The MuSig Schnorr Signature Scheme

January 9, 2019

This report investigates Schnorr Multi-signautre Schemes (MuSig), which makes use of key aggregation and is provably secure in the *plain public-key model*.

Signature aggregation involves mathematically combining several signatures into a single signature, without having to prove Knowledge of Secret Keys (KOSK). This is known as the *plain public-key model* where the only requirement is that each potential signer has a public key. The KOSK scheme requires that users prove knowledge (or possession) of the secret key during public key registration with a certification authority, and is one way to generically prevent rogue-key attacks.

Multi-signatures are a form of technology used to add multiple participants to cryptocurrency transactions. A traditional multi-signature protocol allows a group of signers to produce a joint multi-signature on a common message.

Contents

1 Introduction

1.1 Schnorr signatures and their attack vectors

Schnorr signatures produce a smaller on-chain size, support faster validation and have better privacy. They natively allow for combining multiple signatures into one through aggregation and they permit more complex spending policies.

Signature aggregation also has its challenges. This includes the rogue-key attack, where a participant steals funds using a specifically constructed key. Although this is easily solved for simple multi-signatures through an enrollment procedure which involves the keys signing themselves, supporting it across multiple inputs of a transaction requires *plain public-key security*, meaning there is no setup.

There is an additional attack, termed the Russel attack, after Russel O'Connor, who has discovered that for multi-party schemes a party could claim ownership of someone else's private key and so spend their other outputs.

Wuille P., [1] has been able to address some of these issues and has provided a solution which refines the Bellare-Neven (BN) scheme. He also discussed the performance improvements that were implemented for the scaler multiplication of the BN scheme and how they enable batch validation on the blockchain.[2]

1.2 MuSig

A traditional multi-signature scheme is a combination of a signing and verification algorithm, where multiple signers (each with their own private/public key) jointly sign a single message, resulting in a combined signature. This can then be verified by anyone knowing the message and the public keys of the signers.

MuSig is a multi-signature scheme that is novel in combining:

1. Support for key aggregation;
2. Security in the *plain public-key model*.

There are two versions of MuSig, that are provably secure, which differ based on the number of communication rounds:

- Three-round MuSig only relies on the Discrete Logarithm (DL) assumption, on which ECDSA (Elliptic Curve Digital Signature Algorithm) also relies
- Two-round MuSig instead relies on the slightly stronger One-More Discrete Logarithm (OMDL) assumption

1.3 Key aggregation

Explain key aggregation properties sought, i.e.

"To make the traditional approach more effective and without needing a trusted setup, a multi-signature scheme must provide sub-linear signature aggregation along with the following properties: - It must be provably secure in the plain public-key model; - It must satisfy the normal Schnorr equation, whereby the resulting signature can be written as a function of a combination of the public keys; - It must allow for Interactive Aggregate Signatures (IAS) where the signers are required to cooperate; - It must allow for Non-interactive Aggregate Signatures (NAS) where the aggregation can be done by anyone; - It must allow each signer to sign the same message; - It must allow each signer to sign their own message.

This is different to a normal multi-signature scheme where one message is signed by all. MuSig provides all of those properties."

The term *key aggregation* refers to multi-signatures that look like a single-key signature, but with respect to an aggregated public key that is a function of only the participants' public keys. Thus, verifiers do not require the knowledge of the original participants' public keys- they can just be given the aggregated key. In some use cases, this leads to better privacy and performance. Thus, MuSig is effectively a key aggregation scheme for Schnorr signatures.

There are other multi-signature schemes that already exist that provide key aggregation for Schnorr signatures, however they come with some limitations, such as needing to verify that participants actually have the private key corresponding to the pubic keys that they claim to have. *Security in the plain public-key model* means that no limitations exist. All that is needed from the participants is their public keys. [1]

2 Applications of multi-signatures

Recently the most obvious use case for multi-signatures is with regards to Bitcoin, where it can function as a more efficient replacement of $n-of-n$ multisig scripts (where the signatures required to spend and the signatures possible are equal in quantity) and other policies that permit a number of possible combinations of keys.

A key aggregation scheme also lets one reduce the number of public keys per input to one, as a user can send coins to the aggregate of all involved keys rather than including them all in the script. This leads to smaller on-chain footprint, faster validation, and better privacy.

Instead of creating restrictions with one signature per input, one signature can be used for the entire transaction. Traditionally key aggregation cannot be used across multiple inputs, as the public keys are committed to by the outputs, and those can be spent independently. MuSig can be used here (with key aggregation done by the verifier).

On a technical standing, in order to combine all the transaction inputs' signatures, a multi-signature scheme is not necessary, instead an aggregate signature scheme can be used. The distinction is simply that in an aggregate signature, each signer can also sign their own message, instead of only one message shared by all.

No non-interactive aggregation scheme are known that only rely on the DL assumption, but interactive schemes are trivial to construct where a multi-signature scheme has every participant sign the concatenation of all messages. Maxwell G., *et al.* [3] focusing on key aggregation for Schnorr Signatures shows that this is not always a desirable construction, and gives an IAS variant of BN with better properties instead. [1]

2.1 Multi-signatures

Introduced by Itakura *et al.* [4], multi-signature protocols allow a group of signers (that individually possess their own private/public key pair) to produce a single signature σon a message m. Verification of the given signature σcan be publicly performed given the message and the set of public keys of all signers.

A simple way to change a standard signature scheme into a multi-signature scheme is to have each signer produce a stand-alone signature for m with its private key and to then concatenate all individual signatures.

The transformation of a standard signature scheme to a multi-signature scheme needs to useful and practical, thus the newly calculated multi-signature scheme must produce signatures where the size is independent of the number of signers and similar to that of the original signature scheme. [3]

2.1.1 Rogue attacks

Rogue attacks are a significant concern when implementing multi-signature schemes. Here a subset of corrupted singers, manipulate the public keys computed as functions of the public keys of honest users, allowing them to easily produce forgeries for the set of public keys (despite them not knowing the associated secret keys).

Initial proposals from [5], [6], [7], [8], [9], [10] and [11] were thus undone before a formal model was put forward along with a provably secure scheme from Micali *et al.* [12]. Unfortunately, despite being provably secure their scheme is costly and an impractical interactive key generation protocol. [3]

A means of generically preventing rogue-key attacks is to make it mandatory for users to prove knowledge (or possession) of the secret key during public key registration with a certification authority [13]. Certification authority is a setting known as the KOSK assumption. The pairing-based multi-signature

schemes by Boldyreva [14] and Lu *et al.* [15] rely on the KOSK assumption in order to maintain security. However, this as can be seen from [16]and [13] this assumption is problematic.

As it stands, the Bellare and Neven [16] provides the most practical multi-signature scheme, based on the Schnorr signature scheme, which is provably secure that does not contain any assumption on the key setup. Since the only requirement of this scheme is that each potential signer has a public key, this setting is referred to as the *plain-key model.*

2.1.2 Recap on the Schnorr signature scheme

The Schnorr signature scheme uses:[17]

- A cyclic group G of prime order p
- A generator $g$of G
- A hash function H
- A private/public key pair is a pair $(x, X)\epsilon\{0, ..., p-1\}\times G$ where $X = g^x$
- To sign a message m, the signer draws a random integer r in Z_p, computes $R = g^r$, $c = H(X, R, m)$, and $s = r + cx$
- The signature is the pair (R, s) , and its validity can be checked by verifying whether $g^s = RX^c$

The above described is referred to as the so-called "key-prefixed" variant of the scheme, which sees the public key hashed together with R and m[18]. This variant was thought to have a better multi-user security bound than the classic variant [19], however in [20] the key-prefixing was seen as unnecessary to enable good multi-user security for Schnorr signatures.

For the development of the new Schnorr-based multi-signature scheme [3], key-prefixing seemed a requirement for the security proof to go through, despite not knowing the form of an attack. The rationale also follows the process in reality, as messages signed in Bitcoin always indirectly commits to the public key.

2.1.3 Design of a Schnorr multi-signature scheme

The naive way to design a Schnorr multi-signature scheme would be as follows:

- A group of n signers want to cosign a message m
  - Let $L = \{X_1 = g^{x_1}, ..., X_n = g^{x_n}\}$ be the multi-set of all public keys [1]

[1]No constraints are imposed on the key setup, the adversary thus can choose corrupted public keys at random, hence the same public key can appear more than once in L

- Each cosigner randomly generates and communicates to others a share $R_{i=}g^{r_i}$
- Each of the cosigners them computes $R = \Pi_{i=1}^{n} R_i$, $c = H(\tilde{X}, R, m)$
- Where $\tilde{X} = \Pi_{i=1}^{n} X_i$ is the product of individual public keys, and a partial signature $s_i = r_i + cx_i$
- Partial signatures are then combined into a single signature (R, s) where $s = \Sigma_{i=1}^{n} si \bmod p$
- The validity of a signature (R, s) on message m for public keys $\{X_1, ...X_n\}$is equivalent to $g^s = R\tilde{X}^c$ where $\tilde{X} = \Pi_{i=1}^{n} X_i$ and $c = H(\tilde{X}, R, m)$
- Note that this is exactly the verification equation for a traditional key-prefixed Schnorr signature with respect to public key $\tilde{X}$, a property termed *key aggregation*

However, as mentioned above, [7], [9], [10] and[12] these protocols are vulnerable to a rogue-key attack where a corrupted signer sets its public key to $X_1 = g^{x_1}(\Pi_{i=2}^{n} X_i)^{-1}$, allowing the signer to produce signatures for public keys $\{X_1, ...X_n\}$ by themself.

2.1.4 Micali-Ohta-Reyzin Multi-signature scheme

The Micali-Ohta-Reyzin multi-signature scheme [12] solves the rogue-key attack using a sophisticated interactive key generation protocol.

2.1.5 The Bagherzandi *et al.* scheme

Bagherzandi *et al.* [21] reduced the number of rounds from three to two using an homomorphic commitment scheme. Unfortunately, this increases the signature size and the computational cost of signing and verification.

2.1.6 The Ma *et al.* scheme

Ma *et al.* [22] proposed a signature scheme that involved the "double hashing" technique, which sees the reduction of the signature size compared to Bagherzandi *et al.* [21] while using only two rounds.

However, neither of these two variants allow for key aggregation.

Multi-signature schemes supporting key aggregation are easier to come by in the KOSK model. In particular, Syta *et al.* proposed the CoSi scheme which can be seen as the naive Schnorr multi-signature scheme described earlier where the co-signers are organized in a tree structure for fast signature generation.

2.1.7 Bellare and Neven signature scheme

Bellare-Neven (BN) [16] proceeded differently in order to avoid any key setup. It is a more widely known *plain public-key multi-signature scheme*, that does not support key aggregation. It is possible to use BN multi-signatures where

the individual keys are MuSig aggregates. BN multi-signature scheme is secure without such assumptions. Below are details:

- Call $L = H(X_{1,}X_{2...})$
- Each signer chooses a random nonce r_i and shares $R_i = r_iG$ with the other signers
- Call R the sum of the R_i points
- Each signer computes $s_i = r_i + H(L, X_{i,}R, m)x_i$
- The final signature is (R, s) where $s$is the sum of the s_i values
- Verification requires $sG = R + H(L, X_{1,}R, m)X_2 + ...$

Technically, BN has a pre-commit round, where the signers initially reveal $H(R_i)$ to each other, prior to revealing the R_i points themselves. This step is a requirement in order to prove security under the DL assumption, but it can be dismisses if instead the OMDL assumption is accepted.

Furthermore, when an IAS is desired (where each signer has their own message), $L = H((X_1, m_{1)}, (X_2, m_2), ...)$ and $s_i = r_i + H(L, R, i)x_i$ is used for signing (and analogous for verification).

The resulting signature does not satisfy the normal Schnorr equation anymore, nor any other equation that can be written as a function of a combination of the public keys; the key aggregation property is lost in order to gain security in the plain public-key model.

Bellare and Neven showed that this yields a multi-signature scheme provably secure in the *plain public-key* model under the Discrete Logarithm assumptions, modeling $H$and H'as random oracles. However, this scheme does not allow key aggregation anymore since the entire list of public keys is required for verification.

$c = H_{sig}(< L >, R, m)$ yields a secure scheme, however does not allow key aggregation since verification is impossible without knowing all the individual singer keys.

2.1.8 The formation of MuSig

This is where MuSig comes in. It recovers the *key aggregation property without losing security:*

- Call $L = H(X_1, X_2...)$
- Call X the sum of all $H(L, X_i)X_i$
- Each signer chooses a random nonce r_i, and shares $R_i = r_iG$ with the other signers
- Call R the sum of the R_i points

- Each signer computes $s_i = r_i + H(X, R, m)H(L, X_i)x_i$
- The final signature is (R, s) where s is the sum of the s_i values
- Verification again satisfies $sG = R + H(X, R, m)X$

So what was needed was to define X not as a simple sum of the individual public keys X_i,but as a sum of multiples of those keys, where the multiplication factor depends on a hash of all participating keys. [1]

Their main idea is to have each cosigner use a distinct "challenge" when computing their partial signature $s_i = r_i + c_i x_i$, defined as $c_i = H(< L > X_i, R, m)$, where as before.

2.2 Interactive Aggregate Signatures

In some situations, it may be useful to allow each participant to sign a different message rather than a single common one. An IAS is one where each signer has its own message m_i to sign, and the joint signature proves that the i-th signer has signed m_i . These schemes are considered to be more general than multi-signature schemes, however they are not as flexible than non-interactive aggregate signatures [23, 24] and sequential aggregate signatures [25].

According to Bellare *et al.* [16], they suggested a generic way to turn any multi-signature scheme into an IAS scheme by the signer running the multi-signature protocol using as message the tuple of all public keys/message pairs involved in the IAS protocol.

For BN's scheme and the scheme of the Blockstream signature scheme, this does not increase the number of communication rounds as messages can be sent together with shares R_i.

However, the problem arise with the generic construction in the *plain public-key model.* Below is a closer look at the transformation and show that is subtly fails to provide the strongest security guarantees in the *plain public key model.*

2.2.1 Applications of IAS

With regards to digital currency schemes, where all participants have the ability to validate transactions, these transactions consist of output (which have a verification key and amount) and inputs (which are references to outputs of earlier transactions). Each input contains a signature of a modified version of the transaction to be validated with its referenced output's key. Some outputs may require multiple signatures to be spent. Transactions spending such an output are referred to as m-of-n multi-signature transactions [26], and the current implementation corresponds to the trivial way of building a multi-signature scheme by concatenating individual signatures. Additionally, a threshold policy can be enforced where only m valid signatures out of the n possible ones are needed to redeem the transaction (again this is the most straightforward way to turn a multi-signature scheme into some kind of basic threshold signature scheme).

While several multi-signature schemes could offer an improvement over the currently available method, two properties increase the possible impact:

- The availability of key aggregation removes the need for verifiers to see all the involved key, improving bandwidth, privacy, and validation cost
- Security under the *plain public-key model* enables multi-signatures across multiple inputs of a transaction, where the choice of signers cannot be committed to in advance. This greatly increases the number of situations in which mulit-signatures are beneficial.

2.2.2 Native multi-signature support

An improvement is to replace the need for implementing $n - of - n$ multi-signatures with a constant-size multi-signature primitive like Bellare-Neven. While this is on itself an improvement in terms of size, it still needs to contain all of the signers' public keys. Key aggregation improves upon this further, as a single-key predicate can be used instead which is both smaller and has lower computational cost for verification. It also improves privacy, as the participant keys and their count remain private to the signers.

When generalizing to the $m-of-n$ scenario, several options exist. One is to forego key aggregation, and still include all potential signer keys in the predicates while still only producing a single signature for the chosen combination of keys. Alternatively, a Merkle tree [27]where the leaves are permitted combinations of public keys (in aggregated form), can be employed. The predicate in this case would take as input an aggregated public key, a signature and a proof. Its validity would depend on the signature being valid with the provided key, and the proof establishing that the key is in fact one of the leaves of the Merkle tree, identified by its root hash. This approach is very generic, as it works for any subset of combinations of keys, and as a result has good privacy as the exact policy is not visible from the proof. It is only feasible however when the total number of combinations is practical.

Some key aggregation schemes that do not protect against rogue-key attacks can be used instead in the above cases, under the assumption that the sender is given a proof of knowledge/possession for the receivers' private keys. However, these schemes are difficult to prove secure except by using very large proofs of knowledge[12]. As those proofs of knowledge/possession for the receivers' private keys. However, these schemes are difficult to prove secure except by using very large proofs of knowledge/possession do not need to be seen by verifiers they are effectively certified by the sender's signature on the transaction which includes the predicate- they do not burden validation. However, passing them around to senders is inconvenient, and easy to get wrong. Using a scheme that is secure in the *plain public-key model* categorically avoids these concerns.

Another alternative is to use an algorithm whose key generation requires a trusted setup, for example in the KOSK model. While many of these schemes have been proven secure [14][15], they rely on mechanisms that are usually not implemented by certification authorities [16] [13].

2.2.3 Cross-input multi-signatures

The previous sections explained how the numbers of signatures per input can generally by reduced to one, but one can go further and replace it with a single signature per transaction. Doing so requires a fundamental change in validation semantics, as the validity of separate inputs is not longer independent. As a result, the outputs can no longer be modeled as predicates. Instead, they are modeled as functions that return a boolean (data type with only two possible values) plus a set of zero or more public keys.

Overall validity requires all returned booleans to be True and a multi-signature of the transaction with L the union of all returned keys.

More concretely, this can be implemented by providing an alternative to the signature checking opcode OP_CHECKSIG and related opcodes in the Script language. Instead of returning the result of an actual ECDSA verification, they always return True, but additionally add the public key with which the verification would have taken place to a transaction-wide multi-set of keys. Finally, after all inputs are verified, a multi-signature present in the transaction is verified against that multi-set. In case the transaction spends inputs from multiple owners, they will need to collaborate to produce the multi-signature, or choose to only use the original opcodes. Adding these new opcodes is possible in backward-compatible way. [3]

2.2.4 Protection against rogue-key Attacks

In Bitocoin, when taking cross-input signatures into account, there is no published commitment to the set of signers, as each transaction input can independently spend an output that requires authorization from distinct participants. This functionality was not restricted as it would then interfere with fungibility improvements such as CoinJoin [28]. Due to the lack of certification, security against rogue-key attacks is of great importance.

If it is assumed that transactions used a single multi-signature that was vulnerable to rogue-attacks, like the simpler schemes previously described, an attacker could identify an arbitrary number of outputs he wants to steal, with the public keys $X_1, ..., X_{n-t}$ and then use the rogue-key $X_{n-t+1}, ..., X_n$ such that he can sign for the aggregated key $\tilde{X}$. He would then send a small amount of his own money to outputs with predicates corresponding to the keys $X_{n-t+1}, ..., X_n$. Finally, he can create a transaction that spends all of the victim coins together with the ones he just created by forging a multi-signature for the whole transaction.

It can be seen that in the case of mulit-signatures across inputs, theft can occur by merely being able to forge a signature over a set of keys that includes at least one key not controlled by the attacker, just what the *plain public-key model* considers a win for the attacker. This is in contrast to the single-input multi-signature case where theft is only possible by forging a signature for the exact (aggregated) keys contained in an existing output. As a result, it is no longer possible to rely on proofs of knowledge/possession that are private to the

signers.

2.3 Bitcoin $m-of-n$ multi-signatures

Muti-signatures refer to requiring more than one key to authorize a transaction. Currently, standard transactions on the Bitcoin network can be referred to as single-signature transactions, as they require only one signature, from the owner of the private key associated with the Bitcoin address. However, the Bitcoin network supports much more complicated transactions which can require the signatures of multiple people before the funds can be transferred. These are often referred to as $m-of-n$ transactions, where m represents the signatures required to spend, while n represents the signatures possible.[29]

2.3.1 Use cases for $m-of-n$ multi-signatures

When $m = 1$ and $n > 1$ it is considered a shared wallet, which could be used for small group funds that do not require much security. It is the least secure multi-sig option because it is not multi-factor. Any compromised individual would jeopardize the entire group. Examples of use cases include funds for a weekend or evening event, or a shared wallet for some kind of game. Besides being convenient to spend from, the only benefit of this setup is that all but one of the backup/password pairs could be lost and all of the funds would be recoverable.

When $m = n$, it is considered a partner wallet, which brings with it some nervousness as no keys can be lost. As the number of signatures required increases, the risk also increases. This type of multi-signature can be considered as a hard multi-factor authentication.

When $m < 5n$, it is considered a buddy account, which could be used for spending from corporate group funds. Consequence for the colluding minority need to greater than possible benefits. It is considered less convenient than a shared wallet, but much more secure.

When $m > 5n$, a consensus account is termed. The classic multi-signature wallet is a 2 of 3 and is a special case of a consensus account. A 2 of 3 scheme has the best characteristics for creating new bitcoin addresses and for secure storing and spending. One compromised signatory does not compromise the funds. A single secret key can be lost and the funds can still be recovered. If done correctly, off-site backups are created during wallet setup. The way to recover funds is known by more than one party. The balance of power with a multi-signature wallet can be shifted by having one party control more keys than the other parties. If one party controls multiple keys, there is a greater risk of those keys not remaining as multiple factors.

When $m = 5n$, it is referred to as a split account, and is an interesting use case, as there would be 3 of 6 where one person holds 3 keys and 3 people hold 1 key. In this way one person could control their own money, but the funds could still be recoverable even if the primary key holder were to disappear with all of his key. As n increases, the level of trust in the secondary parties can

decrease. A good use case might be a family savings account that would just automatically become an inheritance account if the primary account holder were to die.[29]

3 The MuSig signature scheme

The new proposed Schnorr-based multi-signature scheme can be seen as a variant of the BN scheme, allowing key aggregation in the *plain public-key model.* This scheme consists of three rounds, the first two being exactly the same as in BN. Challenges c_i are changed from $c_i = H(< L > X_i, R, m)$ to $c_i = H_{agg}(< L > X_i).H_{sig}(\tilde{X},R,m)$, where $\tilde{X}$ is the so-called aggregated public key corresponding to the multi-set of public keys $L = \{X_1, ...X_n\}$, defined as $\tilde{X} = \prod_{i=1}^{n} X_i^{a_i c} = R\tilde{X}^c$ where $c = H_{sig}(\tilde{X}, R, m)$.

Basically , the key aggregation property has been recovered and can now be enjoyed by the naive scheme, which respect to a more complex aggregation key $\tilde{X} = \prod_{i=1}^{n} X_i^{a_i c}$.

3.1 Revisions

In a previous version of the paper by Maxwell *et al.* [3] published on 15 January 2018 they proposed a 2-round variant of MuSig, where the initial commitment round is omitted claiming a security proof under the One More Discrete Logarithm (OMDL) assumptions ([30], [31]). Drijvers *et al* [32] then discovered a flaw in the security proof and showed that through a meta-reduction the initial multi-signature scheme cannot be proved secure using an algebraic black box reduction under the DL or OMDL assumption.

In more details, it was observed that in the 2-round variant of MuSig, an adversary (controlling public keys $X_2, ..., X_n$) can impose the value of $R = \Pi_{i=1}^{n} R_i$ used in signature protocols since he can choose $R_2, ..., R_n$ after having received R_1 from the honest signer (controlling public key $X_1 = g^{x_1}$). This prevents one to use the initial method of simulating the honest signer in the Random Oracle model without knowing x_1 by randomly drawing s_1 and c, computing

Despite this, there is no attack currently known against the 2-round variant of MuSig and that it might be secure, although this is not provable under standard assumptions from existing techniques. [3]

4 Conclusions, Observations and Recommendations

However, the case of interactive signature aggregation where each signer signs their own message must still be proven by a complete security analysis.

5 Contributors

References

[1] P. Wuille, "Key Aggregation for Schnorr Signatures," 2018. [Online]. Available: https://blockstream.com/2018/01/23/musig-key-aggregation-schnorr-signatures/

[2] Blockstream, "Schnorr Signatures for Bitcoin - BPASE '18," 2018. [Online]. Available: https://blockstream.com/2018/02/15/schnorr-signatures-bpase/

[3] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, "Simple Schnorr Multi-Signatures with Applications to Bitcoin," pp. 1–34, 2018.

[4] K. Itakura, "A public-key cryptosystem suitable for digital multisignatures," *NEC J. Res. Dev.*, vol. 71, 1983.

[5] C.-M. Li, T. Hwang, and N.-Y. Lee, "Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders," in *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1994, pp. 194–204.

[6] L. Harn, "Group-oriented (t, n) threshold digital signature scheme and digital multisignature," *IEE Proceedings-Computers and Digital Techniques*, vol. 141, no. 5, pp. 307–313, 1994.

[7] P. Horster, M. Michels, and H. Petersen, "Meta-Multisignature schemes based on the discrete logarithm problem," in *Information Securityâthe Next Decade*. Springer, 1995, pp. 128–142.

[8] K. Ohta and T. Okamoto, "A digital multisignature scheme based on the Fiat-Shamir scheme," in *International Conference on the Theory and Application of Cryptology*. Springer, 1991, pp. 139–148.

[9] S. K. Langford, "Weaknesses in some threshold cryptosystems," in *Annual International Cryptology Conference*. Springer, 1996, pp. 74–82.

[10] M. Michels and P. Horster, "On the risk of disruption in several multiparty signature schemes," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 1996, pp. 334–345.

[11] K. Ohta and T. Okamoto, "Multi-signature schemes secure against active insider attacks," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 82, no. 1, pp. 21–31, 1999.

[12] S. Micali, K. Ohta, and L. Reyzin, "Accountable-subgroup multisignatures," in *Proceedings of the 8th ACM conference on Computer and Communications Security*. ACM, 2001, pp. 245–254.

[13] 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 Cryptographic Techniques*. Springer, 2007, pp. 228–245.

[14] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme," in *International Workshop on Public Key Cryptography*. Springer, 2003, pp. 31–46.

[15] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters, "Sequential aggregate signatures and multisignatures without random oracles," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2006, pp. 465–485.

[16] M. Bellare and G. Neven, "Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma," *Acm Ccs*, pp. 390–399, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1180453

[17] C.-P. Schnorr, "Efficient signature generation by smart cards," *Journal of cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[18] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of Cryptographic Engineering*, vol. 2, no. 2, pp. 77–89, 2012.

[19] D. J. Bernstein, "Multi-user Schnorr security, revisited." *IACR Cryptology ePrint Archive*, vol. 2015, p. 996, 2015.

[20] E. Kiltz, D. Masny, and J. Pan, "Optimal security proofs for signatures from identification schemes," in *Annual Cryptology Conference*. Springer, 2016, pp. 33–61.

[21] A. Bagherzandi, J. H. Cheon, and S. Jarecki, "Multisignatures Secure under the Discrete Logarithm Assumption and a Generalized Forking Lemma," *Proceedings of the 15th ACM conference on Computer and communications security - CCS '08*, p. 449, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1455770.1455827

[22] C. Ma, J. Weng, Y. Li, and R. Deng, "Efficient discrete logarithm based multi-signature scheme in the plain public key model," *Designs, Codes and Cryptography*, vol. 54, no. 2, pp. 121–133, 2010.

[23] 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*. Springer, 2003, pp. 416–432.

[24] M. Bellare, C. Namprempre, and G. Neven, "Unrestricted aggregate signatures," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2007, pp. 411–422.

[25] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham, "Sequential aggregate signatures from trapdoor permutations," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2004, pp. 74–90.

[26] G. Andersen, "M-of-N Standard Transactions," 2011.

[27] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Conference on the theory and application of cryptographic techniques*. Springer, 1987, pp. 369–378.

[28] G. Maxwell, "CoinJoin: Bitcoin privacy for the real world," 2013. [Online]. Available: https://bitcointalk.org/index.php?topic=279249.0.

[29] B. W. Contributors, "Multisignature," 2017.

[30] M. Bellare and A. Palacio, "GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks," in *Annual International Cryptology Conference*. Springer, 2002, pp. 162–177.

[31] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko, "The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme." *Journal of Cryptology*, vol. 16, no. 3, 2003.

[32] M. Drijvers, K. Edalatnejad, B. Ford, and G. Neven, "Okamoto Beats Schnorr: On the Provable Security of Multi-Signatures," Tech. Rep., 2018.