Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 6467866, 12 pages
https://doi.org/10.1155/2022/6467866

Research Article

An Anonymous Verifiable Random Function with Applications in Blockchain

Shuang Yao [1,2] **and Dawei Zhang** [1,2]

[1]*Department of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China*
[2]*Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing 100044, China*

Correspondence should be addressed to Dawei Zhang; dwzhang@bjtu.edu.cn

Received 15 November 2021; Revised 14 March 2022; Accepted 29 March 2022; Published 19 April 2022

Academic Editor: Liquan Chen

Verifiable random function is a powerful function that provides a noninteractively public verifiable proof for its output. Recently, verifiable random function has found essential applications in designing secure consensus protocols in blockchain. How to construct secure and practical verifiable random functions has also attracted more and more attention. In this paper, we propose a practical anonymous verifiable random function. Security proofs show that the proposed anonymous verifiable random function achieves correctness, anonymity, uniqueness, and pseudorandomness. In addition, we show a concrete application of our proposed anonymous verifiable random function in blockchain to improve the consensus mechanism for Hyperledger fabric. Finally, we implement the proposed anonymous verifiable random function and evaluate its performance. Test results show that the proposed anonymous verifiable random function supports faster computing operations and has a smaller proof size.

1. Introduction

The notion of verifiable random function was proposed by Micali et al. [1] in 1999. In a verifiable random function, the verifier can verify the function value y generated by the prover using a random message m and secret key sk via the proof π and public key pk by the prover in a noninteractive and public way as shown in Figure 1. A common verifiable random function should satisfy the following security properties. First, the verifier that receives the function value y and the corresponding proof π generated by the prover is able to verify that y is computed correctly on input m. Second, there is a unique function value y corresponding to each public key pk and the verifiable random function input m. Finally, there is no efficient adversary that can distinguish a function value y from a random element.

Since verifiable random function was proposed, it has been widely used in practice such as lottery systems [2], E-cash [3, 4], and many other situations [5]. It is also applied in Domain Name Security Extension (DNSSEC) protocol [6] to prevent offline zone enumeration attacks. Besides these traditional usages, one of the most attractive applications recently is that it is used in blockchain to improve consensus mechanisms. Consensus mechanisms play an important role in blockchain for the reason that they are responsible for achieving data consistency among untrusted nodes in blockchain. Applications utilizing verifiable random function include Algorand [7], Dfinity [8], and Ouroboros [9–11]. The main route that verifiable random function used in consensus mechanism can be roughly summarized as follows. Take Algorand for example; each user has a public key pk and a secret key sk. The user computes the function value y through verifiable random function on a random input m. If y lies in the preset range, the user will be a committee member. However, this relationship can also be publicly verified by all users in Algorand through the verification algorithm of verifiable random function. Furthermore, in public blockchain, current Proof-of-Stake (PoS) protocols inherently disclose both the identity and the wealth of the stakeholders and thus seem incompatible with privacy-preserving cryptocurrencies [12], so it is necessary to provide a construction a privacy-preserving PoS

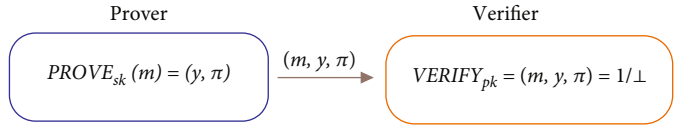

Figure 1: Verifiable random function.

protocol, that is, one where the identity of the lottery winner is kept secret by the protocol to satisfy the anonymity requirement. This also promotes the introduction of anonymous verifiable random function.

Numerous efforts have been invested in the pursuit of diversifying and simplifying constructions and underlying assumptions of verifiable random functions [13–21]. Most of them are based on RSA assumption or other complex assumptions. As verifiable random function has been gradually applied in various scenarios recently, how to construct a more efficient verifiable random function that satisfies different security properties attracts more and more attention. Ganesh et al.[12] put forward the first verifiable random function that is anonymous, which can be seen as an independent interest. There are also many other constructed verifiable random functions based on various theoretical assumptions. They all achieve the basic properties of verifiable random function except for anonymity. Therefore, in this paper, we aim to construct a more efficient anonymous verifiable random function (AVRF) that is applicable for building secure consensus mechanisms.

In addition, as one of the most popular consortium blockchains, Hyperledger fabric [22], has leveraged the benefits of both public and private blockchains. The consensus framework in Hyperledger fabric is different from public blockchain. In Hyperledger fabric, there are three types of nodes. They are endorsers, orderers, and validators. Endorsers endorse a transaction that is proposed by the transaction proposer; then, orderers block transactions and broadcast them. Validators validate transactions and update blocks on the chain. However, there is a drawback in Hyperledger fabric. Endorsing peers (endorsers) are quite essential in Hyperledger fabric as they are responsible for executing transactions proposals to ensure the transaction legality and they can directly process lots of sensitive transaction data, but these endorsing peers in consortium blockchain such as Hyperledger fabric are predetermined and endorser's identity is known to all participants. Thus, they are more likely to be attacked such as attacks possible on selective endorsers to block certain transactions, and it is necessary to construct an optimized consensus scheme with privacy properties such as randomness and anonymity for permissioned blockchain Hyperledger fabric. More precisely, endorsers should be chosen randomly, and their identities are anonymous to avoid possible attacks. Therefore, we also utilize our proposed anonymous verifiable random function to optimize the consensus mechanism in Hyperledger fabric for eliminating this drawback.

The main contributions of this paper are summarized as follows:

(i) We propose and construct an anonymous verifiable random function. The verifier can publicly verify the correctness of the function value that is the output of the prover. Meanwhile, the prover is anonymous to the verifier

(ii) We show a concrete application of the proposed verifiable random function. We use our anonymous verifiable random function to improve the consensus mechanism of Hyperledger fabric. We also analyze the security and the performance of the optimized consensus mechanism

(iii) We give theoretical analysis of the proposed anonymous verifiable random function. We also implement our anonymous verifiable random function, EC verifiable random function, Dodis verifiable random function, and the Ganesh et al. anonymous verifiable random function to evaluate their performance. Theoretical and experimental analysis results show that our anonymous verifiable random function has higher computation efficiency and a smaller proof size

1.1. Organization. We first introduce some related works in Section 2. We recall some necessary preliminaries in Section 3, and then, in Section 4, we give the concrete construction and detailed security proofs of our anonymous verifiable random function. In Section 5, we show an application of the proposed anonymous verifiable random function to improve consensus mechanism in blockchain. We then implement our anonymous verifiable random function and analyze its performance in Section 6. Conclusions are drawn in Section 7.

2. Related Works

The concept of verifiable random function was first put forth by Micali et al. [1]. Since then, verifiable random function based on elliptic curves [13] and pairing-based verifiable random function [13] have been gradually proposed. Dodis proposed a verifiable random function based on RSA[14]. In addition, some postquantum verifiable random functions [23, 24] are also proposed, but they do not have good performance in practice. Esgin et al.[25] put forward the first practical postquantum verifiable random function. It achieves significant increase in the communication size and was applied to Algorand. There are also many verifiable random functions [15, 21, 26] based on certain theoretical assumptions that have been constructed. Though these verifiable random functions all guarantee the security in pseudorandomness and uniqueness, they do not take the prover's privacy into consideration, and they cannot achieve anonymity. Ganesh et al.[12] constructed the first verifiable random function that is anonymous. In this anonymous

verifiable random function, the verifier can publicly verify the correctness of the function value as well as not reveal the public key in the verification.

In recent, verifiable random function has been widely used for consensus construction because of its randomness and public verifiability. Micali et al. proposed Algorand [7] that combines the verifiable random function and Practical Byzantine Fault Tolerance (PBFT) to select proposer and verifier committees. It avoids targeted attacks at chosen participants and achieves a high efficiency. Dfinity [8] is similar to Algorand. It actually uses Boneh–Lynn–Shacham (BLS) based verifiable random function to produce random seed which acts as the source of randomness for leader selection and leader ranking. Praos [9] is an optimized version of Ouroboros [9]. Instead of using a secure multiparty implementation of a coin-flipping protocol to produce the randomness for the leader election process, Praos uses verifiable random function for random selecting a slot leader from the stakeholder. It also prevents the adversary from learning the slot leader's identity ahead of time. Ganesh et al.[12] take the privacy properties of PoS protocol into consideration. They show that it is possible to add privacy to PoS protocols and give a privacy-preserving version of a popular PoS protocol. Most of these usages of verifiable random function mainly focus on the public blockchain. There are few researches that pay attention to providing privacy-preserving consensus scheme of the permissioned blockchain as their key nodes are predefined, and they are more likely to be attacked.

3. Preliminaries

In this section, we introduce the related hard problem and the complexity assumption, the detail of verifiable random function, and the anonymous random function. Notations used in the paper are summarized in Table 1.

3.1. Hard Problem and Complexity Assumption. Let $\mathbb{G}$ be a cyclic group of order q, where q is a prime number and it is λ bits. g is a generator of group $\mathbb{G}$. Given group elements $(\alpha = g^a, \beta = g^b, \gamma = g^{ab})$ as $a, b, c \in \mathbb{Z}_q^*$, the decisional Diffie-Hellman (DDH) problem is to distinguish between (α, β, γ) and (α, β, g^c).

We say that the DDH assumption holds if there is no probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$ that has advantage at least ε in solving the DDH problem in $\mathbb{G}$.

3.2. Verifiable Random Function. Let verifiable random function be a tuple of algorithms (Gen, Prove, and Verify) that are defined as follows:

(1) $\text{Gen}(1^\lambda)$: the Gen algorithm takes a security parameter λ as input. It generates public key pk and secret key sk. It outputs a key pair (pk, sk)

(2) $\text{Prove}(m, \text{sk})$: the Prove algorithm takes $m \in \{0,1\}^{\text{in}(\lambda)}$ and secret key sk as input. It generates a function value y and a proof π, then it outputs (y, π)

Table 1: Notations.

Symbol	Description
λ	Security parameter
$\mathbb{G}$	A cyclic group
g	A generator of group $\mathbb{G}$
H	Hash function
H_0	Hash function
H_1	Hash function
$\text{In}(\lambda)$	input length
$\text{Out}(\lambda)$	output length
pk	Public key
sk	Secret key

(3) $\text{Verify}(\text{pk}, m, y, \pi)$: the Verify algorithm takes a public key pk, m, a function value y, and a proof π as input. It outputs 1 or $\perp$

The verifiable random function satisfies correctness, pseudorandomness, and uniqueness as defined in the following:

(i) Correctness. For all (pk,sk) generated from the Gen algorithm and all update public key pk' generated by the KeyUpdate algorithm, and all $m \in \{0,1\}^{\text{in}(\lambda)}$, if $(y, \pi) \longleftarrow \text{Prove}(\text{pk}', m, \text{sk})$, then $\text{Verify}(pk', y, \pi) = 1$

(ii) Pseudorandomness. For any pair of PPT $(\mathcal{A}_1, \mathcal{A}_2)$, the following probability is $\text{negl}(\lambda)$:

$$\Pr\left[\begin{array}{c} (\text{pk}, \text{sk}) \longleftarrow \text{Gen}\left(1^\lambda\right); \\ (m, \text{state}) \longleftarrow \mathcal{A}_1^{\text{Prove}(\cdot)}(pk); \\ \eta = \eta' : \; y_0 = F_{sk}(m); y_1 \longleftarrow \{0,1\}^{\text{out}(\lambda)}; \\ \eta \longleftarrow \{0,1\}; \\ \eta' \longleftarrow \mathcal{A}_2^{\text{Prove}(\cdot)}\left(y_\eta, \text{state}\right) \end{array}\right] - \frac{1}{2} \tag{1}$$

Concretely, the definition means that no function value can be distinguished from random, even after seeing any other function values together with their corresponding proofs.

(i) Uniqueness. No PPT adversary $\mathcal{A}$ can output values $(p\text{k}, m, y_1, y_2, \pi_1, \pi_2)$ such that $y_1 \neq y_2$ and

$$\text{Verify}_{\text{pk}}(m, y_1, \pi_1) = \text{Verify}_{\text{pk}}(m, y_2, \pi_2) = 1 \tag{2}$$

3.3. Anonymous Verifiable Random Function. We also briefly review the anonymous verifiable random function proposed in [12]. Let anonymous verifiable random function be a tuple of algorithms (Gen, Update, Prove, and Verify) as defined in the following:

(i) $\text{Gen}(1^\lambda)$: it takes a security parameter λ as input. It generates public key pk and secret key sk. It outputs a key pair (pk, sk)

(ii) $\text{Update}(\text{pk})$: it takes as input the public key pk and updates the public key pk. It outputs the updated public key pk'

(iii) $\text{Prove}(m, \text{pk}', \text{sk})$: it takes as input $m \in \{0,1\}^{\text{in}(\lambda)}$, the updated public key pk', and the secret key sk. It generates a function value Y and a proof π; then, it outputs (pk', Y, π)

(iv) $\text{Verify}(\text{pk}', m, Y, \pi)$: it takes as input the updated public key pk', m, a function value Y and a proof π. It outputs 1 or $\perp$

A function family $F(\cdot)$: $\{0,1\}^{\text{in}(\lambda)} \longrightarrow \{0,1\}^{\text{out}(\lambda)}$ is a family of anonymous verifiable random functions, if there is a tuple of algorithms (Gen, Update, Prove, and Verify) that satisfies the following properties [12]:

(i) Correctness. For all (pk, sk) generated from the Gen algorithm, all update public key pk' generated by the Update algorithm, and all $m \in \{0,1\}^{\text{in}(\lambda)}$, if $(y, \pi) \longleftarrow \text{Prove}(\text{pk}', m, \text{sk})$, then $\text{Verify}(\text{pk}', y, \pi) = 1$

(ii) Pseudorandomness. For any pair of PPT $(\mathscr{A}_1, \mathscr{A}_2)$, the following probability is $\text{negl}(\lambda)$:

$$\Pr\left[\eta = \eta' \wedge m \notin Q_1 \cup Q_2 : \begin{array}{c} (\text{pk}, m) \longleftarrow \text{Gen}\left(1^\lambda\right); \\ (Q_1, m, \text{state}) \longleftarrow \mathscr{A}_1^{\text{Prove}(\cdot)}(\text{pk}); \\ y_0 = F_{sk}(m); \\ y_1 \longleftarrow \{0,1\}^{\text{out}(\lambda)}; \\ \eta \longleftarrow \{0,1\}; \\ \left(Q_2, \eta'\right) \longleftarrow \mathscr{A}_2^{\text{Prove}(\cdot)}\left(y_\eta, \text{state}\right) \end{array}\right] - \frac{1}{2} \tag{3}$$

The sets Q_1, Q_2 contain all the queries made to the Prove oracle. The random variable state stores information that $\mathscr{A}_1$ can save and pass on to $\mathscr{A}_2$.

(i) Uniqueness. No PPT adversary can output values $(\text{pk}, m, y_0, y_0, \pi_0, \pi_1)$ such that $y_0 \neq y_1$ and

$$\text{Verif}y_{\text{pk}}(m, y_0, \pi_0) = \text{Verify}_{\text{pk}}(m, y_1, \pi_1) = 1 \tag{4}$$

(ii) Anonymity. For any PPT algorithm A, the following probability is $\text{negl}(\lambda)$:

$$\Pr\left[\eta = \eta' : \begin{array}{c} (\text{pk}_0, \text{sk}_0) \longleftarrow \text{Gen}\left(1^\lambda\right); \\ (\text{pk}_1, \text{sk}_1) \longleftarrow \text{Gen}\left(1^\lambda\right); \\ m \longleftarrow A(\text{pk}_0, \text{pk}_1); \\ \text{pk}'_0 \longleftarrow \text{Update}(\text{pk}_0); \\ (y_0, \pi_0) = \text{Prove}_{\text{sk}_0}\left(\text{pk}'_0, m\right); \\ \text{pk}'_1 \longleftarrow \text{Update}(\text{pk}_1); \\ (y_1, \pi_1) = \text{Prove}_{\text{sk}_1}\left(\text{pk}'_1, m\right); \\ \eta \longleftarrow \{0,1\}; \\ \eta' \longleftarrow A\left(\text{pk}'_\eta, y_\eta, \pi_\eta\right) \end{array}\right] - \frac{1}{2} \tag{5}$$

4. Construction of Our Anonymous Verifiable Random Function

In this section, we give the concrete construction of the proposed anonymous verifiable random function. The proposed anonymous verifiable random function contains a tuple of algorithms (Gen, Update, Prove, and Verify) as the following shows:

(1) $\text{Gen}(1^\lambda)$: the Gen algorithm takes as input the security parameter λ. This algorithm randomly chooses $a, b \in \mathbb{Z}_q^*$; then, it computes $h_1 = g^a$ and $h_2 = g^b$. Thus, the public key is $\text{pk} = (g, h_1, h_2)$ and the secret key is $\text{sk} = (a, b)$. The Gen algorithm returns public key pk and secret key sk, where the secret key sk is kept secretly

(2) $\text{Update}(\text{pk})$: the Update algorithm takes as input the public key pk. This algorithm randomly chooses $r \in \mathbb{Z}_q^*$; then, it computes $g' = g^r$, $h'_1 = h_1^r$, and $h'_2 = h_2^r$. Therefore, the updated public key is set as $\text{pk}' = (g', h'_1, h'_2)$. The Update algorithm returns the updated public key pk'

(3) $\text{Prove}(\text{pk}', m, \text{sk})$: the Prove algorithm takes as input the updated public key pk', a random input m, and secret key sk. This algorithm generates the function value y and the corresponding proof π as the following shows:

(i) It calculates $\sigma = H_0(h'^{\,b}_1, m)$ and $s = b + \sigma/a$. So the function value can be computed as $y = H_1(s \oplus m)$

(ii) It sets the proof π as $\pi = (\sigma, s)$, and the function value is y. It returns the updated public key, the function value, and the proof (pk', y, π)

(4) $\text{Verify}(\text{pk}', m, y, \pi)$: the Verify algorithm takes as input the updated public key pk', a random input m, function value y, and the proof $\pi = (\sigma, s)$. It computes $w = h'^{\,s}_1 \cdot g'^{-\sigma}$; then, it determines whether equation (6) and equation (7) hold:

$$H_0(w, m) = \sigma, \tag{6}$$

$$H_1(s \oplus m) = y \tag{7}$$

If equation (6) and equation (7) all hold, the Verify algorithm outputs 1. Otherwise, the Verify algorithm outputs $\perp$.

We then prove the proposed anonymous verifiable random function satisfies correctness, anonymity, uniqueness, and pseudorandomness.

(i) Correctness. The correctness of the proposed anonymous verifiable random function represents that it can generate a function value y on any random input m with secret key through the Prove algorithm and also compute a proof π that y was computed correctly

For all public key and secret key $(\mathrm{pk}, \mathrm{sk})$ generated by the Gen algorithm, all updated public key pk' generated from the Update algorithm, all $m \in \{0,1\}^{\mathrm{in}(\lambda)}$, proof π, and function value y generated by the Prove algorithm, we have

$$h'_1{}^{s} \cdot g'^{-\sigma} = g^{ra(b+\sigma/a)} \cdot g^{-r\sigma} = g^{rab} \cdot g^{r\sigma} \cdot g^{-r\sigma} = h'_1{}^{b}. \tag{8}$$

So we get $H_0(h'_1{}^{s} \cdot g'^{-\sigma}, m) = \sigma$ and $H_1(s \oplus m) = y$. Therefore, the function value y can be determined by secret key sk and m and can be verified by proof π and public key pk. The proposed anonymous verifiable random function satisfies correctness.

(ii) Anonymity. The anonymity of the proposed verifiable random function means that the verification does not reveal the public key. We adopt the idea about anonymity from the original anonymous verifiable random function that there are lots of public keys under the same secret key, and two different evaluations under the same secret key cannot be linked to a public key

We prove that the proposed anonymous verifiable random function is anonymous as the following shows.

Theorem 1. *If the DDH assumption holds in group $\mathbb{G}$, the proposed anonymous verifiable random function satisfies anonymity.*

Proof of Theorem 1. Let $\mathscr{A}$ be the adversary that wins the anonymity game. We can build an algorithm $\mathscr{B}$ to break the DDH assumption. $\mathscr{B}$ receives $(g, g_1 = g^a, g_2 = g^b, g_3 = g^c)$ and determines whether it is a DDH tuple or not. The algorithm $\mathscr{B}$ performs as the following shows:

(i) The algorithm $\mathscr{B}$ randomly selects $r \in \mathbb{Z}_q^*, \alpha \in \{0,1\}$. It computes the public key pk_α as $\mathrm{pk}_\alpha = (g^r, g_1 = g^{ar}, g^{dr})$; then, it honestly executes the Gen algorithm to generate $\mathrm{pk}_{1-\alpha}$. The algorithm $\mathscr{B}$ returns pk_0 and pk_1 to the adversary $\mathscr{A}$

(ii) Once receiving the random input m, the algorithm $\mathscr{B}$ computes the updated public key as $\mathrm{pk}' = (g_2^r = g^{br}, g_3^r = g^{cr}, g^{dbr})$. It sets $w = g^{ad}$; then, it computes $\sigma = H_0(w, m)$, $s = d + \sigma/a$ and the function value $y = H_1(s \oplus m)$. It sets the proof π as $\pi = (\sigma, s)$. The algorithm $\mathscr{B}$ returns (pk', y, π) to the adversary $\mathscr{A}$

(iii) Let η be the output of the adversary $\mathscr{A}$. If $\eta = \alpha$, the algorithm $\mathscr{B}$ outputs "DDH tuple," otherwise the algorithm $\mathscr{B}$ outputs "not a DDH tuple."

□

Supposing the adversary $\mathscr{A}$ wins the anonymity game, then the probability $\eta = \eta'$ that we defined in the anonymity experiment is $\Pr[\eta = \eta'] \geq 1/2 + (\mathrm{Adv}_{\mathscr{A}}/2)$. So we get

$$\mathrm{Adv}_{\mathscr{B}} = |\Pr[\mathscr{B} \text{ outputs } 1 | \mathrm{DDH}] - \Pr \mathscr{B} \text{ outputs } 1 | \text{non} - \mathrm{DDH}|. \tag{9}$$

If the adversary $\mathscr{B}$ receives a non-DDH tuple, then the view of the adversary $\mathscr{B}$ is independent of η for the reason that $\mathrm{pk}' = (g^{br}, g^{abr}) = (g^{br}, (g^{br})^a)$. Thus, pk' is a correctly updated public key of pk_η. The probability of $\mathscr{B}$ outputs 1 is the same as the probability that the adversary $\mathscr{A}$ wins the anonymity game we defined. Therefore, we have

$$\Pr[\mathscr{B} \text{ outputs } 1 | \mathrm{DDH}] = \Pr\Big[\mathscr{A} \text{ outputs } \eta' = \eta \text{ in the real anonymity game}\Big]. \tag{10}$$

Then, if the algorithm $\mathscr{B}$ receives a non-DDH tuple, then the view of the adversary $\mathscr{A}$ is independent of η because $\mathrm{pk}' = (g^{br}, g^{cr})$ is independent of both pk_0 and pk_1. So the algorithm $\mathscr{B}$ cannot guess α with probability more than 1/2, so we have $\Pr[\mathscr{B} \text{ outputs} 1 | \text{non} - \mathrm{DDH}] = 1/2$. Thus, we have

$$\mathrm{Adv}_{\mathscr{B}} \geq \left|\frac{1}{2} + \frac{\mathrm{Adv}_A}{2} - \frac{1}{2}\right| \geq \mathrm{Adv}_A/2. \tag{11}$$

Therefore, the proposed anonymous verifiable random function satisfies anonymity as the DDH assumption holds.

(i) Uniqueness. The uniqueness of the proposed anonymous verifiable random function means that the function value y is uniquely determined by the corresponding secret key sk and a random input m, and accepting proofs only exist for this function value y

Suppose that for all $(\mathrm{pk}, \mathrm{sk})$ generated by the Gen algorithm, all updated public key pk' generated from the Update algorithm, and all $m \in \{0,1\}^{\mathrm{in}(\lambda)}$, there exits the tuple (y_0, y_1, π_0, π_1) such that $\mathrm{Verify}(\mathrm{pk}', m, y_1, \pi_1) = \mathrm{Verify}(\mathrm{pk}',$

$m, y_0, \pi_0) = 1$, we can get $s_1 = b + \sigma_1/a = b + H_0((h'_1)^b, m)/a$ and $s_0 = b + \sigma_0/a = b + H_0((h'_1)^b, m)/a$, so

$$s_1 = s_2, \tag{12}$$

and according to the definition of y, we have $y_1 = H_1(s_1 \oplus m)$ and $y_2 = H_2(s_2 \oplus m)$. From equation (12), we have$y_1 = y_0$, as this is in contradiction with$y_1 \neq y_0$. Therefore, for all (pk, sk), all update pulic key pk′, and all $m \in \{0,1\}^{\text{in}(\lambda)}$, there does not exit any tuple (y_0, y_1, π_0, π_1) such that Verify (pk′, m, y_1, π_1)= Verify(pk′, m, y_0, π_0) = 1 and $y_1 \neq y_0$. The proposed anonymous verifiable random function satisfies uniqueness.

(ii) Pseudorandomness. We prove the pseudorandomness of the proposed anonymous verifiable random function as the following shows

Theorem 2. *If the DDH assumption holds in group $\mathbb{G}$, the proposed anonymous verifiable random function satisfies pseudorandomness.*

Proof of Theorem 2. Let $\mathbb{G}$ be a group and g is the generator of $\mathbb{G}$. Suppose that there is an adversary $\mathcal{A}$ that can break the pseudorandomness experiment we defined; then, we build a series of games as the following shows. Let W_0 be the probability that the adversary$\mathcal{A}$wins Game 0. Let $\text{Adv}_{\mathcal{A}}$ be the advantage of the adversary $\mathcal{A}$ in the pseudorandomness experiment. □

Game 0. This is the original pseudorandomness game we defined. The challenger $\mathcal{C}$ and the adversary $\mathcal{A}$ are interacted as the following shows:

(i) The challenger $\mathcal{C}$ computes public key and secret key $(\text{pk}, \text{sk}) \longleftarrow \text{Gen}(1^\lambda)$. It sends the generated public key pk to the adversary $\mathcal{A}$

(ii) The adversary $\mathcal{A}$ queries the oracle $\mathcal{O}_{\text{Prove}}$. The challenger $\mathcal{C}$ answers these queries

(iii) Once the challenger $\mathcal{C}$ receiving the message m^* that is sent by the adversary $\mathcal{A}$, the challenger $\mathcal{C}$ computes $(y_0, \pi_0) \longleftarrow \text{Prove}(\text{pk}, m^*, \text{sk})$ and randomly chooses $y_1 \in \{0,1\}^{\text{out}(\lambda)}$. It randomly chooses $\eta \in \{0,1\}$ and returns y_η to the adversary $\mathcal{A}$

(iv) The adversary $\mathcal{A}$ outputs η' which is the guess of η, and the adversary $\mathcal{A}$ wins the game if $\eta = \eta'$

So we get $\text{Adv}_{\mathcal{A}} = |\Pr[W_0] - 1/2|$.

Game 1. Game 1 is the same as Game 0 except that we make a change. We compute $g'^{\gamma\theta}$ for randomly chosen $\gamma, \theta \in \mathbb{Z}_q^*$ instead of computing $h'^{\,b}_1$. Let W_1 be the event that $\eta = \eta'$ in Game 1. The challenger $\mathcal{C}$ and the adversary $\mathcal{A}$ are interacted as the following shows:

(i) The challenger $\mathcal{A}$ computes public key and secret key $(\text{pk}, \text{sk}) \longleftarrow \text{Gen}(1^\lambda)$. It sends the generated pk to the adversary $\mathcal{A}$

(ii) The adversary $\mathcal{A}$ queries the oracle $\mathcal{O}_{\text{Prove}}$. The challenger $\mathcal{C}$ answers these queries

(iii) Once the challenger $\mathcal{C}$ receiving the message m^* that is sent by the adversary $\mathcal{C}$, it randomly chooses $\gamma, \theta \in \mathbb{Z}_q^*$ and computes $w = g'^{\gamma\theta}$. The challenger $\mathcal{C}$ computes $\sigma = H_0(w, m)$, $s = \gamma + \sigma/\theta$ and the function value $y_0 = H_1(s \oplus m)$. It sets $\pi = (\sigma, s)$; then, it obtains (y_0, π). The challenger $\mathcal{C}$ randomly chooses $y_1 \in \{0,1\}^{\text{out}(\lambda)}$. It randomly chooses $\eta \in \{0,1\}$ and returns y_η to the adversary $\mathcal{A}$

(iv) The adversary $\mathcal{A}$ outputs b', and it wins the game if $\eta = \eta'$.Since the adversary $\mathcal{A}$'s output of b' is independent of b, we have $\Pr[W_1] = 1/2$

Lemma 3. *We prove that $|\Pr[W_0] - \Pr[W_1]| = \varepsilon$, where ε is the advantage of some efficient algorithms to break the DDH advantage. It is negligible.*

Proof of Lemma 3. In Game 0, we have the tuple $(g', g'^a, g'^b, g'^{ab})$, while in Game 1, we have the tuple $(g', g'^a, g'^b, g'^{\gamma\theta})$. The adversary $\mathcal{A}$ cannot recognize the difference under the DDH assumption. We define a distinguishing algorithm $\mathcal{D}$. If the input to $\mathcal{D}$ is in the form of $(g', g'^a, g'^b, g'^{ab})$, the computation proceeds as in Game 0. So we have $\Pr[\mathcal{D}(g', g'^a, g'^b, g'^{ab}) = 1 : a, b \in \mathbb{Z}_q^*] = \Pr[W_0]$. If the input to $\mathcal{D}$ is in the form of $(g', g'^a, g'^b, g'^{\gamma\theta})$, the computation proceeds as in Game 1. So we have $\Pr[\mathcal{D}(g', g'^a, g'^b, g'^{\gamma\theta}) = 1 : a, b \in \mathbb{Z}_q^*] = \Pr[W_1]$. □

So the advantage to break the DDH assumption Adv_{DDH} is equal to $|\Pr[W_0] - \Pr[W_1]|$. As Adv_{DDH} is negligible, $|\Pr[W_0] - \Pr[W_1]| = \varepsilon$, ε is negligible.

According to the above proof, we have $\text{Adv}_{\mathcal{A}} = |\Pr[W_0] - (1/2)| = \varepsilon$, ε is negligible. Therefore, the proposed anonymous verifiable random function satisfies pseudorandomness.

5. Application of the Proposed AVRF in Blockchain

In this section, we show a specific application of the proposed anonymous verifiable random function in blockchain. As the key nodes in consortium blockchain such as endorsing peers in Hyperledger fabric are predetermined and fixed, they are more likely to be attacked. Therefore, we use the proposed anonymous verifiable random function to improve the consensus mechanism for Hyperledger fabric by randomly choosing endorsing peers instead of presetting

them. The improved consensus scheme is aimed at making the identity of endorsing peer random. It also provides identity privacy preservation of endorsing peers and reduces the risk attack of endorsing peers.

5.1. Hyperledger Fabric Consensus Mechanism Optimization Based on the Proposed AVRF. The consensus mechanism in Hyperledger fabric [22] is in the form of a more flexible trust model called "endorse-order-validate" which is different from consensus mechanism in public blockchain [27–29]. As we can see in Figure 2, in Hyperledger fabric, there are three types of nodes. They are endorsers, orderers, and validators. Firstly, in the endorsement phase, endorsers are predetermined and fixed. Endorsers execute transactions and record these results. Secondly, in the ordering step, it uses a pluggable consensus protocol to produce a totally ordered sequence of endorsed transactions grouped in blocks. These endorsed transactions are broadcasted to all peers via the gossip protocol. Next, in the validation step, validators validate the state changes from endorsed transactions with respect to the endorsement policy in the validation step.

In Hyperledger fabric, endorsing peers (endorsers) are quite essential as they are responsible for executing transactions proposals to ensure the transaction legality, and they can directly process lots of sensitive transaction data. However, as endorsers' identities are public and fixed, they are more likely to be attacked. Besides, the number of endorsers is small compared to other peers' numbers in general. It is even in single digits in some systems. This makes that there are many-to-one relationships between clients and endorsers, so it is difficult for endorsers to process transactions timely, which increases the transaction processing time.

In accordance with the above problems, we construct a noninteractive, verifiable, and optimized consensus scheme for randomly selecting endorsers based on the proposed anonymous verifiable random function. We use the candidate set of endorsing peers and randomly select endorsing peers in the candidate set through our anonymous verifiable random function. The usage of anonymous verifiable random function achieves the identity privacy of endorsers before endorsement, and this randomly expands the number of endorsing peers.

As we can see in Figure 3, the optimized consensus scheme based our anonymous verifiable random function is defined as follows:

(1) The client generates proposal proposal = <req, m, clientsig > . req is the transaction data which includes chaincode and its parameters. m is the random input that satisfies $m \in \{0,1\}^{\text{in}(\lambda)}$. The client signs these data and generates clientsig. It sends the proposal to the candidate set of endorsing peers; then, the client starts a timer

(2) The candidate endorsing peer verifies the signature clientsig to check the integrity. If the verification fails, it aborts. Otherwise, the candidate endorsing peer performs as follows:

(i) The candidate endorsing peer executes the anonymous verifiable random function Update algorithm to generate the update public key pk'. It executes $\text{Prove}(\text{pk}', m, \text{sk})$ to get the function value y and proof π

(ii) The candidate endorsing peer compares whether $(H(y)/2^{h\text{len}}) > \eta$ holds. η is the predetermined threshold. H is a hash function and $h$len is the length of $H(y)$. If it holds, it means that the candidate endorsing peer is an endorser. It goes to the next step. Otherwise, it aborts

(iii) If a candidate endorsing peer has confirmed that it is an endorser, it executes the proposal to generate read and write set rw_set and the endorsing result ed; then, it computes the signature of $(\text{rw\_set}, \text{ed}, (\text{pk}', y, \pi))$ as epsig, while si $\text{g}_{\text{sk}} = ra$ and $\text{sig}_{\text{pk}} = h'_1 = g^{ra}$. Therefore, the proposal response message as res_pro = <rw_set, ed, (pk', y, π), epsig > . It sends the proposal response message *res_pro t*o the client

(3) The client continuously receives proposal response messages res_pro from different endorsers before the timer runs out. It performs as the following shows:

(i) The client verifies the signature epsig for checking the integrity. If the verification fails, it aborts. Otherwise, it executes $\text{Verify}(\text{pk}', y, \pi)$ to verify the function value y. On one hand, if there is an adversary that replaces (pk', y, π) without secret key, which may lead to some malicious endorsing peers without endorsers qualifications to become logical endorsers. This will influence transaction endorsing results. However, when the client receives the replaced response message, it first verifies the signature epsig, then it verifies the function value y. For the reason that the signature satisfies unforgeability and the anonymous verifiable random function satisfies uniqueness, the replaced response message will not pass these verifications, and the malicious endorsing peer without endorser qualification will not become the logical endorser to influence transaction endorsing results. On the other hand, our anonymous verifiable random function can also be extended to provide some level of unpredictability under malicious key generation. In order to achieve this goal, in the Prove algorithm, it adds a computation $v = H_2(y, m)$, where H_2 is a hash function. Also, let $\pi = (\sigma, s, y)$. It outputs (π, v). In the Verify algorithm, it adds an verification to check whether $v = H_2(y, m)$ holds. In this case, our extended anonymous verifiable random function can provide unpredictability under malicious key generation. It means that an adversary that can maliciously choose the verifiable random keys cannot skew the output distribution, as long as the adversary has no information on the

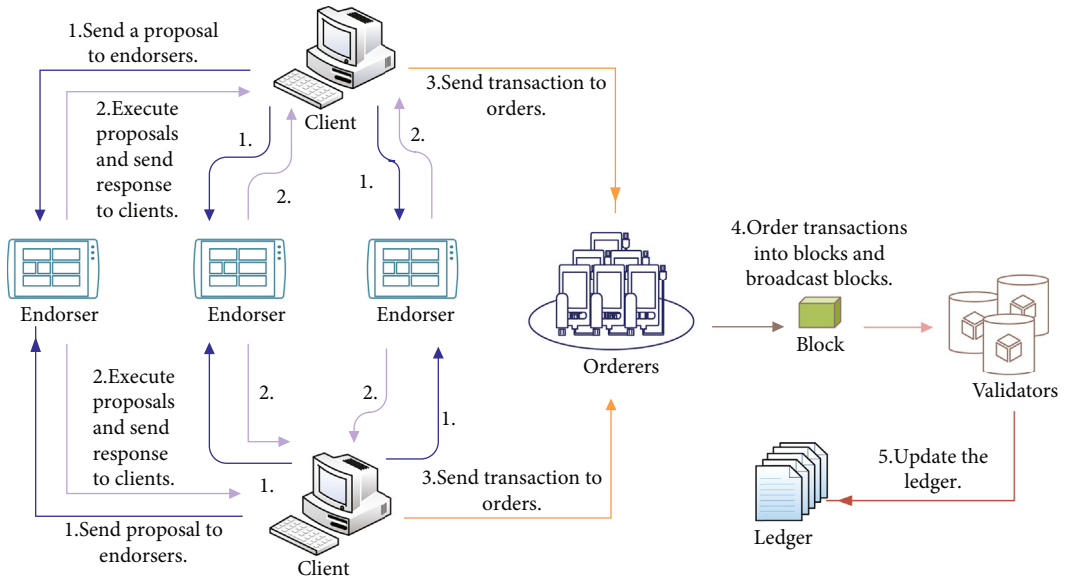

Figure 2: The consensus mechanism of Hyperledger fabric.

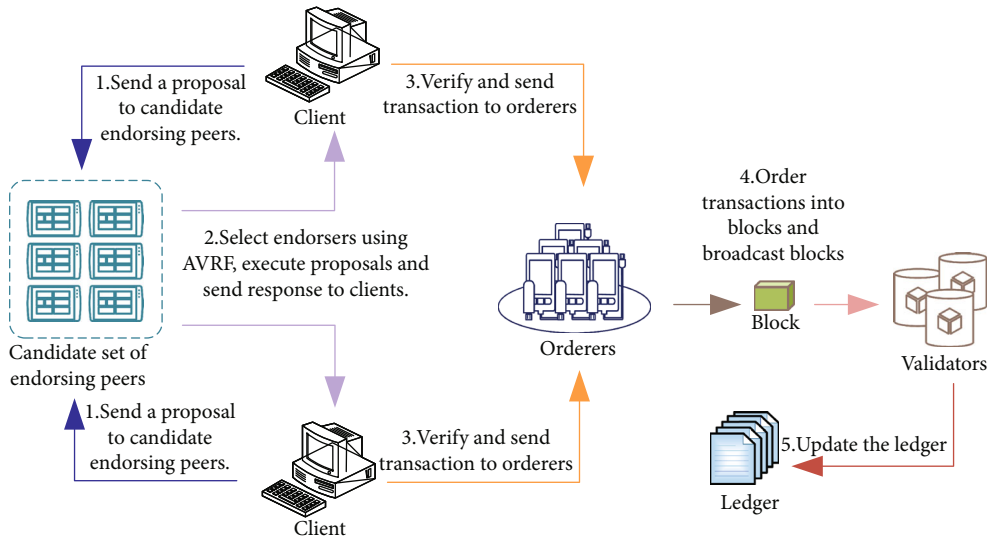

Figure 3: The optimized consensus scheme.

random input m when choosing its verifiable random function keys. We adopt the idea about unpredictability under malicious key generation from [9] and [25] that have given us detailed explanation and proof

(ii) The client computes $H(y)/2^{hlen}$ and checks whether $(H(y)/2^{hlen}) > \eta$ holds. If it holds, the proposal response message is from a logical endorser. The client sends these transactions to orderers

(4) Orderer monitors and receives all transactions and block transactions as block $< \text{tx} >^{n}_{\text{sign}}$. Let tx^n denote that there are n orderly transactions in a block. Orders sign blocks and broadcast them

(5) Validator receives the block, then it verifies the signature of the block the read and write set and updates the ledger. After all these steps, it represents that a consensus of transactions initiated by clients is gained

5.2. Analysis of the Optimized Consensus Scheme

5.2.1. Security Analysis

(1) Randomness. In our optimized consensus scheme, endorsers are randomly selected from the candidate set of

Table 2: Efficiency comparison.

	Dodis VRF	Ganesh et al. AVRF	The proposed AVRF
Prove time complexity	$P + E_T + E$	$5E + 2H$	$4E + 2H$
Verify time complexity	$E_T + 3P + M$	$4E + 2M + 2H$	$2E + M + H$
Proof size	$\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$	$\lvert\mathbb{G}\rvert + 2\lvert\mathbb{Z}_q\rvert$	$2\lvert\mathbb{Z}_q\rvert$
Prove computation overhead (ms)	2.0	1.9	1.6
Verify computation overhead (ms)	3.3	1.2	0.9

endorsing peers via the proposed anonymous verifiable random function instead of predetermined. Whether a candidate endorsing peer is an endorser or not is determined by the function value y which is the random output of the anonymous verifiable random function's Prove algorithm. Only the function value y satisfies that $(H(y)/2^{hlen}) > \eta$; a candidate endorsing peer is chosen as an endorser. As we can see from the definition of anonymous verifiable random function, y satisfies randomness, so the selection of endorsers is random. This reduces the centralization of endorsing peers.

At the same time, in the original consensus scheme, clients continue to process the transaction only after they have compared all endorsing results that were sent from endorsers and results are all same. So, the adversary can easily control endorsing results to destroy the correctness of transaction results if it has successfully attacked only one endorser to make endorsing results inconsistent and the client aborts the transaction. Furthermore, if the decision strategy is modified, the endorsing results are valid only if there are more than half of the results that are consistent. In this case, adversary can control the endorsing result to destroy the correctness of transaction results if more than half of endorsing peers are malicious. On the contrary, in our optimized consensus scheme, clients do not have to compare endorsing results from all predetermined endorsing peers. Endorsers are chosen randomly and dynamically; this will reduce the probability of adversary's influence on the transaction.

(2) Anonymity. In our optimized consensus scheme, endorser's identity is verifiable and anonymous to the client. As the proposed anonymous verifiable random function satisfies correctness, the endorsing peer's identity can be verified. Furthermore, observers cannot obtain the result about which candidate endorsing peers have been chosen if secret keys are not leaked. Moreover, the client can use the endorsing peer's update public keys to verify the identity validity of the endorsing peer, so the client cannot recognize the identity of endorsing peers for the reason that the proposed anonymous verifiable random function satisfies anonymity. Concretely, verification using update public keys will not reveal endorsing peers' public keys. The anonymity of our optimized consensus scheme provides privacy preservation of endorsing peers and reduces their risk of being attacked.

5.2.2. Performance Analysis of the Optimized Consensus Scheme. On one hand, for the same m, different secret key sk will generate different function value y by the Prove algorithm of the proposed anonymous verifiable random function. Therefore, different candidate endorsing peer will generate different function values y in the same transaction. Some of candidate endorsing peers will become endorsers for this transaction, while the rest of endorsing peers in candidate set will become other transactions' endorsers. The randomness of function value y ensures that the transaction is uniformly distributed to candidate endorsing peers. This reduces the workload of each endorsing peer and improves concurrent processing of transactions.

On the other hand, the transaction delay is the time it takes to initiate a proposal, endorse, validate, order, and commit transactions to the ledger. In our optimized consensus scheme, as the transaction flow is the same as the original consensus scheme except for the endorsing step, the main factor that increases the transaction processing time is that there is an extra endorser selection process and endorser identity's verification process. According to the prove time and the verify time of the proposed anonymous verifiable random function in Table 2, they are both milliseconds. It is negligible compared with the whole transaction processing time. Thus, the impact of the proposed anonymous verifiable random function on the transaction delay is negligible and there is no much difference on transaction delay between our optimized consensus scheme and the original consensus scheme.

6. Implementation and Evaluation

In this section, in order to give a better evaluation of the performance about our proposed anonymous verifiable random function, we give a reference implementation of our anonymous verifiable random function as well as the anonymous verifiable random function proposed in [12] in Python language. For convenience, we call it Ganesh et al. anonymous verifiable random function. We also implement another two representative verifiable random functions the Dodis verifiable random function [13] which is used in Algorand and the EC verifiable random function [6] that has been widely used in many scenarios such as in DNSSEC. We use the Charm [30] library to implement the elliptic curve group operations. We measure the prove time and the verify time of these verifiable random functions. Our tests are performed on a Linux desktop with an 8-core Intel Core i7-8550U 2.00 GHz processor and 8 GB of RAM. We also average the performance over 50 runs.

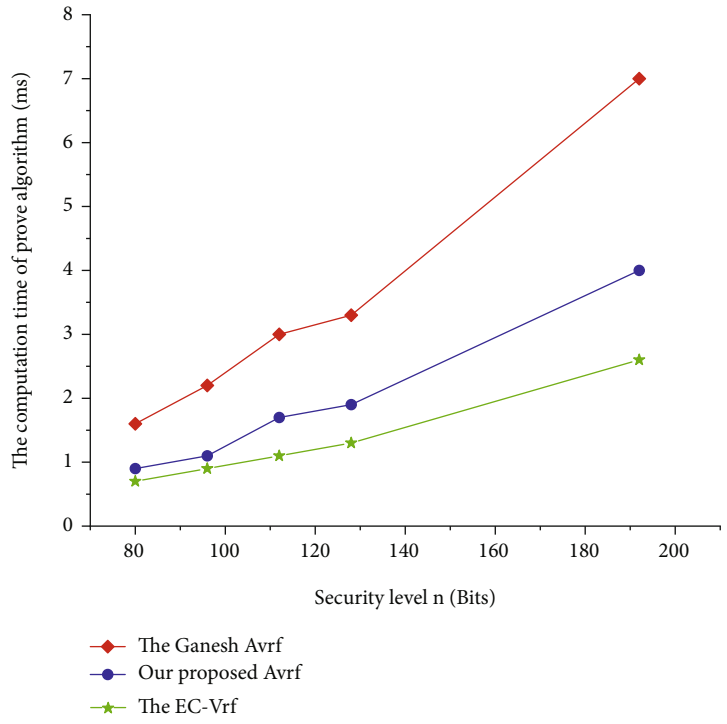

Figure 4: Prove time comparison.

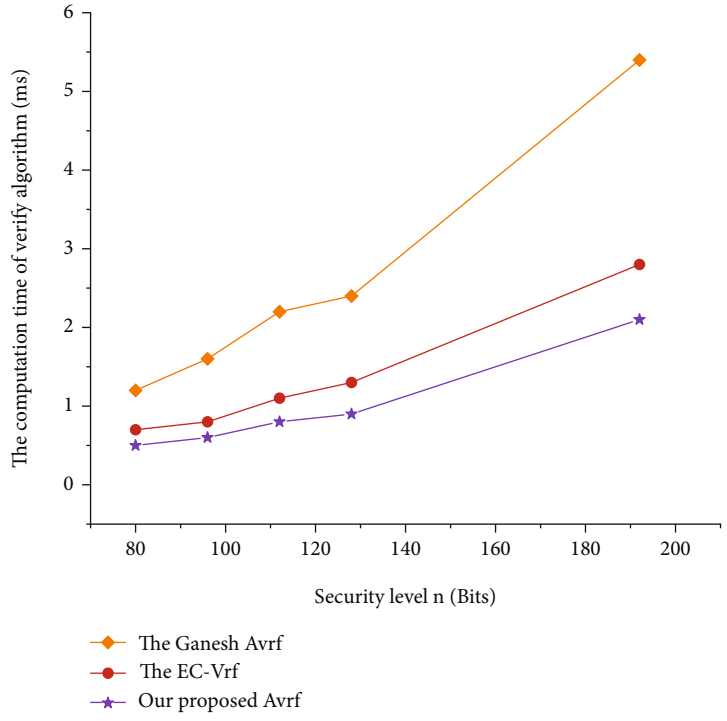

Figure 5: Verify time comparison.

In Table 2, we give the efficiency analysis by comparing our proposed anonymous verifiable random function, the Ganesh et al. anonymous verifiable random function, and the Dodis verifiable random function in terms of time complexity, computation overhead, and the size of proof π. We denote E as exponentiation operation in group $\mathbb{G}$, H as hash function, E as multiplication operation in group $\mathbb{G}_T$, P as pairing operation, M as multiplication operation in group $\mathbb{G}$, $|\mathbb{G}|$ as the size of elements in group $\mathbb{G}$, and $|\mathbb{Z}_q|$ as the size of elements in $\mathbb{Z}_q$. As we can see from Table 2, verify times

of the Dodis verifiable random function, Ganesh et al. anonymous verifiable random function, and our proposed anonymous verifiable random function are, respectively, 3.3 ms, 1.2 ms, and 0.9 ms with the 80-bit security level. It is obvious that our anonymous verifiable random function has the best performance in terms of the verify time and the proof size.

In Figure 4, we compare the computation of prove time among our anonymous verifiable random function, the Ganesh et al. anonymous verifiable random function, and the EC verifiable random function. As we set the security level as 80 bits, 96 bits, 112 bits, 128 bits, and 192 bits, respectively, the prove time of Ganesh et al. anonymous verifiable random function grows from 1.6 ms to 7.0 ms while our proposed anonymous verifiable random function increases from 0.9 ms to 4.0 ms. It is obvious that our anonymous verifiable random function has lower prove computation overhead compared with the Ganesh et al. anonymous verifiable random function. However, the prove computation overhead of our proposed anonymous verifiable random function is a little higher than the EC verifiable random function for the reason that there are extra exponentiation operations in our proposed verifiable random function to achieve anonymity, while the EC verifiable random function is not anonymous.

In Figure 5, we compare the computation of verify time among our anonymous verifiable random function, the Ganesh et al. anonymous verifiable random function and the EC verifiable random function. When security levels are set to be 80 bits, 96 bits, 112 bits, 128 bits, and 192 bits, respectively, the verify time of the Ganesh et al. anonymous verifiable random function grows from 1.2 ms to 5.4 ms and the EC verifiable random function grows from 0.7 ms to 2.9 ms. In our verifiable random function, it increases from 0.5 ms to 2.1 ms. Our anonymous verifiable random function also has the lowest verify computation overhead among these three verifiable random functions.

Therefore, the proposed anonymous verifiable random function is efficient according to the above analytical measurements and experimental evaluation as it has shorter prove and verify time as well as a smaller proof size.

7. Conclusions

In this paper, we construct an efficient anonymous verifiable random function which has a potential utilization in blockchain to build secure consensus protocols. Specially, our proposed verifiable random function is anonymous. It means that the verification will not reveal the public key of the prover. We also analyze and prove its security properties. Furthermore, we give a concrete utilization of our proposed anonymous verifiable random function to optimize the consensus mechanism of Hyperledger fabric. In addition, we implement and evaluate the proposed anonymous verifiable random function and another three representative verifiable random functions. Experimental results show that the proposed anonymous verifiable random function has lower computation overhead and a smaller proof size compared with other representative verifiable random functions. The proposed anonymous verifiable random function can also be applied to other permissioned blockchains as their transactions are processed by certain key nodes. However, to achieve a practical postquantum anonymous verifiable random function is still for future work.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This paper was supported by National Natural Science Foundation of China (Grant no. U21A20463).

References

[1] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *40th Annual Symposium on Foundations of Computer Science (cat. No. 99CB37039)*, pp. 120–130, New York, NY, USA, 1999.

[2] S. Micali and R. L. Rivest, *Micropayments revisited. Cryptographers' Track at the RSA Conference*, Springer, 2002.

[3] M. H. Au, W. Susilo, and Y. Mu, "Practical compact e-cash," in *Australasian Conference on Information Security and Privacy*, pp. 431–445, Berlin, Heidelberg, 2007.

[4] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya, "Compact e-cash and simulatable VRFs revisited," in *International Conference on Pairing-Based Cryptography*, pp. 114–131, Berlin, Heidelberg, 2009.

[5] B. Mishra, D. Jena, and S. Patnaik, "Privacy preserving file auditing schemes for cloud storage using verifiable random function," *International Journal of Communication Networks and Distributed Systems*, vol. 26, pp. 50–75, 2021.

[6] D. Papadopoulos, D. Wessels, S. Huque, M. Naor, J. Vˇcelák, and S. Goldberg, *Making NSEC5 practical for DNSSEC*, Cryptology ePrint Archive, 2017.

[7] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles*, pp. 51–68, Shanghai, China, 2017.

[8] T. Hanke, M. Movahedi, and D. Williams, "DFINITY technology overview series, consensus system," https://arxiv.org/abs/1805.04548.

[9] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros Praos: an adaptively-secure, semisynchronous proof-of-stake blockchain," *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2018, pp. 66–98, Cham, 2018.

[10] C. Badertscher, P. Gaži, A. Kiayias, A. Russell, and V. Zikas, "Ouroboros Genesis: composable proof-of-stake blockchains with dynamic availability," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 913–930, 2018.

[11] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: a provably secure proof-of-stake blockchain protocol,"

in *Annual International Cryptology Conference*, pp. 357–388, Cham, 2017.

[12] C. Ganesh, C. Orlandi, and D. Tschudi, "Proof-of-stake protocols for privacy-aware blockchains," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 690–719, Cham, 2019.

[13] Y. Dodis and A. Yampolskiy, *A verifiable random function with short proofs and keys. International Workshop on Public Key Cryptography*, Springer, 2005.

[14] Y. Dodis, *Efficient construction of (distributed) verifiable random functions. International Workshop on Public Key Cryptography*, Springer, 2003.

[15] N. Chandran, S. Raghuraman, and D. Vinayagamurthy, *Constrained pseudorandom functions: verifiable and delegatable*, Cryptology ePrint Archive, 2014.

[16] G. Guo, Y. Zhu, E. Chen, G. Zhu, D. Ma, and W. C. Chu, "Continuous improvement of script-driven verifiable random functions for reducing computing power in blockchain consensus protocols," *Peer-to-Peer Networking and Applications*, vol. 15, pp. 304–323, 2021.

[17] G. Fuchsbauer, "Constrained verifiable random functions," in *International Conference on Security and Cryptography for Networks*, pp. 95–114, Cham, 2014.

[18] S. Hohenberger and B. Waters, "Constructing verifiable random functions with large input spaces," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 656–672, Berlin, Heidelberg, 2010.

[19] T. Jager, "Verifiable random functions from weaker assumptions," in *Theory of Cryptography Conference*, pp. 121–143, Berlin, Heidelberg, 2015.

[20] D. Hofheinz and T. Jager, "Verifiable random functions from standard assumptions," in *Theory of Cryptography Conference*, pp. 336–362, Berlin, Heidelberg, 2016.

[21] N. Bitansky, "Verifiable random functions from non-interactive witness-indistinguishable proofs," *Journal of Cryptology*, vol. 33, pp. 459–493, 2020.

[22] E. Androulaki, A. Barger, V. Bortnikov et al., "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, pp. 1–15, Porto, Portugal, 2018.

[23] S. Yamada, "Asymptotically compact adaptively secure lattice IBEs and verifiable random functions via generalized partitioning techniques," in *Annual International Cryptology Conference*, pp. 161–193, Cham, 2017.

[24] R. Goyal, S. Hohenberger, V. Koppula, and B. Waters, "A generic approach to constructing and proving verifiable random functions," in *Theory of Cryptography Conference*, pp. 537–566, Cham, 2017.

[25] M. F. Esgin, V. Kuchta, A. Sakzad et al., "Practical postquantum few-time verifiable random function with applications to Algorand," in *International Conference on Financial Cryptography and Data Security*, pp. 560–578, Berlin, Heidelberg, 2021.

[26] L. Kohl, "Hunting and gathering–verifiable random functions from standard assumptions with short proofs," in *IACR International Workshop on Public Key Cryptography*, pp. 408–437, Cham, 2019.

[27] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[28] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[29] E. B. Sasson, A. Chiesa, C. Garman et al., "Zerocash: decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*, pp. 459–474, Berkeley, CA, USA, 2014.

[30] J. A. Akinyele, C. Garman, I. Miers et al., "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, pp. 111–128, 2013.