A note on "industrial blockchain threshold signatures in federated learning for unified space-air-ground-sea model training"

Zhengjun Cao, Lihua Liu

Abstract. We show that the threshold signature scheme [J. Ind. Inf. Integr. 39: 100593 (2024)] is insecure against forgery attack. An adversary can find an efficient signing algorithm functionally equivalent to the valid signing algorithm, so as to convert the legitimate signature (sig, s, r_x) of message m into a valid signature (sig, s, r'_x) of any message m'.

Keywords: Signing algorithm; verification algorithm; threshold signature; forgery attack.

1 Introduction

The purpose of a digital signature is to provide a means for an entity to bind its identity to a piece of information. The process of signing transforms the message and some secret information held by the entity into a tag called a signature [6]. A verification algorithm is a method for verifying that a digital signature is authentic, i.e., was indeed created by the specified entity.

Threshold signature extends the general signature to the scenario of multiple signers. These signers collaboratively generate a signature for a same message. Gennaro et al. [4, 5] presented some multiparty threshold ECDSA schemes. Canetti et al. [2] discussed the problem of threshold ECDSA with identifiable aborts. Wong et al. [7] investigated the real threshold ECDSA. Bouez and Singh [1] proposed a threshold ECDSA without roll call.

Very recently, Chen et al. [3] have designed a threshold signature scheme by integrating an SM2 digital signature in the ISO/IEC standards with the t-party multiplication protocol. Though the signature scheme is interesting, we find it is insecure. An adversary can find an efficient signing algorithm functionally equivalent to the valid signing algorithm, even though he cannot compute the private key information of any signer.

2 Review of Chen et al.'s signature scheme

The Chen et al.'s protocol [3] aims to provide a threshold signing scheme in industrial blockchain-based federated learning. It assumes that there are n signature nodes, with the threshold value

Z. Cao, Department of Mathematics, Shanghai University, Shanghai, 200444, China.
L. Liu, Department of Mathematics, Shanghai Maritime University, Shanghai, 201306, China.
Email: liulh@shmtu.edu.cn

t. That means any t nodes can collaboratively generate a valid signature for the target message m. The involved symbols and their descriptions are listed below (Table 1).

p

Table 1: Symbols and descriptions

symbol	description
SN_i	signature node
VN_i	verification node
$\boldsymbol{P}$	t of n signature nodes
p, q	prime numbers
$E(F_p)$	elliptic curve
$\mathbb{G}$	elliptic curve group with generator G

The threshold signature scheme can be restated as follows (Table 2).

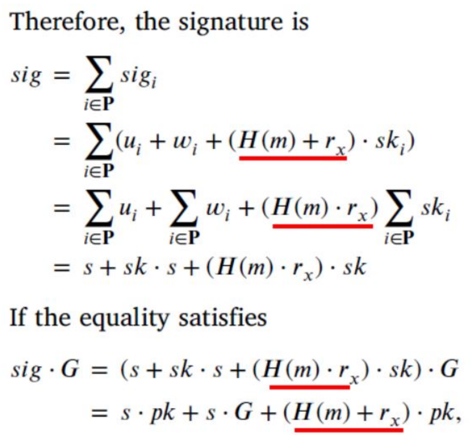
Table 2: The Chen et al.'s threshold signature scheme

Setup. Let $E(F_p)$ be an elliptic curve on the finite field F_p, defined by $y^2 = x^3 + ax + b \bmod p$, where p is a big prime number. $\mathbb{G}$ is an additive cyclic elliptic curve group with a generator G of prime order q. $H : \{0,1\}^* \to Z_q^*$ is a hash function. The system parameter is $param = \{p, q, a, b, G, H\}$.
Distributed key generation. Each node SN_i generates a random polynomial $f_i()$ of degree $t-1$, and sends the share $f_i(j)$ to the node SN_j. After receiving the shares from other nodes, SN_i computes $f(i) = \sum_{j\in[n]} f_j(i), \quad L_i = f(i)\cdot G$. Then generate the public/private key pair (pk_i, sk_i), such that $sk_i = \mathrm{k}_i^{\boldsymbol{P}}(0)\cdot f(i), \quad pk_i = sk_i \cdot G$, where $\mathrm{k}_i^{\boldsymbol{P}}(0)$ is the Lagrange coefficient calculated by SN_i that it is reconstructing sk_i with the parties in $\boldsymbol{P}$. The signature nodes compute a shared public key $pk = \sum_{i\in\boldsymbol{P}} \mathrm{k}_i^{\boldsymbol{P}}(0)\cdot L_i$, where $\lvert\boldsymbol{P}\rvert = t$.
Sign. Each node SN_i picks $s_i \in Z_q$ and invokes *t-party multiplication protocol* to compute $\sum_{i\in\boldsymbol{P}} u_i = \prod_{i\in\boldsymbol{P}} s_i = s$. For any two nodes SN_i and SN_j, they invoke the *two-party multiplication protocol* to compute $w_i^{j,1} + w_j^{i,1} = sk_i \cdot u_j, \quad w_i^{j,2} + w_j^{i,2} = sk_j \cdot u_i$. Here $\{sk_i, u_i\}$ and $\{sk_j, u_j\}$ are the inputs of SN_i and SN_j, respectively. They receive $\{w_i^{j,1}, w_i^{j,2}\}$ and $\{w_j^{i,1}, w_j^{i,2}\}$ as outputs, respectively. Then each node SN_i computes $w_i = sk_i \cdot u_i + \sum_{j\in\boldsymbol{P}\setminus\{i\}} (w_i^{j,1} + w_i^{j,2})$, and the pre-signature segment $sig_i = u_i + w_i + (H(m) + r_x)\cdot sk_i$, where m is the message to be signed.
Verify. All nodes calculate $sig = \sum_{i\in\boldsymbol{P}} sig_i$ and check if $sig\cdot G = s\cdot pk + s\cdot G + (H(m) + r_x \cdot pk)$. If true, accept the signature. Otherwise, reject it.

3 Analysis of the signature scheme

3.1 Some typos

⋄ As we see, the sum $H(m) + r_x$ and the product $H(m) \cdot r_x$ are inconsistently used in the core equations for signing and verification (see Fig.1).



Therefore, the signature is

$$\begin{aligned} sig &= \sum_{i \in \mathbf{P}} sig_i \\ &= \sum_{i \in \mathbf{P}} (u_i + w_i + \underline{(H(m) + r_x)} \cdot sk_i) \\ &= \sum_{i \in \mathbf{P}} u_i + \sum_{i \in \mathbf{P}} w_i + \underline{(H(m) \cdot r_x)} \sum_{i \in \mathbf{P}} sk_i \\ &= s + sk \cdot s + (H(m) \cdot r_x) \cdot sk \end{aligned}$$

If the equality satisfies

$$\begin{aligned} sig \cdot G &= (s + sk \cdot s + \underline{(H(m) \cdot r_x)} \cdot sk) \cdot G \\ &= s \cdot pk + s \cdot G + \underline{(H(m) + r_x)} \cdot pk, \end{aligned}$$

Figure 1: The screenshot for some typos (page 8, Ref.[3])

To keep its consistency, the original verification equation

$$sig \cdot G = s \cdot pk + s \cdot G + (H(m) + r_x \cdot pk) \tag{1}$$

should be corrected as

$$sig \cdot G = s \cdot pk + s \cdot G + (H(m) + r_x) \cdot pk \tag{1'}$$

due to that

$$\begin{aligned} pk &= \sum_{i \in \boldsymbol{P}} \mathrm{k}_i^{\boldsymbol{P}}(0) \cdot L_i = \sum_{i \in \boldsymbol{P}} \mathrm{k}_i^{\boldsymbol{P}}(0) \cdot f(i) \cdot G \\ &= \sum_{i \in \boldsymbol{P}} sk_i \cdot G = \sum_{i \in \boldsymbol{P}} pk_i. \end{aligned}$$

⋄ In the standard SM-2 digital signature algorithm (page 4, [3]), the symbol r_x represents the x-coordinate of the point $R = k \cdot G = (r_x, y_x)$, where k is a random number picked by the singer. But in the threshold signature scheme, the value r_x is never specified. To ensure each node can complete the signing phase, it should introduce other mechanisms for generating the signature value r_x.

3.2 Insecure against forgery attack

The threshold signature scheme tries to extend the standard SM-2 signature to the scenario of multiple signers. In the original, the final signature σ consists of two values r and s (page 4, [3]), while the new has three values sig, s and r_x. We find the signature is not tightly bound to the signers' secret keys and signed message m. An adversary can forge signatures for any message. Concretely, the adversary can convert a legitimate signature (sig, s, r_x) of message m into a valid signature (sig, s, r'_x) of any message m'.

For example, given the message and signature $\{m, sig, s, r_x\}$ and the target message m', the adversary computes

$$r'_x = h(m) + r_x - h(m') \bmod q \tag{2}$$

where h is a public hash function and the prime number q is the order of underlying elliptic curve group. Both are available to the adversary.

We now show that the forged signature (sig, s, r'_x) of the message m' can pass the verification equation. Note that the legitimate signature (sig, s, r_x) satisfies that

$$sig \cdot G = s \cdot pk + s \cdot G + (H(m) + r_x) \cdot pk \tag{3}$$

Combining Eq.(2) and Eq.(3), we have

$$sig \cdot G = s \cdot pk + s \cdot G + (H(m') + r'_x) \cdot pk \tag{4}$$

Hence, the forged signature (sig, s, r'_x) of the message m' will be accepted by the verifier.

4 Conclusion

We show that the Chen et al.'s threshold signature scheme is insecure against forgery attack. We also corrected some typos in the original presentation. The findings in this note could be helpful for the future works on designing such threshold signature schemes.

References

[1] A. Bouez and K. Singh. One round threshold ECDSA without roll call. In Mike Rosulek, editor, *Proc. Topics in Cryptology - CT-RSA 2023 - Cryptographers' Track at the RSA Conference 2023, San Francisco, CA, USA, April 24-27, 2023, Proceedings*, volume 13871 of *Lecture Notes in Computer Science*, pages 389–414. Springer, 2023.

[2] R. Canetti, R. Gennaro, S. Goldfeder, N. Makriyannis, and U. Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *Proc. CCS'20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1769–1787. ACM, 2020.

[3] J. Chen, Z. Wang, G. Srivastava, T. A. Alghamdi, F. Khan, S. Kumari, and H. Xiong.

Industrial blockchain threshold signatures in federated learning for unified space-air-ground-sea model training. *J. Ind. Inf. Integr.*, 39:100593, 2024.

[4] J. Doerner, Y. Kondi, E. Lee, and A. Shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *Proc. 2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 1051–1066. IEEE, 2019.

[5] R. Gennaro and S. Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proc. 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1179–1194. ACM, 2018.

[6] A. Menezes, P. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, USA, 1996.

[7] H. Wong, J. Ma, H. Yin, and S. Chow. Real threshold ECDSA. In *Proc. 30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023.* The Internet Society, 2023.