A Cryptanalysis on “Compact Multiple Attribute-Based Signatures With Key Aggregation and Its Application”

Ziyuan Liu1, a), Xingbo Dong1, b), Zhe Jin1, c), Syh-Yuan Tan2, d),

Swee-Huay Heng2, e)

*1Anhui Provincial Key Laboratory of Secure Artificial Intelligence, Anhui Provincial International Joint Artificial Intelligence, Anhui University, Hefei, 230093, Anhui, China.*

*2Centre for Intelligent Cloud Computing, CoE for Advanced Cloud, Faculty of Information Science and Technology, Multimedia University, Jalan Ayer Keroh Lama, 75450 Bukit Beruang, Melaka, Malaysia.*

*e)Corresponding author: shheng@mmu.edu.my*

*a)luckyzy98@gmail.com*

*b)xingbo.dong@ahu.edu.cn*

*c)jinzhe@ahu.edu.cn*

*d)sytan@mmu.edu.my*

**Abstract.** The multi-signature scheme is a crucial component in blockchain consensus protocols enabling the establishment of trust among signers up to a certain degree as defined by a predetermined system threshold. It is an ongoing challenge to reduce the storage, bandwidth and computational cost of a multi-signature scheme. In 2022, Guo et al. proposed a multi-attribute-based signature scheme with key aggregation and proved its existential unforgeability under chosen message attack based on the computational Diffie-Hellman assumption. In this cryptanalysis, we demonstrate that Guo et al.’s scheme fails to achieve unforgeability under both the key-only attack and the known-message attack. Moreover, if an adversary in a no-message attack is allowed to corrupt at least one signer, the scheme can be completely broken. We identify the root cause and discuss several attempted fixes though to no avail.

# Introduction

In distributed systems such as blockchain, the integration of multiple signatures [1, 2, 3] and attribute-based signatures [4, 5, 6, 7] offers significant advantages compared to using either scheme independently. Traditional multi-signature schemes primarily focus on preventing single points of failure and achieving consensus through multiple-key endorsements, but they lack fine-grained access control and transaction anonymity. Conversely, stand-alone attribute-based signatures enable the verification of specific attributes to preserve privacy, yet their applicability in multi-party authentication and batch verification is limited. By combining the two approaches, the resulting scheme, namely, Multiple Attribute-Based Signatures [8] capitalizes on the strengths of multi-signatures in defending against attacks and ensuring transaction validity, while employing attribute verification to facilitate selective disclosure of information. This integration reduces data redundancy and computational overhead, thereby meeting the comprehensive requirements for security, efficiency, and privacy protection in large-scale blockchain applications [9, 10, 11, 12].

In 2022, Guo and co-authors proposed a multi-attribute signature framework with key aggregation (M-ABS-KA) [8] that supports a flexible threshold predicate. Their design combines individual public keys into a single aggregated key and produces a succinct multi-signature that can be verified collectively, yielding much faster batch verification compared to verifying each signature separately. Notably, both the size of the combined signature and the length of the aggregated public key remain constant, regardless of how many signers or attributes are involved.

The security of M-ABS-KA is demonstrated under the assumption that the Computational Diffie–Hellman problem is hard, within a formal model that includes three oracles: the *Key-Extraction* oracle, the *Key-Aggregation-Extraction* oracle, and the *MultiSign* oracle. These oracles respectively model an adversary’s ability to corrupt users, to compute aggregated public keys, and to request signatures on chosen messages and attribute predicates.

However, we discover four security vulnerabilities in their scheme. We list down the attacks, which exploit the vulnerabilities, in the ascending difficulty level:

* Key-Only Attack 1 - we show that without access to any of the oracles, an adversary who knows only the public parameters can mount a key-only attack to forge a multi-signature for any message and predicate of its choice.
* Key-Only Attack 2 - an attacker can achieve the same goal as above but it has access to the *Key-Aggregation-Extraction* oracle.
* Known-Message Attack - if an attacker has access to the *MultiSign* oracle in addition to the ability in key-only attacks, it can forge a signature for any on behalf.
* No-Message Attack - if an attacker has access to the *Key-Extraction* oracle in addition to the ability in the key-only attacks, it can achieve a total break by extracting the authority's master secret key *x* that allows her to act as the authority and subsequently impersonate all users under the scheme

An oracle represents a black-box input source available to the adversary. For example, a signing oracle describes any method such as dustbin digging, sniffing, spoofing, or unauthorized access via a hacked user account, by which an adversary can obtain a signature. Similarly, a key-registration oracle abstracts the process of registering or injecting public-key material, whether through a rogue certificate request or a compromised registration service. This abstraction extends to other oracles used in our attacks, including *Key-Aggregation-Extraction*, *MultiSign*, and *Key-Extraction* oracles.

The remainder of this paper is structured as follows. Section II provides a concise overview of the mathematical foundations underlying Guo et al.’s M-ABS-KA scheme. In Section III, we detail the proposed attacks. Finally, Section IV offers a summary and compares the depth and nuances of the different attack types.

# Preliminaries

## Bilinear Map

Assume and are two different cyclic groups of prime order , and let and be generators of . We call a bilinear map admissible when it satisfies the following properties:

* Bilinearity: Applying exponents to the inputs in any order yields the same pairing result. Concretely, for all and any , .
* Non-degeneracy: The pairing does not collapse nontrivial group elements to the identity in . In particular, is the unit element of .
* Efficiently Computable: There exists a polynomial-time algorithm that, given any pair , outputs .

## Lagrange Interpolation Polynomial

The Lagrange interpolant is the unique polynomial of degree at most that exactly fits the data points for . It can be written as where each basis polynomial is as in Equation (1).

(1)

## Notations And Description For The Paper

We list the related notations in Table 1 and recall Guo et al.'s M-ABS-KA scheme [8]..

**Setup**: On input the security parameter , the trusted authority (TA) first fixes the attribute universe . It also chooses a default attribute set of size and designates a distinguished attribute . Let be a bilinear pairing. The TA picks a generator and samples a random exponent . It also selects a random group element and computes and Let , outputs as the public parameters, while the master secret key is .

**KeyGen**: The authority then selects a random polynomial of degree such that . For each attribute in the default set , it picks a random scalar to compute for each , the secret key and the public key as shown in Equations (2) to (5):

(2)

(3)

(4)

(5)

**TABLE 1.** List of notations.

|  |  |
| --- | --- |
| **Notations** | **Descriptions** |
| △ | The Lagrange coefficient. |
| *U* | The attribute set in universe. |
|  | The default attribute set. |
|  | The attribute set of a valid signer. |
|  | The subset of the intersection of *S* and *S*∗ . |
|  | The random default attribute set. |
|  | The hash function for *i* = 0, 1, 2. |
|  | The public key of a signer. |
|  | The forged public key of a attacker. |
|  | The secret key of a signer. |
|  | The predicate in attribute-based signature scheme. |
|  | The generated aggregate signature. |
|  | Attacker’s forged aggregate signature. |
|  | The generated aggregate public key. |
|  | Attacker’s forged aggregate public key. |

**MultiSign**: When creating a signature on the message with respect to the threshold predicate (such that ), specifies the threshold and denotes the valid signer’s attribute set. Then, the signer first randomly selects two subsets: of size and of size Let . They compute Equations (6) and (7):

(6)

(7)

followed by the multi-signature as . The signer outputs .

**KeyAgg**: Using the same exponent and Lagrange coefficients , the authority sets where and are as shown in Equations (8) and (9):

(8)

(9)

**MultiSignVer**: To validate under threshold , the verifer checks whether Equation (10) holds:

(10)

If the equality holds, output ; otherwise, output .

# cryptanalysis

In this part, we show how to break the unforgeability of Guo’s M-ABS-KA scheme by mounting one of our two key-only attacks or a known-message attack. We also present a no-message attack that can recover the master secret key and achieves a total break.

## Key-Only Attack 1

In the first key-only attack, we consider the scenario of a malicious attacker which forges a signature on any message with the knowledge of public parameter only. With that said, after sees the authority public parameter, it can sign on behalf of all registered users on any message.

1. knows the public parameter but not the master secret key .
2. Before forging the aggregated signature on and the threshold predicate , simulates the public keys with as follows. randomly selects and crafts the public keys as shown in Equations (11) and (12):

(11)

(12)

1. simulates the aggregate signature as in Euqation (13):

(13)

to set the final signature as where and are crafted as in Equations (14) and (15):

(14)

(15)

where is the Lagrange coefficient while can be any attribute set of size desired by .

1. When the authority runs $KeyAgg$ algorithm, the resulted aggregated public key is as shown in Equations (16) and (17):

(16)

(17)

1. Finally, Let , passes to the verifier which always returns accept because Equation (18) holds:

(18)

## Key-Only Attack 2

We show that can also break the unforgeability with another key-only attack scenario that makes use of from the *Key-Aggregation-Extraction* oracle.

1. .
2. Then impersonates the attribute authorities to create user attribute aggregated public keys as follows. Let , given the s as in Equations (19) and (20):

(19)

(20)

from the *Key-Aggregation-Extraction* oracle, can compute: to simulate new s as in Equations (21) and (22):

(21)

(22)

1. Let , . forges a signature as specified in Equation (23):

(23)

where are the same as in the previous steps.

1. Let and , the verifier always accept because the Equation (24) holds:

(24)

## Known-Message Attack

In this attack, with the knowledge of a victim 's signature tuple , can forge a signature on any tuple . We explain how user forges successfully as follows.

1. We simply assume that is in possession of the public parameter. obtains 's signature () through the *MultiSign* oracle. Let a random nonce as specified in Equation (25):

(25)

can compute the value and subsequently the value .

1. Next, can freely choose and whose signature it wants to forge on. Let , can simulate the signature using Equation (26):

(26)

as the forgery.

1. The verifier always accepts because Equation (27) holds:

(27)

## No-Message Attack

In this attack, can corrupt users through *Key-Extraction* oracle. It can also be viewed as an internal attack carried out by a malicious user.

1. For any user, can issue a single *Key-Extraction* oracle query to obtain $-many secrets from Equation (28):

(28)

from his secret key as shown in Equation (29):

(29)

1. Let be the threshold, by Lagrange interpolation, the master secret key can be revealed as in Equation (30):

(30)

1. In the case where the M-ABS-KA scheme generates only a private key for a user , that is, each user shares the same random polynomial , queries *Key-Extraction* oracle for times and extracts as in the previous step.

It is obvious that once the master secret key *x* is leaked, the scheme is completely broken.

# discussion

The root cause of the above attacks is the lack of an authentication mechanism in the public key issuing protocol. This allows anyone, with only the public parameters, to forge a multi-signature on any message. With that being said, the trusted authority does not have a role in providing security guarantees, as demonstrated in the first attack. In the second attack , we assume access to the key aggregation service is provided exclusively by the trusted authority, without compromising its integrity. This demonstrates that the generation of an aggregated public key by a trusted authority does not improve security.

We are unable to repair the scheme due to a fundamental flaw in the key generation algorithm. Addressing the issues mentioned above would essentially entail redesigning the entire scheme which is out of the scope of this cryptanalysis. Nevertheless, we share some attempts that we have made.

A quick solution is to add Equation (31):

(31)

but it can only solve the key-only attack 1. The known-message attack and no-message attack are due to the insufficient protection of the master secret key . In this case, the known-message attack can bypass to forge signatures using known public parameters, while the total break under internal attack can even directly obtain . To solve the key-only attack 2 and the known-message attack, we can delete and slightly change the partial signature as in Equation (32):

(32)

with the original verification equation can be simplified to that shown in Equation (33):

(33)

However, this solution cannot avoid the key-only attack. A quick solution for solving the no-message attack is to delete the $sk\_{i,2}$ because it is not involved in the signature generation, but this solution cannot solve the key-only attacks and known-message attack.

# Acknowledgments

This work was supported in part by High-end Foreign Experts Recruitment Plan (G2023019010L) by the Ministry of Science and Technology China, in part by the National Natural Science Foundation of China (Nos. 62376003) and Anhui Provincial Natural Science Foundation (No. 2308085MF200), in part by the Telekom Malaysia Research \& Development Grant (RDTC/221045).

# References

1. K. Gai, S. Wang, H. Zhao, Y. She, Z. Zhang, and L. Zhu, *IEEE Transactions on Computational Social Systems* **10**, 2201–2213 (2023).
2. Z. Wang, *IEEE Internet of Things Journal* **12**, 4525–4535 (2025).
3. Z. Wang, S. Yiu, and L. Lan, *IEEE Transactions on Dependable and Secure Computing*, 1–15 (2025).
4. Y. Jiang, X. Xu, and F. Xiao, *IEEE Transactions on Network and Service Management* **19**, 3884–3895 (2022).
5. N. Wu, L. Xu, and L. Zhu, *Future Generation Computer Systems* **141**, 186–196 (2023).
6. S. Ling, K. Nguyen, D. H. Phan, K. H. Tang, H. Wang, and Y. Xu, “Fully Dynamic Attribute-Based Signatures for Circuits from Codes,” in *Public-Key Cryptography — PKC 2024*, Vol. 14601, edited by Q. Tang and V. Teague (Springer Nature Switzerland, Cham, 2024), pp. 37–73.
7. C. Delerablée, L. Gouriou, and D. Pointcheval, “Attribute-Based Signatures with Advanced Delegation and Tracing,” *CT-RSA 2024* (2024), major revision.
8. R. Guo, K. Li, X. Li, Y. Zhang, and X. Li, *IEEE Systems Journal* **16**, 3025–3035 (2022).
9. Z. He and P. Turner, *Applied Sciences* **12**, 3723 (2022).
10. D. B. Rawat, V. Chaudhary, and R. Doku, *Journal of Cybersecurity and Privacy* **1**, 4–18 (2020).
11. D. Ressi, R. Romanello, C. Piazza, and S. Rossi, *Journal of Network and Computer Applications* **225**, 103858 (2024).
12. T. Nguyen, H. Nguyen, and T. Nguyen Gia, *Journal of Network and Computer Applications* **226**, 103884 (2024).
13. V. K. Yadav, R. K. Yadav, S. Verma, and S. Venkatesan, *The Journal of Supercomputing* **78**, 2386–2416 (2022).