ARPA Network Litepaper

Penrose

August 24, 2022

1 Introduction

As a fundamental infrastructure of the Metaverse, the blockchain has proven its great significance of being a distributed global ledger. During the past decade, academic and industrial experts have made vast efforts to improve the privacy, scalability and interoperability of it. An effective means to enhance the blockchain is by verifiably outsourcing on-chain computation and storage to off-chain through threshold cryptography schemes. These schemes offer trust by their provable security and decentralization, which matches the distributed and trustless nature of the blockchain. At the same time, the blockchain can, as a bulletin board, serve as a reliable broadcast channel for these cryptographic algorithms. Therefore, combining the threshold cryptography with blockchains can simultaneously strengthen the blockchains and facilitate the implementation of the threshold cryptography.

In this paper, we propose the ARPA Network, a permissionless distributed network to provide essential threshold signature capability for blockchains and Web3. Roughly speaking, threshold signature is a protocol processing signature-related functions among a group of nodes through multi-party computation (MPC). It can help keeping secrets distributedly, reaching consensus by a majority vote, or hiding identity of signers. With the aid of the ARPA Network, one can build applications regrading secure key management, anonymous transaction, cross-chain messaging, quorum approval, distributed randomness generation, etc. Like the blockchain, trust of the ARPA Network comes from the distribution across independent nodes. Those nodes are grouped and execute the Boneh–Lynn–Shacham (BLS) threshold signature scheme. Thanks to the aggregatability of BLS signature and node management mechanism we apply, ARPA Network has the following features.

- Decentralization, ARPA nodes provides threshold signature service in a decentralized manner. Trust is distributed to multiple entities located in different regions running individual nodes. This manner offers better tamper-proof in the physical level.
- Flexibility, ARPA protocol supports a variety of signature applications. Users are allowed to customize their signature policy, import or export secrets, and choose security level
- Verifiability, ARPA signature scheme provides users the ability to trivially verify their signatures. By virtue of underlying cryptographic primitive, The signatures are unlikely to be forged or manipulated
- Non-interactivity, ARPA protocol avoids heavy synchronous communication in the signature generation phase. An non-interactive procedure guarantees reliable service status and flexible network topology.
- High availability, ARPA signature service keeps a high availability thanks to its decentralization and non-interactivity. With the growth of its network size, the downtime will reduce accordingly.

- Multiple Chain Supported, ARPA network is designed to adapt to multiple chains, which facilitates developers from multiverse building their applications. Our underlying threshold signature network may keep consistency between the different data replicas.

In the remainder of this paper, we first overview the existing threshold signature schemes and point out the reason why threshold BLS signature is suitable for distributed systems in section 2. Then section 3 lists out presuppositions and the building blocks of our design. A walk-through of our implemented protocol is given in section 4, including the cryptographic part and the node management mechanism. And section 5 presents the system design by showing the high-level architecture of the network. Finally, section 6 demonstrates Randcast as a use case of the ARPA Network.

2 Background

Threshold signature has been researched and applied a lot in the near few years. Many classical signature algorithms has been thresholdized, including BLS, ECDSA[8], EdDSA[14], RSA[7]. However, when applied to a distributed system, not all threshold signature schemes are well suited. For example, threshold ECDSA takes 3 to 13 synchronous communication rounds depending on different construction method[1]. Under real world conditions, transmission latency determines the performance of the MPC protocols. Multiple rounds of communications will certainly influence the average speed of threshold signature generation. Moreover, a failed node may even cause protocol abort. Fortunately, BLS signature utilizes a bilinear pairing which provides homomorphic mapping between the secret keys and the signatures. This makes asynchronous threshold signatue scheme possible.

BDN18 describes a BLS multi-signature for bitcoin, rather than a threshold signature. GLOW20 targets a distributed strong pseudorandom function, and tries to extend their DVRF to threshold BLS signature by replacing the DKG algorithm. AFAICS, it brings a bit more security to threshold signature at the cost of introducing new ZKP validations. Regarding Gennaro has proven JFDKG is sufficiently secure for discrete-log problem, our design is good enough for a threshold signature service.

3 Prosuppositions

While building an underlying threshold signature service for the blockchain, we should firstly clarify the security and communication model. The following outlined assumptions are highly relevant to the characteristics of the blockchain, where the system is distributed and permissionless while the participants are economically rational but potentially malicious.

3.1 Security Assumption

We assume that a static, malicious, honest-majority adversary. The attacker can corrupt up to t of the n parties in the network in one time, where $t < n/2$. The corrupted parties may divert from the prescribed protocol in any way. Considering the malicious behaviors, honest-majority is the best achievable threshold for protocols that provide both secrecy and robustness [9]. The static adversary would chooses the corrupted parties far ahead of time which means getting control of particular party is non-trivial. But it is possible that the corrupted parties may vary during a long time. Key rotation or refreshment scheme is introduced to cope with the long-term key exposure.

3.2 Communication Model

The distributed signature network is composed of a set of n parties $P_1, \cdots, P_n$ that connected by a complete network of private point-to-point channels. In addition, the parties have access to a dedicated broadcast channel. Several works[10, 11, 5] have researched on the threshold signature without a preset reliable broadcast. They assimilate various secret sharing schemes into reliable broadcast or consensus protocol. e.g. AVSS[5] merges a bivariate polynomial into Bracha's reliable broadcast [4]. Gennaro[9] leverages Byzantine fault tolerant (BFT) protocol[6] to build the DKG for the Internet. It can be concluded from these articles that constructing a reliable broadcast channel is equivalent to reaching a consensus among nodes. Considering there are kinds of blockchains and smart contracts that help us broadcast and record messages publicly, it is reasonable to offload this part of protocol to blockchains.

3.3 Synchrony Assumption

Based on the block generation mechanism, we may assume a partially synchronous communication model: protocol proceeds in synchronized rounds and messages are received by their recipients within some specified time bound. The block time of a blockchain that guarantees liveness can be used as the synchronized clock in the threshold signature. The drawback is that the typical value of the block time is several orders of magnitude larger than that of an Internet transmission. The partial synchrony opens up an attack way that, an adversary can observe the messages of the uncorrupted parties, then decide on his action in each round of the protocol, and still get his messages delivered to the rest of the parties on time. In the worst case, the adversary may speak last in every communication round to exploit the back-running[9].

Although asynchronous verifiable secret sharing scheme is intractable to design, there are several works researching on it. Unconditionally secure AVSS has a communication complexity of $\Omega(n^5)$, making it unrealistic to use. Comprimising the unconditional security assumption

4 ARPA Threshold Signature Protocol

Based on the assumptions raised previously, ARPA Network builds up the protocol by assimilating Joint-Feldman Distributed Key Generation (DKG)[13] into standard BLS signature[2]. Furthermore, ARPA Network combines the threshold BLS signature with the blockchain. In high level, The user requests the service by sending a transaction to the blockchain. The task will be assigned to one group of network nodes. The nodes collectively generate a signature on the message provided and return it back to the blockchain. In the process, the threshold BLS signature is responsible for generating a decentralized tamper-proof signature, and the blockchain provides a reliable broadcast channel as well as coordinating functionality. The signature scheme is comprised of two main parts, distributed key generation and threshold signature. ARPA threshold signature protocol also defines the procedure of nodes registration, secession and grouping.

Before diving deep into the protocol, we have to provide some math preliminaries. The magical, widely-used BLS signature relies on bilinear pairing.

4.1 Distributed Key Generation

As an essential component of threshold cryptosystems, a distributed key generation (DKG) protocol is reponsible for generating private and public keys of participants. The underlying secret sharing scheme used in the DKG decides the relationship of key shares hold by each participant. This also fundamentally determines the key management policy and the signature generation algorithm used in the threshold signature scheme. Why JF-DKG is secure enough for TSS BLS

Algorithm 1 Joint-Feldman Distributed Key Generation[9]

```
Require: Adversary threshold $t$, group size $n$, a elliptic curve $\mathbb{G}_2$ with a prime order $r$ and a generator $g_2$.
Ensure: Shamir secret shares $sk_i$ for party $P_i$ respectively.
1: Each party $P_i$ chooses a $f_i(z) = \sum_{k=0}^{t} a_{ik} z^k$, where $a_{ik} \in_R \mathbb{Z}_r^*$, broadcasts commitment $C_{ik} = g_2^{a_{ik}}$ for $k \in [0, t]$. Each party $P_i$ computes the shares $s_{ij} = f_i(j) \bmod r$ for $j \in [1, n]$ and sends $s_{ij}$ to party $P_j$ secretly.
2: Each party $P_j$ verifies the shares his received $g_2^{s_{ij}} = \prod_{k=0}^{t} (C_{ik})^{j^k}$ for $i \in [1, n]$. If the check for an index $i$ fails, $P_j$ raises a compliant against party $P_i$
3: Party $P_i$ reveals the shares corresponding to raised complaints. Each party $P_j$ check the validity between complaints and equations. Any failed party will be contained in local disqualified set of each party. $QUAL$ is defined as the set of non-disqualified parties.
4: The public key $y$ is computed as $pk = \prod_{i \in QUAL} C_{i0}$. The secret shared value $sk$ itself is not computed by any party, but it is equal to $sk = \sum_{i \in QUAL} a_{i0} \bmod r$. Each party $P_j$ sets his share of the secret as $sk_j = \sum_{i \in QUAL} s_{ij} \bmod r$, and corresponding public key share as $pk_j = \prod_{i \in QUAL} g_2^{s_{ij}} = \prod_{i \in QUAL} \prod_{k=0}^{t} (C_{ik})^{j^k}$
```

4.2 threshold BLS signature

BLS signature is first proposed as a short signature scheme where the signatures consist only of a single group element. ElGamal type schemes such as digital signature algorithm (DSA) and elliptic curve digital signature algorithm (ECDSA) produces signatures comprised of a pair of integers. Therefore, BLS signatures are about half of the length of those produced by other widely used schemes at the same security level[12]. BLS signatures utilize a bilinear pairing which offers several interesting features. Firstly, BLS signatures are deterministic given particular message and keypair, unlike ECDSA which requires fresh random value for each signing. This prevents signers from biasing results by repeated signing attempts. Secondly, BLS signatures are aggregatable which means specific computations on generated signatures are possible. This makes a difference between the threshold BLS signature and other threshold signatures. Multiple rounds of synchronous communication is unnecessary for combining partial BLS signatures.

The scheme is provably secure under the CDHP, assuming the hash function acts like a random oracle.

The BLS signature scheme consists of the following sub-procedures. Without loss of generality, we represent signatures by elements in $\mathbb{G}_1$ to achieve a more compact signature.

Key Generate To generate a key pair, a signer first chooses a random secret $sk \in_R \mathbb{Z}_r^*$ and computes the corresponding public key as $pk = g_2^{sk} \in \mathbb{G}_2$.

Sign To compute a BLS signature σ on an arbitrary message m, the signer computes $\sigma = sk \times H(m) \in \mathbb{G}_1$. $H(m)$ is a hash-to-curve function that maps an arbitrary bit string to an element in $\mathbb{G}_1$

Verify To verify the validity of a BLS signature σ on a given message m, the verifier checks if $(g_2, pk, H(m), \sigma)$ is a Diffie-Hellman quadruple which will be the case that $e(\sigma, g_2) = e(H(m), pk)$ holds

A threshold signature scheme is basically computing a signature among a group of independent nodes by multi-party computation (MPC). Considering the different processes of diverse signature schemes, the construction of their threshold versions will be very dissimilar. For example, the threshold ECDSA involves homomorphic multiplication of encrypted messages which results in a complex sub-protocol called Multiplicative to Additive (MtA). It takes several rounds of synchronous communications to process. As for the BLS signature, thanks to the its homomorphic property, the threshold BLS signature is neat and clear. Its sub-procedures are as follows.

Distributed Key Generate The n parities jointly execute Algorithm 1 to generate a group public key $pk \in \mathbb{G}_2$, a virtual private key $sk \in \mathbb{Z}_r^*$, and their shares pk_i, sk_i.

Partial Signature Generate To sign a message m, each party P_i individually sign the partial BLS signature $\sigma_i = sk_i \times H(m)$ by his private key share sk_i.

Partial Signature Verify To validate a partial BLS signature, one uses the corresponding public key share to check if $e(H(m), pk_i) = e(\sigma_i.g_2)$ holds.

Signature Reconstruct To reconstruct the BLS signature σ on m, one have to collect $t+1$ valid and independent partial signatures, then compute

$$\sigma = \prod_{k=0}^{t} \sigma_{i_k} \prod_{m=0, m\neq k}^{t} \frac{i_m}{i_m - i_k}, \quad i_k, i_m \in [1, n].$$

Verify To verify the validity of a BLS signature σ on a given message m, the verifier checks if $(g_2, pk, H(m), \sigma)$ is a Diffie-Hellman quadruple, which will be the case that $e(\sigma, g_2) = e(H(m), pk)$ holds.

Thanks to the properties of the Shamir secret-sharing (SSS) scheme, the generated signature σ is identical no matter which partial signatures are chosen during reconstruction. Meanwhile, the determinism of BLS signature also guarantees the immutability of the threshold signature no matter what kinds of (computationally bounded) attacks are conducted . We now have a decentralized, verifiable, tamper-proof threshold BLS signature scheme.

4.3 Nodes Management Protocols

4.3.1 Node Grouping Agreement

4.3.2 Node Registration

4.4 Attack Vectors & Countermeasures

group corruption

5 System Architecture

6 Randcast

Like the physical world, where the whole universe is built upon random motions of molecules, random numbers are ubiquitous and essential in the Metaverse. A trustworthy and reliable pseudo-random number generator is a cornerstone to either blockchain infrastructures or applications built upon. In this section, we instantiate Randcast as a randomness generation application of the ARPA Network. Inheriting from the threshold signature network, Randcast also has the following features.

- Decentralization, Randcast produces random number in a decentralized manner, entropy is gathered from multiple entities located in different regions running individual nodes. This manner offers unpredictability and fairness in the physical level.

- Uniqueness, Randcast generates random numbers depending only on request messages and node secrets. Given certain signing group and user input, the randomness is unique and tamper-proof which mitigates corruption inside the network.
- Verifiability, Randcast provides everyone the ability to check the validity of random number. By virtue of underlying cryptographic primitive, The random number is unlikely to be forged or manipulated
- Non-interactive, Randcast avoids heavy synchronous communication in the random generation phase. An non-interactive procedure guarantees a better service status.
- High availability, Randcast owns a high availability thanks to its decentralization and non-interactive. With the growth of its network size, the downtime will reduce accordingly.
- Multiple Chain Supported, Randcast is designed to adapt to multiple chains, which facilitates developers from multiverse casting "randomness" spells. Our underlying threshold signature network will keep consistency between the different data replicas.

A BLS 12-381 specification

BLS12-381 is a specific curve of a pairing-friendly curve family. It has an embedded degree as 12 and a 381-bit field prime. The public parameters are outlined as follows[3].

u = -0x	`d2010000 00010000`
k =	`12`
q = 0x	`1a0111ea 397fe69a 4b1ba7b6 434bacd7 64774b84 f38512bf 6730d2a0 f6b0f624 1eabfffe b153ffff b9feffff ffffaaab`
r = 0x	`73eda753 299d7d48 3339d808 09a1d805 53bda402 fffe5bfe ffffffff 00000001`
$E(\mathbb{F}_q) :=$	$y^2 = x^3 + 4$
$\mathbb{F}_{q^2} :=$	$\mathbb{F}_q[i]/(x^2 + 1)$
$E'(\mathbb{F}_{q^2}) :=$	$y^2 = x^3 + 4(i + 1)$

References

[1] Jean-Philippe Aumasson, Adrian Hamelink, and Omer Shlomovits. A survey of ecdsa threshold signing. *Cryptology ePrint Archive*, 2020.

[2] Dan Boneh, Sergey Gorbunov, Riad S. Wahby, Hoeteck Wee, Christopher A. Wood, and Zhenfei Zhang. BLS Signatures. Internet-Draft draft-irtf-cfrg-bls-signature-05, Internet Engineering Task Force, June 2022. Work in Progress.

[3] Sean Bowe. BLS 12-381: New zk-SNARK Elliptic Curve Construction. `https://electriccoin.co/blog/new-snark-curve/`, 2017. [Online; accessed 31-May-2022].

[4] Gabriel Bracha. An asynchronous [(n-1)/3]-resilient consensus protocol. In *Proceedings of the third annual ACM symposium on Principles of distributed computing*, pages 154–162, 1984.

[5] Christian Cachin, Klaus Kursawe, Anna Lysyanskaya, and Reto Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 88–97, 2002.

[6] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

[7] Ivan Damgård and Maciej Koprowski. Practical threshold rsa signatures without a trusted dealer. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 152–165. Springer, 2001.

[8] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ecdsa with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1179–1194, 2018.

[9] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, 2007.

[10] Aniket Kate and Ian Goldberg. Distributed key generation for the internet. In *2009 29th IEEE International Conference on Distributed Computing Systems*, pages 119–128. IEEE, 2009.

[11] Aniket Kate, Yizhou Huang, and Ian Goldberg. Distributed key generation in the wild. *Cryptology ePrint Archive*, 2012.

[12] Alfred Menezes. An introduction to pairing-based cryptography. *Recent trends in cryptography*, 477:47–65, 2009.

[13] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.

[14] Douglas R Stinson and Reto Strobl. Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates. In *Australasian Conference on Information Security and Privacy*, pages 417–434. Springer, 2001.