Hindawi
Security and Communication Networks
Volume 2022, Article ID 9983995, 18 pages
https://doi.org/10.1155/2022/9983995

Research Article

Decentralized, Privacy-Preserving, Single Sign-On

Omid Mir,[1] **Michael Roland**,[2] **and René Mayrhofer**[2]

[1]*Johannes Kepler University Linz, LIT Secure and Correct Systems Lab, Linz, Austria*
[2]*Johannes Kepler University Linz, Institute of Networks and Security, Linz, Austria*

Correspondence should be addressed to Omid Mir; mir@ins.jku.at

Received 2 April 2021; Accepted 11 November 2021; Published 22 January 2022

Academic Editor: David Meghias

In current single sign-on authentication schemes on the web, users are required to interact with identity providers securely to set up authentication data during a registration phase and receive a token (credential) for future access to services and applications. This type of interaction can make authentication schemes challenging in terms of security and availability. From a security perspective, a main threat is theft of authentication reference data stored with identity providers. An adversary could easily abuse such data to mount an offline dictionary attack for obtaining the underlying password or biometric. From a privacy perspective, identity providers are able to track user activity and control sensitive user data. In terms of availability, users rely on trusted third-party servers that need to be available during authentication. We propose a novel decentralized privacy-preserving single sign-on scheme through the Decentralized Anonymous Multi-Factor Authentication (DAMFA), a new authentication scheme where identity providers no longer require sensitive user data and can no longer track individual user activity. Moreover, our protocol eliminates dependence on an always-on identity provider during user authentication, allowing service providers to authenticate users at any time without interacting with the identity provider. Our approach builds on threshold oblivious pseudorandom functions (TOPRF) to improve resistance against offline attacks and uses a distributed transaction ledger to improve availability. We prove the security of DAMFA in the universal composibility (UC) model by defining a UC definition (ideal functionality) for DAMFA and formally proving the security of our scheme via ideal-real simulation. Finally, we demonstrate the practicability of our proposed scheme through a prototype implementation.

1. Introduction

Authenticated Key Exchange (AKE) is one of the most broadly used cryptographic primitives that enable two parties to create a shared key over a public network. Typically, the parties need to have authentication tokens, e.g., cryptographic keys (asymmetric or symmetric high-entropy keys) or short secret values (low-entropy passwords). They also securely store these authentication tokens in a trusted service provider during the registration phase. There are various types of authentication factors such as knowledge, possession, and physical presence; low-entropy passwords are widely present in practice. An example of an authentication protocol that relies on passwords is Password-Based Authenticated Key Exchange (PAKE) [1].

However, passwords are usually vulnerable to both online and offline attacks [2, 3]. An attacker who compromises the data stored with the service provider (user account data, consisting of usernames and associated (potentially salted) password hashes) can run an offline dictionary attack on that data. Such an attack leads to the disclosure of user accounts and this has happened several times in the past, cf. [2, 4, 5]. Even if low-entropy passwords are correctly salted and hashed, they still do not resist the brute force of modern hardware. Already in 2012, a rig of 25 GPUs could test up to 350 billion guesses per second in an offline dictionary attack [6].

Multi-Factor Authentication (MFA) schemes overcome this risk by adding additional authentication factors. MFA combines (low-entropy) passwords with, e.g., secret values stored in physical tokens. Recent advancements in fingerprint readers and other sensors have led to the increased usage of smartphones and biometric factors in MFA schemes (e.g, the use of biometrics to securely retrieve private information [8]) Figure 1, although these methods make the guessing of authentication factors more difficult. However, some MFA schemes incorporate password authentication and second-factor authentication as separate mechanisms and store a salted password hash (or biometric) on the server, leading to different vulnerabilities such as spoofing and offline attacks [7, 9]. In other words, an adversary compromising the server is still able to recover the actual password (even if that password is no longer usable without the additional associated factors). Moreover, mobile devices (smartphones, wearables, FIDO U2F, etc.) are considered more likely to be subject to loss or theft, and particularly smartphones and wearables open a large, high-risk attack surface for malware [10, 11].

In general, authentication schemes are designed to uniquely identify a user. Consequently, they do not aim at protecting user privacy, and users' activity in the digital world can easily be logged and analyzed. Leakage of individual information may have serious consequences for users (including financial losses). To meet the increasing need of privacy protection in the digital world, multi-factor authentications are enhanced with privacy-preserving technologies. For instance, anonymous authentication schemes allow a member of a legitimate group, called a prover, to convince a verifier that it is a member of the group without revealing any information that would uniquely identify the prover within the group. Various schemes for anonymous password authentication have been proposed, e.g., [12–15]. In particular, anonymous password authentication promises unlinkability: The prover (e.g., the server of a service or identity provider) should not be able to link user authentications. Therefore, for any two authentication sessions, the prover is unable to determine if they have been performed by the same user or two different users.

1.1. Building a Fully Decentralized Authentication Architecture.

An Identity Provider (IDP) with a centralized database of authentication data of all users could easily provide an MFA scheme and offer convenient single sign-on (SSO) to other services for its users [16]. SSO allows users to once receive a single token (identity) provided by IDP and repeatedly authenticate themselves to servics providers. Several initiatives such as PRIMA [17], OAuth [18], SAML [19], and OpenID [16] let service providers take advantage of another centralized identity provider to authenticate users without becoming responsible for managing account passwords. In all these systems, the authentication follows a similar scheme (see Figure 2) [20]:

(1) In the registration phase, the user creates credentials (e.g., a username/ID and a password) and passes them to the IDP (a trusted server) which stores the username together with the hash of the password.

(2) In the authentication phase, the IDP verifies the user-supplied sign-on credential by matching the username and password hash.

(3) After successful verification, the IDP issues an authentication credential (a digital signature or a message authentication code) using a master secret key that authenticates the user to the service provider (e.g., a website) they want to visit.

However, this kind of centralized system poses several challenges:

(1) The IDP represents a single point of failure and an obvious target for attacks, such as:

(a) extraction of the secret key to forge tokens, which enable access to arbitrary services and data in the system; (b) capturing hashed passwords (or biometrics) to run offline dictionary attacks in order to recover user credentials, both potentially resulting in severe damage to the reliability of the system [20].

(2) The IDP is actively involved in each authentication session and can, therefore, track user activity, leading to serious privacy issues [21, 22].

(3) The IDP takes a significant amount of control over the digital identity away from the user. Users cannot fully manage and store their identity by themselves but always need to rely on and interact with an available IDP that offers the identity management system to them and the service provers they want to interact with (active verification).

1.2. Our Contribution.

To address the above challenges, we construct a novel decentralized privacy-preserving single sign-on scheme using a new Decentralized Anonymous Multi-Factor Authentication (DAMFA) scheme, where the process of user authentication no longer depends on a single trusted third party. Instead, it is fully decentralized onto a shared ledger to preserve user privacy while maintaining the single sign-on property. That is, users do not need to register their credentials with each service provider individually. The scheme also permits services where authenticating users remain anonymous within a group of users. Subsequently, our scheme does not require the IDP to be online during the verification (passive verification). Moreover, since there is no single third party (i.e., the IDP) in control of the whole authentication process, user and usage tracking by the IDP is inhibited.

The passive verification property of our scheme allows service providers to authenticate users at any time without requiring additional interaction with an IDP except what is available on the shared ledger. This property removes the cost of running secure channels between the service provider and the identity provider. Simultaneously, the IDP is eliminated as a single point of failure and attack within the authentication process.

The scheme relies on personal identity agents as auxiliary devices that assist the user in the authentication process. The personal identity agents participate in a threshold secret sharing scheme to store the distributed private key of their users. In the authentication phase, the user unlocks their private key through a combination of biometrics and a password, combining biometric, knowledge, and possession factors. The distributed architecture prevents offline attacks

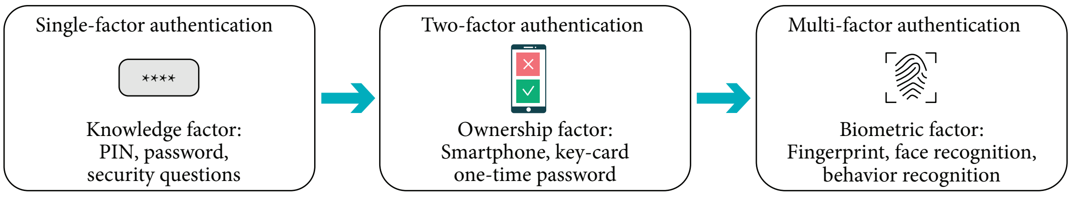
Single-factor authentication
****
Knowledge factor: PIN, password, security questions
Two-factor authentication
Ownership factor: Smartphone, key-card one-time password
Multi-factor authentication
Biometric factor: Fingerprint, face recognition, behavior recognition

Figure 1: Evolution of authentication methods from SFA to MFA [7].

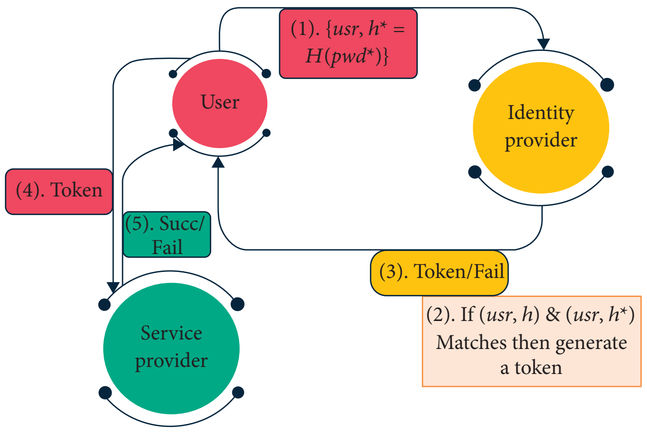
(1). $\{usr, h^* = H(pwd^*)\}$
User
Identity provider
(4). Token
(5). Succ/Fail
(3). Token/Fail
(2). If (usr, h) & (usr, h^*) Matches then generate a token
Service provider

Figure 2: The generic flow diagram shows the authentication phase of a password-based token method. The figure does not include the registration phase where the users store their username (usr) and hashed password (h) with the identity provider.

against data extracted from compromised agents, as long as only a set of agents below the threshold is compromised or corrupted.

We define the ideal functionality and real-world definitions for the security of our DAMFA scheme. We prove our construction's security via ideal-real simulation, showing the impossibility of offline dictionary attacks. Finally, we demonstrate that our protocol is efficient and practical through a prototypical implementation and through a comparison of our scheme with other SSO works.

2. Related Work

2.1. Single-Factor (Password) Authentication Key Exchange. For a long time, knowledge was (and still is) used as a primary means of authentication. Single-factor authentication based on passwords and PINs is a mechanism that is well-studied. Bellovin and Merritt [24] proposed Encrypted Key Exchange (EKE) where a client and a server share a password and use it to exchange encrypted information to agree on a common session key. EKE was followed by several enhancements (cf. [25–27]). Bellare et al. [1] expanded this to a general formal provable model for Password Authentication Key Exchange (PAKE). After that, two generic schemes of PAKE were proposed by Gennaro and Lindell [28] and by Groce and Katz [29] which are among the most efficient ways of constructing PAKE in the standard model.

Benhamouda and Pointcheval [30] explicitly introduce a verifier into the authenticated key exchange, where a verifier is a hash value or transformation $V = H(s, \mathrm{pw})$ of the secret password pw with a public salt s, and the server stores the pair (s, V) for each user.

2.2. Multi-Factor Authentication. A single knowledge-based authentication factor has the disadvantage that an adversary needs to only compromise that single factor. Multi-factor authentication (MFA) overcomes this by combining multiple different factors. The widely used combination is long-term passwords with secret keys, possibly stored in tokens (e.g., FIDO U2F). Shirvanian et al. [31] introduce a framework to analyze such two-factor authentication protocols. In their framework, the participants are a user, a client (e.g., a web browser), a server, and a device (e.g., a smartphone). In the authentication phase, the user sends a password and some additional information provided by the device. In most existing solutions, including Refs. [31–33], during the registration process, the user gets a value called the "token," while the server records a hashed password. During the authentication phase, the two required factors (the password and the token) are sent to a verifier.

Jarecki et al. [34] provide a device-enhanced password-authenticated key exchange protocol employing mobile device storage as a token. This setting serves two purposes: Firstly, for an adversary to successfully mount an offline dictionary attack, they must corrupt the login server in addition to the mobile device storage. Secondly, the user must confirm access to the mobile device storage during login.

Another popular factor used to authenticate users to remote servers is biometrics [35–38]. Fleischhacker et al. [39] also propose a modular framework called MFAKE which models biometrics following the liveness assumption of Pointcheval and Zimmer [37]. However, Zhang et al. [40] demonstrate that their scheme does not adequately protect privacy. Indeed, biometric authentication becomes a weak point when the framework directly uses the biometric template for authentication. In addition, it requires to, respectively, execute a lot of sub-protocols which makes the scheme inefficient.

2.3. Anonymous Authentication. Another approach towards user authentication is the anonymous password authentication protocol proposed by Viet et al. [12]. They combine an oblivious transfer protocol and a password-authenticated key exchange scheme. Further enhancements were proposed by Refs. [14, 15, 38].

An anonymous authentication protocol permits users to authenticate themselves without disclosing their identity and becomes an important method for constructing privacy-preserving authenticated public channels.

Zhang et al. [40] presented a new anonymous authentication protocol that relies on a fuzzy extractor. They consider a practical application and suggest several authentication factors such as passwords, biometrics (e.g., fingerprint), and hardware with reasonably secure storage (e.g., smartphone).

2.4. Summary of Related Works. Single-factor authentication based on passwords is a primary means of many authentication protocols [1, 25, 28, 41]. Multi-factor authentication (MFA) overcomes the problem of compromise in a single factor by combining multiple different factors [31, 34, 40, 42, 43]. An anonymous authentication protocol permits users to authenticate themselves without disclosing their identity [12, 14, 15, 44]. Finally, SSO allows users to once receive a single token provided by IDP and repeatedly authenticate themselves to service providers [16, 17, 19, 45, 46].

3. Building Blocks

3.1. Pointcheval and Sanders Signature. Our work relies on the credentials scheme proposed by Pointcheval and Sanders [47]. The scheme works in a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of type 3, with a bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ and has the following algorithms:

(1) $\text{Setup}(1^\lambda) \longrightarrow (\text{params})$: Choose a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ with order p, where p is a prime number. Let g_1 be a generator of $\mathbb{G}_1$, and g_2 a generator of $\mathbb{G}_2$. The system parameters are

$$\text{params} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2). \tag{1}$$

(2) $\text{KeyGen}(\text{params}) \longrightarrow (sk, vk)$: Choose a random secret key $sk = (x, y) \in Z_p$. Parse params, and publish the verification key

$$vk = (g_2, X, Y) = (g_2, g_2^x, g_2^y). \tag{2}$$

(3) $\text{Sign}(\text{params}, sk, m) \longrightarrow (\sigma)$: Parse $sk = (x, y)$. Pick a random element $h \in \mathbb{G}_1$, and output

$$\sigma = (h, s) = (h, h^{x+y\cdot m}). \tag{3}$$

(4) $\text{Verify}(pk, m, \sigma)$: Parse σ as (σ_1, σ_2) and check whether $\sigma_1 \neq 1_{G_1}$ and $e(\sigma_1, X\cdot Y^m) = e(\sigma_2, g_2)$ are both satisfied. In the positive case, it outputs 1, otherwise 0.

The signature $\sigma = (h, s)$ is randomizable by choosing a random $r' \in Z_p$ and computing $\sigma' = (h^{r'}, s^{r'})$. The above scheme can be modified to obtain a signature on a hidden message (commitment) and also offers a protocol to show a zero-knowledge proof of a signature $\sigma = (\sigma_1, \sigma_2)$.

3.2. Oblivious Pseudo-random Function (OPRF). A pseudo-random function (PRF) F is a function that takes two inputs: a secret function key k and a value x to compute on. It outputs $F_k(x)$, a function picked randomly from a PRF family, which is secure if it is distinguishable from a random function with the same domain and range with a negligible probability for all probabilistic polynomial time (PPT) distinguishers. An oblivious PRF (OPRF, cf. [48]) is a protocol between two parties (a sender and a receiver) that securely computes $F_k(x)$ where both x and k are the inputs of sender and receiver, respectively, such that no party learns anything except for the input holder that learns $F_k(x)$.

A threshold OPRF (TOPRF, cf. [49]) is an extension of the OPRF which allows a group of servers to secret share a key k for a PRF F with a shared PRF evaluation protocol which lets the user compute $F_k(x)$ on an input x, so that both x and k are secret if no more than t of n servers are corrupted (see Figure 3).

A formal definition of the TOPRF protocol as a realization of the TOPRF functionality is given in Figure 4. Note that we just duplicate these functionalities so that readers can easily follow our ideal functionality and construction (for more details see [49]).

3.3. Secret Sharing Scheme. A secret sharing scheme consists of two PPT algorithms [50]: First, TSSGen generates n shares of the secret key K as $\langle k_1, \ldots, k_n\rangle \longleftarrow \text{TSSGen}(K)$, and second TSSRecon uses t shares to retrieve the primary secret value K as $K \longleftarrow \text{TSSRecon}(s_1, \ldots, s_t)$. The security assumption of this scheme is that any amount of shares below the threshold does not disclose any info about the secret key.

3.4. Public Append-Only Ledger. A ledger allows us to keep a list of public information and maintains the integrity of the dataset. It guarantees a consistent view of the ledger for every party. Every user can insert information into the ledger and, once some data are uploaded, nobody can delete or modify it. Moreover, the ledger assures the correctness of pseudonyms and guarantees that no one can impersonate another participant to release information. Furthermore, it distributes up-to-date data to all participants. In this paper, we assume this assumption holds and construct our system on the blockchain technique as a public append-only ledger (blockchain). There are already some works constructing advanced applications based on this assumption, such as Refs. [51–53]. Yang et al. [54] formally define a public append-only ledger, which we use for constructing our DAMFA system (see Figure 5).

F_B executes the following steps with parties $\{PA_1, \ldots, PA_n\}$ and an ideal adversary $\mathcal{S}$ as follows:

(1) Initialize. Initialize creates an empty list L_p in the beginning.

(2) Store. On input $(\text{Store}, PA_i, \text{Nym}_u^o, M)$, checks that Nym_u^o is a valid pseudonym for PA_i, then stores the tuple (Nym_u^o, M) to L_p and declares to $\mathcal{S}$ that a new item was appended to the list L_p.

Key and server initialization. A random key $k \leftarrow Z_p$ is secret shared using Shamir's scheme with parameters n, t; each server $S_i, i \in [n]$, receives a share k_i. Also, they use a hash function as $H_g : M \rightarrow \mathbb{G}$.
Threshold oblivious computation of $F_k(x)$:

- On input x, user U picks $r \leftarrow Z_p$ and computes $A := H_g(x)^r$; it chooses a subset SR of $[n]$ of size $t+1$ and sends to each server $S_i, i \in \mathsf{SR}$, the value A and the subset SR.
- Upon receiving the message A from U, server S_i verifies that $A \in \mathbb{G}$ and, upon successfuly verification, responds with $b_i := A^{\lambda_i \cdot k_i}$ where λ_i is a Lagrange interpolation coefficient for index i and subset SR.
- When U receives messages b_i from each server $S_i, i \in \mathsf{SR}$, U outputs the value
$$H\left(x, \left(\prod_{i \in \mathsf{SR}} b_i\right)^{1/r}\right)$$
as the result of $F_k(x)$.

Figure 3: (n, t)-threshold computation in a TOPRF protocol [49].

Assume $tx(p, S)$ and $T(p, x)$ are undefined for all p, x, S.

Initialization.

- On message $(Init, \mathsf{sid}, \mathsf{SI})$ from S, ignore if $|\mathsf{SI}| \neq n$ or S is active. Otherwise, mark S as "active" and if no record $\langle \mathsf{sid}, [\ldots] \rangle$ exists, let $t^* = \mathsf{Corrupted}$ be the subset of SI that is corrupted. If $t^* \leq t$ then picks any previously unused label p and records $\langle \mathsf{sid}, \mathsf{SI}, p \rangle$. Sends $(Init, \mathsf{sid}, S, \mathsf{SI}, p)$ to $\mathcal{A}^*$.
- On message $(Init, \mathsf{sid}, \mathcal{A}^*, p)$ from $\mathcal{A}^*$, check that p is a label that has not been used before, record $\langle \mathcal{A}^*, p \rangle$ and return $(Init, \mathsf{sid}, \mathcal{A}^*, p)$ to $\mathcal{A}^*$.
- On message $(InitComplete, \mathsf{sid}, S)$ from $\mathcal{A}^*$, retrieve tuple $\langle \mathsf{sid}, \mathsf{SI}, p \rangle$. Ignore the message if there is no such tuple, $S \notin \mathsf{SI}$, or not all servers in SI are active. Otherwise, send $(InitComplete, \mathsf{sid})$ to S and mark S as "initialized".

Evaluation.

- On message $(Eval, \mathsf{sid}, \mathsf{ssid}, \mathsf{SE}, x)$ from $P \in \{U, \mathcal{A}^*\}$, retrieve $\langle \mathsf{sid}, \mathsf{SI}, p \rangle$ if $P = U$ or $\langle \mathcal{A}^*, p \rangle$ if $P = \mathcal{A}^*$. Ignore the message if there is no such tuple or if $|\mathsf{SE}| \neq t+1$. Otherwise, record $\langle \mathsf{ssid}, P, p, \mathsf{SE}, x \rangle$ and send $(Eval, \mathsf{sid}, \mathsf{ssid}, P, \mathsf{SE})$ to $\mathcal{A}^*$.
- On message $(SndrComplete, \mathsf{sid}, \mathsf{ssid}, S)$ from $\mathcal{A}^*$, retrieve tuple $\langle \mathsf{sid}, \mathsf{SI}, p \rangle$. Ignore the message if there is no such tuple, if $S \notin \mathsf{SI}$, or if S is not initialized. Otherwise, set $tx(p, S)$++ (or set it to 1 if $tx(p, S)$ is undefined) and send $(SndrComplete, \mathsf{sid}, \mathsf{ssid})$ to S.
- On message $(RcvComplete, \mathsf{sid}, \mathsf{ssid}, P = \{U, \mathcal{A}^*\}, p^*)$ from $\mathcal{A}^*$, retrieve $\langle \mathsf{ssid}, P, p, \mathsf{SE}, x \rangle$. Ignore the message if there is no such tuple or if any of the following conditions fail: (i) if $p^* = p$ then $|\{S \in \mathsf{SI} \mid tx(p, S) > 0\}| > t$, (ii) if all servers in SE are honest then $p^* = p$. Otherwise, if $p^* = p$ then set $tx(p, S)$−− for any $t+1$ distinct $S \in \mathsf{SI}$ s.t. $tx(p, S) > 0$. Then, if $T(p^*, x)$ is defined, send $(Eval, \mathsf{sid}, T(p^*, x))$ to P. Otherwise, pick $\rho \leftarrow \{0,1\}^l$ and set $T(p^*, x) := \rho$. Finally, send $(Eval, \mathsf{sid}, \mathsf{ssid}, \rho)$ to P.

Figure 4: Functionality F_{TOPRF} [49].

F_B executes the following steps with parties $\{\mathsf{PA}_{\mathsf{i}1}, \ldots, \mathsf{PA}_{\mathsf{i}n}\}$ and an ideal adversary $\mathcal{S}$ as follows:

- Initialize. Initialize creates an empty list L_p in the beginning.
- Store. On input $(\mathsf{Store}, \mathsf{PA}_{\mathsf{i}i}, \mathsf{Nym}_u^o, M)$, checks that Nym_u^o is a valid pseudonym for $\mathsf{PA}_{\mathsf{i}i}$, then stores the tuple (Nym_u^o, M) to L_p and declares to $\mathcal{S}$ that a new item was appended to the list L_p.
- Retrieve. On input $(\mathsf{Retrieve}, \mathsf{PA}_{\mathsf{i}i})$, returns the list L_p to $\mathsf{PA}_{\mathsf{i}i}$.

Figure 5: Functionality F_B [54].

(3) Retrieve. On input $(\text{Retrieve}, PA_i)$, returns the list L_p to PA_i.

3.5. Zero-Knowledge Proof of Knowledge. In a zero-knowledge proof of knowledge system [55], a prover proves to a verifier that it possesses the witness for a statement without revealing any additional information. In this paper, we use noninteractive zero-knowledge proofs known as Fiat-Shamir heuristic [56] as they have the advantage of being noninteractive. For example, *NIZKPoK* denotes a noninteractive zero-knowledge proof of the elements x and y as $NIZKPoK\{(x, y) : h = g^x \wedge c = g^y\}$ that satisfies both $h = g^x$ and $c = g^y$. Values (x, y) are assumed to be hidden from the verifier. Similarly, the algorithm can admit a message as input, thus it is also called signature proof of knowledge denoted as $ZKSoK[m]\{(x, y){:}\ \ h = g^x \wedge c = g^y\}$.

3.6. Dynamic Accumulators. A dynamic accumulator is a primitive allowing a large set of values to be accumulated into a single quantity, the accumulator. For each value, there exists a witness which is the evidence attesting that the value is indeed contained in the accumulator. The proof of showing that a value is part of an accumulator can be zero-knowledge proof, which reveals neither the value nor the witness to the verifier. Camenisch et al. [57] define a concrete construction of dynamic accumulators with the five algorithms AccSetup, AccAdd, AccUpdate, AccWitUpdate, and AccVerify:

(1) AccSetup: This is the algorithm to output the public parameters. Select bilinear groups $\text{params}_{BM} = (q, \mathbb{G}, \mathbb{G}_T, e, g)$ with a prime order p and a bilinear map e. Select $g \in \mathbb{G}$. Select $\gamma \in Z_p$. Generate a key pair *msk* and *pk* for a secure signature scheme. Compute and publish $\{p, \mathbb{G}, T, e, g,\ g_1 = g^{\gamma_1}, \ldots,\ g_n = g^{\gamma_n}, g_{n+2} = g^{\gamma^{n+2}}, \ldots, g_{2n} = g^{\gamma^{2n}}\}$ and $z = e(g, g)^{\gamma_{n+1}}$ as the public parameters.

(2) AccAdd $(sk_A, i, acc_V, state_U)$. Compute $\omega = \prod_{j \in V}^{j \neq i} g_{n+1-j+i}$ and a signature σ_i on $g_i \parallel i$ under signing key sk. The algorithm outputs $wit_i = (\omega, \sigma_i, g_i)$, an updated accumulator value $acc_{V \cup i} = acc_V \cdot g_{n+1-i}$, and $state_{U \cup i} = (U \cup \{i\}, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$.

(3) AccUpdate: This is the algorithm to compute the accumulator using the public parameters. The accumulator acc_V of V is computed as
$$acc_V = \prod_{i \in V} g_{n+1-i}, \tag{4}$$

(4) AccWitUpdate: This is the algorithm to compute the witness that values are included in an accumulator, using the public parameters. Given V and the accumulator acc_V, the witness of values $i_1, \ldots, i_k$ in U is computed as
$$\omega' = \omega \cdot \frac{\prod_{j \in V / V_\omega} g_{n+1-j+i}}{\prod_{j \in V_w / V} g_{n+1-j+i}}. \tag{5}$$

(5) AccVerify: This is the algorithm to verify that values in U are included in an accumulator, using the witness and the public parameters. Given acc_V, $state_U$, and ω, accept if

$$\frac{e(g_i, acc_V)}{e(g, \omega)} = z. \quad (6)$$

As Camenisch et al. [57] point out, the purpose of an accumulator is to have accumulator and witnesses of size independent of the number of accumulated elements.

3.7. Pedersen Commitments. Using a commitment scheme, users can bind themselves to a chosen value without revealing the actual value to a third party receiving the commitment. Thereby, a user cannot change their choice (binding), and, at the same time, the recipient of a commitment does not learn anything about the actual value the user committed to (hiding of the value). Pedersen commitments [58] have a group $\mathbb{G}$ of prime order q and generators $(g_0, \ldots, g_m)$ as public parameters. For committing to the value $(z_1, \ldots, z_m) \in Z_q$, a user picks a random $r \in Z_q$ and sets $C = \text{PedCom}(z_1, \ldots, z_m; r) = g_0^r \prod_{i=1}^m g_i^{z_i}$.

4. Decentralized Anonymous Multi-Factor Authentication (DAMFA)

We build a new practical decentralized multi-factor authentication scheme, Decentralized Anonymous Multi-Factor Authentication (DAMFA), where the process of user authentication no longer depends on a single trusted third party. The scheme also permits services where authenticating users remain anonymous within a group of users. Subsequently, our scheme does not require the IDP to be online during the verification. To protect the private key of their user, we use personal identity agents as auxiliary devices that participate in a threshold secret sharing scheme to store the distributed private key of the user.

4.1. System Model. The overall system model of DAMFA is shown in Figure 6. The protocol is executed between four participants:

(1) User U: A user who wants to access various services offered by different service providers. During the registration phase (which runs only once), U obtains a biometric template Bio from a sensor and chooses a password pw. In the authentication phase, users U interact with a set of personal identity agents to authenticate themselves in an anonymous manner.

(2) Personal identity agent PA_i: We associate each user with a set of personal agents which are auxiliary devices that assist a user in creating a credential for authentication. These personal agents remain under the administrative control of their associated users, who can freely choose where to run them. For example, they could be run on a smart home controller, at a cloud provider, or even on a mobile phone. U generates a private key and executes threshold secret sharing on the private key to generate secret shares of that private key. The user stores the secret shares among their personal agents such that each PA_i has one share of the overall secret key.

(3) Service provider (verifier) SP: These are the service providers (untrusted and distributed servers) that require authentication from a user U. After verifying a user's credentials, they provide access to the corresponding service.

(4) Identity provider IDP: The identity provider is an entity that issues credentials to users. These credentials grant permission to use specific services by proving membership of a specific permission group (clients, employees, department members, account holders, subscribed users, etc.).

In addition, users act as nodes in the blockchain network: They collaboratively maintain a list of credentials in a public ledger (blockchain) and enforce a specific credential issuing policy when adding to that list. For more details on how these steps work, we refer to subsection 4.3., High-Level View.

4.2. Threat Model. In order to demonstrate the security of the proposed protocol, we determine the capabilities and possible actions of an attacker. We consider a PPT attacker who has perfect control of the communication channels. They can eavesdrop all messages in public channels and also modify, add, and remove messages on the network. The attacker can, at any time, corrupt $(t-1)$ of the user's agents (no more than threshold t), in which case the attacker knows all the long-term secrets (such as private keys or master shared keys).

In the proposed protocol, we consider some privacy requirements such as unlinkability, identity privacy, and user data privacy: Unlinkability means that an adversary cannot distinguish a user who is authenticating from any (other) user who has authenticated in the past. Identity privacy means that an adversary cannot determine if a given authentication credential belongs to a specific user. User data privacy means that an adversary cannot learn anything about the user's sensitive authentication data (i.e., biometric data, password).

4.3. High-Level View. To build a fully decentralized authentication architecture, we need to set up a small distributed shared database (to store credentials) between nodes. Data are highly available, but nobody has control over the database. Furthermore, users would never want to modify data in the past. User data need to be immutable, and data should be publicly accessible. We employ a public append-only ledger in order to fulfill our requirements. A ledger (blockchain) maintains the integrity of the dataset and guarantees a consistent view of the data for every party. Every participant can append information to the ledger and, once uploaded, nobody can delete or modify the data.

Definition 1 (DAMFA). A DAMFA system consists of a global transaction ledger instead of a single party

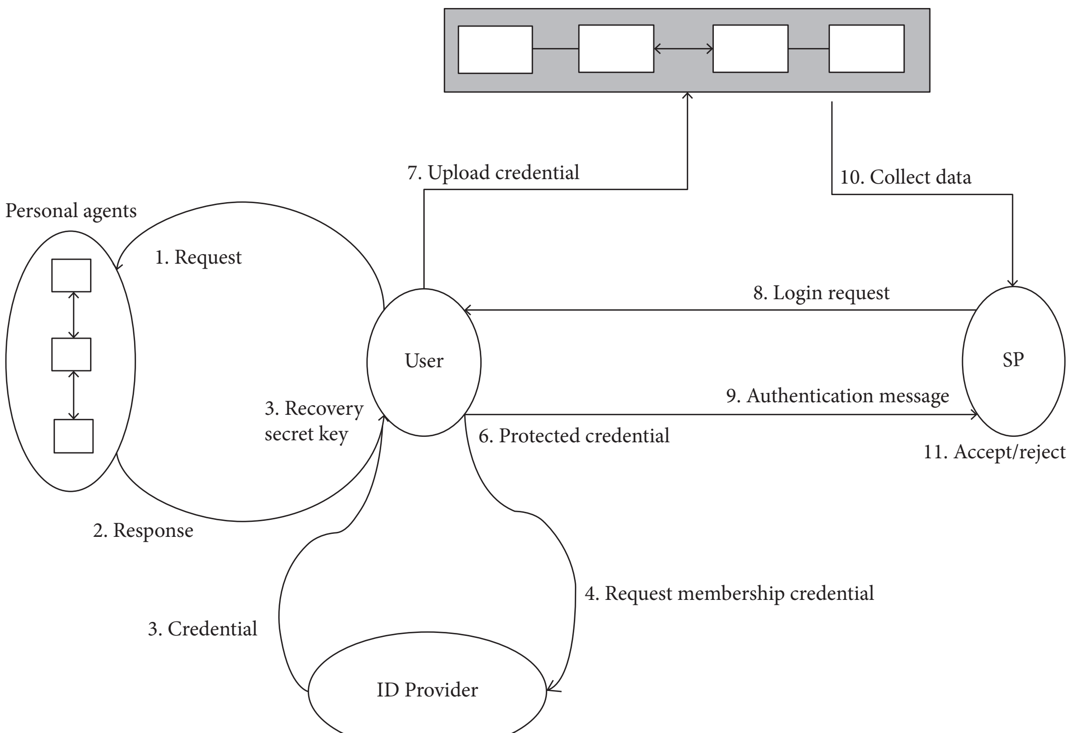

Figure 6: A system model of the DAMFA scheme.

representing the organization. Moreover, the DAMFA scheme consists of the following phases:

(1) Setup: In the setup phase, we define the public parameters and execute the following algorithm: U generates a private key and executes threshold secret sharing (TSS) on the private key to generate shares of that secret. The user stores the secret shares among their personal agents (similar to the initialization of TOPRF [49], done via a distributed key generation for discrete-log-based systems, e.g. Ref. [59]).

(2) Registration: In the registration phase, the user U first selects a password pw and collects their biometric Bio at a sensor. Then, U runs the TOPRF protocol by interacting with personal agents to reconstruct the TOPRF secret key. After that, the IDP issues a membership credential that shows that U is a valid member (employee, account holder, subscribed user, etc.). For this purpose, U sends a request with a pseudonym and a (noninteractive) zero-knowledge proof (NIZK) which indicates they are the owner of the pseudonym (they know the secret key that belongs to the pseudonym) and authenticate themselves to the IDP. Then, U receives a membership credential, which is a signature on their pseudonym. The user U creates a pseudonym Nym_u^o and verification information, namely, a protected credential PC_i, by encrypting the membership credential with the TOPRF secret key. Subsequently, U computes a NIZK proof that (1) the credential PC_i and the pseudonym contain the same secret key and (2) proof of knowledge of the signature which is issued by the ID provider (i.e., she has valid group membership). Note that the user can execute these actions in an offline state because no interaction with the public ledger is required. Finally, nodes accept the credential to the ledger if and only if this proof is valid.

(3) Authentication: The user U attempts to access the services of an SP in an anonymous and unlinkable way. SP authenticates the user if and only if the user provides a valid credential. First, a service provider sends an authentication request (which is a signature) to U. The user inserts the password pw* and the biometric Bio* and runs the TOPRF protocol by interacting with personal agents to reconstruct the TOPRF secret value. U first scans the public ledger to obtain the accumulator AC, which is a set $\overrightarrow{\mathrm{PC}} = \{\mathrm{PC}_1, \ldots, \mathrm{PC}_n\}$ consisting of all credentials belonging to a specific IDP. Then, U finds their own protected credential PC_i^* within this set (via the pseudonym Nym_u^o). U decrypts PC_i^* using the TOPRF secret key and recovers the initial credential (a signature from IDP). U presents the credential under a different pseudonym Nym_u^v by proving in zero-knowledge that (1) they know a credential PC_i on the ledger from IDP, (2) the credential opens to the same secret key as their own pseudonym Nym_u^v, and (3) they prove possession of a membership credential from IDP (the signature), cf. [52]. SP scans the public ledger to obtain the accumulator AC which is a set $\overrightarrow{\mathrm{PC}} = \{\mathrm{PC}_1, \ldots, \mathrm{PC}_n\}$ consisting of all credentials belonging to a specific organization.

Then, it checks the validity of the candidate credential by finding the candidate credential in the set $PC_i^* \in \overrightarrow{PC}$ and checking proof of knowledge on the credential and pseudonym.

4.4. The DAMFA Functionality. We formally define the proposed scheme's security by presenting its ideal functionality that is implemented via a trusted party F_{TOPRF} with a public ledger. All communication takes place through this ideal trusted party. In the UC framework [60, 61], there may be some copies of the ideal functionality running in parallel. Each one is supposed to have a unique session identifier (SID). Each time a message is sent to a specific copy of functionality, such that this message contains the SID of the copy that is intended for. As noted in Ref. [49], we also use the ticketing mechanism, which ensures that in order to test a password and biometric guess, the attacker must impersonate $t+1$ agents. To this end, they define a counter $tx(p, PA_i)$ for each $PA_i \in \text{SI}$ in which the parameter p is also used to identify it. In addition, when an agent $PA_i \in \text{SI}$ completes its interaction, the functionality increases the counter $tx(p, PA_i)$. On the other hand, when a user, either honest or corrupt, completes an interaction that is associated to PA_i, $tx(p, PA_i)$ decreases by 1. It ensures that for any honest agent PA_i, the number of user-completed OPRF evaluations with PA_i is no more than the number of agent-completed OPRF evaluations of PA_i. It sets $t+1$ agent tickets for accessing the proper TOPRF result by reducing (nonzero) ticket counters $tx(p, PA_i)$ for an arbitrary set of $t+1$ agents in SI. The ideal functionality as:

4.4.1. Registration

(i) Upon receiving (Reg, sid, SI, pw, Bio) for $|\text{SI}| = PA_n$ from U, records this message and sends $(\text{Reg}, U, \text{sid}, \text{SI})$ to $\mathscr{A}^*$ (Ignores other Reg cmd). Computes a secret key K using TOPRF protocol F_{TOPRF} and if $|\text{SI} \cap \text{CorrSrv}| \geq t+1$ then sends $(K, \text{pw}, \text{Bio})$ to $\mathscr{A}^*$.

(ii) Upon receiving $(\text{SReg}, \text{sid}, PA_i)$ from $\mathscr{A}^*$, if a record $\langle \text{Reg}, U, \text{sid}, \text{SI}, \text{pw}, \text{Bio} \rangle$ exists and $PA_i \in \text{SI}$ then marks PA as active and sends $(SInit, \text{sid})$ to PA.

(iii) Upon receiving $(UReg, \text{sid})$ from $\mathscr{A}^*$, if the record $\langle \text{Reg}, U, \text{sid}, \text{SI}, \text{pw}, \text{Bio} \rangle$ exists and all agents in SI are marked active, then runs a commitment scheme F_{Com} and an encryption F_{Enc} to get (τ_i, γ_i), respectively, and sets the pseudonym as $\text{Nym}_u^o = \tau_i$ and $\text{PC}_i = \gamma_i$ as the credential. It records $\langle \text{Nym}_u^o, \text{PC}_i, U, \text{SI}, K \rangle$, sends $(\text{sid}, \text{Nym}_u^o, \text{PC}_i)$ and $(\text{RegComplete}, \text{sid}, \text{SI})$ to its public ledger and $\mathscr{A}^*$, respectively.

4.4.2. Authentication

(i) Upon receiving $(\text{Auth}, \text{sid}, \text{ssid}, \text{SR}, \text{pw}', \text{Bio}')$ for $|\text{SR}| = t+1$ from U^*, retrieves $\langle \text{Reg}, U, \text{sid}, \text{SI}, \text{pw}, \text{Bio}, K \rangle$, records $\langle \text{Auth}, U^*, \text{sid}, \text{SI}, \text{SR}, \text{pw}, \text{pw}', \text{Bio}, \text{Bio}' \rangle$, and sends $(\text{Auth}, U^*, \text{sid}, \text{ssid}, \text{SR})$ to $\mathscr{A}^*$. Ignores future Auth commands involving the same ssid.

(ii) Upon receiving $(\text{SAuth}, \text{sid}, \text{ssid}, PA_i)$ from $\mathscr{A}^*$, if $PA_i \in \text{SR}$ is marked active then sets $tx(PA_i)++$ (sets it to 1 if it is undefined) and sends $(\text{SAuth}, \text{sid}, \text{ssid})$ to PA_i.

4.4.3. Password and Biometric Test

(iii) After receiving $(\text{TestPwBio}, \text{sid}, PA_i, \text{pw}^*, \text{Bio}^*)$ from $\mathscr{A}^*$, if $tx(PA_i) > 0$ then sets tested (pw) = tested(pw*) and (Bio) = tested(Bio*) $\cup\, PA_i$ and $tx(PA_i){:} = tx(PA_i) - 1$, retrieves $\langle \text{Reg}, U, \text{SI}, \text{pw}, \text{Bio}, K \rangle$ and if $|\text{SI} \cap (\text{tested}(\text{pw}^*) \wedge \text{tested}(\text{Bio}^*) \cup \text{CorrSrv})| \geq t+1$ and if $\text{pw}^* = \text{pw}$ and $\text{Bio} = \text{Bio}^*$, then returns sk to $\mathscr{A}^*$ and marks the record compromised and responses to $\mathscr{A}^*$ with "correct guess", else returns FAIL.

4.4.4. Authentication for Service Provider

(i) GetCredList: Every participant can obtain all data in the public ledger of the trusted party via submitting a "retrieve" request to F_{DAMFA}. SP then retrieves the intended credential PC_i issued by Nym_u^o from F_{TOPRF} and accepts functionality's assertion only if $\text{PC}_i \subset \overrightarrow{\text{PC}}$.

(ii) Key generation: Upon receiving $(UAuth, \text{sid}, \text{ssid}, P_i, \text{SR}, sk)$, for $|\text{SR}| = t+1$ from $\mathscr{A}^*$, if there is a record $\langle \text{Auth}, P, \text{sid}, \text{ssid}, \text{SI}, \text{SR}, \text{pw}, \text{Bio}, \text{pw}', \text{Bio}' \rangle$, where $P \in \{U, SP\}$ then do:

(a) If this record is compromised so that $\text{pw}^* = \text{pw}$ and $\text{Bio}^* = \text{Bio}$ or $(\text{SR} \subseteq \text{CorrSrv})$, then output (sid, sk) to player P_i.

(b) Else, if this record is fresh, and if there is a record $(P, \text{pw}', \text{Bio}', sk')$ with $\text{pw}' = \text{pw}$ and $\text{Bio}' = \text{Bio}$, then sends sk' (a random key) to player P_i.

(c) In any other case, picks a random key sk and sends (sid, sk) to P_i.

Definition 2 (Secure DAMFA). Let Π be a probabilistic polynomial time protocol for the DAMFA functionality. We say that Π is secure if for every PPT real world adversary $\mathscr{A}$ attacking DAMFA, there exists a PPT ideal world simulator $\mathscr{S}$ such that for both the real and ideal world interactions, outputs of registration and authentication phases are computationally indistinguishable: $\text{Real}_{\mathscr{A}}(1^\lambda) \approx \text{Ideal}_{\mathscr{S}}(1^\lambda)$.

4.5. Our Construction

4.5.1. Setup Phases

We select a bilinear pairing $e{:}\ \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ that is efficiently computable, nondegenerate, and three groups with prime order p. We let g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $g_t = e(g_1, g_2)$ the generator of $\mathbb{G}_T$. Note that it is assumed to support one-way Bio-hash function H_1, which resolves the recognition error of general hash functions [62]. We consider two additional hash functions as $H_2\colon M \longrightarrow \{0,1\}^{\lambda}$ and $H_g\colon M \longrightarrow \mathbb{G}_1$. We publish params $\longleftarrow (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, h_{nym}, H_1, H_2, H_g)$ as the set of system parameters where $h_{nym} \in \mathbb{G}_1$. The user U generates a private key K, then executes a secret sharing construction scheme on K to create secret keys for each personal agent $\langle k_1, k_2, \ldots, k_n \rangle \longleftarrow TSS(K)$. U stores secret shares among personal agents.

4.5.2. Registration Phase

To register a user to the system, U first chooses a password pw and scans her biometric impression Bio at the sensor. Then, U runs the following steps to register herself in the system.

(i) A user runs TOPRF protocol [49] with agents to compute the secret value $usk = F_K(\text{pw}, \text{Bio})$ as follows:

  (a) The user U picks a random number $r \in Z_p$ and computes $A = H_g(\text{pw}, H(\text{Bio}))^r$ and sends the message $M_1 = \{A\}$ to all PA_i.

  (b) Upon receiving the message $M_1 = \{A\}$ from the user, each PA_i computes $b_i = A^{k_i} = H_g(\text{pw}, H_1(\text{Bio}^*))^{\lambda_i \cdot k_i \cdot r}$ by Lagrange interpolation coefficients and secret key k_i (s.t. $K = \sum_{i \in \text{SR}} \lambda_i \cdot k_i$). They return the message $M_2 = \{b_i\}$ to U.

  (c) After receiving all the messages b_i from personal agents, U computes: $C = \prod_{i \in \text{SR}} b_i^{r^{-1}}$ $= H_g(\text{pw}, H_1(\text{Bio}))^{K} \longrightarrow usk = h(\text{pw}, C)$.

(ii) In order to obtain a membership credential from IDP, we use PS signatures protocol [47] to derive a signature on a hidden committed message as follows:

  (a) KeyGen(pp): The IDP runs this algorithm to generate private and public keys. This algorithm selects $(x, y, y_1) \longleftarrow Z_p$, computes $(X, Y, Y_1) \longrightarrow (g_1^x, g_1^y, g_1^{y_1})$ and $(X', Y', Y_1') \longrightarrow (g_2^x, g_2^y, g_2^{y_1})$, and sets $sk \longrightarrow (X, y, y_1)$ and $pk \longrightarrow (g_1, g_2, Y, X', Y')$.

  (b) Protocol. A user first selects a random $r_2 \longleftarrow Z_p$ and computes $C = g_1^{r_2} \cdot Y^{usk}$, which is a commitment on her secret key. She then sends C to the IDP. They both run a proof of knowledge of the opening of the commitment (authentication). If the signer is convinced, the IDP selects a random $u \longleftarrow Z_p$ and returns $\sigma \longleftarrow (\sigma_1 = g_1^u, \sigma_2 = (X \cdot C \cdot Y_1^{m})^u)$. The user can now unblind the signature σ and get a valid signature over her secret key and the message m_1 by computing $\sigma \longleftarrow (\sigma_1, \sigma_2/(\sigma_1)^{r_2})$ described in Sect. 3.1.

  (c) Verify. To verify this signature, the user can execute this algorithm and compute:

  (d) Verify(pk, m, σ): $e(\sigma_1, X' \cdot Y'^{usk} \cdot Y_1'^{m_1}) = e(\sigma_2, g_2)$.

(iii) CreatePC. The user generates a protected credential with TOPRF secret key usk derived from the password and the biometric: U picks a random number $s \in Z_p$ to generate a pseudonym as $\text{Nym}_u^o = g_1^s \cdot h_{nym}^{usk}$ and computes an El-Gamal encryption of the credential σ with secret TOPRF values usk into a ciphertext as: $\text{PC}_i = [\sigma]_{usk}$.

(iv) Proof. A NIZK proof of knowledge of the credential (PS signature [47]) works as follows: U selects random $r_3, t_1 \longleftarrow Z_p$ and computes $\sigma' \longleftarrow (\sigma_1^{r_3}, (\sigma_2 \cdot \sigma_1^{t_1})^{r_3})$. U sends $\sigma' = (\sigma_1', \sigma_2')$ to the verifier and carries out a zero-knowledge proof of knowledge (such as the Schnorr's interactive protocol) of m, usk, and t_1 such that

$$\pi = NIZK \left\{ \begin{array}{c} (s, m_1, t_1, usk)\colon \text{Nym}_u^o = g_1^s \cdot h_{nym}^{usk} \wedge \\ \\ \text{PC}_i = Enc_{usk}(\sigma) \wedge \\ \\ e(\sigma_1', Y)^{usk} \cdot e(\sigma_1', g_2)^{t_1} \cdot (\sigma_1', Y_1)^{m_1} \\ \\ = \dfrac{e(\sigma_2', g_2)}{e(\sigma_1', X)} \end{array} \right\}. \tag{7}$$

(v) At the end of this phase, U submits the resulting values $(\text{PC}_i, \pi, \text{Nym}_u^o)$ to the public ledger nodes where π is a proof of knowledge on the Nym_u^o and the PC_i. If the signature verifies successfully, output 1, otherwise 0. The nodes should accept values to the ledger if this algorithm returns 1.

4.5.3. Authentication Phase

In this phase, a user authenticates herself to the service provider and establishes a session key with the service provider. The following steps are executed by U, PA, and SP:

(i) First of all, the server chooses a secret key $y \longleftarrow Z_p$ and computes $Z \longleftarrow g_1^y$. Then, SP generates a signature σ_s on message Z (i.e., Schnorr's signature [55]) using its secret key and sends the message $M_1 = \{Z, \sigma_s\}$ to the user.

(ii) When receiving a pair (Z, σ_s), the client verifies whether σ_s is valid on message Z under the SP's public key. If σ_s is valid, U inserts pw* and scans her personal biometric impression Bio* at the sensor.

(iii) The user interacts with personal agents and runs the necessary steps to compute the TOPRF protocol

$F_K(\text{Bio}^*, \text{pw}^*) = usk = h(\text{pw}^*, \prod_{i\in \text{SR}} b_i^{r^{-1}})$.
Then, U decrypts ciphertext $[\sigma]_{usk}$ with the TOPRF secret key usk to recover the credential σ.

(iv) Show: The user creates a NIZK π ensuring that the credential is well-formed and the credential related to the same secret values as her pseudonym. Here we prove: (1) she knows a credential on the ledger from the IDP, (2) the credential includes the secret key as her pseudonym, (3) she possesses of a credential (signature). We use the bilinear maps accumulator [57] to accumulate the group elements $\{g_1, \ldots, g_n\}$ instead of, e.g., the integers $\{1, \ldots, n\}$. In addition, Camenisch et al. [57] describe an efficient zero-knowledge proof of knowledge such as Schnorr's protocol [55, 56] that a committed value is in an accumulator. See Refs. [57, 63] to find how this proof works.

U runs the following steps to authenticate herself:

(a) The user selects a random number $r_4 \in Z_p$ to generate a pseudonym $\text{Nym}_u^v = g_1^{r_4} \cdot h_{nym}^{usk}$ for communication with service providers.
(b) U picks random numbers $d, t_2 \longleftarrow Z_p$ and computes a randomized commitment credential (like in the previous step) as $\sigma' \longleftarrow (\sigma_1^{r_2}, (\sigma_2 \cdot \sigma_1^{t_2})^{r_2})$.
(c) Then, U calculates $D = g_1^d$, a secret session key $\text{SK} = Z^d = g_1^{y\cdot d}$ and $\text{Hmac}(\text{SK}, D, Z)$.
(d) For a set of credentials $\overrightarrow{\text{PC}}$, U computes an accumulator and witness as $AC = \text{Accumulate}(\text{params}, \overrightarrow{\text{PC}})$ and $\omega = \text{GenWitness}(\text{params}, \overrightarrow{\text{PC}}, \text{PC}_i^*)$, carries out a zero-knowledge proof of knowledge of the credential, and outputs the following proof of knowledge π such that

$$NIZK\begin{cases} (usk, \omega, d, t_2, m, r_4): \\ \text{AccVerify}(\text{params}, AC, \omega) = 1 \wedge \\ e(\sigma_1', Y)^{usk} \cdot e(\sigma_1', g_2)^{t_2} \cdot (\sigma_1', Y)^m \\ = \dfrac{e(\sigma_2', g_2)}{e(\sigma_1', X)} \wedge \\ \text{PC}_i = Enc_{usk}(\sigma) \wedge D = g_1^d \wedge \\ \text{Nym}_u^v = g_1^{r_4} \cdot h_{nym}^{usk} \end{cases} \qquad (8)$$

Finally, U sends the message $M_4 = \{\text{Nym}_u^v, D, \text{Hmac}, \pi\}$, to the service provider.

(i) After receiving the message $M_4 = \{\text{Nym}_u^v, D, \text{Hmac}, \pi\}$, from the user, the service provider first scans through the ledger to obtain a set $\overrightarrow{\text{PC}}$ consisting of all credentials belonging to IDP. First, SP computes the accumulator $AC = \text{Accumulate}(\text{params}, \overrightarrow{\text{PC}})$. Then, it verifies that $\pi = 1$ is the aforementioned proof of knowledge on PC_i and Nym_u^v using the known public values. If the proof verifies successfully, output 1, SP computes the session key as follows: $\text{SK} = D^y = g_1^{y\cdot d}$.

Then, SP computes $\text{Hmac}^*(\text{SK}, D, Z)$ and checks $\text{Hmac} = \text{Hmac}^*$. If $\pi = 1$ and Hmac holds, SP accepts SK as the session key and also the user is authentic.

Note that we can simply send σ' alongside the message of the proof of knowledge. With this, we can prove the construction is a Σ-protocol (see Ref. [47] to see how proof of knowledge of PS signature works).

4.6. Optimization. To exploit the accumulator AC in our construction which can be computed incrementally, we consider that any node mining a new block can add this block's accumulator to the previous one. The node stores the result as a new accumulator value in the transaction at the beginning of the new block, namely, the accumulator checkpoint. Peer nodes validate this computation before accepting the new block into the blockchain. With this optimization, SP no longer needs to compute the accumulator AC. Instead, SP can merely reference the current block's accumulator checkpoint and compute the secret key SK starting from the checkpoint preceding her mint (instead of starting at the beginning).

Theorem 1. *Our proposed protocol is secure against any nonuniform PPT adversary corrupting $t-1$ many personal agents PA by assuming that the El-Gamal encryption, zero-knowledge proof of signature, and the TOPRF protocol are secure and also the hash function is collision resistant.*

4.7. Security Proofs of Theorem 1

4.7.1. Proof Sketch. Our construction DAMFA is modular and relies directly on the TOPRF and the zero-knowledge proof. The security is then straightforwardly inherited from those algorithms:

The credential security requires that no adversary is able to present a credential (guess passwords and biometrics) and generate a session key, which they have not had any access to. If we use a TOPRF on passwords and biometric of users, then the security properties of TOPRF would make it hard to guess. The proof is once again twofold:

(i) First, the authentication is done through a zero-knowledge proof. At this step, the adversary presents an invalid credential or manages to build a valid proof. Hence, the adversary breaks the soundness of the underlying proof of knowledge we used, or else uses a valid credential.
(ii) At this step, we now assume the adversary wins by using a valid credential. We now rely on the obliviousness of the TOPRF. We interact with a TOPRF challenge to answer every adversarial request, and at the end, we can use the (valid) credential output by the adversary to break the TOPRF obliviousness, which leads to the conclusion.

4.7.2. Anonymity. During the registration phase, when a user reveals her pseudonym but does not (intentionally) reveal her secret key usk, no adversary should learn any information about the secret key or the identity. Besides, during the authentication phase, a user proves her credential using zero-knowledge proof, which reveals no additional information about her secret key and identity to the SP.

The simulator $\mathcal{S}$ is essentially an ideal world adversary that interacts with the functionality F_{DAMFA} and the environment ξ. We also assume that our zero-knowledge signature of knowledge includes an efficient extractor and a simulator and also that the signature is unforgeable. To guarantee that the view of the environment in the ideal world is indistinguishable from its view in the real world, it has to invoke the real-world adversary $\mathcal{A}$ by simulating all other entities for $\mathcal{A}'$. Then, for the most parts, the simulator follows the action of adversary $\mathcal{A}'$ appropriately.

4.7.3. Description of the Simulator. Once the adversary registers a new user to the system via storing a tuple $(\mathrm{Nym}_u^o, \mathrm{PC}_i, \pi_i)$ to the bulletin board, the simulator registers this user in the ideal world via the following process. It makes an interface between honest parties in the real world (which are the user U and $n-t+1$ personal agents denoted by PA_i where $i = t, \ldots, n$ wlog. since all personal agents in our solution are identical) and corrupted parties in the ideal world (which are the service provider SP and t personal agents denoted by PA_{ic} where $ic = (1, \ldots, t)$. The simulator behaves as follows:

(1) Registration

(1) Upon receiving $(\mathrm{Reg}, \mathrm{sid}, U, \mathrm{SI})$ from F_{DAMFA}, ignores it if $|\mathrm{SI}| \neq PA_n$. Otherwise, records $\langle U, \mathrm{sid}, \mathrm{SI}\rangle$ and sends $(\mathrm{Send}, (\mathrm{sid}, 0), U, PA, \mathrm{SI})$ to $\mathcal{A}'$ for all $PA \in \mathrm{SI}$. If F_{TOPRF} sends $(K, \mathrm{pw}, \mathrm{Bio})$, records it.

Remark 1. Since $\mathcal{S}$ simulates PA_{ic} in the ideal world, $\mathcal{S}$ receives whatever they receive from F_{DAMFA}.

(2) After receiving $(\mathrm{sid}, \mathrm{PA}_{\mathrm{ic}}, \mathrm{PC}_i, \mathrm{Nym}_u^o, \pi_i)$ from $\mathcal{A}'$ for some $PA_i \in \mathrm{SI}$, it checks if it has a record of $(U, k_{\mathrm{ic}}, \mathrm{Nym}_u^o)$ on its list of users. If the user with Nym_u^o exists, then $\mathcal{S}$ retrieves K associated with $(U, k_{\mathrm{ic}}, \mathrm{Nym}_u^o)$ and proceeds. The simulator then employs the knowledge extractor to obtain usk. If it is not on the list, $\mathcal{S}$ follows the protocol to register Nym_u^o as a user by choosing a random password pw^* and Bio^*. It generates secret shares k_{ic}' on K for each corrupted personal agent, records $\langle \mathrm{Reg}, U, \mathrm{sid}, \mathrm{SI}, \mathrm{pw}^*, \mathrm{Bio}^*, k_{\mathrm{ic}}, K\rangle$, and sends $\langle k_{\mathrm{ic}}\rangle$ to $PA_{\mathrm{ic}} \in \mathrm{SI}$ and $\mathcal{A}'$.

(3) Upon receiving $(\mathrm{RegComplete}, \mathrm{sid}, \mathrm{SI})$ from $\mathcal{A}'$, retrieves

$\langle \mathrm{Reg}, U, \mathrm{sid}, \mathrm{SI}, \mathrm{pw}^*, \mathrm{Bio}^*, k_{\mathrm{ic}}, K\rangle$ computes a pseudonym Nym_u^v and a credential $\mathrm{PC}_i' = h \cdot g^{usk}$ where $usk_{\mathrm{ic}} = F_K(\mathrm{pw}^*, \mathrm{Bio}^*)$. It records $\langle \mathrm{Nym}_u^v, \mathrm{PC}_i', U, \mathrm{SI}, usk_{\mathrm{ic}}\rangle$ and sends $(\mathrm{sid}, \mathrm{PA}_{\mathrm{ic}}, \mathrm{PC}_i, \mathrm{Nym}_u^v, \pi_i)$ to its public ledger and $\mathcal{A}'$ where π_i is proof of knowledge. $\mathcal{S}$ stores $(\mathrm{pw}^*, \mathrm{Bio}^*, K, usk_{\mathrm{ic}}, \mathrm{Nym}_u^v, \mathrm{PC}_i', \pi_i)$ in its list of granted credentials.

Remark 2. When an honest user wants to establish a credential through the functionality, the simulator creates a credential and uses the extractor of the signature of knowledge to simulate the associated proof. It then transmits the credential information $(\mathrm{PC}_i', \pi_i, \mathrm{Nym}_u^v)$ to the trusted store.

(2) Authentication

(1) Upon receiving $(\mathrm{Auth}, U^*, \mathrm{sid}, \mathrm{ssid}, \mathrm{SR})$ where $|\mathrm{SR}| \geq t+1$ from $\mathcal{A}'$, retrieves $\langle \mathrm{Nym}_u^v, \mathrm{PC}_i', U, \mathrm{SI}, usk_{\mathrm{ic}}\rangle$ corresponding to U as stored in the registration phase. If there is a set $(\mathrm{Bio}, \mathrm{pw}, K)$ stored in the registration phase and usk_{ic} is defined, then executes the TOPRF protocol with each personal agent using the password pw^* and Bio^* and receives $\rho_{\mathrm{ic}} = T(p, (\mathrm{pw}^*, \mathrm{Bio}^*))$ from F_{TOPRF} and sends $(\mathrm{Auth}, \mathrm{sid}, \mathrm{ssid}, U, \mathrm{SR})$ to $\mathcal{A}'$.

Remark 3. The initialization also specifies a parameter p used to identify a table $T(p, .)$ of random values that define the proper PRF values computed by the user when interacting with any subset of $t+1$ honest servers from the set SI. An additional parameter p^*, and corresponding tables $T(p^*, .)$, can be specified by the adversary to represent rogue tables with values computed by the user in the interaction with corrupted servers (see more on this [49]).

(2) Upon receiving $(\mathrm{Auth}, \mathrm{sid}, \mathrm{ssid}, U, \rho_{\mathrm{ic}})$ from F_{TOPRF}, $\mathcal{S}$ recovers SR and usk_{ic} corresponding to U as stored during the registration phase in the database (ignores this message if no corresponding tuples exist). $\mathcal{S}$ checks $\rho_{\mathrm{ic}} = usk_{\mathrm{ic}}$ and if each PA_{ic} used the correct corresponding $share_{\mathrm{ic}} = (usk_{\mathrm{ic}}, k_{\mathrm{ic}})$ values. Ignores this message if either of the following conditions fails: if $\rho_{\mathrm{ic}} = usk_{\mathrm{ic}}$ then $|S| tx(p, S) > 0| > t$ or all servers in SR are honest. Otherwise, sends $(\mathrm{Auth}, \mathrm{sid}, \mathrm{SR}, \mathrm{pw}^*, \mathrm{Bio}^*, sk)$ to $F_{DA\ MFA}$ where sk is a random secret key and sets for $(flag, \mathrm{pw}^*, \mathrm{Bio}^*, sk)$ as follows:

(a) Case 1: Correct $\mathrm{share}_{ic} = (\rho_{ic}, k_{ic})$ employed by the adversary in the real protocol. $\mathcal{S}$ detects this by verifying that $usk_{\mathrm{ic}} = \rho_{\mathrm{ic}}$. Therefore, $\mathcal{S}$ sets $(flag, \mathrm{pw}^*, \mathrm{Bio}^*, sk) = (1, ., .)$ and sends $(usk_{\mathrm{ic}}, k_{\mathrm{ic}})$ in its database to F_{DAMFA} where $usk_{\mathrm{ic}}, k_{\mathrm{ic}}$ was sent by F_{DAMFA}.

(b) Case 2: Otherwise, incorrect $usk_{\mathrm{ic}}, k_{\mathrm{ic}}$ employed by the adversary in the real protocol. $\mathcal{S}$ detects this by verifying that $usk_{\mathrm{ic}} \neq \rho_{\mathrm{ic}}$. So, $\mathcal{S}$ sets $(flag, \mathrm{pw}^*, \mathrm{Bio}^*, sk) = (0, ., .)$ and defines x as the set of values pw and Bio in the dictionary such that $T(p^*, (\mathrm{pw}, \mathrm{Bio}))$ is defined. For every x in lexicographic order, sets $v := T(p^*, x)$ and checks if $v = usk_{\mathrm{ic}}$. If so, sets $(\mathrm{flag}, \mathrm{pw}^*, \mathrm{Bio}^*, sk) := (2, x, sk^*)$ and breaks the loop. If the above loop processes all pw and

Bio without breaking, sets $(\text{flag}, \text{pw}^*, \text{Bio}^*, sk) = (0, ., .)$.

(3) On receiving $(\text{Auth}, \text{sid}, \text{ssid}, \text{SR}, x = \{pw^*, \text{Bio}^*\})$ from party $P \in (U, \mathcal{A}')$ and $(\text{Auth}, \text{sid}, \text{ssid}, P, \rho_{\text{ic}})$ from $\mathcal{A}'$, recovers usk_{ic} corresponding to U as stored in step 1. It ignores this message if either of the following conditions fails: If $\rho_{\text{ic}} = usk_{\text{ic}}$ then $|S|tx(p, S) > 0| > t$ or if all servers in SR are honest. Otherwise, picks $T(p^*, x) \longleftarrow \{0, 1\}^l$ if it has not been defined and sends $(\text{Auth}, \text{sid}, \text{ssid}, T(p^*, x))$ to $\mathcal{A}'$. If $\rho_{\text{ic}} = usk_{\text{ic}}$ (without resulting in the failure of conditions) then adds every $PA \in \text{SR}$ to tested(x) and sends $(\text{TestPwBio}, \text{sid}, PA, \text{pw}^*, \text{Bio}^*)$ to F_{DAMFA}. If F_{DAMFA} replies sk, then records it.

Remark 4. F_{DAMFA} employs the ideal user-provided password and biometric test in the ideal world. Therefore, if the adversarial personal agents in the real world acted honestly, it means that the simulator provided correct pairs (usk_i, k_i). Then, the calculated credential and pseudonymous will be valid (consisting in the ledger) since it is computed using the actual password and biometric. On the other hand, if personal agents acted maliciously in the real world, $\mathcal{S}$ would have detected this in the previous step and would have provided wrong pairs to F_{DAMFA} in the ideal world. So, in both worlds, the response will be invalid.

(4) Upon receiving $(\text{Auth}, \text{sid}, \text{ssid}, \text{SR}, \text{Nym}_u^v, \text{PC}_i)$ from F_{DAMFA}, $\mathcal{S}$ forwards $\langle \text{Nym}_u^v, \text{PC}_i \rangle$ to the $\mathcal{A}'$ in the real world.

(3) The Indistinguishability

(i) $Game_{Real}$. This is the real world: the system constructed in this work is run between $n - t + 1$ honest parties and t parties controlled by the adversary.

(ii) $Game_1$. This is identical to $Game_{Real}$ except that the encryption generated in the registration phase by honest users is replaced with a simulated one. Indistinguishability between $Game_{Real}$ and $Game_1$ comes from the El-Gamal encryption security properties.

(iii) $Game_2$. This is identical to $Game_1$ except that in TOPRF, each share (b_i and usk) generated by honest users using an actual password pw and biometric Bio is replaced by pw* and Bio* chosen randomly. Since, $\mathcal{S}$ does not have the correct password and biometric, indistinguishability between $Game_1$ and $Game_2$ comes from the indistinguishability of the TOPRF algorithm and TSS construction.

(a) Reduction 1. The TOPRF security ensures that senders (adversarial personal agents) cannot distinguish between the receiver (the simulated user) input, whether they are the actual password pw and Bio or another randomly chosen pair of password pw* and biometric Bio*.

(b) Reduction 2. The TSS security ensures that less than the threshold number of agents cannot reconstruct the secret and also cannot check if the shares are indeed related to the same secret. Therefore, there is no efficient way for the adversary to distinguish this from real behavior since one more agent needs to be corrupted to mount a successful offline attack.

(iv) $Game_3$. This game is identical to $Game_2$ except that an authentication response (Nym_o^v and PC_i^*), which are two random group elements generated by the adversary will be rejected if the extracted secret key does not fulfill the requirements. Indistinguishability between $Game_2$ and $Game_3$ comes from the verified consistency of the bilinear pairing algorithm and the simulation breaks the soundness of the underlying proof of knowledge we used before (assuming that there is no hash collision).

(v) $Game_4$. This is the world simulated by $\mathcal{S}$. It is not hard to check that $Game_{\text{ideal}}$ is identical to $Game_4$.

We already know that the possibility of TOPRF and NIZK proofs to break is negligible.

5. Implementation

In this section, we illustrate the practicability of the proposed protocol. To this end, we provide the public ledger part which is realized by well-known blockchains, namely, Namecoin and Ethereum. The results are summarized in Table 1. Here, initial data size shows the size of the blockchain needed for downloading and storage. Initial sync time is the time required to sync and connect to the blockchain. Confirmation time is the time required to confirm that the data are uploaded in the blockchain.

5.1. Namecoin Implemention. The public ledger can be implemented by a blockchain system. One of the smooth ways to realize a public ledger is using Namecoin blockchain. Namecoin allows for registering names and stores related values in the blockchain, which is a securely distributed shared database. It also enables a basic feature to query the database and to retrieve the list of existing names and associated data. Thus, we can store credentials, scan them based on namespace, and then verify them. We execute the following steps in order to participate in the Namecoin system and store credentials by the namecoin id as pseudonyms:

(i) We need to install a Namecoin client that has a full copy of the Namecoin blockchain and keep it in sync with the P2P network by fetching and validating new blocks from connected peers. We use implementation of the Namecoin client [64], which can be controlled by HTTP JSON-RPC, command line, or graphical interface. It spontaneously connects to the Namecoin network and downloads the blockchain.

(ii) The Namecoin client also creates the user's wallet, which includes the private key of Namecoin address of the user.

Table 1: Comparison of public ledger instantiations.

Properties	Namecoin	Ethereum (Rinkeby)
Initial data size	≈5.08 GB	≈5.3 GB
Initial sync time	≈3 h	≈3 h
Cost	0.069 USD	0.022 5 USD
Confirmation time	10 min/2 h	a few seconds/3 min

(iii) To save credentials in the blockchain, the user needs to register a namespace "id/name" as the owner of the name by paying a very small fee (currently 0.006 4 USD). An id name can be registered using the Namecoin graphical interface or commands "name_new" and "name_firstupdate." The following description shows how the id name in Namecoin namespace is registered and how those names can be accessed.

```
namecoind name-new id/3608a30756b0...
The output will look like this:
["0e0e03510b0b0b7dbba6e301e519693f6.
8062121b29f3cd3a6652c238360d0d0",
"9f213ff4a582fd65"].
```

This transaction shows a hashed version of the name, salted with a random value (which is "9f213..." for transaction ID "0e0e0351...").

(iv) The user can store arbitrary data as descriptions (which contains a credential) for Namecoin keys using the JSON format: the following codes can be a simple example of the JSON value of an identity name:

```
namecoind name_firstupdate id/3608...
Output:
{"description": "28790de641755e77d1.
3382229156f5c26a9dd8a9673006b...",
"namecoin": "NBvmSUQbRGu..."}
```

Subsequently, the update has been confirmed and transactions have been added to the blockchain. The user has a fully valid credential. To show the credential, *SP* scans through the list of added names and retrieves all credentials via a graphical interface or commands like the following code:

```
namecoind name_list
Output:
[{"name": "id/3608a30756b07e...",
"value": "28790de641755e77d13382.
229156f5c26a9dd8a9673006b15...",
"address": "NBvmSUQbRGunCS...",
"expires_in": 36000}].
```

(1) Cost:

Initially, a reasonable transaction fee of either 0.00 or 0.01 NMC is charged. We can choose this fee based on how fast we want to process a transaction.

(2) Latency:

Namecoin and Bitcoin both attempt to generate blocks every 10 minutes; on average, it takes nearly 5 minutes to see the data appear on the blockchain. In practice, it then takes the necessary time to solidify the transactions and the data to be verified. For Namecoin, it takes about 2 hours to confirm that the data are uploaded in the blockchain (12 confirmations). That is why name_firstupdate will only be accepted after a mandatory waiting period of 12 additional blocks.

Remark 5. Note that these costs and delays occur only once during the setup and registration phases. They do not affect the authentication phase. Thus, we focus on the computation time of the authentication phase that is frequently used in the authentication system (see Section 5.3).

5.2. Ethereum.

Ethereum allows us to test our decentralized application on a local blockchain; we use a test network called Rinkeby to build our decentralized application. We can connect to the Ethereum blockchain and even perform operations such as mine blocks, send transactions, and deploy smart contracts by running an Ethereum node.

(i) We run the Ethereum wallet (minst or geth command line) in order to access to Ethereum protocol and deploy our smart contract.

(ii) To start, we need to sink the Rinkeby network locally and download blockchain which takes a few hours.

(iii) Create an account:

```
Enter a password for your Rinkeby
Account by geth command line or.
Ethereum graphic (Minst).
Geth Version: 1.8.1-stable.
creates an account using geth
command: geth account new.
```

(iv) Next, obtain some Ether so that transactions can be sent. Since we used the Rinkeby testnet, their Ether can be obtained for free at the faucet website. Ether is used to pay transaction fees.

(v) We can deploy smart contracts to store our credentials and names into them. For this purpose, we write our first smart contract in Solidity (Solidity is a high-level contract language that is planned to target the Ethereum Virtual Machine (EVM)) and deploy it through Mist. A simple example code is

```
pragma solidity 0.4.2

contract Test {
string public $NYM$;
string public $Z$;

function Test(string $–NYM$, string $–Z$)
{
v1 = $–NYM$;
v2 = $–Z$;
}
}
```

(vi) We can also see the option to watch previously deployed contracts and tokens. We can click on “Watch Contracts” at the bottom and enter the contract’s name and contract address.

(1) Cost:

All transactions need some amount of gas to motivate processing. A transaction fee is between 0 and 0.000 424 ETHER depending on how fast we want to approve the blockchain transaction.

(2) Latency:

Ethereum creates a new block every few seconds so that the data will appear on the blockchain instantly. As mentioned in Ethereum Blog, 10 confirmations are sufficient to achieve a similar security degree as that of 6 confirmations in Bitcoin. It takes around 3 minutes to confirm the transaction/data. Note that these costs and delays occur only once during the setup and registration phases.

5.3. Performance of the Authentication System. We now examine the performance of our anonymous authentication system. There are two main steps: the registration phase and the authentication phase. However, since time-critical operations in both registration and authentication phases are the same, we concentrate our evaluation on the efficiency of these processes. These processes include OPRF, issuing/receiving a credential, and proving knowledge of the signature and pseudonym. To simplify the evaluation criteria of the experiment results, we only assume a simple policy with a threshold $t = 2$ for two agents. The experiment is based on a laptop with Intel Core i5-6200U CPU 2.30 GHz, 8.00 GB RAM, and 64-bit Ubuntu OS in Java 8, building upon the upb.crypto library (available at https://github.com/cryptimeleon) [65]. This library offers elliptic curve math and several useful building blocks for anonymous credentials like Sanders signatures [47], Pedersen’s commitment [58], Nguyen’s accumulator [66], Shamir secret sharing, generalized Schnorr protocols, proofs of partial knowledge [67], Damgård’s technique for concurrently black-box secure Sigma protocols, and the Fiat-Shamir heuristic [56]. Table 2 shows the computational performances of the protocols over 50 iterations. For issuing and proving protocols in such a way that a certain policy is satisfied by a credential, we assume equality of two attributes as Policy: StuID = “11111” and GENDER = “male” and credential: certifying only these attributes.

5.4. Computational and Communication Complexity. We analyze the communication and the computation complexity of our proposed protocol using the size of each element exchange involved in our protocol, the number of exponentiation needed for issuing a credential (executed only once in the registration phase) and the proving of a credential (the most frequently executed phase), respectively. We show the following efficiency analysis in Table 3. r, t, $E_{\mathbb{G}_1}$, and P denote the number of attributes that can be certified, the number of agents that need to be connected, the cost of exponentiation in $\mathbb{G}_1$, and the cost of a pairing computation, respectively. By $\mathrm{POK}\{E_{\mathbb{G}_1}[n]\}$ (resp. $\mathrm{POK}\{P[n]\}$), we denote the cost of proving knowledge of n secrets involved in a multi-exponentiation (resp. pairing-product) equation, and $Ver(POK)$ indicates the cost of verifying this proof.

Table 2: Performance of the authentication protocol.

Sub-protocol	Duration (ms)
OPRF	30
ProveNym	6
IssCred	25
ProveCred	33

5.5. Comparison. We provide a comparison of DAMFA with some of the most popular SSO schemes in Table 4. We compare DAMFA with the above schemes in terms of Decentralization (Decent.), Passive verification (PV), Multi-Factor (MF), Formal definitions (FD), Anonymity (Anony.), and Selective Disclosure (SD) attributes. Decent denotes the decentralization of the SSO schemes (i.e., user authentication process no longer depends on a trusted third party). We provide this by applying a distributed transaction ledger and the blind issuing protocol. PV shows that service providers can verify users (who have registered a particular credential) without requiring interaction with an identity provider. We fulfill this property using a distributed transaction ledger and anonymous credentials. Anonymity guarantees that no one can trace or learn information about the user’s identity during the authentication process. We fulfill this property by applying NIZNP + SP signature + Pseudonym. Here, ● denotes that it is unfeasible for IDP’s to track users’ sign-on activity onto different SPs. Also, it shows that it is impossible to correlate multiple accounts created from the same credential on different SPs. Subsequently, ◐ indicates that either IDP’s s or SPs can create a correlation between different accounts of the same user. FD demonstrates if proposed schemes provide a formal security definition. In this case, DAMFA is the only scheme that provides a formal security definition and proof. SD allows to disclose a subset of user attributes and proves statements about their attributes. Finally, to protect the user’s private information against offline (OA) attacks, we use the TOPRF primitive. Here, ◐ means that other related schemes are resistant against offline attacks as long as IDP does not compromise or the theft/loss/corruption of a user’s device does not happen when they use this device as 2FA token. ● means that resistance to offline attacks is satisfied even in the presence of a corrupted IDP or user’s device.

Table 3: DAMFA computation and communication complexity.

Transaction	*TOPRF*		*IssueCred*		*ProveCred*	
	User	PA_i	*User*	IDP	*User*	SP
Computation	$2\ E_{\mathbb{G}_1}$	$E_{\mathbb{G}_1}$	$(r+1)E_{\mathbb{G}_1} + \mathrm{POK}\left\{E_{\mathbb{G}_1}[r+1]\right\}$	$2E_{\mathbb{G}_1} + \mathrm{Ver}(\mathrm{POK})$	$2E_{\mathbb{G}_1} + \mathrm{POK}\{P[r+1]\}$	$\mathrm{Ver}(\mathrm{POK})$
Communication	$(2t)\lvert\mathbb{G}_1\rvert$		$\lvert\mathbb{G}_1\rvert + \lvert\mathrm{POK}\rvert$		$2\lvert\mathbb{G}_1\rvert + \lvert\mathrm{POK}\rvert$	

Table 4: Comparison of single sign-on schemes.

Schemes	Decent.	PV	OA	Anony.	MF	FD	SD
SAML [19]	×	×	◐	◐	✓	×	×
OpenID [16]	×	×	◐	◐	×	×	×
PRIMA [17]	×	×	◐	◐	×	×	≈
IRMA [68]	✓	≈	◐	●	✓	×	✓
EL PASSO [45]	≈	✓	◐	●	✓	×	✓
NextLeap [46]	✓	×	◐	◐	✓	×	✓
DAMFA	✓	✓	●	●	✓	✓	✓

Anonymity: NextLeap relies on unlinkable credentials. However, blinded credentials should be stored at IDP, which allows IDP to perform user tracking. Also, in PRIMA, signon across multiple SPs can be linked. Other schemes do not support unlinkable credentials. —Offline attacks ◐: the related schemes only fulfilled offline attack if IDP is honest. In IRMA, the user's device (i.e., IRMA app) should be secure to provide OA and anonymity. Otherwise, any adversary who gets these can simply impersonate the user (we addressed this open problem in IRMA). —Selective disclosure: PRIMA supports proving statements about attributes, particularly when they are displayed as extra attributes signed by IDP. — Passive verification ≈ : In IRMA, SPs still require to interact with an IRMA API server during the authentication.

6. Conclusion

In this paper, we proposed a decentralized authentication and key exchange system DAMFA (SSO scheme) under TOPRF protocol and standard cryptographic primitives. The proposed scheme builds upon a trustworthy global append-only ledger that does not rely on a trusted server. DAMFA fulfills the following properties:

(1) Decentralization property means that the process of user authentication no longer depends on a trusted party. To realize such a distributed ledger, we propose using the blockchain system already in real-world use with the cryptographic currency Bitcoin.

(2) Passive verification means that service providers who have access to the shared ledger can verify users without requiring interaction with an identity provider.

(3) Single sign-on property ensures that a user logs in with a single ID into the identity provider and then gains access to any of the several related systems. So, users do not need to register with each service provider individually.

(4) Anonymity guarantees that no one can trace or learn information about the user's identity during the authentication process. Finally, we evaluated that our protocol is efficient and practical for authentication systems.

Moreover, we provided comparison of our scheme (DAMFA) with some of the most prominent SSO schemes. To demonstrate a more detailed analysis of the performance of our scheme, we analyzed the communication and the computation complexity of our proposed protocol using the size of each element's exchange involved in our protocol and the number of exponentiation, respectively. We proved our construction's security via ideal-real simulation, showing the impossibility of offline dictionary attacks. Finally, we demonstrated that our protocol is efficient and practical through a prototypical implementation and implemented the public ledger using Ethereum and Namecoin blockchains.

Data Availability

No additional data are available.

Disclosure

This paper is an extended version of the paper entitled "DAMFA: Decentralized Anonymous Multi-Factor Authentication" [23], including complete proofs, formal security models, an Ethereum implementation, a comparison with other SSO schemes, a computation and communication complexity analysis, and improved experimental results.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Johannes Kepler Open Access Publishing Fund and has been carried out within the scope of Digidow, the Christian Doppler Laboratory for Private Digital Authentication in the Physical World. It has partially been supported by the LIT Secure and Correct Systems Lab. The authors are grateful for the financial support by the

Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, the Christian Doppler Research Association, 3 Banken IT GmbH, ekey biometric systems GmbH, Kepler Universitätsklinikum GmbH, NXP Semiconductors Austria GmbH and Co KG, Österreichische Staatsdruckerei GmbH, and the State of Upper Austria.

References

[1] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Proceedings of the International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT)*, pp. 139–155, Springer, Prague, Czech Republic, May 2000.

[2] D. Cameron, "Over 560 million passwords discovered in anonymous online database (2017)," 2017, https://goo.gl/upDqzt.

[3] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2776–2791, 2017.

[4] V. Goel and N. Perlroth, "Yahoo says 1 billion user accounts were hacked," 2016, https://goo.gl/q4WZi9.

[5] Gemalto, "Year of mega breaches & identity theft: findings from the breach level index," 2014, https://www.sbs.ox.ac.uk/cybersecurity-capacity/content/2014-year-mega-breaches-and-identity-theft.

[6] I. Paul, "LinkedIn confirms account passwords hacked," 2012, http://goo.gl/UBWuY0.

[7] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, and Y. Koucheryavy, "Multi-factor authentication: a survey," *Cryptography*, vol. 2, no. 1, p. 1, 2018.

[8] O. Mir, R. Mayrhofer, M. Hölzl, and T.-B. Nguyen, "Recovery of encrypted mobile device backups from partially trusted cloud servers," *in Proceedings of the 13th International Conference on Availability*, pp. 1–10, Reliability and Security, 2018.

[9] S. Jarecki, H. Krawczyk, M. Shirvanian, and N. Saxena, "Two-factor Authentication with end-to-end password security," in *Proceedings of the International Conference on Practice and Theory of Public Key Cryptography (PKC)*, Springer, Rio de Janeiro, Brazil, Cham, March 2018.

[10] M. A. Mustafa, A. Abidin, and E. A. Rúa, "Frictionless authentication system: security & privacy analysis and potential solutions," 2018, https://arxiv.org/abs/1802.07231.

[11] V. Rimmer, D. Preuveneers, W. Joosen, M. A. Mustafa, A. Abidin, and E. A. Rúa, "Frictionless authentication systems: emerging trends, Research challenges and opportunities," 2018, https://arxiv.org/abs/1802.07233.

[12] D. Q. Viet, A. Yamamura, and H. Tanaka, "Anonymous password-based authenticated key exchange," in *Proceedings of the International Conference on Cryptology in India (INDOCRYPT)*, pp. 244–257, Springer, Bangalore, India, December 2005.

[13] Y. Lindell, "Anonymous authentication," *Journal of Privacy and Confidentiality*, vol. 2, no. 2, 2011.

[14] J. Yang and Z. Zhang, "A new anonymous password-based authenticated key exchange protocol," in *Proceedings of the International Conference on Cryptology in India (INDOCRYPT)*, pp. 200–212, Springer, Bangalore, India, December 2008.

[15] S. Shin, K. Kobara, and H. Imai, "Anonymous password-authenticated key exchange: new construction and its extensions," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E93-A, no. 1, pp. 102–115, 2010.

[16] D. Recordon and D. Reed, "Openid 2.0: a platform for user-centric identity management," in *Proceedings of the second ACM workshop on Digital identity management*, pp. 11–16, New York, NY, United States, November 2006.

[17] M. R. Asghar, M. Backes, and M. Simeonovski, "Prima: privacy-preserving identity and access management at internet-scale," in *Proceedings of the 2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, Kansas City, MO, USA, May 2018.

[18] D. Hardt and M. Jones, "The oauth 2.0 authorization framework: bearer token usage," 2012, https://www.omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.b_stds:tech:ietf:oauth_bearer.

[19] OneLogin Inc, "SAML toolkits," 2019, https://developers.onelogin.com/saml.

[20] S. Agrawal, P. Miao, P. Mohassel, and P. Mukherjee, "Pasta: password-based threshold authentication," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pp. 2042–2059, ACM, New York, NY, United States, October 2018.

[21] G. Alpár, J.-H. Hoepman, and J. Siljee, "The identity crisis. security, privacy and usability issues in identity management," 2011, https://arxiv.org/abs/1101.0427.

[22] D. Fett, R. Küsters, and G. Schmitz, "The web SSO standard openid connect: in-depth formal security analysis and security guidelines," in *Proceedings of the Computer Security Foundations Symposium (CSF)*, pp. 189–202, IEEE, Santa Barbara, CA, USA, August 2017.

[23] O. Mir, M. Roland, and R. Mayrhofer, "Damfa: decentralized anonymous multi-factor authentication," in *Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure, ser. BSCI '20*, pp. 10–19, New York, NY, United States, October 2020.

[24] S. M. Bellovin and M. Merritt, "Encrypted key exchange: password-based protocols SecureAgainst dictionary attacks," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, IEEE, Oakland, CA, USA, May 1992.

[25] Augmented Encrypted Key Exchange, "A password-based protocol secure against dictionary attacks and password file compromise," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pp. 244–250, ACM, New York, NY, United States, October 1993.

[26] V. Boyko, P. MacKenzie, and S. Patel, "Provably secure password-authenticated key exchange using diffie-hellman," in *Proceedings of the International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT)*, pp. 156–171, Springer, Bruges, Belgium, May 2000.

[27] O. Goldreich and Y. Lindell, "Session-key generation using human passwords only," *Journal of Cryptology*, vol. 19, no. 3, pp. 241–340, 2006.

[28] R. Gennaro and Y. Lindell, "A framework for password-based authenticated key exchange," *Lecture Notes in Computer Science*, Springer, in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, pp. 524–543, May 2003.

[29] A. Groce and J. Katz, "A new framework for efficient password-based authenticated key exchange," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pp. 516–525, ACM, New York, NY, United States, October 2010.

[30] F. Benhamouda and D. Pointcheval, "Verifier-based password-authenticated key exchange: new models and constructions," *IACR Cryptology ePrint Archive*, vol. 2013, p. 833, 2013.

[31] M. Shirvanian, S. Jarecki, N. Saxena, and N. Nathan, "Two-factor authentication resilient to server compromise using mix-bandwidth devices," in *Proceedings of the Network and Distributed System Security Symposium, (NDSS), San Diego, CA, USA*, February 2014.

[32] J. Bringer, H. Chabanne, and R. Lescuyer, "Software-only two-factor authentication secure against active servers," in *Proceedings of the International Conference on Cryptology in Africa (AFRICACRYPT)*, pp. 285–303, Springer, Fes, Morocco, April 2016.

[33] O. Mir, T. van der Weide, and C.-C. Lee, "A secure user anonymity and authentication scheme using avispa for telecare medical information systems," *Journal of Medical Systems*, vol. 39, no. 9, p. 89, 2015.

[34] S. Jarecki, H. Krawczyk, M. Shirvanian, and N. Saxena, "Device-enhanced password protocols with optimal online-offline protection," in *Proceedings of the ACM on Asia Conference on Computer and Communications Security (ASIA CCS)*, pp. 177–188, ACM, New York, NY, United States, October 2016.

[35] D. He and D. Wang, "Robust biometrics-based authentication scheme for multiserver environment," *IEEE Systems Journal*, vol. 9, no. 3, pp. 816–823, 2015.

[36] X. Huang, Y. Xiang, E. Bertino, J. Zhou, and L. Xu, "Robust multi-factor authentication for fragile communications," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 6, pp. 568–581, 2014.

[37] D. Pointcheval and S. Zimmer, "Multi-factor Authenticated key exchange," in *Proceedings of the 6th International Conference on Applied Cryptography and Network Security (ACNS)*, pp. 277–295, Springer, New York, NY, United States, June 2008.

[38] O. Mir and M. Nikooghadam, "A secure biometrics based authentication with key agreement scheme in telemedicine networks for e-health services," *Wireless Personal Communications*, vol. 83, no. 4, pp. 2439–2461, 2015.

[39] N. Fleischhacker, M. Manulis, and A. Azodi, "A modular framework for multi-factor authentication and key exchange," in *Proceedings of the International Conference on Research in Security Standardisation*, pp. 190–214, Springer, London, United Kingdom, December 2014.

[40] R. Zhang, Y. Xiao, S. Sun, and H. Ma, "Efficient multi-factor authenticated key exchange scheme for mobile communications," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, 2017.

[41] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted online password guessing: an underestimated threat," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 1242–1254, Vienna Austria, October 2016.

[42] D. Wang, D. He, P. Wang, and C.-H. Chu, "Anonymous two-factor authentication in distributed systems: certain goals are beyond attainment," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 4, pp. 428–442, 2014.

[43] D. Debiao He, N. Kumar, J.-H. Jong-Hyouk Lee, and R. S. Sherratt, "Enhanced three-factor security protocol for consumer usb mass storage devices," *IEEE Transactions on Consumer Electronics*, vol. 60, no. 1, pp. 30–37, 2014.

[44] Z. Zhang, K. Yang, X. Hu, and Y. Wang, "Practical anonymous password authentication and TLS with anonymous client authentication," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pp. 1179–1191, ACM, Vienna Austria, October 2016.

[45] Z. Zhang, M. Król, A. Sonnino, L. Zhang, and E. Rivière, "El passo: privacy-preserving, asynchronous single sign-on," 2020, https://arxiv.org/abs/2002.10289.

[46] H. Halpin, "Nextleap: decentralizing identity with privacy for secure messaging," in *Proceedings of the 12th International Conference on Availability*, pp. 1–10, Reliability and Security, Inria de Paris, December 2017.

[47] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *Proceedings of the Cryptographers' Track at the RSA Conference*, pp. 111–126, Springer, San Francisco, CA, USA, March 2016.

[48] S. Jarecki and X. Liu, "Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection," in *Proceedings of the Conference on Theory of Cryptography (TCC)*, pp. 577–594, Springer, San Francisco, CA, USA, March 2009.

[49] S. Jarecki, A. Kiayias, H. Krawczyk, and J. Xu, "TOPPSS: cost-minimal password-protected secret sharing based on threshold OPRF," in *Proceedings of the International Conference on Applied Cryptography and Network Security*, pp. 39–58, Springer, Singapore, June 2017.

[50] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[51] E. B. Sasson, A. Chiesa, C. Garman et al., "Zerocash: decentralized anonymous payments from bitcoin," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pp. 459–474, IEEE, Berkeley, CA, USA, May 2014.

[52] C. Garman, M. Green, and I. Miers, "Decentralized anonymous credentials," in *Proceedings of the 2014 Network and Distributed System Security Symposium, Internet Society, San Diego, California,*, February 2014.

[53] C. Fromknecht, D. Velicanu, and S. Yakoubov, *A Decentralized Public Key Infrastructure with Identity Retention*, IACR Cryptology ePrint Archive, 2014, https://eprint.iacr.org/2014/803.pdf.

[54] R. Yang, M. H. Au, Q. Xu, and Z. Yu, "Decentralized blacklistable anonymous credentials with reputation," *Computers & Security*, vol. 85, pp. 353–371, 2019.

[55] C.-P. Schnorr, "Efficient identification and signatures for smart cards," in *Proceedings of the Annual International Cryptology Conference on Advances in Cryptology (CRYPTO)*, pp. 239–252, Springer, Aarhus, Denmark, May 1990.

[56] A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," in *Proceedings of the Conference on the Theory and Application of Cryptographic Techniques (Eurocrypt)*, pp. 186–194, Springer, Santa Barbara, CA, USA, March 1986.

[57] J. Camenisch, M. Kohlweiss, and C. Soriente, "An accumulator based on bilinear maps and efficient revocation for anonymous credentials," in *Proceedings of the International Workshop on Public Key Cryptography*, pp. 481–500, Springer, Irvine, CA, USA, March 2009.

[58] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proceedings of the Annual International Cryptology Conference (Crypto)*, pp. 129–140, Springer, Santa Barbara, CA, USA, August 1991.

[59] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, pp. 295–310, Springer, Prague, Czech Republic, May 1999.

[60] R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. MacKenzie, "Universally composable password-based key exchange," in *Proceedings of the Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pp. 404–421, Springer, Aarhus, Denmark, May 2005.

[61] J. Camenisch, M. Drijvers, and B. Tackmann, "Multi-protocol uc and its use for building modular and efficient protocols," *IACR Cryptol. ePrint Arch*, vol. 2019, p. 65, 2019.

[62] A. T. B. Jin, D. N. C. Ling, and A. Goh, "Biohashing: two factor authentication featuring fingerprint data and tokenised random number," *Pattern Recognition*, vol. 37, no. 11, pp. 2245–2255, Nov 2004.

[63] M. Chase, S. Meiklejohn, and G. Zaverucha, "Algebraic macs and keyed-verification anonymous credentials," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1205–1216, ACM, Scottsdale Arizona USA, November 2014.

[64] N. team, *Namecoind, Sourcecode of the Namecoin-Client Reference Implementation,* Namecoin, Aug 2016, https://github.com/namecoin/namecoin.

[65] K. Bemmann, J. Blömer, J. Bobolz et al., "Fully-featured anonymous credentials with reputation system," in *Proceedings of the 13th International Conference on Availability, Reliability and Security (ARES 2018), no. 42*, pp. 1–10, IEEE, New York, NY, United States, August 2018.

[66] L. Nguyen, "Accumulators from bilinear pairings and applications," in *Lecture Notes in Computer Science*, vol. 3376, pp. 275–292, Springer, 2005.

[67] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Advances in Cryptology – CRYPTO '94, ser. LNCS,* vol. 839, pp. 174–187, Springer, 1994.

[68] G. Alpár, F. van den Broek, B. Hampiholi, B. Jacobs, W. Lueks, and S. Ringers, "Irma: practical, decentralized and privacy-friendly identity management using smartphones," *HotPETs*, vol. 2017, 2017.