Multiparty protocol for the Elliptic Curve Digital Signing Algorithm: an engineering approach

Nick Dunstone and Eric Billard

December 3, 2022

Abstract

In this paper we focus on a promising field of research for digital signing protection: the splitting of ECDSA signing keys through multiparty computation (MPC). All existing MPC implementations of EC digital signing base their constructs on threshold signing. We decided to reconsider the entire digital signing process, making threshold a parameter of the equation, rather than a pre-requisite. We propose solutions from an engineering perspective and take a pragmatic approach to devising algorithms adapted to different security requirements.
Through this document, we advance several research propositions so that we may forge a strong working relationship with the academic world and will submit all our models for full academic peer review.
Ultimately, we hope that some of the ideas and discussion points we bring forward will contribute to both theoretical and engineering knowledge improvement.

1 Introduction

1.1 The relevance of private keys and digital signing

Digital private keys (also known at signing keys) are in widespread use today (smartphones, transaction payment networks, web-browsers, telecoms, the "Internet of Things" (IoT), etc.) but the recent irruption of blockchain technologies has epitomised the crucial role they play in the digital sphere: private keys, and the digital signatures they provide, are indeed the cornerstone of all DLTs (Decentralised Ledger Technologies, i.e. cryptocurrencies, utility tokens and, in general, all blockchain related ecosystems). The protection of these keys will represent an increasingly vital challenge as the decentralised internet (Web 3.0) and the Metaverse continue to develop towards mass adoption.
Digital signing is commonly used as a proof of control over a digital asset (NFT, cryptocurrency, certificate, etc.) or as a proof of identity, for authentication purposes (transaction networks, IoT, etc.).
This cryptographic operation forms part of a broader suite of protocols (RSA, Elliptic Curve Digital Signing Algorithm, etc.), shared with and understood by all stakeholders to the transaction (e.g. SWIFT network members authenticate at the beginning of their

session through digital signing based on the RSA protocol).
To reduce private key vulnerabilities, a mathematical scheme recently emerged as particularly relevant: multiparty computation (MPC). Recent academic papers have inspired real-life implementations of multiparty digital signing algorithms, particularly for cryptocurrency custody and trading: multiparty threshold ECDSAs.
A digital signing scheme can be defined as the combination of 3 different algorithms [Gennaro R.(2018)]:

i A key generation algorithm, that creates a public / private key pair, where only *this* private key (and none other) can generate a digital signature that can be validated by the corresponding public key

ii A signing algorithm, using the private key to sign a message m (string), thus creating a signature

iii A verification algorithm that will use the public key to confirm that the signature was indeed generated by its associated private key

To be relevant, a digital signature needs to be unforgeable, i.e. no one can realistically produce a valid digital signature for any given public key, unless they use the associated private key.
Therefore, in the context of blockchain, the loss of an unforgeable private key means irremediable loss of all associated digital assets.
Similarly, stealing a private key will provide the hacker with total control over the related digital assets. Private key protection therefore represents a considerable stake. We will use the rest of this paper to explore ways to reinforce protection by producing a valid signature whilst protecting the private key itself: our analysis will focus on the steps (i) and (ii) described above, since the signing algorithm verification (iii) should remain the same for a given signing algorithm, no matter how the signature is generated.

1.2 Multiparty computation for digital signing: state of the art

Private key protection is currently challenged by a single point of failure problem: no matter how sophisticated a protection mechanism can be (e.g. HSM [1]), it cannot cope with all possible risks of key destruction, loss, corruption or robbery as long as the entire private key is located in one single place.
Because multiparty computation (MPC) allows the splitting and storage of private keys across different signing parties, it is emerging as a promising mathematical framework to overcome the single point of failure. Recent academic papers have inspired real-life

[1] HSM: Hardware Security Module, a hardware unit that stores cryptographic keys to keep them private while ensuring they are available to those authorised to use them. The primary objective of HSM security is to control which individuals have access to an organization's digital security keys. HSMs are physical hardware devices and require complex security procedures, careful technical support and administration to operate safely

implementations of multiparty digital signing Algorithms, particularly for cryptocurrency custody and trading: multiparty threshold ECDSA is defined by Gennaro and Goldfeder [Gennaro R.(2018)] as a scheme that:

> "enables n parties to share the power to issue digital signatures under a single public key. A threshold t is specified such that any subset of $t + 1$ players can jointly sign, but any smaller subset cannot. Generally, the goal is to produce signatures that are compatible with an existing centralised signature scheme. In a threshold scheme the key generation and signature algorithm are replaced by a communication protocol between the parties, but the verification algorithm remains identical to the verification of a signature issued by a centralised party."

Such a framework effectively eliminates the single point of failure by obfuscating private keys behind a network of independent players; each one owning only a proportion of the entire key, a key particle. Furthermore, multiparty computation (MPC) in general allows us to share computation tasks amongst various non-trusted players, thus building decentralised protocols resilient to malicious behaviour, up to a certain extent. This combination of decentralised and trustless attributes qualifies MPC as a powerful building block for private key protection.
As for any cybersecurity innovation, interrogations have been raised on MPC potential limitations and vulnerabilities. The NIST (National Institute of Standards and Technology – USA) has consulted a wide range of cryptography experts [Brandao L.(2022)] and is still evaluating multiparty threshold schemes. As a result of these investigations, a new NIST Certification protocol for MPC digital signing algorithms should be adopted in 2023-24.

1.3 Our contribution: an engineering solution to decentralised digital signing

The purpose of this paper is to present Event Horizon Labs pragmatic implementation of MPC for elliptic curves, in particular ECDSA.
Elliptic curve mathematics in general, and ECDSA (Elliptic Curve Digital Signing Algorithm) in particular, underpin almost all decentralised ledger projects (BitCoin, Ethereum, HyperLedger, Stellar, Cardano, Cosmos, Casper, etc.). Whether Proof of Work, Proof of Stake, Byzantine Fault Tolerant or any other type of consensus protocol, digital signing relies on elliptic curve algorithms for their robustness and speed.

2 Solving multiparty ECDSA

In the following paragraphs we will summarize the academical state-of-the-art in multiparty computation ECDSA, provide an overview of our algorithm, and discuss several key differences between our protocol and the main alternatives.

2.1 Most recent MPC algorithms in 2022

Multiparty computation for digital signing has been explored since 2001. It has evolved from an originally costly threshold signing algorithm (t out of n where $2t+1$ nodes at least need to be active in order to generate a signature), to a much leaner formula (t out of n, $t+1$ nodes only required to be active at the time of signing).
A number of intermediary steps led, in 2018-20, to the 2 contributions we are introducing here: (Gennaro/Goldfeder [Gennaro R.(2018)] and Lindell/Nof [Lindell Y.(2018)]).
These 2 papers have been used by most of the existing real-life implementations of the multiparty computation ECDSA by blockchain projects such as Fireblocks, Unbound (Coinbase) or Qredo.
The main problem for elliptic curve multiparty digital signing algorithms resides in achieving the distributed computation of:

- a reference point R, by raising the curve generator g to the inverse of a secret number k: $R = g^{k^{-1}}$
- a factor s, that requires 2 secret values: k and x: $s = k(m + rx)\ mod\ q$ where:
  - x is chosen uniformly at random in Z_q [2], with a matching public key $y = g^x$
  - m is a hashed string, usually the message or transaction signed itself
  - r is the x-coordinate of the curve point (R)
  - q is the prime order of the group G used for this particular algorithm

The secret values x and k need to be shared amongst non-trusted players to produce a valid signature, which means that no information can leak, that could reveal more than is strictly necessary, else the entire private key could be compromised by the malicious parties. A commonly used model to safely propagate secret information is Shamir's Secret Sharing (see [Shamir(1979)] for more details), combined with a homomorphic encryption scheme (e.g. Paillier [Boneh D.(2017)]).
However, with these components, the distributed secret number generation protocol is inefficient, requiring a high level of computational power and time to be resolved.
Before introducing our solution, we will present 2 protocols that intend to resolve this inefficiency in a very similar fashion, their main difference residing in their respective protection mechanisms against dishonest and malicious behaviours.

[2] Given a cyclic group G, belonging to the ECDSA Elliptic curve, of prime order q, generated by an element g, Z_q includes arbitrary strings generated by a hash function H', defined from G to Z_q

2.1.1 Gennaro / Goldfeder

In their 2018-19 paper, Gennaro and Goldfeder [Gennaro R.(2018)] propose an improvement of Gennaro's original attempt to define an optimal MPC threshold Elliptic Curve Digital Signing Algorithm [Gennaro R.(2016)].
Instead of using Shamir's Secret Sharing for the signature generation, they introduce an additive sharing method for the generation and sharing of k, obfuscating the partial value of k (k_i) with random variables.
However, this algorithm still relies on Shamir's secret sharing and Paillier's encryption for the initial generation and sharing of x. Theoretically, this could be a one-time operation, but it is a real-life frequent occurrence: this step, for instance, is required every time a signature is generated under the BIP32 signature scheme (now a standard for BTC wallets). Even without the deterministic key derivation use case, a key must be generated for every new client of the signing service and also every time an existing client requires a new key to be generated for a particular usage / domain, not to mention key renewal and destruction/reconstruction as part of normal security procedures.
The second characteristic of their approach relates to the detection of malicious behaviour: Zero Knowledge Proof [3] usage is drastically reduced by taking an *optimistic* approach *during* the protocol execution, and only control the validity of the signature resulting from its completion: instead of submitting every single exchange between parties to a Zero Knowledge Proof control, which is resource consuming, their proposal is to suppose that the intermediate steps are valid unless proven wrong when checking the final digital signature.

2.1.2 Lindell / Nof

Lindell and Nof [Lindell Y.(2018)] suggest a similar protocol to Gennaro/Goldfeder's, but for 2 main differences:

- The introduction of an MPC version of ElGamal in-the-exponent [4], faster than Pallier, for x secret sharing
- A systematic detection of malicious behaviour, unlike the Gennaro/Goldfeder *optimistic* approach

These 2 differences are significant, but both protocols still rely on distributed encryption (Paillier or ElGamal) to share secret values across the non-trusted signing parties. We propose to optimise the multiparty ECDSA by replacing encryption with random obfuscation, based on a powerful and secure random number generator.

[3] A zero-knowledge proof (ZKP) is a cryptographic approach that allows one validating party, called "the prover" to verify a claim made by another party without the prover disclosing any supporting information. This will allow to control that partial secrets excahnged between the parties do respect the protocol specifications. For more details, see for instance: Golwasser S., Micali S. and Rackoffs C.: "The Knowledge Complexity of Interactive Proof Systems", Society For Industrial and Applied Mathematics, 1989

[4] for a detailed explanation of the algorithm, see [Lindell Y.(2018)], p3

2.2 Event Horizon protocol overview

Our algorithm introduces a number of improvements to existing research works [Gennaro R.(2018)] and [Lindell Y.(2018)] :

- leveraging on random number obfuscation to completely replace secret encryption and the related costly transactions between signing parties
- pre-computation of partial input, speeding up the ECDSA signing process whilst not revealing any sensitive data
- One-Round-Trip Signing algorithm, for optimal performance
- we also put into perspective the usage of threshold in the signing algorithm, and suggest a parametric approach where threshold can be flexibly adjusted or removed
- finally, should threshold signing be activated in our protocol, we propose a different approach than the usual Shamir's Secret Sharing protocol, also relying on random variable obfuscation

2.2.1 Non-threshold signature vs threshold signature

Threshold signing is currently regarded as the natural solution to a robust real-life implementation of a decentralised ECDSA. One of the most common arguments is, some of the n signing parties might not be available when a time sensitive transaction requires immediate action. Thus, needing only 3 out 5 parties to be available at the time of the signature reduces the likelihood of delay. There is however a cost to establishing a threshold signature protocol: an additional level of complexity is introduced, as the initial x secret is shared through the Shamir's Secret Sharing protocol, an additional MPC sub-protocol within the MPC ECDSA. We believe that it is worth at least considering the trade-off between a threshold signing (t nodes required from a population of n) vs all nodes being required to sign (with backup nodes).

- The advantage of threshold signing is to circumvent the need for hot backup nodes by having a sufficiently large population n, such that the probability of a sub-set of t being available is high enough to ensure timely digital signature.
- The disadvantage is the need of a key sharing mechanism, which impacts the protocol complexity, performance, and might lead to unsecure implementations

The non-threshold solution is redundancy: have multiple copies of each signing node type (e.g. "Alice").

- The advantage of the redundant implementation is a faster and leaner digital signing protocol

- The disadvantage is additional key synchronisation and the risk of all nodes of one "party" to be compromised at the same time (e.g. by an attacker learning the authentication key of the node in question, or learning a secret key particle). To some extent these scenarios can be mitigated (e.g. by using a secondary level of signing nodes for authentication or different key splits).

We intend to be agnostic to the debate and provide a framework where threshold becomes a parameter (if $t = n$, there is no threshold) and the entire signing network configuration adapts to keep the operation secure.
Our proposed algorithm allows the signing network topology to be adapted for different security requirements.
Additionally, should threshold signing be required, we propose a lean secret sharing mechanism, again based on random obfuscation rather than encryption.

2.3 Obfuscation vs Encryption

Shamir's Secret Sharing is a generic procedure which does not necessarily adjust well to the very specific use case of digital signing. For this reason, we decided to undertake proactive research into other methods of key generation and protection. We decided to look into the entire ECDSA signing mechanism from the ground up.
In particular, we have explored splitting calculations between nodes, with the addition of obfuscation variables to hide output (both intermediate and final) from receiving nodes which may then in turn perform additional calculations. These obfuscation variables are shared with other signing parties such that adjustments can be made to offset downstream calculations that are obfuscated. The advantage of this model is that the speed of calculations is very fast when compared with innate EC operations.
We have also split the entire process into static and dynamic sections such that the static part can be performed ahead of time.
The static section also includes the random point generation which has the advantage of being agnostic to the sign-time secret key, used in the dynamic section. It is therefore possible in theory to set up farms within nodes to mass produce pre-prepared signature templates, although our current implementation is based on batches of signature templates created at session level. We believe this approach more pragmatic than a protocol requiring encrypted data manipulation.
Finally, one of the main challenges with ECDSA has been to convert distributed multiplicative shares of a shared secret into additive shares. This is a fundamental requirement for an efficient distributed multiparty solution.

2.3.1 Reduced EC calculation overhead

Our solution focuses on point addition over point multiplication for key generation, which makes this operation exceptionally lean. The generation of k requires point multiplication but this can be performed offline prior to signing time, during the static section of the protocol. The advantage of this approach is that point addition is intrinsically less

intense computationally than point multiplication (which requires successive doubling and addition operations up to the size of the curve order n). Because we are using our own solution to convert multiplicative shares to additive shares, we can take advantage of this to reduce the overall elliptic curve computation overhead.

2.3.2 Separation of concerns – divide and conquer

We decided to divide the signing operation into two distinct areas of research: Threshold on one side, signing algorithm on the other side. These can indeed be considered independent components within an entire digital signing ecosystem. Existing research appears to intertwine these concepts to a high degree, giving rise to a potentially more complicated solution at the expense of processing time and complexity. We have chosen to use the terms "authentication nodes" and "signing nodes" to represent the threshold and signing roles respectively.

2.3.2.1 Threshold

Given a population of n parties where a threshold of t are required to sign, the purpose of this stage is to construct a shared secret (in additive form) from a population of t out of n parties. This is akin to what is done with centralised signing models where t users have to present themselves to an HSM to perform a signing operation. The difference here is that the key particles never come together. Our research has focused on using traditional key sharing algorithms to resolve this. When using threshold signing, each authentication node will hold a share of multiple splits of the secret key such that the requirement of t signing parties can be met.

2.3.2.2 Signing Algorithm

Once a given population of signing nodes has been selected (either by the client or by some deterministic system), the selected signing nodes will either:

- have sufficient information in terms of key particles to be able to perform split signing (non-threshold model), or
- be furnished with a fresh split of the secret key from the quorum of t authentication nodes (threshold model). These key particles will be additive shares of the private key

In order to mitigate against a malicious client choosing an incorrect key set in an attempt to force a node to sign with an invalid split key (for the signing set), the key particles will additionally be split by the nodes using obfuscation masks set up as part of the signing preparation.

2.3.3 One-Round-Trip

One-Round-Trip Digital Signing (1RT) is the resulting product of extensive research by Event Horizon. As the name implies, the intention of this construct is for the client node (instructing the signing operation) to make one network round trip to each of the chosen signing nodes (in parallel) to perform a digital signing operation. This makes the algorithm as computationally efficient as possible for signing in a distributed signing system. Furthermore, no costly EC operations are performed at the point of signing, these are all done offline. The signing operation is split into 2 phases:

2.3.3.1 Preparation

During this phase the various obfuscation numbers and intermediate (masked) calculation results are shared between the signing parties. Also in this phase, the EC random point R is created (in stages). This is performed in a similar manner to an EC Diffie-Helman key exchange. Each party will derive the value of R independently and return this to the signing client. The client will be able to verify that all parties have generated the same point. Any prepared signature that does not meet this acceptance criteria will be rejected. It should also be noted that a prepared signature is not tied to any specific signing key and can be used to sign with any distributed shared secret (signing private key), known by the signing nodes at sign-time.

2.3.3.2 Sign-Time

The sign-time part of the calculation entails the splitting of the hash value to be signed into additive shares (by the client). The client then instructs each of the selected signing nodes that collectively hold sufficient key particles to be able to sign. These parties further split the hash part proportions using pre-calculated adjustments, shared during the preparation phase in a manner unknown to the client. The signing nodes then complete the pre-prepared signature calculations for the 'dynamic' part of the calculation (Secret Key particle, hash share and multiplicative split of the k point inverse) and return this and other prepared values to the client. The client then performs some straightforward addition and multiplication to determine the value of s, the second component of the ECDSA signature, (note that r, the first part of the signature has already been determined during the preparation phase). This resulting signature is indistinguishable from an ECDSA signature created by the traditional single point signing algorithm.

2.3.4 Empirical Analysis

We have created a working Proof of Concept (POC) for the full ECDSA signing solution with:

- 1 client node
- 3 signing nodes

The POC is able to generate signing keys in a fully distributed manner (non-threshold at present), prepare signature templates, perform split signing, join partial signatures and verify the resulting signature using the standard ECDSA verification algorithm. This implementation is for the "maximum of one corrupt out of 3 signing-nodes" model. Ongoing research has produced a model for a "maximum of 2 corrupt signing nodes" solution where 2 out of the 3 signing nodes may be compromised.
This POC does not use full scale EC but demonstrates empirically (using 16-bit finite fields) that the entire solution will sign and verify correctly according to the ECDSA algorithm specification.
It does however use real EC over 16-bit finite fields. It is intended that this tool will be used by academic researchers to analyse the security properties of the algorithm.
We are working on a full size EC signing key prototype to demonstrate our multiparty ECDSA performance.

2.3.5 Statistical Sampling Verification Model

As pointed out when comparing [Gennaro R.(2018)] to [Lindell Y.(2018)] one key element of a secure multiparty digital signing algorithm is the way malicious behaviour is detected and mitigated. Whilst Lindell and Nof enforce systematic Zero Knowledge Proof control on every partial secret exchange, Genarro and Goldfeder suggest a game theoretic approach, whereby a signature verification is performed at the end of the whole process.
We could perform a signature verification by the client node for all signatures that are returned from the signing nodes. Performing such validation will add significant processing requirements and slow down signature generation.
We do not believe that this is a binary condition: The need to perform full signature verification vs not performing any signature verification does not seem to be a pragmatic real-world scenario. Our proposal is to perform statistical sampling of signatures to verify correctness. Note that this statistical validation will come in addition to the client checking that all nodes are generating the same R point value at signature preparation stage. The level of sampling is an area of active research but should be sufficient to catch malicious nodes in a timely manner.

3 Research Fields

As much as engineering design and empirical evidence have so far reinforced our confidence in the robustness of our protocol, academical investigation is required to fully validate it. This will also pave the way to any future improvement. The following paragraphs introduce various problems that require academic investigation. They also describe how we will support research teams interested in cooperating with Event Horizon Labs. Amongst many promising potential subjects, we have focused on those that directly impact the foundations of our multiparty ECDSA. We will, however, produce separate papers to cover:

- Multiparty RSA decentralised private key creation, as existing theoretical solutions are far too inefficient to be realistically implemented in a real-life context
- Post quantum multiparty distributed digital signing, as quantum computing progress threatens existing encryption protocols such as elliptic curve-related algorithms and RSA

3.1 Random obfuscation: perfect secrecy evaluation

3.1.1 Summary

Our ECDSA multiparty algorithm relies on random obfuscation to mask secret variables exchanged between non-trusted parties. Random obfuscation is secure only if all perfect secrecy requirements are fulfilled.
As formulated by [Shannon(1948)] in 1949, ciphertext maintains perfect secrecy if:

> "the attacker's knowledge of the contents of the message is the same both before and after the adversary inspects the ciphertext, attacking it with unlimited resources. That is, the message gives the adversary precisely no information about the message contents"

As demonstrated in the same paper, to provide perfect secrecy, an obfuscation protocol must respect a number of key properties:

- the key material must be as long as the ciphertext, when both are represented in bits. In other words: to obfuscate a 128 Kb long string with a random number, the random generator must produce an equally long (128 Kb) random number.
- plaintext and ciphertext, when considered as random variables, must be independent

Since obfuscation variables will be generated by our own pseudo-random generator, we could assume that our random obfuscation secrecy will be as robust as the obfuscation variable entropy. Should the random generator rely on any identifiable pattern, the obfuscation could be lifted and partial secrets leaked to adversaries.
We believe that it is possible to demonstrate that, from a theoretical perspective, the

random obfuscation mask secrecy can be proven to be as secure as the security of the underlying pseudo-random number generator. This is based on the assumption that additive masks (mod curve order) would possess perfect secrecy if the mask itself were to be perfectly random. This would make any attempt at a brute force attack no better than an attack attempting to guess the output of a strong random number generator.
We would like to cooperate with academic researchers to verify this with formal proofs.

3.1.2 Proposed Research protocol

We propose the following level of support to an academic research team willing to resolve this problem:

- An initial workshop and introduction to our ECDSA solution
- Access to the random obfuscation mask specifications as well as a step-by-step walk through with small numbers
- Access to our automated 3-node ECDSA multiparty prototype

3.2 Perfect secrecy obfuscation secret sharing vs Shamir's Secret Sharing

3.2.1 Summary

Assuming Perfect Secrecy for a random obfuscation secret share protocol, information can still be leaked due to some of the signing parties abnormal behaviour (faulty, dishonest or malicious), resulting in sensitive information leakage. In order to safeguard the secret, the protocol that drives the behaviour of honest parties must hold under a number of assumptions.
These assumptions (e.g., at least k out of n parties must be honest to produce a valid signature) determine the level of robustness of the protocol. We therefore propose a rigorous analysis of our algorithm (the procedure to exchange obfuscated information between the parties, the timing of information exchange, etc.) to determine necessary restrictions, limits and conditions under which it is multiparty secure, thus resilient to faulty and hostile behaviours. As a second step, we would propose a comparison of our obfuscation secret share protocol with Shamir's Secret Sharing. Shamir's Secret Sharing is indeed the current reference for MPC threshold signing (both [Gennaro R.(2018)] and [Lindell Y.(2018)] rely on this protocol). As indicated before, it has a significant impact on the digital signing algorithm performance. The comparison should be based on a 2-party secret share scenario, in order to simplify the analysis. Both Shamir's Secret Sharing and the random obfuscation approaches should be analysed, their respective differences exposed and commented.

3.2.2 Proposed Research protocol

We will provide an academic research team willing to resolve this problem with the following support:

- An initial workshop and introduction to our ECDSA solution
- Access to the random obfuscation specifications as well as a step-by-step walk through with small numbers
- Access to our automated 3-node ECDSA multiparty prototype for empirical validation purposes

3.3 Variable pre-calculation: leakage risk analysis

3.3.1 Summary

As indicated in this document (2.2.5) our multiparty ECDSA relies on a preparation phase: obfuscation variables, masked calculation results as well as the elliptic curve random points will be shared between the signing parties.
As is implied with partial secret exchanges between parties, this phase could expose to leakage, and we therefore carefully reviewed every step of the One-Round-Trip protocol, especially the preparation phase.
The problem we would like to submit for further academic investigation is a review of potential attacks and vulnerabilities that could cause information leakages during the preparation phase.
The conclusions should suggest controls to detect abnormal behaviour, and possible improvements or mitigations to the protocol robustness. As for all previous problems, the security/performance trade-off should be commented and clearly exposed.

3.3.2 Proposed Research protocol

We propose the following support to an academic research team willing to resolve this problem:

- An initial workshop and introduction to our One-Round-Trip ECDSA
- Access to the preparation phase detailed specification and prototype
- Access to our automated signing party actions explicit log (describing step by step all actions and calculations performed by each party)
- Access to our automated 3-node ECDSA prototype for empirical validation purposes

3.4 Optimal Signature verification through sampling

3.4.1 Summary

As detailed in this document (2.2.7) our protocol will submit signatures to statistical sampling validation. This signature validity check will allow the detection of faulty or malicious behaviours from any of the parties engaged in the multiparty digital signing. This variant of the "optimistic" approach proposed by [Gennaro R.(2018)] will not impose systematic testing of the signatures, but rather an evolutionary statistical sampling: starting with a threshold k of signatures checked, it will increase verification frequency as soon as invalid signatures are detected.
If invalid signatures are detected (without legitimate error conditions being reported to the client by signing nodes), then the level of sampling should be stepped up and further failures reported against the selected signing nodes such that over time it will be possible to precisely identify any rogue nodes and isolate them from the network.
We would like to submit for academic review the entire approach of sampling vs. systematic signature verification or Zero Knowledge Proof verification of every single partial secret share. The analysis should consider both the level of security on one hand, and the cost in resources (computation/bandwidth, etc.) on the other hand.

3.4.2 Proposed Research protocol

We propose the following protocol to an academic research team willing to resolve this problem:

- An initial workshop and introduction to our ECDSA, with particular focus on final partial signatures merge
- Detailed description of the conceptual model underpinning our process: type and roles of the nodes, exchange protocol and processes involved in signature verification.

References

[Boneh D.(2017)] Goldfeder S. Boneh D., Gennaro R. *Using level-1 homomorphic encryption to improve threshold dsa signatures for bitcoin wallet security.* Latincrypt, 2017.

[Brandao L.(2022)] Davidson M. Brandao L. *Notes on Threshold EdDSA/Schnorr Signatures.* NIST, 2022.

[Gennaro R.(2018)] Goldfeder S. Gennaro R. *Fast Multiparty Threshold ECDSA with Fast Trustless Setup.* ACM, 2018.

[Gennaro R.(2016)] Narayanan A. Gennaro R., Goldfeder S. *Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security.* Springler, 2016.

[Lindell Y.(2018)] Nof A. Lindell Y. *Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody.* ACM, 2018.

[Shamir(1979)] Shamir. *How to share a secret.* ACM, 1979.

[Shannon(1948)] Shannon. *A mathematical theory of communication.* Bell System Technical Journal, 27:379–423 and 623–656, Reprinted in 1974 by D. Slepian, Key Papers in the Development of Information Theory, IEEE Press, 1948.