This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

An Efficient Privacy-Preserving Credit Score System Based on Noninteractive Zero-Knowledge Proof

Chao Lin, Min Luo, Xinyi Huang, *Member, IEEE*, Kim-Kwang Raymond Choo, *Senior Member, IEEE*, and Debiao He, *Member, IEEE*

***Abstract*—Credit system is generally associated with the banking and financial institutions, although it has far reaching implications for residents of countries, such as U.S., particularly for those with a poor credit history. Specifically, a credit score computation (CSC) quantifies an individual's credit value or credit risk, which is used by banking and financial institutions, as well as other entities (e.g., during purchasing of insurance policies and application of rental properties), to facilitate their decision-making (e.g., whether to approve the insurance policy purchase or the level of premium). Although a number of CSC models have been proposed in the literature for supporting different application scenarios, privacy protection of CSC is rarely considered despite the potential for leakage of user private information (e.g., registration, hobbies, credit, relationships, and inquiry). Such information can then be abused for other nefarious activities, such as identity theft and credit card fraud. Thus, in this article, we first analyze the privacy strength of existing CSC models, prior to presenting the formal definition of a privacy-preserving CSC system alongside its security requirements. Then, we propose a concrete construction based on Paillier encryption with three proposed noninteractive zero-knowledge schemes. To demonstrate feasibility of our proposal, we evaluate both its security and performance.**

***Index Terms*—Credit score computation (CSC), noninteractive zero knowledge (NIZK), Paillier encryption, privacy preserving.**

Manuscript received August 25, 2020; revised November 16, 2020; accepted December 12, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 62032005, Grant 61822202, Grant 61932016, Grant 61972294, and Grant 61872192; in part by the National Key Research and Development Program of Chiangdong Key Laboratory of Data Security and Privacy Protection under Grant 2017B030301004; in part by the Special Project on Science and Technology Program of Hubei Province under Grant 2020AEA013; in part by the Science Foundation of Fujian Provincial Science and Technology Agency under Grant 2020J02016; in part by the Natural Science Foundation of Hubei Province under Grant 2020CFA052; and in part by the Wuhan Municipal Science and Technology Project under Grant 2020010601012187. The work of Kim-Kwang Raymond Choo was supported in part by the Cloud Technology Endowed Professorship and in part by the National Science Foundation CREST under Grant HRD-1736209. *(Corresponding author: Min Luo.)*

Chao Lin is with the College of Mathematics and Informatics and the Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou 350117, China, and also with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China (e-mail: linchao91@qq.com).

Min Luo and Debiao He are with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China (e-mail: mluo@whu.edu.cn; hedebiao@163.com).

Xinyi Huang is with the College of Mathematics and Informatics and the Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou 350117, China (e-mail: xyhuang81@gmail.com).

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security and the Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: raymond.choo@fulbrightmail.org).

Digital Object Identifier 10.1109/JSYST.2020.3045076

I. Introduction

THE credit system is a platform that provides some form of credit evaluation for both individuals and nonindividual entities (e.g., organizations), which determines the "financial trustworthiness" of the individual and/or nonindividual entity [1], [2]. For instance, in the U.S., credit score is widely used in a broad range of applications, for example to determine whether an individual's application for, say a credit card, home/automobile loan, etc., will be approved or an individual has to pay a higher insurance premium or higher interest rate (due to low credit score).

As shown in Fig. 1, a credit system generally comprises three types of participants (i.e., users, credit bureaus, and creditors). The user is a key part of the credit system, whose credit and loan activities (new account creation, account balance/credit card utilization, credit inquiries, and payment history) are reported to the credit bureaus. The latter is responsible for collecting, recording, and distributing relevant information (collectively referred to as "credit data") about the user's credit activities [3].

Such credit data are then requested by the creditors to compute the user's credit score, which is then used to inform some decision-making. In other words, the credit score represents the credit value/risk/health of an individual or entity, which is a reference value for trust assessment [4]. The score is generated by analyzing the user's credit data using an algorithm (i.e., risk model), a process known as credit score computation (CSC). Different models consider different factors and weights to compute the final credit score. For example, FICO scores are calculated based on the user's payment history (35%), amounts owed (30%), length of credit history (15%), new credit (10%), and credit mix (10%).[1]

There are different risk models in the literature, such as least squares support vector machines ensemble models for credit scoring [5], a measure of creditworthiness for sound financial decision-making [6], a partial credit model [7], and a fuzzy logistic regression based credit scoring model [8]. Using these models, creditors can take as input the credit data obtained from the bureaus and quickly obtain a credit score. However, these models do not consider the privacy protection of user credit data and their corresponding weights. That is, credit data in existing risk models are obtained directly by the creditors and the weights

[1]https://www.myfico.com/credit-education/whats-in-your-credit-score/ last accessed December 11, 2018.

1937-9234 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Texas at San Antonio. Downloaded on March 16,2021 at 12:11:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

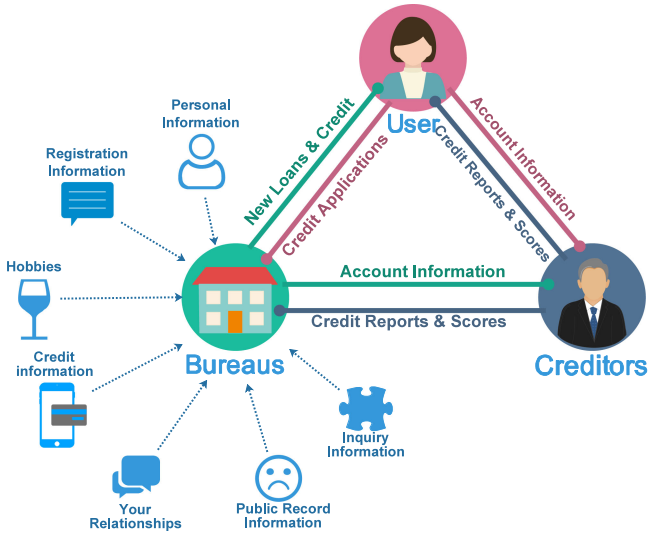


Fig. 1. General architecture of a credit system.

in the model are all publicly known. This is a limiting factor in establishing a diversified and multilevel credit system.

User credit data can be mined for commercial/financial gains, for example by reselling such information to marketers and other business entities, or even cybercriminals. The data can then be mined to profile users (e.g., user online and shopping behaviors in order to provide targeted advertising), facilitate cybercriminal activities, such as identity theft, or identify individuals to facilitate illegal tracking and surveillance (e.g., by nation states) [9]–[11]. Clearly, this is a topic of concern to most users. Also, depending on the application context, making weights used in the computation of the scores publicly known can also be abused by individuals or entities to game the system, for example to take a particular course of actions to enhance the credit score.

To protect the privacy of weights and credit data, the first thought is using a secure two-party computation [12], [13]. It enables creditor and bureaus to evaluate a function (i.e., CSC formula) cooperatively without revealing to either party anything (e.g., weights and credit data) beyond the final credit score. For example, a simple function of computation based on weight computing is $f(k_1, \ldots, k_t, m_1, \ldots, m_t) = \Sigma_{i=1}^{t}(k_i \cdot m_i)$, where k_i and m_i are weight and credit data, respectively. However, the existing two-party computation solutions, such as [14]–[16], generally do not consider the property of verification, meaning that a curious creditor (or bureaus) may provide fake weights (resp. credit data) to obtain each other's private information. Worse still, the creditor may lie about the final credit score to the bureaus or user. Thus, specific zero-knowledge proofs should be designed to further enhance the privacy of credit data and weights.

In this article, inspired by Goethals *et al.* [15], we also explore the potential of homomorphic encryption (HE) in the privacy-preserving design of CSC. Here, an HE scheme (e.g., [17], [18]) allows one to update the message in ciphertext without the need for decryption. That is, given encryption $E(k_1), \ldots, E(k_t), E(m_1), \ldots, E(m_t)$ of messages $k_1, \ldots, k_t, m_1, \ldots, m_t$, one can efficiently combine the ciphertext of $f(k_1, \ldots, k_t, m_1, \ldots, m_t)$, where $f(\cdot)$ is an efficiently computable function (mainly related to addition or multiplication operation in this article).

As mentioned earlier, a zero-knowledge proof tool [19]–[21] is required to achieve a privacy-enhancing CSC. Specifically, we need to prove three statements without revealing extra information, besides determining the validity of these statements. The first one is provided by creditor to state that ciphertexts are really corresponding to its weights and these weights are all in reasonable range, the second is for bureaus to prove that computed ciphertext is correctly embedded with suitable credit data, and the final is for creditor to prove that the obtained credit score is consistent with the ultimate ciphertext. Here, the reasonable range is to prevent creditor and bureaus from obtaining the other party's private information (i.e., weights or credit data) via providing fake information.

In a typical real-world application, a noninteractive zero-knowledge (NIZK) proof (e.g., [22], [23]) tool is more practical, since we can avoid interactions and this reduces the communication cost. Thus, in this article, we first find an applicable HE scheme (i.e., Paillier encryption scheme [24], [25] with more efficient decryption algorithm) and then propose three NIZK schemes to support the design of a privacy-preserving CSC (PCSC) system.

In this article, we model a PCSC system designed to support the credit system. Unlike prior works that consider the design of risk models, we focus on its privacy protection instead. Specifically, our main contributions are summarized as follows.

1) We propose the first formal description of a PCSC system alongside its security goals (i.e., weight confidentiality and credit confidentiality). This definition can be used in CSC, as well as other computations, such as digital asset settlement.
2) We introduce a concrete construction of PCSC, where the CSC is based on the weight. In the construction, we use Paillier encryption to hide the credit data and weights, and design three NIZK schemes to prove the validity of three statements mentioned earlier.

Having introduced our key contributions above, we will now explain the layout of this article. Related preliminaries are presented in Section II. In Section III, we present the formal description of a PCSC system with its security goals, prior to presenting the concrete construction with security analysis in Section IV. In Section V, we evaluate the performance of our proposal. Finally, we conclude this article in Section VI.

II. Preliminaries

In this section, we present the notations, system model of PCSC, and relevant cryptographic primitives.

The notation λ denotes a security parameter, $\{A_i\}_{i=1}^{t}$ denotes the set $\{A_1, \ldots, A_t\}$, and $y \leftarrow A(x)$ denotes the invoking of algorithm A using x as an input and receiving y as the output. We let $|N|$ denote the length of an integer N, and also denote $X \overset{c}{\approx} Y$ as the two distribution ensembles X and Y, which

Authorized licensed use limited to: University of Texas at San Antonio. Downloaded on March 16,2021 at 12:11:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

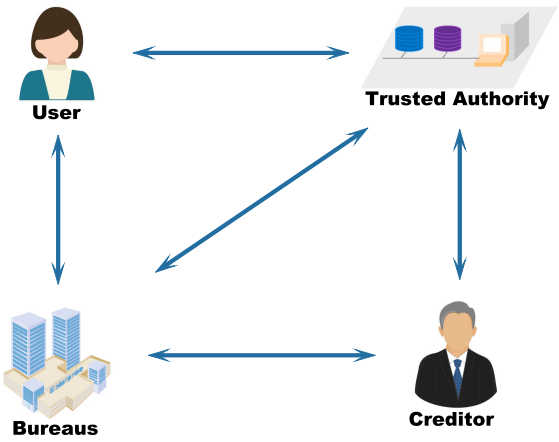


Fig. 2. System model of PCSC systems.

are computationally indistinguishable. A negligible function is denoted as $\text{negl}(\lambda)$, which satisfies that $\forall m > 0, \exists n_0$, s.t., $\forall \lambda > n_0 : \text{negl}(\lambda) < 1/\lambda^m$. A $\mathcal{PPT}$ algorithm is marked as an algorithm with probabilistic polynomial time. The algorithm returns 1 for acceptance and returns 0 for rejection.

A. System Model

We will now briefly introduce the system model, which comprises trusted authority, user, bureaus, and creditor—see also Fig. 2. They communicate with each other through a secure channel that could be established, say using the secure socket layer protocol. We assume trusted authority, bureaus, and creditor have sufficient computing resources to execute the CSC.

1) *Trusted authority:* This component is a trusted third-party responsible for the generation of system parameters (e.g., cryptographic parameters, such as the common reference string (CRS) for the user, bureaus, and creditor to generate/verify NIZK proofs).
2) *User:* This component is the main part of providing credit data via participating in different credit and loan activities. Actually, the user's credit data (including new account creation, account balance/credit card utilization, credit inquiries, payment history, and so forth) are collected by different trusted parties, such as bank, securities company, and asset management company. In our model, we assume that the user's credit data are trusted according to the credit endorsement of these trusted parties.
3) *Bureaus:* This component is responsible for securely collecting, recording, and distributing the user's credit data, as well as cooperating with the creditor to compute the user's credit score. According to general data protection regulation (EU) 2016/679,[2] business processes cannot make user's data available publicly without explicit, informed consent. Thus, it is reasonable to assume that the bureaus will strictly execute the procedure of PCSC, and not directly transmits credit data to the creditor. However, it is also curious and, hence, maybe provide fake credit data for revealing weights of creditor.
4) *Creditor:* This component owns its secret weights for computing the user's credit score via interacting with the bureaus. Here, we assume that it is malicious, in other words, it may lie about the final credit core for saving computation costs or use malicious weights for disclosing credit data of bureaus.

On basic of the aforementioned system model, for protecting the privacy of credit data and weight, the bureaus and creditor are expected to collaborate with each other to compute the final credit score, but without revealing the detail of their data (i.e., credit data and weight). Thus, the design of PCSC system not only requires that the credit data and weight are dealt with under the ciphertext status, but also provides the verifiability of these data. The latter refers to that the credit data (or weights) are not those unreasonable values, such as $(1, 0, 0, \ldots, 0)$, otherwise they will completely reveal the weights (resp., credit data). Accordingly, the bureaus and creditor have to prove their credit data and weights are in the range $[1, \mathcal{L})$, respectively, where there exists a $u, l \in \mathbb{N}$ such that $\mathcal{L} = u^l$.

B. Cryptographic Primitives

1) Pairing: In this article, our construction is based on cryptographic pairings. Thus, for sake of description, we denote $\mathsf{BPG}(\lambda)$ as a pairing generator that takes as input a security parameter λ and returns public parameters of bilinear group $BPP = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$ with the following notations.

1) p is a λ-bit prime number.
2) $\mathbb{G}_1$ and $\mathbb{G}_2$ are additive cyclic groups and $\mathbb{G}_T$ is a multiplicative one, all of which are with order p. G_1 and G_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.
3) $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a nondegenerate bilinear map, and $e(G_1, G_2)$ is one generator of $\mathbb{G}_T$.
4) $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ is a computable isomorphism, that is, $G_1 = \psi(G_2)$.
5) $\forall x, y \in \mathbb{Z}_p, e(xG_1, yG_2) = e(G_1, G_2)^{xy}$.
6) $\forall G \in \mathbb{G}_1, H \in \mathbb{G}_2, e(G, H)$ can be efficiently computed.

2) NIZK Argument and Σ-Protocols: In a noninteractive argument system, there is only a single message sent by a prover P to a verifier V, which is used to prove a relation R. That is, P generates a proof π using an instance/witness pair (x, w), and sends it to V. Then, V verifies (x, π) and returns 1 if valid, or 0 otherwise. A (P, V) is called a noninteractive argument system for R, if it has the properties of completeness and soundness defined below. Moreover, if a noninteractive argument system has also zero-knowledge feature (i.e., leaking no extra information to the verifier, besides the validity of the statement), it will be an NIZK argument system.

Concretely, a triple of $\mathcal{PPT}$ algorithms $(\mathsf{G}, \mathsf{P}, \mathsf{V})$ in the CRS model is called an NIZK argument system for language L_R, if it has the following properties.

1) *Completeness:* For each $\text{crs} \leftarrow \mathsf{G}(\lambda)$ and $(x, w) \in R$, where the crs is generated by a trusted party, there is

$$\Pr[\pi \leftarrow \mathsf{P}(\text{crs}, x, w) : \mathsf{V}(\text{crs}, x, \pi) = 1] = 1 - \text{negl}(\lambda).$$

[2]https://eur-lex.europa.eu/eli/reg/2016/679/oj

Authorized licensed use limited to: University of Texas at San Antonio. Downloaded on March 16,2021 at 12:11:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

2) *Soundness:* For all nonuniform $\mathcal{PPT}$ prover P^*, there is

$$\Pr\left[\begin{array}{c}\text{crs} \leftarrow \mathsf{G}(\lambda), (x,\pi) \leftarrow \mathsf{P}^*(\text{crs}) : \\ x \notin L_R \wedge \mathsf{V}(\text{crs}, x, \pi) = 1\end{array}\right] = \text{negl}(\lambda).$$

3) *Zero knowledge:* There exists a $\mathcal{PPT}$ simulator $S = (S_1, S_2)$, such that for all stateful nonuniform $\mathcal{PPT}$ adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there is

$$\left|\begin{array}{l}\Pr\left[\begin{array}{c}\text{crs} \leftarrow \mathsf{G}(\lambda) \\ (x,w) \leftarrow \mathcal{A}_1(\text{crs}) : \\ \pi \leftarrow \mathsf{P}(\text{crs}, x, w)\end{array}\begin{array}{c}\\ (x,w) \in R \wedge \\ \mathcal{A}_2(\text{crs}, x, \pi) = 1\end{array}\right] \\ -\Pr\left[\begin{array}{c}(\text{crs}, td) \leftarrow S_1(\lambda) \\ (x,w) \leftarrow \mathcal{A}_1(\text{crs}) : \\ \pi \leftarrow S_2(\text{crs}, x, td)\end{array}\begin{array}{c}\\ (x,w) \in R \wedge \\ \mathcal{A}_2(\text{crs}, x, \pi) = 1\end{array}\right]\end{array}\right| = \text{negl}(\lambda).$$

For perfect zero knowledge, the definition is changed so that the aforementioned probability equals 0.

A Σ-protocol is three-move interactive zero-knowledge argument systems that allow a prover P to convince a $\mathcal{PPT}$ verifier V about the validity of a statement $(x, w) \in R$. After the interactive conversation, a form (a, c, z) will be generated, where a and z are computed by P according to the randomly chosen challenge c from V. Here, a Σ-protocol refers to three slightly different properties, namely, *completeness* (i.e., (x, a, c, z) will be accepted if there exist some efficient function ϕ such that $\phi(x, a, c, z) = 1$), *special soundness* (i.e., the witness w can be efficiently recovered using two accepting (x, a, c, z) and $(x, a, c,' z')$ where $c \neq c'$), and *special honest-verifier zero knowledge* (i.e., there exists a simulator S that can output an accepting (x, a, c, z) according to the given c, and the output is indistinguishable from a real interaction between P and V).

Note that a Σ-protocol can be turned into a NIZK argument via Fiat–Shamir heuristic [26] and a secure hash function $\mathcal{H}$. That is, P first computes a and uses the $\mathcal{H}$ to get the challenge $c = \mathcal{H}(x, a)$, then it executes the following Σ-protocol to compute z and sends (x, a, c, z) to V. One can prove the soundness and zero knowledge of the new protocol in the random oracle (RO) mode [27] by replacing $\mathcal{H}(x, a)$ with an RO [26].

3) Boneh–Boyen Signature Scheme: Our constructed NIZK proofs will involve the Boneh–Boyen signature [28], which has been proven secure against a weak chosen-message attack. It consists of the following $\mathcal{PPT}$ algorithms.

1) *BKG*(λ): This algorithm randomly chooses $x \in \mathbb{Z}_p$ to compute $Y = xG_2$. It finally returns the public/private key pair (pk_s, sk_s), where $pk_s = Y$ and $sk_s = x$.
2) *BSign*(sk_s, m): This algorithm computes $S = (x + m)^{-1} G_1$ using its private key $sk_s = x$, and returns the message/signature pair (m, S).
3) *BVerify*(pk_s, m, S): This algorithm uses the public key $pk_s = Y$ to check that $e(S, Y + mG_2) \stackrel{?}{=} e(G_1, G_2)$. It finally returns 1 if the equation holds on; otherwise, it returns 0.

4) Paillier Encryption Scheme: The HE adopted in our PCSC is Paillier encryption [24], [25], which is proven to be secure in the IND-CPA model. It comprises following $\mathcal{PPT}$ algorithms (i.e., PKG, PEnc, PDec, $\mathsf{Extractor}$).

1) *PKG*(λ): This algorithm randomly chooses an admissible RSA modulus $n = p \cdot q$ and sets $d = lcm(p - 1, q - 1)$, then returns the public/private key pair (pk_e, sk_e), where $pk_e = n$ and $sk_e = d$.
2) *PEnc*(pk_e, m): This algorithm randomly chooses $r \leftarrow \mathbb{Z}_n^*$ and computes $c = (1 + n)^m \cdot r^n \mod n^2$, and returns c as the ciphertext of m.
3) *PDec*(sk_e, c): This algorithm computes $m = L(c^d \mod n^2) \cdot \mu \mod n$, where $\mu = [L(1 + n)^d \mod n^2)]^{-1}$ and $L(\cdot)$ is a function defined as $u \in S_n$, $L(u) = (u - 1)/n$, for $S_n = \{u < n^2 | u \equiv 1 \mod n\}$.
4) *PExt*$(sk_e = d, pk_e = n, m, c)$: This algorithm first computes $s = c \cdot (1 + n)^{-m} \mod n^2$, $s' = s \mod n$, and then invokes extended Euclidean algorithm to obtain $(-a, b)$ such that $-ad + bn = 1$. Finally, it returns the randomness in the ciphertext, that is, $r = (s')^{\frac{ad+1}{n}}$.

The Paillier encryption scheme satisfies the homomorphic addition property of plaintexts, that is, $\mathsf{PDec}(sk_e, \mathsf{PEnc}(pk_e, m_1) \cdot \mathsf{PEnc}(pk_e, m_2)) = m_1 + m_2$. This also implies the property of homomorphic scalar multiplication $\mathsf{PDec}(sk_e, (\mathsf{PEnc}(pk_e, m_1))^{m_2}) = m_1 \cdot m_2$.

III. Proposed PCSC System

In this section, we describe the proposed PCSC system and its security goals. For simplicity, we consider a server acts as the creditor where it owns the private parameters (i.e., weights) and the client owns the credit data. They execute the procedure of computing a credit score. Let $f(k_1, \ldots, k_t, m_1, \ldots, m_t)$ be the function of computing a credit score, where $\{k_i\}_{i=1}^t$ is the private parameter owned by the server, $\{m_i\}_{i=1}^t$ is the credit data owned by the client, and t is the number of credit data items.

In a PCSC system, there are three NIZK arguments and a HE scheme, that is, proof of initial weight (PIW) (to prove knowing weights in the ciphertexts and they are in reasonable range), proof of embedded data (PED) (to prove a ciphertext is correctly embedded with the client's credit data and all the credit data are in reasonable range), and proof of final score (PFC) (to prove the final score to be consistent with the final ciphertext). Note that the reasonable range is denotes as $[1, \mathcal{L})$, where there exists a $u, l \in \mathbb{N}$ such that $\mathcal{L} = u^l$.

1) $PP \leftarrow \mathbf{Setup}(\lambda)$: This algorithm is invoked by a trusted party to generate a list of system public parameters. It takes as input a security parameter λ, and returns system public parameters PP.
2) $(pk_e, sk_e, x_0, \pi_0) \leftarrow \mathbf{ServerInital}(PP, \{k_i\}_{i=1}^t)$: This algorithm is invoked by the server to generate the ciphertext of own private weights with a PIW proof. It takes as input PP (system public parameters) and $\{k_i\}_{i=1}^t$ (the server's private weights), it will abort if the input weights are unreasonable (i.e., $\exists k_i \notin [1, \mathcal{L})$ for some $i \in [1, t]$). Otherwise, it generates (pk_e, sk_e) (a public/private key pair in the HE system) and the ciphertext $\{c_i\}_{i=1}^t$ of weights. It also generates an instance $x_0 = (c_1, \ldots, c_t)$ and the corresponding witness $w_0 = (m_1, \ldots, m_t)$, then produces a PIW proof $\pi_0 = P_0(PP, x_0, w_0)$. Finally, it returns (pk_e, sk_e, x_0, π_0).

Authorized licensed use limited to: University of Texas at San Antonio. Downloaded on March 16,2021 at 12:11:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

3) $(x_1, \pi_1) \leftarrow \mathbf{DataEmbed}(PP, \{m_i\}_{i=1}^t, x_0, \pi_0)$: This algorithm is invoked by the client to embed their credit data into the ciphertext of private parameters with a PED proof. It takes as input PP (system public parameters), $\{m_i\}_{i=1}^t$ (client's credit data), $(x_0 = \{c_i\}_{i=1}^t, \pi_0)$ (a PIW instance and proof), and pk (server's public key). It will abort if the range proof π_0 is invalid or the input credit data are unreasonable (i.e., $\exists m_i \notin [1, \mathcal{L})$ for some $i \in [1, t]$); otherwise, it returns a PED proof (x_1, π_1), where $x_1 = (\{c_i\}_{i=1}^t, y, pk_e)$ (here, y is the final embedded ciphertext), $\pi_1 = P_1(x_1, w_1)$, and $w_1 = (m_1, \ldots, m_t)$.
4) $(x_2, \pi_2) \leftarrow \mathbf{ScoreExtract}(PP, sk_e, x_1, \pi_1)$: This algorithm is invoked by the server to decrypt the final credit score and generate an NIZK proof to prove this decryption. It takes as input PP (system public parameters), sk_e (its secret key), and $(x_1 = (\{c_i\}_{i=1}^t, y, pk), \pi_1)$ (a PED instance and proof). It will abort if π_1 is invalid; otherwise, it returns a PFC proof (x_2, π_2), where $x_2 = (m, y, pk)$ (here, m is the final credit score), $\pi_2 = P_2(x_2, w_2)$, and $w_2 = sk_e$.
5) $\{0, 1\} \leftarrow \mathbf{Verify}(PP, x_2, \pi_2)$: This algorithm is invoked by the client to verify the correctness of its credit score. It takes as input PP (system public parameters) and $(x_2 = (m, y, pk_e), \pi_2)$ (a PFC instance and proof), and returns 1 or 0 to show that the credit score is correct or not.

Next, we describe the security goals that a PCSC system should satisfy, including *weight confidentiality* and *credit confidentiality*.

1) Weight Confidentiality: This property guarantees that the procedure of PCSC will not reveal any privacy information of weights to the malicious client. That is, no bounded adversary $\mathcal{A}$ can distinguish between two procedures of computing credit score, where the private weights are assigned by $\mathcal{A}$.

Definition 1: A PCSC system $\Pi =$ (Setup, ServerInital, DataEmbed, ScoreExtract, Verify) satisfies weight confidentiality if no $\mathcal{PPT}$ adversary $\mathcal{A}$ can distinguish the following experiments with a nonnegligible probability:

$\mathrm{EXP}_{\mathrm{wc}}^{\mathrm{b}}(\Pi, \mathcal{A}, \lambda)$

$PP \leftarrow \mathsf{Setup}(\lambda); \mathcal{L}_k \leftarrow \varnothing; \mathcal{L}_m \leftarrow \varnothing$

$(pk, sk, x_0 = (\{c_i\}_{i=1}^t), \pi_0) \leftarrow \mathcal{A}^{\mathsf{ServerInital}(PP, \{k_i\}_{i=1}^t)}$

$(x_1 = (\{c_i\}_{i=1}^t, y, pk_e), \pi_1)$

$\leftarrow \mathcal{A}^{\mathsf{DataEmbed}(PP, \{m_i\}_{i=1}^t, x_0, \pi_0, pk_e)}$

$\mathcal{L}_k \leftarrow \mathcal{L}_k \cup \{\{k_i\}_{i=1}^t\}; \mathcal{L}_m \leftarrow \mathcal{L}_m \cup \{\{m_i\}_{i=1}^t\}$

$(x_2 = (m, y, pk_e), \pi_2) \leftarrow \mathcal{A}^{\mathsf{ScoreExtract}(sk_e, x_1, \pi_1)}$

$(\{k_i^0\}_{i=1}^t, \{k_i^1\}_{i=1}^t) \leftarrow \mathcal{A}(\mathcal{L}_k)$

$(pk_e^*, sk_e^*, x_0^b = (\{c_i^b\}_{i=1}^t), \pi_0^b) \leftarrow \mathsf{ServerInital}(PP, \{k_i^b\}_{i=1}^t)$

$\{\{m_i^*\}_{i=1}^t\} \nsubseteq \mathcal{L}_m; (x_1^* = (\{C_i^b\}_{i=1}^t, y^b, pk_e^*), \pi_1^*) \leftarrow$

$\mathsf{DataEmbed}(PP, \{m_i^*\}_{i=1}^t, x_0^b, \pi_0^b, pk_e^*)$

$(x_2^*, \pi_2^*) \leftarrow \mathsf{ScoreExtract}(PP, sk_e^*, x_1^*, \pi_1^*)$

$b' \leftarrow \mathcal{A}(pk, x_0^b, \pi_0^b, x_1^*, \pi_1^*, x_2^*, \pi_2^*)$.

That is

$$|\Pr[\mathrm{EXP}_{\mathrm{wc}}^0(\Pi, \mathcal{A}, \lambda) = 1] - \Pr[\mathrm{EXP}_{\mathrm{wc}}^1(\Pi, \mathcal{A}, \lambda) = 1]| \leq \mathrm{negl}(\lambda).$$

2) Credit Confidentiality: This property is similar to weight confidentiality, which prevents credit data in the procedure of PCSC from being revealed. It requires that no bounded adversary $\mathcal{B}$ can distinguish two procedures of computing credit score, where the credit data are assigned by $\mathcal{B}$.

Definition 2: A PCSC system $\Pi =$ (Setup, ServerInital, DataEmbed, ScoreExtract, Verify) satisfies credit confidentiality if no $\mathcal{PPT}$ adversary $\mathcal{B}$ can distinguish the following experiments with a nonnegligible probability:

$\mathrm{EXP}_{\mathrm{cc}}^{\mathrm{b}}(\Pi, \mathcal{B}, \lambda)$

$PP \leftarrow \mathsf{Setup}(\lambda); \mathcal{L}_k \leftarrow \varnothing; \mathcal{L}_m \leftarrow \varnothing$

$(pk, sk, x_0 = (\{c_i\}_{i=1}^t), \pi_0) \leftarrow \mathcal{B}^{\mathsf{ServerInital}(PP, \{k_i\}_{i=1}^t)}$

$(x_1 = (\{c_i\}_{i=1}^t, y, pk), \pi_1) \leftarrow \mathcal{B}^{\mathsf{DataEmbed}(PP, \{m_i\}_{i=1}^t, x_0, \pi_0, pk)}$

$\mathcal{L}_k \leftarrow \mathcal{L}_k \cup \{\{k_i\}_{i=1}^t\}; \mathcal{L}_m \leftarrow \mathcal{L}_m \cup \{\{m_i\}_{i=1}^t\}$

$(x_2 = (m, y, pk), \pi_2) \leftarrow \mathcal{B}^{\mathsf{ScoreExtract}(sk, x_1, \pi_1)}$

$\{\{k_i^*\}_{i=1}^t\} \nsubseteq \mathcal{L}_k$

$(pk, sk, x_0^* = (\{c_i^*\}_{i=1}^t), \pi_0^*) \leftarrow \mathsf{ServerInital}(PP, \{k_i^*\}_{i=1}^t)$

$(\{m_i^0\}_{i=1}^t, \{m_i^1\}_{i=1}^t) \leftarrow \mathcal{B}(\mathcal{L}_m);$

$(x_1^b = (\{c_i^*\}_{i=1}^t, y^b, pk), \pi_1^b) \leftarrow$

$\mathsf{DataEmbed}(PP, \{m_i^b\}_{i=1}^t, x_0^*, \pi_0^*, pk)$

$(x_2^b, \pi_2^b) \leftarrow \mathsf{ScoreExtract}(PP, sk, x_1^b, \pi_1^b)$

$b' \leftarrow \mathcal{B}(pk, x_0^*, \pi_0^*, x_1^b, \pi_1^b, x_2^b, \pi_2^b)$.

That is

$$|\Pr[\mathrm{EXP}_{\mathrm{cc}}^0(\Pi, \mathcal{B}, \lambda) = 1] - \Pr[\mathrm{EXP}_{\mathrm{cc}}^1(\Pi, \mathcal{B}, \lambda) = 1]| \leq \mathrm{negl}(\lambda).$$

IV. Our Construction

In this section, we propose a concrete construction of PCSC, where the CSC is based on a weight computing, namely, the goal function is $f(k_1, \ldots, k_t, m_1, \ldots, m_t) = \Sigma_{i=1}^t (k_i \cdot m_i)$. Specifically, the proposed construction is based on Paillier encryption scheme and three designed NIZK schemes based on Camenisch *et al.*'s range proofs [29] (i.e., PIW, PED, and PFC). Prior to presenting the construction of our PCSC, we first introduce the design of PIW, PED, and PFC. It is worth noted that other HE schemes and range proof protocols can also be adopted in our PCSC system. Our choice of Paillier encryption scheme is due to its more efficient decryption than other HE schemes. Correspondingly, Camenisch *et al.*'s range proof protocol can be conveniently integrated with Σ protocols for supporting proof of Paillier ciphertexts.

Authorized licensed use limited to: University of Texas at San Antonio. Downloaded on March 16,2021 at 12:11:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

A. Design of NIZKs

The PIW is designed for proving the statement $\{(x_0 = (c_1, \ldots, c_t), w_0 = (k_1, \ldots, k_t)) : c_i = \mathsf{PEnc}(pk_e, k_i) \wedge k_i \in [1, \mathcal{L})\}$, the PED is for $\{(x_1 = (c_1, \ldots, c_t, y, pk_e), w_1 = (m_1, \ldots, m_t)) : y = \Pi_{i=1}^{t}(c_i^{m_i}) \wedge m_i \in [1, \mathcal{L})\}$, and the PFC is for $\{(x_2 = (m, y, pk_e), w_2 = sk_e) : y = \mathsf{PEnc}(pk_e, m)\}$, where $c_1, \ldots, c_t$ are Paillier ciphertexts and (pk_e, sk_e) is the public/private key pair in Paillier encryption system (i.e., $pk_e = n, sk_e = d$). In this article, all of them are constructed by applying Fiat–Shamir heuristic and a secure hash function $\mathcal{H}$ to a Σ-protocol, which can achieve perfect zero knowledge in the standard model, though the soundness of our constructions is proved in the RO model.

Notably, in PIW and PED, we also require proving the range of k_i and m_i, thus we utilize the range proof proposed in [29]. That is, to realize the range proof of $k_i \in [1, u^l) \ \ \forall i \in [1, t]$ (or $m_i \in [1, u^l)$), we need to prove $k_i \in [0, u^l)$ and $k_i - 1 \in [0, u^l)$ (resp., $m_i \in [0, u^l)$ and $m_i - 1 \in [0, u^l)$). Specifically, we write these weights (resp., credit data) in u-ary notation $k_i = \Sigma_{j=0}^{l-1}(k_{i,j} \cdot u^j)$ (resp., $m_i = \Sigma_{j=0}^{l-1}(m_{i,j} \cdot u^j)$) and show that each coefficient $k_{i,j}$ (resp., $m_{i,j}$) is in the range $[0, u)$.

1) Proof of Initial Weight: The designed PIW consists of following three algorithms (i.e., PIW-Gen, PIW-GenProof, and PIW-VerProof). Here, we adopt an RO $\mathcal{H}$ that can be instantiated by a secure hash function mapping any string into a l-bit one, where 2^l is smaller than any of the prime factors of n (the public key in the Paillier encryption scheme).

1) *PIW-Gen:* This algorithm is invoked by a trusted party to generate a CRS and related private key. It takes as input a secure parameter λ, and invokes BPG and BKG to obtain $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$ and $(pk_s = xG_2, sk_s = x)$, respectively. Then, it invokes BSign to sign each element $i \in [0, u)$ and obtains $S_i = (x + i)^{-1}G_1$. Finally, it returns $crs_0 = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2, pk_s, \{S_i\}_{i=0}^{l-1}, \mathcal{H})$ and a private key sk_s.
2) *PIW-GenProof:* This algorithm takes as input crs_0, $x_0 = (n, c_1, \ldots, c_t)$ (where $c_i = (1+n)^{\Sigma_{j=0}^{l-1}(k_{i,j}u^j)} r_i^n \bmod n^2$), $w_0 = (k_1, \ldots, k_t)$ (where $k_i = \Sigma_{j=0}^{l-1}(k_{i,j}u^j)$). For all $i \in [1, t], j \in [0, l)$, it first randomly chooses $v_{i,j} \in \mathbb{Z}_p$ to compute $V_{i,j} = v_{i,j} S_{k_{i,j}}$. Then, it computes $n_p = n * p$ and randomly chooses $\alpha_{i,j}, \ \beta_{i,j} \in \mathbb{Z}_{n_p}, \ \gamma_i \in \mathbb{Z}_n$ to compute $a_{i,j} = e(V_{i,j}, \ G_2)^{-\alpha_{i,j}} \cdot e(G_1, G_2)^{\beta_{i,j}}$, $\chi_i = (1+n)^{\Sigma_{j=0}^{l-1}(\alpha_{i,j}u^j)} \cdot \gamma_i^n \bmod n^2$, $\chi_i' = (1+n)^{-1} \cdot (1+n)^{\Sigma_{j=0}^{l-1}(\alpha_{i,j}u^j)} \cdot \gamma_i^n \bmod n^2$, $h_0 = \mathcal{H}(x_0||V_{1,0}|| \ldots ||V_{t,l-1}||a_{1,0}|| \ldots ||a_{t,l-1}||\chi_1|| \ldots ||\chi_t|| \ \chi_1'|| \ldots ||\chi_t')$. Next, it computes $z_{k_{i,j}} = \alpha_{i,j} - k_{i,j} \cdot h_0 \bmod n_p$, $z_{v_{i,j}} = \beta_{i,j} - v_{i,j} \cdot h_0 \bmod n_p$, $z_{r_i} = \gamma_i \cdot r_i^{-h_0} \bmod n$. Finally, it returns $x_0 = (n, c_1, \ldots, c_t)$, $\pi_0 = (h_0, \ V_{i,j}, a_{i,j}, \chi_i, \chi_i,' \ z_{k_{i,j}}, z_{v_{i,j}}, z_{r_i}) \ \ \forall i \in [1, t] \ \ \forall j \in [0, l)$.
3) *PIW-VerProof:* This algorithm takes as input $(\text{crs}_0, x_0, \pi_0)$, and returns 1 or 0 to show this proof valid or not. It first chooses $x_0 = (n, \ c_1, \ldots, c_t)$, $\pi_0 = (h_0, \ V_{i,j}, \ a_{i,j}, \ \chi_i, \ \chi_i,' \ z_{k_{i,j}}, \ z_{v_{i,j}}, \ z_{r_i})$ (where $\forall i \in [1, t] \ \ \forall j \in [0, l)$), then computes $h_0' = \mathcal{H}(x_0||V_{1,0}|| \ldots ||V_{t,l-1}||a_{1,0}|| \ldots ||a_{t,l-1}||\chi_1|| \ldots ||\chi_t|| \ \chi_1'|| \ldots ||\chi_t')$. If $h_0' \neq h_0$, then it returns 0; otherwise, it verifies $\chi_i \stackrel{?}{=} c_i^{h_0'} \cdot (1+n)^{\Sigma_{j=0}^{l-1}(z_{k_{i,j}}u^j)} \cdot z_{r_i}^n \bmod n^2$, $\chi_i' \stackrel{?}{=} c_i^{h_0'} \cdot (1+n)^{-1} \cdot (1+n)^{\Sigma_{j=0}^{l-1}(z_{k_{i,j}}u^j)} \cdot z_{r_i}^n \bmod n^2$ and $a_{i,j} \stackrel{?}{=} e(V_{i,j}, y)^{h_0'} \cdot e(G_1, G_2)^{z_{v_{i,j}}} \cdot e(V_{i,j}, G_2)^{-z_{k_{i,j}}}$. It finally returns 1 if all these equations hold on, and 0 otherwise.

2) Proof of Embedded Data: To construct a PED for our PCSC, we also adopt a secure hash function $\mathcal{H}$ as mentioned earlier. The constructed PED is described as the following algorithms.

1) *PED-Gen:* This algorithm is consistent to that in PIW, that is, it takes as input a secure parameter λ and returns a reference string $\text{crs}_1 = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2, pk_s, \{S_i\}_{i=0}^{l-1}, \mathcal{H})$ and a private key sk_s.
2) *PED-GenProof:* This algorithm takes as input crs_1, $x_1 = (c_1, \ldots, c_t, \ y, n)$, $w_1 = (m_1, \ldots, m_t)$ (where $m_i = \Sigma_{j=1}^{l-1}(m_{i,j}u^j)$, $y = \Pi_{i=1}^{t} c_i^{m_i}$) and returns a PED proof π_1. For all $i \in [1, t], \ j \in [0, l)$, it first randomly chooses $v_{i,j} \in \mathbb{Z}_p$ to compute $V_{i,j} = v_{i,j} S_{m_{i,j}}$. Then, it computes $n_p = n * p$ and randomly chooses $\alpha_{i,j}, \ \beta_{i,j} \in \mathbb{Z}_p$, to compute $a_{i,j} = e(V_{i,j}, G_2)^{-\alpha_{i,j}} \cdot e(G_1, G_2)^{\beta_{i,j}}$, $\chi = \Pi_{i=1}^{t}(c_i^{\Sigma_{j=0}^{l-1}(\alpha_{i,j}u^j)}) \bmod n^2$, $\chi' = \Pi_{i=1}^{t}(c_i^{-1}) \cdot \Pi_{i=1}^{t}(c_i^{\Sigma_{j=0}^{l-1}(\alpha_{i,j}u^j)}) \bmod n^2$, $h_1 = \mathcal{H}(x_1||V_{1,0}|| \ldots ||V_{t,l-1}||a_{1,0}|| \ldots ||a_{t,l-1}||\chi||\chi')$. Next, it computes $z_{m_{i,j}} = \alpha_{i,j} - m_{i,j} \cdot h_1 \bmod n_p$, $z_{v_{i,j}} = \beta_{i,j} - v_{i,j} \cdot h_1 \bmod n_p$. Finally, it returns $(x_1 = (c_1, \ldots, c_t, \ y, n), \ \pi_1 = (h_1, \ V_{i,j}, \ a_{i,j}, \chi, \chi,' \ z_{m_{i,j}}, \ z_{v_{i,j}})) \ \ \forall i \in [1, t] \ \ \forall j \in [0, l)$.
3) *PED-VerProof:* This algorithm takes as input $(\text{crs}_1, x_1, \pi_1)$, and returns 1 or 0 to show this proof is valid or not. It first chooses $x_1 = (c_1, \ldots, c_t, y, n), \pi_1 = (h_1, V_{i,j}, a_{i,j}, \chi, \chi,' z_{m_{i,j}}, z_{v_{i,j}})$ (where $\forall i \in [1, t] \ \ \forall j \in [0, l)$), then computes $h_1' = \mathcal{H}(x_1||V_{1,0}|| \ldots ||V_{t,l-1}||a_{1,0}|| \ldots ||a_{t,l-1}||\chi||\chi')$. If $h_1' \neq h_1$, then it returns 0; otherwise, it verifies $\chi \stackrel{?}{=} \Pi_{i=0}^{t}(c_i^{\Sigma_{j=0}^{l-1}(z_{m_{i,j}}u^j)}) \cdot y^{h_1'} \bmod n^2$, $\chi' \stackrel{?}{=} \Pi_{i=1}^{t}(c_i^{-1}) \cdot \Pi_{i=0}^{t}(c_i^{\Sigma_{j=0}^{l-1}(z_{m_{i,j}}u^j)}) \cdot y^{h_1'} \bmod n^2$ and $a_{i,j} \stackrel{?}{=} e(V_{i,j}, y)^{h_1'} \cdot e(G_1, G_2)^{z_{v_{i,j}}} \cdot e(V_{i,j}, G_2)^{-z_{m_{i,j}}}$. It finally returns 1 if these equations hold on, and 0 otherwise.

3) Proof of Final Score: To construct a PFC for our PCSC, we also consider a secure hash function $\mathcal{H}$ as mentioned above in our design. Here, the PFC consists of following three algorithms (i.e., PFC-Gen, PFC-Gen, and PFC-Gen).

1) *PFC-Gen:* This algorithm is invoked by a trusted party to generate a CRS. It takes input a secure parameter λ and generates a CRS $\text{crs} = \mathscr{D}$, where $\mathscr{D}$ is the description of the aforementioned hash function (i.e., $\mathcal{H}$).
2) *PFC-GenProof:* This algorithm takes as input $\mathscr{D} = \mathcal{H}, x_2 = (m, y, n), w_2 = d$ and returns a PFC proof π_2. It first invokes $\mathsf{PExt}(d, n, m, y)$ to extract the randomness

Authorized licensed use limited to: University of Texas at San Antonio. Downloaded on March 16,2021 at 12:11:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

r in the ciphertext y. Then, it randomly chooses $r_a \in \mathbb{Z}_n^*$, and computes $R = r_a^n \bmod n^2$, $h_2 = \mathcal{H}(x_2||R)$ and $z = r_a \cdot r^{h_2} \bmod n$. Finally, it returns $(x_2 = (m, y, n), \pi_2 = (h_2, R, z))$.

3) *PFC-VerProof:* This algorithm takes as input $(\mathscr{D} = \mathcal{H}, x_2, \pi_2)$ and returns 1 or 0 to show this proof is valid or not. It first parses (x_2, π_2) as $(x_2 = (m, y, n), \pi_2 = (h_2, R, z))$ and computes $h_2' = \mathcal{H}(x_2||R)$. If $h_2' \neq h_2$, it returns 0; otherwise, it verifies the equation $z^n = R \cdot [y/(1+n)^m]^{h_2'} \bmod n^2$ holds or not. If not, this algorithm returns 0; otherwise, it returns 1.

Theorem 1: The aforementioned proofs are combined by Σ protocols and Fiat–Shamir heuristic theorem, and they can be proven as NIZK arguments (i.e., satisfying properties of completeness, soundness, and zero knowledge) in the RO model. These properties can be trivially proved and interested readers can refer to the work in [20], [29], and [30] for the details.

B. Proposed PCSC

In this section, we will present the construction of our PCSC. Note that our PCSC can concurrently support a server to compute multiple clients' credit scores. For sake of simplicity, here we only describe a server Alice and a client Bob in the PCSC. The detail is described as follows upon the aforementioned system building blocks (the Paillier encryption scheme, PIW, PED, and PFC).

1) *Setup:* A trusted party invokes PIW-Gen, PED-Gen, and PFC-Gen to generate a CRS and private key denoted as $\text{crs} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2, pk_s, \{S_i\}_{i=0}^{l-1}, \mathcal{H})$ and sk_s, respectively. Finally, both Alice and Bob obtain the CRS crs for the execution of PCSC.
2) *ServerInital:* Alice first invokes the PKG algorithm to obtain its public/private key pair, that is, $(pk_e, sk_e) \leftarrow \mathsf{PKG}(\lambda)$, where $pk_e = n, sk_e = d$. Then, it encrypts its private parameters $(\{k_i\}_{i=1}^t)$ via $c_i = \mathsf{PEnc}(pk_e, k_i) = (1+n)^{k_i} \cdot r_i^n \bmod n^2$ for $i = 1, \ldots, t$. Next, it invokes PIW-GenProof to obtain a PIW proof $x_0 = (n, c_1, \ldots, c_t), \pi_0 = (h_0, V_{i,j}, \chi_i, \chi_i{}', z_{k_{i,j}}, z_{v_{i,j}}, z_{r_i})$, where $\forall i \in [1, t] \quad \forall j \in [0, l)$. Finally, it sends the (x_0, π_0) to Bob.
3) *DataEmbed:* After receiving (x_0, π_0), Bob first invokes PIW-VerProof to check its validation. If not, it aborts; otherwise, it parses $x_0 = (n, c_1, \ldots, c_t)$ and combines its credit data $\{m_i\}_{i=1}^t$ to compute $y = \Pi_{i=1}^t (c_i^{m_i}) \bmod n^2$. In addition, it invokes PED-GenProof to compute the PED, namely, $x_1 = (c_1, \ldots, c_t, y, n), \pi_1 = (h_1, V_{i,j}, \chi, \chi,{}' z_{m_{i,j}}, z_{v_{i,j}})$, where $\forall i \in [1, t] \quad \forall j \in [0, l)$. Finally, it replies (x_1, π_1) to Alice.
4) *ScoreExtract:* To obtain the final credit score of Bob, Alice first verifies the received PED proof from Bob, that is, it invokes PED-VerProof(crs, x_1, π_1). It aborts if this invocation returns 0; otherwise, it parses $x_1 = (c_1, \ldots, c_t, y, n)$ and decrypts the ciphertext y to obtain the score $m \leftarrow \mathsf{PDec}(sk_e, y)$. Moreover, it uses PFC-GenProof to compute the PFC proof (x_2, π_2), where $x_2 = (m, y, n), \pi_2 = (h_2, R, z)$. Finally, it replies (x_2, π_2) to Bob.
5) *Verify:* After receiving the (x_2, π_2) from Alice, Bob can verify the correctness of its credit score via PFC-VerProof(crs, x_2, π_2). If the returned result is 1, meaning that the credit score is correct and this algorithm returns 1; otherwise, it returns 0.

Theorem 2: The proposed PCSC system $\Pi =$ (Setup, ServerInital, DataEmbed, ScoreExtract, Verify) is secure (i.e., with *weight confidentiality* and *credit confidentiality*).

Proof: To prove the weight confidentiality, we now describe the following hybrid experiments $(\text{EXP}_{\text{wc}}^0, \text{E}_1, \text{E}_2, \text{EXP}_{\text{wc}}^1)$.

1) *Experiment:* E_1. The experiment E_1 modifies EXP_{wc}^0 by using the simulator $S1$ (see Algorithm 1) to produce the PIW proof. More formally, in the ServerInital, the challenger invokes PKG to generate a public/private key pair (n, d) and randomly chooses ciphertext $c_1, \ldots, c_t$. Then, the challenger invokes $S1(\lambda)$ to generate the PIW proof $(x_0^0, \pi_0^{0'})$ instead of invoking *PIW-GenProof*. The rest procedure of E_1 is the same as that of EXP_{wc}^0. For the PIW scheme with special honest verifier zero knowledge, the distribution of the simulated proofs is identical to that in EXP_{wc}^0, we have
$$|\Pr[\text{EXP}_{\text{wc}}^0(\Pi, \mathcal{A}, \lambda) = 1] - \Pr[\text{E}_1(\Pi, \mathcal{A}, \lambda) = 1]| \leq \text{negl}(\lambda).$$
2) *Experiment:* E_2. The experiment E_2 modifies E_1 during invoking ServerInital. The challenger simulates an instance x_0^1 as that in EXP_{wc}^1 by computing $(x_0^1, \pi_0^{1'}) \leftarrow S1(\lambda)$. Also, due to the special honest verifier zero knowledge of proposed PIW scheme, we have $\{x_0^0, \pi_0^{0'}\} \overset{c}{\approx} \{x_0^1, \pi_0^{1'}\}$. Hence
$$|\Pr[\text{E}_1(\Pi, \mathcal{A}, \lambda) = 1] - \Pr[\text{E}_2(\Pi, \mathcal{A}, \lambda) = 1]| \leq \text{negl}(\lambda).$$
Moreover, it is obvious to see the following relation according to the zero-knowledge property:
$$|\Pr[\text{E}_2(\Pi, \mathcal{A}, \lambda) = 1] - \Pr[\text{EXP}_{\text{wc}}^1(\Pi, \mathcal{A}, \lambda) = 1]| \leq \text{negl}(\lambda).$$
From the aforementioned discussion, we can sum over the final probability that represents the adversary $\mathcal{A}$'s advantage in distinguishing the $(\text{EXP}_{\text{wc}}^0, \text{E}_1, \text{E}_2, \text{EXP}_{\text{wc}}^1)$. That is
$$|\Pr[\text{EXP}_{\text{wc}}^0(\Pi, \mathcal{A}, \lambda) = 1] - \Pr[\text{EXP}_{\text{wc}}^1(\Pi, \mathcal{A}, \lambda) = 1]| \leq \text{negl}(\lambda).$$

Our proposed PCSC also satisfies the goal of credit confidentiality. The proof is similar to that of weight confidentiality (via analyzing hybrid experiments $\text{EXP}_{\text{cc}}^0, \text{E}_1{}', \text{E}_2{}', \text{EXP}_{\text{cc}}^1$), but the modification is on the PED proof but not PIW proof. Specifically, we replace PED-GenProof with $S2$ (see Algorithm 2) to generate the PED proof, and hence construct two experiments $\text{E}_1{}'$ and $\text{E}_2{}'$. The former uses $S2$ to generate the PED proof $(x_1^0, \pi_1^{0'})$, but the latter $(x_1^1, \pi_1^{1'})$. Due to the special honest verifier zero knowledge of PED scheme, we have the indistinguishability between EXP_{cc}^0 and $\text{E}_1{}'$, then $\text{E}_1{}'$ and $\text{E}_2{}'$, and finally $\text{E}_2{}'$ and

Authorized licensed use limited to: University of Texas at San Antonio. Downloaded on March 16,2021 at 12:11:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Algorithm 1: Simulator $S1$.

Input: a security parameter λ

Output: public parameters PP, the proof π_0

1: $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2, pk_s, \mathcal{O}_H) = \mathsf{Sim1}(\lambda)$[3]

2: Given an instance $x_0 = (n, c_1, \ldots, c_t)$ to be proved.

3: Randomly choose $h_0^* \leftarrow \{0,1\}^l$ (i.e. a l-bit string), $v_{i,j}^* \in \mathbb{Z}_p$ and $z_{k_{i,j}}^* \in \mathbb{Z}_p$, $z_{v_{i,j}}^* \in \mathbb{Z}_p$, $z_{r_i}^* \in \mathbb{Z}_n$.

4: Compute $V_{i,j}^* = v_{i,j}^* S_{k_{i,j}}$, $\chi_i^* = c_i^{h_0^*} \cdot (1+n)^{\Sigma_{j=0}^{l-1}(z_{k_{i,j}^*} u^j)} \cdot z_{r_i}^{*n} \bmod n^2$, $\chi_i'^* = c_i^{h_0^*} \cdot (1+n)^{-1} \cdot (1+n)^{\Sigma_{j=0}^{l-1}(z_{k_{i,j}^*} u^j)} \cdot z_{r_i}^{*n} \bmod n^2$ and $a_{i,j}^* = e(V_{i,j}, y)^{h_0^*} \cdot e(G_1, G_2)^{z_{v_{i,j}^*}} \cdot e(V_{i,j}, G_2)^{-z_{k_{i,j}^*}}$.

5: Set $\mathcal{O}_H(V_{1,0}^* || \ldots || V_{t,l-1}^* || a_{1,0}^* || \ldots || a_{t,l-1}^* || \chi_1^* || \ldots || \chi_t^* || \chi_1'^* || \ldots || \chi_t'^*) = h_0^*$.

6: return $\pi_1^* = (h_0^*, V_{i,j}^*, \chi_i^*, \chi_i'^*, z_{k_{i,j}}^*, z_{v_{i,j}}^*, z_{r_i}^*)$

Algorithm 2: Simulator $S2$.

Input: a security parameter λ

Output: public parameters PP, the proof π_1

1: $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2, pk_s, \mathcal{O}_H) = \mathsf{Sim2}(\lambda)$[4]

2: Given an instance $x_1 = (c_1, \ldots, c_t, y, n)$ to be proved.

3: Randomly choose $h_1^* \leftarrow \{0,1\}^l$ (i.e. a l-bit string), $v_{i,j}^* \in \mathbb{Z}_p$ and $z_{m_{i,j}}^* \in \mathbb{Z}_p$, $z_{v_{i,j}}^* \in \mathbb{Z}_p$, $z_{r_i}^* \in \mathbb{Z}_n$.

4: Compute $V_{i,j}^* = v_{i,j}^* S_{m_{i,j}}$, $\chi^* = \Pi_{i=0}^t (c_i^{\Sigma_{j=0}^{l-1}(z_{m_{i,j}}^* u^j)}) \cdot y^{h_1^*}$, $\chi'^* = \Pi_{i=1}^t (c_i^{-1}) \cdot \Pi_{i=0}^t (c_i^{\Sigma_{j=0}^{l-1}(z_{m_{i,j}}^* u^j)}) \cdot y^{h_1^*}$ and $a_{i,j}^* = e(V_{i,j}^*, y)^{h_1^*} \cdot e(G_1, G_2)^{z_{v_{i,j}}^*} \cdot e(V_{i,j}^*, G_2)^{-z_{m_{i,j}}^*}$.

5: Set $\mathcal{O}_H(V_{1,0}^* || \ldots || V_{t,l-1}^* || a_{1,0}^* || \ldots || a_{t,l-1}^* || \chi^* || \chi'^*) = h_1^*$.

6: return $\pi_0^* = (h_1^*, V_{i,j}^*, \chi^*, \chi'^*, z_{m_{i,j}}^*, z_{v_{i,j}}^*)$

$\mathrm{EXP}_{\mathrm{cc}}^1$ against a $\mathcal{PPT}$ adversary $\mathcal{B}$. That is

$$|\Pr \mathrm{EXP}_{\mathrm{cc}}^0(\Pi, \mathcal{B}, \lambda) = 1] - \Pr[\mathrm{EXP}_{\mathrm{cc}}^1(\Pi, \mathcal{B}, \lambda) = 1]| \leq \mathrm{negl}(\lambda).$$

V. Performance Analysis

To our best of knowledge, our proposed PCSC is the first one to deal with the privacy protection of credit score systems. Thus, to demonstrate its feasibility, we directly analyze the storage and communication overhead in our proposal, together with estimating its computation overhead in a proof-of-concept implementation. Here, we denote t as the number of credit data items.

[3]Sim1 is similar to the PIW-Gen except that the hash function is simulated as an RO.

[4]Sim2 is similar to the PED-Gen except that the hash function is simulated as an RO.

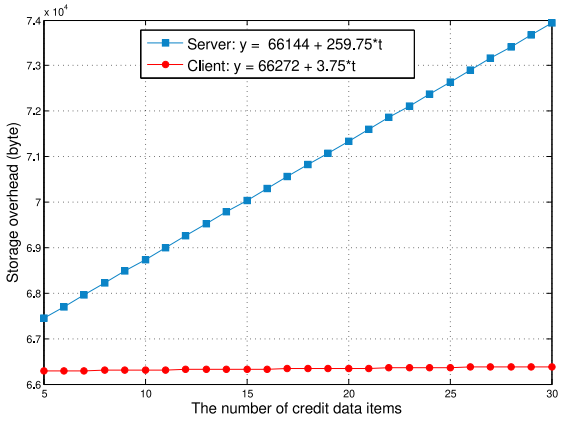


Fig. 3. Storage overhead of each participant in our PCSC system.

TABLE I
Communication Overhead

Phase	Theory	Practice (bytes)
Setup	$\lvert\mathbb{Z}_p\rvert + 1025\lvert\mathbb{G}_1\rvert + 2\lvert\mathbb{G}_2\rvert$	65888
ServerInital	$(t+1)\lvert\mathbb{Z}_n\rvert + 2t\lvert\mathbb{Z}_{n^2}\rvert$ $+\lvert\mathbb{Z}_{2^l}\rvert + tl\lvert\mathbb{G}_1\rvert$ $+ tl\lvert\mathbb{G}_T\rvert + 2tl\lvert\mathbb{Z}_{n^p}\rvert$	$2944t + 160$
DataEmbed	$2\lvert\mathbb{Z}_{n^2}\rvert+\lvert\mathbb{Z}_{2^l}\rvert + tl\lvert\mathbb{G}_1\rvert$ $+ tl\lvert\mathbb{G}_T\rvert + 2tl\lvert\mathbb{Z}_{n_p}\rvert$	$2424t + 160$
ScoreExtract	$\lvert\mathbb{Z}_{2^l}\rvert + \lvert\mathbb{Z}_{n^2}\rvert + \lvert\mathbb{Z}_n\rvert$	416
Verify	$\lvert\mathbb{Z}_{2^l}\rvert + \lvert\mathbb{Z}_{n^2}\rvert + \lvert\mathbb{Z}_n\rvert$	416

Let the system security parameter $\lambda = 80$ and, hence, we adopt the public key of Paillier encryption system $|n| = 1024$ b and the Barreto–Naehrig [31] over base field $\mathbb{F}_{256}$ to achieve this security level. This means that the elements in $\mathbb{Z}_p, \mathbb{Z}_n, \mathbb{Z}_{n^2}$, $\mathbb{Z}_{2^l}$, $\mathbb{Z}_{n_p}$, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ can be represented in 32, 128, 256, 32, 160, 64, 128, and 384 B, respectively, where 2^l is smaller than the prime factor of n. Since we choose the SHA256 to instantiate our argument, the length of a hash element is 32 B. The length of each credit data and weight is 30 b, meaning that their space is $[0, 2^{30})$ (which is enough for the practical credit data or weight). Note that the aforementioned $\mathcal{L} = u^l$ is set as $u = 2^{10}$ and $l = 3$ in our implementation.

A. Storage

In our system, both the server and the client need to store the public parameters and signatures (i.e., $|\mathbb{Z}_p| + 1025|\mathbb{G}_1| + 2|\mathbb{G}_2|$, about 65 888 B) generated in the Setup phase, together with the public key of the HE system (about 128 B an element in $\mathbb{Z}_n$). Here, the server has to store an additional secret key (also 128 B in $\mathbb{Z}_n$). Moreover, the server and client need to store the weight and credit data (which are both about $30t$ b), respectively. To verify the PED proof, the sever needs to store its generated ciphertexts $(c_1, \ldots, c_t)$, and the client requires another storage 256 B (i.e., the storage of y) for the verification of PFC proof. In all, the storage overheads of the server and client are $66144 + 259.75t$ and $66\,272 + 3.75t$ B, respectively.

For a further intuitive analysis, we also discuss the relationship between the storage overhead and the number of credit data items. As shown in Fig. 3, although both the storage overheads of the server and client are linear to the number of credit data items,

Authorized licensed use limited to: University of Texas at San Antonio. Downloaded on March 16,2021 at 12:11:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE II
TIME COST OF OUR PCSC SYSTEM (ms)

Algo. \ t	5	10	15	20	25	30
Setup	4480.09	4536.31	4529.85	4534.23	4588.71	4590.84
SeverInital	3188.27	5937.86	8950.22	12637.8	15433.1	18587
DataEmbed	4803.52	9584.87	14765.2	20408.7	25844.1	30748.5
ScoreExtract	2729.81	5287.96	8125.67	10783.5	13198.1	15908.5
Verify	168.93	166.69	170.64	170.16	170.84	171.39

the rate of storage overhead increase of the client is less than that of the server. Specifically, the storage overhead of the client is approximately constant (not more than 67 kB) comparing to that of the server. This result seems not ideal enough from the view of the sever, however, some real-world applications are with few terms (e.g., Fair Issaac Corporation is only based on five factors as mentioned above) and could be supported by our proposal. In addition, the server in our model generally refers to some participants with enough resources, including storage and computation and, hence, can tolerate this storage overhead.

B. Communication

This section discusses the communication overhead for the operations of our proposed PCSC system. First, in the Setup phase, the TA needs to share public parameters and signatures with server and client, which is of length $|\mathbb{Z}_p| + 1025|\mathbb{G}_1| + 2|\mathbb{G}_2|$ (about 65 888 B). To generate a credit score based on t items of credit data, the server should produce the ciphertext of its own weights (i.e., $\{c_i\}_{i=1}^t$) together with a PIW proof. This is executed in the ServerInital algorithm involving length of $(t+1)|\mathbb{Z}_n| + 2t|\mathbb{Z}_{n^2}| + |\mathbb{Z}_{2^l}| + tl|\mathbb{G}_1| + tl|\mathbb{G}_T| + 2tl|\mathbb{Z}_{n^p}|$ (about $2944t + 160$ B). In addition, after the client receives these (x_0, π_0), it will invoke the DataEmbed algorithm (i.e., it embeds its credit data $\{m_i\}_{i=1}^t$ into $\{c_i\}_{i=1}^t$ and then generates a PED proof to reply the server). This transmission is with the overhead of $2424t + 160$ B, that is, the length of (y, π_1). Note that the sever has stored its ciphertext $\{c_i\}_{i=1}^t$, and the client does not need to send them again. The additional communication overhead is caused in ScoreExtract algorithm (requiring about 416 B that the client replies (m, h_2, R, z) to the server).

The result is summarized in Table I, from which we can clearly learn the relationship between the communication overhead and the number of credit data items (i.e., t). That is, both ServerInital and DataEmbed are with a dramatically linear increase rate, but the other three Setup, ScoreExtract, and Verify are constant (i.e., 65 888, 416, and 416 B, respectively). As discussed in Section V-A, these results are also acceptable in the real-world applications.

C. Computation

We performed a proof-concept implementation of our PCSC system to evaluate its computation overhead. The system configuration is the Window system (Windows 7, 64 b) with an Intel (R) Core (TM) i7-6700 CPU at 3.40 GHz and 8-GB RAM. Note that in the experimental evaluation, we considered the number of credit data items from 5 to 30 with an interval 5 (i.e., $t = 5$,10,15,20,25,30) to test the time costs of each algorithm. The time costs of each algorithm in our PCSC system are shown in Table II. One can find that the time costs of each algorithm are all linear increasing to the number of credit data items except that of Setup and Verify. The most expensive time cost is about 30 s even the number of credit data items reaches to 30. On the one hand, computing a secure and verifiable credit score does not have to be real-time, and hence these time costs are still tolerable for achieving a stronger security. On the other hand, we can seek for more efficient HE algorithms and designing corresponding NIZK proofs to improve the performance.

VI. CONCLUSION

Credit score is increasingly been used in a number of countries and context, as a key determinant of one's (credit) worth in a credit system. To mitigate limitation of existing risk models, we focused on privacy protection of CSC in this article. Specifically, we designed a PCSC system and described its security requirements (i.e., weight confidentiality and credit confidentiality). To the best of authors' knowledge, this is the first such system with formal security definitions. We then presented a concrete construction based on Paillier encryption, with three purposefully designed NIZK schemes. We also gave the security proof of the proposal and evaluated its performance to demonstrate feasibility.

However, the size of PIW and PED proofs increases significantly as the number of credit data items increases. This incurs significant storage and communication costs. Therefore, in our future research, we intend to enhance the design by having a constant proof size for better supporting the PCSC system.

REFERENCES

[1] Z. Wang, S. Yan, and C. Zhang, "Active learning with adaptive regularization," *Pattern Recognit.*, vol. 44, no. 10/11, pp. 2375–2383, 2011.
[2] B. Gutierrez-Nieto, C. Serrano-Cinca, and J. Camon-Cala, "A credit score system for socially responsible lending," *J. Bus. Ethics*, vol. 133, no. 4, pp. 691–701, 2016.
[3] L. Thomas, J. Crook, and D. Edelman, *Credit Scoring and its Applications*, vol. 2. Philadelphia, PA, USA: SIAM, 2017.
[4] R. Zeidan, C. Boechat, and A. Fleury, "Developing a sustainability credit score system," *J. Bus. Ethics*, vol. 127, no. 2, pp. 283–296, 2015.
[5] L. Zhou, K. K. Lai, and L. Yu, "Least squares support vector machines ensemble models for credit scoring," *Expert Syst. Appl.*, vol. 37, no. 1, pp. 127–133, 2010.
[6] Y. Li, J. Gao, A. Z. Enkavi, L. Zaval, E. U. Weber, and E. J. Johnson, "Sound credit scores and financial decisions despite cognitive aging," *Proc. Nat. Acad. Sci. USA*, vol. 112, no. 1, pp. 65–69, 2015.
[7] G. N. Masters, "Partial credit model," in *Handbook of Item Response Theory*, vol. 1. London, U.K.: Chapman & Hall, 2016, pp. 137–154.
[8] S. Y. Sohn, D. H. Kim, and J. H. Yoon, "Technology credit scoring model with fuzzy logistic regression," *Appl. Soft Comput.*, vol. 43, pp. 150–158, 2016.

Authorized licensed use limited to: University of Texas at San Antonio. Downloaded on March 16,2021 at 12:11:27 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[9] H. J. Smith, T. Dinev, and H. Xu, "Information privacy research: An interdisciplinary review," *MIS Quart.*, vol. 35, no. 4, pp. 989–1015, 2011.

[10] D. Malandrino and V. Scarano, "Privacy leakage on the web: Diffusion and countermeasures," *Comput. Netw.*, vol. 57, no. 14, pp. 2833–2855, 2013.

[11] Y. Li, W. Dai, Z. Ming, and M. Qiu, "Privacy protection for preventing data over-collection in smart city," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1339–1350, May 2016.

[12] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay—A secure two-party computation system," in *Proc. 13th Conf. USENIX Secur. Symp.*, vol. 13, no. 1, pp. 1–20, 2004.

[13] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits," in *Proc. 20th USENIX Secur. Symp.*, San Francisco, CA, USA, no.1, pp. 1–35, 2011.

[14] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2004, pp. 1–19.

[15] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen, "On private scalar product computation for privacy-preserving data mining," in *Proc. 7th Int. Conf. Inf. Secur. Cryptology.*, Seoul, South Korea, 2004, pp. 104–120.

[16] R. Dowsley, J. van de Graaf, D. Marques, and A. C. A. Nascimento, "A two-party protocol with trusted initializer for computing the inner product," in *Proc. 11th Int. Workshop Inf. Secur. Appl.*, Jeju Island, South Korea, Aug. 2010 pp. 337–350.

[17] C. Gentry and D. Boneh, *A Fully Homomorphic Encryption Scheme*, vol. 20. Stanford, CA, USA: Stanford Univ. Press, 2009.

[18] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proc. 29th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2010, pp. 24–43.

[19] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *J. Cryptology*, vol. 1, no. 2, pp. 77–94, 1988.

[20] I. Damgård, "On σ-protocols," Lecture Notes, Dept. Comput. Sci., Univ. Aarhus, Aarhus, Denmark, 2002.

[21] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM J. Comput.*, vol. 18, no. 1, pp. 186–208, 1989.

[22] M. Blum, A. De Santis, S. Micali, and G. Persiano, "Non-interactive zero knowledge," *Siam J. Comput.*, vol. 20, no. 6, pp. 1084–1118, 1991.

[23] J. Groth, R. Ostrovsky, and A. Sahai, "New techniques for noninteractive zero-knowledge," *J. ACM*, vol. 59, no. 3, 2012, Art. no. 11.

[24] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, Prague, Czech Republic, May. 1999 pp. 223–238.

[25] I. Damgård, M. Jurik, and J. B. Nielsen, "A generalization of Paillier's public-key system with applications to electronic voting," *Int. J. Inf. Secur.*, vol. 9, no. 6, pp. 371–385, 2010.

[26] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proc. Adv. Cryptology*, Santa Barbara, CA, USA, 1986, pp. 186–194.

[27] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. 1st ACM Conf. Comput. Commun. Secur.*, Fairfax, VA, USA, Nov. 1993, pp. 62–73.

[28] D. Boneh and X. Boyen, "Short signatures without random oracles," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2004, pp. 56–73.

[29] J. Camenisch, R. Chaabouni, and A. Shelat, "Efficient protocols for set membership and range proofs," in *Proc. 14th Int. Conf. Theory Appl. Cryptology Inf. Secur.*, Melbourne, VIC, Australia, Dec. 2008, pp. 234–252.

[30] S. Ma, Y. Deng, D. He, J. Zhang, and X. Xie, "An efficient NIZK scheme for privacy-preserving transactions over account-model blockchain," *IEEE Trans. Dependable Secure Comput.*, 2017. doi: 10.1109/TDSC.2020.2969418.

[31] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Proc. 12th Int. Workshop Sel. Areas Cryptography*, Kingston, ON, Canada, Aug. 2005, pp. 319–331.

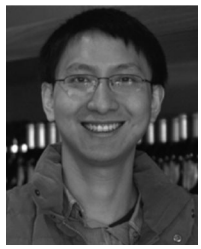
[32] C. Cachin and J. Camenisch, Eds., *Advances in Cryptology—EUROCRYPT 2004*, vol. 3027. Berlin, Germany: Springer, 2004.

Chao Lin received the Ph.D. degree from the School of Cyber Science and Engineering, Wuhan University, Wuhan, China, in 2020.

He is currently with the College of Mathematics and Informatics and the Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou, China. His research interests mainly include applied cryptography and blockchain technology.

Min Luo received the Ph.D. degree in computer science from Wuhan University, Wuhan, China, in 2003.

He is currently an Associate Professor with the School of Cyber Science and Engineering, Wuhan University. His research interests mainly include applied cryptography and blockchain technology.



Xinyi Huang (Member, IEEE) received the Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Wollongong, NSW, Australia, in 2009.

He is currently a Professor with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China, and the Co-Director of the Fujian Provincial Key Laboratory of Network Security and Cryptology. He has authored more than 100 research papers in refereed international conferences and journals. His research interests include applied cryptography and network security.

Dr. Huang is an Associate Editor for the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING. His work has been cited more than 8400 times at Google Scholar (H-Index: 49).

Kim-Kwang Raymond Choo (Senior Member, IEEE) received the Ph.D. degree in information security from the Queensland University of Technology, Brisbane, QLD, Australia, in 2006.

He currently holds the Cloud Technology Endowed Professorship with The University of Texas at San Antonio, San Antonio, TX, USA.

Dr. Choo is an IEEE Computer Society Distinguished Visitor from 2021 to 2023, included in Web of Sciences Highly Cited Researcher in the field of cross-field in 2020, and in 2015 he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen-Nuremberg. He is the recipient of the 2019 IEEE Technical Committee on Scalable Computing (TCSC) Award for Excellence in Scalable Computing (Middle Career Researcher), the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, the British Computer Society's 2019 Wilkes Award Runner-up, the 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, the Fulbright Scholarship in 2009, the 2008 Australia Day Achievement Medallion, and the British Computer Society's Wilkes Award in 2008. He has also received best paper awards from the *IEEE Consumer Electronics Magazine* in 2020, EURASIP *Journal on Wireless Communications and Networking* in 2019, IEEE TrustCom 2018, and ESORICS 2015; the Korea Information Processing Society's JIPS Survey Paper Award (Gold) 2019; the IEEE Blockchain 2019 Outstanding Paper Award; and the Best Student Paper awards from Inscrypt 2019 and ACISP 2005.

Debiao He (Member, IEEE) received the Ph.D. degree in applied mathematics from the School of Mathematics and Statistics, Wuhan University, Wuhan, China, in 2009.

He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. He has authored or coauthored more than 100 research papers in refereed international journals and conferences, such as IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION SECURITY AND FORENSIC, and *USENIX Security Symposium*. His main research interests include cryptography and information security, in particular, cryptographic protocols.

Dr. He was the recipient of the 2018 IEEE SYSTEMS JOURNAL Best Paper Award and the 2019 IET Information Security Best Paper Award. His work has been cited more than 7000 times at Google Scholar. He is in the Editorial Board of several international journals, such as the *Journal of Information Security and Applications*, *Frontiers of Computer Science*, and *Human-Centric Computing and Information Sciences*.

Authorized licensed use limited to: University of Texas at San Antonio. Downloaded on March 16,2021 at 12:11:27 UTC from IEEE Xplore. Restrictions apply.