A Survey on Threshold Signature Schemes

Sinan Ergezer, Holger Kinkelin*, Filip Rezabek*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: sinan.ergezer@tum.de, kinkelin@net.in.tum.de, frezabek@net.in.tum.de

***Abstract*—Threshold signatures, an alternative of current signature systems, provide better security than its formers. Nevertheless, constructing a threshold signature scheme from a non-threshold one is a complex process that requires two main challenges to be solved. The first challenge, namely the key generation part can be centralized or decentralized. The solution to the second challenge, distributed signing, is affected by the signature scheme being used. In this paper, these two main challenges will be analyzed by giving insights on the signature schemes ECDSA, BLS, and Schnorr.**

***Index Terms*—threshold crypto schemes, key generation, distributed signing**

1. Introduction

With the current development in the cryptocurrency industry, the necessity of secure digital signatures has also grown proportionally. As of 2018, the amount of cryptocurrency daily stolen is equivalent to 1.7 billion dollars. [1] That means, there are some huge problems regarding the authentication of the parties in a digital transaction. There are numerous ways for authentication in a digital currency transaction. [2]

The most widely used way for authentication is, single signature, which only has one approver and one private key taking part in the transaction; that also makes it the least secure option, since single point of failure/attack is present. That means, if the signing party goes unavailable, the signing cannot be succeeded. Similarly, an adversary can directly forge the signature if he/she can compromise the signing party. As a solution, multisignature was proposed. In multisignature, there are multiple approvers, having a signature each. This is inefficient, since multiple are needed for every transaction, but also insecure since information of the other parties that involved are revealed by the time of every signing. This led to an inevitable upgrade of multisignature: the threshold signature system, which eliminates single point of failure and attack. In a threshold cryptosystem, there are also multiple parties involved in the signature, but one single key is shared among them. This makes it hard to forge the signature since an adversary must compromise the threshold number of parties. In a threshold signature system, not all of the parties need to be present during the signing phase, if a pre-determined number t of n parties are available, signing can be successfully done.

The rest of the paper is structured as follows:

1) Background section 2
2) Threshold Key Generation section 3
3) Comparison and Use Cases of Threshold Signature Schemes section 4
4) Conclusion section 5

2. Background

Before diving into the threshold signature schemes, we will introduce the general concept of a digital signature scheme to the reader.

2.1. Structure of a Digital Signature

A digital signature scheme can be divided into three main algorithms: key generation, signing and verification [3]. In key generation, a pair of keys are generated: a signing key(private key) which we sign the message with and a public key to verify this message. Let us express the the output of key-gen algorithm as a tuple (sk, pk). The signing function sig takes the secret key sk and a message m as an input. It outputs the signed message m' as follows: $sig(sk, m) \rightarrow m'$. And lastly the verification function $ver(pk, m, m')$ verifies whether the signature is valid. That means, decryption of m' with the help of pk should be equal to message m.

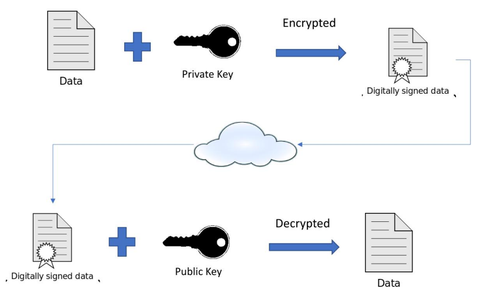

Figure 1: Digital Signature [4]

Before getting started with the digital signature schemes, the concept of elliptic curves should be introduced to the reader since it plays a significant role in the signature schemes ECDSA and BLS.

2.2. Elliptic Curves

An elliptic curve is an algebraic structure that is used for cryptographic purposes. It is a plane curve over a finite field, which has the mathematical term $y^2 = x^3 + ax + b$, and as a result of its term, it is symmetrical about the x-axis [5]. The elementary operation over an elliptical curve is the point addition. To add two points P and Q that are on the curve following steps are done [6] :

1) Determine the curve between the two points
2) Make the curve intersect with the ellliptic curve to find the third point R'
3) Reflect the the third point to find the resulting point R

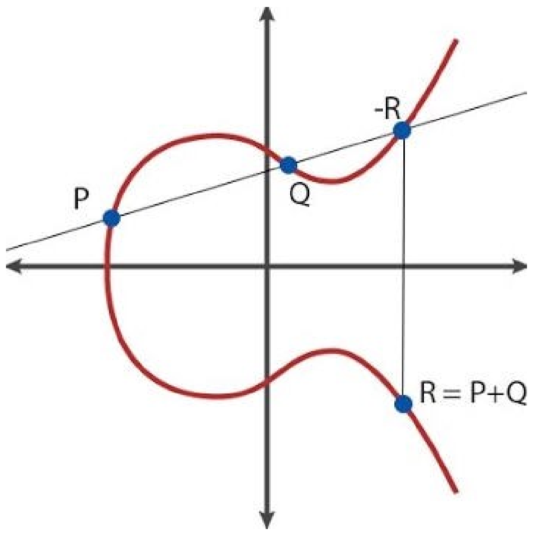

Figure 2: Point Addition in an Elliptic Curve [7]

2.3. Discrete Logarithm Problem

The mathematical property of elliptic curves enables us to get to every other point in the curve if we add any chosen point on the curve to itself enough times. We call this arbitrarily chosen starting point the base point or the generator G of the elliptic curve. That way, every point P can be written as a multiple of G: $P = nG$ where n is a positive integer. As a result, multiplying the generator n times should have had $\mathcal{O}(n)$ complexity, since we are doing n point additions. But this is not the case, since with the help of the property $nP + rP = (n + r)P$ we only have $\mathcal{O}(\log n)$ complexity. We have an efficient method for calculation nG as it has $\mathcal{O}(\log n)$ complexity, but no feasible method is known for deducing n from P and G. This problem is known as the "Discrete Logarithm Problem" in cryptography [8].

2.4. Elliptic Curves and Public Key Cryptography

In a cryptographic scheme that uses an elliptic curve, the public key is computed using xG where x is the private key and G the generator. We know due to the Discrete Logarithm Problem that there is no efficient method for finding the private key from the public key. If the private key were a 256-bit integer, it would take on average 2^{128} calculations for finding it. This leads us to one of the core principles of public key cryptography: It should be computationally infeasible to figure out the private key given the public key [9].

2.5. Digital Signature Schemes

The key generation and signing functions of the signature Schemes ECDSA, BLS and Schnorr are presented in the following, where the verification functions are omitted, due to brevity. ECDSA [10]:

- Key-Gen : Random integer x on the elliptic curve, which is the private key. Public key := $y = g^x$
- Signing:
  - Hash the message : $h = H(m)$
  - Generate a random nonce k
  - let $R = kG$ and take the x-coordinate of it
  - let $s = k^{-1}(h + r * x) modn$ where $k^{-1}(modn)$ is modular inverse function Our signature σ is the pair (r, s)

BLS: [11]:

- Key-Gen : Random integer x on the elliptic curve, which is our private key. Public key : $y = g^x$
- Signing : The bitstring of the message m is hashed using a hash function : $h = H(m)$ The signature is: $\sigma = h^x$

Schnorr: [12]:

- Key-Gen: Random x from Schnorr group, which is our private key. Our public key is:$y = g^x$
- Signing:
  - let $r = g^k$ where k is a random integer from the Schnorr Group
  - let $e = H(r||m)$: the bitstring r and the message m is concatenated and then hashed
  - let $s = k - xe$. Our signature σ is the pair (s, e)

3. Threshold Signatures

The most complex challenge of a threshold scheme is the key generation part. Before explaining the centralized and the decentralized approaches, the concept of secret sharing will be introduced. Secret sharing helps a party to distribute it's secret among other parties. In the centralized approach, the secret is the private key that is distributed among other parties by a chosen party. In the decentralized approach, there is no such authority, the generated key share of each party is distributed among every other party.

3.1. Shamir's Secret Sharing

Shamir's Secret Sharing is based on the idea that t points are sufficient to uniquely define a polynomial of degree $t - 1$. F is a finite field of size q, where q is a prime number [13].

Start by choosing $t-1$ positive integers as coefficients to our polynomial f : $a_1, .., a_{t-1}$ with every a_i being an element of our finite field F. Let a_0 be the private key.

Construct for $1 <= i <= n$, $s_i = (i, f(i) mod q)$: Every party i is given its secret key share sk_i. With any t parties of n, it would be possible to reconstruct our polynomial through interpolation [14]. Thus, it will be possible to acquire the secret S. This is also the threshold property: with our threshold number of parties being present, the private key will be reconstructed.

3.2. Verifiable Secret Sharing(VSS)

In Shamir's Secret Sharing there is no verification ensured for the secret that is shared. Or in other words, one cannot ensure that the secret share is valid. In VSS, the party which distributes the key shares also gives a commitment along with the key shares [15]. Every party can verify their share with the help of this commitment.

3.3. Centralized Key Generation Approach

In the centralized approach, a single party known as "dealer" generates public and private key pairs, whereas the private key is our private key from the previous secret sharing algorithms [16].The most naive secret sharing method to use in the centralized approach is Shamir's Secret Sharing, which is acceptable if the dealer is a "trusted authority", meaning that it is not assumed to be malicious. If the dealer is untrusted, a secret sharing method with additional verification functionality is needed, which makes VSS a more secure choice.

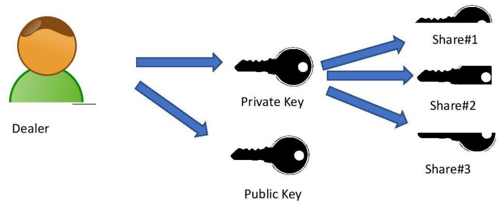

Figure 3: Centralized Key Generation

3.4. Decentralized Key Generation Approach

In the decentralized key generation, there is not any authority who generates the private key. In this approach, each player of the threshold signature computes their own share. This is done using a distributed key generation algorithm known as DKG [17]. A DKG algorithm consists of two phases: the sharing phase and the reconstruction phase.

1) Sharing phase: Every party does a secret sharing of their randomly chosen secret x using VSS to the other parties. In the end, each party has a share S_i of the secret S, (the private key), which is a linear combination of the shared values.
2) Reconstruction phase: Each party declares its secret share, which is reconstructed with the help of a reconstruction function Rec and a consequent broadcast of the public key is followed.

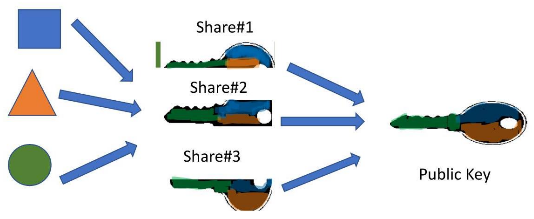

Figure 4: Distributed Key Generation

4. Comparison and Usage of Threshold Signature Schemes

4.1. Trade-Off Between the Two Approaches

In the centralized version, as aforementioned, the dealer can be trusted or untrusted. Verifying the key shares through VSS seems to solve the security problem, but it does not provide a full security. In the centralized approach, there is always a "single point of attack/failure", which is in the side of the authority. If the authority goes unavailable, the key generation can not be made. Or an access to the authority's side by an adversary could mean a reveal of the private key. Those both scenarios are meant to be tackled by the decentralized approach, which has no single point of failure. It could be said that the centralized approach is not complex but insecure because participating parties rely on one single party. On the other hand, the implementation of DKG is more complex, lasting multiple computational steps. Also, communication overhead is present between the parties, which could cause a problem for devices with limited computational power. It could nevertheless be said that this computational burden could still be tolerated since key generation is not a process that is repeated frequently. Generally, if no security concerns are encountered, the same key can be used for signing different messages.

4.2. Solution to the Distributed Signing Problem

In the signing phase of a threshold signature scheme, every party should sign the message m with their own share of the secret key obtained from the KeyGen phase [2].The most important part is that each party should never reveal their secret share and nor should the secret key be reconstructed at any point, which would cause a direct single point of attack.

4.3. Difficulty of Transforming Signature Schemes into Their Threshold Versions

Implementing a distributed signing protocol proved to have technical difficulties [18]. Threshold ECDSA signatures need additional cryptographic primitives such as Paillier encryption in the signing phase. This complexity is present in the signature schemes coming from the DSA family [19]. There is no standardized solution for the threshold signing protocol of ECDSA. Also, there is not

property	ECDSA	BLS	Schnorr
sig. aggregation	no	yes	yes
determinism	no	yes	yes

TABLE 1: Table for Property Comparison of Each Threshold Scheme

any known version of threshold ECDSA that does not use multiple rounds, in other words, the parties should exchange secret information with each other multiple times.

BLS and Schnorr can easily be transformed into threshold versions since they are suitable for signature aggregation in contrary to ECDSA. That means, with their current schemes, every party signs the message using their own key share sk_i. The resulting signature can then be additively combined into one signature.

4.4. Property Comparison of Each Threshold Scheme

A table that compares the properties of the threshold schemes is presented above.Signature aggregation is already presented in subsection 4.3. Another property is determinism, which means: "For any given public key and message there can be only one valid signature" [20].

4.5. Use Cases of Threshold Signatures

Threshold Signatures are still a prototyped technology which does not have any standardized usage, although there is still ongoing research and development regarding the adoption of threshold signatures in various fields.

4.5.1. Bitcoin Transactions. A standard Bitcoin transaction uses single signature ECDSA with being behind today's signature technology. The adoption of multisignature wallets is also significantly low, as of 2016, multisignature wallets are being used in only 11 percent of the transactions [21]. Although the cryptocurrency industry is unable to catch the current signature technology, there is a continous research on the compatibility with Bitcoin: in well-known papers of Gennaro et al [18] and Goldfeder et al. [22], threshold ECDSA signature schemes are proposed for usage in Bitcoin. Despite the difficulty in the adaptation of ECDSA into a threshold scheme as stated in subsection 4.3, the reason why ECDSA is selected is convenience. ECDSA is the standard signature scheme for Bitcoin.

4.5.2. Certificate Authorities. Certificate Authorities are trusted third parties, who validate the identity of internet entities by digitally signing certificates [23]. Since two parties rely on these digital certificates when communicating with each other, an adversary that can forge the signature of that authority could directly influence the communication. Using threshold signature mechanism would be a more secure choice, as presented in the paper of Zhou et. al [24].

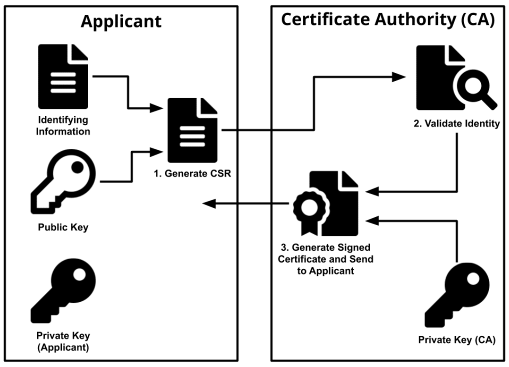

Figure 5: Certificate Signing Mechanism [23]

5. Conclusion

In threshold signatures, there are two main challenges to tackle. The first one, also the most complex one, is the key generation part. The key generation has two main approaches. In the centralized approach, a trusted authority exists, who generates the private key and distributes it among the parties who will sign the message. In this approach, the parties do not have any computational burden but it is not secure enough due to having a single point of failure. In the decentralized approach the single point of failure is eliminated, but this time each party directly takes part in the key generation. The complexity of distributed signing problem varies because of the nature of the scheme being used. It is a known fact that threshold ECDSA scheme is difficult to implement, whereas threshold BLS and threshold Schnorr do not bring any implementation difficulties by dint of signature aggregation property. Even so, the standard threshold signature schemes to be used and the approaches going to be taken to solve the two mentioned challenges are still unknown. Threshold signature systems remain to be a prototype.

References

[1] G. Chavez-Dreyfuss, “Cryptocurrency thefts, scams hit $1.7 billion in 2018: report ,” *Reuters*, 2019, [Online; accessed 16-August-2020].

[2] S. Alex. (2020) Threshold Signature Explained— Bringing Exciting Applications with TSS . https://medium.com/@arpa/threshold-signature-explained-brining-exciting-apps-with-tss-8a75b43e19bf. [Online; accessed 16-August-2020].

[3] R. Gennaro and S. Goldfeder, “Fast multiparty threshold ecdsa with fast trustless setup,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1179–1194. [Online]. Available: https://doi.org/10.1145/3243734.3243859

[4] https://thecybersecurityman.com/2017/12/12/understanding-the-cia/, [Online; accessed 16-August-2020].

[5] Y. El Housni, “Introduction to the Mathematical Foundations of Elliptic Curve Cryptography,” 2018.

[6] B. Bill, “Adding Points in Elliptic Curve Cryptography ,” *Medium*, 2019, [Online; accessed 16-August-2020].

[7] https://hal.archives-ouvertes.fr/hal-01914807/document, [Online; accessed 16-August-2020].

[8] D. J. Pantůček, “Elliptic curves: discrete logarithm problem ,” *Trustica*, 2018, [Online; accessed 16-August-2020].

[9] M. D. Ryan, "Computer Security lecture notes," https://www.cs.bham.ac.uk/~mdr/teaching/modules/security/lectures/public_key.html, 2008, [Online; accessed 16-August-2020].

[10] S. Nakov, "ECDSA: Elliptic Curve Signatures ," 2019, [Online; accessed 16-August-2020].

[11] S. Varadaraj, "The BLS Signature Scheme — A short intro," 2018, [Online; accessed 16-August-2020].

[12] "Schnorr Signature," [Online; accessed 16-August-2020].

[13] E. Rafaloff, "Shamir's Secret Sharing Scheme ," 2018, [Online; accessed 16-August-2020].

[14] B. Archer and E. W. Weisstein, "Lagrange Interpolation ," [Online; accessed 16-August-2020].

[15] B. Schoenmakers, *Verifiable Secret Sharing*. Boston, MA: Springer US, 2005, pp. 645–647. [Online]. Available: https://doi.org/10.1007/0-387-23483-7_452

[16] C. Li and J. Pieprzyk, *Conference Key Agreement from Secret Sharing*. Springer,Berlin,Heidelberg, 2001, vol. 1587.

[17] A. Kate, Y. Huang, and I. Goldberg, "Distributed key generation in the wild." *IACR Cryptology ePrint Archive*, vol. 2012, p. 377, 2012.

[18] R. Gennaro, S. Goldfeder, and A. Narayanan, *Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security*. Springer, 2016, pp. 156–174.

[19] P. MacKenzie and M. K. Reiter, "Two-party generation of dsa signatures," in *Advances in Cryptology — CRYPTO 2001*, J. Kilian, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 137–154.

[20] A. Block, "Secret Sharing and Threshold Signatures with BLS," 2018, [Online; accessed 16-August-2020].

[21] O. Ogundeji, "Multisig Best for Bitcoin Wallet Security But Only 11 Percent Use It ," *CoinTelegraph*, 2016, [Online; accessed 16-August-2020].

[22] S. e. a. Golfeder, "Securing Bitcoin wallets via a new DSA/ECDSA threshold signature scheme ."

[23] A. Russell, " What is a Certificate Authority (CA)?" 2019, [Online; accessed 16-August-2020].

[24] L. Zhou, F. B. Schneider, and R. Van Renesse, "Coca: A secure distributed online certification authority," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, p. 329–368, Nov. 2002. [Online]. Available: https://doi.org/10.1145/571637.571638