文章编号:1671-4229(2025)01-0001-08

基于 SM2 的可穿刺签名方案

周　权[1]，谢淑婷[1*]，曾志康[1]，陈丽丽[2]，卢子冲[2]
(1.广州大学 数学与信息科学学院,广东 广州　510006; 2.广州大学 计算机科学与网络工程学院, 广东 广州　510006)

摘　要: SM2 签名算法是我国商用密码体系的重要组成部分,现已被广泛应用于多个领域。然而,私钥泄露的风险和签名的前向安全性问题仍然备受关注。对此,文章提出了一个基于 SM2 的可穿刺签名方案(SM2-PS),支持在密钥泄露的情况下保证历史签名的安全性。SM2-PS 方案支持对任何特定部分的消息进行穿刺,且其密钥穿刺操作仅需一次从布隆过滤器中删除关键元素即可完成。在椭圆曲线离散对数问题假设下,SM2-PS 方案是满足存在性不可伪造的。性能分析及对比表明,SM2-PS 方案在密钥生成和签名验证上的计算效率分别提高了 51.83%和 94.43%,其签名长度仅为 0.156 KB。

关键词: SM2 数字签名; 私钥泄露; 可穿刺签名; 布隆过滤器; 前向安全性
中图分类号: TN918.4　　**文献标志码:** A

Puncturable signature scheme based on SM2

ZHOU Quan[1], *XIE Shu-ting*[1*], *ZENG Zhi-kang*[1], *CHEN Li-li*[2], *LU Zi-chong*[2]
(1. School of Mathematics and Information Sciences, Guangzhou University, Guangzhou 510006, China;
2. School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou 510006, China)

Abstract: The SM2 signature algorithm is an essential component of commercial cryptography systems in China and has been widely applied in various fields. However, the risks associated with private key leakage and the issue of forward security in signatures continue to receive significant attention. To address these concerns, this paper proposes a puncturable signature scheme based on SM2 (SM2-PS), ensuring historical signatures' security even during key leakage. The SM2-PS scheme supports puncturing specific parts of a message, and its key puncturing operation requires only a single deletion of critical elements from a Bloom filter. Under the assumption of the elliptic curve discrete logarithm problem, the SM2-PS scheme enjoys existential unforgeability against chosen-message attacks. Performance analysis and comparisons show that the SM2-PS scheme improves computational efficiency in key generation and signature verification by up to 51.83% and 94.43%, respectively, while the signature length is only 0.156 KB.

Key words: SM2 digital signature; key leakage; puncturable signature; bloom filter; forward security

SM2 是我国密码标准中规定的数字签名算法,其安全性基于椭圆曲线离散对数问题(Elliptic Curve Discrete Logarithm Problem, ECDLP)在有限域上难以求解的假设[1]。由于其具有较高的效率

收稿日期: 2024-09-06; **修回日期:** 2024-11-20
基金项目: 国家重点研发计划资助项目(2021YFA1000600); 国家自然科学基金资助项目(12171114);广州大学研究生创新能力培养资助项目(JCCX2024-012)
作者简介: 周权(1971—),男,副教授,博士. E-mail:zhouqq@gzhu.edu.cn
*通信作者. E-mail:s.t.xie@foxmail.com
引文格式: 周权,谢淑婷,曾志康,等. 基于 SM2 的可穿刺签名方案[J]. 广州大学学报(自然科学版),2025,24(1):1-8.

和安全性,SM2 成为我国数字签名应用的首选,且随着数字时代的快速发展,现已出现了一些基于 SM2 的签名算法,如盲签名[2-3]和环签名[4-5]等。然而,随着 SM2 签名算法的普及和网络攻击的多样化,用户密钥泄露的事件时常发生,攻击者可通过已泄露的用户密钥生成任何消息的合法签名,从而严重危及用户的隐私安全。

为了解决数字签名方案中因用户密钥泄露而引发的安全问题,学者们提出了前向安全数字签名方案的概念及其形式化定义[6-7]。前向安全确保即使长期密钥在未来被破解或泄露,也不会影响先前签名的安全。随着学者们对前向安全数字签名方案的深入研究,提出了大量在安全性和效率方面都具有各自优势的方案[8-11]。然而,传统的前向安全数字签名是通过周期性更新密钥来实现的,会产生一些额外的开销且难以在特定的时间间隔内撤销特定的消息。对此,学者们尝试以不同的方式来实现数字签名的前向安全。

可穿刺签名[12]通过选择消息、更新密钥来实现对签名能力的细粒度撤销。与一般的数字签名方案相比,它引入了"穿刺"算法,允许签名者选择特定消息进行密钥更新。具体而言,签名者通过对所选消息进行穿刺,从而生成穿刺密钥,使得签名者能够在除所选消息外的任何消息上进行签名。此外,文献[13]进一步推广了可穿刺签名的定义,提出了与穿刺密钥相关联的字符串可以是消息的特定部分(如前缀),而不仅仅是整个消息,这种方式简化了对具有相同前缀的消息进行撤销的过程。

布隆过滤器是在空间效率和查询时间上具有巨大优势的数据结构,它可用较小的内存空间来存储大量的元素且可在常数时间内判断一个元素是否在某个集合中。尽管其存在假阳性率,但可通过适当的参数调节[14],将假阳性率控制在可接受的范围内。对此,使用布隆过滤器来构建安全、高效的前向安全数字签名方案是一种潜在的方式。

为解决上述问题,促进国密的广泛应用,本文提出了一种基于 SM2 的可穿刺签名方案(SM2-PS)。该方案通过动态地更新私钥,确保每个操作或签名生成时使用独立、不同的私钥,从而最大限度地减少私钥泄露的潜在风险,并实现了签名的细粒度前向安全性。此外,SM2-PS 方案适用于需要对数字签名进行长期有效保护的应用场景,特别是在私钥泄露后,仍能保证已生成签名的安全性,其在金融(防止财务欺诈和伪造交易)、政府(提升政府服务的安全性和可信度)和物联网(增强整个物联网系统的安全性)等多个领域都具有广泛的应用前景。具体地说,本文的主要贡献如下:

(1)拓展了 SM2 签名以满足细粒度前向安全性,避免了因密钥泄露所造成的隐私威胁;

(2)提出了一种 SM2-PS 方案,支持密钥穿刺。此外,所提方案的密钥穿刺操作仅通过一次从布隆过滤器中删除关键元素即可实现;

(3)证明了所提 SM2-PS 方案是满足存在性不可伪造(Existential Unforgeability against Chosen-MessageAttacks, EUF-CMA)的,并通过实验分析对比了其在计算和存储开销上与已有方案的优势。

1 相关工作

SM2 签名被广泛应用于商用密码产品和系统中,在保障信息安全方面发挥了重要作用。文献[2]提出一种基于 SM2 签名的无证书盲签名协议,该协议与其他同类型方案相比具有较低的计算开销。文献[3]提出一种基于 SM2 签名的两方协作盲签名方案,其在传统盲签名的基础上增加了签名参与者的方式,满足了不可伪造性和不可链接性。然而,盲签名的完全盲化带来了一些不可控因素,如提高了非法分子滥用签名的风险。文献[4]提出一种基于 SM2 签名的环签名方案,并对其进行推广,进而提出基于 SM2 签名的可链接环签名的两种变型。文献[5]提出一种基于 SM2 的可否认环签名方案,能有效降低通信开销,但该方案仅实现了一定程度上的匿名性。

前向安全签名方案是保证在密钥泄露之前所生成签名的安全性,即拥有泄露密钥的攻击者不能伪造之前生成的签名。1997 年,文献[6]提出了前向安全签名的概念,1999 年,文献[7]提出了一种基于 Fiat-Shamir 的前向安全数字签名方案,该方案通过私钥更新、公钥不变来提供前向安全性。随着学者们对可实现前向安全数字签名方案

的深入研究,提出了很多在安全性和效率方面各有优势的方案。文献[8]提出了一种可实现高效签名和验证的前向安全签名方案。文献[9]对文献[7]中的私钥进行优化,使其在具有更短密钥的同时,提高其方案的安全性。文献[10]提出一种基于零知识证明技术来实现前向安全的聚合签名方案,该方案的复杂性与时间周期无关,并且允许任意时间段的多用户签名。文献[11]提出一种可实现前向安全的聚合签名方案,该方案基于中国剩余定理,有效降低了存储空间,并在随机预言机模型下证明了方案具有不可伪造性。

可穿刺签名方案允许对任何消息更新其签名密钥,得到的穿刺签名密钥可以对除所选消息之外的所有消息生成签名。文献[12]提出了可穿刺签名的概念,但其构造是基于不可区分混淆[15]和单向函数的,在实际应用中会产生较大的计算开销。与此同时,该方案中与可穿刺签名密钥相关联的消息是完整的签名消息,这使得方案的效率较低。文献[16]将统计约束承诺与非交互式零知识证明相结合,提出了一种新的可穿刺签名方案。与其余可穿刺签名方案不同的是,它与签名密钥相关联的字符串可以是消息的任何前缀,这使得方案变得更加灵活和高效,而且它在每次穿刺操作中更新的是公钥而不是密钥。虽然该方案有一定的优势,但是在实际应用中,连续对更新的公钥进行验证是很低效的,而且对于系统用户来说,维护其他用户更新的公钥是很困难的。文献[13]提出了一种与布隆过滤器相结合的可穿刺签名方案,该方案在签名大小和算法效率方面都优于以前的方案,尤其是在密钥穿刺效率方面。该方案基于双线性映射中的强 Diffie-Hellman 假设,并在随机预言机模型中被证明是安全的。文献[17]提出一种基于身份的可穿刺签名方案,该方案的构造基于布隆过滤器,并且在前向安全性和密钥更新效率方面都优于传统的方案。在计算性 Diffie-Hellman 假设下,该方案在随机预言模型下证明了存在不可伪造性。文献[18]通过将身份作为前缀,从基于身份的签名中提出了可穿刺签名的一般构造。在通用框架的帮助下,在格、双线性映射和多元二次多项式上提出了不同的可穿刺签名实例,所有这些都支持预定的穿刺时间和有效的密钥穿刺操作。

2　预备知识

2.1　可穿刺签名

令$\mathcal{M}$为消息空间。可穿刺签名方案由以下 4 种算法组成,包括系统初始化(*Setup*)、密钥穿刺(*Puncture*)、消息签名(*Sign*)和签名验证(*Verify*)。

(1)系统初始化。输入安全参数λ,输出公钥 pk 和初始密钥 sk。

(2)密钥穿刺。输入当前密钥 sk 和字符 $str \in \mathcal{M}$,输出穿刺密钥 sk',字符 str 被视为穿刺前缀。

(3)消息签名。输入穿刺密钥 sk' 和消息 $m \in \mathcal{M}$,如果字符 str 没有被穿刺,则输出签名 σ;否则,输出 $\perp$。

(4)签名验证。输入公钥 pk、消息 m 和签名 σ,如果 σ 是 m 的有效签名,则返回 1;否则,返回 0。

2.2　布隆过滤器

布隆过滤器[14]是一种概率数据结构,它能在允许一定假阳性率的情况下,快速判断元素 s 是否属于集合$\mathcal{S}$。简单地说,对于任何 $s \in S$,布隆过滤器输出 1;对于任何 $s \notin S$,它可能以小概率错误地输出 1。令集合$\mathcal{U}$为元素空间,定义在集合$\mathcal{U}$上的布隆过滤器由概率多项式时间(Probabilistic Polynomial-Time, PPT)算法的三元组(*Gen*, *Update*, *Check*)组成,其构造如下。

(1)布隆过滤器生成 $Gen(\ell,k)$。输入两个整数 $\ell,k \in \mathbb{N}$,该算法首先对 k 个通用哈希函数 $H_1,\cdots,H_k$ 进行采样,其中,$H_j: U \to [\ell]$,然后初始化 ℓ-比特数组 $T=0^{\ell}$,并令 $H=\{H_j\}_{j\in[k]}$,最后返回(H,T)。

(2)布隆过滤器更新 $Update(H,T,u)$。为了将元素 $u \in U$ 添加到$\mathcal{S}$中,该算法对任意 $j \in [k]$,更新 $T[H_j(u)]=1$,而 T 的其他位置不变,最后返回 T'。

(3)元素检查 $Check(H,T,u)$。为了检查元素 u 是否在$\mathcal{S}$中,该算法计算 $\wedge_{j\in[k]} T[H_j(u)]$,若算法返回 1,则 $u \in S$;否则,返回 0。

3　SM2-PS 方案定义

3.1　形式化定义

所提的 SM2-PS 方案由 5 个多项式时间算法

组成,其包括系统初始化 *Setup*、密钥生成 *KeyGen*、密钥穿刺 *Puncture*、消息签名 *Sign* 和签名验证 *Verify*。

(1)系统初始化 $Setup(1^\lambda)\rightarrow pp$。输入安全参数 λ,输出系统的公开参数 pp。

(2)密钥生成 $KeyGen(pp,\ell,k)\rightarrow(pk,sk)$。输入系统公开参数 *params* 和两个整数 $\ell,k\in\mathbb{N}$,输出公钥 pk,私钥 sk。

(3)密钥穿刺 $Puncture(pp,sk,str)\rightarrow sk'$。输入系统公开参数 pp、当前用户的私钥 sk 和字符 $str\in\mathcal{M}$,输出更新后的私钥 sk'。

(4)消息签名 $Sign(pp,sk,m)\rightarrow\sigma$。输入系统公开参数 pp、当前用户私钥 sk 和待签名的消息 m,输出签名 σ。

(5)签名验证 $Verify(pp,pk,\sigma,m)\rightarrow(1/0)$。输入系统公开参数 pp、用户公钥 pk、签名 σ 和消息 m,如果 σ 是 m 的有效签名,则验证算法输出1,否则输出0。

3.2 安全性定义

所提SM2-PS方案的存在性不可伪造通过一个挑战者$\mathcal{C}$和一个敌手$\mathcal{A}$之间的安全游戏 $Ga^{EUF\text{-}CMA}_{\mathcal{A}\leftrightarrow\mathcal{C}}$ 来定义,其主要包括以下阶段。

(1)系统初始化。$\mathcal{C}$初始化系统,生成 pp,并将其发送给$\mathcal{A}$。

(2)查询阶段。在此阶段,$\mathcal{A}$可以自适应地进行下列查询。

(i)哈希查询。$\mathcal{A}$向$\mathcal{C}$发起关于用户 *ID* 的哈希查询,$\mathcal{C}$返回相应的哈希结果给$\mathcal{A}$。

(ii)私钥提取查询。$\mathcal{A}$向$\mathcal{C}$发起关于用户 *ID* 的密钥生成查询,$\mathcal{C}$返回相应的私钥 sk 给$\mathcal{A}$。

(iii)签名查询。$\mathcal{A}$向$\mathcal{C}$发起关于用户 *ID* 和消息 m 的签名查询,$\mathcal{C}$返回相应的签名 σ 给$\mathcal{A}$。

(iv)私钥穿刺查询。$\mathcal{A}$向$\mathcal{C}$发起关于用户 *ID* 和消息 m 的穿刺查询,$\mathcal{C}$返回被穿刺过的私钥 sk'。

(3)伪造阶段。结束上述查询后,$\mathcal{A}$输出一个关于(m^*,ID^*)的签名 σ^*,若$\mathcal{A}$能生成有效的签名,则称其赢得了游戏。

定义1(存在性不可伪造) 如果对于任何PPT敌手$\mathcal{A}$,能以可忽略的概率赢得上述游戏 $Ga^{EUF\text{-}CMA}_{\mathcal{A}\leftrightarrow\mathcal{C}}$,则所提方案SM2-PS是存在性不可伪造的。

4 SM2-PS方案构造

所提SM2-PS方案主要包括系统初始化、密钥生成、密钥穿刺、消息签名和签名验证,其具体构造如下。

4.1 系统初始化

该算法输入安全参数 1^λ,生成阶为素数 p 的循环群$\mathbb{G}$,其中,$P=(x_P,y_P)$是$\mathbb{G}$的生成元;随后选择两个安全的哈希函数 $h_1:\{0,1\}^*\rightarrow\mathbb{Z}_p^*$ 和 $h_2:\{0,1\}^*\times\mathbb{G}\rightarrow\mathbb{Z}_p^*$;最后,发布系统公开参数 $pp=\{\mathbb{G},p,P,h_1,h_2\}$。

4.2 密钥生成

该算法依次执行下列过程以输出用户 *ID* 的公钥 pk 和私钥 sk。

(1)运行 $Gen(\ell,k)$ 算法生成$(\{H_j\}_{j\in[k]},T)$,其中,$T=0^\ell$;

(2)选择 $a\in_R\mathbb{Z}_p^*$,计算 $sk_i=\dfrac{a}{a+h_1(i)}$,$pk_i=sk_i\cdot P$ 和 $d=a\cdot P$;

(3)输出公钥 $pk=(\{H_j\}_{j\in[k]},pk_i,d)$、私钥 $sk=(T,\{sk_i\}_{i\in[\ell]})$。

4.3 密钥穿刺

该算法依次执行下列过程以输出更新私钥 sk'。

(1)将字符 *str* 添加到布隆过滤器的集合中,执行 $Update(\{H_j\}_{j\in[k]},T,str)$ 算法进行更新,输出 T'。

(2)对每一个 $i\in[\ell]$,定义

$$sk'_i=\begin{cases}sk_i, & T'[i]=0,\\ \perp, & \text{otherwise},\end{cases}$$

其中,$T'[i]$表示 T'的第 i 个比特。

(3)输出更新后的私钥 $sk'=(T',\{sk'_i\}_{i\in[\ell]})$。

4.4 消息签名

给定前缀为 m'的消息 m,若 $Check(H,T,m')\rightarrow1$,则输出 $\perp$;否则,若 $Check(H,T,m')\rightarrow0$,则至少存在一个 $i^*\in\{i_1,\cdots,i_k\}$,使得 $sk_{i^*}\neq\perp$,其中,$i_j=H_j(m')(j\in[k])$。最后,签名者依次进行下列操作以输出消息签名 σ。

(1)选择随机数 $u,v\in\mathbb{Z}_p^*$,计算

$$\begin{cases} R = u \cdot P = (R_x, R_y), \\ w = v \cdot d, \\ h = H_2(m, w), \\ z = (v - h) \cdot sk_{i^*}。 \end{cases}$$

(2)计算 $r = (R_x + h) \bmod (p)$ 和 $s = (1 + sk_{i^*})^{-1}(u - r \cdot sk_{i^*}) \bmod (p)$,若 $s = 0, r = 0$ 或 $r + u = 0$,则重新选择 u,否则继续。

(3)输出签名 $\sigma = (r, s, h, z, i^*)$。

4.5　签名验证

在此阶段,验证者依次进行下列操作以进行签名验证。

(1)检查 $r, s \in \mathbb{Z}_p^*$ 是否成立,若不成立,则签名无效。

(2)计算 $h' = h_2(m, w')$ 和 $t' = (r + s) \bmod (p)$,其中,$w' = z \cdot (h_1(i^*) \cdot P + d) + h \cdot d$。若 $t' = 0$,则签名无效。

(3)计算 $R' = s \cdot P + t' \cdot pk_i = (R'_x, R'_y)$ 和 $r' = (R'_x + h') \bmod (p)$。

(4)验证 $r = r'$ 和 $i^* \in S_{m'} \wedge h'$ 是否成立,其中,$S_{m'} = \{H_j(m') \mid j \in [k]\}$。若成立,则签名验证通过,否则签名无效。

5　正确性和安全性分析

5.1　正确性分析

若所提 SM2-PS 方案是正确的,则

(1) 对任意按正确方式生成的公开参数 $pp \leftarrow Setup(1^\lambda)$、任意用户初始公私钥 $(pk, sk) \leftarrow KeyGen(pp, \ell, k)$ 以及任意签名消息 $m \in \mathcal{M}$,有 $Verify(pp, pk, Sign(pp, sk, m), m) \rightarrow 1$ 成立。具体地,若密钥是初始的并且没有被穿刺,则有

$$\begin{aligned} R' &= s \cdot P + t' \cdot P_i = s \cdot P + (r + s) \cdot sk_{i^*} \cdot P = \\ &(1 + sk_{i^*}) \cdot s \cdot P + r \cdot sk_{i^*} \cdot P = \\ &(1 + sk_{i^*})(1 + sk_{i^*})^{-1}(u - r \cdot sk_{i^*}) \cdot P + \\ &r \cdot sk_{i^*} \cdot P = (u - r \cdot sk_{i^*}) \cdot P + r \cdot sk_{i^*} \cdot P = \\ &u \cdot P = R。 \end{aligned}$$

此外,由于

$$\begin{aligned} w' &= z \cdot (h_1(i^*) \cdot P + d) + h \cdot d = \\ &(v - h) \cdot sk_{i^*} \cdot (h_1(i^*) \cdot P + a \cdot P) + h \cdot d = \\ &(v - h) \cdot \frac{a}{a + h_1(i^*)} \cdot (h_1(i^*) + a) \cdot P + ha \cdot P = \\ &(v - h) \cdot a \cdot P + ha \cdot P = va \cdot P = v \cdot d = w。 \end{aligned}$$

故有

$$h' = h_2(m, w') = h_2(m, w) = h。$$

由于 $h = h', R = R'$,故有 $R_x = R'_x, r = r'$。

(2)对重复穿刺的用户私钥 $sk' \leftarrow Puncture(pp, sk, m')$ 和任意签名消息 $m \in \mathcal{M}$,有 $Verify(pp, pk, Sign(pp, sk', m), m) \rightarrow 0$ 成立。具体地,若消息 m' 被穿刺,则由布隆过滤器的正确性,可知 $Check(H, T, m') = 1$,这意味着用于对以 m' 为前缀的消息进行签名的所有密钥都已被删除。因此,具有前缀为 m' 的消息 m 的签名失败。

5.2　安全性分析

定理 1　如果挑战者 $\mathcal{C}$ 能以不可忽略的优势解决 ECDLP 问题,则敌手 $\mathcal{A}$ 能以不可忽略的优势在游戏 $Ga^{EUF\text{-}CMA}_{\mathcal{A} \leftrightarrow \mathcal{C}}$ 中获胜,并通过构造一个多项式时间算法来打破所提 SM2-PS 方案的 EUF-CMA。

证明　假设 $\mathcal{A}$ 打破了所提 SM2-PS 方案的 EUF-CMA,则存在可解决 ECDLP 实例 (Q, aQ) 的 $\mathcal{C}$ 可计算出 a。

(1)初始化阶段。$\mathcal{C}$ 执行系统初始化算法生成系统公开参数 pp,并将 pp 发送给 $\mathcal{A}$。

(2)$\mathcal{A}$ 可以自适应地执行下列操作,同时,$\mathcal{C}$ 创建列表 $\mathcal{L}_{H_2}$、$\mathcal{L}_{sk}$,初始时均为空集。

(i)h_2 查询 $\mathcal{Q}_{h_2}$。$\mathcal{C}$ 维持存储格式为 $\{ID, m, w, h\}$ 的列表 $\mathcal{L}_{H_2}$,接收到 $\mathcal{A}$ 的 $\mathcal{Q}_{h_2}$ 后,$\mathcal{C}$ 验证 $ID \overset{?}{\in} \mathcal{L}_{h_2}$,若 $ID \in \mathcal{L}_{h_2}$,则返回 h 给 $\mathcal{A}$ 以作为响应;否则,选择 $z_1 \in_R \mathbb{Z}_p^*$ 以作为响应,并将 $\{ID, m, w, z_1\}$ 添加到 $\mathcal{L}_{H_2}$。

(ii)私钥提取查询 $\mathcal{Q}_{sk}$。$\mathcal{C}$ 维持存储格式为 $\{ID, sk, pk\}$ 的列表 $\mathcal{L}_{sk}$,接收到 $\mathcal{A}$ 的 $\mathcal{Q}_{sk}$ 后,$\mathcal{C}$ 验证 $ID \overset{?}{\in} \mathcal{L}_{sk}$,若 $ID \in \mathcal{L}_{sk}$,则返回 sk 给 $\mathcal{A}$ 以作为响应;否则,执行下列操作。

※选择 $z_2, z_3 \in_R \mathbb{Z}_p^*$,运行 $Gen(z_2, z_3)$ 算法生成 $(\{H'_j\}_{j \in [z_3]}, T')$,其中,$T = 0^{z_2}$。

※选择 $z_4 \in_R \mathbb{Z}_p^*$,计算 $sk'_i = \dfrac{z_4}{z_4 + h'_1(i)}, pk'_i = sk'_i \cdot P$ 和 $d' = z_4 \cdot P$。

※输出公钥 $pk = (\{H'_j\}_{j \in [z_3]}, pk'_i, d')$ 及私钥 $sk' = (T', \{sk'_i\}_{i \in [z_2]})$。

※返回 sk' 给 $\mathcal{A}$ 以作为响应,并将 $\{ID, sk', pk'\}$ 添加到 $\mathcal{L}_{sk}$。

(iii)签名查询 $\mathcal{Q}_{sig}$。接收到 $\mathcal{A}$ 关于 m 的 $\mathcal{Q}_{sig}$ 后,

C 运行 $Check(H,T,m')$,判断其值为 1/0。若 $Check(H,T,m')\to 1$,则输出 $\perp$;否则,若 $Check(H,T,m')\to 0$,则至少存在一个 $i^*\in\{i_1,\cdots,i_k\}$,使得 $sk_{i^*}\neq\perp$,其中,$i_j=H_j(m')(j\in[k])$。若 $sk_{i^*}\in\mathcal{L}_{sk}$,则运行 *Sign* 算法以生成 m 的签名 σ;否则,执行下列操作。

※选择 $z_5,z_6\in\mathbb{Z}_p^*$,计算

$$\begin{cases}R'=z_5\cdot P=(R'_x,R'_y),\\ w'=z_6\cdot d,\\ h'=h_2(m,w'),\\ z'=(z_6-h')\cdot sk'_{i^*}。\end{cases}$$

※计算 $\{r',s'\}$,若 $s=0,r=0$ 或 $r+u=0$,则重新选择 z_5,否则输出签名 $\sigma'=(r',s',h',z',i^{*\prime})$。

$$\begin{cases}r'=(R'_x+h')\bmod(p),\\ s=(1+sk'_{i^*})^{-1}(z_5-r'\cdot sk'_{i^*})\bmod(p)。\end{cases}$$

(iv)私钥穿刺查询 $\mathcal{Q}_{punc}$。接收到 $\mathcal{A}$ 关于 sk 的 $\mathcal{Q}_{punc}$ 后,$\mathcal{C}$ 运行 $Update(\{H_j\}_{j\in[k]},T,str)$ 算法以进行更新,输出 T'。此外,对于每一个 $i\in[\ell]$,定义

$$sk'_i=\begin{cases}sk_i, & T'[i]=0,\\ \perp, & \text{otherwise},\end{cases}$$

然后输出更新后的私钥 $sk'=(T',\{sk'_i\}_{i\in[\ell]})$。

(3)伪造阶段。结束上述查询之后,假设 A 能以不可忽略的概率赢得游戏 $Ga^{EUF\text{-}CMA}_{\mathcal{A}\leftrightarrow\mathcal{C}}$,这意味着它能根据分叉引理[19]伪造出两个合法的签名 σ^* 和 $\sigma^{*\prime}$。若签名 σ^* 和 $\sigma^{*\prime}$ 是有效的签名,则有下列公式成立。

$$r^*=h^*+s^*\cdot x_P+t^*\cdot sk_i\cdot x_P, \quad (1)$$

$$r^{*\prime}=h^*+s^{*\prime}\cdot x_P+t^{*\prime}\cdot sk_i\cdot x_P。 \quad (2)$$

由式(1)和式(2),可得到

$$(r^*-r^{*\prime})=(s^*-s^{*\prime})x_P+(t^*-t^{*\prime})sk_i\cdot x_P。$$

C 输出 $((r^*-r^{*\prime})-(s^*-s^{*\prime})x_P)(t^*-t^{*\prime})^{-1}x_p^{-1}$ 作为 ECDLP 问题的解,而这与 ECDLP 问题是困难的相矛盾。因此,SM2-PS 方案是 EUF-CMA。 □

6 性能分析与对比

本文的模拟实验均在配备了 64 位操作系统的电脑(Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19GHz, RAM 16.0GB)上使用 Python 的 Hashlib 和 Pymcl 等模块进行。具体地,首先对每个常见的操作(如哈希、点乘等)执行 1 000 次取平均值来评估其运行效率。此外,还总结了 SM2-PS 方案中相关组件的大小,其结果如表 1 所示。紧接着,通过与 PS 方案[13]和 IBPS 方案[17]在密钥生成、消息签名和签名验证中的计算开销以及公私钥和签名的存储开销来评估所提方案的性能。

表 1 运行时间及组件大小

Table 1 Runtimes and component sizes

符号	说明	时间/ms
T_h	h 运算的时间	0.002 71
T_H	布隆过滤器哈希 H 运算的时间	0.002 25
T_{az}	$\mathbb{Z}_p^*$ 中元素加法运算的时间	0.001 01
T_{mz}	$\mathbb{Z}_p^*$ 中元素乘法运算的时间	0.002 57
T_{ag}	$\mathbb{G}$中元素加法运算的时间	0.000 19
T_{mg}	$\mathbb{G}$中元素乘法运算的时间	0.000 21
T_m	点乘运算的时间	0.000 77
T_e	双线性运算的时间	0.122 41
$T_{\exp}$	指数运算的时间	0.001 08
符号	说明	大小/B
$\lvert\mathbb{Z}_p\rvert$	$\mathbb{Z}_p$的比特长度	32
$\lvert\mathbb{G}\rvert$	$\mathbb{G}$的比特长度	64
$\lvert H_j\rvert$	H_j 的比特长度	1

6.1 计算开销

在本文方案中,密钥生成阶段通过 ℓ 个 h 运算、ℓ 个$\mathbb{Z}_p$中的加法运算、ℓ 个$\mathbb{Z}_p$中的乘法运算和 2 个点乘运算来生成密钥,其计算开销为 $\ell(T_h+T_{az}+T_{mz})+2T_m$(0.127 ms);签名方通过 1 个 h 运算、4 个$\mathbb{Z}_p$中的加法运算、3 个$\mathbb{Z}_p$中的乘法运算和 2 个点乘运算来生成消息签名,其计算开销为 $T_h+4T_{az}+3T_{mz}+2T_m$(0.016 ms);而验证方通过 2 个 h 运算、k 个 H 运算、2 个$\mathbb{Z}_p$中的加法运算、3 个$\mathbb{G}$中的加法运算和 5 个点乘运算来进行签名验证,其计算开销为 $2T_h+kT_H+2T_{az}+3T_{ag}+5T_m$(0.034 ms)。

在 PS 方案[13]中,密钥生成阶段通过 ℓ 个 h 运算、ℓ 个$\mathbb{Z}_p$中的加法运算、ℓ 个$\mathbb{Z}_p$中的乘法运算、$\ell+1$ 个点乘运算和 1 个双线性运算来生成密钥,其计算开销为 $\ell(T_h+T_{az}+T_{mz}+T_m)+T_m+T_e$(0.264 ms);签名方通过 1 个 h 运算、1 个$\mathbb{Z}_p$中的加法运算、1 个点乘运算和 1 个指数运算来生成消息签名,其计算开销为 $T_h+T_{az}+T_m+T_{\exp}$(0.006 ms);而验证方通过 2 个 h 运算、k 个 H 运

算、1 个$\mathbb{G}$中的加法运算、1 个$\mathbb{G}$中的乘法运算、1 个点乘运算、1 个指数运算和 1 个双线性运算来进行签名验证,其计算开销为 $2T_h + kT_H + T_{ag} + T_{mg} + T_m + T_{\exp} + T_e$(0.153 ms)。

在 IBPS 方案[17]中,密钥生成阶段通过 ℓ 个 h 运算、2ℓ 个$\mathbb{Z}_p$中的乘法运算、ℓ 个点乘运算和 $3\ell+2$ 个指数运算来生成密钥,其计算开销为 $\ell(T_h + 2T_{mz} + T_m + 3T_{\exp}) + 2T_{\exp}$(0.239 ms);签名方通过 1 个$\mathbb{Z}_p$中的乘法运算、1 个点乘运算和 2 个指数运算来生成消息签名,其计算开销为 $T_{mz} + T_m + 2T_{\exp}$(0.006 ms);而验证方通过 1 个 h 运算、2 个$\mathbb{Z}_p$中的加法运算和 5 个双线性运算来进行签名验证,其计算开销为 $T_h + 2T_{az} + 5T_e$(0.617 ms)。

详细的对比结果如表 2 所示。在密钥生成阶段,所提 SM2-PS 方案优于对比方案(IBPS 和 PS),计算效率分别提高了 46.80% 和 51.83%;在消息签名阶段,SM2-PS 方案的计算开销相比于 IBPS 和 PS 分别增加了 65.63% 和 65.19%,这主要是由于所提 SM2-PS 方案采用了国产化的数字签名算法,导致计算开销略有增加;在签名验证阶段,SM2-PS 的计算开销优于对比方案(IBPS 和 PS),效率分别提高了 94.43% 和 77.48%。综上所述,尽管消息签名阶段的开销有所增加,所提 SM2-PS 方案在密钥生成和签名验证阶段表现出显著的计算效率优势,总体上在计算开销上具有一定的优势。

表 2　计算和存储开销的对比

Table 2　Comparison of computational and storage overhead

方案	计算开销			存储开销		
	密钥生成	消息签名	签名验证	私钥	公钥	签名
PS 方案[13]	$\ell(T_h + T_{az} + T_{mz} + T_m) + T_m + T_e$	$T_h + T_{az} + T_m + T_{\exp}$	$2T_h + kT_H + T_{ag} + T_{mg} + T_m + T_{\exp} + T_e$	$\ell\lvert\mathbb{G}\rvert + \ell\lvert\mathbb{Z}_p\rvert$	$2\lvert\mathbb{G}\rvert + k\lvert H_j\rvert$	$2\lvert\mathbb{Z}_p\rvert + \lvert\mathbb{G}\rvert$
IBPS 方案[17]	$\ell(T_h + 2T_{mz} + T_m + 3T_{\exp}) + 2T_{\exp}$	$T_{mz} + T_m + 2T_{\exp}$	$T_h + 2T_{az} + 5T_e$	$\ell\lvert\mathbb{Z}_p\rvert + (\ell+2)\lvert\mathbb{G}\rvert + k\lvert H_j\rvert$	—	$\lvert\mathbb{Z}_p\rvert + 4\lvert\mathbb{G}\rvert$
本文方案	$\ell(T_h + T_{az} + T_{mz}) + 2T_m$	$T_h + 4T_{az} + 3T_{mz} + 2T_m$	$2T_h + kT_H + 2T_{az} + 3T_{ag} + 5T_m$	$2\ell\lvert\mathbb{Z}_p\rvert$	$2\lvert\mathbb{G}\rvert + k\lvert H_j\rvert$	$5\lvert\mathbb{Z}_p\rvert$

注:ℓ 表示布隆过滤器的长度,k 表示布隆过滤器中无偏哈希函数的数量。

6.2　存储开销

在本文方案中,其私钥由两部分组成:布隆过滤器的比特数组 T 和 ℓ 个 $sk_i = \dfrac{a}{a + h_1(i)}$,其中,$T$,$sk_i \in \mathbb{Z}_p$,故其私钥 sk 的存储开销为 $2\ell\lvert\mathbb{Z}_p\rvert$(1.250 KB);其公钥为 $pk = (\{H_j\}_{j\in[k]}, pk_i, d)$,其中,$pk_i$,$d \in \mathbb{G}$,故公钥 pk 的存储开销为 $2\lvert\mathbb{G}\rvert + k\lvert H_j\rvert$(0.135 KB);而在消息签名阶段中,其生成的签名值为 $\sigma = (r, s, h, z, i^*)$,其中,$r, s, h, z, i^* \in \mathbb{Z}_p$,故签名 σ_i 的存储开销为 $5\lvert\mathbb{Z}_p\rvert$(0.156 KB)。

在 PS 方案[13]中,其私钥由两部分组成:布隆过滤器的比特数组 T 和 ℓ 个 $sk_i = \dfrac{s}{s + h_1(i)} \cdot P_1$,其中,$T \in \mathbb{Z}_p$,$sk_i \in \mathbb{G}$,故其私钥 sk 的存储开销为 $\ell\lvert\mathbb{G}\rvert + \ell\lvert\mathbb{Z}_p\rvert$(1.875 KB);其公钥为 $pk = (\{H_j\}_{j\in[k]}, g, P_{pub})$,其中,$g, P_{pub} \in \mathbb{G}$,故公钥 pk 的存储开销为 $2\lvert\mathbb{G}\rvert + k\lvert H_j\rvert$(0.135 KB);而在消息签名阶段中,其生成的签名值为 $\sigma = (h, S, i_{j^*})$,其中,$h, i_{j^*} \in \mathbb{Z}_p$,$S \in \mathbb{G}$,故签名 σ_i 的存储开销为 $2\lvert\mathbb{Z}_p\rvert + \lvert\mathbb{G}\rvert$(0.125 KB)。

在 IBPS 方案[17]中,其私钥由 4 部分组成:布隆过滤器的比特数组 T、布隆过滤器中无偏哈希函数 H、ℓ 个 $sk_i = g_2^{\alpha}(u_0\prod_{j=1}^{\gamma}u_j^{\theta_j})^{r_1}\widetilde{H}(i)^{r_2}$、$k_1 = g^{r_1}$ 和 $k_2 = g^{r_2}$,其中,$T \in \mathbb{Z}_p$,且 $sk_i, k_1, k_2 \in \mathbb{G}$,故其私钥 sk 的存储开销为 $\ell\lvert\mathbb{Z}_p\rvert + (\ell+2)\lvert\mathbb{G}\rvert + k\lvert H_j\rvert$(2.010 KB);而在消息签名阶段中,其生成的签名值为 $\sigma = (i^*, \sigma_0, \sigma_1, \sigma_2, \sigma_3)$,其中,$i^* \in \mathbb{Z}_p$,且 $\sigma_0, \sigma_1, \sigma_2, \sigma_3 \in \mathbb{G}$,故签名 σ_i 的存储开销为 $\lvert\mathbb{Z}_p\rvert + 4\lvert\mathbb{G}\rvert$(0.281 KB);由于该方案使用系统参数进行验证,故在此不考虑其存储开销。

详细的对比结果如表 2 所示。在公钥的存储

开销上,SM2-PS 方案的存储开销与 PS 方案一致,均为 0.135 KB;在私钥的存储开销上,SM2-PS 方案的开销明显优于 IBPS 方案和 PS 方案;在签名的存储开销上,SM2-PS 方案的开销要高于 PS 方案,而低于 IBPS 方案,这是因为所提 SM2-PS 方案在原有 SM2 签名算法中增加了穿刺功能,导致存储开销有所增加。综上所述,尽管所提 SM2-PS 方案在签名存储开销上略有增加,但在私钥存储上表现出显著优势,使得方案总体上在存储开销上具有一定的优势。

7 总 结

本文提出了一种基于 SM2 的可穿刺签名方案(SM2-PS),实现了在私钥泄露的情况下也能保证之前生成签名的安全性。该方案支持对特定消息进行穿刺,从而动态地更新私钥,避免了私钥泄露后对历史签名的威胁。此外,SM2-PS 方案结合了布隆过滤器的优势,使得穿刺操作只需要一次删除关键元素即可完成。安全性分析表明,SM2-PS 方案在 ECDLP 问题下存在不可伪造性。性能分析及对比表明,SM2-PS 方案在计算和存储开销上具有一定的优势。综上所述,SM2-PS 方案在安全性和效率上均具有一定优势,适用于需要长期保护数字签名安全的应用场景,如金融、政府和物联网等领域。本文为 SM2 签名算法的进一步发展提供了重要的理论支持,具有较广泛的应用前景。

参考文献:

[1] 国家密码管理局. 椭圆曲线公钥密码算法:GM/T2003—2012 SM2[S]. 北京: 中国标准出版社, 2010:24-25.

[2] 唐卫中, 张大伟, 佟晖. 基于 SM2 的无证书盲签名方案[J]. 计算机应用研究, 2022, 39(2): 552-556.

[3] 白雪, 秦宝东, 郭瑞, 等. 基于 SM2 的两方协作盲签名协议[J]. 网络与信息安全学报, 2022, 8(6): 39-51.

[4] 范青, 何德彪, 罗敏, 等. 基于 SM2 数字签名算法的环签名方案[J]. 密码学报, 2021, 8(4): 710-723.

[5] 包子健, 何德彪, 彭聪, 等. 基于 SM2 数字签名算法的可否认环签名[J]. 密码学报, 2023, 10(2): 264-275.

[6] Anderson R. Invited Lecture at the 4th ACM Conference on Computer and Communications Security, April 01-05, 1997[C]. New York: ACM, 1997.

[7] Bellare M, Miner S K. Advances in Cryptology — CRYPTO'99, August 15-19, 1999[C]. Berlin: Springer, 1999.

[8] Itkis G, Reyzin L. Advances in Cryptology — CRYPTO 2001, August 19-23, 2001[C]. Berlin: Springer, 2001.

[9] Abdalla M, Reyzin L. Advances in Cryptology-ASIACRYPT 2000, December 03-07, 2000[C]. Berlin: Springer, 2000.

[10] Lee J, Kim J, Oh H. Forward-secure multi-user aggregate signatures based on zk-SNARKs[J]. IEEE Access, 2021, 9: 97705-97717.

[11] 韦性佳, 芦殿军. 基于中国剩余定理的前向安全的聚合签名方案[J]. 计算机技术与发展, 2021, 31(4): 137-141.

[12] Bellare M, Stepanovs L, Waters B. Advances in Cryptology-EUROCRYPT 2016, May 08-12, 2016[C]. Berlin: Springer, 2016.

[13] Li X Y, Xu J, Fan X, et al. Puncturable signatures and applications in proof-of-stake blockchain protocols[J]. IEEE Transactions on Information Forensics and Security, 2020, 15: 3872-3885.

[14] Bloom B H. Space/time trade-offs in hash coding with allowable errors[J]. Communications of the ACM, 1970, 13(7): 422-426.

[15] Sanjam G, Craig G, Shai H, et al. 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, October 27-29, 2013[C]. Piscataway: IEEE, 2013.

[16] Shai H, Yuval I, Abhishek J, et al. Advances in Cryptology-ASIACRYPT 2017, December 03-07, 2017[C]. Berlin: Springer, 2017.

[17] 杨冬梅, 陈越, 魏江宏, 等. 基于身份的可穿刺签名方案[J]. 通信学报, 2021, 42(12): 17-26.

[18] Jiang M, Duong D H, Susilo W. Computer Security-ESORICS 2022, September 26-30, 2022[C]. Berlin: Springer, 2022.

[19] Pointcheval D, Stern J. Security arguments for digital signatures and blind signatures[J]. Journal of Cryptology, 2000, 13(3): 361-396.

【责任编辑: 卓祯雨】