Network3: A Protocol for Decentralized, Authenticated, Anonymous, and Reliable Data Transmission and Computation

Network3 Labs, 2023

***Abstract*—This paper introduces Network3, a novel blockchain-oriented protocol designed for decentralized, authenticated, anonymous, and reliable data transmission and computation. Responding to the inherent trustworthy challenges of the mainnet, Network3 integrates an efficient anonymous certificateless signcryption (CLSC) algorithm, a data correctness verification mechanism, an IP anti-tracking measure, and a decentralized reliable federated learning (FL) framework. The CLSC algorithm ensures identity authentication and secure data sharing under anonymity. The data verification mechanism addresses potential data inaccuracies, blending homomorphic encryption, secret sharing, and Reed-Solomon coding. The IP anti-tracking feature facilitates completely anonymous communication. Moreover, Network3 provides a decentralized AI framework by handling the intractable challenges encountered by the existing FL technologies. By bolstering the capabilities of mainnet and integrating unique features such as anonymous communication, privacy-preserving authentication, decentralized AI, Network3 heralds notable progress in blockchain ecosystems. It paves the way for improvements in decentralization, anonymity, intelligence, and reliability, fostering an environment conducive to future advancements.**

***Index Terms*—Web3.0, Data Transmission, Computation, Federated Learning, Signcryption, Decentralization, Anonymity, Intelligence.**

I. Introduction

In the emerging era of Web3.0, Ethereum leads the way as an open-source, blockchain-based platform, enabling developers to craft and deploy smart contracts and decentralized applications (dApps). Despite the transformative potential of these features, the inherent scalability and security challenges associated with most mainstream blockchain mainnets impede wider adoption [1]–[3]. Various projects like Filecoin [4], Arweave [5], Sia [6], and Flux [7] have addressed these issues from the standpoints of decentralized storage and computation. However, a gap still exists in the area of decentralized communication for reliable data transmission. This is critical because data exchange impacts the reliability of transmitted transactions, the security of communicating parties, and the overall user experience within the blockchain ecosystem. Therefore, developing a decentralized data transmission protocol that enhances mainstream blockchain mainnets' scalability and security is of crucial significance.

This paper presents Network3, a novel Layer2 protocol designed explicitly for decentralized, authenticated, anonymous, intelligent, and reliable data transmission and computation. The driving force behind this research lies in the intrinsic need to advance mainstream blockchain mainnets' capabilities, allowing for larger-scale, more secure applications, while simultaneously preserving anonymity and data integrity in an ever-increasing data-driven society.

Network3 responds to a series of stringent requirements. It aims to ensure seamless and secure data transmission, preserve anonymity, resist diverse attack vectors, and tackle the paramount challenge of inaccurate data from receivers. Moreover, it seeks to improve upon existing schemes in terms of computational efficiency and communication costs, both vital for real-world deployment.

Current state-of-the-art protocols and algorithms fall short in addressing these needs in a comprehensive manner. Most fail to achieve a harmonious balance between anonymity and decentralization, or they struggle with efficiency and robustness against common attacks [8]–[11]. Others lack mechanisms to validate the accuracy of received data [12]. Therefore, a new approach that tackles these problems holistically, like Network3, is critically needed in the crypto-ecosystem.

Network3 revolves around key technologies, including an efficient anonymous certificateless signcryption (CLSC) algorithm, a decentralized reputation mechanism, and an IP anti-tracking measure. The anonymous CLSC algorithm offers a unique blend of identity authentication and secure data sharing under anonymous conditions, backed by thorough security and performance analysis. The decentralized data correctness verification mechanism, infused with homomorphic encryption, secret sharing, and Reed-Solomon coding, provides a solution to potential inaccuracies in received data. The IP anti-tracking measure ensures a fully anonymous data transmission experience.

The contributions of Network3 are summarized as follows.

- Firstly, we propose a highly efficient anonymous CLSC algorithm for authenticated transmission and offer a detailed algorithm construction with security and performance analysis, underscoring the algorithm's effectiveness and practicality.
- Secondly, we devise a decentralized rating-based data correctness verification mechanism to address the issue of data inaccuracies at the receiving end.
- Thirdly, we design a novel IP anti-tracking mechanism to achieve completely anonymous data transmission and protect the inherent freedom for the ubiquitous interaction and cooperation.
- Finally, we present a groundbreaking decentralized federated learning framework, thoughtfully designed to tackle

the core challenges associated with realizing practical and dependable decentralized AI capabilities.

These innovations promise to significantly augment mainnet capabilities and contribute to the ongoing discussion around blockchain openness, security, anonymity, and intelligence. Moreover, the achieved features — anonymity, authenticity, reliability, and decentralized intelligence — set Network3 apart as a groundbreaking solution in the realm of Layer2 protocols.

The remainder of this paper is organized as follows: Section II provides a detailed introduction of the proposed CLSC-enabled efficient anonymous authentication. Section III describes the data correctness mechanism. Section IV delves into the IP anti-tracking mechanism. Section V incorporates edge computing to facilitate outsourced computation. Section VI demonstrates the decentralized FL framework. Finally, Section VII concludes the paper, providing an overview of the innovations and features achieved by Network3.

II. Efficient Anonymous Authentication

A. Preliminary

1) Bilinear Pairing: Let $\mathbb{GF}(p)$ denote a finite field, where p is a large prime number. We consider an elliptic curve $E_p(a,b)$ over $\mathbb{GF}(p)$, which can be defined as the set of ordered pairs $(x,y) \in \mathbb{GF}(p) \times \mathbb{GF}(p)$ that fulfill the condition $y^2 \equiv x^3 + ax + b \pmod p$, given that $a, b \in \mathbb{GF}(p)$ and $4a^3 + 27b^2 \not\equiv 0 \pmod p$. This leads to the construction of an additive cyclic group, denoted $\mathbb{G}_1$, and a multiplicative cyclic group, $\mathbb{G}_2$, both of which have the prime order p. These groups are constituted of all points residing on the elliptic curve, complemented by the point at infinity. A random generator of $\mathbb{G}_1$ is designated as P. We define a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ (known as a type-1 pairing) that adheres to the following three properties:

- **Bilinearity:** For any $X, Y \in \mathbb{G}_1$ and any $a, b \in \mathbb{Z}_p^*$, it holds that $e(aX, bY) = e(X, Y)^{ab}$.
- **Non-degeneracy:** If we designate $1_{\mathbb{G}_2}$ as the identity element of $\mathbb{G}_2$, it should be ensured that $e(X, Y) \neq 1_{\mathbb{G}_2}$ for any $X, Y \in \mathbb{G}_1$.
- **Computability:** For any $X, Y \in \mathbb{G}_1$, the value $e(X, Y)$ can be efficiently computed.

2) Security Assumptions: Two fundamental problems play a crucial role in ensuring the robustness of Network3 protocol. These problems, known as the Elliptic Curve Discrete Logarithm Problem (ECDLP) and the Elliptic Curve Computational Diffie-Hellman Problem (ECCDHP), are outlined below:

- **ECDLP:** Given an elliptic curve $E_p(a,b)$ over a finite field $\mathbb{GF}(p)$, and two points $P, Q \in \mathbb{G}_1$, the ECDLP is to find an integer x such that $Q = xP$ if one exists. It is conjectured that no efficient algorithm can solve this problem, making it the foundation of the security in elliptic curve cryptography. The intractability of ECDLP ensures that, even if an attacker knows P and Q, they cannot feasibly compute the integer x.
- **ECCDHP:** For an elliptic curve $E_p(a,b)$ over a finite field $\mathbb{GF}(p)$, and given $P, aP, bP \in \mathbb{G}_1$, the ECCDHP asks for the computation of the point abP. In the realm of cryptographic systems, ECCDHP is considered computationally hard, assuming the hardness of the ECDLP. In simpler terms, if one can solve ECCDHP efficiently, then they can also solve ECDLP efficiently, which is conjectured to be infeasible.

B. Algorithm Syntax

The foundational cryptographic underpinning of the efficient anonymous authentication protocol in Network3 is the avant-grade certificateless signcryption, which simultaneously provides encryption and signature, thereby ensuring data confidentiality, integrity, authentication. To bolster the protocol's efficiency, we incorporate the concept of online/offline encryption into the signcryption operations. The key solution to provide immaculate identity anonymity for mainstream blockchain mainnet users lies in the construction of a one-time anonymized public key for both communicating parties during each interaction. Attackers are incapable of linking the anonymized public key to any real identity. A smart contract, denoted as SC_{KGC}, supplants a centralized KGC to produce public parameters, generate partial private keys, and distribute them to the corresponding users. Any legitimate user can query public parameters. Specifically, the anonymous authentication scheme is composed of the following eight probabilistic polynomial time (PPT) algorithms.

- $\mathbf{Setup}(1^k)$. The setup algorithm is run by the smart contract SC_{KGC} deployed on mainstream blockchain mainnets. It takes as input the security parameter k, and outputs the master secret key $MSK = s$ and system parameter $param$. For simplicity, the public parameters are omitted in the descriptions of the subsequent algorithms.
- $\mathbf{PPKGen}(ID, MSK)$. This algorithm is run by smart contract SC_{KGC} to create partial private keys for mainstream blockchain users. Taking as input a user's identity $ID \in \{0,1\}^*$ and the master secret key MSK, the algorithm outputs a partial private key PPK_{ID}.
- $\mathbf{KeyGen}(PPK_{ID})$. The algorithm is run by the user. Given a partial private key PPK_{ID}, this algorithm outputs the public key PK_{ID}, private key SK_{ID}, and user identity tag Tag_{ID}.
- $\mathbf{Anonymization}(PK_A, PK_B)$. The algorithm is run by the data sender. Given the public key PK_A of the sender and the public key PK_B of the receiver, this algorithm outputs the anonymized public key AK_A and AK_B of both communicating parties.
- $\mathbf{OffSign}(SK_A, PK_A, PK_B, AK_B)$. The algorithm is performed by the data sender ID_A in an offline mode. Specifically designed for blockchain users with constrained resources, this phase enables offloading of computational tasks to reliable offline equipment. Taking as input the sender's private key SK_A and public key PK_A, and the receiver's public key PK_B, this algorithm outputs an offline signcryption result δ. Note that the transmitted message is not required in this phase.
- $\mathbf{OnSign}(\delta, m, Tag_A, PK_B, AK_B)$. The algorithm is run by the data sender ID_A. Given a message m, an offline signcryption result δ, the identity tag Tag_A, the receiver's

public key PK_B, and the receiver's anonymized public key AK_B, this algorithm outputs a ciphertext σ.

- $\mathbf{KeyRec}(Mat_A, Mat_B)$. The algorithm is run by the smart contract SC_{KGC} to recover the anonymized public key AK_B for the receiver ID_B. Given the associated materials Mat_A and Mat_B, provided by the sender and the receiver respectively, this algorithm verifies the validity of AK_B before transmitting it to ID_B.
- $\mathbf{Unsigncryption}(\sigma, AK_A, AK_B)$. The algorithm is run by the data receiver ID_B. Given the ciphertext σ, the sender's anonymized public key AK_A, and the receiver's anonymized public key AK_B, this algorithm outputs the exchanged message m if the verification is successful; otherwise, it outputs a failure symbol $\perp$.

C. Concrete Construction

In this section, we introduce the concrete construction of the proposed anonymous authentication scheme.

- $\mathbf{Setup}(1^k)$. Given the security parameter k, two cyclic groups $(\mathbb{G}_1, \mathbb{G}_2)$ are chosen, where $\mathbb{G}_1$ represents an additive group of a large prime order $p > 2^k$ and $\mathbb{G}_2$ corresponds to a multiplicative group of the identical prime order p. A mapping is acknowledged, such that $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. P, a random generator in $\mathbb{G}_1$, is selected and $g = e(P, P)$ is calculated. The system utilizes four hash functions as follows: $H_1 : \{0,1\}^* \to \mathbb{Z}_p^*$, $H_2 : \mathbb{G}_1 \to \mathbb{Z}_p^*$, $H_3 : \mathbb{G}_2 \to \{0,1\}^n$, and $H_4 : \{0,1\}^n \times \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{Z}_p^*$, where n denotes the number of bits within the message to be transmitted. The smart contract SC_{KGC} then randomly selects $s \in \mathbb{Z}_p^*$ as the master secret key MSK, and defines the system public key $MPK = s \cdot P$. Following this, it publishes system parameters $Param = \{\mathbb{G}_1, \mathbb{G}_2, e, n, p, P, MPK, g, H_1, H_2, H_3, H_4\}$ on mainstream blockchain mainnets and privately stores MSK.
- $\mathbf{PPKGen}(ID, MSK)$. The data sender computes $A_1 = ID_A + ID_A \cdot MPK$ and $A_2 = ID_A \cdot P$, and sends $\{A_1, A_2, t_0\}$ to the smart contract SC_{KGC} through public channel, where $t_i, i \in \mathbb{N}$ is the current timestamp. Upon receiving the message, SC_{KGC} is able to get the real identity of the data sender by means of $ID_A = A_1 - s \cdot A_2$. Subsequently, SC_{KGC} calculates $PPK_A = s \cdot H_1(ID_A)$ and returns $A_3 = PPK_A + s \cdot A_2$ to the data sender, where PPK_A is the generated partial private key.
- $\mathbf{KeyGen}(PPK_{ID})$. Upon receiving A_3, the data sender calculates the partial private key through $PPK_A = A_3 - ID_A \cdot MPK$. Then, he/she verifies the accuracy of the partial private key generation by assessing the validity of the equation $e(PPK_A, P) = e(s \cdot H_1(ID_A), P) = e(H_1(ID_A), MPK)$. The data sender accepts PPK_A if the equation holds; otherwise, he/she invokes the smart contract SC_{KGC} for regeneration.
  Afterwards, the sender selects a random integer $a_A \in \mathbb{Z}_p^*$ and computes $A_4 = PPK_A + a_A$. Following this, the public key is calculated as $PK_A = A_4 \cdot P \in \mathbb{G}_1$. Moreover, the sender chooses another secret value $b_A \in \mathbb{Z}_p^*$, calculates $A_5 = b_A + H_2(PK_A)$ and the identity tag $Tag_A = g^{A_5^{-1}}$. Finally, the secret key is generated as $SK_A = A_5^{-1} \cdot A_4^{-1} \cdot P$. The secret values a_A, b_A along with the secret key SK_A are securely stored in local trusted storage, while the public key PK_A and user identity tag Tag_A are published on mainstream blockchain mainnets.
- $\mathbf{Anonymization}(PK_A, PK_B)$. Given the sender's public key PK_A and the receiver's public key PK_B, the sender selects a random number $\theta \in \mathbb{Z}_p^*$ to construct the one-time anonymized public key $AK_A = (\theta + H_2(\theta \cdot PK_B)) \cdot PK_A$ for the sender and $AK_B = (\theta + H_2(\theta \cdot PK_A)) \cdot PK_B$ for the receiver. These keys, AK_A and AK_B, provide temporary anonymity for the communicating parties. Their embeddings within transactions are openly published on the blockchain.
- $\mathbf{OffSign}(SK_A, PK_A, PK_B, AK_B)$. The sender selects a random integer $\alpha \in \mathbb{Z}_p^*$ and computes $\beta = \alpha(\theta + H_2(\theta \cdot PK_A))$, $r = g^\beta$. In addition, it computes $U = \alpha \cdot AK_B = \alpha(\theta + H_2(\theta \cdot PK_A))PK_B$ and $V = (\theta + H_2(\theta \cdot PK_B))^{-1} SK_A + H_3(r) \cdot P$. Finally, this algorithm outputs an offline signcryption result $\delta = \{\beta, r, U, V\}$.
- $\mathbf{OnSign}(\delta, m, Tag_A, PK_B, AK_B)$. Upon receiving δ, the sender computes $c = m \oplus H_3(r)$. Furthermore, it computes $h = H_4(m, Tag_A, AK_B, r)$ and $W = (\beta + h)(\theta + H_2(\theta \cdot PK_B))^{-1} \cdot A_4^{-1} \cdot P$. Finally, the data sender encapsulates the ciphertext $\sigma = \{c, U, V, W, t_1\}$ into a transaction and publishes it on mainstream blockchain mainnets.
- $\mathbf{KeyRec}(Mat_A, Mat_B)$. This algorithm aims to recover the anonymized public key AK_B for the receiver. Initially, the sender generates $L_A = (\theta + H_2(\theta \cdot PK_A)) \cdot P$, $\widehat{L_A} = AK_B + (\theta + H_2(\theta \cdot PK_A))MPK$, and submits $Mat_A = \{L_A, \widehat{L_A}, t_2\}$ to the blockchain network. The receiver then selects two random numbers $\iota, \kappa \in \mathbb{Z}_p^*$, computes $L_B = \kappa \cdot P$, and $\widehat{L_B} = \iota \cdot B_5 + \kappa \cdot MPK$. Following this, the receiver initiates a query for the anonymized public key AK_B by transmitting $Mat_B = \{L_B, \widehat{L_B}, \iota^{-1} SK_B, t_3\}$ to the SC_{KGC} smart contract. After validating the freshness of the timestamp t_3, SC_{KGC} calculates $\iota \cdot B_5 = \widehat{L_B} - s \cdot L_B$. In parallel, the anonymized public key of the receiver can be recovered by $AK_B = \widehat{L_A} - s \cdot L_A$. Subsequently, SC_{KGC} verifies the following equation:

$$\begin{aligned} & e(\iota \cdot B_5 \cdot AK_B, \iota^{-1} SK_B) \\ &= e(B_5(\theta + H_2(\theta PK_A))PK_B, SK_B) \\ &= e(B_5(\theta + H_2(\theta PK_A))B_4 \cdot P, B_4^{-1} B_5^{-1} P) \\ &= e((\theta + H_2(\theta PK_A)), P) \\ &= e(L_A, P). \end{aligned} \tag{1}$$

  If this equation fails to hold, the algorithm outputs a failure symbol $\perp$ implying that the anonymized public key AK_B is incorrect. Otherwise, SC_{KGC} calculates $AK_B + s \cdot L_B$ and returns the result to the receiver. Finally, the receiver can construct his/her anonymized public key through $AK_B = AK_B + s \cdot L_B - \kappa \cdot MPK$.
- $\mathbf{Unsigncryption}(\sigma, AK_A, AK_B)$. The data receiver searches for the corresponding transaction information

based on the obtained anonymized public key AK_B and retrieves the ciphertext $\sigma = \{c, U, V, W, t_1\}$. First, the receiver calculates r through the following equation:

$$\begin{aligned} e(U, B_4^{-1}P) &= e(\alpha(\theta + H_2(\theta \cdot PK_A))B_4P, B_4^{-1}P) \\ &= e(\alpha(\theta + H_2(\theta \cdot PK_A))P, P) \\ &= e(\beta P, P) \\ &= g^{\beta} = r. \end{aligned} \tag{2}$$

Then, the receiver obtains the original message through $m = c \oplus H_3(r)$. Afterwards, the receiver can get the identity tag Tag_A of the sender according to the following equation:

$$\begin{aligned} e(AK_A, V - H_3(r)P) &= e((\theta + H_2(\theta \cdot PK_B)) \cdot PK_A, \\ &\quad (\theta + H_2(\theta \cdot PK_B))^{-1} SK_A) \\ &= e(A_4P, A_4^{-1}A_5^{-1}P) \\ &= e(P, A_5^{-1}P) \\ &= g^{A_5^{-1}} = Tag_A. \end{aligned} \tag{3}$$

Subsequently, the receiver could compute $h = H_4(m, Tag_A, AK_B, r)$ and verify whether $r = e(AK_A, W)$ holds according to the following equation:

$$\begin{aligned} e(AK_A, W)g^{-h} &= e((\theta + H_2(\theta PK_B))PK_A, \\ &\quad (\beta + h)(\theta + H_2(\theta PK_B))^{-1}A_4^{-1}P)g^{-h} \\ &= e(A_4P, (\beta + h)A_4^{-1}P)g^{-h} \\ &= e(P, (\beta + h)P)g^{-h} \\ &= g^{\beta+h}g^{-h} = r. \end{aligned} \tag{4}$$

If this equation is valid, it validates that the r calculated in Equation 2 is accurate, signifying that the message m originates from the legitimate sender who has the anonymized public key AK_A. This further assures that the message was not subject to any form of tampering during transmission. However, if the equation does not hold, the algorithm will output a failure symbol $\perp$, indicating a discrepancy.

The entire processes are illustrated in Fig. 1.

D. Security Analysis

In this section, we will discuss several potential strengths and security features of the proposed authentication scheme.

- **Anonymity & Confidentiality:** The primary purpose of anonymity is to conceal a user's real identity, which our scheme achieves. Despite the fact that an adversary can capture the transmitted messages, they cannot infer the true identity of the sender due to the use of anonymized public keys and identity tags, which only the receiver can interpret. Other blockchain users will only know that a transaction has occurred between two anonymous users, without knowledge of their actual identities. Additionally, the transmitted data are kept private through the XOR encryption, preventing data disclosure against eavesdropping.
- **Resistance to KGC Compromised Attack:** Traditional anonymous certificateless schemes often feature a centralized KGC. These can be vulnerable to compromise, leading to potential security risks. Our scheme replaces the traditional KGC with a smart contract, which greatly increases the difficulty of a successful attack by requiring control of over 51% of the mainstream blockchain ecosystem.
- **Replay Attack Prevention:** Our scheme defends against replay attacks, where an adversary intercepts and resends a signature, by incorporating timestamps into the querying and signcryption processes. Every verifier needs to check the timestamp's validity, which limits the adversary's ability to trick the verifier with a previously intercepted signature.
- **Man-in-the-Middle Attack Defense:** To prevent an adversary from intercepting and altering all transmitted messages, we implement secure encryption with elliptic curves using L_B submitted by the receiver. This ensures that only the intended receiver, who knows the random value κ, can decrypt the returned anonymized public key. The underlying mathematical complexity is based on the hardness of ECDLP.
- **Forgery Attack Resistance:** Given the security assumption previously mentioned, the likelihood of an adversary successfully forging a valid signature is negligible. The receiver can accurately verify the result using the anonymized public key provided by the smart contract and the corresponding ciphertext, as demonstrated in the unsigncryption algorithm.
- **Protection Against Big Data Clustering Attack:** All transaction information is publicly visible on the blockchain network, which can potentially be exploited by adversaries to infer users' real information. Our scheme mitigates this risk by using one-time anonymized public keys, preventing adversaries from associating multiple transactions with a specific user.

Therefore, the proposed scheme can successfully defend against common attacks to ensure security and privacy in blockchain transactions.

E. Performance Evaluation

In this section, we analyze the computation and communication cost of the proposed anonymous certificateless signcryption scheme with those of Xiong [13], Zhou *et al.* [14], Tseng *et al.* [15], Mandal *et al.* [16], and Xu *et al.* [17]. The comparative results are shown in Table I. Since the dominating factors in each schemes' computational overheads are the exponentiation operation, point multiplication operation, and bilinear pairing operation, we can draw an intuitive conclusion according to Table I that the proposed scheme presents higher computation efficiency in comparison to its counterparts, especially in the signcryption phase. Additionally, our scheme excels in terms of communication cost, leading to a reduction in bandwidth usage and data transmission latency. These distinct benefits underscore the innovative nature and valuable contribution of Network3 protocol.

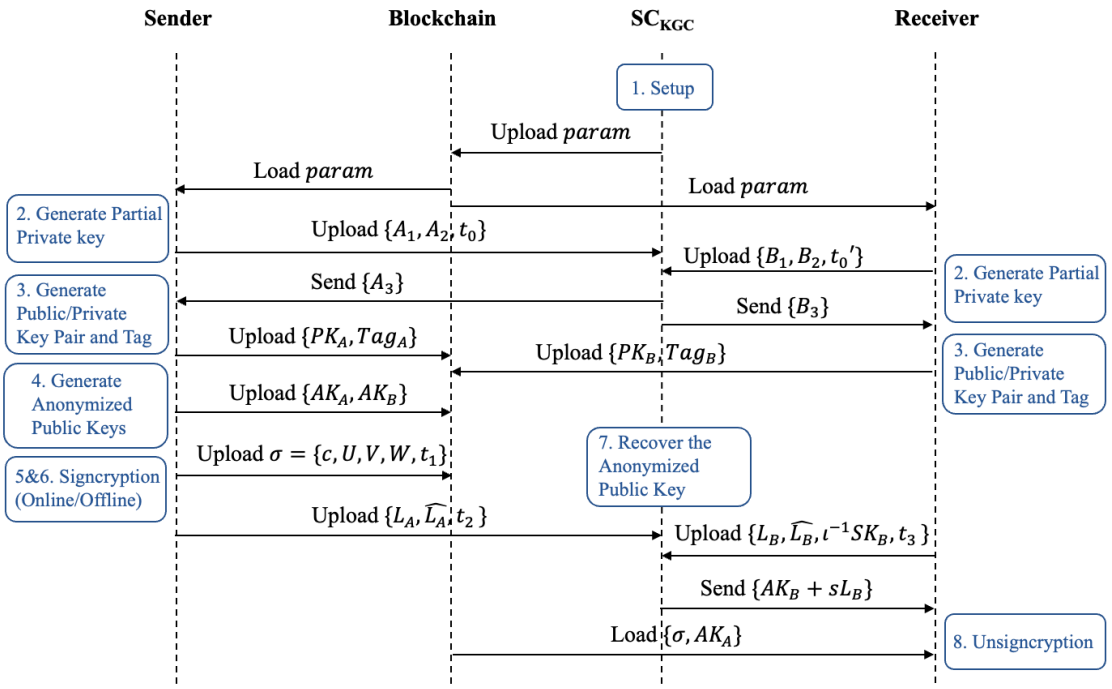

Fig. 1. Anonymous authentication scheme processes.

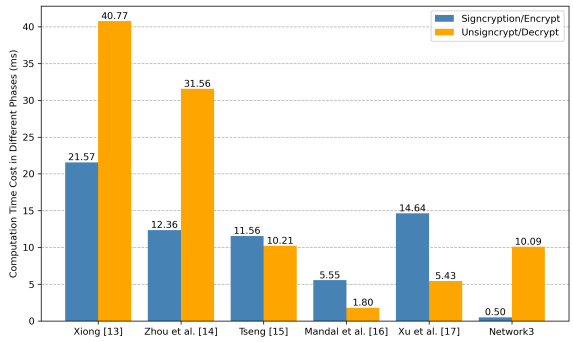

Fig. 2. Computation Overheads of Different Schemes.

To offer a clear evaluation, we employ the highly-regarded Charm library to gauge the execution time of key cryptographic processes. Our tests are performed on a MacBook Pro equipped with an Intel Core i7 @ 2.2 GHz, 16 GB DDR4, and running on Mojave 10.14.6. For elliptic curve and pairing-based operations, we opt for the MNT159 curve. The results, averaged from 100 test runs, are presented in Table II. Based on the data in Table I, we calculate the computational time for all the examined schemes, as illustrated in Figure 2. Notably, Network3 displays remarkable signcryption efficiency in comparison to other schemes, boasting a peak reduction rate of 97%. Its unsigncryption time also stands out favorably. In sum, these findings underscore the practicality and viability of Network3.

III. Data Correctness Verification Mechanism

Malicious or selfish data senders might attempt to transmit deceptive or counterfeit data to unsuspecting recipients. Even more severe, toxic data can compromise the system's normal operation. Therefore, an efficient mechanism for verifying data correctness is crucial. Such a mechanism empowers crypto users to validate the data they receive and, if necessary, lodge a complaint. Consequently, malicious data senders who behave irresponsibly should face punitive measures. Network3 has designed a decentralized, privacy-preserving mechanism for verifying data correctness, which leverages the ratings given to transmitted data by stakeholders in the blockchain network. The specific procedures of this mechanism can be divided into seven phases, which are outlined below.

A. Initialization

Upon discovering that the received data is incorrect, the data receiver can activate the correctness verification contract, denoted as SC_{CV}, by submitting evidence to prove the inaccuracy of the data received. The SC_{CV} then assembles a decentralized verification committee composed of m raters by randomly issuing verification invitations to stakeholders within the blockchain. In each iteration, the minimum number of stakeholders are selected and contacted. For instance, in the first iteration, m raters are contacted. If only x of them respond, an additional $(m - x)$ raters will be contacted in the second iteration. Alternatively, the SC_{CV} can broadcast requests to all raters; however, this may result in redundant replies as only m responses are needed. Upon the formation of the verification committee, the SC_{CV} disseminates the evidence of data inaccuracy to each rater for decentralized evaluation and rating.

TABLE I
COMPUTATION AND COMMUNICATION COSTS OF THE ANONYMOUS AUTHENTICATION SCHEME

Schemes	Signcrypt/Encrypt	Unsigncrypt/Decrypt	Ciphertext Size
Xiong [13]	$4T_{pair} + 9T_{exp}$	$9T_{pair} + 3T_{exp}$	$4\|\mathbb{G}_1\| + \|\mathbb{G}_2\| = 3104\ bits$
Zhou *et al.* [14]	$2T_{pair} + 8T_{exp}$	$7T_{pair} + 2T_{exp}$	$4\|\mathbb{G}_1\| + 2\|\mathbb{G}_2\| = 4128\ bits$
Tseng *et al.* [15]	$2T_{hash} + 5T_{mul} + 2T_{pair} + T_{exp}$	$2T_{hash} + 2T_{mul} + 2T_{pair} + T_{exp}$	$2\|\mathbb{Z}_p^*\| + 3\|\mathbb{G}_1\| + \|\mathbb{G}_2\| = 2904\ bits$
Mandal *et al.* [16]	$12T_{hash} + 11T_{mul}$	$9T_{hash} + 3T_{mul}$	$n + 3\|\mathbb{Z}_p^*\| + 4\|\mathbb{G}_1\| = 2720\ bits$
Xu *et al.* [17]	$3T_{hash} + 3T_{mul} + 3T_{pair}$	$3T_{hash} + 2T_{mul} + T_{pair}$	$n + \|\mathbb{Z}_p^*\| + 2\|\mathbb{G}_1\| + \|\mathbb{G}_2\| = 2384\ bits$
Network3	$T_{hash} + T_{mul}$	$T_{hash} + T_{mul} + 3T_{pair} + T_{exp}$	$n + 3\|\mathbb{G}_1\| = 1720\ bits$

* *$T_{hash}, T_{mul}, T_{pair}, and\ T_{exp}$ denote the running time of a hashing operation, a point multiplication operation in $\mathbb{G}_1$, a bilinear pairing operation in $\mathbb{G}_1$, and an exponentiation operation in $\mathbb{G}_2$, respectively. $n, |\mathbb{Z}_p^*|, |\mathbb{G}_1|, and\ |\mathbb{G}_2|$ represent the length of the transmitted message, the length of the element in $\mathbb{Z}_p^*$, the length of the element in $\mathbb{G}_1$, and the length of the element in $\mathbb{G}_2$, respectively.*
* *Note that testers can employ the **MIRACL** library to measure the execution time of each invoked cryptographic operation. This allows them to accurately determine computational overheads, thereby facilitating in-depth comparative analysis and further optimization of the scheme.*
* *To calculate the ciphertext size in this table, we set $n = 160\ bits, |\mathbb{Z}_p^*| = 160\ bits, |\mathbb{G}_1| = 520\ bits, |\mathbb{G}_2| = 1024\ bits$.*

TABLE II
RUNNING TIME OF CRYPTOGRAPHIC OPERATIONS

Symbol/Operation	Running Time
T_{hash}	0.05 ms
T_{mul}	0.45 ms
T_{exp}	0.45 ms
T_{pair}	4.38 ms

B. Rating Submission

Let $Rater_k$ $(k \in [1, m])$ represent the k-th stakeholder in the verification committee, each equipped with a pair of public/private keys $(pubK_k, priK_k)$. Upon receiving the data evidence, $Rater_i$ $(i \in [1, m])$ reviews it and generates an integer $f_i \in [-10, 10]$ based on their evaluation. A high rating indicates superior data quality and vice versa.

If the rating scores are made public within the blockchain network, malicious adversaries may link them to a specific stakeholder through analysis attacks, leading to potential privacy breaches and retaliations. Furthermore, raters might always provide positive ratings to maintain a good reputation, rendering the correctness verification mechanism ineffective. Therefore, protecting rating privacy to encourage honest rating is of great significance. We employ a symmetric homomorphic encryption algorithm proposed in [18], to ensure both rating confidentiality and computational efficiency. Each $Rater_i$ generates a random secret $K_i \in \mathbb{Z}_p^*$. For instance, the encryption of a private feedback f_i is defined as:

$$c_i = E_{K_i}(f_i) = [(f_i + K_i) \times MK] \bmod p, \tag{5}$$

where $E_K(f)$ represents encrypting f with K, parameter p is a prime, and MK is a master key. The decryption of c_i is defined as:

$$f_i = D_{K_i}(c_i) = [c_i \times MK^{-1} - K_i] \bmod p, \tag{6}$$

where $D_K(c)$ represents decrypting c with K and MK^{-1} is the multiplicative inverse of MK modulo p. Assuming c_1 and c_2 are the ciphertexts of data f_1 and f_2 under the secret keys k_1 and k_2, respectively. According to (5), the homomorphic property supports direct computation on the ciphertexts as the following equation:

$$\begin{aligned} c_1 + c_2 &= E_{k_1}(f_1) + E_{k_2}(f_2) \\ &= [(f_1 + f_2) + (k_1 + k_2)] \times MK \bmod p \\ &= E_{(k_1+k_2)}(f_1 + f_2). \end{aligned} \tag{7}$$

Naturally, we can obtain $(f_1 + f_2)$ by:

$$f_1 + f_2 = D_{(k_1+k_2)}(c_1 + c_2). \tag{8}$$

Specifically, each $Rater_i$ encrypts f_i with K_i and uploads their ciphertext c_i to the mainstream blockchain mainnets.

C. Key Shares Dissemination

SC_{CV} can simply gather $\sum_{i=1}^{m} c_i$ from the blockchain network. However, SC_{CV} needs the value of $\sum_{i=1}^{m} f_i$ to calculate the average rating of the data evidence. According to (8), we have:

$$\sum_{i=1}^{m} f_i = D_{\sum_{i=1}^{m} K_i}(\sum_{i=1}^{m} c_i) \tag{9}$$

As demonstrated in (9), to compute the average rating, SC_{CV} needs to obtain $\sum_{i=1}^{m} K_i$. However, it is evidently impractical and insecure for raters to disclose their secret keys to the public. To solve this problem, we employ *Additive Secret Sharing* (ASS) to store the secret keys privately in the verification committee by dividing each secret key $K_i, i \in [1, m]$ into x shares. Despite being widely praised for its simplicity and efficiency, which outperforms other secret sharing schemes like *Shamir Secret Sharing* [19], ASS is deficient in robustness, as the loss or destruction of some shares may lead to a failure in reconstructing the secret. Therefore, we integrate $(x, m)-$*Reed-Solomon Codes* [20], a powerful error-correcting coding technique, to enhance the fault tolerance, scalability, and security of ASS. Assuming that $Rater_i$ splits the secret into x $(x < m)$ pieces, the design details proceed as follows.

a) Key Splitting and Encoding: $Rater_i$ selects a random value s_h^i as the h-th secret share of K_i, where h belongs to $[1, x)$. Then, we can obtain the last secret share by:

$$s_x^i = K_i - \sum_{h=1}^{x-1} s_h^i. \tag{10}$$

Therefore, $Rater_i$ gets x secret shares. Afterward, $Rater_i$ encodes these secret shares through *Reed-Solomon Codes* and generates m encoded shares, denoted as e_l^i ($l \in [1, m]$). Note that $m > x$, meaning that we introduce redundant information to deal with the situation of share damage or loss.

b) Encryption and Distribution: In order to secure the encoded shares from potential eavesdropping by other entities within the blockchain network or external attackers, each $Rater_i$ encrypts share using the public key of the intended rater. In Network3, we employ the *Paillier encryption* scheme [21], an asymmetric homomorphic encryption algorithm. For example, e_l^i is encrypted using the public key of $Rater_l$ ($l \in [1, m]$), with the result representing by $\mathcal{E}_{pubK_l}(e_l^i)$. That is, $Rater_i$ generates $\{\mathcal{E}_{pubK_1}(e_1^i), \mathcal{E}_{pubK_2}(e_2^i), ..., \mathcal{E}_{pubK_m}(e_m^i)\}$ for K_i and transmits them to the corresponding $Rater_l$ through the mainstream blockchain network. It is noteworthy that each rater will receive a total of m encrypted encoded shares from the other raters along with his own part, and the dissemination of one secret only needs to conducted once.

D. Key Reconstruction

Leveraging the additive homomorphic property of *Paillier encryption*, the encrypted shares corresponding to $Rater_l$ can be multiplied together, which is equal to the encryption result of the summed encoded shares, as shown below:

$$\prod_{i=1}^{m} \mathcal{E}_{pubK_l}(e_l^i) = \mathcal{E}_{pubK_l}(\sum_{i=1}^{m} e_l^i). \tag{11}$$

Therefore, $Rater_l$ can obtain $\sum_{i=1}^{m} e_l^i$ through an asymmetric decryption operation using $priK_l$.

Subsequently, each rater independently decrypts the product of the encrypted encoded shares $\mathcal{E}_{pubK_l}(\sum_{i=1}^{m} e_l^i)$ using their secret key $priK_l$ and forwards the result to SC_{CV}. Hence, SC_{CV} obtains $\{\sum_{i=1}^{m} e_1^i, \sum_{i=1}^{m} e_2^i, ..., \sum_{i=1}^{m} e_x^i\}$. Then, SC_{CV} performs the *Reed-Solomon* decoding operation to obtain $\sum_{i=1}^{m} K_i$.

The shares dissemination and secret sum reconstruction details are illustrated in Table. III.

E. Average Rating Calculation

Afterwards, V_i calculates $\sum_{i=1}^{m} E_{K_i}(f_i)$ using the ciphertexts gathered from blockchain. Then, it decrypts the calculated sums using $\sum_{i=1}^{m} K_i$ to obtain $\sum_{i=1}^{m} f_i$ according to (9). Finally, the average rating result f^{aver} can be computed through $f^{aver} = \frac{\sum_{i=1}^{m} f_i}{m}$. The process of key sum reconstruction and average rating computation is illustrated in Fig. 3. It is important to note that the real identities of the feedback providers remain confidential throughout the entire data correctness verification process.

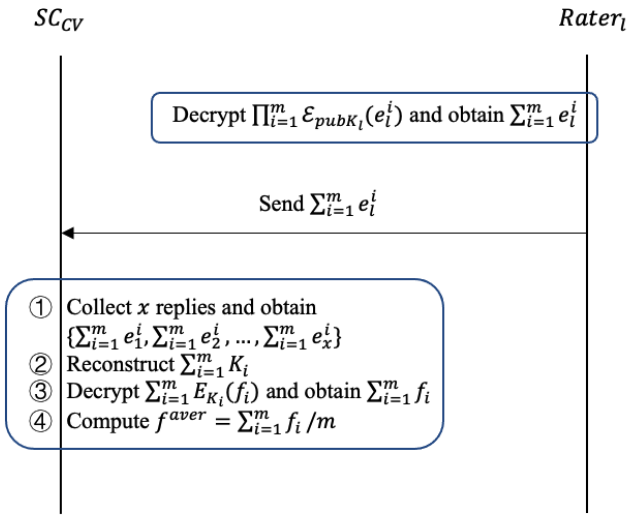

Fig. 3. Process of key reconstruction and average rating calculation.

F. Punishment and Incentive

If the calculated f^{aver} is less than 0, SC_{CV} publishes a symbol † along with the data evidence on mainstream blockchain mainnets, indicating that the transmitted data is incorrect. Then, the deposit of the data sender will be forfeited at a penalty rate $\eta_S \in [0, 1]$, which are partly distributed to the data receiver as compensation and to the raters as verification rewards. If the calculated f^{aver} lies within the range $[0, 5]$, SC_{CV} publishes a symbol ♢, implying that the received data are valid and correct. Otherwise, if the calculated f^{aver} falls between $(5, 10]$, SC_{CV} publishes a symbol ‡, which suggests that the received data are indisputably correct and the data receiver is likely to have made a false complaint. In this case, the deposit of the data sender will be confiscated at a penalty rate $\eta_R \in [0, 1]$, which are partly transferred to the data sender as compensation and to the raters as verification rewards.

In addition, it is important to note that the deposit amount, penalty rates (i.e., η_S, η_R), and incentive rewards are closely tied to the tokenomic model of Network3. These elements are intricately designed to ensure balanced participation, promote honesty, and foster an environment conducive for growth and sustained operation of the Network3 ecosystem.

G. Key Shares Update

To safeguard against privacy breaches and security threats, $Rater_i$ should periodically update the secret key K_i ($i \in [1, m]$). However, $Rater_i$ would need to re-encrypt the rating value each time a new key is generated, which would incur substantial computation and communication overhead. As an alternative, the key shares of K_i can be updated periodically instead of generating a new secret key. $Rater_i$ selects a new value s_h^i for each $h \in [1, x)$ and derives a new s_x^i according to (10). Subsequently, $Rater_i$ generates m new encoded shares utilizing (x, m) − *Reed-Solomon Codes* and distributes them to the corresponding $Rater_l$ ($l \in [1, m]$). This key update approach enhances security while minimizing overhead.

TABLE III
SHARES DISSEMINATION AND SECRETS RECONSTRUCTION

	Raters	**Key Shares**	**Encoded Shares**	$Rater_1$	$\cdots$	$Rater_m$
Shares Dissemination	$Rater_1$	$K_1 = s_1^1 + s_2^1 + \cdots + s_x^1$	$e_l^1, l \in [1, m]$	$\mathcal{E}_{pubK_1}(e_1^1)$	$\cdots$	$\mathcal{E}_{pubK_m}(e_m^1)$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$Rater_i$	$K_i = s_1^i + s_2^i + \cdots + s_x^i$	$e_l^i, l \in [1, m]$	$\mathcal{E}_{pubK_1}(e_1^i)$	$\cdots$	$\mathcal{E}_{pubK_m}(e_m^i)$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$Rater_m$	$K_m = s_1^m + s_2^m + \cdots + s_x^m$	$e_l^m, l \in [1, m]$	$\mathcal{E}_{pubK_1}(e_1^m)$	$\cdots$	$\mathcal{E}_{pubK_m}(e_m^m)$
Secrets Reconstruction	——	Based on *ASS* and *Reed-Solomon Codes* $\Downarrow$ $\sum_{i=1}^{m} K_i = \sum_{i=1}^{m} \sum_{h=1}^{x} s_h^i = \sum_{l=1}^{x} \sum_{i=1}^{m} e_l^i$		Based on the homomorphic property $\Downarrow$ $\prod_{i=1}^{m} \mathcal{E}_{pubK_1}(e_1^i) = \mathcal{E}_{pubK_1}(\sum_{i=1}^{m} e_1^i)$	$\cdots$	Based on the homomorphic property $\Downarrow$ $\prod_{i=1}^{m} \mathcal{E}_{pubK_m}(e_m^i) = \mathcal{E}_{pubK_m}(\sum_{i=1}^{m} e_m^i)$

IV. ANONYMOUS COMMUNICATION MECHANISM

With the proliferation and advancement of network technology, there is a growing emphasis on network security. Researchers have devised numerous solutions in response to a range of security issues. While the technology for encrypting and safeguarding communication content has matured, securing the identities of communicating parties remains a challenge. Critical sectors such as electronic voting, electronic banking, e-commerce, and electronic auctions require the confidentiality of user identities. However, existing network protocols, including HTTP and TCP/IP, are open systems. Eavesdroppers can easily extract crucial information from intercepted data packets, such as IP addresses of the parties, message length, packet exchange timing, and frequency. This enables eavesdroppers to deduce user identities, and in conjunction with other data, glean valuable insights without accessing communication content.

In particular, contemporary cryptography conceals communication content effectively through encryption. However, it struggles to hide the location information and communication patterns of the sender and receiver. In other words, cryptographic techniques are insufficient in preventing inference of sender and receiver details via traffic analysis. Attackers can leverage source and destination addresses, message lengths, and other data, combined with eavesdropping results and traffic analysis, to discern additional information, including address details and communication patterns. These insights can be partly inferred and derived from data packet control information and characteristics. Consequently, when specific network nodes or data transmitters and receivers handle substantial information volumes, adversaries detecting such activity, even without knowledge of the content, can identify active nodes. This turns them into lucrative targets for potential attacks.

Anonymous communication aims to conceal communication relationships within business flows without altering existing network protocols. This ensures that eavesdroppers cannot directly discern or deduce the communication links between parties involved. This study initiates by examining Tor, which is a globally well-known anonymous communication system, and subsequently devises an enhanced anonymous communication mechanism.

A. The Tor System

The Tor anonymous communication system, often referred to as the second-generation onion routing system [22], operates via a network of onion routers (OR), also known as Tor nodes. These onion routers facilitate the transfer of data from the source to the destination. The primary objective of each onion router is to ensure that external observers cannot correlate input and output data. In other words, it is designed to make it impossible for an external entity to infer the corresponding output data packet based on its input counterpart. This feature effectively prevents attackers from conducting communication flow analysis.

In the Tor anonymous communication system, users seeking anonymity initiate the Onion Proxy (OP) program. The OP is responsible for establishing communication links and encrypting/decrypting data. It begins by retrieving information about all Tor nodes from the directory server. Subsequently, it randomly selects a node from the pool of Tor nodes and engages in a key negotiation process. This negotiation process employs the short-term Diffie-Hellman key exchange protocol, and the TLS protocol is further employed to enhance the confidentiality of the channel and the forward security of the information. All subsequent data transmissions occur through this established channel.

Following this, the OP proceeds to expand connections to other Tor nodes through the channel, exchanges keys with them, and establishes a multi-layer encryption channel. Before transmission, the data is encrypted by the OP in reverse order as it passes through the Tor nodes, akin to the layers of an onion. During the transmission process, the encrypted data is decrypted at each Tor node, one layer at a time, until it reaches the final node. At this point, the data is fully decrypted and forwarded to the destination. Upon the return journey from the destination, the data is encrypted at each Tor node and subsequently decrypted in layers, ultimately being delivered to the user-side application.

It is important to note that each Tor node is only privy to its own encryption and decryption keys. This means that external attackers and even colluding Tor members would be unable to access the plaintext of communication data unless they manage to obtain the keys of every Tor node within the communication path. Unlike traditional anonymous communication systems,

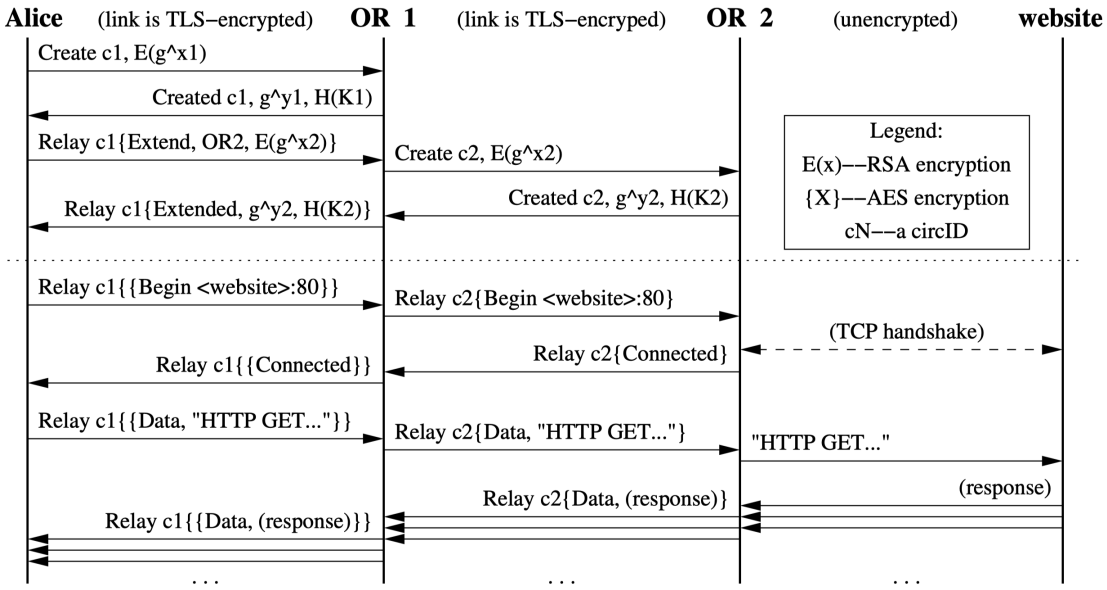

Fig. 4. Alice builds a two-hop circuit and begins fetching a web page. (cf. [22])

Tor does not employ precise mixing techniques, such as data batching. Instead, it stores data packets to be transmitted in the cache of their respective streams. Data packets are circulated among various streams in a round-robin fashion. This ensures equitable forwarding of all connected data. When the cache of a particular stream becomes empty, the system skips that connection and proceeds to forward data from the next available non-empty connection cache. Tor prioritizes low latency and refrains from employing traditional packet operations such as delaying, reordering, batching, or padding discarding.

Figure 4 provides a visual representation of Alice's process in establishing a two-hop circuit and initiating the retrieval of a web page through Tor. For a more comprehensive understanding of this process, interested readers are encouraged to consult the detailed information available in [22].

B. The Optimized Mechanism

Indeed, a fundamental weakness is discernible within the Tor system: the entry node becomes privy to the identity information of the anonymous communication initiator, while the exit node gains access to the identity information of the anonymous communication recipient. Consequently, the aspiration for complete anonymity remains unrealized. To address this predicament, we present an optimized anonymous communication system solution, guaranteeing the security of both the sender and receiver in the anonymous communication system without substantial increase in communication latency.

1) Crowds Anonymous System: Crowds [23] is a communication system that ensures sender anonymity by employing a unique concept - the idea of "blending into a crowd". In essence, it establishes a collective of numerous anonymous users, with the anonymous sender concealed among these group members. Each member within the group acts as a proxy. When a member intends to initiate anonymous communication, they send a request to other members. Upon receiving the request, the member can opt to either directly forward it to the destination node or relay it to other members. This approach provides protection for the sender while also safeguarding the anonymity of other members.

To illustrate the process, every host under anonymous protection runs a program known as the Jondo agent. Within this agent, they specify the host name and port number in their browser, designating Jondo as the HTTP proxy. Jondo then obtains the list of system members from the system management node known as the "blender". When a browser request is initiated, it is first directed to the local Jondo. This local Jondo performs a coin toss to determine whether to send the request to the destination or forward it to the next Crowds node. If it is sent to the destination, a connection is established accordingly. In cases where it is forwarded to the next node, the local Jondo randomly selects a Crowds member from the list, establishes a connection with them, and forwards the request. The process is repeated with each subsequent Crowds node. This series of coin tosses continues until the request ultimately reaches its destination.

In the Crowds anonymous communication system, connection establishment eschews traditional public key encryption, instead utilizing the symmetric key unique to each Jondo node. These symmetric keys are acquired from the "blender server". Owing to the absence of public key operations, the Crowds system boasts notably low communication latency. However, its security level does not match that of public key encryption-based anonymous communication systems. This system is primarily designed for web browsing. The Crowds system improves its anonymity as more users join the network. The author has demonstrated that as the number of users, denoted as n, approaches infinity, the Crowds system can achieve near-perfect anonymity regardless of the presence of

local eavesdroppers, Crowds conspiracy attackers, or target-end attackers.

2) Multicast Mechanism: Multicast communication plays a pivotal role in facilitating one-to-many communication among hosts over the Internet. It distinguishes itself from unicast communication by addressing a group of hosts as opposed to a singular, specific host. By harnessing broadcast or multicast technology for data transmission, the recipient can effectively blend into the multicast or broadcast group, thereby ensuring the anonymity of the receiver.

Here's how it works: When there is a message to be dispatched to a host group, multicast simplifies the process. The sending host merely transmits the message, and all hosts belonging to the same group can receive it. Within the multicast paradigm, each host group comprises a collection of zero or more hosts, each identified by a unique Class D interface address. To facilitate this, hosts employ the Internet Group Management Protocol (IGMP) to notify their local router of their intent to join and receive data from a particular multicast group. Routers also periodically query the LAN to ascertain the activity status of known group members through the IGMP protocol.

Multicast routing mechanisms predominantly employ a distribution tree structure. This entails the establishment of a distribution tree connecting routers, linking the sending host with all intended receiving hosts. Multicast messages traverse this distribution tree, creating duplicates at the branch nodes. Notably, within the multicast paradigm, individual real nodes remain oblivious to the specifics of multicast group members. Consequently, it becomes exceedingly challenging for attackers to gain access to information regarding all routers within the multicast tree. This intrinsic feature bolsters the security of multicast communication. Note that more nodes in a multicast group would lead to stronger anonymity. The number of nodes included in a group is a customized parameter instead of a fixed system integer.

3) Proposed Mechanism: In this system, when a user wishes to send a message anonymously, they initiate a series of steps. Firstly, the user launches the OP program and selects a Tor node. Subsequently, the user activates their personal Jondo Agent program and employs a data packet forwarding strategy akin to that of Crowds, thereby achieving sender anonymity. The core structure of the forwarding network still leverages Crowds, although it avoids the adoption of a fully random routing selection strategy. The randomness in Crowds path selection could lead to excessively long paths, resulting in undesirable communication delays. To mitigate this, the system imposes a constraint on the maximum path length.

The primary objective of the forwarding network is to mask the sender's identity, thereby concealing the genuine communication users amidst numerous anonymous user groups. This objective is congruent with the principles of Crowds. However, owing to the security limitations of Crowds, it was initially designed exclusively for web browsing. In Network3, the forwarding system harmonizes with Tor and encompasses all the network functionalities that Tor offers. The visual representation of this approach is depicted in Figure 5.

When anonymous users desire to initiate an anonymous connection, the sequence of actions unfolds as follows: the user launches the OP program, selects a Tor node, and initiates the Diffie-Hellman handshake protocol, as shown in Figure 4. Following this, the user initiates their personal Jondo Agent program, resulting in the establishment of a path. The path length is randomized but constrained by a specified maximum path length, while the member selection strategy adheres to randomness.

The path selection process in the Jondo program incorporates two crucial parameters: the forwarding probability denoted as P_f and the maximum path length N. P_f serves as the threshold value used to decide whether to forward data or submit it. N, on the other hand, dictates the maximum path length and is a random integer generated by the sender's Jondo program, varying between 3 and 8. Once P_f and N are determined, the user initiates a coin toss, generating a random probability value between 0 and 1. If this value is less than P_f, a Jondo is randomly selected as the next hop node, and the maximum path length N decreases by 1. The data is then transmitted to the selected next hop node, which checks if the received N value is greater than 0. If N is greater than 0, the coin tossing process continues. If N equals 0, the routing process concludes, with this node becoming the final node in the forwarding network. During the coin tossing process, if the random value exceeds P_f, irrespective of N, the routing process ends, and the data is submitted to the relay network.

This process continues until the exit node of the relay link dispatches the data packet to the multicast group where the intended recipient is located. The data is disseminated in multicast format, signifying that each group member receives the data packet. However, only the recipient in possession of the decryption key can open it, thereby ensuring receiver anonymity.

The introduction of the maximum path length during the path establishment process serves to circumvent excessively long paths, which could lead to unwarranted communication delays. Randomization plays a pivotal role in preserving system security by preventing the forwarding node from discerning whether it functions as the initial forwarding node. Once the link within the forwarding network is established, it assumes the sole responsibility of data forwarding. In contrast, relay network manages network congestion control, data integrity verification, and forward security, among other functions.

The proposed structure is illustrated in Figure 5, with S symbolizing the sender and R representing the receiver. Sender S_1 encrypts the data with the public key of receiver R_1 and then proceeds to distribute it randomly within the group to which S_1 belongs. Once the data reaches S_4, it is handed off to the relay network, making its way through the network until it reaches the link's end. The exit node subsequently transmits the data packet to the multicast group containing receiver R_1. All users within this multicast group receive the packet, but only R_1, who possesses the private key to decrypt it, can access its contents. All other users can merely discard the encrypted packet, thereby ensuring the privacy of the communication.

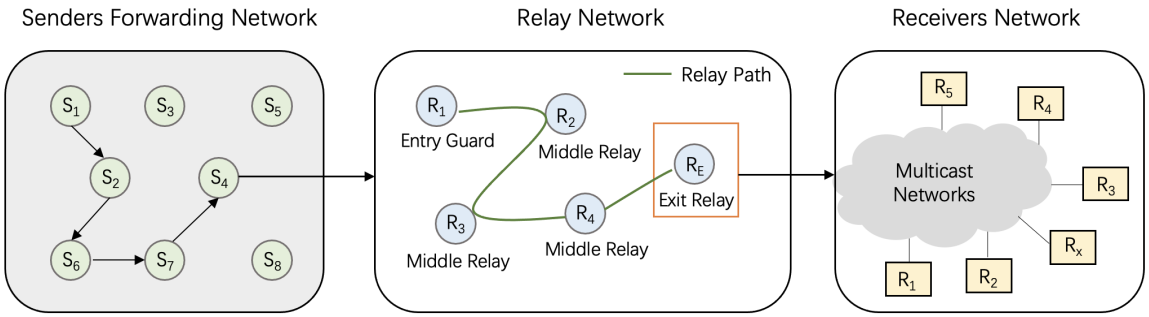

Fig. 5. Network3 Anonymous Communication Mechanism

C. Security Analysis

This system inherits all the robustness against attacks that the Tor anonymous communication system offers. It combines the advantages of Tor and introduces two added layers of security due to its hybrid nature. In the context of attacks, this system is more resilient because it is challenging to conduct a complete communication flow analysis by focusing solely on Tor or the forwarding network. The hybrid system presents a higher level of security. When an attacker targets relay or the forwarding network, they can only infer limited information; for instance, an attack on Tor may reveal the sender's identity, but the receiver's anonymity remains intact. Similarly, attacking the forwarding network solely reveals the sender's identity but doesn't compromise the anonymity of the receiver. As long as the blender node's security is maintained, the user's selection of forwarding users remains entirely random, ensuring the entire communication's security. As shown in literature [23]–[25], this system is proven to be secure, guaranteeing the overall anonymity of the communication.

- **Low Latency:** The implementation of the forwarding network is efficient and straightforward. Data link establishment and data forwarding occur at high speed because they use symmetric keys. The primary factors affecting latency are network bandwidth and congestion. The relay network employs congestion control algorithms and token bucket strategies to ensure network robustness, making the hybrid system achieve very low communication delays.
- **Data Security:** In Crowds, any forwarding node can access plain text communication, leaving data security uncertain. In the proposed system, all data in the forwarding network are encrypted by the OP, ensuring data security. It also enables data integrity verification and forward security.
- **Local Eavesdropping:** Local eavesdroppers can only observe communications of local users. However, the anonymity of the receiver is maintained because data is disseminated in multicast mode. Local eavesdroppers cannot access any information about the receiver.
- **Multicast Group Eavesdropping:** Eavesdroppers within the multicast group may receive the data packet, but it remains encrypted. Only the recipient with the decryption key can access the plain text data. Moreover, the sender's anonymity is preserved since data packets are forwarded within Crowds, making it difficult for eavesdroppers to ascertain the sender's information.

V. Edge Computing-driven Outsourced Computation

To address the challenges stemming from the inherent limitations of terminal equipment (such as mobile phones, wearable devices, .etc), characterized by finite computational resources and incapacity to execute intricate operations efficiently, the integration of a distributed edge computing framework emerges as a sophisticated and technically potent solution. The implementation of this framework strategically aims to mitigate user computing thresholds, abate protocol execution delays, enhance operational efficiency, and fortify architectural scalability. This section elucidates the technical intricacies and principles underlying this innovation, underscoring its pertinence in the context of modern distributed systems.

At the core of our approach is the selective offloading of computationally intensive tasks, notably the signcryption, signature verification, key encryption, and other computation-intensive processes, to edge servers. These edge servers are judiciously chosen due to their substantial computational prowess. By relocating these high-resource-demanding tasks, our framework ameliorates the computational burden on the terminal devices, which, given their constrained resources, would otherwise grapple with the onerous demands of these operations. This offloading paradigm substantially ameliorates the overall computational costs associated with terminal devices, contributing significantly to the optimization of operational performance.

The edge server assumes a pivotal role within this architecture. It takes on the responsibility of executing computationally intensive operations offloaded from the terminal devices. This strategic distribution of computational tasks across edge servers not only guarantees the efficient execution of these operations but also elevates the architecture's robustness. The crux of this approach hinges on the notion of outsourced computation, a concept characterized by the seamless delegation of specific tasks to edge servers.

Outsourced computation is not merely a delegation of tasks; rather, it constitutes a nuanced process characterized by

secure computation, data confidentiality, and integrity. When a terminal device or user necessitates the execution of computationally demanding operations, the system orchestrates the allocation of these tasks to specific edge servers selected based on their computational prowess. The process adheres to stringent security protocols to ensure the confidentiality and integrity of the data, even in the midst of computations performed on edge servers.

The synergistic interplay between edge computing and outsourced computation is the hallmark of our innovative framework. This symbiotic relationship is predicated on the judicious utilization of computational resources, stringent security protocols, and the meticulous orchestration of task offloading. It not only surmounts the computational constraints posed by terminal equipment but also fosters an architecture that is inherently resilient, scalable, and primed for optimum performance.

VI. Decentralized Federated Learning Framework

We introduce a novel decentralized anonymous transmission protocol in the preceding section, emphasizing its significance within the context of our broader exploration of AI-enabled intelligent computing. This facet is integral to the envisioned Web3 paradigm, which, in its truest form, champions openness, decentralization, self-motivation, and diversification. Presently, AI advancements are predominantly governed by industry giants like Google and OpenAI, relegating the majority of users to passive roles. This misalignment with the principles of Web3 or Depin prompts us to address this issue head-on. Consequently, in the ensuing sections, we present a pioneering decentralized federated learning framework. This framework is designed not only to rectify the current landscape but also to infuse greater intelligence into the evolving DePin ecosystem.

A. Background and Motivation

With the swift advancement of intelligent applications fueled by extensive data, there has been a substantial enhancement in the efficiency of various aspects of human life and production. However, this progress has given rise to heightened discussions and apprehensions concerning data security and privacy. Particularly, in scenarios where data producers are dispersed across heterogeneous networks, the effective and judicious management of this distributed data has emerged as a pressing challenge in the realm of artificial intelligence. This is underscored by the dual imperative faced by big data service providers: the need to deliver efficient intelligent services while respecting user privacy concerns. Google addressed this conundrum by pioneering the federated learning framework (FL) in 2016 [26], [27].

FL, a distributed machine learning paradigm, offers a solution to the challenge of data islands in the big data landscape. It enables the preservation of data privacy by facilitating collaborative data sharing and machine learning modeling across multiple terminals. In the FL framework, individual distributed devices conduct local model training without the necessity of uploading private data to a central server. Instead, they transmit concise model update information. The central server then consolidates this update information, orchestrates the optimization of a global model, and subsequently redistributes the optimized model to each device for the subsequent round of training [28]. This technical breakthrough effectively dismantles data islands, enabling the proficient sharing and utilization of vast user data. Federated learning has thus garnered considerable attention from both academia and industry.

While federated learning (FL) introduced a groundbreaking privacy-preserving approach to distributed model training in the AI community, the current FL framework confronts a spectrum of formidable challenges.

- Firstly, the presence of a centralized model aggregator in traditional FL frameworks poses considerable security risks, including susceptibility to single-point attacks and privacy breaches. This renders it unsuitable for large-scale decentralized environments like DePin.
- Secondly, the vulnerability to malicious attackers, who can masquerade as local trainers or global aggregators, raises the specter of compromised model accuracy through the uploading of fake model updates.
- Thirdly, the diverse roles within the FL framework, encompassing local trainers, verifiers, and aggregators, lack an effective and customized incentive mechanism. This deficiency diminishes the willingness of participants to contribute their computing power and network bandwidth to the iterative processes.
- Lastly, the large size of model updates/parameters and model dissemination results in substantial communication costs and synchronization latency.

These challenges have assumed heightened significance as FL is increasingly considered the most promising technology for establishing a decentralized machine learning architecture. This architecture not only facilitates privacy-preserving model training using the heterogeneous computing power of myriad intelligent devices but also addresses the challenges posed by the burgeoning applications of large language models, such as ChatGPT.

In response to these challenges, there have been noteworthy technological solutions. Decentralized approaches to model aggregation, where the responsibility for aggregation is distributed across multiple entities rather than relying on a single centralized aggregator, have been explored to mitigate the risk of single-point attacks [29]–[31]. Cryptographic techniques, such as secure multi-party computation (SMPC) and homomorphic encryption, are being employed to enhance the security of the model aggregation process and protect against privacy breaches. Robust mechanisms for detecting and mitigating the impact of fake model updates have been proposed, leveraging anomaly detection and reputation systems.

However, despite these advancements, there remain gaps and limitations. Ensuring the reliability, scalability and efficiency of decentralized model aggregation methods is an ongoing area of research. Robustness against sophisticated adversarial attacks and the development of adaptive incentive mechanisms that account for the dynamic nature of FL frameworks are active areas of exploration. Moreover, addressing the communication costs and synchronization latency associated

with unpredictable model updates is critical for the continued success and adoption of federated learning.

In our project, we present a groundbreaking decentralized FL framework designed to effectively tackle the aforementioned challenges. In essence, our solution incorporates several key innovations. Firstly, we integrate layer-2 blockchain technology to decentralize the FL processes, mitigating security risks associated with centralized models. Additionally, we introduce an asynchronous local parameter verification mechanism to enhance the robustness of local computations. To further bolster the integrity of the model, a global model verification mechanism is implemented. Our framework also introduces a proof-of-contribution consensus, providing a much more efficient foundation for FL operations. Building upon this consensus, we devise an adaptive incentive mechanism tailored to engage and motivate all participant types in the FL ecosystem. This multifaceted approach aims to not only overcome the limitations of traditional FL frameworks but also establish a more secure, efficient, and participatory paradigm for decentralized machine learning.

B. Overall Framework and Processes

1) Notation Description: Network3 employs a specified number of iteration cycles to update the FL global model. Each cycle consists of five key components: model training, local parameter verification, aggregation, global parameter verification, and block generation. The iteration cycle set is represented by $R = \{1, 2, \ldots, j\}$ $(j \in N_+)$, where $|R|$ serves as a hyperparameter indicating the maximum number of iteration cycles required for model convergence in FL.

The participating nodes are represented by the set $N = \{n_1, n_2, ..., n_i\}$ $(2 \leq i \leq k)$, where k denotes the total number of nodes in the system. At the commencement of the j-th round of iteration, all nodes are divided into four different roles based on their historical contributions ($|T| + |E| + |A| + |G| = |N|$). These roles include trainers $T = \{t_1, t_2, ..., t_x\}$ $(0 < x < k - 3)$ responsible for training local models, local validators $E = \{e_1, e_2, ..., e_y\}$ $(0 < y < k - 3)$ who assess the model parameters sent by trainers, aggregators $A = \{a_1, a_2, ...a_z\}$ $(0 < z < k - 3)$ who aggregate the legitimate parameters evaluated by E, and global validators $G = \{g_1, g_2, ..., g_w\}$ $(0 < w < k - 3)$ who verify the global parameters and generate new blocks.

In the jth iteration, Network3 coordinates all nodes to update the FL global model, denoted as M_j. In the subsequent iteration, M_j will serve as the local model for each node, utilized for model training and parameter evaluation to update M_{j+1}, and so on. The evaluator E needs to use the local dataset to evaluate the model parameters sent by the trainer T. In the j-th iteration, the evaluation result of e_y for t_x is represented by the voting value $v^j(y, x)$, which is sent to aggregators for final verification. For each trainer t_x, the aggregator needs to collect and aggregate all evaluation results $v^j(y, x)$ related to t_x. The evaluation results will ultimately determine whether the model update uploaded by t_x can pass the verification. In Network3, the blockchain saves the cumulative rewards obtained by each node, referred to as its contribution. The contribution of node n_i at the beginning of the j-th round is represented by $\mathbb{C}_i^j$. If node n_i is elected in this round as the evaluator, then its contribution $\mathbb{C}_i^j$ will be used as the voting weight in the verification period.

2) Framework Overview: As illustrated in Figure 6, the proposed framework involves five distinct types of participants. First and foremost, the task publisher, desiring to train an application-specific ML model, releases task descriptions onto blockchain. These descriptions encompass data requirements, convergence demands, rewards, and other pertinent details. Additionally, he/she provides the initial model. Smart contracts are pre-deployed to facilitate task selection, role assignment, and model distribution to trainers. The trainers T use their private dataset to train the local model and subsequently upload the trained model parameters to evaluaters. The local evaluator E are responsible for evaluating the model parameters uploaded by the trainers. The aggregators A play a crucial role in aggregating all legal parameters. The global validators G verify the validity of the global model and generate new blocks.

AI data annotation is a crucial step in the data preprocessing phase, as supervised machine learning models rely on annotated data to recognize recurring patterns. The algorithm, when exposed to a substantial amount of labeled data, becomes adept at identifying similar patterns in new, unlabeled data. Hence, data scientists leverage meticulously annotated data to effectively train machine learning models. It is important to highlight that, in the context of federated learning, the data annotation process is a one-time operation executed by the trainer on the local dataset during the initialization phase. This process stands apart from the iterative stages of federated learning training and inference. Consequently, we exclude the data annotation process from the architecture's considerations. By default, trainers locally complete the data annotation, utilizing the prepared dataset to train the model and acquire local parameters. Recognizing the labeling operations' cost for training nodes, we incorporate the dataset size as a primary factor when determining incentives for local trainers. A larger dataset implies a higher labeling cost for the default trainer, resulting in correspondingly higher incentives. This approach ensures a balanced consideration of the resource-intensive nature of data labeling operations in the incentivization model.

At the end of each round of global model iteration, the adaptive incentive algorithm will allocate corresponding rewards according to the role and work intensity of each node, and update their contribution. At the beginning of the next iteration, Network3 reassigns roles to each node based on its contribution to ensure the liquidity of nodes in the system. All selected nodes engage in model training, parameter evaluation, aggregation, and global validation and block generation tasks, ensuring the timely and positive updating of the global model.

C. Local Parameter Verification Mechanism

Upon receiving local parameters P_x^j from trainer t_x, local validator e_y first checks the integrity and authenticity by verifying the signature. Subsequently, e_y broadcasts P_x^j to other local validators. Then, e_y queries a hash table HT on the

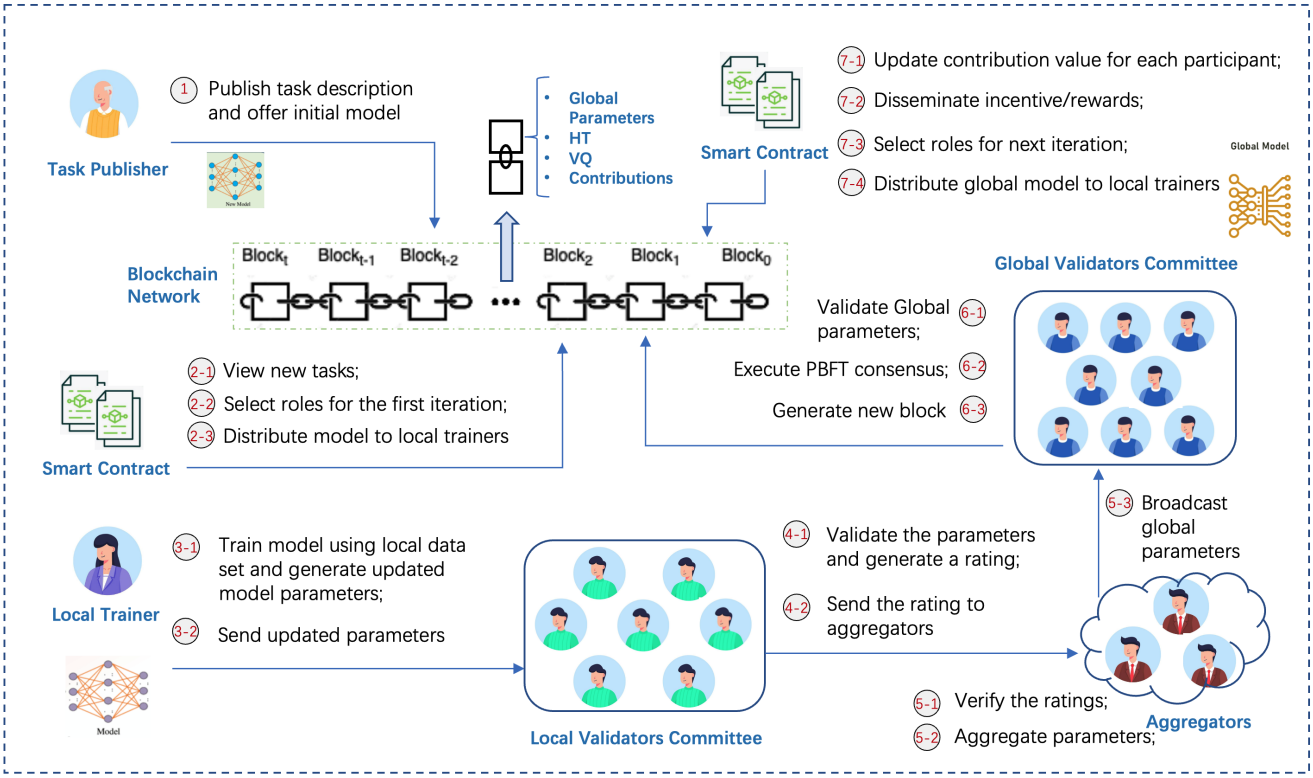

Fig. 6. Processes of the proposed decentralized FL framework

blockchain to confirm whether P_x^j has passed the verification. Concretely, the parameters are considered successfully verified if $HT(P_x^j) \geq 0.5$. If the verification is successful, e_y will not continue to process the parameters. However, if the verification fails, the local dataset is employed to perform the parameter evaluation process. In this case, the evaluator only needs to use the private dataset to train the local model for one epoch, simulating the reasonable decline expected from a legitimate trainer after using the local dataset to train the global model. Finally, the evaluator e_y will send the evaluation results to the aggregators in the form of a voting value $v^i(y, x)$.

Aggregators maintain a hash table HT that stores the local parameters uploaded by local trainers. If the current number of positive votes exceeds half the number of local validators, subsequent evaluation processes will be unnecessary. For example, for trainer t_x, the following verification results are equal in HT: $HT(P_x^j) = \{True, True\}$ and $HT(P_x^j) = \{True, True, False\}$. In order to improve the efficiency and robustness of local parameter verification, we design different voting weights for each evaluator based on their historical contribution. Through reliable records stored on the blockchain, the voting weight of evaluator e_y will be calculated by his/her contribution $\mathbb{C}_y^j$. This mechanism ensures that malicious nodes always maintain a lower voice in the verification period. The final verification result depends not only on the number of favorable votes but also on the historical contribution of the evaluator, further reducing the probability of successful attacks on local validators. Specifically, the aggregators use the following formula to determine whether the model parameters uploaded by the trainer can pass the legality verification:

$$HT(P_x^j) = \frac{exp(\mathbb{C}_y^j + \beta) \cdot v^j(y, x)}{\sum_{i=0}^{|E|} exp(\mathbb{C}_i^j + \beta)} + HT'(P_x^j) \tag{12}$$

We use a normalized exponential function to quantify the evaluater's recognition of the parameters uploaded by the trainer. The hash table $HT(P_x^j)$ stores the verification results of the parameters P_x^j uploaded by the trainer t_x. The aggregator updates $HT(P_x^j)$ with the votes $v^j(y, x)$ collected in this round and the contribution $\mathbb{C}_y^j$ of the evaluator to which the vote belongs. Formula 12 describes the specific update method, where $HT(P_x^j)$ and $HT'(P_x^j)$ represent the aggregator's verification results before and after updating the model parameter P_x^j respectively. β is a hyperparameter used to address the issue of variable memory overflow caused by excessive contribution in the middle and late stages of operation. After completing the update, the aggregator determines whether P_x^j passes the verification based on the updated $HT(P_x^j)$: if $HT(P_x^j) \geq 0.5$, the majority of legal evaluators have recognized the model update parameter P_x^j uploaded by trainer t_x in this round; otherwise, the aggregator refrains from using local parameters P_x^j to update the global model during this iteration.

D. Global Model Verification Mechanism

After aggregating local parameters into global model, the aggregators transmit the global model to the global validators

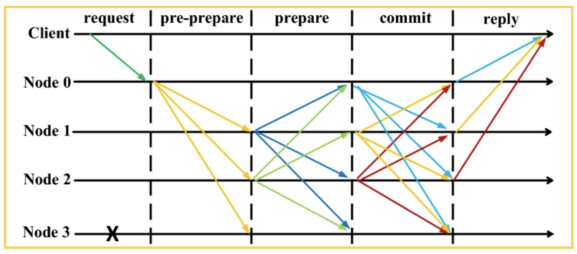

Fig. 7. PBFT consensus process

committee. After a certain number of global gradients are collected, the leader of the PBFT verification committee initiates verification of the global gradients in a certain order. Each global verifier votes yes or no for the global gradient based on the Krum algorithm [32]. and the first global gradient to receive more than 2/3 of the verifier's yes votes is selected as the global gradient for this iteration, and the validation process ends. The process of PBFT is illustrated in Figure 7. Finally, the aggregator whose global model is successfully chosen perform as block generator in this interaction. He/she will package the global parameter, the hash table HT, and the voting queue VQ into a block and broadcast in the blockchain network, ensuring that the global model M_j can be updated timely and legally in each iteration.

E. Adaptive Incentive Mechanism

We propose an adaptive incentive mechanism for each type of participants in the system based on their work intensity and contribution.

1) Rewards for Local Trainers: The trainer plays an important role in the proposed decentralized FL system, as the parameters from local training directly affect the prediction performance of the global model. Given the diversity in the quality of local data samples across nodes, updating the global model with low-quality or maliciously manipulated model parameters may cause a serious decrease in prediction accuracy. Therefore, the rewards assigned to trainers must be contingent upon both the legitimacy of the uploaded parameters and the intensity of training.

As shown in Formula 13, the expected reward obtained by the trainer t_x in the j-th iteration is calculated based on its dataset size and the number of training rounds. Following decentralized parameter verification, the verification results for the parameters P_x^j are permanently recorded on the blockchain. Therefore, whether the trainer t_x ultimately obtains its expected reward r_x^j depends on the legality verification result of its uploaded parameters $HT(P_x^j)$. In Formula 13, s_x represents the local dataset size of t_x, $epochs$ represents the number of training rounds of t_x, and the reward unit $\hat{r}$ is a hyperparameter.

$$r_x^j = \begin{cases} |s_x| \cdot |epochs| \cdot \hat{r}, & \text{if } HT(P_x^j) \geq 0.5, \\ 0, & \text{if } HT(P_x^j) < 0.5. \end{cases} \tag{13}$$

2) Rewards for Local Validators: Decentralized parameter verification is crucial for the reliable operation of Network3, ensuring the benign iteration of the global model within a certain cycle. Unlike trainers, local evaluators adopt a proxy assessment method to evaluate received model parameters. Evaluators only need to train a local model for one epoch using a private dataset to simulate the reasonable decline that a legitimate trainer would show after training the global model with local data [33]. Additionally, the reward r_y^j given to the validator e_y will be based on the time sequence of the node's votes in the verification period. This approach not only alleviates blockage issues in the aggregator's parameter verification process but also enhances the parallel efficiency of evaluators performing cross-assessments. VQ is a queue that stores the voting sequence of all local evaluators, maintained by the aggregator and recorded on the blockchain. Based on the index of the vote, $r_y(j)$ is calculated using Formula 14.

$$r_y^j = \sum_i^{|T|} \frac{|VQ| - VQ.FetchIndex(v^j(y,i))}{VQ} \cdot |s_y| \cdot \hat{r} \tag{14}$$

3) Rewards for Aggregators: Unlike traditional blockchain incentive mechanisms, which involve competition among miners to generate blocks, Network3 does not employ a competitive mechanism among miners. Instead, at the end of each iteration, the aggregator packages and uploads the intermediate results generated in that round to the blockchain. Network gives lower rewards to aggregators, partly because the aggregator's task is of low workload, not involving the training and evaluation of model parameters. On the other hand, providing lower rewards to aggregators contributes to maintaining liquidity among local validators. Considering that the computational and storage costs for aggregators mainly come from maintaining the parameter verification hash table HT and the evaluator voting queue VQ, their rewards r_a^j are calculated using Formula 15.

$$r_a^j = |HT| \cdot |VQ| \cdot \hat{r} \tag{15}$$

4) Rewards for Global Validators: Global validators play a crucial role in the block creation process by verifying the global models generated by aggregators and determining the final result by executing PBFT consensus. Therefore, they will get equitable rewards derived from the tokens generated during the creation of a new block in the blockchain network. The concrete amount relies on the tokenomic design and the number of validators in the PBFT committee, as depicted in Formula 16.

$$r_g^j = R/m, \tag{16}$$

where R denotes the corresponding value of the rewarded token and m represents the number of global validators.

F. Future Directions

1) Zero Knowledge Machine Learning: Zero-knowledge proofs, a branch of cryptography, have seen significant development in recent years, especially in the field of blockchain. In Layer1, zero-knowledge proofs can be employed for scalability and privacy protection. Layer2 can utilize zero-knowledge proofs to interact with Layer1, ensuring transaction security

while enabling off-chain computation to reduce the cost of complex calculations.

Zero-knowledge proofs [34] allow users to prove knowledge or possession of information without revealing the actual information. The Prover uses system-inputted information to create a proof, and the Verifier validates the computational result of the proof without gaining knowledge of the specific content of the information. Additionally, zero-knowledge proofs can verify the validity of a dataset while simultaneously protecting data privacy. These characteristics can be applied to safeguard the security and privacy of machine learning.

However, the computational cost of generating zero-knowledge proofs is extremely high. Despite the rapid development of hardware acceleration technologies for zero-knowledge proofs, it is still far from sufficient to meet the validation computation needs of the machine learning training process. Therefore, our priority is to first complete the simple validation and architecture design of distributed federated learning on the network3 platform. We then plan to incorporate zero-knowledge proofs into the model's inference process.The future challenge for zero-knowledge proofs in our system is to enable secure usage of models by individuals using machine learning without revealing the model's content. This is crucial for protecting data privacy when dealing with sensitive data. Current zero-knowledge proofs can achieve privacy-preserving solutions, including model privacy, data privacy, and transparent verification in privacy computation scenarios [35].

Security requirements for machine learning models include confidentiality, integrity, and availability [36]. Confidentiality ensures that unauthorized users cannot access sensitive information in the machine learning system, including both training data and information related to the model's architecture and parameters. Integrity demands that the model's predictions adhere to the expected outcomes, and availability requires the machine learning system to provide normal services even when faced with abnormal or malicious inputs.For model users, they expect the models provided by service providers to be as desired, the training results to be accurate, and their data privacy to be protected. For model providers, they wish to safeguard their models from being known by others in their entirety and to be protected against malicious attacks from users.

Zero-knowledge proofs can play a significant role in meeting the security and privacy requirements of machine learning. The use of zero-knowledge proofs in machine learning is commonly referred to as ZKML (Zero Knowledge Machine Learning). ZKML is essentially divided into zero-knowledge during the model training process and zero-knowledge during the model usage process [37]. In the realm of zero-knowledge during the model training process, Gensyn [38] has gained notable attention. Their primary focus is on designing a probability auditing system based on gradient descent. They employ model checkpoints to enable a distributed GPU network to provide training services for full-scale models. However, we believe that using zero-knowledge proofs to monitor the cost of model training is currently impractical. At least for now, the computational complexity and cost of zero-knowledge proofs fall far short of meeting the demands of real-world production environments. Therefore, our focus is centered on implementing a system for distributed federated learning using a straightforward verification process, while incorporating the use of zero-knowledge proofs in the model usage phase.

The implementation approaches for ZK proof systems include ZK-Circuit and ZKVM. ZK-Circuit functions by directly transforming a program into constraints and feeding them into the proof system. On the other hand, systems based on ZKVM execute programs using an Instruction Set Architecture (ISA), generating an execution trace in the process. This execution trace is then mapped into constraints and submitted to the proof system. ZK-Circuit exhibits notable performance advantages and is more concise, while ZKVM is more developer-friendly, as developers are not required to have an in-depth understanding of each constraint condition.

For ZK-Circuit, the Modulus Labs team recently published a paper titled "The Cost of Intelligence" [39], in which they conducted benchmark tests on existing zero-knowledge proof systems with various model scales. They compared six zero-knowledge proof systems: Groth16, Gemini, Winterfell, Halo2, Plonky2, and zkCNN [40]. Currently, proof generation using systems like Plonky2 on powerful AWS machines takes approximately 50 seconds for models with around 18 million parameters.The paper ultimately guides them to choose the GKR scheme as the optimal ZKML verifier. While they acknowledge the effective result verification achieved by the zkcnn scheme, they highlight that the communication overhead of GKR should not be overlooked, considering it is the only interactive scheme among the options. In their paper, Modulus appears to have optimized the commitment scheme significantly, greatly reducing communication costs. This suggests that the GKR scheme is a reasonable choice as the best solution for ZKML.

We will also investigate how to use zkcnn's scheme in future model result validators, but we will focus on completing layer2 which provides routing and arithmetic. existing ZKVM has reached the condition of satisfying simple production environments, and most of ZKVM achieves $\mathbb{O}(1)$ time complexity for result validation. for the zk validator we will choose the DelphinusLab's zkwasm [41]. zkwasm is based on ZKSNARK, is non-interactive, has negligible communication overhead, and has negligible verification complexity compared to the complexity of the initial function, and its cost of use is very low.

Furthermore, we have also taken note of the OP Stack Zero Knowledge Proof scheme [42], aimed at using zero-knowledge proofs to demonstrate the effectiveness of fraud proofs in Optimism, thereby shortening the withdrawal window to 7 days. This approach combines the strengths of Optimism and zero-knowledge; zero-knowledge proofs can reduce the challenge time in Optimism, while Optimism can save costs and overhead for zero-knowledge. The optimistic approach holds great promise in machine learning, especially for widely used machine learning applications like ChatGPT. Regular users can easily bear short-term risks and are not interested in understanding the details of the model. Moreover, compared to ZKML, opML [43] provides low-cost and efficient machine learning proofs. OpML has lower hardware requirements, and

it can even run large language models on a standard PC.

2) Model Compression and Pruning: Model compression and pruning have emerged as promising techniques to alleviate communication overheads in distributed federated learning, addressing the challenges posed by transmitting large model updates over communication channels. Model compression involves reducing the size of the model by employing techniques such as quantization, which reduces the precision of the model's parameters, and knowledge distillation, where a compact model is trained to mimic the behavior of a larger model. On the other hand, model pruning focuses on eliminating redundant or less influential parameters, leading to a sparser model. These techniques not only facilitate faster communication but also contribute to lower storage requirements, a critical consideration in federated learning scenarios with resource-constrained devices. Key technologies in this domain include designing efficient compression algorithms, exploring optimal pruning strategies, and investigating ways to balance model accuracy and size. Challenges revolve around finding the right trade-off between model compression and maintaining predictive performance, adapting these techniques to diverse model architectures, and ensuring compatibility with privacy-preserving federated learning frameworks. In addition, instead of horizontally partitioning the data across devices, model parallelism involves partitioning the model itself. Each device is responsible for a specific portion of the model, reducing the amount of information exchanged during updates.

Future research directions may explore the integration of adaptive compression methods, leveraging insights from transfer learning, and addressing the unique challenges associated with non-IID (non-Independently and Identically Distributed) data distributions in federated learning environments.

VII. Conclusion

In conclusion, Network3 presents a groundbreaking protocol stack tailored to address the intricate challenges that have surfaced on the mainnet. By skillfully integrating advanced technologies, including an efficient anonymous certificateless signcryption (CLSC) algorithm, a robust decentralized data correctness verification mechanism, IP anti-tracking measures, and a reliable decentralized federated learning framework, Network3 redefines the landscape of decentralized data transmission and intelligent computing.

The CLSC algorithm stands as a beacon of identity authentication and secure data sharing within an anonymous realm. Network3's data verification mechanism provides a potent and decentralized solution to the problem of data inaccuracies. Moreover, the proposed anonymous communication mechanism ensures a perfect anonymous pattern for Web3.0 participants. Furthermore, the decentralized FL framework addresses the primary challenges posed by existing FL technologies, providing a promising and practical architecture for intelligent, secure, decentralized computing.

This protocol not only augments the capabilities of the mainnet but also carves a path for the future of DePin ecosystems. It enhances intelligence, bolsters anonymity, and reinforces reliability, setting the stage for an environment conducive to further innovation and progress.

References

[1] T. Ma, "Cybersecurity and ethereum security vulnerabilities analysis," *Highlights in Science, Engineering and Technology*, vol. 34, pp. 375–381, 2023.

[2] A. Petcu, B. Pahontu, M. Frunzete, and D. A. Stoichescu, "A secure and decentralized authentication mechanism based on web 3.0 and ethereum blockchain technology," *Applied Sciences*, vol. 13, no. 4, p. 2231, 2023.

[3] N. F. Samreen and M. H. Alalfi, "An empirical study on the complexity, security and maintainability of ethereum-based decentralized applications (dapps)," *Blockchain: Research and Applications*, vol. 4, no. 2, p. 100120, 2023.

[4] D. P. Bauer, "Filecoin," in *Getting Started with Ethereum: A Step-by-Step Guide to Becoming a Blockchain Developer*. Springer, 2022, pp. 97–101.

[5] S. Williams, V. Diordiiev, L. Berman, and I. Uemlianin, "Arweave: A protocol for economically sustainable information permanence," *arweave. org, Tech. Rep*, 2019.

[6] D. Vorick and L. Champine, "Sia: Simple decentralized storage," *Retrieved May*, vol. 8, p. 2018, 2014.

[7] A. Zamyatin, N. Stifter, P. Schindler, E. Weippl, and W. J. Knottenbelt, "Flux: Revisiting near blocks for proof-of-work blockchains," *Cryptology ePrint Archive*, 2018.

[8] T. Min and W. Cai, "Portrait of decentralized application users: an overview based on large-scale ethereum data," *CCF Transactions on Pervasive Computing and Interaction*, vol. 4, no. 2, pp. 124–141, 2022.

[9] U. Jafar, M. J. Ab Aziz, Z. Shukur, and H. A. Hussain, "A cost-efficient and scalable framework for e-voting system based on ethereum blockchain," in *2022 International Conference on Cyber Resilience (ICCR)*. IEEE, 2022, pp. 1–6.

[10] A. Q. Md, S. S. S. Narayanan, H. Sabireen, A. K. Sivaraman, and K. F. Tee, "A novel approach to detect fraud in ethereum transactions using stacking," *Expert Systems*, p. e13255, 2023.

[11] Z. Chen and K. Omote, "Toward achieving anonymous nft trading," *IEEE Access*, vol. 10, pp. 130 166–130 176, 2022.

[12] T. Sato, M. Imamura, and K. Omote, "Threat analysis of poisoning attack against ethereum blockchain," in *Information Security Theory and Practice: 13th IFIP WG 11.2 International Conference, WISTP 2019, Paris, France, December 11–12, 2019, Proceedings 13*. Springer, 2020, pp. 139–154.

[13] H. Xiong, "Toward certificateless signcryption scheme without random oracles," *Cryptology ePrint Archive*, 2014.

[14] C. Zhou, G. Gao, and Z. Cui, "Certificateless signcryption in the standard model," *Wireless Personal Communications*, vol. 92, pp. 495–513, 2017.

[15] Y.-F. Tseng and C.-I. Fan, "Provably cca-secure anonymous multi-receiver certificateless authenticated encryption." *Journal of Information Science & Engineering*, vol. 34, no. 6, 2018.

[16] S. Mandal, B. Bera, A. K. Sutrala, A. K. Das, K.-K. R. Choo, and Y. Park, "Certificateless-signcryption-based three-factor user access control scheme for iot environment," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3184–3197, 2020.

[17] G. Xu, J. Dong, and C. Ma, "A certificateless encryption scheme based on blockchain," *Peer-to-Peer Networking and Applications*, vol. 14, no. 5, pp. 2952–2960, 2021.

[18] C. Castelluccia, A. C. Chan, E. Mykletun, and G. Tsudik, "Efficient and provably secure aggregation of encrypted data in wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 3, pp. 1–36, 2009.

[19] M. Muhil, U. H. Krishna, R. K. Kumar, and E. M. Anita, "Securing multi-cloud using secret sharing algorithm," *Procedia Computer Science*, vol. 50, pp. 421–426, 2015.

[20] R. J. McEliece and D. V. Sarwate, "On sharing secrets and reed-solomon codes," *Communications of the ACM*, vol. 24, no. 9, pp. 583–584, 1981.

[21] P. Paillier, "Paillier encryption and signature schemes." 2005.

[22] R. Dingledine, N. Mathewson, P. F. Syverson *et al.*, "Tor: The second-generation onion router." in *USENIX security symposium*, vol. 4, 2004, pp. 303–320.

[23] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM transactions on information and system security (TISSEC)*, vol. 1, no. 1, pp. 66–92, 1998.

[24] J. Feigenbaum, A. Johnson, and P. Syverson, "Probabilistic analysis of onion routing in a black-box model," *ACM Transactions on Information and System Security (TISSEC)*, vol. 15, no. 3, pp. 1–28, 2012.

[25] S. Rass, R. Wigoutschnigg, and P. Schartner, "Doubly-anonymous crowds: Using secret-sharing to achieve sender-and receiver-anonymity." *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, vol. 2, no. 4, pp. 27–41, 2011.

[26] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[27] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[28] S. Banabilah, M. Aloqaily, E. Alsayed, N. Malik, and Y. Jararweh, "Federated learning review: Fundamentals, enabling technologies, and future applications," *Information processing & management*, vol. 59, no. 6, p. 103061, 2022.

[29] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2020.

[30] E. T. M. Beltrán, M. Q. Pérez, P. M. S. Sánchez, S. L. Bernal, G. Bovet, M. G. Pérez, G. M. Pérez, and A. H. Celdrán, "Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges," *IEEE Communications Surveys & Tutorials*, 2023.

[31] V. P. Chellapandi, A. Upadhyay, A. Hashemi, and S. H. Żak, "On the convergence of decentralized federated learning under imperfect information sharing," *IEEE Control Systems Letters*, 2023.

[32] M. Shayan, C. Fung, C. J. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1513–1525, 2020.

[33] H. Chen, S. A. Asif, J. Park, C.-C. Shen, and M. Bennis, "Robust blockchained federated learning with model validation and proof-of-stake inspired consensus," *arXiv preprint arXiv:2101.03300*, 2021.

[34] Zero-knowledge proofs — ethereum.org. [Online]. Available: https://ethereum.org/sw/zero-knowledge-proofs/

[35] Zkml: Bridging ai/ml and web3 with zero-knowledge proofs. [Online]. Available: https://hackmd.io/@cathie/zkml

[36] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, "Sok: Security and privacy in machine learning," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2018, pp. 399–414.

[37] zkml: Evolving the intelligence of smart contracts through zero-knowledge cryptography. [Online]. Available: https://medium.com/1kxnetwork/zkml-evolving-the-intelligence-of-smart-contracts-through-zero-knowledge-cryptography-e6725412bbd1

[38] Gensyn litepaper: The hyperscale, cost-efficient compute protocol for the world's deep learning models. [Online]. Available: https://docs.gensyn.ai/litepaper/

[39] M. Labs. (2023) The cost of intelligence: Proving machine learning inference with zero-knowledge. [Online]. Available: https://drive.google.com/file/d/1tylpowpaqcOhKQtYolPlqvx6R2Gv4IzE/view

[40] T. Liu, X. Xie, and Y. Zhang, "zkcnn: Zero knowledge proofs for convolutional neural network predictions and accuracy," Cryptology ePrint Archive, Paper 2021/673, 2021, https://eprint.iacr.org/2021/673. [Online]. Available: https://eprint.iacr.org/2021/673

[41] S. Gao, H. Fu, H. Zhang, J. Zhang, and G. Li, "Zawa: A zksnark wasm emulator," *Proc. ACM Program. Lang.*, vol. 1, no. 1, November 2022. [Online]. Available: https://jhc.sjtu.edu.cn/ hongfeifu/manuscriptb.pdf

[42] Ethereum Optimism Ecosystem Contributions, "Rfp: Op stack zero knowledge proof #61," GitHub, https://github.com/ethereum-optimism/ecosystem-contributions/issues/61?ref=blog.oplabs.co.

[43] hyperoracle. (2023) Opml: Optimistic machine learning on blockchain. [Online]. Available: https://ethresear.ch/t/opml-optimistic-machine-learning-on-blockchain/16234