2023年全国大学生信息安全竞赛

作品报告

作品名称：基于后量子隐私增强保护的电动汽车充电方案

电子邮箱：1324794246@qq.com

提交日期：2023.6.14

填写说明

1. 所有参赛项目必须为一个基本完整的设计。作品报告书旨在能够清晰准确地阐述（或图示）该参赛队的参赛项目（或方案）。

2. 作品报告采用A4纸撰写。除标题外，所有内容必需为宋体、小四号字、1.5倍行距。

3. 作品报告中各项目说明文字部分仅供参考，作品报告书撰写完毕后，请删除所有说明文字。(本页不删除)

4. 作品报告模板里已经列的内容仅供参考，作者可以在此基础上增加内容或对文档结构进行微调。

5. 为保证网评的公平、公正，作品报告中应避免出现作者所在学校、院系和指导教师等泄露身份的信息。

目录

摘　要

电动汽车（EV）在全球范围内的快速发展促进了私人和公共充电设备（EVSE）的技术发展，为了适应快速充电网络需求，EVSE 需要与云服务、EV 及其电池管理系统（BMS）进行通信。EVSE 连接的复杂性造成了巨大的网络安全问题。网络攻击者可以通过使用不安全的电池系统作为未授权的接口，入侵充电设备、滥用电网资源、盗取充电桩终端控制权从而获取用户隐私数据等。目前业界已有许多通过 EVSE 远程连接窃取账户凭证或中断充电的网络攻击案例，即使是 2022 年发布的较为完善的欧洲开放式充电协议 OCPP2.01 在安全认证和隐私保护上仍存在密码算法**资源消耗大、效率低、不抗量子**等缺陷。一个安全且高效的认证协议和隐私保护机制对保证电动汽车充电桩的网络安全尤为重要。

针对上述问题，我们提出了基于国密标准的后量子签名方案***PQSM4***和后量子隐私增强群签名方案***PQSM4-EPID***，以解决电动汽车充电的认证问题和隐私安全问题，为电动汽车的应用提供更加安全可靠的技术保障，实现信息安全自主可控。

本作品的优势和创新性如下：

基于国密算法，安全自主可控

本文提出的隐私增强签名方案属于一类特殊的群签名，能够在保障群签名不可伪造的同时隐藏签名者身份。其后量子安全性仅依赖于国密标准算法$SM4$的安全性，满足自主可控要求。

成员存在的零知识证明结构优化

本方案采用了基于多方安全计算的零知识证明技术，该技术能够将分组密码转化为后量子群签名，然而，对于国密算法的转化将导致计算复杂度与通信复杂度大幅增长(签名大小约 **5280152 字节**，签名时间超过 **1 分钟**)。为了解决该问题，我们结合可恢复零知识证明技术,将基于$Merkle$树的静态累加器构造成员证明的方案优化为基于CDS的 $One\text{-}out\text{-}of\text{-}N\text{-}Proof$证明方案[28]，群签名大小和群员动态管理的时空消耗显著降低(相较[6]的框架降低约 2 倍的群签名大小)。

签名原语 SM4 的与门优化

我们通过使用贪心算法和复合域变换对$SM4$在多方安全计算电路上的与门数量进行了深入优化，将S盒的与门数量由 **246** 降至 **32**，进一步降低了证明规模。

预处理和在线阶段分离

对于我们自主设计的签名算法和群签名算法，我们借鉴了基于分组密码的 KKW 签名系统框架，对签名进行预处理和在线阶段的分离操作，减少了 2 倍的在线签名时间。

关键词：抗量子数字签名；后量子隐私增强方案；国密算法；电动汽车充电系统

第一章 作品概述

1.1 背景分析

1.1.1 电动汽车充电技术面临认证难题

面对燃油汽车尾气排放所造成的污染与石油资源的过度消耗引发的环境问题和能源问题，以及随着智能电网的建设与发展，以移动储能与环保节能为特性的新能源电动汽车成为世界汽车工业发展的未来。当前在世界各国政府的推动下，全世界各大汽车公司都积极参与行动，为电动汽车快速发展做准备；国内汽车生产企业对能够实现节能与减排目标的新能源电动汽车的研发投入了空前的热情。

新事物的发展必然需要迎接各种挑战，电动汽车也不例外。为了满足电动汽车用户不断增长的充电网络需求，充电设备之间需要实现高效联动和高速联网，即公共充电设备（EVSE）和电池管理系统（BMS）必须与各种云服务平台和电动汽车进行通信。但与此同时，电网与公共充电设备间的电源管理和自动化控制也让充电设备的网络服务更加复杂，带来了许多网络安全问题。

近年来，关于电动汽车充电系统的网络安全问题频发。2022 年 2 月 4 日，牛津大学和 Armasuisse S+T 的 Sebastian 等人对于复合充电系统 CCS 提出了 BrokenWire 的攻击[1]，通过无线信号，可在 47 米左右干扰电动汽车和充电设备，达到破坏充电的目的[30]。2023 年 2 月 1 日，Lionel Richard Saposnik 和 Doron Porat 发现了开放充电协议 OCPP 的充电点间连接处理不当和弱身份验证策略两个漏洞，攻击者可以在此基础上接收敏感信息（如驱动程序个人信息、信用卡凭证、充电电固件更新服务器的凭证等）[2]。同样在 2023 年 5 月 6 日，工信部通报的存在问题的 APP（SDK）中有 11 款 APP 涉及新能源汽车充换电运营相关的平台存在违规收集个人信息等行为。

在当前电动汽车充电技术的阶段，一个安全有效的认证机制尤为重要。电动汽车在接入充电站进行充电前需要对电动汽车的身份进行认证以确保电网安全，保证仅向授权的车辆提供充电服务。在认证过程中若不采取有效措施，用户的信息（如身份信息，账单明细等）完全暴露在非授权者或攻击者的前面，而这些数据信息往往与用户

[1] https://www.brokenwire.fail/

[2] https://www.saiflow.com/hijacking-chargers-identifier-to-cause-dos/

的生活习惯、个人爱好以及生命财产息息相关，如果没有可靠的安全认证机制对这些信息加以保护，势必会对用户的隐私安全产生很大的威胁。

为解决隐私保护问题，常用的方法有基于公钥基础设施的签名、权威机构颁发假名等，但是，基于公钥基础设施的签名存在证书管理的问题，这给中心带来了巨大的成本；依赖于权威机构来颁发假名，用户的真实身份受到权威机构的控制，从而导致过度集中化，对权威的攻击可能会泄露用户隐私。此外，权威机构需要存储所有车辆的假名，产生巨大的开销。因此，设计一种不依赖单一权威的、高效的、条件隐私保护的隐私保护方案是非常有必要的。

但我国在电动汽车充电领域中的密码算法仍处于技术空白阶段。目前发行的关于电动汽车充电系统信息安全的国家标准GB/T 41578-2022中，没有明确对于保障硬件安全、软件安全、数据安全、通信安全应该采用的密码算法标准。反观国外，已经提出了诸如以$SHA256$或$SHA384$作为密码算法的通信安全标准，以$RSA\text{-}PSS$或$ECSchnorr$密码算法作为签名算法、以$SHA256$作为标准哈希函数的数据安全标准。于是我们针对电动汽车充电系统的数据安全领域，自主设计了基于国密的抗量子数字签名$PQSM4$，以及后量子隐私增强方案$PQSM4\text{-}EPID$。

1.1.2 国密算法应用现状

近年来，全球政治环境的不确定性增加，贸易争端不断，网络安全问题日益突出，因此密码算法的核心技术独立自主权变得尤为重要。为了避免外国密码算法可能存在的后门和安全隐患，国家密码管理局大力推行国产加密算法，这些算法被广泛应用于社会公共卫生系统、国家金融系统等重要领域。

为了加强信息安全，我国已经颁布了《中华人民共和国密码法》，并发布了《金融和重要领域密码应用与创新发展工作规划》等相关政策。此外，国务院还颁布了《“十四五”数字经济发展规划》，提出了数字经济安全体系概念，要求加强密码应用安全性评估，促进数据加密技术的发展。GB35114 国家强制标准要求采用基于国密算法和部件的数字证书设备身份认证技术。可以预见，国密的地位将会越来越高。

1.1.3 量子计算机的发展与威胁

自 20 世纪后期以来，科学家们在量子计算机的研制、量子计算和量子通信技术等方面取得了令人鼓舞的成就。量子计算机的应用主要包括保密通信、量子算法和快

速搜索三个方面。与经典计算相比，量子计算机具有明显的优越性。

随着量子计算机的快速发展，当前广泛成熟使用的经典密码算法面临着巨大的威胁和挑战。传统的公钥密码体制基于离散对数、整数分解等数学难题，然而，一旦实用化的量子计算机出现，将能够破解这些问题，导致当前的公钥密码体制面临极大的危险，并引发大量的安全问题。

在量子计算机的威胁下，抗量子计算攻击的密码体制成为确保信息安全的重要技术，抗量子计算攻击的密码体制能够在量子计算机的攻击下保持安全，是量子信息时代网络与信息安全的关键保障。目前，研究抗量子密码体制已经成为科学研究领域的重要热点。

抗量子密码体制也称为后量子密码体制($PostQuantum\text{-}cryptography$)[15]。目前，美国和欧洲正在大力对其投入研究，并快速推动其实用化。2015 年 3 月，欧洲电信标准协会$ETSI$成立“量子密码安全工业标准工作组”，主要负责抗量子密码算法的征集、评估和标准制定。同年，欧盟相继启动抗量子密码算法项目$PQCRYPTO$和$SAFECRYPTO$。2016 年 4 月，美国国家标准与技术研究院($NIST$)公布抗量子密码标准化计划。

以往的抗量子密码体制主要包括基于哈希、编码、格以及多变量的密码方案。近年来，密码学界逐渐开始研究基于对称密码原语的抗量子密码体制方案。与其他方案相比，基于对称密码原语的方案具有加解密计算效率高、算法空间复杂度低、安全性不基于任何代数结构以及困难问题等优势。因此，我们基于对称密码原语构造出一种抗量子数字签名。

1.2 我们的工作

考虑到电动汽车充电认证场景下缺乏可靠的安全认证机制，会对用户的隐私安全造成极大的威胁，且现有的电动汽车充电系统网络安全问题频发，设计一个不依赖单一权威的、高效的、条件隐私保护的隐私保护方案尤为重要。为了充分保证电动汽车在认证过程中的匿名性和身份认证性，本文针对电动汽车充电认证设计了后量子隐私增强方案，作品的主要技术结构如下图所示：

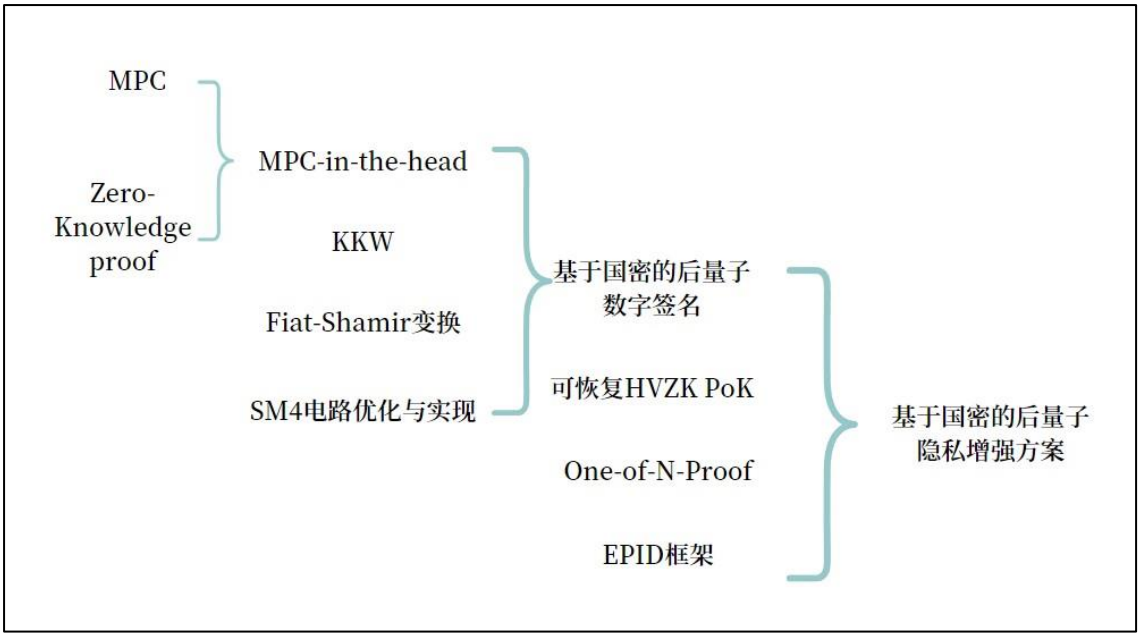

图 1-1 作品主要技术结构图

具体来说，我们的主要工作如下：

- **构建基于国密的后量子数字签名**

我们基于KKW、$MPC\text{-}in\text{-}the\text{-}head$框架等技术，实现了基于$SM4$算法的签名系统$PQSM4$，使得签名方案更具有抗量子攻击的能力，同时保证了签名方案的高效性和可扩展性。

为了实现国密化的签名方案，使用$SM4$对称原语。由于$SM4$加密为非对称的$Feistel$结构，需要使用更多掩码来掩盖每一轮加密后的中间值。考虑到用于加密的明文由各方共享，不需要设置均匀随机的掩码，我们设置各方明文的初始掩码均为 0，以此来降低各方需要共享的消息值。

- **对 SM4 在多方安全计算电路上的与门数量进行优化**

在$MPC\text{-}in\text{-}the\text{-}head$框架下实现$SM4$加密算法要求算法使用电路实现。在执行门电路运算时，相比XOR门运算，AND门运算通常需要更长的时间。国密$SM4$算法中唯一的非线性组件为S盒，这也是制约$SM4$运行效率的重要因素，因此优化S盒与门数量是我们的首要工作。

我们对$SM4$的S盒进行了两次优化。第一次优化采用贪心算法，将每个S盒 3069 个与门降至246个与门。第二次优化我们将$SM4$的S盒映射到已有的AES的S盒上65]，利用仿射变换[12][9][13]以及$RNBP$算法[1]，将单个S盒的与门数量降至 32 个。这

使得$PQSM4$算法在保证量子安全的同时进一步提高运行效率。

- **优化 EPID 成员证明方案**

为了在保障签名者的匿名性的同时提高效率，我们对$Dan\ Boneh$等人在 2018 年提出的基于对称原语（$Hash, PRFs$等）构建的抗量子$EPID$签名方案进行优化，针对原协议中使用$Merkle$树的静态累加器构造成员证明的方案，我们替换为基于$HVZK$的$One\text{-}out\text{-}of\text{-}N\text{-}Proof$方案。$One\text{-}out\text{-}of\text{-}N\text{-}Proof$使用环签名的思想，集成动态累加器和消息签名的功能，提高$PQSM4\text{-}EPID$群签名方案的签名效率，缩短签名大小。

- **预处理和在线阶段分离处理**

为了进一步提高在线签名的签名效率，我们借鉴KKW的签名框架，对签名和群签名都进行了预处理阶段和在线阶段的分离处理，减少了在线阶段约 2 倍的签名时间。

表 1-1 PQSM4 优化效率对比

Scheme	Sign Size/B	Sign Time/ms	Vrfy Time/ms
$SM4$ 未优化	5280152	(超过一分钟)	(超过一分钟)
$SM4$-246 与门优化Ⅰ	435756	12881.00	7708.00
$SM4$-32 与门优化Ⅱ	62524	2114.00	1141.00
$SM4$-32 代码优化Ⅰ	63108	546.38	220.26
$SM4$-32 代码优化Ⅱ	62857	219.56	134.20
$SM4$-32 预处理/在线分离	63074	104.50 (预处理 121.31)	130.11

表 1-2 PQSM4-EPID 实现效率

Group Size	Sign Size/B	Sign Time/ms		Vrfy Time/ms
		Preprocessing	online	
2	79135	139.4	132.7	183.5
4	102352	187.6	177.1	241.8
8	149203	264.8	250.2	356.4
16	242118	432.8	417.0	609.9
32	428588	784.5	779.6	1084.8
64	799956	1457.7	1470.8	2138.9

- **将基于国密的后量子隐私增强方案应用于电动汽车充电认证场景中**

我们构建了基于国密算法的安全多方计算下抗量子攻击的电动汽车充电认证系统。不仅能够保证群中各车辆的无条件匿名性，还能确保群中其他车辆不能伪造真实签名车辆的签名，综合了匿名性和认证性的安全目标，在保护电动汽车隐私的同时实现身份认证。

1.3 应用前景

基于后量子隐私增强保护的电动汽车充电方案有着广阔的应用前景。

1.推动电动汽车行业发展：建立电动汽车充电桩产品信息安全认证体系能够促进电动汽车行业的健康发展。通过确保充电桩产品的信息安全认证，企业能够采取必要的措施保护用户的隐私和数据安全，防止恶意攻击和数据泄露等风险。这项认证不仅可以提高充电桩的安全性能，还为用户提供可靠和安全的充电服务，从而增加对电动汽车的信任度，推动电动汽车的进一步普及。

2.为政府监管提供依据和参考：在国内，2020 年 11 月 2 日国务院印发《新能源汽车产业规划(2021-2035 年)》，明确提出要实现公共领域用车全面电动化和加快建设汽车强国。电动汽车充电桩产品信息安全认证体系的建立可以为政府监管提供依据和参考，确保充电桩的安全性能符合国家和行业标准。同时政府可以通过政策支持和奖励机制，鼓励企业参与认证，推动充电桩行业的安全发展。

3.促进国际标准统一与交流：截至 2023 年 6 月，国际上依旧未构建一套完善且高效安全的电动汽车充电系统的密码安全标准，国外的充电协议如$OCPP2.01$等都局限于$RSA\text{-}PSS$或$ECSchnorr$等非对称签名算法，存在着签名效率低，不抗量子攻击等缺陷，我国成功建立的电动汽车充电桩产品信息安全认证体系可以为国际标准制定提供宝贵的经验和参考。这将促进国际标准的统一和交流，为全球电动汽车充电桩行业的安全发展做出贡献。

第二章 预备知识

2.1.1 数字签名

数字签名是一种用于确保数字文档的真实性、完整性和不可否认性的技术。数字签名可以用于验证签名者的身份，以及确保被签名的文档在传输和存储过程中没有被篡改。数字签名通常由两个部分组成：签名过程和验证过程。

在签名过程中，签名者使用自己的私钥对文档进行签名，在验证过程中，接收者使用签名者的公钥对签名后的文档进行验证，以确保文档没有被篡改，并且签名者确实是文档的真实签名者,这个过程可以保证文档的真实性、完整性和不可否认性。

群签名是一种特殊的数字签名技术，它允许一个群体中的成员对某个文档进行签名，而不需要透露签名者的身份。群签名技术可以用于保护群体成员的隐私，同时确保签名文档的真实性和完整性。

在群签名中，每个群体成员都有一个私钥和一个公钥。群体管理员拥有一个特殊的私钥，可以用来对群体成员进行管理。当一个群体成员要对某个文档进行签名时，他可以使用自己的私钥和群体公钥进行签名，生成一个群签名。验证群签名时，可以使用群体公钥和签名来验证文档的真实性和完整性，但是无法确定签名者的身份。

群签名算法主要包括创建群、加入群、签名、验证签名等步骤。

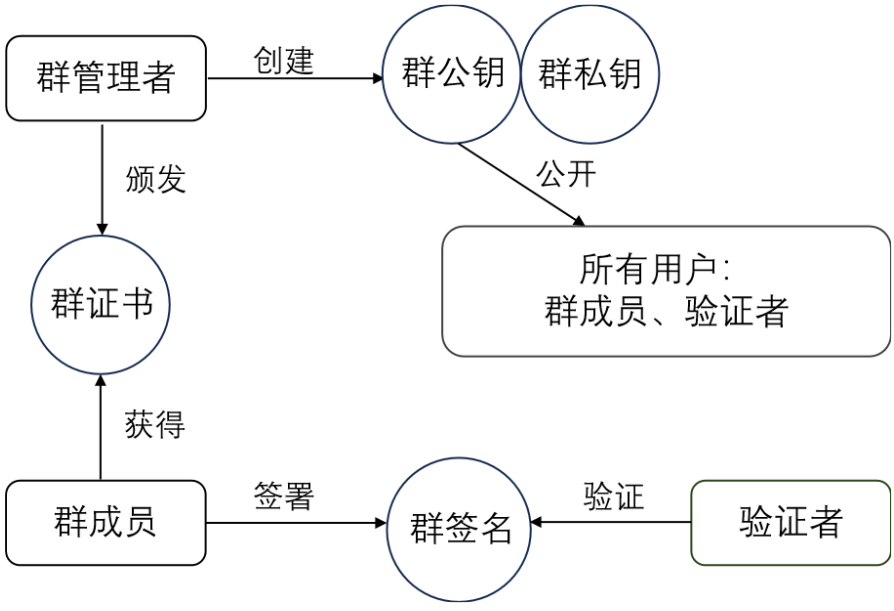

图 2-1 群签名算法

2.1.2 零知识证明

零知识证明协议由参与方证明者和验证者共同执行，允许证明者说服验证者相信某个特定的断言，但是除了该断言本身的正确性以外，不会泄露任何额外的信息（知识）。零知识证明需要满足以下基本性质：

- **完备性**(Completeness)：只要证明者拥有相应的知识，并且诚实执行证明算法，那么就能通过验证者的验证，即证明者有足够大的概率使验证者确信断言的正确性。
- **知识可靠性**（Knowledge Soundness)：如果证明者没有相应的知识，则无法通过验证者的验证，即证明者欺骗验证者的概率可以忽略。
- **零知识性**(Zero-Knowledge)：证明者在交互过程中仅向验证者透露是否拥有相应知识的陈述，不会泄露任何关于知识的信息。

根据证明者和验证者之间的交互形式，零知识证明可被分为交互式零知识证明和非交互式零知识证明。

交互式零知识证明（Interactive zero-knowledge proofs)：证明者和验证者需要进行多次互动，验证者会不断提出问题来挑战证明者，证明者则要不断回应这些挑战，直到验证者被说服。

但交互式零知识证明要求证明者和验证者同时在线并且只对原始的验证者有效，其他的验证者都不能信任这个证明，通信开销极大，为了解决交互式零知识证明面临的问题，非交互式零知识证明应运而生。

非交互式零知识证明（Non-interactive zero-kowledge proofs)：证明者只需要在向验证者发送一个证明，任何验证者可以随时对该证明信息进行验证，并且只需要验证一次就能够判定是否选择相信证明者。这种证明方式比交互式证明需要更多的算力来完成。

在我们的应用场景中，非交互式零知识证明(NIZK)允许证明者说服持有电路 C 的验证者来相信证明者知道一个输入(证据)w使得$C(w)=1$。

2.1.3 Σ-协议与 Fiat-Shamir 变换

Σ-协议($\Sigma-protocol$)是一种交互式的零知识证明协议。如果参与方 P 和 V 之间的一个关于关系 R 的协议π满足如下情况，那么就称该协议为一个$\Sigma-protocol$：

π按照如下方式执行：

承诺($commit$)：P 发送消息 a 给 V 。

挑战($challenge$)：V 发送一个随机挑战值 e 给 P。

响应($response$)：P 以消息 z 作为对 V 挑战的回应。

同时，$\Sigma - protocol$满足如下性质：

—(完备性)如果参与方 P 和 V 都是诚实的并且 $y \in L$，那么 $Pr[(P,V)(y) = accept] = 1$，其中 $L = \{y|\exists x\ s.t\ R(y,x) = 1\}$。

—(s-特殊可靠性)对于任意的 y 以及任意$s(s > 2)$个接受通信文本(accepting transcript)的集合 $\{(a, e_i, z_i)\}_{i\in[s]}$，其中对于 $i \neq j$ 有 $e_i \neq e_j$，那么存在一个提取器 E 可以有效地提取出y的一个合法的证据 w。

—(诚实验证者零知识性)存在一个概率多项式时间的模拟器 S 在输入$y \in L$ 以及 e 的情况下，能够输出一个通信文本(a', e, z')与协议的真实通信文本(a, e, z) 具有相同的概率分布。

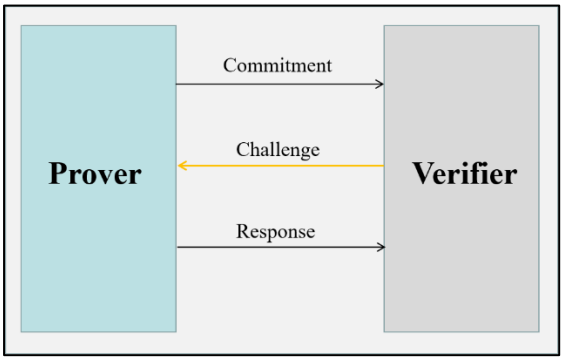

图 2-2 交互式$\Sigma - protocol$

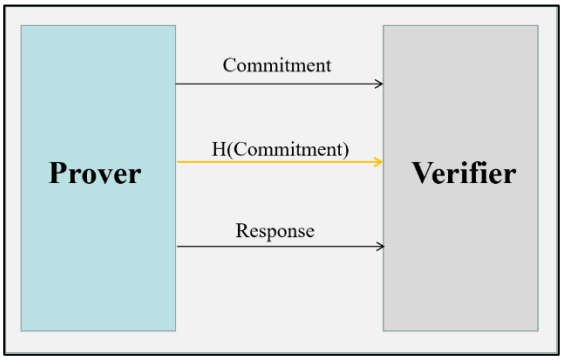

图 2-3 非交互式$\Sigma - protocol$

Fiat-Shamir 变换是一种将交互式证明零知识协议转换为非交互式零知识证明的技术，由 Shamir 和 Fiat 在 1986 年提出[31]。Fiat-Shamir 变换的基本思想是使用一个哈希函数来代替验证者的角色。哈希函数将随机数和其他公共参数作为输入，并输出一个随机的挑战值。证明者使用这个挑战值来计算响应，并将响应和其他公共参数一起发送给验证者。验证者检查响应是否满足验证等式，并接受或拒绝证明。可以利用 Fiat-Shamir 变换将$\Sigma - protocol$转化为非交互式零知识证明。

2.1.4 MPC 协议介绍

MPC全称Multi-PartyComputation,即多方计算,是解决某些问题的协议的总称。这些问题通常涉及多个参与方，每个参与方持有一定的隐私数据，希望不公开这些数据，但又可以利用这些数据计算某一函数，每个参与方获得相应的计算结果（可能相同，可能不同）。我们结合[3]，给出 MPC 的正式定义如下：

考虑一个公共函数$f:(\{0,1\}^k)^n \to \{0,1\}^l$，有$n$个玩家$P_1,\ldots,P_n$，每个玩家都模拟成$PPT$机器。每个玩家$P_i$都保存一个值$x_i \in \{0,1\}^k$，并且希望在保持自己的输入私密的情况下计算出$y = f(x)$，其中$x = (x_1,\cdots,x_n)$。玩家可以使用点对点安全通道$CH_{i,j}$在同步模式下进行通信。这些可以是经典的安全通道（即加密通道）或更强大的通道。如果需要，我们还允许玩家使用广播通道。为了实现他们的目标，玩家共同运行一个n方 MPC 协议Π_f。

Π_f是一个n个玩家的协议，通过下一个消息函数进行指定：有几轮通信，每轮中玩家P_i以$CH_{i,j}$(或广播通道）发送一个消息，该消息计算为P_i的内部状态（他的初始输入x_i和随机带k_i）和P_i在前几轮通信中收到的消息的确定性函数。玩家P_j的视图，表示为$View_{P_j}(x)$，定义为私有输入x_j，随机带k_j和P_j在执行Π_f期间接收到的所有消息的串联。每个通道$CH_{i,j}$都定义了视图一致性关系。例如，在普通通道中，如果在$View_{P_j}(x)$中报告的由P_i传入的消息等于由$View_{P_i}(x)$暗示的传出消息$(i \neq j)$，则**两个视图是一致**的。更强大的通道是通过某些函数ϕ定义的，我们说两个视图是一致的，如果发送方的视图暗示通道的输入x，接收方的视图暗示输入y并包含输出z，使得$z = \phi(x,y)$。

最后，输出y可以从任何视图$View_{P_i}(x)$计算，即存在n个函数$\Pi_{f,1},\cdots,\Pi_{f,n}$，使得对于所有$i \in [n]$，$y = \Pi_{f,i}\left(View_{P_i}(x)\right)$。为了保护隐私，协议$\Pi_f$需要被设计成这样一种方式，即好奇的玩家$P_i$不能从他的视图$View_{P_i}(x)$推断出关于$x_j(j \neq i)$的信息。另外一个安全属性是鲁棒性，即确保作弊的玩家P_i（可能不遵循协议的指示）不能误导诚实的玩家，诚实的玩家仍然计算出正确的输出y。

第三章 系统设计

本章节中，我们在3.2~3.5节中阐述技术原理，3.6节具体阐述如何将技术原理应用在电动汽车充电认证方案中，3.7节为该方案的安全性分析。

本文使用的关键密码算法及其相关符号表示如下表所示。

表 3-1 符号说明

符号	含义
$\boldsymbol{\lambda_\alpha}$	导线 α的掩码值
$[\boldsymbol{\lambda_\alpha}]$	导线 α的掩码份额
$[\boldsymbol{\lambda_n}]$	与门的校正位
$\boldsymbol{z_\alpha}$	导线 α 的真实值
$\boldsymbol{PRF}$	由SM4构建的伪随机函数
$\boldsymbol{PPT}$	概率多项式时间
$\boldsymbol{q}()$	多项式函数
$\boldsymbol{sk_{gp}}$	群管理私钥
$\boldsymbol{pk_{gp}}$	群公钥
$\boldsymbol{X}$	群成员公钥/承诺列表
$\boldsymbol{SigX}$	sk_{gp}对X的签名
$\boldsymbol{KeyRLs}$	密钥撤销列表
$\boldsymbol{Sig\ RLs}$	签名撤销列表

3.1 系统方案

我们采用基于多方安全计算的零知识证明技术，将国密算法转化为后量子群数字签名，然而，对于国密算法的转化将导致计算复杂度与通信复杂度巨额增长，为了解决该问题，我们结合可恢复零知识证明技术，将基于 Merkle 树的静态累加器构造成员证明的方案优化为基于 CDS 的 One out of n Proof 证明方案[28]。此外，我们对 SM4 在多方安全计算电路上的与门数量进行了深入优化，将 S 盒的与门数量由 246 降至 32，进一步降低了证明规模。最终实现我们基于国密的后量子隐私增强方案，并将其部署到电动汽车充电认证环境中。

3.2 作品实现原理

3.2.1 基于$MPC-in-the-head$的零知识证明技术

2007 年，Ishai 等人展示了如何通过任意的 MPC 协议和承诺混合模型(Commitment-hybrid model)，对于任意 NP 关系 R，获得具有最小渐进稳健性误差的零知识证明，该方法被称为“MPC-in-the-head”。接下来我们将会简要回顾其构造，并且介绍我们的后量子数字签名中 MPC-in-the-head 的构造。

$\boldsymbol{MPC-in-the-head}$[3]：假设$\Pi_f$是一个具有完美正确性的 MPC 协议，由于$\Pi_f$的差异，应用“MPC-in-the-head”可以获得略有不同的零知识证明协议，但是结构都是Σ-协议的结构。“MPC-in-the-head”的主要思想为：假设y是零知识协议的公共输入，x是证明者的私有输入，满足$R(y,\ x)=1$。证明者随机取 n 个值x_1、x_2、$\cdots x_n$($x=x_1\oplus x_2\oplus\cdots\oplus x_n$)，将 n 方的输入函数定义为$f_y(x_1\oplus x_2\oplus\cdots\oplus x_n)=R(y,\ x_1\oplus x_2\oplus\cdots\oplus x_n)$，并且根据输入$x_1$、$x_2$、$\cdots x_n$模拟运行协议$\Pi_f$。模拟运行之后，证明者计算对 n 方中每一方视图的承诺$Com\left(View_{P_i}(x_1\oplus x_2\oplus\cdots\oplus x_n)\right), i\in[1,n]$。验证者挑战证明者，打开其中一部分承诺。最后当且仅当所有打开的视图彼此一致，与输出$f_y(x_1\oplus x_2\oplus\cdots\oplus x_n)$相等，验证者证明。

$\boldsymbol{MPC-in-the-head(KKW)}$[7]:带有预处理的“$MPC-in-the-head$”在原始版本的基础上进行了改进，能够以更短的证明时间，实现零知识证明方案的完备性。

其主要思想为：对于证明的命题$C(w)=y$,将零知识证明协议分为两个阶段，即预处理阶段和在线阶段。预处理阶段时，证明者为每一方生成随机掩码，用于隐藏证据w；在线阶段时，证明者使用各方掩码份额和电路的掩码输入，模拟 MPC 协议的执行。验证者将对这两个阶段分别进行挑战。

实现“MPC-in-the-head(KKW)”的主要技术如下：掩码共享方案中，令$[x]$为一比特的掩码份额，满足$x=[x_1]\oplus[x_2]\oplus\cdots\oplus[x_n]$。在基于异或的本方案中，需要知道 n 方所有人的输入，才能恢复出正确的掩码。不妨将函数 C 转化为仅含有与门和异或门的逻辑电路，设 n 方的 MPC 协议为Π，协议由 n 方P_1、P_2、$\cdots P_n$共同执行。设z_α表示电路$C(w)$中导线α的真实值，λ_α表示电路$C(w)$中导线α的掩码值，且$\hat{z}_\alpha \stackrel{\text{def}}{=} z_\alpha\oplus\lambda_\alpha$，其中掩码份额$[\lambda_\alpha]_i$由各方持有。

—预处理阶段

证明者为各方生成随机掩码，各方拥有的值包括：电路输入的掩码$[\lambda_\alpha]_i$、每个与门的输出掩码$[\lambda_\gamma]_i$和辅助位掩码$[\lambda_{\alpha,\beta}]_i$。$[\lambda_\alpha]_i$和$[\lambda_\gamma]_i$可以用带有随机种子$seed_i$的伪随机发生器(PRG)生成。因此，将随机种子$seed_i$而不是掩码份额发送给 Pi，以减小签名大小。由于$\lambda_{\alpha,\beta} \stackrel{\text{def}}{=} \lambda_\alpha \cdot \lambda_\beta$，所以不能仅用种子生成$[\lambda_{\alpha,\beta}]_n$。实际上，$\lambda_{\alpha,\beta}$的 n−1 份额是由 PRG 产生的，而$P_n$的份额由$[\lambda_{\alpha,\beta}]_n = (\lambda_\alpha \cdot \lambda_\beta) \oplus [\lambda_{\alpha,\beta}]_1 \oplus [\lambda_{\alpha,\beta}]_2 \oplus \cdots \oplus [\lambda_{\alpha,\beta}]_n$计算，起到“校正位”的作用。因此，除种子外，参与方P_n需要所有与门生成的$[\lambda_{\alpha,\beta}]_n$。

—在线阶段

每一方P_i各自运行 MPC 协议Π，以拓扑的结构逐个门地计算电路 C。对于输入线为α和β，输出线为γ的逻辑门：

- 如果是 XOR 门，各方计算输出 $\hat{z}_\gamma = \hat{z}_\alpha \oplus \hat{z}_\beta$及掩码份额$[\lambda_\gamma] = [\lambda_\alpha] \oplus [\lambda_\beta]$
- 如果是 AND 门，各方计算并共享份额$[s] := \hat{z}_\alpha[\lambda_\beta] \oplus \hat{z}_\beta[\lambda_\alpha] \oplus [\lambda_{\alpha,\beta}] \oplus [\lambda_\gamma]$，通过各方共享份额重构$s$，各方就能够计算输出$\hat{z}_\gamma = s \oplus \hat{z}_\alpha \hat{z}_\beta$。计算结果满足$\hat{z}_\gamma = z_\gamma \oplus \lambda_\gamma$。

最后，每一方P_i都能计算电路输出$\hat{z}_\gamma$。通过份额重构输出掩码λ_γ，获得 MPC 协议 Π 的输出$\hat{z}_\gamma = z_\gamma \oplus \lambda_\gamma$。

综上可得，在线阶段的运行过程是确定的，通信复杂度和与门数量呈正相关。并且，减少验证一致性和计算与门输出的交互信息，能够有效地减小签名大小，加速签名验证时间。对此，我们通过降低电路中与门数量的方法进行方案优化。

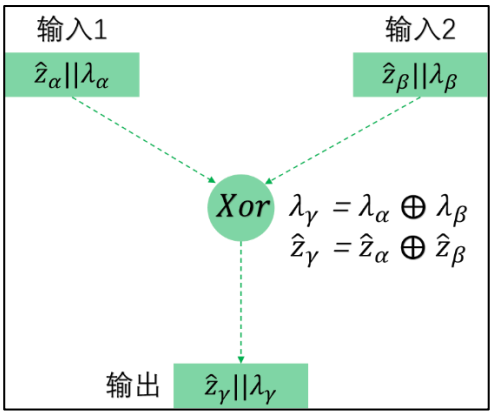

图 3-1 ***XOR*** 门

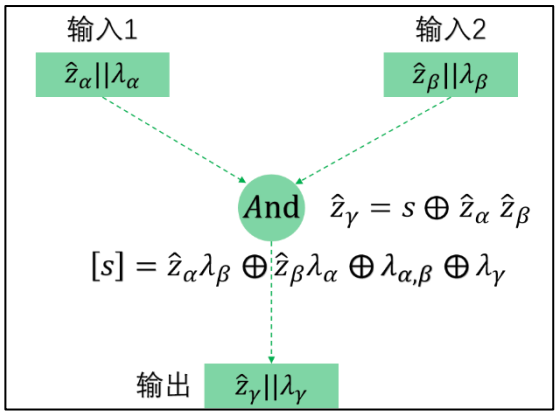

图 3-2 ***AND*** 门

3.2.2 KKW 协议

KKW 是 2018 年由 Katz 等人提出的 n 个参与方的诚实验证者零知识证明方案。

我们给出针对$C(w)=y$的原始 KKW 协议π^F的抽象描述，π^F包括预处理阶段π^F_{pre}和在线阶段π^F_{on}，其中π^F_{pre}利用“cut-and-choose”的技术[7]证明了n个模拟参与方状态生成的随机性和正确性，而π^F_{on}证明了 MPC 协议中每个模拟参与方视图的正确性和一致性。

—预处理阶段 $\pi^F_{pre}(1^\kappa)$

- 第 1 轮——对M个实例的掩码进行承诺

  对于每个实例$j\in[M]$，证明者 P 为关于电路C的 MPC 协议预先生成对应的掩码λ_j。由于λ_j是根据分配给n个模拟参与方的状态$\text{state}_{j,1},\ldots,\text{state}_{j,n}$（即分配给每个模拟参与方的随机种子以及最后一个模拟参与方的修正比特），因此 P 只需要对这些状态进行承诺即可。将这些承诺记作com^F_{pre}并发送给验证者 V。

- 第 2 轮——对预处理阶段的挑战

  V 随机选择一个子集合$\mathcal{C}\subset[M]$发送给 P，满足$|\mathcal{C}|=\tau$。V 要求 P 打开属于集合$[M]\backslash\mathcal{C}$中的实例的承诺，V 验证实例$j\in[M]\backslash\mathcal{C}$中对应掩码$\lambda_j$的随机性和正确性。

- 第 3 轮——对预处理阶段挑战的回应

  P 计算被挑战实例对应承诺的打开（opening）。将这些回应记作resp^F_{pre}并发送给 V。

—在线阶段 $\pi^F_{on}(w,\{state_{j,i}\}_{j\in\mathcal{C},i\in[n]})$

- 第 3 轮——对每个模拟参与方的视图进行承诺

  对每个实例$j\in\mathcal{C}$，P 利用w以及掩码λ_j模拟运行计算$C(w)=y$的 MPC 协议，并且 P 为 MPC 协议中每个模拟参与方的视图进行承诺。将这些承诺记作com^F_{on}并发送给 V。（为了简化表示，掩码后的电路输入值，例如$\widehat{w}'_j=w\oplus\lambda_{j,w'}$，看作是$\text{com}^F_{on}$的一部分。）

- 第 4 轮——对在线阶段的挑战

  V 随机挑选一个集合$\mathcal{P}=\{p_j\}_{j\in\mathcal{C}}$发送给 P，其中$p_j\in[n]$。对于每个实例$j\in\mathcal{C}$，V 要求 P 打开除了第$p_j$个模拟参与方以外的其余模拟参与方的视图的承诺，V 验证该实例中$n-1$个模拟参与方视图的一致性。

- 第 5 轮——对在线阶段挑战的回应

对每个实例$j \in \mathcal{C}$，P 计算被挑战模拟参与方对应的承诺的打开（opening）。将这些回应记作resp_{on}^{F}并发送给 V。

—验证过程

1. 对于属于集合$[M]\backslash\mathcal{C}$的预处理阶段被打开的实例，V 根据resp_{pre}^{F}得到com_{pre}^{F}的部分打开，同时检查这些实例中掩码的随机性和正确性。
2. 对于属于集合$\mathcal{C}$的预处理阶段未被打开的实例，V 根据resp_{on}^{F}模拟计算$C(w) = y$的 MPC 协议，并得到com_{on}^{F}的打开以及com_{pre}^{F}的剩余部分打开。
3. V 检查模拟 MPC 协议的输出的正确性以及com_{pre}^{F}和com_{on}^{F}的一致性。

3.2.3 可恢复诚实验证者零知识证明

令R表示在多项式时间内可高效确定的二元 NP-关系，同时L_R表示由R定义的 NP-语言。也就是说，当且仅当存在一个w使得$(x, w) \in R$时，有$x \in L_R$。在我们考虑的场景下，证明者 P 和验证者 V 可以顺序地执行$q(\kappa)$次零知识证明，其中每次证明被抽象为一次会话，同时将第t次会话记作$\mathrm{session}(t)$，其中$(1 \le t \le q(\kappa))$。在初始会话$\mathrm{session}(1)$结束后，P 和 V 可以高效地恢复后续会话$\mathrm{session}(t)$，其中$(1 < t \le q(\kappa))$。在每次会话$\mathrm{session}(t)$中，P 通过运行协议$\Pi = (P^{(t)}, V^{(t)})$，说服 V 相信他知道关于断言x的一个合法的证据w。令$P^{(t)} = P(x, w, pr_t, ps_t)$表示证明者 P 在会话$\mathrm{session}(t)$中的策略，输入包括公共输入$x$（断言），私有输入$w$（证据），证明者的随机性$pr_t$以及证明者的状态$ps_t$，这里，$ps_t$表示 P 在会话$\mathrm{session}(t-1)$结束后的状态。类似的，$V^{(t)} = V(x, vr_t, vs_t)$表示验证者 V 在会话$\mathrm{session}(t)$中的策略，输入包括公共输入$x$，验证者的随机性$vr_t$以及验证者的状态$vs_t$。

我们考虑的重点是如何将“普通的”诚实验证者零知识证明协议$\Pi' = (P', V')$有效转化为可恢复诚实验证者零知识证明协议$\Pi = (P^{(t)}, V^{(t)})$。接下来我们给出可恢复诚实验证者零知识证明的定义。

定义 1（可恢复诚实验证者零知识证明，Resumable HVZK Proof of Knowledge）如果一个协议$\Pi = \{(P^{(t)}, V^{(t)})\}$满足如下的性质要求，那么称这个协议是对于关系$R$的可恢复诚实验证者零知识证明协议：

- **完整性**：如果证明者和验证者在每个会话$\text{session}(t)(1 \le t \le q(\kappa))$中以输入$x \in L_R$以及$w \in R_x$诚实地执行协议$(P^{(t)}, V^{(t)})$，那么验证者总会接受证明者的证明。
- **可恢复诚实验证者零知识性**：令$\text{view}_V^P(x, w)$表示证明者和验证者在所有会话中的交互脚本，存在一个 PPT 模拟器Sim，满足对于所有的$x \in L_R$和$w \in R_x$，都有$\text{Sim}(x) \approx_c \text{view}_V^P(x, w)$。
- **可恢复知识合理性**：对于每个会话$\text{session}(t)$，存在一个概率的知识提取器$\mathcal{E}$，使得对于任意的$\hat{P}^{(t)}$以及任意的$x \in L_R$，满足：
  - 令$\delta_t(x)$表示输入为x的情况下，验证者在$\text{session}(t)$中与$\hat{P}^{(t)}$交互之后接受证明的概率。如果$\delta_t(x) > \xi_t(x)$，那么在给定$x \in L_R$以及对$\hat{P}^{(t)}$的 oracle 访问权的情况下，提取器$\mathcal{E}$可以在被$O(1/(\delta_t(x) - \xi_t(x)))$限定的期望时间内提取出一个合法的证据$w \in R_x$。

    这里ξ_t表示$\text{session}(t)$的合理性误差。
- **恢复高效性**：对于每个会话$\text{session}(t)$，其中$t > 1$，$(P^{(t)}, V^{(t)})$的计算复杂度及通信复杂度应低于对于关系R的原始诚实验证者零知识证明协议$\Pi' = (P', V')$。即可恢复诚实验证者零知识证明协议的每个恢复会话相比于初始会话应当是高效的。

令C为函数F的电路表示，定义关系R的断言为$x = (C, y)$，对应的证据为w，满足当且仅当$C(w) = y$（即$F(w) = y$）时，有$(x, w) \in R$。

若在初始会话中 P 和 V 就$C(w) = y$（即$F(w) = y$）进行相应的零知识证明，而在后续会话中，仅对$f(w') = y$进行相应的零知识证明，同时 P 向 V 提供w与w'的一致性证明，从而确保可以满足可恢复知识合理性这一安全属性。这里的函数 F 需要满足一个称为可提取分解的特殊性质，其定义如下。

定义 2（可提取分解，Extractable Decomposition）令$F:\{0,1\}^{\kappa} \to \{0,1\}^{\kappa'}$表示一个具有分解$F = f \circ g$的函数。如果对于所有的$x \in \{0,1\}^{\kappa}$，存在一个高效的提取器$\mathcal{E}_D$，满足 $\mathcal{E}_D(g(x)) = x$，那么称这个分解是可提取的。

考虑$F(w) = Enc(w,m)$，其中$\mathrm{Enc}(w,m)$为一个分组密码算法，私钥为w，明文为m。一个典型的分组密码算法包括多轮迭代，同时每一轮的输入包括前一轮的输出以及对应的由主密钥w导出的轮密钥。注意到许多分组密码算法中的密钥生成算法是可逆的，即可以由子密钥反推回对应的主密钥，这就意味着许多分组密码算法满足上述可提取分解的性质。具体而言，给定一个n轮的分组密码算法（将明文m视作算法的公开常数），将前$n-1$轮及密钥生成算法记作g，最后一轮记作f。假定$g(w)$的输出为w'，包括第$n-1$轮的输出以及第n轮的轮密钥，f的输入为w'，对应的输出为最后的密文。显然，w可以从w'中高效地提取出来，因此对该分组密码的上述分解$f \circ g$满足可提取分解的性质。

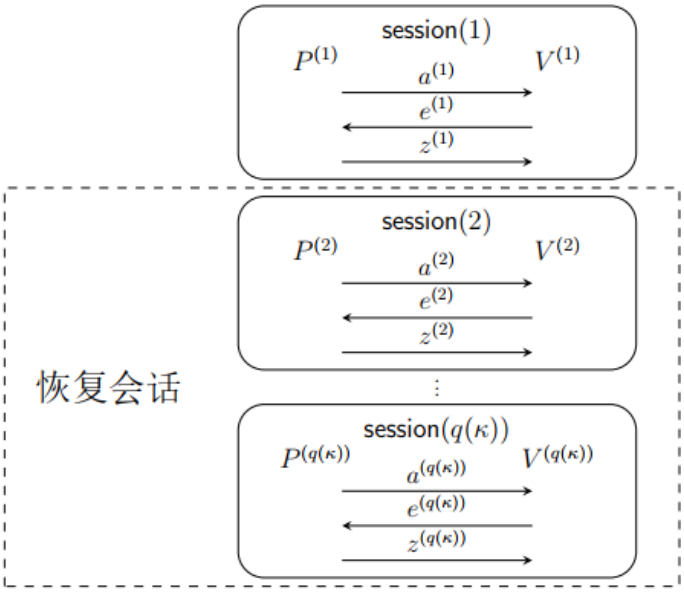

图 3-3 诚实验证者零知识证明的会话恢复

3.3 基于国密的后量子数字签名

本节我们将介绍我们基于多方安全计算的零知识证明技术（$MPC-in-the-head$），应用国密算法 SM4 构造的后量子数字签名方案。

我们构建的基于国密的后量子数字签名方案在遵循 KKW 框架的前提下，引入国密

算法 SM4 作为对称密码原语。我们使用“MPC-in-the-head”的技术，基于安全多方计算协议对 SM4 电路构造零知识证明。

在我们的数字签名体系中，签名者随机生成明文和私钥，并使用私钥对明文加密生成公钥。具体签名过程包括预处理阶段和在线阶段，其中预处理阶段使用签名消息 message 生成电路掩码并提交承诺，再利用“cut-and-choose”的技术证明了 N 个模拟参与方状态生成的随机性和正确性，而在线阶段证明了 MPC 协议中每个模拟参与方视图的正确性和一致性。即检验协议输出是否为挑战 c 对应的公钥 pk。

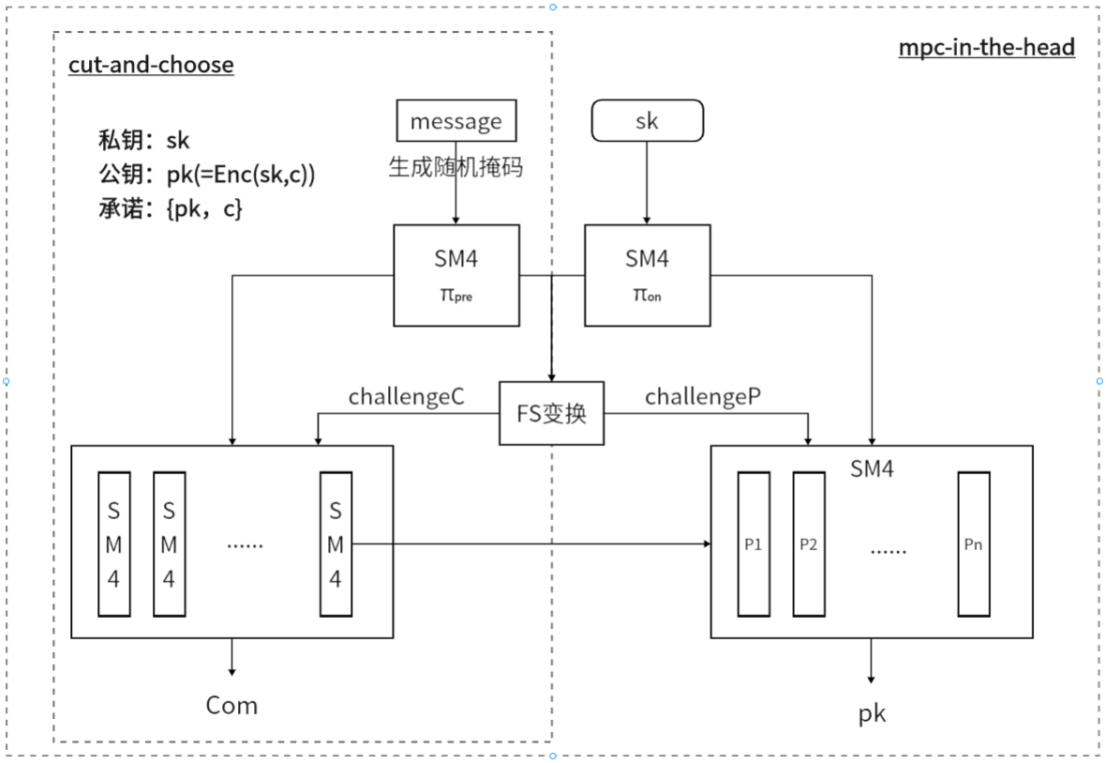

图 3-4 基于国密的后量子数字签名

具体过程如下：

***Round* 1 *Commitment*阶段：**

对于 M 次预处理中的第 j 次，证明者执行如下操作：

- 选择均匀随机的$\kappa - bit$的主种子$seed_j^*$，通过伪随机生成器生成 64 方的种子$seed_{j,1} \cdots seed_{j,64}$。计算电路中所有$AND$门校正位$aux_j \in \{0,1\}^{|C|}$。
- 对于 63 方中的第 i 方来说，$state_{j,i} = seed_{j,i}$，最后一方$state_{j,64} = seed_{j,64}||aux_j$。计算各方的承诺$com_{j,i} \coloneqq Com(state_{j,i})$。
- 证明者模拟在线阶段，通过$\{state_{j,i}\}$模拟电路运行，生成第 i 方需要广播的消

息$msgs_{j,i}$。

- 定义$h_j \coloneqq H(com_{j,1},\cdots,com_{j,n})$，$h'_j \coloneqq H(\{\hat{z}_{j,\alpha}\},msgs_{j,1},\cdots,msgs_{j,n})$
- 计算$h \coloneqq H(h_1,\cdots,h_M)$和$h' \coloneqq H(h'_1,\cdots,h'_M)$，发送$h^* \coloneqq H(h,h')$给验证者。

***Round 2 Challenge*阶段：**

验证者通过随机谕言机，从 h^* 中提取随机挑战 k 次 C 和对应的 P。对于挑战C的预处理阶段，要求打开除挑战P以外 n-1 方的视图，即打开对这 n-1 方的承诺以及协议中在线执行的视图。

***Round 3 Respond*阶段：**

对于不属于挑战 C 的实例j，证明者发送主种子 $seed_j^*$ 和 h'_j 给验证者。对于挑战 C 的实例化，证明者发送$\{state_{j,i}\}_{i\neq p_j}$、$com_{j,p}$、$\{\hat{z}_{j,\alpha}\}$和$msgs_{j,p}$给验证者。

验证阶段：

如果挑战者检验以下式子均成立，则零知识证明成立。

1. 对于挑战 C 中的实例的非挑战方，即 $j \in C_k$，$i \neq P_j$时，验证者使用$\{state_{j,i}\}_{i\neq p_j}$生成的承诺$\{com_{j,i}\}$，并计算$h_j \coloneqq H(com_{j,1},\cdots,com_{j,n})$。
2. 对于非挑战 C 的实例化，即$j \notin C$,执行在线阶段得到h_j和h'_j，计算$h \coloneqq H(h_1,\cdots,h_M)$。
3. 对于挑战 C 实例化，即$j \in C_k$时，验证者通过$\{state_{j,i}\}_{i\neq p_j}$和$msgs_{j,p}$模拟在线阶段，检验电路的输出值。
4. 模拟在线阶段时，可以计算 $h'_j \coloneqq H(\{\hat{z}_{j,\alpha}\},msgs_{j,1},\cdots,msgs_{j,n})$，$h' \coloneqq H(h'_1,\cdots,h'_M)$，检验 $H(h,h')$ 是否与 h^* 一致。

3.4 对 SM4 电路实现优化

在 Picnic2 方案中，电路 C 为 LowMC 原语，由于这 LowMC 的乘法门个数较少，在执行 MPC 协议时，产生的通信开销相比传统密码算法较低。但是，由于 LowMC 中唯一的非线性部件 S 盒较为简单，且尚未经过安全性检验，其是否是一个安全的加密算法尚未可知，而且现在已经研究出了针对 LowMC 的攻击算法[16][17][18]，其安全性进一步受到挑战。我们将 SM4 在多方安全计算电路上实现，并对其进行优化，来构造我们的后量子数字签名方案。

由于我们的方案中，非线性部件中的与门需要各方共享份额完成运算，因此我们

应尽量减少与门数量。而 SM4 含有 32 轮密钥生成与加密过程，每一轮密钥生成和加密过程都需要过 4 个 S 盒，每个 S 盒需要 3069 个与门[20]，这就意味着如果直接应用 SM4，总计需要 785664 个与门，这使得国密化的后量子数字签名开销过大，所以为了减少交互和签名大小，降低时空开销，我们应首先优化 S 盒的与门数量：

我们经过两次算法优化，第一次优化使用贪心算法，对电路表达式中的最大共享因子合并，将每个 S 盒所需要的与门数量降至 246；第二次优化，我们通过复合域变换，将 SM4 的 s 盒映射到 AES 的 S 盒[21]，利用 AES 的 S 盒的电路拆分提取更小的非线性部件电路，将 SM4 每个 S 盒数量优化到 32，显著地降低了时空开销。

将我们的进行深入优化后多方安全计算电路上的 SM4 的与门数量与其它常见电路实现进行对比(见表)，可以发现我们优化后的与门数量少于目前大多数的电路实现方案。

表 3-2 与门数量对比

算法	方案	*AND*	*XOR/XNOR*
SM4	[25]	58	99
	[26]	63	157
	[27]	36	134
	ours	**32**	**81**

具体优化过程如下：

我们从 s 盒的卡诺图中取最小项，递归拆分非门，计算其 s 盒的布尔表达式，得到 s 盒输出关于输入的 8 个表达式[5]。由于篇幅有限，下面仅展示Y_0的表达式:

$$Y_0 = 1+x_1+x_3+x_4+x_0x_1+x_0x_2+x_0x_3+x_0x_4+x_0x_5+x_1x_3+x_1x_5+x_1x_6+x_2x_3+x_2x_6+x_2x_7+x_3x_6+x_4x_7+x_5x_6+x_0x_1x_2+x_0x_1x_4+x_0x_1x_6+x_0x_1x_7+x_0x_2x_4+x_0x_3x_5+x_0x_4x_5+x_0x_4x_7+x_0x_5x_6+x_0x_5x_7+x_1x_2x_4+x_1x_2x_5+x_1x_2x_7+x_1x_3x_5+x_1x_3x_6+x_1x_4x_5+x_1x_5x_6+x_1x_5x_7+x_2x_3x_4+x_2x_3x_6+x_2x_3x_7+x_2x_4x_5+x_2x_4x_6+x_2x_5x_7+x_3x_4x_5+x_3x_5x_7+x_3x_6x_7+x_4x_5x_6+x_5x_6x_7+x_0x_1x_2x_3+x_0x_1x_2x_6+x_0x_1x_3x_4+x_0x_1x_3x_5+x_0x_1x_5x_6+x_0x_1x_5x_7+x_0x_1x_6x_7+x_0x_2x_3x_4+x_0x_2x_4x_6+x_0x_2x_5x_6+x_0x_3x_5x_7+x_0x_5x_6x_7+x_1x_2x_3x_5+x_1x_2x_4x_6+x_1x_2x_4x_7+x_1x_2x_5x_6+x_1x_3x_4x_5+x_1x_3x_4x_7+x_1x_3x_6x_7+x_1x_4x_5x_6+x_1x_4x_6x_7+x_1x_5x_6x_7+x_2x_3x_4x_5+x_2x_3x_4x_7+x_2x_3x_5x_7+x_2x_4x_5x_7+x_3x_4x_5x_6+x_3x_4x_5x_7+x_0x_1x_2x_3x_5+x_0x_1x_2x_3x_6+x_0x_1x_2x_3x_7+x_0x_1x_2x_4x_5+x_0x_1x_2x_4x_7+x_0x_1x_2x_6x_7+x_0x_1x_3x_4x_6+x_0x_1x_3x_5x_6+x_0x_1x_3x_6x_7+x_0x_1x_4x_6x_7+x_0x_1x_5x_6x_7+x_0x_2x_3x_4x_6+x_0x_2x_3x_5x_7+x_0x_2x_3x_6x_7+x_0x_2x_4x_5x_6+x_0x_3x_4x_5x_6+x_0x_3x_4x_6x_7+x_1x_2x_3x_4x_6+x_1x_2x_3x_4x_7+x_1x_2x_3x_5x_6+x_1x_2x_3x_5x_7+x_1x_2x_3x_6x_7+x_1x_2x_4x_5x_6+x_1x_2x_4x_5x_7+x_1x_2x_4x_6x_7+x_1x_3x_4x_5x_6+x_1x_3x_4x_5x_7+x_1x_3x_4x_6x_7+x_2x_3x_4x_5x_6+x_2x_3x_4x_6x_7+x_2x_3x_5x_6x_7+x_0x_1x_2x_3x_4x_7+x_0x_1x_2x_3x_5x_7+x_0x_1x_2x_3x_6x_7+x_0x_1x_2x_4x_5x_6+x_0x_1x_2x_4x_6x_7+x_0x_1x_2x_5x_6x_7+x_0x_1x_3x_4x_5x_6+x_0x_1x_3x_4x_5x_7+x_0x_1x_3x_4x_6x_7+x_0x_1x_3x_5x_6x_7+x_0x_2x_3x_4x_5x_7+x_0x_2x_3x_4x_6x_7+x_1x_2x_3x_4x_5x_7+x_1x_2x_4x_5x_6x_7+x_1x_3x_4x_5x_6x_7+x_0x_1x_2x_3x_4x_5x_6+x_0x_1x_2x_4x_5x_6x_7+x_0x_1x_3x_4x_5x_6x_7$$

图 3-5 Y_0表达式

第一次优化：我们使用贪心算法，按照长度，从小项到大项进行计算，对于每个项，都从长度小 1 的项中寻找最优解，若在小 1 长度的项中找不到，我们则继续往长度小 2 的项中寻找。编写程序计算得到 246 个与门（下面附部分代码和部分结果）。

```
for now in CircleData:
    length = len(now)
    if length == 2:
        if not now in AND_has[0]:
            AND_has[0].append(now)
            sum += 1
            aa += 1
            AND_NEED_SAVE.setdefault(now,0)
            AND_NEED_SAVE[now]+=1
            print(f"\033[31m{now}\033[0m    ——>   {now}",aa)
        else:
            print(f"\033[31m{now}\033[0m    ——>   {now}")
        continue
    now_ = now[:]
    length_ = length
    pre_list = []
    for i in range(length_-2,-1,-1):
        for j in AND_has[i]:  #从大到小查找优化
            if ainb(j,now_):
                pre_list.append(j)
                AND_NEED_SAVE.setdefault(j,0)
                AND_NEED_SAVE[j]+=1
                for letter in j: #删除
                    now_ = now_.replace(letter,"")
    AND_has[length-2].append(now)
    if not now_ == "":
        AND_NEED_SAVE.setdefault(now,0)
        AND_NEED_SAVE[now]+=1
    #计算与门数量
    sum+=len(pre_list)
    if now_ == "":
        sum -= 1
        aaa = ""
    else:
        aa += 1
        aaa = aa
    print(f"\033[31m{','.join(pre_list)}\033[0m {now_}\t——>   ",now,aaa)
print("优化后：",sum,"与门")
```

第一次优化：贪心算法

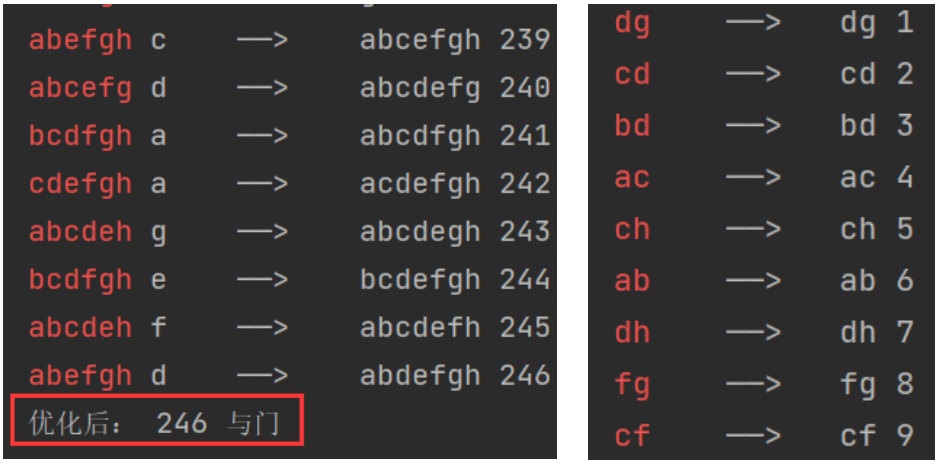

图 3-6 部分结果

得到 246 个与门之后，我们进一步查找资料对其进行优化。

第二次优化：随着 Boyar 等人提出了一种 AES 算法的 s 盒布尔电路的实现方法，且考虑到 SM4 算法和 AES 算法的 s 和构造相似，都是基于优先于 GF(2^8) 上构造，且都包含仿射变换和求逆运算[22]。因此我们可以考虑利用 AES 的 s 盒计算得到 SM4 的 s 盒输出。

对 SM4 算法的布尔电路的优化我们始终以优化 AND 门为主，XOR 门为辅，其 s 盒我们可以表示成仿射变换+非线性变换+仿射变换的形式；而 AES 算法的 s 盒我们可以表示成线性变换+非线性变换+线性变换的形式。我们使用 AES 算法 s 盒的非线性变换，并将非线性变换前后的仿射变换和线性变换合成一个仿射变换，再利用 RNBP 算法，保存和使用优化效果最佳的路线，从而优化 XOR 门的数量。最终我们使用 116 个 AND 门和 84 个 XOR 门构造 SM4 算法 s 盒的布尔电路。

我们可以把 SM4 的 s 盒分成以下三个步骤：顶层仿射变换、非线性变换以及底层仿射变换，其实现如下：

$$S_{sm4}(x) = (M_2B)F(UM_1(x) + UC_1) + M_2T + C_2$$

我们把$UM_1(x) + UC_1$称为顶层仿射变换，F 为非线性变换，$(M_2B)F(x) + M_2T + C_2$为底层仿射变换。

对于顶层仿射变换，输入$x_0, x_1 \dots x_7$输出$y_0, y_1 \dots y_{21}$；对非线性变换 F 输入为$y_0, y_1 \dots y_{21}$，输出为$z_0, z_1 \dots z_{17}$；对于底层仿射变换输入$z_0, z_1 \dots z_{17}$，输出为$y_0, y_1 \dots y_7$。

我们从 s 盒中分离出两个仿射变换，剩下的非线性变换 F 中仅剩 **32 个与门**。

$$
\begin{array}{lll}
y[1] = x[4] \oplus x[7]; & y[11] = x[1] \oplus x[3]; & y[14] = x[4] \oplus y[11]; \\
y[19] = x[0] \oplus x[5]; & y[21] = x[1] \oplus y[19]; & t[1] = x[2] \oplus x[6]; \\
y[12] = x[1] \oplus t[1]; & y[13] = y[14] \oplus y[12]; & y[16] = y[21] \oplus y[13]; \\
y[6] = x[0] \oplus y[16]; & y[7] = y[1] \oplus y[16]; & y[0] = y[11] \oplus y[7]; \\
y[5] = x[6] \oplus y[0]; & y[2] = y[13] \oplus y[5]; & y[8] = x[5] \oplus y[7]; \\
y[3] = y[5] \oplus y[8]; & y[4] = y[12] \oplus y[3]; & y[9] = y[2] \oplus y[4]; \\
y[10] = y[19] \oplus y[8]; & y[15] = y[6] \oplus y[0]; & y[17] = y[16] \oplus y[15]; \\
y[18] = y[7] \oplus y[2]; & y[20] = t[1] \oplus y[15]; & \\
y[0] = y[0] \oplus 1; & y[1] = y[1] \oplus 1; & y[2] = y[2] \oplus 1; \\
y[3] = y[3] \oplus 1; & y[4] = y[4] \oplus 1; & y[5] = y[5] \oplus 1; \\
y[7] = y[7] \oplus 1; & y[10] = y[10] \oplus 1; & y[15] = y[15] \oplus 1; \\
y[17] = y[17] \oplus 1; & y[19] = y[19] \oplus 1; &
\end{array}
$$

图 3-7 顶层仿射变换

$$
\begin{array}{lll}
u[0] = z[1] \oplus z[13]; & u[1] = z[2] \oplus u[0]; & u[2] = z[12] \oplus u[1]; \\
u[3] = z[7] \oplus z[10]; & u[4] = z[5] \oplus u[2]; & u[5] = z[0] \oplus z[16]; \\
u[6] = z[1] \oplus z[3]; & u[7] = z[15] \oplus u[4]; & u[8] = u[5] \oplus u[6]; \\
s[6] = u[7] \oplus u[8]; & u[10] = z[8] \oplus u[3]; & u[11] = z[4] \oplus z[16]; \\
s[7] = u[7] \oplus u[11]; & u[13] = z[11] \oplus u[8]; & u[14] = z[17] \oplus u[13]; \\
u[15] = z[9] \oplus u[4]; & u[16] = z[10] \oplus u[14]; & s[2] = z[4] \oplus u[16]; \\
u[18] = s[7] \oplus u[14]; & s[1] = u[15] \oplus u[18]; & u[20] = u[10] \oplus u[15]; \\
s[3] = z[5] \oplus u[20]; & u[22] = z[6] \oplus u[3]; & u[23] = z[3] \oplus u[22]; \\
s[4] = u[15] \oplus u[23]; & u[25] = z[11] \oplus z[14]; & u[26] = u[10] \oplus u[25]; \\
s[5] = u[1] \oplus u[26]; & u[28] = u[23] \oplus u[25]; & u[29] = u[16] \oplus u[28]; \\
s[0] = z[13] \oplus u[29]; & & \\
s[0] = s[0] \oplus 1; & s[1] = s[1] \oplus 1; & s[3] = s[3] \oplus 1; \\
s[6] = s[6] \oplus 1; & s[7] = s[7] \oplus 1; &
\end{array}
$$

图 3-8 底层仿射变换

$$
\begin{array}{lll}
t[2] = y[12] \cdot y[15]; & t[3] = y[3] \cdot y[6]; & t[4] = t[3] \oplus t[2]; \\
t[5] = y[4] \cdot y[0]; & t[6] = t[5] \oplus t[2]; & t[7] = y[13] \cdot y[16]; \\
t[8] = y[5] \cdot y[1]; & t[9] = t[8] \oplus t[7]; & t[10] = y[2] \cdot y[7]; \\
t[11] = t[10] \oplus t[7]; & t[12] = y[9] \cdot y[11]; & t[13] = y[14] \cdot y[17]; \\
t[14] = t[13] \oplus t[12]; & t[15] = y[8] \cdot y[10]; & t[16] = t[15] \oplus t[12]; \\
t[17] = t[4] \oplus t[14]; & t[18] = t[6] \oplus t[16]; & t[19] = t[9] \oplus t[14]; \\
t[20] = t[11] \oplus t[16]; & t[21] = t[17] \oplus y[20]; & t[22] = t[18] \oplus y[19]; \\
t[23] = t[19] \oplus y[21]; & t[24] = t[20] \oplus y[18]; & t[25] = t[21] \oplus t[22]; \\
t[26] = t[21] \cdot t[23]; & t[27] = t[24] \oplus t[26]; & t[28] = t[25] \cdot t[27]; \\
t[29] = t[28] \oplus t[22]; & t[30] = t[23] \oplus t[24]; & t[31] = t[22] \oplus t[26]; \\
t[32] = t[31] \cdot t[30]; & t[33] = t[32] \oplus t[24]; & t[34] = t[23] \oplus t[33]; \\
t[35] = t[27] \oplus t[33]; & t[36] = t[24] \cdot t[35]; & t[37] = t[36] \oplus t[34]; \\
t[38] = t[27] \oplus t[36]; & t[39] = t[29] \cdot t[38]; & t[40] = t[25] \oplus t[39]; \\
t[41] = t[40] \oplus t[37]; & t[42] = t[29] \oplus t[33]; & t[43] = t[29] \oplus t[40]; \\
t[44] = t[33] \oplus t[37]; & t[45] = t[42] \oplus t[41]; & \\
z[0] = t[44] \cdot y[15]; & z[1] = t[37] \cdot y[6]; & z[2] = t[33] \cdot y[0]; \\
z[3] = t[43] \cdot y[16]; & z[4] = t[40] \cdot y[5]; & z[14] = t[29] \cdot y[2]; \\
z[15] = t[42] \cdot y[9]; & z[16] = t[45] \cdot y[14]; & z[17] = t[41] \cdot y[8];
\end{array}
$$

图 3-9 非线性变换

表 3-3 相关数据的对比

Scheme	Sign size(B)	Sign Time(ms)	Vrfy Time(ms)
SM4 246AND GATE	435756	12881.00	7708.00
SM4 32AND GATE	62524	2114.00	1141.00
LowMC	33040	159.00	106.00
SM4 32AND GATE WITH 20 THREADS	63108	546.38	220.26

我们进一步使用多线程加速。启用 20 个线程进行并行计算，结果得到在平均签名大小为 63108.00bytes 时，平均签名的时间为 546.38ms，平均验证的时间为 220.26ms。这不仅仅提高了国密化后的应用价值和可行性，而且当我们应用到电路中，我们能够得到更大的优化。

3.5 后量子隐私增强方案的优化与构建

本节将介绍我们改进后的后量子隐私增强签名方案，首先介绍 Dan Boneh 等人在[6]的$construction\ 9$中提出的 EPID 抗量子理论框架。

— $\boldsymbol{GAInit(1^{\lambda})}$:

组管理员M运行$S.Keygen(\mathbf{1}^{\boldsymbol{\lambda}})$，获得$(sk_{gp}, pk_{gp})$并输出$pk_{gp}$，$sk_{gp}$保密。

组管理员M运行$Ac.Agen(\mathbf{1}^{\boldsymbol{\lambda}})$，获得公钥$pk_{\wedge}$并将其输出。

— $\boldsymbol{GAJoin_{M,Pi}}\left((sk_{gp}, pk_{gp}, \boldsymbol{X}), pk_{gp}\right)$:

-组管理员M给成员P_i发送挑战c_i。

-P_i选择一个随机秘钥$sk_i \leftarrow_r (0,1)^{\lambda}$并将$t_i^{join} = f(sk_i, c_i)$发送给$M$。

-M 定义$x_i = (t_i^{join}, c_i)$，令$X = X \cup x_i$

令$\Lambda := MerkleTop(X)$作为累加器

产生签名$\sigma_{\wedge} = Sign(sk_{gp}, \Lambda + states)$

然后M构造$wit_{x_i} = AwitCreate(pk_{\wedge}, \Lambda, X, x_i)$（Merkle 树路径）

构造证书$cert_i = (x_i, wit_{x_i})$。

将$(cert_i, \sigma_{\wedge}, \Lambda)$发给$P_i$。

组成员P_i的私钥为sk_i且双方都获得$(cert_i, \sigma_{\wedge}, \Lambda)$

— $\boldsymbol{GAReJoin_{M,Pi}}\left((\boldsymbol{sk_{gp}, pk_{gp}, X, \Lambda, \sigma_{\wedge}}), (\boldsymbol{pk_{gp}, cert_i})\right)$:

-P_i发送 $cert_i$给群管理员 M

-M检查$cert_i$中的签名$(Ac.AVerify(pk_{\wedge}, \Lambda, cert_i))$，失败 return ⊥。

-创建一个新的$wit_{x_i} = AwitCreate(pk_{\wedge}, \Lambda, X, x_i)$

-构造新的证书$cert_i = (x_i, wit_{x_i})$。

后续同$GAJoin$。

— $\boldsymbol{GASign(pk_{gp}, sk_i, cert_i, m, SigRL)}$:

-$Verify(pk_{gp}, \sigma_{\wedge}, \Lambda)$验证累加器合法性。

- $r \leftarrow_r (0,1)^{\lambda} \backslash c_i$

- $t \leftarrow (f(sk_i, r), r)$

- $\pi \leftarrow P(public(\lambda, m, pk_{gp}, t, SigRL, KeyRL), private(sk_i, cert_i), R_1)$

- $sig \leftarrow (t, \pi)$

其中R_1为证明以下关系从而证明$private(sk_i, cert_i)$知识的真实性：

- $t \leftarrow (f(sk_i^*, r), r)$
- $r \neq c_i^*$
- $AVerify\left(pk_{gp}, \Lambda, wit_{x_i}^*, \left(t_i^{join*}, c_i^*\right)\right)$
- $t_i^{join*} = f(sk_i^*, c_i^*)$
- 对于所有的$sig_j \in SigRL \rightarrow t_s ig_j \neq \left(f\left(sk_i^*, r_{sig_j}\right), r_{sig_j}\right)$

— $\boldsymbol{GAVerify(pk_{gp}, m, KeyRL, SigRL, sig)}$：

检查签名$\sigma_\wedge$：$Verify(pk_{gp}, \sigma_\wedge, \Lambda)$

检查协议π：$V\left((\lambda, m, pk_{gp}, t, SigRL, KeyRL, \Lambda), \pi\right)$

对于每一个$sk_j \in KeyRL$，检查 $t \neq (f(sk_j, r), r)$.

检查$sig \notin SigRL$

本方案设计在遵循 EPID 框架的前提下做出了许多改进，我们将自主设计的基于$SM4$的后量子签名算法作为本方案的签名算法原语，并弃用了原方案中基于$Merkle$静态累加器结构的成员证明器AC，替换为基于CDS的$One\ of\ N\ Proof$成员证明方案，同时使用$Resumable\ HVZKPoK$的技术优化了$Proof$大小[28]。我们的成员证明方案集成了动态累加器和消息签名的功能，减少了在一定成员数量范围内的签名大小和成员动态管理的时空消耗，同时减少了额外的公共参数需求(累加器的秘钥)。基于优化后的动态累加器结构，我们删去了原框架中的$ReJoin$函数，并删去了$Verify$协议中对撤销秘钥列表$KeyRL$的检验步骤。

相较于 Dan Boneh 等人提出的原 EPID 框架，本方案更加适用于小范围的物联网应用场景。在同样的密码学原语基础上，我们的方案具有相对更短的证明大小和验证时间、更加简洁的交互过程和更强的成员隐私性。

下面具体介绍本文的方案部分改进过程

3.5.1 成员证明方案优化

Dan Boneh 在[6]的$construction\ 9$中提出的 EPID 抗量子理论方案的成员证明需要依赖对 Merkle 哈希树结构静态累加器的零知识证明，限制了成员更新对累加器的

更新，造成了较大的成员管理消耗，增加了成员管理难度，且其基于哈希的特性也导致签名大小复杂数常量过大[8]。

我们的后量子隐私增强方案在满足安全性定义的前提下，对其静态累加器的结构进行了优化。我们将 Merkle 树的结构替换为线性表存储的动态结构，成员存在证据转换为线性表的索引 (复杂度 O(1)),成员证明使用基于 CDS 的 One-out-of-N Proof 证明方案，在动态累加器的基础上集成了消息签名的功能，从而减少了签名方案在一定成员数量范围内的签名大小和成员动态管理的时空消耗。

在我们的群签名方案中我们使用基于 2.4 节构造的签名方案，构建 one-out-of-N proof 来实现成员证明,具体原理如下:公开考虑N个成员的群组公钥列表{pk1,c1}、{pk2,c2}…，对于成员 t 生成其成员证明，需要以群组公钥列表中的所有公钥作为承诺运行 N 次签名。对于其余成员提前随机好对应挑战，对于自己的承诺，通过 FS 获取挑战 challenge，并将自己的挑战设置为上值的异或值。由于其余承诺提前获取了挑战信息，通过运行模拟器即可成功运行，对于自己的承诺，其挑战基于不可预测的 challenge 生成，故而证明者在不知道任何一个成员的私钥的情况下，无法生成可通过验证的成员证明。这样成员证明基于群组公钥列表的线性表数据结构，能够很好地进行成员动态管理。

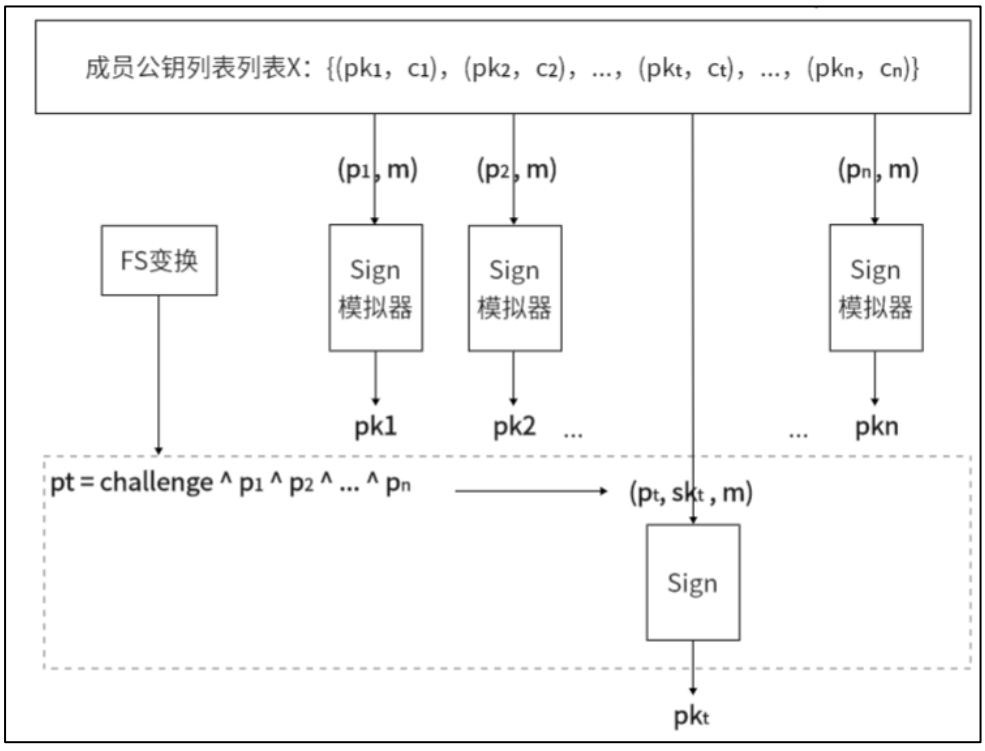

图 3-10 One-Out-of-N 结构图

下表给出了群签名方案构造中常用的 3 种密码累加器构造方案[26][36][19]和我们的累加器构造方案的详细比较。其中 S 代表静态累加器，$\boldsymbol{D}$代表动态累加器，$\boldsymbol{U}$代表通用累加器，$\boldsymbol{w}$为成员证明更新，$\boldsymbol{u}$为非成员证明更新，$\boldsymbol{a}$为添加成员，$\boldsymbol{s}$为删除成员，$|Pk_{acc}|$表示公共参数，$|Proof|$表示成员证据大小。

表 3-4 群签名方案构造比较

实现工具	假设	方案	类型	抗量子	成员动态管理				B	$\lvert Pk_{acc}\rvert$	$\lvert Proof\rvert$
					w	u	a	s			
RSA体制	$S-RSA$	BM	S	×	−	−	−	−	×	$O(1)$	$O(1)$
		BP	S	×	−	−	−	−	×	$O(1)$	$O(1)$
		CL	D	×	−	−	✓	×	×	$O(1)$	$O(1)$
		LLX	DU	×	✓	✓	✓	×	×	$O(1)$	$O(1)$
		BBF	DU	×	✓	✓	✓	✓	×	$O(1)$	$O(1)$
双线性映射	$q-SDH$	NY	D	×	✓	−	×	×	✓	$O(q)$	$O(1)$
		DT	DU	×	✓	✓	×	×	✓	$O(q)$	$O(1)$
		$ATSM$	DU	×	✓	✓	×	×	✓	$O(q)$	$O(1)$
	$q-DEH$	CKS	D	×	✓	−	×	✓	✓	$O(q)$	$O(1)$
Merkle 哈希树	CRH	DB	S	✓	−	−	−	−	×	$O(1)$	$O(logN)$
$one\ out\ of\ N$	对称密码	$\boldsymbol{ours}$	D	✓	✓	−	✓	✓	×	$None$	$O(N)$

除了以上数据，还有 Benoit Libert 等人在 2016 年提出的基于格的群签名累加器[29],其$ProofSize$为$O(logN)$，且能够做到抗量子安全，但是其基于格的密码系统空间占用过大，因此只能够提供理论上的安全，在此不作比较。

3.5.2 成员证明大小优化

虽然$one\ out\ of\ N\ Proof$模式下的累加器可以对成员进行更好的成员动态管理，在一定程度上通过合并签名功能的优势减小通讯复杂度，且基于对称密码的特性使得其原语运行速度快于 Merkle 累加器模式下的原语运行速度。但其成员证明大小复杂度为$O(N)$的劣势不可忽视。故而我们通过使用$resumable\ HVZKPoK$的技术大大降低了其证明大小随着成员数量而增长的趋势。

可以注意到，one out of N 的成员证明本质上是证明者进行 N 次同一电路的零

知识证明，通过可恢复 HVZKPok 技术对 SM4 电路进行提取分解，对于不知道私钥的 n-1 个承诺，证明者只需要运行最后 4 轮的 SM4 电路的模拟器即可，减小了 7/8 因为成员增长而增加的签名大小。

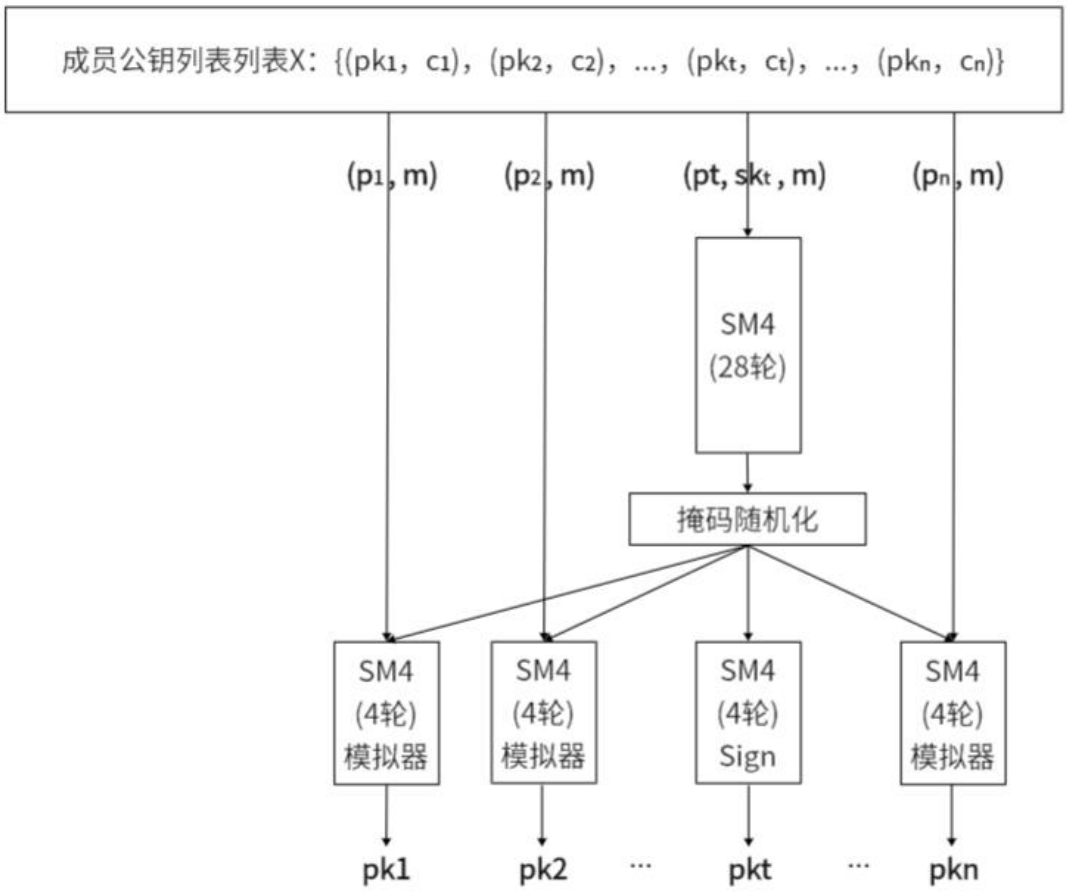

图 3-11 优化的 One-Out-of-N 结构图

经过优化后，我们的方案在成员数量为2^5数量级左右时能够得到比 Merkle 累加器方案更短的证明大小。[28]中对同一环签名分别应用 Merkle 树累加器方案和应用 *one out of N Proof* 累加器方案在不同的成员大小下的对比如下：

表 3-5 成员证明大小对比[7] [28]

成员数量	2	2^2	2^3	2^4	2^5	2^6	2^7
$\|P\|$	$70KB$	$106KB$	$142KB$	$177KB$	$213KB$	$249KB$	$285KB$
$\|P\|$	$20KB$	$30KB$	$47KB$	$82KB$	$151KB$	$290KB$	$567KB$

3.5.3 后量子签名算法应用与协议函数删减

在 Dan Boneh 在[6]的*construction* 9的 EPID 架构中，由于 Merkle 树结构的累加器为静态累加器，群成员将其证书作为 Merkle 树中的叶子来实现构建 Merkle 树。然而每有一个成员加入时都需要发布一个新的 Merkle 根，这减少了匿名集的大小。故而在 Dan Boneh 的 EPID 架构中还需要一个*Rejoin*的协议函数使群成员定期“重新加入群组”中更新他们的 Merkle 根，从而增加匿名集的大小。

我们的方案中应用了我们自主设计的基于*SM4*的后量子签名算法来构建*one out of N Proof*的累加器方案，在具体实例化中，由于该模式下的累加器为线性表的结构，且累加器内的所有成员证书被直接作为成员证明的输出，故而只需要群成员自主更新官方维护的成员承诺列表(累加器)即可。节省了服务端的交互成本。

其次，由于公钥的合法性直接由官方维护的累加器中可以得知（累加器可以动态删减不合法的成员），故而在所有的验证环节中，对于 KeyRL 的检查可以删去。减少了验证成本和交互成本。

3.6 应用场景与技术实现

3.6.1 应用场景

我们将构建的后量子隐私增强方案应用于电动汽车充电过程中的匿名认证系统中，方案不仅能对电动汽车安全有效的认证，也能确保电网安全，保证仅向授权的车辆提供充电服务，对非授权的电动汽车或电池管理系统拒绝服务，流程如下：

电动汽车在请求充电服务时，首先生成充电请求M，发送给充电站，充电站检查本地是否存有该车所在的群ID和群公钥，若没有，向充电服务提供商请求这些信息，获取对应群ID和群公钥后，充电站对不会暴露任何车身份的情况下电动汽车的身份进行验证，通过充电站的验证后，电动汽车将自己的充电请求消息m用临时会话密钥加密后发送给充电站，获取充电服务。

3.6.2 参与方

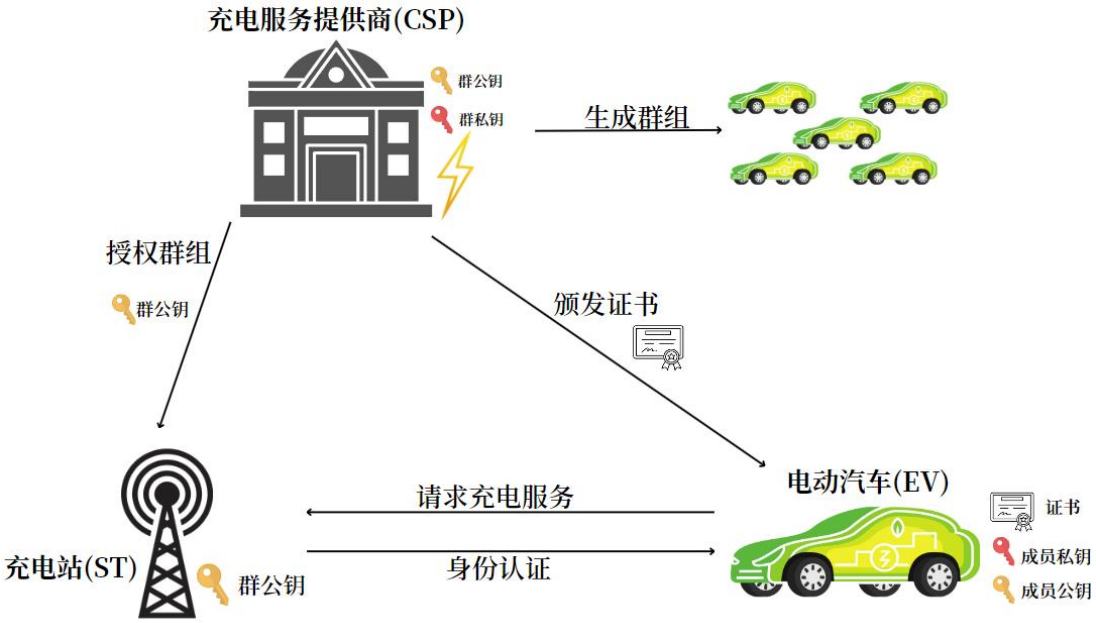

图 3-12 方案参与方

如图所示，我们优化后的后量子隐私增强方案中有三个角色：电动汽车(EV)、充s 电站(ST)、充电服务提供商(CSP)。

1. 充电服务提供商(CSP)

充电服务提供商(CSP)负责向每个充电站分配或发行群ID和群公钥。群ID用于标识每个群组，而群公钥用于验证群组中的EV。CSP维护所有的电动汽车成员资格(即

成员承诺$\{p_k, challenge\}$)。在创建新的群时，CSP会为该群生成一个新的群私钥和群公钥，同时可生成$EVprk$，用于嵌入EV充电设备的安全存储空间中。

2. 充电站(ST)

充电站(ST)通过群公钥来检查和验证由EV生成的签名，以确保它们是合法授权的 EV 且属于正确的群。使用CSP提供的群公钥，ST能够验证群中任何EV的签名，而不会暴露任何EV身份。ST可以使用此信息来授予或拒绝EV申请的充电服务，具体取决于验证结果。

3. 电动汽车(EV)

电动汽车(EV)对应于许多EV群成员中的一个单独成员，所有EV成员共享相同的安全访问级别。EV通过与CSP的交互生成成员认证私钥并获得成员资格证书，成员私钥用于匿名身份验证和识别目的，只有授权的车辆且在通过验证后才可获取充电服务。

表 3-6 充电协议主要参与方

参与方	主要持有参数	作用
CSP	群公钥 群私钥 成员资格列表	1. 分配群 ID 和群管理员密钥 2. 维护所有电动汽车成员资格和最新的撤销列表 3. 生成电动汽车私钥(用于加密) 4. 生成群公钥用于签名验证
ST	成员资格列表 群公钥	1. 检查和验证由 EV 生成的签名 2. 验证 EV 身份 3. 根据验证结果授予或拒绝 EV 申请的充电服务
EV	证书 成员私钥 成员公钥	1. 生成正式私钥用于认证签名 2. 提交充电服务请求，并等待 ST 的验证和授权

3.6.3 电动汽车加入充电群组

1. 系统初始化

CSP 执行$\boldsymbol{GAInit}$，生成系统参数、群公钥pk_{gp}及群私钥sk_{gp}。

— $\boldsymbol{GAInit}(\mathbf{1}^{\lambda})$：

充电服务提供商M运行$S.Keygen(\mathbf{1}^{\lambda})$，获得$(sk_{gp}, pk_{gp})$并输出$(sk_{gp}, pk_{gp})$保密。

2. 电动汽车注册

CSP 执行$\boldsymbol{GAJoin}$，根据 EV 的身份信息生成身份凭证证书及各项参数，并将其秘密分发给每个 EV。

— $\boldsymbol{GAJoin_{M,Pi} < (gsk, gpk, X)，\ gpk > :}$

-CSPM给 EV 成员P_i发送挑战c_i。

-P_i选择一个随机秘钥$sk_i \leftarrow \{0,1\}^{128}$并将$p_{ki} = f(sk_i, c_i)$发送给$M$。

-M定义$x_i = (pk_i, c_i)$，令$X = X \cup x_i$

-M更新X：第 0 位为$\sigma_X = S.Sign(sk_{gp}, X[1])$

M生成$cert_i = (pk_i, \mathrm{X}, \mathrm{index_{in_X}})$

将$cert_i$发给P_i。

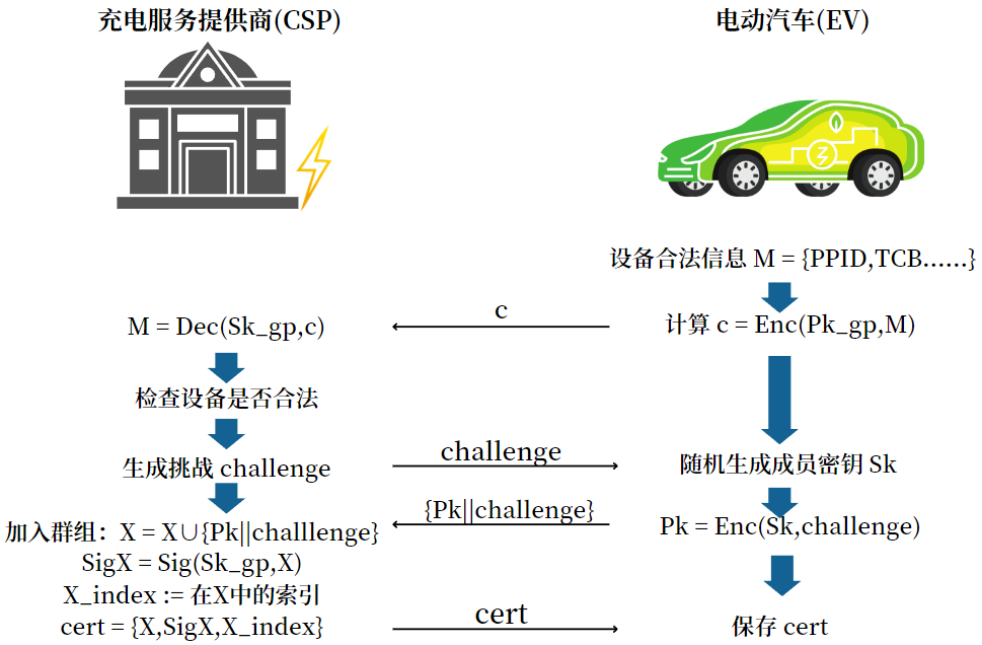

图 3-13 电动汽车注册

3.6.4 电动汽车请求充电服务

EV_i生成充电请求消息 M，根据自己的身份凭证及成员私钥生成一个群签名，并将其发送给充电站(ST)验证。

— $\boldsymbol{GASign(pk_{gp},\ sk_i,\ cert_i, m, SIG-RL)}$:

$-\ S.Verify(pk_{gp}, X)$ *//验证X 的合法性*

- $\pi = P(public(\lambda, m,\ pk_{gp}, X), private(sk_i))$ ➔ $T.proof(sk_i, X)$

- $sig = (\pi, X)$

3.6.5 充电站验证

充电站(ST)执行该功能，根据群签名、群公钥、证书等验证电动汽车(EV)的身份。

— $\boldsymbol{GAVerify(gpk, m, SIG\text{-}RL, sig)}$:

- $S.Verify(pk_{gp}, X)$
- $V\left((\lambda, m, pk_{gp}, Sig\text{-}RL, X, certi), \pi\right) = T.Verify(\pi, X)$?

-检查$sig \notin SIG\text{-}RL$

-如果以上条件都满足则判断出该签名有效且正确，否则签名无效，直接丢弃该签名消息，不授予充电服务。

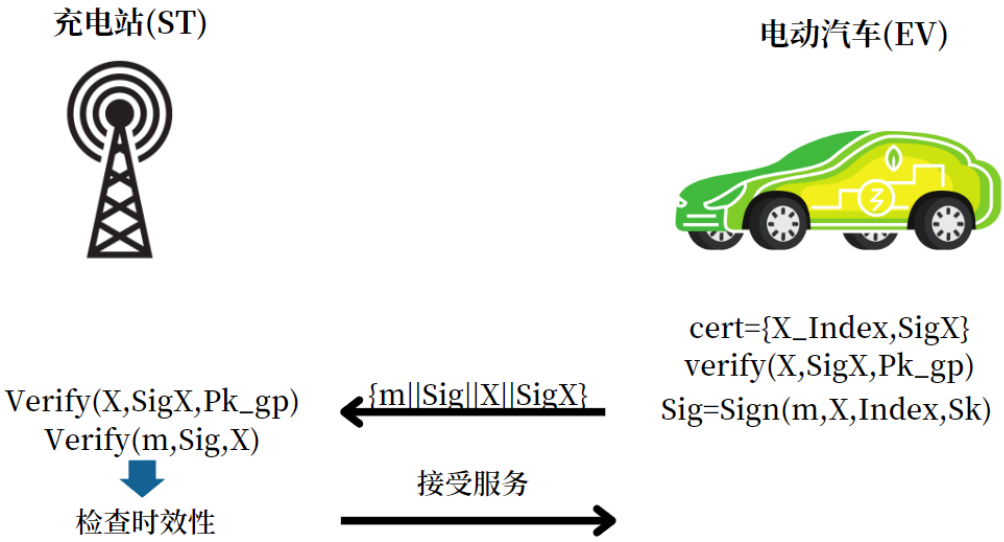

图 3-14 充电站验证

3.6.6 电动汽车充电与费用结算

电动汽车通过ST的验证后，将自己的充电请求消息m用临时会话密钥加密后发送给ST，充电请求信息 m包括：EV当前剩余电量，需求电量以及身份信息等。

ST 收到 EV 发送的消息后，首先用自己的私钥解密消息后，验证时间戳 t 的有效性，然后验证签名的有效性和正确性，若通过验证，对EV提供充电服务。

电动汽车充电完成以后，ST产生一个支付凭证，加密后发送给 EV，验证签名的正确性以及信息的完整性，若正确则完成支付。

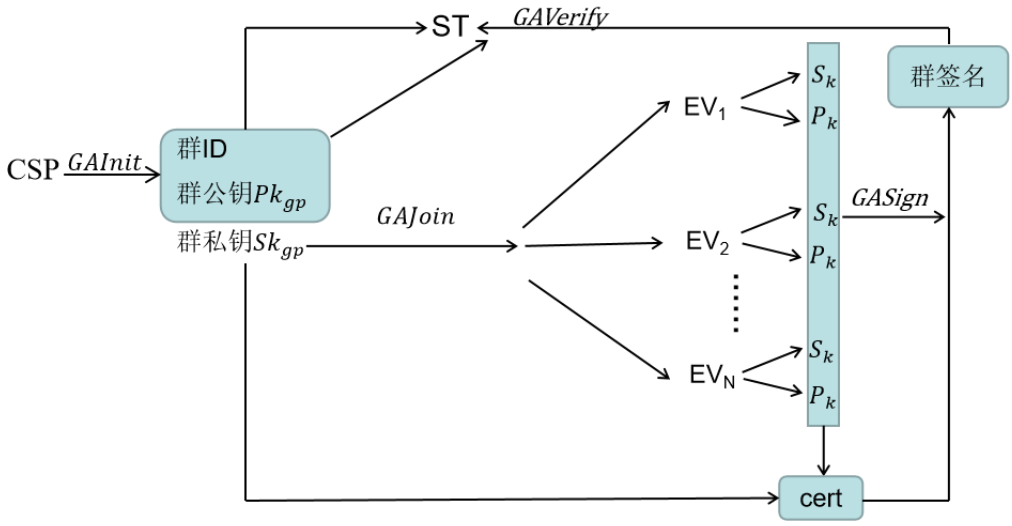

图 3-15 认证方案运行模式

3.7 方案安全性分析

基于 Dan Boneh 等人在[6]提出的安全性分析模型，对我们优化后的后量子隐私增强方案进行安全性分析[11]。

3.7.1 伪随机函数 PRF

伪随机函数$F:\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$是一个有效可计算的定长 PRF 函数，当且仅当对于所有概率多项式时间区分器 D，需要满足下式：

$$\left|Pr[D^{F_k}(1^n)=1]-Pr[D^{f_n}(1^n)=1]\right|$$

是可忽略的，其中$k \leftarrow 0,1^n_{random}$，$f_n$从 F 的不同映射的函数集合中均匀随机选取。

我们的方案中采用国家商用密码 SM4 作为伪随机函数构造多方安全算法电路，满足原方案的算法安全性要求。

3.7.2 签名算法 S

签名算法 S 是一个包含$\{KeyGen, Sign, Verify\}$的 PPT 算法，其需要满足的两个属性分别是正确性和不可伪造性，具体定义如下:

-正确性(Correctness) 对于一个正确的公私钥对(pk, sk)，在签名算法 S 的函数计算下，满足 $Pr[Verify(pk,m,Sign(sk,m))=1]=1$

-不可伪造性(unforgeablility) 要求计算能力有限的对手不能伪造签名或伪

造概率可以忽略不计，我们的方案选择在选择消息攻击下定义不可伪造性(Existential Unforgeability Under Chosen Message Attack(EUF-CMA))。如果对于所有的 PPT 能力攻击者 A，我们需要满足：

$$Pr(sk,pk) \leftarrow KeyGen(1^{\lambda}),(m^{*},\sigma^{*}) \leftarrow A.Sign(sk,*)(pk) \mid$$
$$Verify(pk,m^{*},\sigma^{*}) = 1 \wedge m^{*} \notin Q^{sign}] \leq negl(\lambda)$$

其中Q^{sign}表示攻击者对签名做的查询信息集合。

我们的方案中使用了我们独立设计的基于对称原语的抗量子 HVZK 签名方案作为签名算法 S，能够很好的满足上述安全性定义要求。

3.7.3 方案完备性证明

完备性即不受任何形式的撤销影响的组成员在拥有生成合法签名的所有信息的情况下，其签名可被验证通过。

假设签名方案 S 的正确性成立，证明系统π为零知识证明系统，f 为 PRF 函数，Sig_i为在安全参数为λ的 EPID 签名方案 G 中成功运行 JOIN 的组成员P_i生成的签名集合，则该方案是正确的当且仅当下式满足:

$$For: Sig_i \notin SigRL \rightarrow$$
$$Pr[GVerify(pk_{gp},m,KeyRL,SigRL,GSign(pk_{gp},sk_i,cert_i,m)) \neq 1] < negl(\lambda)$$

上式成立的前提是签名和证明系统的正确性，对于某些已经撤销的密钥sk_i或具有密钥随机对(sk_i,r)的签名，我们需要证明未撤销的签名或密钥sk_j不会意外地满足关系$(F(sk_j,r),r) = (F(sk_j,r'),r')$。我们称满足这些关系的对象为$BAD\ KEY$和$BAD\ SIG$。假设 F 是一个 PRF 函数，我们证明上述情况发生的概率可忽略不计，下面只包括了对$BAD\ KEY$的证明，$BAD\ SIG$的证明方法是一致的。

令$t \coloneqq \{pk,challenge\}$考虑一个成员承诺列表$T = (t_0,\dots,t_k)$存储了对所有$sk_i \in KeyRL$和$sk_j$，查询$f(sk_i,r)$的回复，$BAD\ KEY$只会在某个 i 满足$f(sk_i,r) = f(sk_j,r)$的情况下存在。在 f 满足 PRF 函数安全性定义的情况下，T与一系列随机字符串列表T'是不可区分的(如果将其中某个 PRF 输出替换为随机字符串)。如果敌手能够区分出T中由 PRF 生成的t_i和同样的列表T'中被随机字符串替换的t_i'，那么他就能够根据给出的(声称在 PRF 下生成的)列表并判断某字符串是 PRF 的输出还是一个真正的随机函数的输出。我们定义最终的混合列表为$T^{*} = (t_0',\dots,t_k')$，则该混合列表满足：

$$Pr[BAD\ KEY] = Pr[\exists i \in [0,k-1] \rightarrow (t_k' = t_i')] \leq negl(\lambda)$$

3.7.4 方案匿名安全证明

定义 3（匿名实验函数 $\boldsymbol{ANON[A,\lambda,b]}$）在敌手$A$和挑战者$C$之间进行匿名实验，$\lambda$为安全参数，$b$为挑战者$C$的输入。

1. $setup$：敌手A生成(pk_{gp}, sk_{gp}),并将gpk发送给挑战者C
2. $queries$：A可以对C根据需要进行尽可能多的查询：

   -$Join$. A请求创建新的成员P_i。C、A运行$Join\left((pk_{gp}, sk_{gp}), pk_{gp}\right)$,且$C$扮演$P_i$角色。$A$获得$cert_i$, C获得$(sk_i, cert_i)$.

   -$Sign$. A请求来自P_i对消息 m 的签名，同时从实验过程生成的签名中取部分子集作为$SIG-RLs$，C计算$sig \leftarrow Sign(pk_{gp}, sk_i, cert_i, m, SigRL)$并发送给$A$。

   -$Corrupt$. A请求P_i的私钥，C发送sk_i
3. $Challenge$：A向C发送消息 m,签名撤销列表$SigRL$和两个组成员号i_0、i_1。C计算$sig^* \leftarrow Sign(pk_{gp}, sk_{i_b}, cert_{i_b}, m, SigRL)$并发送给$A$。此处需保证$sig^*$不在$SigRL$中。
4. $restructed\ queries$：A可以像上面一样对C进行查询,但不可对P_{i_0}、P_{i_1}进行$Corrupt$查询。
5. $output$：A输出b'作为$ANON[A,\lambda,b]$的返回值

对于 EPID 签名方案G，如果不存在PPT计算能力的敌手可以以大于可忽略的概率赢得匿名试验，则称签名方案G是匿名的 EPID 签名方案。即对于任意的PPT计算能力的敌手A，需要满足:

$$|Pr[ANON[A,\lambda,0]=1]-Pr[ANON[A,\lambda,1]=1]| \le negl(\lambda)$$

假设证明系统π为零知识证明系统，f为PRF函数，根据[6]的附录 C 说明，如果敌手输入$i_0=x_0, i_1=x_1$，其中$x_0, x_1 \in [N]$(N为组成员个数上界)。那么对于所有的PPT计算能力的敌手A:

$$|Pr[ANON[A,\lambda,0]=1 | i_0=x_0, i_1=x_1]$$
$$-Pr[ANON[A,\lambda,1]=1 | i_0=x_0, i_1=x_1]| \le negl(\lambda)$$

如果我们从$[N]$中随机选择x_0, x_1。则有

$$Pr[i_0 = x_0] = Pr[i_1 = x_1] = \frac{1}{N}$$

由于x_0, x_1是由敌手A独立选取的，所以

$$Pr[i_0 = x_0, i_1 = x_1] = \frac{1}{N^2}$$

$$\Longrightarrow \frac{1}{N^2}|Pr[ANON[A,\lambda,0]=1] - Pr[ANON[A,\lambda,1]=1]| \leq negl(\lambda)$$

$$\Longrightarrow |Pr[ANON[A,\lambda,0]=1] - Pr[ANON[A,\lambda,1]=1]| \leq N^2 negl(\lambda)$$

因为$N^2 negl(\lambda)$在λ的安全参数下任然可以忽略不计，匿名安全性证明完成。

3.7.5 方案不可伪造性安全证明

定义 4 （不可伪造性实验函数 $\boldsymbol{FORGE[A,\lambda]}$）在敌手$A$和挑战者$C$之间进行匿名实验，$\lambda$为安全参数

1. $setup$：挑战者C生成(pk_{gp}, sk_{gp})，并将pk_{gp}发送给敌手A。C创建一个腐败方(corrupted party)集合U，初始化$U = \emptyset$
2. $queries$：A可以对C根据需要进行尽可能多的查询：

   -$Join$. A请求创建新的成员P_i，有以下两种情况：

   i. C内部运行$Join$，加入P_i，保留$sk_i, cert_i$并发送$cert_i$给A

   ii. C A一起运行$Join$，A扮演P_i的角色，A获得$(sk_i, cert_i)$，C获得$cert_i$，同时A发送sk_i给C，C将i存储到U中。

   -$Sign$. A请求来自P_i对消息 m 的签名，同时从实验过程生成的签名中取部分子集作为$SIG-RLs$，C计算$sig \leftarrow Sign(pk_{gp}, sk_i, cert_i, m, SigRL)$并发送给$A$(此处不可发送对$m^*$的$Sign$查询)。

   -$Corrupt$. A请求P_i的私钥，C发送sk_i给A并将i存储到U中。
3. $Forgery$：A输出消息$m^*, KeyRL^*$，$SigRL^*$，和签名sig^*。

   如果$Verify(pk_{gp}, m^*, KeyRL^*, SigRL^*, sig^*) = 1$，且对于任意$i \in U, sk_i \in KeyRL, sig^* \in SigRL$，那么$FORGE[A,\lambda] = 1$($A$伪造成功)

   否则$FORGE[A,\lambda] = 0$(A伪造失败)

对于 EPID 签名方案G，如果不存在PPT计算能力的敌手能够以大于可忽略的概率赢得不可伪造性实验，则称签名方案G是不可伪造安全的 EPID 签名方案。即对于任意PPT计算能力的敌手A，需要满足：

$$Pr[FORGE[A,\lambda]=1]\leq negl(\lambda)$$

假设证明系统π为仿真可提取性的零知识证明系统，f为PRF函数且哈希函数抗碰撞，S为不可伪造签名方案，根据[6]的附录 C 说明，能够满足上式证明方案G满足不可伪造性签名方案。

第四章 作品测试与分析

4.1 TCP 服务器系统设计

4.1.1 系统实现环境

我们将后量子隐私增强方案应用于电动汽车无线充电认证系统，将软件代码实例化在局域网内，将 IP 设为环回地址进行代码测试。

以下是服务器搭建系统的参数说明

表 4-1 参数说明

服务器环境	参数
IP 地址	127.0.0.1(测试)
端口号	4221
操作系统	Windows11-AMD64
处理器	12th Gen Intel(R) Core(TM)i7-12700H
内核数量	14
逻辑处理器数量	20
L1/L2/L3 缓存	1.2/11.5/24.0MB
内存	16G

4.1.2 TCP 交互系统设计

我们为服务端设置**三个交互服务**(入群、更新证书、退群)和一个管理功能（签名撤销管理），同时在服务器中实时管理群成员公钥集(成员承诺集)、签名撤销列表$(sig\ RLs)$和密钥撤销列表$(sk\ RLs)$。

我们为客户端设置**两个协议服务**(签名，验签)。

具体交互系统设计如下：

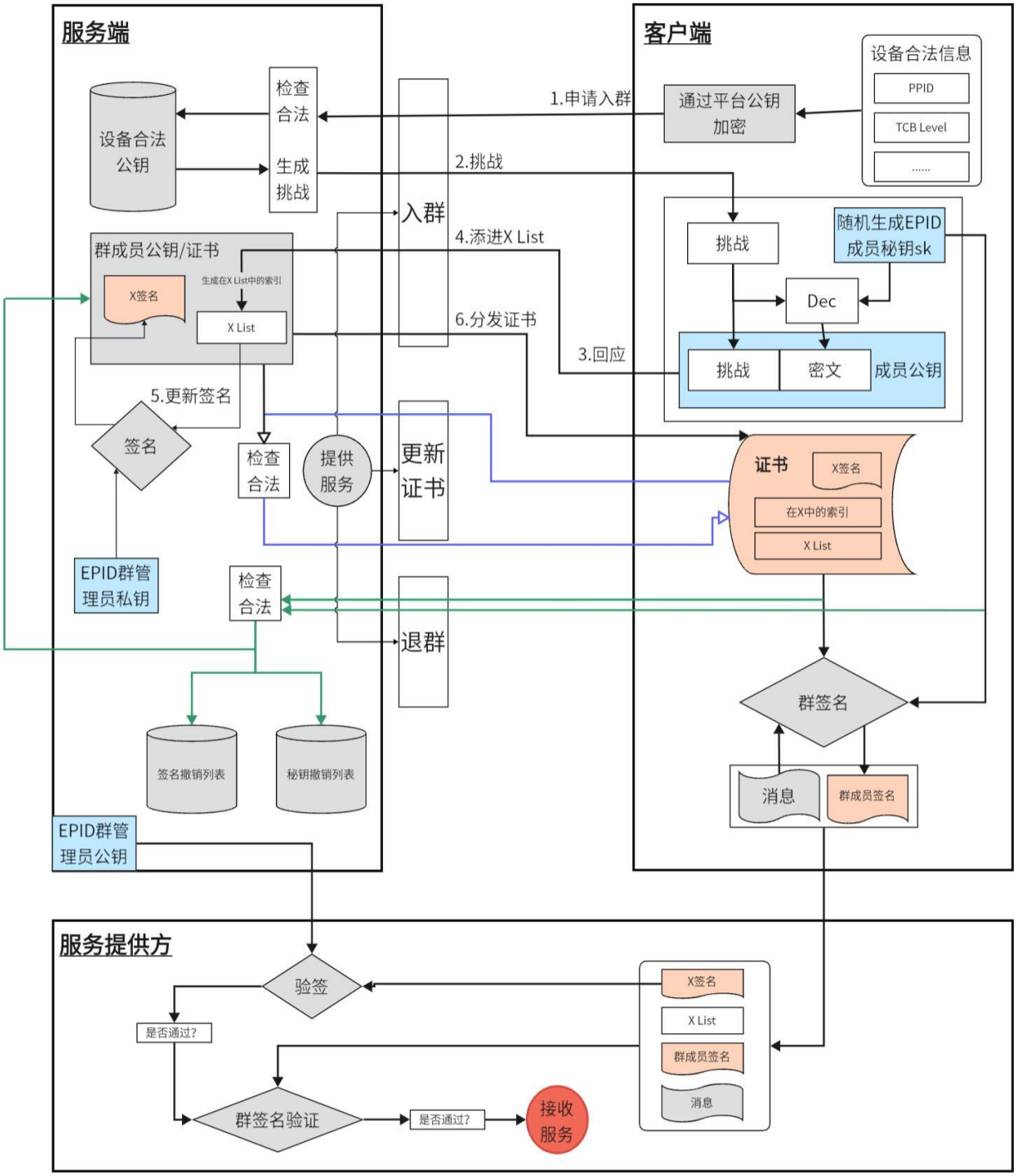

图 4-1 TCP 框架图

上图包含三方的信息交互来实现基于*EPID*的远程认证协议的配置(入群)和认证(签名、验证)过程：

1. **申请入群** 客户端生成当前设备的合法信息，使用平台服务端的公钥加密后发送给服务端。
2. **服务器挑战** 客户端通过检查设备是否合法，是否为当前平台的设备。若通过检测，为此设备提供*EPID*组服务，并将*EPID*群组信息和挑战发送回客户端。

3. **回应挑战** 客户端自己生成一个随机的$EPID$成员私钥，利用平台在硬件中设置的私钥PRK对$EPID$成员密钥进行加密保管。随后利用$EPID$成员私钥对挑战进行加密得到$EPID$成员公钥，同时根据自己的客户端信息回应服务端挑战。
4. **更新并分发证书** 将成员公钥添加至$X\ List$末尾，使用$EPID$群管理员私钥更新$X\ List$签名，并记录该成员公钥在$X\ List$的索引，$\{X\ List, SigX, index_in_X\}$作为证书发送回客户端。

上述过程为$EPID$远程认证协议的配置协议。对于$EPID$签名的具体细节在$HVZK$签名算法中讲述，此处我们描述服务提供方的验签步骤：

1. **接收服务申请** 接收到客户端发送的初始化请求，其中声明自己是$EPID$组的群成员。如果服务提供者希望获取服务，则通过向客户端请求更新的签名撤销列表和密钥撤销列表来确定后续是否继续接收服务。
2. **接收客户端挑战回应** 服务接收者对客户端发出随机数挑战，服务端通过对挑战进行处理合法生成$Report$，并使用$EPID$群成员私钥对$Report$进行签名操作，(发送的签名还需使用服务端公钥加密签名，只有服务端可以验证签名真伪，此处省略)。服务接收者接收到客户端的消息和签名后，首先检查私钥和签名是否被撤销，随后执行验签操作决定是否接收服务。

4.2 TCP 多线程对话系统代码搭建

我们使用 Windows 环境下的<Winsock2.h>进行服务端的 TCP 平台搭建，同时使用 CloseHandle(CreateThread()函数创建多个接收链接的线程，保证了服务器对多个用户的请求并行处理。

```
1.  //1、初始化 socket 库
2.  WSADATA  wsaData;
3.  WSAStartup(MAKEWORD(2,  2),  &wsaData);
4.  //2、创建 socket
5.  SOCKET  servSock  =  socket(AF_INET,  SOCK_STREAM,  0);
6.  SOCKADDR_IN  servAddr;
7.  servAddr.sin_family  =  AF_INET;
8.  servAddr.sin_addr.S_un.S_addr  =  inet_addr("127.0.0.1");
9.  //端口号设置为 4221
10.  servAddr.sin_port  =  htons(4221);
11.  //3、绑定服务端
12.  bind(servSock,  (SOCKADDR*)&servAddr,  sizeof(servAddr));
13.  WSAEVENT  servEvent  =  WSACreateEvent();
14.  WSAEventSelect(servSock,  servEvent,  FD_ALL_EVENTS);
```

```
15.   cliSock[0]  =  servSock;
16.   cliEvent[0]  =  servEvent;
17.   //5、开启监听
18.   //监听队列长度为 10
19.   listen(servSock,  10);
20.   //6、创建接受链接的线程
21.   CloseHandle(CreateThread(NULL,  0,  servEventThread,  (LPVOID)&servSock,
0,  0));
22.   cout  <<  "\n\n================群服务端开启================"  <<  endl;
23.   cout  <<  "服务:  1.申请入群  2.更新证书  3.撤销私钥"  <<  endl;
24.     while  (1)
25.     {
26.       char  contentBuf[BUFFER_SIZE/400]  =  {  0  };
27.       char  sendBuf[BUFFER_SIZE/400]  =  {  0  };
28.       cin.getline(contentBuf,  sizeof(contentBuf));
29.       sprintf(sendBuf,  "[群管理员]%s",  contentBuf);
30.       //发送管理员消息
31.       for  (int  j  =  1;  j  <=  total;  j++)
32.         send(cliSock[j],  sendBuf,  sizeof(sendBuf),  0);
33.     }
34.   WSACleanup();
```

客户端的搭建，我们使用同样的库环境和代码设计，此处不作演示。

对于不同的客户端请求和不同的服务端信息，我们设计了相应的报文头，在每一个 TCP 数据报的数据段中，我们添加了 EPID 请求报文格式，具体设计如下：

```
1. //  X索引数据大小
2. #define  X_INDEX_BYTE  4
3. //客户端信息数据大小
4. #define  NAME_MAX_BYTES  12
5. //群组 id 数据大小
6. #define  GROUP_ID_BYTES  4
7. /*  任务报文头  */
8. #define  JOIN1  0
9. #define  JOIN2  1
10. #define  UPDATE  2
11. #define  REVOKEKEY  3
```

表 4-2 报文头格式

任务报文头(0 bytes)	待填充状态(4 bytes)	数据报文长度(8 bytes)
设备信息报文(12 bytes)		处理群组 ID(16 bytes)

针对 EPID 报文格式的数据，我们使用 Msgs 的结构体同一处理所有任务和报文：

```
//消息报文头提取
void Deal_recv(size_t* Task, size_t* recv_len, char* buffer) {
  *Task = *(size_t*)(buffer);
  *recv_len = *(size_t*)(buffer + 8);
}
//获取任务类型、发送方状态和数据长度
size_t Task;
size_t recv_len;
Deal_recv(&Task, &recv_len, buffer);
//提取后续消息
MsgsInit(&message, recv_len - 16);
Msgsmemcpy(&message, buffer + 16, recv_len - 16);
```

4.3 三个交互服务代码实现

对于目前设计的三个交互服务和两个协议服务——加群，更新证书，退群和签名、验证，其中加群、更新证书和退群需要客户端与服务端之间的交互，其余交互过程不在EPID报文范围内(如验证操作为客户端与服务提供方在具体软件上的数据报交换)。故而在代码中不存在报文头处理的情况。

4.3.1 加群服务

1. 申请入群(客户端)

输入客户端相关信息 name、ctx，生成对应 Report。设置报文头任务段为 JOIN1 并将 Report 截取到 message 中。同时对相应的报文段进行消息长度检查。

```
1. /* 成员初始化 */
2. void EPID_Member_Init(GP_MEMBER_CTX* ctx, const char* name, int Join_group, Msgs* message);16);
3. //初始化
4. ......
5. MsgsInit(message, 16 + NAME_MAX_BYTES + sizeof(Join_group));
6. //设置报文头
7. size_t temp = JOIN1;
8. memcpy(message->message, &temp, 8);
9. message->message_len = 16;
10. //检查信息长度
11. if (strlen(name) > NAME_MAX_BYTES)
12. ......
13. memcpy(ctx->name, name, strlen(name) + 1);
14. Msgsmemcpy(message, ctx->name, NAME_MAX_BYTES);
15. //申请加群
16. ctx->Groupid = Join_group;
17. Msgsmemcpy(message, &Join_group, sizeof(Join_group));
```

2. 挑战(服务端)

接收到客户端的加群请求后，服务端对请求内的数据进行检验，此处为软件实现，并不存在硬件设置的 PRK，故而仅检验数据大小合法性和消息提取。

```
1. //提取消息
2. char Name[NAME_MAX_BYTES];
3. memcpy(Name, message->message, NAME_MAX_BYTES);
4. int Groupid = *(message->message + NAME_MAX_BYTES);
5. //获取对应群组信息
6. GP_MANAGER_CTX* ctxgp = ctxgp_Pool + Groupid;
```

随后生成对应的挑战 c，设置报文头任务段为 JOIN1，将群组公钥pk_{gp}，挑战 c 截取到 message 中发送给客户端。

```
1. //初始化消息
2.  MsgsInit(message,  16  +  X_EACH_SIZE_BYTE  +  CHALLENGE_SIZE_BYTE);
3. //设置报文头任务段
4.  size_t  temp  =  JOIN1;
5.  memcpy(message->message,  &temp,  8);
6.  message->message_len  =  16;
7. //获取群组公钥信息
8.  Msgsmemcpy(message,  ctxgp->pk.c,  CHALLENGE_SIZE_BYTE);
9.  Msgsmemcpy(message,  ctxgp->pk.pk,  SM4_SIZE_BYTE);
10. //生成挑战
11.  if  (random_bytes(message->message  +  message->message_len,  CHALLENGE_SI
ZE_BYTE)  !=  0)  ......
```

3. 回应挑战(客户端)

客户端获取服务端发送的报文并检查数据合法性，随后通过使用自己的 EPID 成员私钥sk，通过 prf 函数(SM4)对挑战 c 加密生成 EPID 成员公钥pk。将成员信息、和挑战c截取到 message 中发送给服务端，并设置报文头任务段为 JOIN2。

```
1. ///  检查消息长度
2. ......
3. ///  提取群组挑战，群组公钥、群组信息
4. .......
5. ///  初始化消息
6.  MsgsInit(message,  16  +  NAME_MAX_BYTES  +  GROUP_ID_BYTES  +  X_EACH_SIZE
_BYTE);
7.  ///  设置报文头任务段
8.  size_t  temp  =  JOIN2;
9.  memcpy(message->message,  &temp,  8);
10.  message->message_len  =  16;
11.  ///  回应挑战并生成相应公私钥
12.  fprintf(stdout,  "基于挑战生成成员公钥...  \n");
13.  fflush(stdout);
14.  int  ret  =  keygenerator(&ctx->pk,  &ctx->sk,  0);
15.  ......
16.  ///  发送成员信息和公钥
17.  printf("发送成员信息(%d  Bytes)\n",  NAME_MAX_BYTES  +  GROUP_ID_BYTES);
18.  EPIDprintHex("发送成员公钥（基于给定挑战）
",  &ctx->pk,  X_EACH_SIZE_BYTE);
19.  Msgsmemcpy(message,  ctx->name,  NAME_MAX_BYTES);
20.  Msgsmemcpy(message,  &ctx->Groupid,  GROUP_ID_BYTES);
21.  Msgsmemcpy(message,  &ctx->pk,  X_EACH_SIZE_BYTE);
```

4. 添加并分发证书(服务端)

先对消息进行合法性检验，提取客户端发送的成员公钥pk、成员信息 ctx、和挑战c，验证成员合法性。随后将$t=(c,pk)$加入到 X 末尾并设置临时变量 index_in_X 为当前 t 在 X 中的索引。

```
/// 检查消息长度
.....
/// 提取成员公钥，成员信息、成员挑战
.......
//获取对应群组信息
GP_MANAGER_CTX* ctxgp = ctxgp_Pool + Groupid;
//提前扩容 X 内存
ctxgp->MemSize++;
if (ctxgp->MemSize + 2 >= ctxgp->MaxSizeNow)
                .....
//添加公钥承诺
memcpy(ctxgp->ComList.X + X_EACH_SIZE_BYTE * (ctxgp->MemSize - 1), &Memberpk, X_EACH_SIZE_BYTE);
```

最后需要使用群组私钥sk_{gp}对 X 签名得到 SigX，将 SigX，X，index_in_X 打包作为证书 cert 发送给客户端。

```
//对 X 签名
size_t signature_len = picnic_signature_size(Picnic2_L1_FS);
ctxgp->ComList.XSig = (uint8_t*)malloc(signature_len);
if (ctxgp->ComList.XSig == NULL) {
  printf("failed to allocate signature\n");
  exit(-1);
}
int ret = EPID_picnic_sign(&ctxgp->sk,&ctxgp->pk,(const char*)ctxgp->ComList.X, X_EACH_SIZE_BYTE * (size_t)ctxgp->MemSize, ctxgp->ComList.XSig, &signature_len);
.....
 size_t temp = JOIN2;
 memcpy(message->message, &temp, 8);
 message->message_len = 16;
```

5. 结束 Join(客户端)

进行类似的消息检查和报文头检查，提取$cert$，使用pk_{gp}对其中的$SigX$检查签名合法性。具体代码较为简单不再展示。

以上为加入群组的代码实现，对于证书更新的代码实现与上述的 4、5 函数类似不再展示

4.3.2 退群服务

当服务端接收到客户端发送来的报文头任务段为$REVOKEKEY$的报文时，提取报文中的群成员私钥sk、群成员公钥$t=(pk、c)$。检查 t 是否在 X 中。

```
1. //提取消息
2. ......
3. //检查是否在群组中
4. printf("检查是否为群成员...\n");
5. uint32_t  now_Index  =  find_in_X(&Memberpk,  old_Index,  ctxgp->ComList.X,
ctxgp->MemSize)
6.   if  (now_Index  ==  -1)
7.       printf("非本群成员！\n");
```

如果检查通过，那么使用PRF检查sk和$t=(pk、c)$的关联性，如果关联性通过，在X中将该成员信息删去。同时将sk加入密钥撤销列表$Key\ RLs$中。需要注意的是，由于X采用的是线性表的存储方式，我们需要将删去的成员位补充，为了不影响大部分的成员在X中的索引，我们采取最后一位成员向前移位的填充方式，即将X中索引最大的成员公钥移动到删去的成员位处。

上述操作后，对X进行签名操作，更新$SigX$。

```
1. else  {
2.     if  (now_Index  ==  ctxgp->MemSize  -  1)  //如果是最后一个人
3.       memset(ctxgp->ComList.X  +  now_Index,  0x00,  X_EACH_SIZE_BYTE);
4.     else  {
5.       memcpy(ctxgp->ComList.X  +  now_Index,  ctxgp->ComList.X  +  ctxgp->Mem
Size  -  1,  X_EACH_SIZE_BYTE);
6.       memset(ctxgp->ComList.X  +  ctxgp->MemSize  -  1,  0,  X_EACH_SIZE_BYTE
);
7.     }
8.     ctxgp->MemSize--;
9.   }
10.   //对 X 签名
11. ......
12. //将私钥添加入密钥撤销列表
13. RevokeKey(sk,  &ctxgp);
```

4.4 两个协议服务实现

4.4.1 签名服务

客户端输入消息m、成员私钥sk，首先通过pk_{gp}检查$SigX$的合法性。再通过$One\ out\ of\ N\ Proof$签名协议，以 X 为承诺列表，sk为私钥对 m 签名得到 proof π，则$sig = (\pi, X)$。

```
1. //提取签名消息
2. uint8_t* text = (uint8_t*)malloc(message->message_len);
3. memcpy(text, message->message, message->message_len);
4. size_t text_len = message->message_len;
5.         //签名
6. size_t signature_len;
7. uint8_t* signature = EPID_MemberProof_sign(&ctx->sk, &ctx->pk, &signa
   ture_len,text, text_len, Member_size, ctx->Cert.ComList.X, ctx->Cert.
   Index_in_X, PreT, OnT);
8. if (signature_len == 0)
9.  return -1;
10. //设置报文头
11. MsgsInit(message, 16 + message->message_len + 2 * sizeof(size_t) + t
   ext_len + signature_len);
12. size_t temp = SIGN;
13. memcpy(message->message, &temp, 8);
14. message->message_len = 16;
15. //发送信息
16. ......
17. return signature_len;
18.}
```

4.4.2 验签服务

首先通过pk_{gp}检查$SigX$的合法性。再通过$One\ out\ of\ N\ Proof$签名协议，以 X 为承诺列表，对m和sig进行签名验证。

```
1.int EPID_Verify(uint8_t** text, Msgs* message, size_t Member_size, uin
t8_t* X)
2.{//提取消息
3. size_t text_len, signature_len;
4. uint8_t* signature;
5. uint8_t* temp = message->message;
6. memcpy(&text_len, temp, sizeof(size_t));
7. *text = (uint8_t*)malloc(text_len);
8. temp += sizeof(size_t);
```

```
9. memcpy(*text, temp, text_len);
10. memset(*text + text_len, 0x00, 1);
11. temp += text_len;
12. memcpy(&signature_len, temp, sizeof(size_t));
13. signature = (uint8_t*)malloc(signature_len);
14. temp += sizeof(size_t);
15. memcpy(signature, temp, signature_len);
16. //验证
17. int ret = EPID_MemberProof_verify(signature, signature_len, *text, t
    ext_len, Member_size, X);
18. if (ret == -1)  return -1;
19. return 0;}
```

4.5 运行界面

为了演示所有的协议服务和交互细节，以下使用基于 EasyX 搭建的 APP 和内部交互信息演示(cmd 命令行)结合的方式展现协议交互。

接下来演示多个车载功耗终端申请入群，并在签名前检查是否更新证书，通过签名进行服务请求和撤销密钥退出群组的过程。为了更好地演示，此处开启了两个车载功耗终端与充电桩服务端的交互界面。

图 4-2 ABC123 车载功耗终端申请入群

图 4-3 ABC321 车载功耗终端申请入群

图 4-4 ABC321 车载功耗终端成功申请充电服务

图 4-3、图 4-4 展示了产品序列号 ABC321 成功入群并签名的过程。

ABC321 的入群导致 ABC123 车载终端所在的群发生群成员变动，其证书过期，如果不经过证书更新，则此时群中仍然只有 ABC123，由于群签名不允许只有一个成员的情况下签名，此时 ABC123 进行服务请求（签名）将导致签名失败（见图 4-5）。

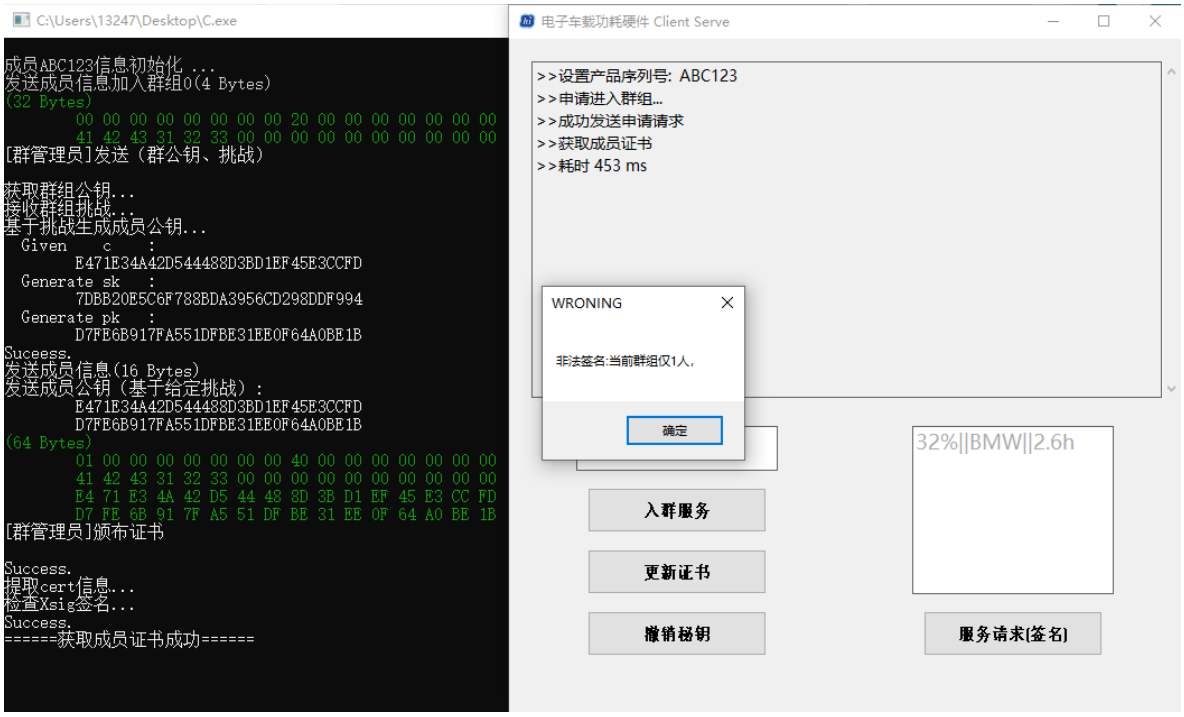

图 4-5 ABC123 车载功耗终端证书过期，签名响应失败

我们只需要更新证书后重新进行服务请求即可（图 4-6）。

图 4-6 ABC123 车载功耗终端更新证书，成功申请充电服务

此时服务端接收到两个合法的充电请求，并对将两个合法请求加入充电请求队列等待处理（见图 4-7）。

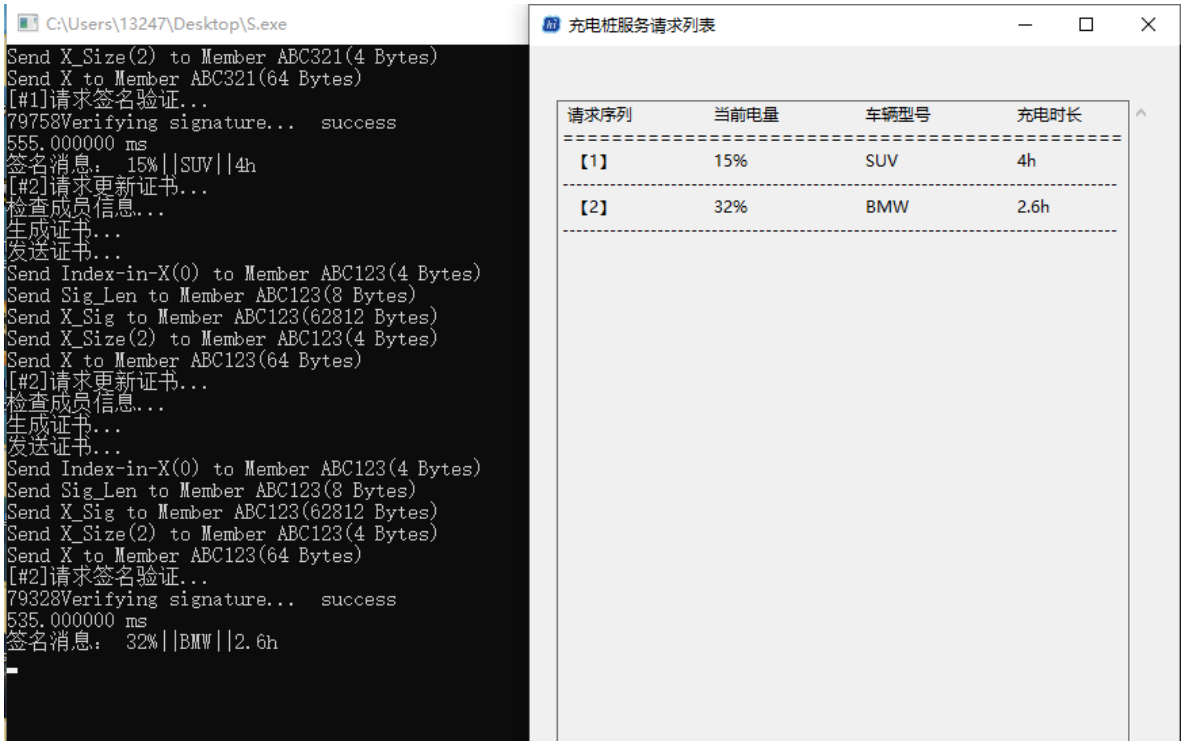

图 4-7 充电桩服务请求列表

若 ABC123 车载功耗中断请求退群服务，相当于撤销授权证书，设备未授权后无法继续进行充电服务（图 4-8）。

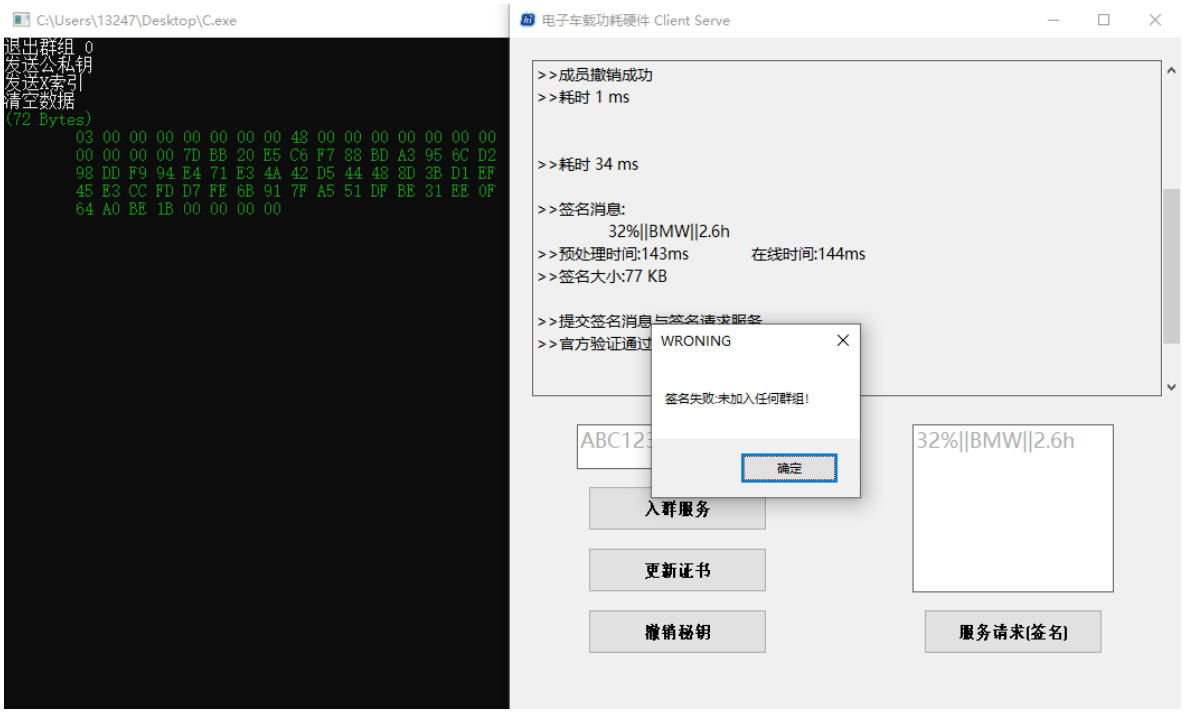

图 4-8 ABC123 车载功耗终端撤销密钥，无法申请充电服务

此时 ABC321 的证书过期，如果不更新证书，将无法申请充电服务（图 4-9）

图 4-9 ABC123 车载功耗终端证书过期，申请充电服务失败

4.6 性能测试

对 PQSM4 测试时空开销（签名大小、签名/验签时间）如下图所示，由柱状图可见，相比未优化的 SM4 电路，优化后的 SM4 电路大幅降低了时空开销，有较好的效果。

PQSM4 的签名大小优化对比如**图 4-10** 所示，可以看到我们的优化效果非常显著。

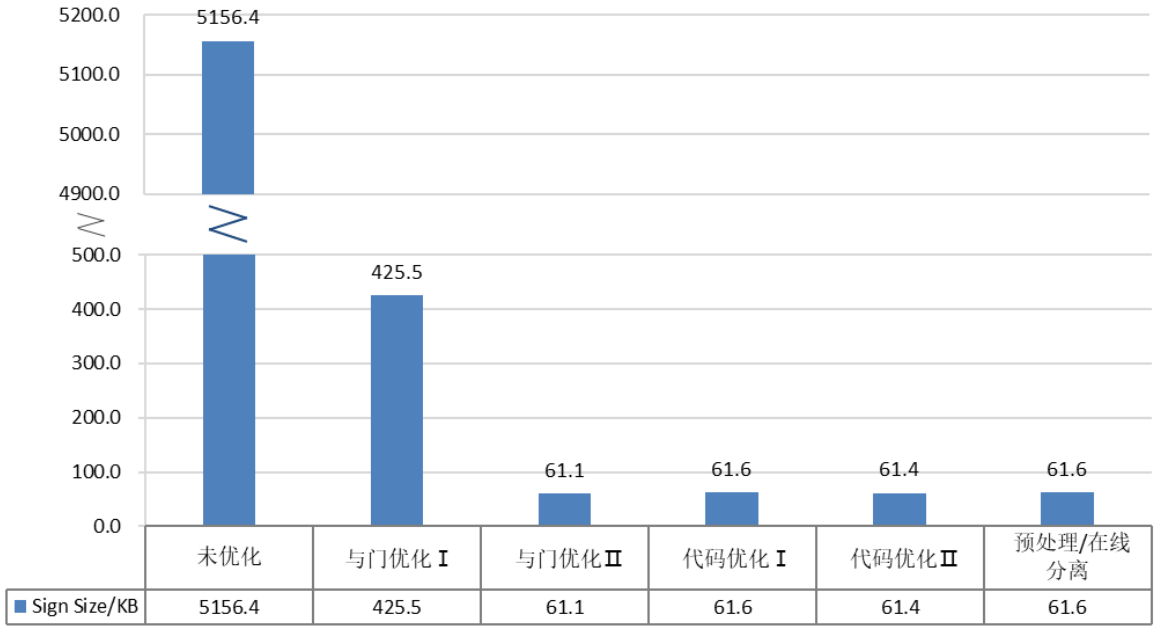

图 4-10 PQSM4 签名大小优化(单位 KB)

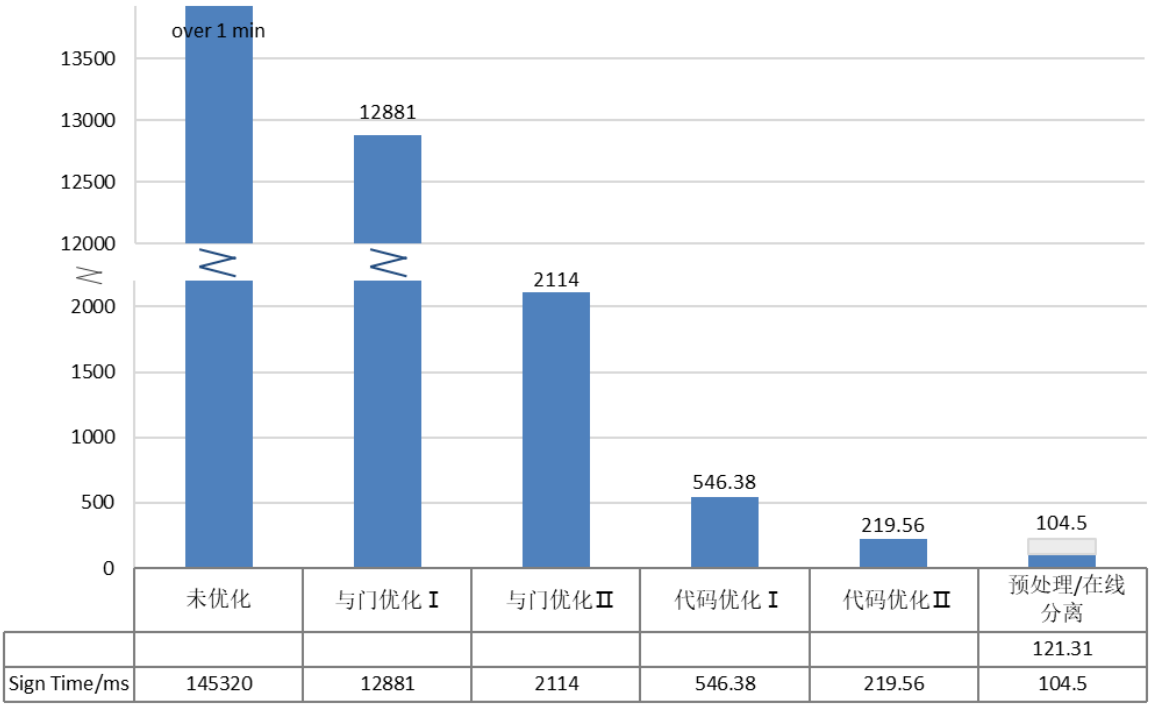

图 4-11 PQSM4 签名时间优化(单位 ms)

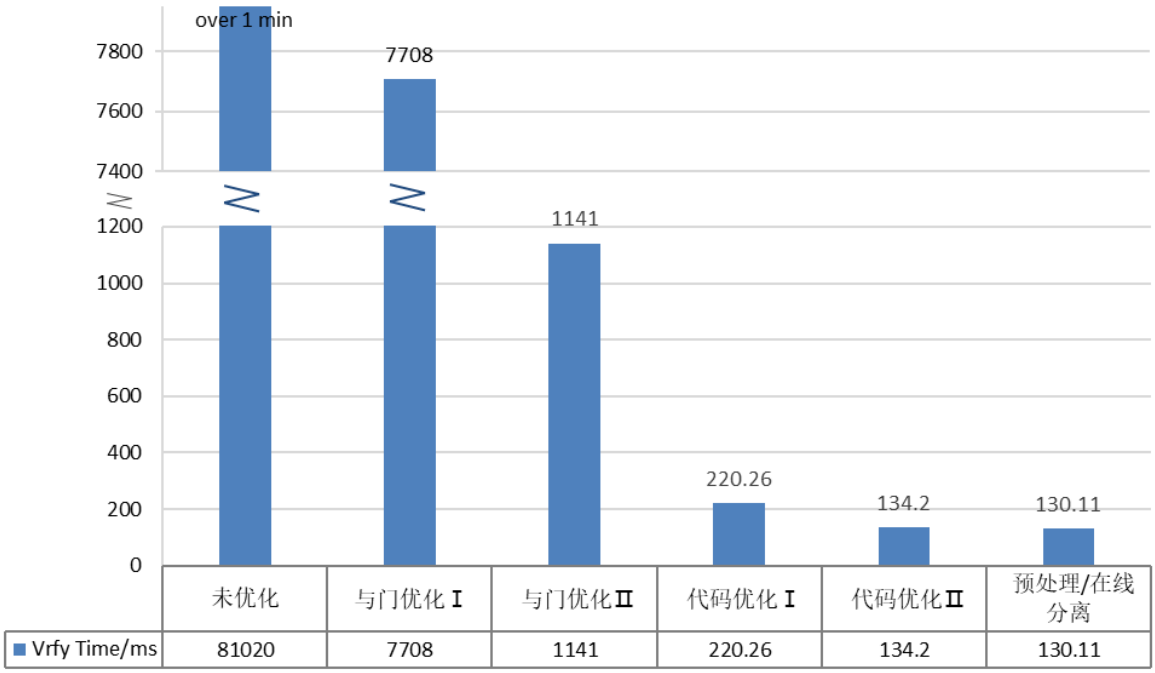

图 4-12 PQSM4 验证时间优化(单位 ms)

对 PQSM4-EPID 在不同成员大小的情况下测试时空开销（签名大小、签名/验签时间）具体数据如图 4-13 所示。

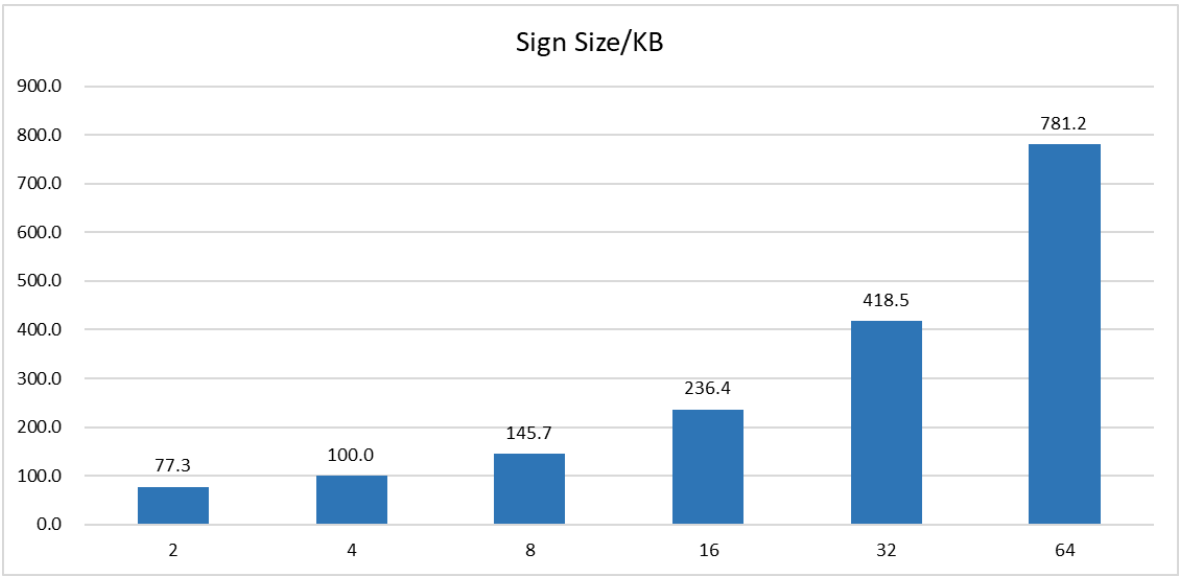

图 4-13 PQSM4-EPID 签名大小与成员大小关系(单位 KB)

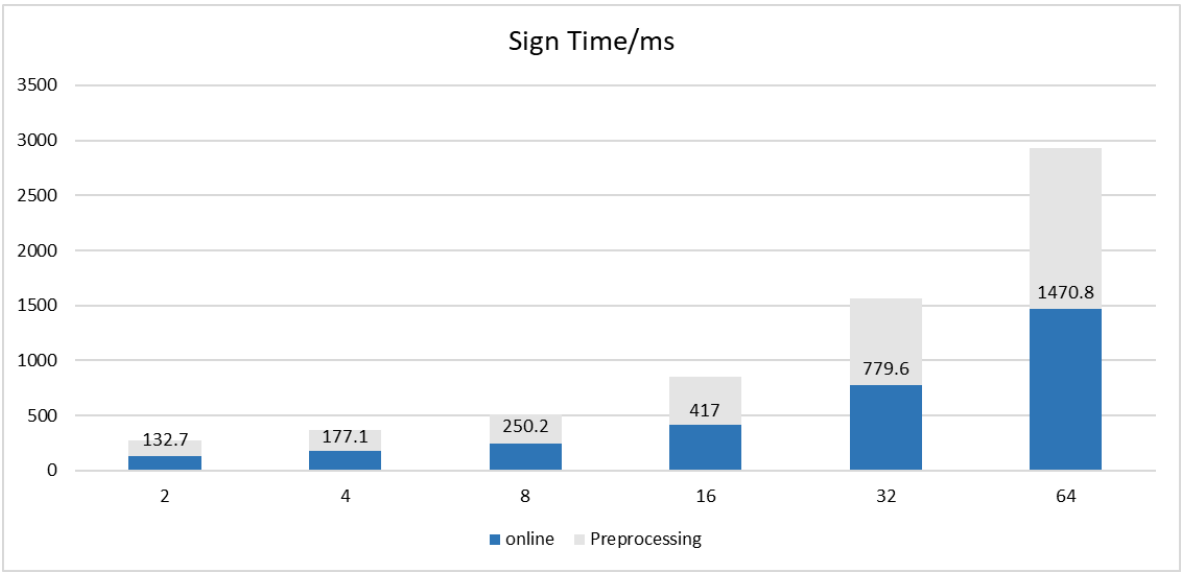

图 4-14 PQSM4-EPID 签名时间与成员大小关系(单位 ms)

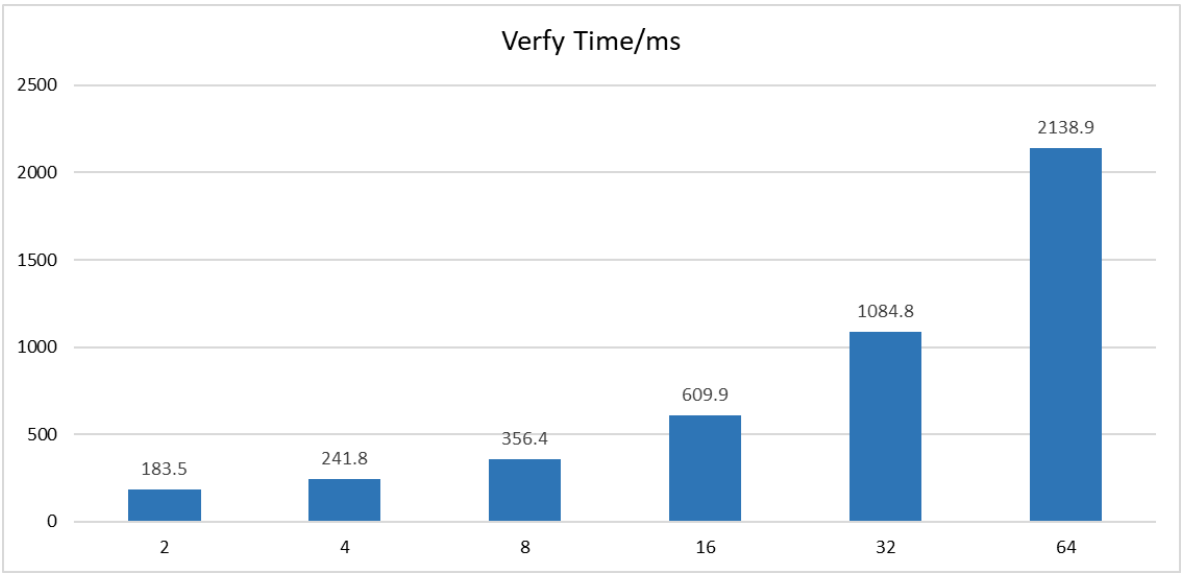

图 4-15 PQSM4-EPID 验证时间与成员大小关系(单位 ms)

第五章 创新性说明

本作品针对现有的充电协议消耗大，效率低，不抗量子等缺陷，提出了一个量子安全且高效的方案，设计并实现了 PQSM4-EPID 基于国密后量子隐私增强保护的电动汽车充电认证方案。具体来说，我们的作品具有以下优势和创新点：

1） 基于国密算法，安全自主可控

本文提出的后量子隐私增强方案的后量子安全性仅基于国密标准算法 SM4 的安全性，满足自主可控要求。

2） 可恢复零知识证明技术的优化

在将国密 SM4 转化为后量子群签名的过程中，带来了计算复杂度和通信复杂度的巨额增长(签名大小约 **5280152 字节，**签名时间超过了 **1 分钟**)。为了解决该问题，我们采用了可恢复零知识证明技术，使用基于 CDS 的 $one\text{-}out\text{-}of\text{-}N\text{-}Proof$ 成员证明方案，能够降低 **7/8** 因电动汽车成员增加而带来的签名大小的增长。

3） SM4 在多方安全计算电路上的优化

本文采用贪心算法和复合域变换对 SM4 算法在多方安全计算电路上的与门数量进行了深入优化，将 S 盒的与门数量降至 **32**，进一步提高了签名速度和效率，降低了证明的规模。

4） 采用签名分离技术，降低在线时间

本文采用了签名分离技术，在电动汽车签名的过程中，将签名分割成预处理阶段和在线阶段，减少了 **2 倍**的在线签名阶段时间开销，同时有利于后续并行化处理加速。

5） 构建基于国密后量子隐私增强方案的电动汽车认证方案

本文构建了基于国密后量子隐私增强方案的电动汽车认证方案:**PQSM4-EPID**，具有无条件匿名性和不可伪造性，综合了匿名性和认证性的安全目标，在保护电动汽车隐私的同时，实现身份认证。

第六章 总结

6.1 作品总结

本作品中，我们设计实现了一种基于国密算法的后量子隐私增强方案，采用了基于多方安全计算的零知识证明技术，将 SM4 转化为后量子群数字签名，我们对其实现效率进行深入优化，结合可恢复零知识证明技术，将成员证明方案优化为基于 CDS 的 One-of-N-Proof 证明方案；同时我们对国密 SM4 在多方安全计算电路上的与门数量进行了优化，大大提升了方案效率，本作品为我国在安全多方计算领域提供了一定的技术储备，为保护数据隐私和数据安全提供了更加可靠的保障。

之后我们将其应用于电动汽车充电场景中，实现了一套组件完全国密化且抗量子攻击的电动汽车充电认证系统。该认证系统中我们达到电动汽车认证中隐私性、公正性、准确性以及可验证性等多方面的要求，使得该系统可以应用于多种场景。

同时，本方案中仍然存在一些不足。例如，我们通过国密算法 SM4 来保证了加密算法的安全和自主可控性，但是和针对性设计的轻量级密码算法 LowMC 等专用算法相比，依旧存在着生成的签名较大、签名时间较长等问题，需要在安全和效率的权衡上做进一步的改进。其次，我们提出的后量子隐私增强方案不仅仅局限于电动汽车充电认证系统，实际上本方案所能应用的领域十分广泛，例如在区块链、电子投票、匿名协作等多个方面，都可以进行进一步的方案部署。虽然文中并没有彻底解决这些不足，但都针对性地做出了优化改进。如对于签名时间较长的问题,可以采用优化成员证明方案；并行执行多个 S 盒变换提升签名的速度；优化电路设计降低电路复杂度等等。

6.2 未来展望

本作品构造的后量子隐私增强方案不仅能够保证群中成员的无条件匿名性，还能确保群中其他成员不能伪造真实签名者的签名，综合了匿名性和认证性的安全目标，将组件完全国密化的同时对其进行大量优化，实现了一套完全基于国密的抗量子攻击的隐私增强方案。同时，我们的方案可以适用于多种场景，如区块链中的节点间身份信息的确认、电子现金中的现金认证和消费内容隐私、数据共享、物联网中的匿名认证机制等。在这些领域中，隐私保护工作尤为重要，但同时这些领域也更容易受到未来的量子攻击的威胁。因此，开发抗量子体制已成为当前研究的重要方向之一。在未

来的工作中，我们将进一步改进签名的效率，拓宽应用范围，并将其应用于更多适合的场景中。

随着电动汽车的普及，充电技术已经成为了一种重要的充电方式。然而，当前的充电系统在认证方面仍然存在一些安全隐患，例如可能会受到量子攻击等。本作品提出的一种抗量子攻击的隐私增强方案不仅可以保证电动汽车主人的无条件匿名性，还可以确保其他人无法伪造真实汽车的身份信息。这种方案可以应用于多种场景，例如在停车场、家庭或商业场所中的充电站点，以及在城市中的公共充电站点中。

在未来，随着电动汽车的普及，一个安全高效的电动汽车充电系统将受到更多人的青睐，因此在安全认证方面的需求也将越来越大。本作品提出的抗量子攻击的隐私增强方案可以为电动汽车的充电认证系统提供重要的保障，保证认证系统的安全性和可靠性。同时，我们将进一步改进方案的效率和适用范围，以便更好地满足未来电动汽车认证系统的需求。

参考文献

[1] Quan Quan Tan and Thomas Peyrin.Improved Heuristics for Short Linear Programs[C].Singapore:Nanyang Technological University.

[2] Yuyang Sun.A New Garbled Circuit Implementation of SM4 Block Cipher[D].ShanDong:ShanDong University,2022

[3] Giacomelli I , Orlandi C , Madsen J . ZKBoo: Faster Zero-Knowledge for Boolean Circuits[C]// 2016.

[4] Chase M , Derler D , Goldfeder S , et al. Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives[J]. ACM, 2017:1825-1842.

[5] Gligoroski D , Elisabeth M , Moe G . On Deviations of the AES S-box when Represented as Vector Valued Boolean Function. 2008.

[6] Dan B , Eskandarian S , Fisch B . post-quantum epid group signatures from symmetric primitives[J]. 2019.

[7] Katz J , Kolesnikov V , Xiao W . Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures[C]// the 2018 ACM SIGSAC Conference. ACM, 2018.

[8] Camacho P , Hevia A , Kiwi M , et al. Strong Accumulators from Collision-Resistant Hashing[J]. Springer Berlin Heidelberg, 2008.

[9] 蒲金伟, et al."SM4 抗差分功耗分析轻量级门限实现." 计算机应用 ..

[10] 易训.(1995).关于 S—盒的布尔函数表达式. 通信保密(02),55-60.

[11] 冯翰文,刘建伟 & 伍前红.(2021).后量子安全的群签名和环签名. 密码学报(02),183-201. doi:10.13868/j.cnki.jcr.000430.

[12] 李建立,莫燕南,粟涛 & 陈弟虎.(2022).基于国密算法 SM2、SM3、SM4 的高速混合加密系统硬件设计. 计算机应用研究 (09),2818-2825+2831. doi:10.19734/j.issn.1001-3695.2022.02.0072.

[13] 李新超,钟卫东,刘明明 & 李栋.(2018).基于秘密共享的 SM4 算法 S 盒实现方案. 计算机工程(11),148-153. doi:10.19678/j.issn.1000-3428.0051707.

[14] [姚键. 国产商用密码算法研究及性能分析[J]. 计算机应用与软件, 2019, 36(6):7.

[15] Hellman M. New directions in cryptography[J]. 1976.

[16] Cryptanalysis of Full LowMC and LowMC-M with Algebraic Techniques

[17] Fukang Liu, Gaoli Wang, Willi Meier, Santanu Sarkar, Takanori Isobe:

[18] Algebraic Meet-in-the-Middle Attack on LowMC. IACR Cryptol. ePrint Arch. 2022: 19

[19] 苗美霞,武盼汝 & 王赟玲.(2022).密码累加器研究进展及应用. 西安电子科技大学学报(01),78-91. doi:10.19665/j.issn1001-2400.2022.01.008.

[20] 李晓东 & 陶涛.(2012).求 S 盒布尔表达式的若干算法探讨. 计算机工程与应用(35),85-87.

[21] 刘景美,赵林森,王静,王新梅.确定 RijndaelS 盒布尔函数的等价方法[J].华中科技大学学报(自然科学版),2010,38(01):58-60.DOI:10.13245/j.hust.2010.01.023.

[22] 孙宇阳.一种新的 SM4 分组密码算法的姚氏乱码电路协议实现.2022.山东大学,MA thesis.

[23] SM3 密码杂凑算法. Standards.

[24] SM4 分组密码算法. Standards.

[25] Mart ´ ınez-Herrera, A.F., Mex-Perera, J.C., Nolazco-Flores, J.A.: Merging the Camellia,SMS4 and AES S-Boxes in a single S-Box with composite bases. In: Information Security,16th International Conference, ISC 2013, Dallas, Texas, USA, November 13-15, 2013,Proceedings, pp. 209–217 (2013)

[26] Bai, X., Xu, Y., Guo, L.: Securing SMS4 cipher against differential power analysis and itsVLSI implementation. In: IEEE Singapore International Conference on CommunicationSystems, pp. 167–172 (2009)

[27] Abbasi, I., Afzal, M.: A compact S-Box design for SMS4 block cipher. IACR CryptologyePrint Archive 2011, 522 (2011)

[28] Zhang H, Wei P, Xue H, et al. Resumable zero-knowledge for circuits from symmetri c key primitives[C]//Information Security and Privacy: 27th Australasian Conferenc e, ACISP 2022, Wollongong, NSW, Australia, November 28–30, 2022, Proceedings. C ham: Springer International Publishing, 2022: 375-398.

[29] Libert B, Ling S, Nguyen K, et al. Zero-knowledge arguments for lattice-based accumu lators: logarithmic-size ring signatures and group signatures without trapdoors[C]//Adv ances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on th e Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2 016, Proceedings, Part II 35. Springer Berlin Heidelberg, 2016: 1-31.

[30] S. Köhler, R. Baker, M. Strohmeier, and I. Martinovic, "Brokenwire: Wireless Disrupti on of CCS Electric Vehicle Charging," arXiv preprint arXiv:2202.02104, 2022.

[31] FIAT,A.,AND SHAMIR,A. How to prove yourself: Practical solutions to identification and signature problems. In CRYPTO(1986), pp. 186–194.