A Study on Multi signature for Blockchains

Gulab Das
Department of Mathematics, Govt. N. PG College of Science, Raipur, Chhattisgarh, India.

B.P. Tripathi
Department of Mathematics, Govt. N. PG College of Science, Raipur, Chhattisgarh, India.

Swati Verma
Department of Mathematics, School of Science, OP Jindal University, Raigarh, Chhattisgarh, India.

Anurag Verma
Department of Mathematics, Govt. N. PG College of Science, Raipur, Chhattisgarh, India.

Abstract. We suggest a research of authentication techniques in the Bitcoin ecosystem as a blockchain application in this article. First, from the standpoint of replacing the current banking system, the features provided by Bitcoin and its security requirements are studied. The blockchain use case of the Bitcoin ecosystem has current transaction authentication techniques that have been well researched in the literature. The topic of whether the common account and proxy notions of banking services are likewise feasible in Bitcoin is posed in light of this. According to many experts, there are solutions based on responsible subgroup multi-signature schemes and bilinear pairings. In this work, the aforementioned question is addressed using the aforementioned scheme. We have the chance to use the public key aggregation process with accountable subgroup multi-signature (ASM), which is built from Boneh, Lynn, and Shacham (BLS) signature schemes. This makes it possible for multiple users to jointly sign the same message, and only one public key is required to validate the signature. This method is quite easy to implement in Bitcoin and provides for significant cost savings when it comes to storing public keys in transaction scripts. Our methods can be applied in many other situations where multiple signatures are required, in addition to reducing the size of the Bitcoin blockchain. Both signature compression and public-key aggregation are supported by all of our constructions. Therefore, the verifier simply needs a brief multisignature, a brief aggregation of the public keys of the parties, and the message m to confirm that several parties signed a shared message m. Future scope of proposed research is that it could be applicable any other blockchain structure where multi- signature is to be used.

Keywords: Accountable Subgroup, Block Chain, Bitcoin, Multi signature, Accountable Subgroup Multi signature.

Received: 07.05.2023 **Revised: 12.09.2023** **Accepted: 15.11.2023**

1. Introduction

Some of the chances that are recognized in the financial system are available in the present Bitcoin system, which is employed as a payment mechanism. For instance, the common account in a banking sys- tem is analogous to multi-signature, which enables several users to sign Bitcoin transactions. In this essay, studies on how to use transaction scripts, such as Multisignature, more successfully and efficiently are addressed. Consider a scenario where n parties generate key pairs for a signature system separately. Sometime later all n parties want to sign the same message m. A multi-signature scheme [14, 18] is a protocol that enables the n signers to jointly generate a short signature σ on m so that σ convinces a verifier that all n parties signed m. Specifically, the verification algorithm is given as input then public keys, the message m, and the multi-signature σ. The algorithm either accepts or re- jects σ. The multi-signature σ should be short; its length should be in- dependent of the number of signers n. We define this concept more precisely in the next section, where we also present the standard security model for such

schemes [18]. Secure multi-signatures have been constructed from Schnorr signatures(e.g. [3]), from BLS signatures (e.g. [4]), and from many other schemes as discussed in Section 2.A more general concept called an aggregate signature scheme [6] lets each of the parties sign a different message, but all these signatures can be aggregated into a single short signature σ. This brief signature should once again persuade the validator that each signer has signed the intended message. Applications to Bitcoin Multi-signatures and aggregate signatures can be used to shrink the size of the Bitcoin blockchain [19]. In recent work, Maxwell, Poelstra, Seurin, and Wuille [18] suggest using multi-signatures to shrink the transaction data associated with Bitcoin Multi-signature addresses. Conceptually, a Multi-signature address is the hash of n public keys pk1, .., pkn along with some number t ∈ {1, ..., n} called a threshold (see [22, 17] for details). When spending money from this address, one must generate a transaction that has all n public keys (pk1, pk2, …, pkn). It then adds this transaction to the blockchain, followed by t valid signatures obtained from t of the n public keys. The transaction data serves as the message being signed inall t signatures. In reality, multi-signature addresses frequently utilize t= n, requiring signatures from each of the n public keys for each trans- action. A multi-signature strategy can be used in this situation to com- press all n signatures into a single short signature. As a result, there are fewer transactions overall and less information is published to the blockchain. It should be noted that compressing the signatures does not save much space because we still need to write all n public keys to the blockchain. Maxwell et al.[17], building on the work on Bellare and Neven [3], construct a Schnorr-based multi-signature scheme that also supports public key aggregation; the verifier only needs a short aggregate public key instead of an explicit list of all n public keys. This method results in an n-of-n. The data recorded to the blockchain in a spending transaction is this single short aggregated public key, a single short compressed signature, and the message. Multi-signature addresses are simply the hash of the short aggregate public key. This information is adequate to persuade the verifier that the transaction was signed by each of the n signers. By a factor of n, it reduces the volume of data written to the blockchain. This primitive is referred to as a multi- signature technique with public key aggregation by Maxwell et al. Two rounds of communication between the signing parties are required by their signature protocol, and they demonstrate the security of their method by making one additional discrete-log assumption (as assumption introduced in [2]). Recent research [10] has revealed that the security proof is flawed and that security cannot be demonstrated under this premise. The question of whether their approach can be demonstrated secure under other assumptions or in a general group model is still un- resolved.

Applications to Blockchain:

One form of data structure is the block- chain. It has the ability to store data and establish links between objects. The cryptographic, or mathematical, hash algorithm SHA-256 is employed to connect the blocks. Let's consider three numbers of blocks informally (Block-X, Block-Y, Block-Z). Using the same method, the entire Block-hash X's value is saved in the Block-Y, and the entire Block-hash Y's value is stored in the Block-Z. Block-X is connected to Block-Y in this manner, and Block-Y is connected to Block-Z. Block- Y, which stores the hash of Block-X, will also change if Block-X is changed in any way that affects the hash value of Block-X.

Similar to how Block-Y is impacted by changes in Block-Z. When there are many blocks, changing must be done from the first modified block to the last one, and in some hashing circumstances, this operation may be very impossible. On this restriction, the Bitcoin ecosystem is built. The blocks contain information about "transactions," which are acts of transferring money between parties using Bitcoin. Utilizing the immutable data structure characteristic. The Bitcoin ecosystem guaran- tees consumers the preservation of their valuables. But there is still a problem, and that is transaction authentication. An amount of a transaction may be spent or transferred by the real owner before it is stored safely.

Digital signature, another cryptographic technique, is used to verify transactions. Contrary to conventional usage, ownership of a digital item in Bitcoin is not established by the

owner signing it; rather, the prior owner must sign it and store the new owner's public key in it. The example of the paper check can be used to demonstrate this. When using the paper check payment method, the owner of the funds is the per- son whose name is placed in the "Pay to the order of" field, not the signer.
To spend money using the Bitcoin system, the owner must provide a valid signature that was created using a private key that matched the public key that was used in the prior transaction. This strategy offers two advantages: privacy and decentralized authentication. Privacy of Bitcoin ownership is offered as a result of the fact that no one is aware of whose public key is used for a certain transaction. Additionally, since a certificate authority is not required, Bitcoin transactions are de- centralized. The Bitcoin system is the only source of assurance. That transaction signing and validation procedure is reliable and effective.

There are numerous transaction authentication standards in the Bitcoin ecosystem. P2PK (Pay To Public Key), P2PKH (Pay To Public Key Hash), and multi-signature are a few of them.
Multi-signature is used for dual ownership of Bitcoin amounts or for extra security requirements by adding a second authentication element, such as a Bitcoin wallet, in the event that one or more private keys are stolen. A complex framework underlies multi-signature. It is thorough- ly explained in Section 3.5. From this point, "Accountable Subgroup Multi-Signature (ASM)" is a possible improvement for Bitcoin's need for multiple signatures. Instead of having to keep all of the public keys in Multi-signature, ASM allows the signer's public key to be sufficient to store in the transaction. A lot of data and computational savings are provided by this. Additionally, the ASM's responsibility attribute offers another. This characteristic makes all group members accountable for the agreements made by a subgroup. By doing this, the chance that other group members will reject the transactions carried out by the sub- group of users is completely avoided. The BLS (Boneh-Lynn-Shacham) signature scheme [7], which is based on bilinear pairings, is used to build the recommended ASM schema in [5]. Key Aggregation and Group Setup are two additional processes in the ASM schema compared to standard signature systems. The key aggregation stage generates the aggregated public key. The membership keys collected from each member's public and secure key are generated for each group member during group setup. As a result, the accountability is given since the membership keys are required at the Signing stage.

1.1. Outline of the Paper

The remainder of the essay is structured as follows. The associated works are addressed in section 2. The notion of blockchain is explained in Section 3 along with some preliminary topics. The Bitcoin ecosystem, a case study of blockchain, is also briefly examined up to the transactions stage in Section 3, with an eye toward how its transaction authentication methods function. Multi-signatures with key aggregation from pairings are discussed in section 4. Section 5 provides a detailed explanation of the responsible subgroup multi-signature method, which is suggested as an alternative to Bitcoin multi-signature. Finally, section 6 contains the conclusions.

2. Related Work

Even if Multi-signature can reduce the signature value, holding all public keys still occupies a significant portion of transaction size in the Bitcoin ecosystem. Multi-signatures have been worked on different mathematical bases e.g. on RSA [14, 23], discrete logarithm problem [17, 21]. To reduce transaction sizes, the aggregating public-key algorithm has been clearly handled only in [17] and [5] the “public-key aggregation” is named firstly in the work of Maxwell [17]. According to [17], in signing stage, there is a need for two tour of transmission between all signers. But one round of transmission can be enough by [3]. With the name of “Accountable subgroup multi-signatures (ASM)”, this approach was firstly worked by Micali, Ohta and Reyzin [18] take it to the next level via adding key aggregation and using parings further. Multi-signatures have been studied extensively based on RSA [14]. Defending against rogue public-key attacks has always been a primary concern in the context of multi-signature schemes based on discrete-log and pairings [18, 3, 2, 26] and is the main reason for the added complexity in discrete-log-based multisignature systems. Aggregate signatures

[6, 1] are a closely related concept where signatures by different signers on different messages can be compressed together. Sequential aggregate signatures [16, 8] are a variant where signers take turns adding their own signature onto the aggregate. The concept of public key aggregation in addition to signature compression has not been explicitly discussed in the plain public key model until [20] and this work.

3. Preliminaries

In this section, we go over a few tools that are helpful for our plan. We outline the key methods used in our design and illustrate the definitions of security features through an interactive game.

3.1. Blockchain

The blockchain is an ordered, linked list of blocks that functions as a data structure. Each block links back to the one before it in a chain. Be- cause the connected blocks form a chain in this fashion, we refer to the technology as blockchain. The blockchain data can be kept in a data- base or in a file similar to a plain text file. A left-to-right stack of blocks, with the first block serving as the base of the chain, can be compared to the blockchain. In the language of blockchain, "top" (or "tip") refers to the most recent block, and "height" refers to the number of blocks from the first block.

Each block in a blockchain is identified by its hash value, which is produced using the SHA-256 cryptographic hashing technique and placed in the block's header. In addition, each block contains a reference to its parent block and the parent block hash in the block header. A chain that starts with the first block is created by using hash values to link each block to its predecessor. It is conceivable for there to be many blocks with the same parent's hash value. For instance, a block may have more than one kid. There can only be one parent block due to the single parent hash value field.

The hash value of the current block is impacted since the previous block's hash is recorded in the block header. This implies that the hash value of the child also changes if the predecessor's hash value does. Any change to the parent block's value will result in a change to the predecessor's hash. Since the predecessor's hash value has changed, the successor's prior block link must also be updated. Additionally, this causes the successor's hash to change, which calls for a change in the link of the two-next successor, which in turn causes it to change in or- der, and so on. Because of the cascade effect, it is impossible to change a block with numerous descendants without also changing all descend- ants blocks. This needs huge computational power, so being a long chain of blocks gives us the irreversibility property of the blockchain, that is one of the most important properties of blockchains security [24].

3.2. Bitcoin

Blockchain is used in Bitcoin, which is the most popular use in this field. The information regarding the amount of Bitcoin sent from one person to another is contained in blocks, which are used to store Bitcoin transactions. By itself, Bitcoin constitutes a cryptocurrency ecosystem. It offers numerous opportunities to use money in ways we have never thought about. Let's use the analogy between Bitcoin transactions and bank checks as an example. The recipient identification, or beneficiary in banking terms, that is listed on "Pay to the order of," is done using the bitcoin address. The checks don't need to be for a specific account; they might be written in the name of a company, an institution, a corporation, or even in cash.

3.3. Payment To Public Key (P2PK)

Pay-to-public-key is a less complex way to send money using bitcoin than other methods. A script for pay-to-public-key locking is handled as follows:

<Public key of A>OPCHECKSIG OP − CHECKSIG is an ECDSA signature verification procedure to check whether the signature of a user A is valid for a given public key in a transaction locking script. The unlocking script is a simple signature: (Signature by Private Key of A) The combined script, to be validated, is: (Signature by Private Key of A) (Public Key of A) OP-CHECKSIG It is computationally hard to obtain the private key from the public key in an acceptable time period. So the public key is safely used as an address for receiving Bitcoin payments. In P2PK script template, public keys are called P2PK addresses. But nowadays, to ensure better

security P2PK addresses are left in place to P2PKH addresses [24,25].

3.4 Pay to Public Key Hash (P2KH)

The most common type of transaction script in the Bitcoin network is P2PKH script template. It's locking script contains a public key hash instead of a plain public key as a Bitcoin address. Therefore, the public key must be presented to unlock amount of Bitcoin and also a digital signature created by the corresponding private key.

To illustrate P2PKH script template with an example, let Alice pay to Bob's Cafe. Alice made a payment of 0.015 Bitcoin to the Bitcoin address of the cafe. The locking script of transaction output would be like: OP-DUP OP-HASH160 (Cafe Public Key Hash) OP-EQUAL OP-
CHECKSIG Cafe Public Key Hash is the Bitcoin address of the cafe. The unlocking script corresponding to the locking script is: (Cafe Signature) (Cafe Public Key) The validation script is as given below (Cafe Signature) (Cafe Public Key) OP-DUP OP-HASH160 (Cafe Public Key Hash) OP-EQUAL OP-CHECKSIG This combined script's execution results TRUE if the unlocking script matches the conditions set by the locking script. In particular, the result will be TRUE if the unlock- ing script has a valid signature generated by the cafe's private key that corresponds to the public key hash set as a hypothec [24,25].

3.5 Multisignature (M-of-N Multi-Signature)

In a multi-signature script template, at least M of the N public keys that are recorded in the locking script must match the associated signatures in order to release the hypothec. This is often referred to as an M-of-N scheme, where N is the overall key count and M is the minimum number of valid signatures. In order to construct a transaction that is valid to spend a certain amount of bitcoin, a 2-of-3 multisignature requires that three public keys are specified as signers group, at least two of which must be used to create signatures. Standard multi-signature scripts may have various restrictions, such as the ability to use a group of signers with a maximum of 15 participants and only 15 published public keys. The maximum number can be changed in response to advancements in computing power throughout time. A locking script that sets an M-of-N multi-signature condition often takes the following form: M Public Keys 1 and 2 are listed below. N is the total number of public keys, and M is the minimum number of signatures needed to spend the output in the OP-CHECKMULTISIG command. For instance, the locking script in a 2-of-3 multi-signature is as follows: Two (Public Key A and B) 3 OP-CHECKMULTISIG.
The matching script for unlocking that consists of two signatures is OP-0 (Signature B) (Signature C), or another two-signature combination linked to the declared three public keys. Hence, the validation scriptis:
OP-0 (Signature B)(Signature C) 2 (Public Key A) (Public Key B) (Public Key C) 3 OP-CHECKMULTISIG. If two signatures in the unlocking script match the two of three public keys in the locking script, combined validation script will result TRUE [24, 25].

4 Multi-Signatures with Pairing-Based Key Aggregation

We start by introducing our brand-new public-key aggregation compatible pairing-based multi-signature approach. Asymmetric bilinear groups are those in which one of the two groups has a more compact representation. Public keys and signatures must reside in different groups in order to use the pairing-based techniques below. It would make sense to utilize the more compact group for signatures as a single public key is used to sign numerous messages for standard signatures. However, this may no longer be the case since our methods below allow for the aggregation of both public keys and signatures, and the appropriate choice of groups may now heavily rely on the specific application. We outline our plans below:

Putting public keys in G2 and signatures in G1, but we don't specify which group has a more condensed representation. Keep in mind that there are effective hash functions that map into both groups [28, 29, 11].

4.1 Description of our Pairing-Based Scheme

Our pairing-based multi-signature with public-key aggregation MSP is built from the BLS

signature scheme [7].
The system uses the hash functions H_0: {0, 1}* → G_2 and H_1: {0, 1}* → $Z_q$and is secure in the basic public key paradigm.

Generators of Parameters: Pg(κ) sets up as a bilinear group (q, G1,G2, Gt, e, g1, g2) ←G(κ) by Pg(k) and generates keys according to par ← (q, G1, G2, Gt, e, g1, g2) .

Key Generation:

K Ag({pk1, ..., pkn})

$$apk \leftarrow \prod_{i=1}^{n} pk_i^{H_1(pk_i,\{pk_1,\ldots,pk\}_n)}$$

Signing: Signing is a single round protocol. Sign(par, {pk$_1$, ..., pk$_n$, sk$_i$,m}) computes $s_i \leftarrow H_0(m)a_i.sk_i$ where $a_i \leftarrow H_1(pk_i, \{pk_1, ..., pk_n\})$. Send s_i to a designated combiner who computes the final signature as $\sigma \leftarrow \prod_{j=1}^{n} s_i$.This authorized combiner may be an outside party or one of the signers.

Multi-Signature Verification: Vf (par, apk, m, σ) outputs 1 iff $e(\sigma, g_2^{-1}).e(H_0(m), apk) = 1_{Gt}$.

Batch Verification:
We point out that checking a batch of b multi-signatures is quicker than verifying each one individually. Consider a set of triples (m_i, i, apk_i) for i= 1,..., b, where apk_i is the combined public key required to validate the multi-signature □*i*on m_i. This will help you understand how. If each message is unique, we can check each triple as a batch using signature aggregation as described as follows:

- Create an aggregate signature with the formula $\sigma^{\sim} = \sigma_1, \sigma_2, \ldots\ldots\sigma_n \in G_1$.
- Accept as valid all b multi-signature tuples iff
$e(\sigma^{\sim}, g_2) = e(H_0(m_1), apk_1), \ldots, e(H_0(m_b), apk_b)$.

By doing this, validating the b multi-signatures just requires b + 1 pairings as opposed to 2b pairings to verify each one individually. This straight forward batching method can only be applied when each individual message (m_1,..., m_b) exists. If some messages are repeated, batch verification can be performed by first selecting random exponents (ρ_1, ρ_2,... ρ_b) ←{1, ...,2^{κ} }, where κ is the security parameter, computing $\sigma^{\sim} = \sigma_1^{\rho_1}, \sigma_2^{\rho_2}, \ldots\ldots\sigma_b^{\rho_b} \in G_2$ and checking that
$e(\sigma^{\sim}, g_2) = e(H_0(m_1), apk_1^{\rho 1}), \ldots, e(H_0(m_b), apk_b^{\rho_b})$.
The pairings on the right side can, of course, be combined for messages that are sent repeatedly.

4. Accountable Subgroup Multi-Signature:

Micali, Ohta, and Reyzin [18] defined an accountable-subgroup multisignature (ASM) scheme where any subset S of a group of PK, where PK is the set of the public keys of the signers, can create a valid multisignature that can be verified by the public keys of signers in the sub- set. An established access rule can be added to the ASM scheme to more carefully

decide if the subset S is permitted to sign on behalf of PK. For instance, the ASM technique obtains the threshold signature property that authentication can be performed by a minimum number of signers by specifying |S| 't' as a condition.
ASM scheme verification initially requires a description of the set S of signers in the group PK. Verification procedure is based on a compact aggregate public key and signature [17, 5]. In the ASM approach, the aggregate public key can be created from the publicly available signers' public keys; nevertheless, a membership key that is unique to the group is necessary to sign messages on PK's behalf, please. Through a one-time group setup, the group-specific membership key is created.

5.1. Bilinear Groups and Pairing Based Cryptography

Let ζ be a bilinear group generator that takes as an input a security parameter 1^k and outputs the descriptions of multiplicative groups $\Lambda = (q, G_1, G_2, G_t, e, g_1, g_2)$ where G_1, G_2 are groups of prime order q, and G_t be another cyclic group of order q written multiplicatively, a bilinear map $e : G_1 \times G_2 \rightarrow G_t$, and g_1 and g_2 are generators of the groups G_1 and G_2, respectively. For later use, it is denoted that $\Lambda = (q, G_1, G_2, G_t, e)$. If the bilinear map e is a paring, it has the following properties [9]:

Bi-linearity
For all $a, b \in F^*_q$, for all $P \in G_1, Q \in G_2 : e(P^a , Q^b) = e(P, Q)^{ab}$

Non-degeneracy
$e \neq 1$

Computability
There exists an efficient algorithm to compute e. As a usage example of bilinear pairings and to construct ASM [5] lets look BLS [7] signature scheme. In addition to bilinear paring requirements, $H_0 : M \rightarrow G_1$ is a hash function. BLS works as:

- Key generation: Select a $sk \leftarrow Z_q$ randomly and return (pk, sk)where

$pk \leftarrow g_2^{sk} \in G_2$

- Sign(sk, m): Return $\sigma \leftarrow H_0(m)^{sk} \in G_1$.
- Verify(pk, m, σ): If $e(\sigma, g_2) = e(H_0, pk)$ return "true", else"false".

In BLS schema, there also can be a simple signature aggregation. Given (pk_i, m_i, σ_i) for i = 1, ..., n, σ_1, ..., $\sigma_n \in G_1$ can be converted by aggregating them like: $\sigma \leftarrow \sigma_1, ..., \sigma_n \in G_1$. To verify the aggregated signature $\sigma \in G_1$: $e(\sigma, g_2) = e(H_0(m_1), pk_1), ..., e(H_0(m_n), pk_n)$. All (pk_i, m_i) for i = 1, ..., n are needed to verify. As a trick, if all messages are chosen the same ($m_1 = ... = m_n$), verification is downgraded on only two pairings: $e(\sigma, g_2) = e(H_0, pk_1, ..., pk_n)$. Hence, the aggregated public key concept is born $apk := pk_1,...,pk_n \in G_2$.

The Rogue Public-Key Attack.

The signature aggregation in BLS is not so secure, it needs to be improved. See Section 5.3. To illustrate its weakness, let's look at the at- tack scenario: "an attacker registers a rogue public key $pk_2 = g_2^{\alpha} .pk^{-1}$ $\in G_2$, where $pk_1 \in G_2$ is a public key of some unsuspecting user Bob, and $\alpha \leftarrow Z_q$ is chosen by the attacker. The attacker can then claim that both he and Bob signed some message $m \in M$ by presenting the aggregate signature $\sigma = H_0(m)^{\alpha}$. This signature verifies as an aggregate of two signatures, one from pk_1 and one from pk_2, because $e(\sigma, g_2) = e(H_0(m)^{\alpha}, g_2) = e(H_0(m), g_2^{\alpha}) = e(H_0(m), pk_1.pk_2)$.
Hence, this σ satisfies. In effect, the attacker committed Bob to the message m, without Bob ever signing m."[5]

5.2. Construction of ASM Scheme

As first defined in [3], an ASM schema which is based on Schnorr sig- nature [27] consists of the algorithms: parameter generation (Pg), key generation (Kg), group setup (G Setup), Sign, key aggregation (KAg), and verification (Vf). The notation par ← Pg is for common parameters generation, (pk, sk) ← Kg(par) is for key generation, mk ← G Setup(sk, PK) is for generating membership key where PK = pk_1, ..., pk_n is a group of signers. Let each signer in PK be assigned a computable index $i \in \{1, ..., |PK|\}$, for example the index of pk in a sorted.

list of PK. A subgroup S ⊆ {1, ..., |PK|} signs a message m with the interactive algorithm σ ← Sign(par, PK, S, sk, mk, m). Getting the public keys of PK, key aggregation algorithm generates the aggregated public key apk [7]. The algorithm Vf(par, apk, S, m, σ) verifies the signature[5].

With this construction strengthened with BLS schema [7], ASM scheme needs all signers, during group setup, join to multi-signatures on the aggregate public key and the index of every signer, such that the i-th signer in PK has a "membership key" which is a multi-signature on (apk, i). In other words, an accountable-subgroup multi-signature consists of the aggregation of the individual signers.

signatures and their membership keys and the aggregate public key of the subgroup S. To verify a signed message by a subgroup S, it can be checked that the signature is a valid aggregate signature where the aggregate public key of the subgroup signed the message and the membership keys corresponding to S [5].

5.3. Accountable-Subgroup Scheme with PoPs

To integrate PoPs, key structure is like $y \leftarrow g2^{sk}$, $\pi \leftarrow H3(y)^{x}$, $H3 : (0, 1)^* \rightarrow Zq$ as a new hash, and $pk \leftarrow (y, \pi)$, also key aggregation differs in that the aggregate of a set of keys PK = (y1, π1), ...,(yn, πn) in addition to the product $Y \leftarrow \Pi^{n} y$, it has the hash of public keys h ← H (PK). i=1 to 3.

The aggregate public key is a pair apk ← (Y, h).The reason to use hash is, when evaluating H2(apk, i), to consider whether i is the index of the target signer in the set PK for which apk is the aggregate public key. Before aggregating, for completeness, first it may be needed to test that: e(H3(yi, yi)) = e(πi, g2) for i = 1, ..., n. G Setup(ski, PK = pk1, ..., pkn) calculates apk = (Y, h) ← KAg(par, PK) and μj,i= H2(apk, j) ski to signer j. Receiving μj,I from all other signers j = i, signer i calculates μj,i
= H2(apk, j) ski and returns the membership key $mki \leftarrow \Pi^{n}$ i=1μj,i. If any signer is not malicious, it must be equal e(mki, g2) = e(H2(apk, Y)). Signing and verification stages are same as pure ASM scheme, except the product of public keys y ← Πj∈Syj instead of pk ← Πj∈Spkj [7].

The scheme uses hash functions $H0 : \{0, 1\}^* \rightarrow G1$, $H1 : \{0, 1\}^* \rightarrow Zq$ and $H2 : \{0, 1\}^* \rightarrow G1$ [28].

Parameters Generation The bilinear group is set up by Pg(κ) function and also it returns parameters par ← (q, G1, G2, Gt, e, g1, g2) ← ζ(κ)

Key Generation. KAg($\{pk_1, ..., pk_n\}$) gets public key values of all signers and returns $apk \leftarrow \prod_{i=1}^{n} pk_i^{H_1(pk_i, \{pk_1, ..., pk_n\})}$

Group Setup. At first, GSetup(sk_i, PK = $\{pk_1, ..., pk_n\}$) checks whether $pk_i \in$ PK and i is the index of pk_i in PK. Signer i computes the aggregate public key apk ← KAg(PK) as well as $a_i \leftarrow H_1(pk_i, PK)$. It then sends $\mu_{j,i} = H_2(apk, j)^{ai.ski}$ from all other signers j = i, it computes $\mu_{j,i} \leftarrow H_2(apk, j)^{ai.ski}$ and returns the membership key mki ← Πn j=1 $mki \leftarrow \prod_{j=1}^{n} \mu i,j$. Note that if all signers behave honestly, we have that $e(g_1, mk_i) = e(apk, H_2(apk, j))$ In other words, this mk_i is a valid multi-signature on the message $H_2(apk, j)$ by all n parties, as defined in the scheme in Section 5.1.

5.4. ASM integration with Bitcoin

The aggregated public key can be used in place of a Bitcoin address to implement the ASM scheme in Bitcoin transaction scripts. The potential application can be demonstrated using scripting language**.**

In place of Multisig's locking script (M-of-N scheme)
M < Public Key 1 >< Public Key 2 > . . . < Public Key N > N OP CHECKMULTISIG
and the unlocking script where OP CHECKM U LT ISIG means Operator that checks the signature validity
OP0 < Signature n1 > ... < Signature nM >, where OP0 refers to an opera- tor that pushes a blank array on the stack, the possible locking script in ASM scheme should be like < Agg. Public Key > OP CHECKASM where OP CHECKASM means Operator that checks the accountable subgroup multi-signature validity
and the unlocking script in ASM(accountable subgroup multi- signature) OP0 < S >< Common Signature of S > where < S > is a short string of signers' indexes and S is the

subgroup of signers. The aggregation aspect of the ASM system allows for size and computation cost savings, as can be observed from the comparison.

6. Conclusion

This paper presents two alternatives to the transaction authentication mechanisms currently used in blockchains, particularly in the Bitcoin environment**.**

The themes of the current structure of the blockchain concept, Bitcoin, its transaction scripts, and the most common authentication systems are first introduced. It is determined that the transactions must be carried out successfully in addition to the requirement to safeguard the Bitcoin assets, which are so valuable and equivalent to money and must be held in a secure environment**.**

Due to its ineffectual use, Accountable Subgroup Multi-Signature (ASM), a type of multi-signature schema, is suggested as an alternative to Multisig transaction script. After first concept of ASM scheme [18], it is improved by [7] to adding the capability of key aggregation to the schema using the pairing based cryptography. As a result, adding ASM to the Bitcoin ecosystem can increase transaction security by aggregating public keys and reducing trans- action size and computational effort. Future scope of proposed re- search is that it could be applicable any other blockchain structure where multisignature is to be used.

References

[1] J. H. Ahn, M. Green, and S. Hohenberger. Synchronized aggregate signatures: new definitions, constructions and applications. In Proceed- ings of the 17th ACM conference on Computer and communications security, pages 473–484,2010.

[2] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one more-rsa-inversion problems and the security of chaum's blind sig- nature scheme. Journal of Cryptology, 16(3),2003.

[3] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Proceedings of the 13th ACM conference on Computer and communications security, pages 390–399, 2006.

[4] A. Boldyreva. Threshold signatures, multisignature and blind sig- natures based on the gap-Diffie-Hellman-group signature scheme. In International Workshop on Public Key Cryptography, pages 31–46. Springer,2003.

[5] D. Boneh, M. Drijvers, and G. Neven. Compact multi-signatures for smaller blockchains. In International Conference on the Theory and Application of Cryptology and Information Security, pages 435–464. Springer,2018.

[6] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In International conference on the theory and applications of cryptographic techniques, pages 416–432. Springer,2003.

[7] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. Journal of cryptology, 17(4):297–319,2004.

[8] K. Brogle, S. Goldberg, and L. Reyzin. Sequential aggregate signatures with lazy verification from trapdoor permutations. Information and computation, 239:356–376,2014.

[9] J. Camenisch, M. Drijvers, and M. Dubovitskaya. Practical secure delegatable credentials with attributes and their application to blockchain. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 683–699,2017.

[10] M. Drijvers, K. Edalatnejad, B. Ford, and G. Neven. Okamoto beats Schnorr: On the provable security of multi-signatures. IACR Cryptology, e-Print Arch., 2018:417,2018.

[11] M. Fukumitsu and S. Hasegawa. An aggregate signature with pre-communication in the plain public key model. In International Work- shop on Security and Trust Management, pages 3–19. Springer,2021.

[12] M. Fukumitsu and S. Hasegawa. An aggregate signature with pre-communication in the plain public key model. In International Work- shop on Security and Trust Management, pages 3–19. Springer,2021.

[13] C. Gentry and Z. Ramzan. Identity-based

aggregate signatures. In International workshop on public key cryptography, pages 257–273. Springer,2006.

[14] K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignature. NEC Research & Development, (71):1–8, 1983.

[15] A. Juels. Financial Cryptography: 8th International Conference, FC 2004, Key West, FL, USA, February 9-12, 2004. Revised Papers, volume 3110. Springer,2004.

[16] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignature without random oracles. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 465–485. Springer,2006.

[17] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. Simple Schnorr multisignature with applications to bitcoin. Designs, Codes and Cryptography, 87(9):2139–2164,2019.

[18] S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignature. In Proceedings of the 8th ACM Conference on Computer and Communications Security, pages 245–254,2001.

[19] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. De- centralized Business Review, page 21260,2008.

[20] J. Nick, T. Ruffing, and Y. Seurin. Musig2: simple two-round Schnorr multi-signatures. In Annual International Cryptology Conference, pages 189–221. Springer,2021.

[21] K. Ohta and T. Okamoto. Multi-signature schemes secure against active insider attacks. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 82(1):21–31,1999.

[22] B. S. Panda and C. K. Giri. Transformative blockchain knacks for bitcoin cryptocurrency and its impacts. The Role of IoT and Block- chain: Techniques and Applications, pages 237–252,2022.

[23] S. Park, S. Park, K. Kim, and D. Won. Two efficient multisignature schemes. In International Conference on Information and Communications Security, pages 217–222. Springer,1997.

[24] S. Patil and P. Puranik. Blockchain technology. International Journal of Trend in Scientific Research and Development, 3(4):573–574, 2019.

[25] M. Poongodi, A. Sharma, V. Vijayakumar, V. Bhardwaj, A. P. Sharma, R. Iqbal, and R. Kumar. Prediction of the price of ethereum blockchain cryptocurrency in an industrial finance system. Computers & Electrical Engineering, 81:106527,2020.

[26] T. Ristenpart and S. Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In Annual International Conference on the Theory and Applications of Crypto-graphic Techniques, pages 228–245. Springer,2007.

[27] C.-P. Schnorr. Efficient signature generation by smart cards. Jour- nal of cryptology, 4(3):161–174,1991.

[28] M. Scott. A note on group membership tests for \g 1, \g 2 and \g t on bls pairing-friendly curves. Cryptology e-Print Archive,2021.

[29] T. Teruya. A note on subgroup security in discrete logarithm- based cryptography. IEICE Transactions on Fundamentals of Electronics, Communications.