, y \leftarrow \pm 2^l \hat{N} \end{cases} \quad (5.1)$$

计算

$$\begin{cases} P = \hat{s}^p \hat{t}^\mu, Q = \hat{s}^q \hat{t}^\nu \bmod \bar{N} \\ D = \hat{s}^\alpha \hat{t}^x, C = \hat{s}^\beta \hat{t}^y \bmod \bar{N} \\ T = Q^\alpha \hat{t}^r \bmod \hat{N} \end{cases} \quad (5.2)$$

向验证者发送 (P, Q, D, C, T, ω) .

2. 验证者回复 $e \leftarrow \pm q$.

3. 证明者设置 $\hat{\omega} = \omega - \nu p$, 向验证者发送 (z_1, z_2, m_1, m_2, v) , 其中

$$\begin{cases} z_1 = \alpha + ep, z_2 = \beta + eq, \\ m_1 = x + e\mu, m_2 = y + e\nu \\ v = r + e\hat{\omega} \end{cases} \quad (5.3)$$

验证: 验证者设置 $R = \hat{s}^{N_0} \hat{t}^\omega$, 并检查

$$\begin{cases} \hat{s}^{z_1} \hat{t}^{m_1} = DP^e \bmod \bar{N} \\ \hat{s}^{z_2} \hat{t}^{m_2} = CQ^e \bmod \bar{N} \\ Q^{z_1} \hat{t}^v = TR^e \bmod \bar{N} \end{cases} \quad (5.4)$$

范围检查: $z_1, z_2 \in \pm\sqrt{N_0}2^{l+\epsilon}$, 确保 $p, q > 2^l$ (假设 $2^{2l+\epsilon} \approx \sqrt{N_0}$).

• 固定范围的 Paillier 加密 ZK_{enc}

附加输入 Ring-Pederson 参数 $\hat{s}, \hat{t} \in Z_{\bar{N}}^*$ 和一个大安全素数 \bar{N} .

公有输入 (N_0, E)

证明者有秘密值 (x, ρ) , 向验证者证明 $x \in \pm 2^l$, $E = (1 + N_0)^x \rho^{N_0} \bmod N_0^2$.

1. 证明者随机选取

$$\begin{cases} \alpha \leftarrow \pm 2^{l+\epsilon}, \mu \leftarrow \pm 2^l \bar{N} \\ r \leftarrow Z_N^*, \gamma \leftarrow \pm 2^{l+\epsilon} \bar{N} \end{cases} \quad (5.5)$$

计算

$$\begin{cases} A = \hat{s}^x \hat{t}^\mu, B = \hat{s}^\alpha \hat{t}^\gamma \bmod \bar{N} \\ C = (1 + N_0)^\alpha r^{N_0} \bmod N_0^2 \end{cases} \quad (5.6)$$

向验证者发送 (A, B, C) .

2. 验证者回复 $e \leftarrow \pm n$.

3. 证明者向验证者发送 (z_1, z_2, z_3) , 其中

$$\begin{cases} z_1 = \alpha + ex, z_2 = r\rho^e \bmod N_0 \\ z_3 = \gamma + e\mu \end{cases} \quad (5.7)$$

验证: 验证者检查

$$\begin{cases} (1 + N_0)^{z_1} z_2^{N_0} = CE^e \bmod N_0^2 \\ \hat{s}^{\hat{z}_1} \hat{t}^{z_3} = BA^e \bmod \bar{N} \end{cases} \quad (5.8)$$

范围检查: $z_1 \in \pm 2^{l+\epsilon}$, 确保 $x \in \pm 2^{l+\epsilon}$.

• **Paillier 加密和 Paillier 承诺** ZK_{enc*}

附加输入 Ring-Pederson 参数 $\hat{s}, \hat{t} \in Z_{\bar{N}}^*$ 和一个大安全素数 \bar{N} .

公有输入 (N_0, N_1, D, C, Y, X)

证明者有秘密值 (x, y, ρ_x, ρ_y) , 向验证者证明 $x \in \pm 2^l, y \in \pm 2^{l'}, X = xG, D = C^x(1 + N_0)^y \rho_x^{N_0} \bmod N_0^2, Y = (1 + N_1)^y \rho_y^{N_1} \bmod N_1^2$.

1. 证明者随机选取

$$\begin{cases} \alpha \leftarrow \pm 2^{l \pm \epsilon}, \beta \leftarrow \pm 2^{l' \pm \epsilon} \\ r_x \leftarrow Z_{N_0}^*, r_y \leftarrow Z_{N_1}^* \\ \gamma \leftarrow \pm 2^{l+\epsilon} \bar{N}, m \leftarrow \pm 2^l \bar{N} \\ \delta \leftarrow \pm 2^{l+\epsilon} \bar{N}, \mu \leftarrow \pm 2^l \bar{N} \end{cases} \quad (5.9)$$

计算

$$\begin{cases} A = C^\alpha (1 + N_0)^\beta r_x^{N_0} \bmod N_0^2, B = \alpha G \\ E = (1 + N_1)^\beta r_y^{N_1} \bmod N_1^2 \\ F = \hat{s}^\alpha \hat{t}^\gamma, H = \hat{s}^x \hat{t}^m \bmod \bar{N} \\ I = \hat{s}^\beta \hat{t}^\delta, J = \hat{s}^y \hat{t}^\mu \bmod \bar{N} \end{cases} \quad (5.10)$$

向验证者发送 (A, B, E, F, H, I, J) .

2. 验证者回复 $e \leftarrow \pm q$.

3. 证明者向验证者发送 $(z_1, z_2, z_3, z_4, w_x, w_y)$, 其中

$$\begin{cases} z_1 = \alpha + ex, z_2 = \beta + ey \\ z_3 = \gamma + em, z_4 = \delta + e\mu \\ w_x = r_x \rho_x^e \bmod N_0, w_y = r_y \rho_y^e \bmod N_1 \end{cases} \quad (5.11)$$

验证: 验证者检查

$$\begin{cases} C^{z_1} (1 + N_0)^{z_2} w_x^{N_0} = AD^e \bmod N_0^2 \\ z_1 G = B + eX \\ (1 + N_1)^{z_2} w_y^{N_1} = EY^e \bmod N_1^2 \\ \hat{s}^{z_1} \hat{t}^{z_3} = FH^e \bmod \bar{N}, \hat{s}^{z_2} \hat{t}^{z_4} = IJ^e \bmod \bar{N} \end{cases} \quad (5.12)$$

范围检查: $z_1 \in \pm 2^{l \pm \epsilon}, z_2 \in \pm 2^{l' \pm \epsilon}$, 确保 $x \in \pm 2^{l \pm \epsilon}, y \in \pm 2^{l' \pm \epsilon}$.

• **Dlog 等式** ZK_{log}

公有输入 (\bar{G}, n, A, B) , $n = |\bar{G}|$

证明者有秘密值 x , 向验证者证明 $A = xB$.

1. 证明者随机选取 $\alpha \leftarrow Z_n$, 计算 $X = \alpha B$, 向验证者发送 X .
2. 验证者回复 $e \leftarrow Z_n$.
3. 证明者向验证者发送 $z = \alpha + ex \bmod n$.

验证: 验证者检查 $zB = X + eA$.

• **Dlog 和固定范围的 Paillier 加密** ZK_{log*}

附加输入 Ring-Pederson 参数 $\hat{s}, \hat{t} \in Z_{\bar{N}}^*$ 和一个大安全素数 \bar{N} .

公有输入 $(\bar{G}, n, N_0, C, X, G)$, $n = |\bar{G}|$, G 是 \bar{G} 的生成元.

证明者有秘密值 (x, ρ) , 向验证者证明 $x \in \pm 2^l$, $C = (1 + N_0)^x \rho^{N_0} \bmod N_0^2$, $X = xG \in \bar{G}$.

1. 证明者随机选取

$$\begin{cases} \alpha \leftarrow \pm 2^{l+\epsilon}, \mu \leftarrow \pm 2^l \bar{N} \\ r \leftarrow Z_{\bar{N}}^*, \gamma \leftarrow \pm 2^{l+\epsilon} \bar{N} \end{cases} \quad (5.13)$$

计算

$$\begin{cases} A = \hat{s}^x \hat{t}^\mu, D = \hat{s}^\alpha \hat{t}^\gamma \bmod \bar{N} \\ B = (1 + N_0)^\alpha r^{N_0} \bmod N_0^2 \\ Y = \alpha G \in \bar{G} \end{cases} \quad (5.14)$$

向验证者发送 (A, B, Y, D) .

2. 验证者回复 $e \leftarrow \pm n$.
3. 证明者向验证者发送 (z_1, z_2, z_3) , 其中

$$\begin{cases} z_1 = \alpha + ex, z_2 = r \rho^e \bmod N_0 \\ z_3 = \gamma + e\mu \end{cases} \quad (5.15)$$

验证: 验证者检查

$$\begin{cases} (1 + N_0)^{z_1} z_2^{N_0} = BC^e \bmod N_0^2 \\ z_1 G = Y + eX \in \bar{G} \\ \hat{s}^{z_1} \hat{t}^{z_3} = DA^e \bmod \bar{N} \end{cases} \quad (5.16)$$

范围检查: $z_1 \in \pm 2^{l+\epsilon}$, 确保 $x \in \pm 2^l$.

• **Paillier 乘法** ZK_{mul}

公有输入 (N, A, B, C) .

证明者有秘密值 (x, ρ_x, ρ_y) , 向验证者证明 $(1 + N)^x \rho_x^N = A$, $C = B^x \rho_y^N \bmod N^2$

1. 证明者随机选取 $\alpha, g, f \leftarrow Z_N^*$, 计算

$$\begin{cases} D = B^\alpha g^N \bmod N^2 \\ E = (1 + N)^\alpha f^N \bmod N^2 \end{cases} \quad (5.17)$$

向验证者发送 (D, E) .

2. 验证者回复 $e \leftarrow Z_n$.

3. 证明者向验证者发送 (z_1, z_2, z_3) , 其中

$$\begin{cases} z_1 = \alpha + ex, z_2 = g\rho_x^e \bmod N \\ z_3 = f\rho_y^e \bmod N \end{cases} \quad (5.18)$$

验证: 验证者检查

$$\begin{cases} B^{z_1} z_2^N = DC^e \bmod N^2 \\ (1 + N)^{z_1} z_2^N = EA^e \end{cases} \quad (5.19)$$

零知识证明的花销如下, 为了确保 80 位的统计安全性和 128 位的计算安全性, 我们在 ZK_{mod} 和 ZK_{prm} 中选择 $m = 80$.

prm, mod 满足正确性, 完备性, 诚实验证者零知识.

$mul, fac, enc, enc*, log, log*$ 满足正确性, 特殊完备性, 诚实验证者零知识.

表 5.1: 零知识证明的计算和通信花销

零知识证明	计算量 (证明者)	计算量 (验证者)	通信量
prm	$80N$	$80N$	160μ
mod	$160N$	$80N$	160μ
fac	$10N$	$11N$	$2k+11\mu$
log*	$1G+5N+1N^2$	$2G+3N+2N^2$	$7k+6\mu$
enc	$5N+1N^2$	$3N+2N^2$	$6k+6\mu$
enc*	$1G+10N+3N^2$	$2G+6N+5N^2$	$17k+12\mu$
log	$1G$	$2G$	$2k$
mul	$2N^2$	$3N+4N^2$	$k+2\mu$

致 谢

山水一程,三生有幸,在武大的三年是我人生中最快乐的时光.校内郁郁葱葱的树木、校外的东湖山水、还有永远向东流去的长江水,都令我感到非常的激动和欢乐.除此之外,如果在理学院学习,那么从哪个方向过来都是上坡,这一路的风景会提醒着你,学无止境呀,要向险远之地,真理的高点前进.

借用我导师说过的话:“年轻人要珍惜自己的时间”,我现在仍深受启发.陈建华老师不辞辛劳的给我们上课,告诉我们要认真学习,并且对我们总是包容和鼓励.在某一个知识点犯错了或者不清楚,会说现在还不熟练,多练习练习就好了.也是这种宽容的心态,促使我们不畏困难和敢于纠错,面对自己或者他人的问题,进行鼓励和帮助.陈老师对我们总是温暖且支持的.

在这里也要感谢我的师兄杜浩瑞、彭聪、师姐范青、陈玉磊、同门刘紫瑶、杨均博.没有他们的关心和支持,我的硕士生涯不会那么顺利的.特别是杜浩瑞师兄,无论是科研还是生活上都给我们很多的指导和帮助.尤其在科研上,为我们保驾护航,告诉我们应该如何做,如何写,有哪些问题和解决方法.不遗余力,亲力亲为.另外我的前几届的师兄师姐们,也在实验室之外帮助了我很多,不胜感激.

最后再次感谢我的导师,感谢我的父母、亲人和朋友,感谢学院的各位领导们对我们的支持和提供的平台以及美丽的武汉大学.

武汉大学学位论文使用授权协议书

本学位论文作者愿意遵守武汉大学关于保存、使用学位论文的管理办法及规定,即:学校有权保留学位论文的印刷本和电子版,并提供文献检索与阅览服务;学校可以采用影印、缩印、数字化或其它复制手段保存论文;在以教学与科研服务为目的前提下,学校可以在校园网内公布部分及全部内容.

- 1、 在本论文提交当年,同意在校园网内以及中国高等教育文献保障系统 (CALIS) 高校学位论文系统提供查询及前十六页浏览服务.
- 2、 在本论文提交 ☒ 当年 / ☐ 一年 / ☐ 两年 / ☐ 三年 / ☐ 五年以后,同意在校园网内允许读者在线浏览并下载全文,学校可以为存在馆际合作关系的兄弟高校用户提供文献传递服务和交换服务.(保密论文解密后遵守此规定)

论文作者 (签名): 梁慧强

学 号: 2020202010068

学 院: 数学与统计学院

日期: 2023 年 5 月 26 日

