Split Credential Authentication: A Privacy-Preserving Protocol with Decoupled Authority and Issuer Roles*

Nuttapong Attrapadung[1], Goichiro Hanaoka[1], Keisuke Hara[1], Ryuya Hayashi[1], Junichiro Hayata[2], Masaki Kamizono[2], Hiroshi Kumagai[2], Takahiro Matsuda[1], Kenta Nomura[2], Tsunekazu Saito[2], and Yuta Takata[2]†

[1] National Institute of Advanced Industrial Science and Technology (AIST), Koto-ku, Tokyo, Japan
[2] Deloitte Tohmatsu Cyber LLC, Chiyoda-ku, Tokyo, Japan

Abstract

Attribute-based authentication (ABA) is a cryptographic protocol which enables access control based on user-specific attributes such as age, affiliation, or location. While this approach offers fine-grained authorization, conventional ABA schemes require users to disclose all attributes in their credentials, posing significant privacy risks. Anonymous credentials (AC) address this issue by allowing users to hide their attributes during issuance and selectively disclose them during authentication. However, existing AC models assume that users interact directly with issuers, which creates practical challenges: issuers are expected to issue credentials without knowing whether the underlying attribute should be authorized. This design raises both security and accountability concerns and often necessitates centralized attribute management, which is undesirable in real-world settings.

In this paper, we propose *Split Credential Authentication* (SCA), a new cryptographic framework that separates attribute management and credential issuance. This separation better reflects real-world institutional settings, where authorities managing user attributes (e.g., municipalities or hospitals) are typically distinct from certificate issuers. At the core of SCA, we introduce a novel cryptographic primitive called *Oblivious Certificate Generation* (OCG), which enables certificate issuance without revealing attribute contents to the issuer, nor linking certificates to specific users from the authority's perspective. We provide a formal definition of OCG and its construction based on standard digital signatures and blind signature schemes satisfying a novel property called splittability. Then, we formalize SCA and its construction based on OCG and non-interactive zero-knowledge proofs to enable selective attribute disclosure. Finally, we demonstrate the applicability of SCA in sensitive domains such as disability services, where it reduces the privacy burden on users while preserving verifiability and policy compliance.

Keywords: Attribute-Based Authentication; Anonymous Credential; Blind Signatures.

1 Introduction

Attribute-based authentication (ABA) is a cryptographic protocol for controlling access to systems or services based on user-specific information, such as age, address, or affiliation. Roughly, in ABA, the process begins with a user registering with a certificate authority (CA). During this initial phase, the user's associated attributes are verified by the CA. Upon successful verification, the CA issues an attribute certificate that securely asserts the user's attribute. After

*Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. 68, December 16-18, 2025, Sapporo , Japan.

†Corresponding author

receiving the certificate, the user can use the certificate to show that it has the corresponding attribute properly. In recent years, the need for utilizing such attribute information has been growing across the world. For example, the World Wide Web Consortium (W3C), a global standardization organization, has been working on standardizing Verifiable Credentials (VCs) [2].

In conventional ABA, one major privacy concern lies in the full disclosure of all attribute information contained within a user's credential. When a user presents her credential to a verifier (service provider), all attributes, regardless of whether they are relevant to the access policy, are typically revealed. This all-or-nothing disclosure model poses a serious threat to privacy, especially in scenarios where sensitive attributes (e.g., health status or organizational affiliation) are embedded in the credential. As one of the primary solutions to this privacy problem, the concept of anonymous credentials (AC) (e.g., [4, 5, 6, 8, 11, 12, 13, 14, 29]) has been proposed. Roughly, compared to ABA, AC offers privacy for users in both the issuance and showing phases. First, during issuance, a user can obtain a credential from the CA without revealing attributes, thereby keeping the user's attribute information hidden from the CA. Second, in the showing phase, the user can demonstrate its possession of only the minimal required attributes, that is, disclose just what is necessary to satisfy the verifier's policy without revealing any additional attribute information.

1.1 Motivation

Although AC is an attractive cryptographic primitive for anonymous authentication, we have the following practical problems when introducing AC in the real world.

As seen above, due to the system model of AC, users can ask an issuer to issue any certificate directly while hiding its attribute to the issuer. From an issuer's perspective, this is not prefer in some cases since it cannot confirm whether it should issue a certificate regarding the hidden attribute for the user. A straightforward solution to this limitation would be for the user to reveal the attribute directly to the issuer. However, this approach not only raises serious privacy concerns for the user, but also results in the issuer gaining centralized access to all attribute information—an undesirable situation from the perspective of data leakage and information control.

Toward resolving the above problem, we focus on a new model of anonymous authentication where *attribute authorities* who manage attributes for users and *certificate issuer* who is responsible only for issuing certificates are separated. This separation allows us to divide institutional responsibilities clearer and is considered to reflect real-world settings more accurately. Specifically, in real-world institutional contexts, the entities responsible for managing attributes (e.g., municipalities, medical institutions, welfare agencies) and those responsible for credential issuance (e.g., certification authorities) are often distinct. We can see that this separation structurally prevents users from unilaterally asserting arbitrary attributes. Concretely, since the certificate issuer only issues credentials based on the verifications received from an (authorized) attribute authorities, the model enforces a binding relationship between verification and issuance. This design mitigates a key limitation of prior anonymous credential schemes, which users could potentially obtain certificates for unassigned attributes.

Why Separation of Roles Remains Necessary. One might consider simply assigning the role of the issuer to the attribute authority as well. However, due to legal requirements and operational costs, it is often impractical for attribute authorities to directly issue certificates by themselves. For example, the level of legal assurance and the range of services that a certificate issuer can support depend on its classification. In this case, strict recognitions are required for certification services with strong legal backing, such as those approved by governmental authorities. This imposes significant barriers for many organizations.

In addition to such a legal issue, the management of dedicated hardware infrastructure imposes a considerable operational burden for attribute authorities. In practical use cases of ABA, CAs are often required to rely on external servers operated by highly trusted providers that meet strict security standards. This reliance introduces not only financial costs associated with secure server hosting, but also significant overhead in terms of operational complexity and system maintenance. Furthermore, the responsibility of securely storing and managing sensitive information, such as cryptographic keys for issuing certificates, within these environments constitutes a non-negligible barrier, particularly for attribute authorities with limited administrative resources.

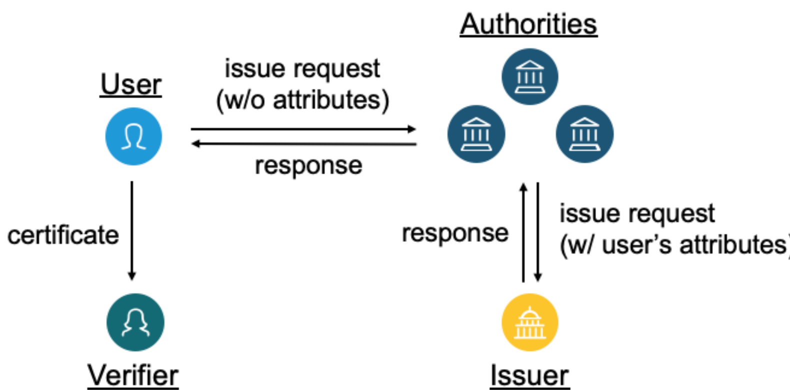

Figure 1: The model of an SCA protocol

1.2 Our Contributions

Based on the above motivation, we propose a new cryptographic protocol called a *split credential authentication* (SCA) protocol, in which the roles of managing users' attributes and issuing certificates are split into distinct authorities. The model of SCA is depicted in Figure 1. More specifically, we have the following three contributions regarding SCA.

1. As a core building block for SCA, we introduce a new cryptographic primitive called *oblivious certificate generation* (OCG), which might be of independent interest. Concretely, we propose the formalization of OCG and its construction based on a (standard) digital signature scheme (e.g., [10, 26]) and a (two-round) blind signature scheme (e.g., [9, 15, 16, 17, 18, 19, 20]) satisfying a special property called splittability.

   Roughly, an OCG protocol is conducted among three entities: a user, an authority, and an issuer. The user initiates the protocol by requesting a certificate from the authority, which verifies the user's attribute `attr` and, if valid, forwards the request to the issuer. The issuer generates an "oblivious" issuance message that ensures two privacy properties: (1) the issuer learns nothing about `attr` (attribute hiding) and (2) the authority cannot link the issued certificate to the user (certificate hiding). The user then derives the certificate using their private information. See Section 3 for the details.

2. We propose the formalization and construction of SCA. Our SCA protocol can be obtained by combining the above OCG protocol and a non-interactive zero-knowledge (NIZK) proof system (e.g., [7, 21, 22, 25]) to allow users to show that they have certificates satisfying required policies for verifiers.

   An SCA protocol is conducted among a user, an authority, an issuer, and a verifier and separated into two phases: One is a certificate generating phase and the other is certificate showing phase. A certificate generating phase is processed among a user, an authority, and an issuer, and the goal of this phase is the same as the above OCG protocol. After obtaining a certificate for an attribute attr, in the certificate showing phase, a user can show a proof that it has a certificate satisfying the policy C requested by a verifier. Here, as security requirements, we need that a user cannot forge a proof not satisfying the policy C (unforgeability) and a verifier cannot obtain the information about attr beyond the fact that $C(\mathsf{attr})$ is valid (privacy). See Section 4 for the details.

3. Finally, we demonstrate the practical relevance of our SCA protocol through real-world applications, particularly in contexts involving sensitive attribute verification, such as disability-related services. Our model enables selective disclosure without revealing underlying personal details, thereby reducing barriers to access caused by privacy concerns, stigma, or discrimination. See Section 5 for the details.

2 Preliminaries

In this section, we recall some notations and cryptographic primitives.

2.1 Notations

In this paper, $x \leftarrow X$ denotes sampling an element x from a finite set X uniformly at random. $y \leftarrow \mathcal{A}(x; r)$ denotes that a probabilistic algorithm $\mathcal{A}$ outputs y for an input x using a randomness r, and we simply denote $y \leftarrow \mathcal{A}(x)$ when we need not write an internal randomness explicitly. Also, $x := y$ denotes that x is defined by y. λ denotes a security parameter. A function $f(\lambda)$ is a negligible function in λ, if $f(\lambda)$ tends to 0 faster than $\frac{1}{\lambda^c}$ for every constant $c > 0$. $\mathsf{negl}(\lambda)$ denotes an unspecified negligible function. PPT stands for probabilistic polynomial time. $\emptyset$ denotes an empty set. If n is a natural number, $[n]$ denotes the set of integers $\{1, \cdots, n\}$. Also, if a and b are integers such that $a \leq b$, $[a, b]$ denotes the set of integers $\{a, \cdots, b\}$.

2.2 Signatures

Here, we recall the definition of signatures.

Definition 2.1 (Signatures)**.** *A signature scheme* SIG *with a message space* $\mathbb{M}$ *consists of the following three PPT algorithms.*

$\mathsf{SIG.KG}(1^\lambda) \rightarrow (\mathsf{vk}, \mathsf{sigk})$: *The key generation algorithm, given a security parameter* 1^λ*, outputs a verification key* vk *and a signing key* sigk*.*

$\mathsf{SIG.Sign}(\mathsf{sigk}, m) \to \sigma$: *The (deterministic) signing algorithm, given a signing key* sigk *and a message* m*, outputs a signature* σ*.*

$\mathsf{SIG.Ver}(\mathsf{vk}, m, \sigma) \to 1/0$: *The (deterministic) verification algorithm, given a verification key* vk*, a message* m*, and a signature* σ*, outputs either* 1 *(accept) or* 0 *(reject).*

As the correctness for SIG*, we require that* $\mathsf{SIG.Ver}(\mathsf{vk}, m, \mathsf{SIG.Sign}(\mathsf{sigk}, m)) = 1$ *holds for all* $\lambda \in \mathbb{N}$*,* $m \in \mathbb{M}$*, and* $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{SIG.KG}(1^\lambda)$*.*

We require that a signature scheme satisfies EUF-CMA security defined as follows.

Definition 2.2 (EUF-CMA security)**.** *Let* $\mathsf{SIG} = (\mathsf{SIG.KG}, \mathsf{SIG.Sign}, \mathsf{SIG.Ver})$ *be a signature scheme. We say that* SIG *satisfies EUF-CMA security if for any PPT adversary* $\mathcal{A}$*, the advantage defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{SIG},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{c} L_{sig} := \emptyset, \\ (\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{SIG.KG}(1^\lambda), \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sign}}(\mathsf{sigk}, \cdot)}(\mathsf{vk}) \end{array} : \begin{array}{c} \mathsf{SIG.Ver}(\mathsf{vk}, m^*, \sigma^*) = 1 \\ \wedge\ m^* \notin L_{sig} \end{array}\right],$$

where $\mathcal{O}_{\mathsf{sign}}(\mathsf{sigk}, \cdot)$ *is a signing oracle which takes a message* m *as input, outputs* $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sigk}, m)$ *to* $\mathcal{A}$*, and appends* m *to* L_{sig}*.*

2.3 Non-Interactive Zero-Knowledge Proof System

Here, we recall the definition of a non-interactive zero-knowledge (NIZK) proof system [22].

Definition 2.3 (NIZK Proof System)**.** *A non-interactive zero-knowledge (NIZK) proof system* NIZK *for an NP relation* $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ *consists of the following three PPT algorithms.*

$\mathsf{NIZK.Setup}(1^\lambda) \to \mathsf{crs}$: *The setup algorithm, given a security parameter* 1^λ*, outputs a common reference string* crs*.*

$\mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{X}, \mathsf{W}) \to \pi$: *The prove algorithm, given a common reference string* crs *and a pair of a statement and a witness* $(\mathsf{X}, \mathsf{W}) \in \mathcal{R}$*, outputs a proof* π*.*

$\mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{X}, \pi) \to 1/0$: *The verify algorithm, given a common reference* crs*, a statement* X*, and a proof* π*, outputs either* 1 *(accept) or* 0 *(reject).*

As the correctness for NIZK*, we require that* $\mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{X}, \pi) = 1$ *holds for all* $\lambda \in \mathbb{N}$*,* $(\mathsf{X}, \mathsf{W}) \in \mathcal{R}$*,* $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$*, and* $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{X}, \mathsf{W})$*.*

We require that an NIZK proof system satisfies CRS indistinguishability, zero-knowledge, and extractability.

Definition 2.4 (CRS Indistinguishability)**.** *Let* $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Ver})$ *be an NIZK proof system. For any PPT adversary* $\mathcal{A}$*, the advantage defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{crs}}_{\mathsf{NIZK},\mathcal{A}}(\lambda) := \left|\begin{array}{l} \Pr[\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda) :\ \mathcal{A}(\mathsf{crs}) = 1] \\ \quad - \Pr[(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Ext}_0(1^\lambda) : \mathcal{A}(\mathsf{crs}) = 1] \end{array}\right|.$$

Definition 2.5 (Zero-Knowledge). *Let* $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Ver})$ *be an NIZK proof system. Let* $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ *be a zero-knowledge simulator for* NIZK. *For any PPT adversary* $\mathcal{A}$, *the advantage defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathcal{A}}(\lambda) := \left| \begin{array}{l} \Pr[\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda) : \ \mathcal{A}^{\mathcal{P}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1] \\ \quad - \Pr[(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}_0(1^\lambda) : \mathcal{A}^{\mathcal{S}(\mathsf{crs},\mathsf{td},\cdot,\cdot)}(\mathsf{crs}) = 1] \end{array} \right|,$$

where $\mathcal{P}$ *and* $\mathcal{S}$ *are oracles that on input* (X, W) *return* $\perp$ *if* $(\mathsf{X}, \mathsf{W}) \notin \mathcal{R}$ *and otherwise return* $\mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{X}, \mathsf{W})$ *and* $\mathsf{Sim}_1(\mathsf{crs}, \mathsf{td}, \mathsf{X})$, *respectively.*

Definition 2.6 (Extractability). *Let* $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Ver})$ *be an NIZK proof system. Let* $\mathsf{Ext} = (\mathsf{Ext}_0, \mathsf{Ext}_1)$ *be an extractor for* NIZK. *For any PPT adversary* $\mathcal{A}$, *the advantage defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{ext}}_{\mathsf{NIZK},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{c} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Ext}_0(1^\lambda), \\ (\mathsf{X}^*, \pi^*) \leftarrow \mathcal{A}, \\ \mathsf{W}^* \leftarrow \mathsf{Ext}_1(\mathsf{crs}, \mathsf{td}, \mathsf{X}^*) \end{array} : \begin{array}{c} \mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{X}^*, \pi^*) = 1 \\ \wedge\ ((\mathsf{X}^*, \mathsf{W}^*) \notin \mathcal{R}) \end{array} \right].$$

2.4 Blind Signature

Here, we recall the definition of two-round (i.e., round-optimal) blind signature [23].

Definition 2.7 (Blind Signature). *A blind signature scheme* BS *with a message space* $\mathbb{M}$ *consists of the following five algorithms.*

$\mathsf{BS.KG}(1^\lambda) \rightarrow (\mathsf{pk}, \mathsf{sk})$: *The key generation algorithm, given a security parameter* 1^λ, *outputs a public key* pk *and a signing key* sk.

$\mathsf{BS}.\mathcal{U}_1(\mathsf{pk}, m) \rightarrow (\mu, \mathsf{st}_\mathcal{U})$: *The user's first message generation algorithm, given a public key* pk *and a message* $m \in \mathbb{M}$, *outputs a first message* μ *and a state* $\mathsf{st}_\mathcal{U}$.

$\mathsf{BS}.\mathcal{S}_2(\mathsf{sk}, \mu) \rightarrow \rho$: *The signer's second message generation algorithm, given a signing key* sk *and a first message* μ, *and outputs a second message* ρ.

$\mathsf{BS}.\mathcal{U}_{\mathsf{der}}(\mathsf{st}_\mathcal{U}, \rho) \rightarrow \sigma$: *The user's signature derivation algorithm, given a state* $\mathsf{st}_\mathcal{U}$ *and a second message* ρ, *and outputs a signature* σ.

$\mathsf{BS.Ver}(\mathsf{pk}, m, \sigma) \rightarrow 1/0$: *The (deterministic) verification algorithm, given a public key* pk, *a message* $m \in \mathbb{M}$, *and a signature* σ, *and outputs either* 1 *(accept) or* 0 *(reject).*

As the correctness for BS, *we require that* $\mathsf{BS.Ver}(\mathsf{pk}, m, \sigma) = 1$ *holds for any* $\lambda \in \mathbb{N}$, $m \in \mathbb{M}$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{BS.KG}(1^\lambda)$, $(\mu, \mathsf{st}_\mathcal{U}) \leftarrow \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}, m)$, $\rho \leftarrow \mathsf{BS}.\mathcal{S}_2(\mathsf{sk}, \mu)$, *and* $\sigma \leftarrow \mathsf{BS}.\mathcal{U}_{\mathsf{der}}(\mathsf{st}_\mathcal{U}, \rho)$.

We require that a blind signature scheme BS satisfies unforgeability and blindness defined as follows.

Definition 2.8 (Unforgeability). *Let* $\mathsf{BS} = (\mathsf{BS.KG}, \mathsf{BS}.\mathcal{U}_1, \mathsf{BS}.\mathcal{S}_2, \mathsf{BS}.\mathcal{U}_{\mathsf{der}}, \mathsf{BS.Ver})$ *be a blind signature scheme.* BS *satisfies unforgeability if for any* $q = \mathsf{poly}(\lambda)$ *and PPT adversary* $\mathcal{A}$ *that makes at most* q *queries, the advantage defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{BS},\mathcal{A}}(\lambda) :=$$

$$\Pr\left[\begin{array}{c} L_{sig} \coloneqq \emptyset, \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{BS.KG}(1^\lambda), \\ \{(m_i, \sigma_i)\}_{i \in [q+1]} \leftarrow \mathcal{A}^{\mathsf{BS}.\mathcal{S}_2(\mathsf{sk},\cdot)}(\mathsf{pk}) \end{array} : \begin{array}{c} \mathsf{BS.Ver}(\mathsf{pk}, m_i, \sigma_i) = 1 \text{ for all } i \in [q+1] \\ \wedge \{m_i\}_{i \in [q+1]} \text{ is pairwise distinct} \end{array}\right],$$

where we say that $\{m_i\}_{i\in[q+1]}$ *is pairwise distinct if we have* $m_i \neq m_j$ *for all* $i \neq j$*.*

Definition 2.9 (Blindness)**.** *Let* $\mathsf{BS} = (\mathsf{BS.KG}, \mathsf{BS}.\mathcal{U}_1, \mathsf{BS}.\mathcal{S}_2, \mathsf{BS}.\mathcal{U}_{\mathsf{der}}, \mathsf{BS.Ver})$ *be a blind signature scheme.* BS *satisfies blindness if for any PPT adversary* $\mathcal{A}$*, the advantage defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{blind}}_{\mathsf{BS},\mathcal{A}}(\lambda) \coloneqq \left|\Pr\left[\begin{array}{c} (\mathsf{pk}, m_0, m_1) \leftarrow \mathcal{A}(1^\lambda), \\ \forall b \in \{0,1\} : (\mu_b, \mathsf{st}^b_{\mathcal{U}}) \leftarrow \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}, m_b), \\ \mathsf{coin} \xleftarrow{\$} \{0,1\}, \\ (\rho_{\mathsf{coin}}, \rho_{1-\mathsf{coin}}) \leftarrow \mathcal{A}(\mu_{\mathsf{coin}}, \mu_{1-\mathsf{coin}}), \\ \forall b \in \{0,1\} : \sigma_b \leftarrow \mathsf{BS}.\mathcal{U}_{\mathsf{der}}(\mathsf{st}^b_{\mathcal{U}}, \rho_b), \\ \text{If } \sigma_0 = \bot \text{ or } \sigma_1 = \bot : (\sigma_0, \sigma_1) \coloneqq (\bot, \bot), \\ \mathsf{coin}' \leftarrow \mathcal{A}(\sigma_0, \sigma_1) \end{array} : \mathsf{coin} = \mathsf{coin}'\right] - \frac{1}{2}\right|.$$

In addition to the above standard properties, we introduce a new property called *splittability*. Roughly, this property requires that the algorithm $\mathsf{BS}.\mathcal{U}_1$ can be divided into two sub-algorithms: one is a randomness generation algorithm (which is independent of a message) and the other is a message hiding algorithm.

Definition 2.10 (Splittability)**.** *Let* $\mathsf{BS} = (\mathsf{BS.KG}, \mathsf{BS}.\mathcal{U}_1, \mathsf{BS}.\mathcal{S}_2, \mathsf{BS}.\mathcal{U}_{\mathsf{der}}, \mathsf{BS.Ver})$ *be a blind signature scheme,* $m \in \mathbb{M}$*, and* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{BS.KG}(1^\lambda)$*. We say that* BS *satisfies splittability if the following three properties hold.*

1. *The algorithm* $\mathsf{BS}.\mathcal{U}_1$*, given a public key* pk *and a message* m*, can split to the following two algorithms.*

   $\mathsf{BS}.\mathcal{U}_1.\mathsf{Rand}(\mathsf{pk}) \rightarrow (r, \mathsf{st}_{\mathcal{U}})$: *The randomness generation algorithm, given a public key* pk*, and outputs a randomness* r *and a state* $\mathsf{st}_{\mathcal{U}}$*.*

   $\mathsf{BS}.\mathcal{U}_1.\mathsf{Hide}(r, m) \rightarrow \mu$: *The (deterministic) message hiding algorithm, given a randomness* r *and a message* m*, and outputs a first message* μ*.*

2. *For* $(\mu, \mathsf{st}_{\mathcal{U}}) \leftarrow \mathsf{BS}.\mathcal{U}_1(\mathsf{pk}, m)$*,* $(r, \mathsf{st}'_{\mathcal{U}}) \leftarrow \mathsf{BS}.\mathcal{U}_1.\mathsf{Rand}(\mathsf{pk})$*, and* $\mu' \leftarrow \mathsf{BS}.\mathcal{U}_1.\mathsf{Hide}(r, m)$*, two distributions* $\{(\mathsf{pk}, \mathsf{sk}, \mu)\}$ *and* $\{(\mathsf{pk}, \mathsf{sk}, \mu')\}$ *are identical.*

3. *The two distributions* $\{r : (r, \mathsf{st}_{\mathcal{U}}) \leftarrow \mathsf{BS}.\mathcal{U}_1.\mathsf{Rand}(\mathsf{pk})\}$ *and* $\{r : r \xleftarrow{\$} \mathcal{R}\}$ *are statistically indistinguishable, where* $\mathcal{R}$ *is a randomness space for* $\mathsf{BS}.\mathcal{U}_1$*.*

Note that some existing blind signature schemes [9, 15] satisfy splittability.

3 Oblivious Certificate Generation Protocol

In this section, we introduce a new cryptographic primitive called *oblivious certificate Generation* (OCG) protocol. Before providing a formal description of an OCG protocol, we give a rough explanation of this primitive.

An OCG protocol is used among three entities: user, authority, and issuer. Firstly, a user makes a query to an authority to ask publishing a certificate. Given a user's query, an authority confirms the user's attribute attr and decides whether it publishes a certificate of attr for the user. If an authority publishes a certificate for attr, it makes a query to an issuer to authenticate the certificate. Given a query by an authority, an issuer generates an *oblivious* message to get a certificate of attr for the user. Here, "oblivious" has two meanings: (1) an issuer cannot get any information of attr (called *attribute hiding* later) and (2) an authority cannot know which certificate is corresponding to the user by itself (called *certificate hiding* later). After getting a message from the issuer, the user can obtain its certificate for attr by using its secret information.

In addition to the above attribute/certificate hiding, we require the following two properties. One is *impersonation resilience* which ensures that a (malicious) user cannot impersonate an authority to get a certificate for some attribute attr' which should not be allowed to publish for the user. The other is *unforgeability* which ensures that a (malicious) user/authority cannot forge a certificate which is not published by an issuer.

3.1 Formalization

In this section, we provide a formalization of OCG.

Definition 3.1 (Oblivious Certificate Generation Protocol)**.** *An oblivious certificating generation protocol* OCG *with an attribute space* $\mathbb{A}$ *consists of the following algorithms.*

$\mathsf{OCG.Set}(1^\lambda) \to \mathsf{pp}$: *The setup algorithm, given a security parameter* 1^λ*, outputs a public parameter* pp*.*

We assume that the following algorithms take pp *as input implicitly.*

$\mathsf{OCG.KG}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: *The (issuer) key generation algorithm, given a security parameter* 1^λ*, outputs a public key* pk *and a signing key* sk*.*

$\mathsf{OCG.AKG}(1^\lambda) \to (\mathsf{apk}, \mathsf{ask})$: *The (authority) key generation algorithm, given a security parameter* 1^λ*, outputs a public key* apk *and a signing key* ask*.*

$\mathsf{OCG.User}(\mathsf{pk}, \mathsf{apk}) \to (\mu, \mathsf{st}_{\mathcal{U}})$: *The user's first message generation algorithm, given an issuer's public key* pk *and an authority's public key* apk*, outputs a first message* μ *and a (secret) state* $\mathsf{st}_{\mathcal{U}}$*.*

$\mathsf{OCG.Aut}(\mathsf{ask}, \mu, \mathsf{attr}) \to \nu$: *The (deterministic) authority's message generation algorithm, given a secret key* ask*, a first message* μ*, and an attribute* attr*, outputs an authority's message* ν*.*

$\mathsf{OCG.AVer}(\mathsf{apk}, \nu) \to 1/0$: *The (deterministic) authority's message verification algorithm, given a public key* apk *and an authority's message* ν*, outputs either* 1 *(accept) or* 0 *(reject).*

$\mathsf{OCG.Man}(\mathsf{sk}, \nu) \to \tau$: *The issuer's message generation algorithm, given a signing key* sk *and a message* ν*, outputs an issuer's message* τ*.*

$\mathsf{OCG.Derive}(\mathsf{st}_{\mathcal{U}}, \tau) \to \mathsf{cert}$: *The user's certification derivation algorithm, given a state* $\mathsf{st}_{\mathcal{U}}$ *and an issuer's message* τ*, outputs a certificate* cert*.*

$\mathsf{OCG.Ver}(\mathsf{pk}, \mathsf{attr}, \mathsf{cert}) \to 1/0$: *The (deterministic) verification algorithm, given a public key* pk*, an attribute* $\mathsf{attr} \in \mathbb{M}$*, and a certificate* cert*, outputs either* 1 *(accept) or* 0 *(reject).*

As the correctness for OCG*, we require that* $\mathsf{OCG.Ver}(\mathsf{pk}, \mathsf{attr}, \mathsf{cert}) = 1$ *and* $\mathsf{OCG.AVer}(\mathsf{apk}, \nu) = 1$ *hold for any* $\lambda \in \mathbb{N}$*,* $\mathsf{attr} \in \mathbb{A}$*,* $\mathsf{pp} \leftarrow \mathsf{OCG.Set}(1^\lambda)$*,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{OCG.KG}(1^\lambda)$*,* $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{OCG.AKG}(1^\lambda)$*,* $(\mu, \mathsf{st}_\mathcal{U}) \leftarrow \mathsf{OCG.User}(\mathsf{pk})$*,* $\nu \leftarrow \mathsf{OCG.Aut}(\mathsf{ask}, \mu, \mathsf{attr})$*,* $\tau \leftarrow \mathsf{OCG.Man}(\mathsf{sk}, \nu)$*, and* $\mathsf{cert} \leftarrow \mathsf{OCG.Derive}(\mathsf{st}_\mathcal{U}, \tau)$*.*

We require that an OCG protocol OCG satisfies attribute hiding, certificate hiding, impersonation resilience against authority, and unforgeability defined as follows.[1]

Definition 3.2 (Attribute Hiding)**.** *Let* $\mathsf{OCG} = (\mathsf{OCG.Set}, \mathsf{OCG.KG}, \mathsf{OCG.AKG}, \mathsf{OCG.User}, \mathsf{OCG.Aut}, \mathsf{OCG.AVer}, \mathsf{OCG.Man}, \mathsf{OCG.Derive}, \mathsf{OCG.Ver})$ *be an OCG protocol.* OCG *satisfies attribute hiding if for any PPT adversary* $\mathcal{A}$*, the advantage defined as follows is negligible*

$$\mathsf{Adv}^{\mathsf{a\text{-}hide}}_{\mathsf{OCG},\mathcal{A}}(\lambda) := \left| \Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{OCG.Set}(1^\lambda), \\ (\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{OCG.AKG}(1^\lambda), \\ (\mathsf{pk}, \mathsf{attr}_0, \mathsf{attr}_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Aut}}(\mathsf{ask},\cdot,\cdot)}(\mathsf{pp}, \mathsf{apk}), \\ \forall b \in \{0,1\} : \\ \quad (\mu_b, \mathsf{st}^b_\mathcal{U}) \leftarrow \mathsf{OCG.User}(\mathsf{pk}), \\ \quad \nu_b \leftarrow \mathsf{OCG.Aut}(\mathsf{ask}, \mu_b, \mathsf{attr}_b), \\ \mathsf{coin} \xleftarrow{\$} \{0,1\}, \\ (\tau_{\mathsf{coin}}, \tau_{1-\mathsf{coin}}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Aut}}(\mathsf{ask},\cdot,\cdot)}(\nu_{\mathsf{coin}}, \nu_{1-\mathsf{coin}}), \\ \forall b \in \{0,1\} : \mathsf{cert}_b \leftarrow \mathsf{OCG.Derive}(\mathsf{st}^b_\mathcal{U}, \tau_b), \\ \text{If } \mathsf{cert}_0 = \bot \text{ or } \mathsf{cert}_1 = \bot : \\ \quad (\mathsf{cert}_0, \mathsf{cert}_1) := (\bot, \bot), \\ \text{Else } \mathsf{coin}' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Aut}}(\mathsf{ask},\cdot,\cdot)}(\mathsf{cert}_0, \mathsf{cert}_1) \end{array} : \mathsf{coin} = \mathsf{coin}' \right] - \frac{1}{2} \right|,$$

where $\mathcal{O}_{\mathsf{Aut}}(\mathsf{ask}, \cdot, \cdot)$ *is an authority's message oracle which takes a user's first message* μ *and an attribute* attr *as input and outputs* $\nu \leftarrow \mathsf{OCG.Aut}(\mathsf{ask}, \mu, \mathsf{attr})$*.*

Definition 3.3 (Certificate Hiding)**.** *Let* $\mathsf{OCG} = (\mathsf{OCG.Set}, \mathsf{OCG.KG}, \mathsf{OCG.AKG}, \mathsf{OCG.User}, \mathsf{OCG.Aut}, \mathsf{OCG.AVer}, \mathsf{OCG.Man}, \mathsf{OCG.Derive}, \mathsf{OCG.Ver})$ *be an OCG protocol.* OCG *satisfies certificate hiding if for any PPT adversary* $\mathcal{A}$*, the advantage defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{c\text{-}hide}}_{\mathsf{OCG},\mathcal{A}}(\lambda) := \left| \Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{OCG.Set}(1^\lambda), \\ (\mathsf{pk}, \mathsf{apk}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}), \mathsf{coin} \xleftarrow{\$} \{0,1\}, \\ \forall b \in \{0,1\} : (\mu_b, \mathsf{st}^b_\mathcal{U}) \leftarrow \mathsf{OCG.User}(\mathsf{pk}), \\ (\tau_{\mathsf{coin}}, \tau_{1-\mathsf{coin}}) \leftarrow \mathcal{A}(\mu_{\mathsf{coin}}, \mu_{1-\mathsf{coin}}, \mathsf{st}), \\ \forall b \in \{0,1\} : \\ \quad \mathsf{cert}_b \leftarrow \mathsf{OCG.Derive}(\mathsf{st}^b_\mathcal{U}, \tau_b), \\ \text{If } \mathsf{cert}_0 = \bot \text{ or } \mathsf{cert}_1 = \bot : \\ \quad (\mathsf{cert}_0, \mathsf{cert}_1) := (\bot, \bot), \\ \text{Else } \mathsf{coin}' \leftarrow \mathcal{A}(\mathsf{cert}_0, \mathsf{cert}_1) \end{array} : \mathsf{coin} = \mathsf{coin}' \right] - \frac{1}{2} \right|.$$

Definition 3.4 (Impersonation Resilience)**.** *Let* $\mathsf{OCG} = (\mathsf{OCG.Set}, \mathsf{OCG.KG}, \mathsf{OCG.AKG}, \mathsf{OCG.User}, \mathsf{OCG.Aut}, \mathsf{OCG.AVer}, \mathsf{OCG.Man}, \mathsf{OCG.Derive}, \mathsf{OCG.Ver})$ *be an OCG protocol.* OCG *satisfies impersonation resilience if for any PPT adversary* $\mathcal{A}$*, the advantage defined as follows is negligible:*

$$\mathsf{Adv}^{\mathsf{imp}}_{\mathsf{OCG},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{c} L_\nu := \emptyset, \\ \mathsf{pp} \leftarrow \mathsf{OCG.Set}(1^\lambda), \\ (\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{OCG.AKG}(1^\lambda), \\ ((\mu^*, \mathsf{attr}^*), \nu^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Aut}}(\mathsf{ask},\cdot,\cdot)}(\mathsf{pp}, \mathsf{apk}) \end{array} : \begin{array}{c} \mathsf{OCG.AVer}(\mathsf{apk}, \nu^*) = 1 \\ \wedge\ ((\mu^*, \mathsf{attr}^*), \cdot) \notin L_{cert} \end{array} \right],$$

[1]We note that certificate hiding is not needed to prove the security properties of our SCA protocol in Section 4, thanks to the zero-knowledge property of the underlying NIZK proof system. However, we introduce it here since certificate hiding is useful if we use it alone in another system.

where $\mathcal{O}_{\mathsf{Aut}}(\mathsf{ask},\cdot,\cdot)$ *is an authority's message oracle which takes a user's first message* μ *and an attribute* attr *as input, outputs* $\nu \leftarrow \mathsf{OCG.Aut}(\mathsf{ask},\mu,\mathsf{attr})$*, and appends* $((\mu,\mathsf{attr}),\nu)$ *into* L_{cert}*.*

Definition 3.5 (Unforgeability)**.** *Let* $\mathsf{OCG} = (\mathsf{OCG.Set},\mathsf{OCG.KG},\mathsf{OCG.AKG},\mathsf{OCG.User},\mathsf{OCG.Aut},\mathsf{OCG.AVer},\mathsf{OCG.Man},\mathsf{OCG.Derive},\mathsf{OCG.Ver})$ *be an OCG protocol.* OCG *satisfies unforgeability if for any* $q = \mathsf{poly}(\lambda)$ *and PPT adversary* $\mathcal{A}$ *that makes at most* q *queries, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{OCG},\mathcal{A}}(\lambda) :=$$
$$\Pr\left[\begin{array}{c} L_{cert} := \emptyset, \\ \mathsf{pp} \leftarrow \mathsf{OCG.Set}(1^\lambda), \\ (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{OCG.KG}(1^\lambda), \\ \{(\mathsf{attr}_i,\mathsf{cert}_i)\}_{i\in[q+1]} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Man}}(\mathsf{sk},\cdot)}(\mathsf{pp},\mathsf{pk}) \end{array} : \begin{array}{c} \forall i \in [q+1] : \mathsf{OCG.Ver}(\mathsf{pk},\mathsf{attr}_i,\mathsf{cert}_i) = 1 \\ \wedge\ \{\mathsf{attr}_i\}_{i\in[q+1]} \text{ is pairwise distinct} \end{array}\right],$$

where $\mathcal{O}_{\mathsf{Man}}(\mathsf{sk},\cdot)$ *is an issuer's message oracle which takes a authority's message* ν *as input, outputs* $\tau \leftarrow \mathsf{OCG.Man}(\mathsf{sk},\nu)$*, and appends* ν *into* L_{cert} *and we say that* $\{\mathsf{attr}_i\}_{i\in[q+1]}$ *is pairwise distinct if we have* $\mathsf{attr}_i \neq \mathsf{attr}_j$ *for all* $i \neq j$*.*

3.2 Construction

In this section, we provide a generic construction based on a splittable blind signature scheme and a (standard) signature scheme. Let $\mathsf{BS} = (\mathsf{BS.KG},\mathsf{BS}.\mathcal{U}_1,\mathsf{BS}.\mathcal{S}_2,\mathsf{BS}.\mathcal{U}_{\mathsf{der}},\mathsf{BS.Ver})$ be a splittable blind signature scheme and $\mathsf{SIG} = (\mathsf{SIG.KG},\mathsf{SIG.Sign},\mathsf{SIG.Ver})$ be a signature scheme. Based on BS and SIG, we give a description of our OCG protocol OCG as follows.

$\mathsf{OCG.Set}(1^\lambda)$: It outputs $\mathsf{pp} := 1^\lambda$.

$\mathsf{OCG.KG}(1^\lambda)$: It computes $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{BS.KG}(1^\lambda)$ and outputs $(\mathsf{pk},\mathsf{sk})$.

$\mathsf{OCG.AKG}(1^\lambda)$: It computes $(\mathsf{vk},\mathsf{sigk}) \leftarrow \mathsf{SIG.KG}(1^\lambda)$ and outputs $(\mathsf{apk},\mathsf{ask}) := (\mathsf{vk},\mathsf{sigk})$.

$\mathsf{OCG.User}(\mathsf{pk})$: It computes $(r,\mathsf{st}_{\mathcal{U}}) \leftarrow \mathsf{BS}.\mathcal{U}_1.\mathsf{Rand}(\mathsf{pk})$ and outputs $(\mu,\mathsf{st}_{\mathcal{U}}) := (r,\mathsf{st}_{\mathcal{U}})$.

$\mathsf{OCG.Aut}(\mathsf{ask},\mu,\mathsf{attr})$: It computes $\nu' \leftarrow \mathsf{BS}.\mathcal{U}_1.\mathsf{Hide}(\mu,\mathsf{attr})$ and $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sigk},\nu')$ and outputs $\nu := (\nu',\sigma)$.

$\mathsf{OCG.AVer}(\mathsf{apk},\nu)$: It computes $v \leftarrow \mathsf{SIG.Ver}(\mathsf{vk},\nu',\sigma)$ and outputs v.

$\mathsf{OCG.Man}(\mathsf{sk},\nu)$: It computes $\tau \leftarrow \mathsf{BS}.\mathcal{S}_2(\mathsf{sk},\nu')$ and outputs τ.

$\mathsf{OCG.Derive}(\mathsf{st}_{\mathcal{U}},\tau)$: It computes $\mathsf{cert} \leftarrow \mathsf{BS}.\mathcal{U}_{\mathsf{der}}(\mathsf{st}_{\mathcal{U}},\tau)$ and outputs cert.

$\mathsf{OCG.Ver}(\mathsf{pk},\mathsf{attr},\mathsf{cert})$: It computes $v \leftarrow \mathsf{BS.Ver}(\mathsf{pk},\mathsf{attr},\mathsf{cert})$ and outputs v.

Due to the correctness of BS and SIG, it is easy to see that the correctness of OCG holds. Moreover, impersonation resilience and unforgeability are immediately followed from the EUF-CMA security of SIG and the unforgeability of BS, respectively. Then, in the following, we show that our OCG protocol OCG satisfies attribute hiding and certificate hiding.

Theorem 3.6. *If* BS *satisfies splittability and blindness, then* OCG *satisfies attribute hiding.*

Proof of Theorem 3.6. Let $\mathcal{A}$ be any PPT adversary that attacks the attribute hiding of OCG. The attribute hiding game with respect to OCG is described as follows.

1. The challenger $\mathcal{CH}$ firstly generates $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{SIG.KG}(1^\lambda)$, sets $(\mathsf{apk}, \mathsf{ask}) := (\mathsf{vk}, \mathsf{sigk})$, and gives apk to $\mathcal{A}$.
2. When $\mathcal{A}$ makes a query of the form (μ, attr) to $\mathcal{O}_{\mathsf{Aut}}$, $\mathcal{CH}$ computes $\nu' \leftarrow \mathsf{BS}.\mathcal{U}_1.\mathsf{Hide}(\mu, \mathsf{attr})$ and $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sigk}, \nu')$ and returns $\nu := (\nu', \sigma)$ to $\mathcal{A}$.
3. Upon receiving $(\mathsf{pk}, \mathsf{attr}_0, \mathsf{attr}_1)$ from $\mathcal{A}$, $\mathcal{CH}$ computes $(r_b, \mathsf{st}^b_{\mathcal{U}}) \leftarrow \mathsf{BS}.\mathcal{U}_1.\mathsf{Rand}(\mathsf{pk})$, $\nu'_b \leftarrow \mathsf{BS}.\mathcal{U}_1.\mathsf{Hide}(r_b, \mathsf{attr}_b)$, and $\sigma_b \leftarrow \mathsf{SIG.Sign}(\mathsf{sigk}, \nu'_b)$ for $b \in \{0,1\}$, sets $\nu_b := (\nu'_b, \sigma_b)$ for $b \in \{0,1\}$, samples $\mathsf{coin} \leftarrow \{0,1\}$, and gives $(\nu_{\mathsf{coin}}, \nu_{1-\mathsf{coin}})$ to $\mathcal{A}$.
4. Upon receiving $(\tau_{\mathsf{coin}}, \tau_{1-\mathsf{coin}})$ from $\mathcal{A}$, $\mathcal{CH}$ computes $\mathsf{cert}_b \leftarrow \mathsf{BS}.\mathcal{U}_{\mathsf{der}}(\mathsf{st}^b_{\mathcal{U}}, \tau_b)$ for $b \in \{0,1\}$, sets $(\mathsf{cert}_0, \mathsf{cert}_1) := (\perp, \perp)$ if $\mathsf{cert}_0 = \perp$ or $\mathsf{cert}_1 = \perp$ holds, and gives $(\mathsf{cert}_0, \mathsf{cert}_1)$ to $\mathcal{A}$.
5. $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$ and terminates.

In the following, we show that there exists a PPT adversary $\mathcal{B}$ against the blindness of BS such that $\mathsf{Adv}^{\mathsf{a\text{-}hide}}_{\mathsf{OCG},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{blind}}_{\mathsf{BS},\mathcal{B}}(\lambda)$.

1. Upon receiving 1^λ from the challenger, $\mathcal{B}$ generates $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{SIG.KG}(1^\lambda)$ and gives vk to $\mathcal{A}$.
2. When $\mathcal{A}$ makes a query (μ, attr) to $\mathcal{O}_{\mathsf{Aut}}$, $\mathcal{B}$ proceeds in the same way as the challenger does in the above game.
3. Upon receiving $(\mathsf{pk}, \mathsf{attr}_0, \mathsf{attr}_1)$ from $\mathcal{A}$, $\mathcal{B}$ also sends $(\mathsf{pk}, \mathsf{attr}_0, \mathsf{attr}_1)$ to its challenger.
4. Upon receiving $(\mu_{\mathsf{coin}}, \mu_{1-\mathsf{coin}})$ from the challenger, $\mathcal{B}$ computes $\sigma_b \leftarrow \mathsf{SIG.Sign}(\mathsf{sigk}, \mu_b)$ for $b \in \{0,1\}$, sets $\nu_b := (\mu_b, \sigma_b)$ for $b \in \{0,1\}$, and gives $(\nu_{\mathsf{coin}}, \nu_{1-\mathsf{coin}})$ to $\mathcal{A}$.
5. Upon receiving $(\tau_{\mathsf{coin}}, \tau_{1-\mathsf{coin}})$ from $\mathcal{A}$, $\mathcal{B}$ sends $(\tau_{\mathsf{coin}}, \tau_{1-\mathsf{coin}})$ to its challenger.
6. Upon receiving $(\mathsf{cert}_0, \mathsf{cert}_1)$ from the challenger, $\mathcal{B}$ gives $(\mathsf{cert}_0, \mathsf{cert}_1)$ to $\mathcal{A}$.
7. When $\mathcal{A}$ outputs coin' and terminates, $\mathcal{B}$ returns coin' to its challenger and terminates.

We can see that $\mathcal{B}$ perfectly simulates the attribute hiding of OCG for $\mathcal{A}$. Note that, due to the splittability of BS, the distribution of $(\nu_{\mathsf{coin}}, \nu_{1-\mathsf{coin}})$ which $\mathcal{B}$ gives to $\mathcal{A}$ is the same as one which the challenger gives to $\mathcal{A}$. Moreover, the value of the challenge bit between $\mathcal{B}$ and its challenger is equal to the value of the challenge bit for $\mathcal{A}$. Thus, we have $\mathsf{Adv}^{\mathsf{a\text{-}hide}}_{\mathsf{OCG},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{blind}}_{\mathsf{BS},\mathcal{B}}(\lambda)$ since $\mathcal{B}$ outputs the bit coin' which is the output of $\mathcal{A}$.

Since BS satisfies the blindness, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{a\text{-}hide}}_{\mathsf{OCG},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds. Therefore, OCG satisfies attribute hiding. $\square$ (**Theorem 3.6**)

Theorem 3.7. *If* BS *satisfies blindness, then* OCG *satisfies certificate hiding.*

Proof of Theorem 3.7. Let $\mathcal{A}$ be any PPT adversary that attacks the certificate hiding of OCG. The certificate hiding game with respect to OCG is described as follows.

1. Upon receiving 1^λ from the challenger $\mathcal{CH}$, $\mathcal{A}$ outputs $(\mathsf{pk}, \mathsf{apk})$.
2. $\mathcal{CH}$ samples $\mathsf{coin} \xleftarrow{\$} \{0,1\}$, computes $(r_b, \mathsf{st}_{\mathcal{U}}^b) \leftarrow \mathsf{BS}.\mathcal{U}_1.\mathsf{Rand}(\mathsf{pk})$ for $b \in \{0,1\}$, and gives $(\mu_{\mathsf{coin}} := r_{\mathsf{coin}}, \mu_{1-\mathsf{coin}} := r_{1-\mathsf{coin}})$ to $\mathcal{A}$.
3. Upon receiving $(\tau_{\mathsf{coin}}, \tau_{1-\mathsf{coin}})$ from $\mathcal{A}$, $\mathcal{CH}$ computes $\mathsf{cert}_b \leftarrow \mathsf{BS}.\mathcal{U}_{\mathsf{der}}(\mathsf{st}_{\mathcal{U}}^b, \tau_b)$ for $b \in \{0,1\}$, sets $(\mathsf{cert}_0, \mathsf{cert}_1) := (\perp, \perp)$ if $\mathsf{cert}_0 = \perp$ or $\mathsf{cert}_1 = 0$ holds, and gives $(\mathsf{cert}_0, \mathsf{cert}_1)$ to $\mathcal{A}$.
4. $\mathcal{A}$ outputs $\mathsf{coin}' \in \{0,1\}$ and terminates.

In the following, we show that there exists a PPT adversary $\mathcal{B}$ against the blindness of BS such that $\mathsf{Adv}^{\mathsf{c\text{-}hide}}_{\mathsf{OCG},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{blind}}_{\mathsf{BS},\mathcal{B}}(\lambda)$.

1. Upon receiving 1^λ from the challenger, $\mathcal{B}$ gives 1^λ to $\mathcal{A}$.
2. Upon receiving $(\mathsf{pk}, \mathsf{apk})$ from $\mathcal{A}$, $\mathcal{B}$ returns pk to its challenger, gets $(\mu_{\mathsf{coin}}, \mu_{1-\mathsf{coin}})$, and returns $(\mu_{\mathsf{coin}}, \mu_{1-\mathsf{coin}})$ to $\mathcal{A}$.
3. Upon receiving $(\tau_{\mathsf{coin}}, \tau_{1-\mathsf{coin}})$ from $\mathcal{A}$, $\mathcal{B}$ also sends $(\tau_{\mathsf{coin}}, \tau_{1-\mathsf{coin}})$ to its challenger.
4. Upon receiving $(\mathsf{cert}_0, \mathsf{cert}_1)$ from the challenger, $\mathcal{B}$ gives $(\mathsf{cert}_0, \mathsf{cert}_1)$ to $\mathcal{A}$.
5. When $\mathcal{A}$ outputs coin' and terminates, $\mathcal{B}$ returns coin' to its challenger and terminates.

We can see that $\mathcal{B}$ perfectly simulates the attribute hiding of OCG for $\mathcal{A}$. Moreover, the value of the challenge bit between $\mathcal{B}$ and its challenger is equal to the value of the challenge bit for $\mathcal{A}$. Thus, we have $\mathsf{Adv}^{\mathsf{c\text{-}hide}}_{\mathsf{OCG},\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{blind}}_{\mathsf{BS},\mathcal{B}}(\lambda)$ since $\mathcal{B}$ outputs coin' which is the output of $\mathcal{A}$.

Since BS satisfies the blindness, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{c\text{-}hide}}_{\mathsf{OCG},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds. Therefore, OCG satisfies certificate hiding. □ (**Theorem 3.7**)

4 Split Credential Authentication Protocol

In this section, as our goal, we introduce a new cryptographic protocol called *split credential authentication* (SCA) protocol. Before providing a formal description of an SCA protocol, we give a rough explanation.

An SCA protocol is used among four entities: a user, an authority, an issuer, and a verifier and separated into two phases: One is a certificate generating phase and the other is certificate showing phase. A certificate generating phase is processed among a user, an authority, and an issuer, and the goal of this phase is the same as an OCG protocol shown in Section 3. After obtaining a certificate for an attribute attr, in the certificate showing phase, a user can show a proof that it has a certificate satisfying the policy C requested by a verifier. Here, as security requirements, in addition to properties for OCG, we need that a user cannot forge a proof not satisfying the policy C (unforgeability) and a verifier cannot obtain the information about attr beyond the fact that $C(\mathsf{attr})$ is valid (privacy).

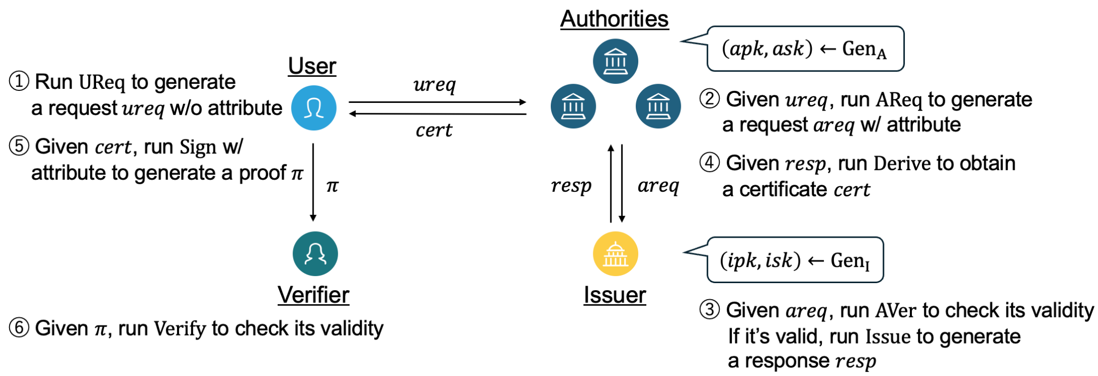

Figure 2: The flowchart of SCA

4.1 Formalization

In this section, we provide the definition of a SCA protocol. The flowchart of how to use SCA is given in Figure 2.

Definition 4.1 (Split Credential Authentication Protocol)**.** *A split credential authentication (SCA) protocol associated with a circuit family* $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ *consists of the following PPT algorithms.*

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: *The setup algorithm, given a security parameter* 1^λ*, outputs a public parameter* pp*. We assume that the following algorithms take* pp *as input implicitly.*

$\mathsf{Gen_A}(1^\lambda) \to (\mathsf{apk}, \mathsf{ask})$: *The authority key generation algorithm, given a security parameter* 1^λ*, outputs a key pair* $(\mathsf{apk}, \mathsf{ask})$*.*

$\mathsf{Gen_I}(1^\lambda) \to (\mathsf{ipk}, \mathsf{isk})$: *The issuer key generation algorithm, given a security parameter* 1^λ*, outputs a key pair* $(\mathsf{ipk}, \mathsf{isk})$*.*

$\mathsf{UReq}(\mathsf{apk}, \mathsf{ipk}) \to (\mathsf{ureq}, \mathsf{st}_\mathcal{U})$: *The user requesting algorithm, given an authority's public key* apk*, and an issuer's public key* ipk*, outputs a user's request* ureq *and secret state* $\mathsf{st}_\mathcal{U}$*.*

$\mathsf{AReq}(\mathsf{ask}, \mathsf{ipk}, \mathsf{ureq}, \mathsf{attr}) \to \mathsf{areq}$: *The (deterministic) authority requesting algorithm, given an authority's secret key* ask*, an issuer's public key* ipk*, and a user's request* ureq*, outputs an authority's request* areq*.*

$\mathsf{AVer}(\mathsf{apk}, \mathsf{areq}) \to 1/0$: *The (deterministic) authority's request verification algorithm, given an authority's public key* apk *and an authority's request* areq*, outputs either* 1 *(accept) or* 0 *(reject).*

$\mathsf{Issue}(\mathsf{isk}, \mathsf{apk}, \mathsf{areq}) \to \mathsf{resp}$: *The issuing algorithm, given an issuer's secret key* isk*, an authority's public key* apk*, and an authority's request* areq*, outputs a response* resp*.*

$\mathsf{Derive}(\mathsf{apk}, \mathsf{ipk}, \mathsf{st}_\mathcal{U}, \mathsf{resp}) \to \mathsf{cert}$: *The derivation algorithm, given an authority's public key* apk*, an issuer's public key* ipk*, a user's secret state* $\mathsf{st}_\mathcal{U}$*, and an issuer's response* areq*, outputs a certificate* cert*.*

$\mathsf{Sign}(\mathsf{apk}, \mathsf{ipk}, \mathsf{attr}, \mathsf{cert}, C) \to \pi$: *The signing algorithm, given an authority's public key* apk*, an issuer's public key* ipk*, an attribute* attr*, a certificate* cert*, and a circuit* C*, outputs a proof* π*.*

$\mathsf{Verify}(\mathsf{apk}, \mathsf{ipk}, C, \pi) \to 1/0$: *The (deterministic) verification algorithm, given an authority's public key* apk*, an issuer's public key* ipk*, a circuit* C*, and a proof* π*, outputs either* 1 *(accept) or* 0 *(reject).*

As the correctness, we require a SCA protocol SCA *to satisfy the following property.*

Correctness. *For all* $\lambda \in \mathbb{N}$*,* attr*, and* $C \in \mathcal{C}_\lambda$ *such that* $C(\mathsf{attr}) = 1$*, we have*

$$\Pr\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), \\ (\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{Gen}_\mathsf{A}(1^\lambda), \\ (\mathsf{ipk}, \mathsf{isk}) \leftarrow \mathsf{Gen}_\mathsf{I}(1^\lambda), \\ (\mathsf{ureq}, \mathsf{st}_\mathcal{U}) \leftarrow \mathsf{UReq}(\mathsf{apk}, \mathsf{ipk}), \\ \mathsf{areq} \leftarrow \mathsf{AReq}(\mathsf{ask}, \mathsf{ipk}, \mathsf{ureq}, \mathsf{attr}), \\ \mathsf{resp} \leftarrow \mathsf{Issue}(\mathsf{isk}, \mathsf{apk}, \mathsf{areq}), \\ \mathsf{cert} \leftarrow \mathsf{Derive}(\mathsf{apk}, \mathsf{ipk}, \mathsf{st}_\mathcal{U}, \mathsf{resp}), \\ \pi \leftarrow \mathsf{Sign}(\mathsf{apk}, \mathsf{ipk}, \mathsf{attr}, \mathsf{cert}, C) \end{array} : \begin{array}{c} \mathsf{Verify}(\mathsf{apk}, \mathsf{ipk}, C, \pi) = 1 \\ \wedge\ \mathsf{AVer}(\mathsf{apk}, \mathsf{areq}) = 1 \end{array}\right] = 1.$$

We require that a SCA protocol satisfies unforgeability, privacy w.r.t. areq, privacy w.r.t. π, and impersonation resilience against authority defined as follows.

Definition 4.2 (Unforgeability)**.** *Let* $\mathsf{SCA} = (\mathsf{Gen}_\mathsf{A}, \mathsf{Gen}_\mathsf{I}, \mathsf{UReq}, \mathsf{AReq}, \mathsf{Issue}, \mathsf{Sign}, \mathsf{Verify})$ *be an SCA protocol.* SCA *satisfies unforgeability if for any* $q = \mathsf{poly}(\lambda)$ *and stateful PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SCA},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Expt}^{\mathsf{unf}}_{\mathsf{SCA},\mathcal{A}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)$ *holds, where the experiment* $\mathsf{Expt}^{\mathsf{unf}}_{\mathsf{SCA},\mathcal{A}}(\lambda)$ *is defined in Figure 3.*

Definition 4.3 (Privacy w.r.t. areq)**.** *Let* $\mathsf{SCA} = (\mathsf{Gen}_\mathsf{A}, \mathsf{Gen}_\mathsf{I}, \mathsf{UReq}, \mathsf{AReq}, \mathsf{Issue}, \mathsf{Sign}, \mathsf{Verify})$ *be an SCA protocol.* SCA *satisfies privacy w.r.t.* areq *if for any stateful PPT adversary* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{priv}_1}_{\mathsf{SCA},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Expt}^{\mathsf{priv}_1,0}_{\mathsf{SCA},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{priv}_1,1}_{\mathsf{SCA},\mathcal{A}}(\lambda) = 1\right]\right| = \mathsf{negl}(\lambda)$*, where the experiment* $\mathsf{Exp}^{\mathsf{priv}_1,\mathsf{coin}}_{\mathsf{SCA},\mathcal{A}}(\lambda)$ *is defined in Figure 4.*

Definition 4.4 (Privacy w.r.t. π)**.** *Let* $\mathsf{SCA} = (\mathsf{Gen}_\mathsf{A}, \mathsf{Gen}_\mathsf{I}, \mathsf{UReq}, \mathsf{AReq}, \mathsf{Issue}, \mathsf{Sign}, \mathsf{Verify})$ *be an SCA protocol.* SCA *satisfies privacy w.r.t.* π *if for any stateful PPT adversary* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{priv}_2}_{\mathsf{SCA},\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Expt}^{\mathsf{priv}_2}_{\mathsf{SCA},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right| = \mathsf{negl}(\lambda)$*, where the experiment* $\mathsf{Expt}^{\mathsf{priv}_2}_{\mathsf{SCA},\mathcal{A}}(\lambda)$ *is defined in Figure 5.*

Definition 4.5 (Impersonation Resilience against Authority)**.** *Let* $\mathsf{SCA} = (\mathsf{Gen}_\mathsf{A}, \mathsf{Gen}_\mathsf{I}, \mathsf{UReq}, \mathsf{AReq}, \mathsf{Issue}, \mathsf{Sign}, \mathsf{Verify})$ *be an SCA protocol.* SCA *satisfies impersonation resilience against authority if for any stateful PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{imp}}_{\mathsf{SCA},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Expt}^{\mathsf{imp}}_{\mathsf{SCA},\mathcal{A}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)$ *holds, where the experiment* $\mathsf{Expt}^{\mathsf{imp}}_{\mathsf{SCA},\mathcal{A}}(\lambda)$ *is defined in Figure 6.*

Experiment $\mathsf{Expt}^{\mathsf{unf}}_{\mathsf{SCA},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{ipk}, \mathsf{isk}) \leftarrow \mathsf{Gen}_\mathsf{I}(1^\lambda)$
$i := 0, L_{\mathsf{cert}}, L_{\mathsf{SIG}}, L_{\mathsf{CorC}} := \emptyset$
$(\mathsf{apk}^*, C^*, \pi^*) \leftarrow \mathcal{A}^{\mathsf{OCert},\mathsf{CorC},\mathsf{OSign}}(\mathsf{pp}, \mathsf{ipk})$
if $\exists(\mathsf{attr}, \cdot) \in L_{\mathsf{CorC}}$ s.t. $C^*(\mathsf{attr}) = 1$ **then return** 0
if $(\mathsf{apk}^*, C^*, \pi^*) \in L_{\mathsf{SIG}}$ **then return** 0
if $\mathsf{Verify}(\mathsf{apk}^*, \mathsf{ipk}, C^*, \pi^*) = 1$ **then return** 1
return 0

Oracle $\mathsf{OCert}(\mathsf{apk}, \mathsf{ask}, \mathsf{attr})$

$i \leftarrow i + 1$
$(\mathsf{ureq}, \mathsf{st}_{\mathcal{U}}) \leftarrow \mathsf{UReq}(\mathsf{apk}, \mathsf{ipk})$
$\mathsf{areq} \leftarrow \mathsf{AReq}(\mathsf{ask}, \mathsf{ipk}, \mathsf{ureq}, \mathsf{attr})$
$\mathsf{resp} \leftarrow \mathsf{Issue}(\mathsf{isk}, \mathsf{apk}, \mathsf{areq})$
$\mathsf{cert} \leftarrow \mathsf{Derive}(\mathsf{apk}, \mathsf{ipk}, \mathsf{st}_{\mathcal{U}}, \mathsf{resp})$
$L_{\mathsf{cert}} \leftarrow L_{\mathsf{cert}} \cup \{(i, \mathsf{apk}, \mathsf{attr}, \mathsf{cert})\}$

Oracle $\mathsf{CorC}(j)$

if $(j, -, -, -) \notin L_{\mathsf{cert}}$ **then return** $\perp$
find cert s.t. $(j, \mathsf{apk}, \mathsf{attr}, \mathsf{cert}) \in L_{\mathsf{cert}}$
$L_{\mathsf{CorC}} \leftarrow L_{\mathsf{CorC}} \cup \{(\mathsf{attr}, \mathsf{cert})\}$
return cert

Oracle $\mathsf{OSign}(j, \mathsf{attr}, C)$

if $C(\mathsf{attr}) = 0$ **then return** $\perp$
if $(j, -, \mathsf{attr}, -) \notin L_{\mathsf{cert}}$ **then return** $\perp$
find apk and cert s.t. $(j, \mathsf{apk}, \mathsf{attr}, \mathsf{cert}) \in L_{\mathsf{cert}}$
$\pi \leftarrow \mathsf{Sign}(\mathsf{apk}, \mathsf{ipk}, \mathsf{attr}, \mathsf{cert}, C)$
$L_{\mathsf{SIG}} \leftarrow L_{\mathsf{SIG}} \cup \{(\mathsf{apk}, C, \pi)\}$
return π

Figure 3: The experiment for defining unforgeability.

4.2 Our Construction

In this section, we give our SCA protocol based on an OCG protocol and an NIZK proof system. [2] Let $\mathsf{OCG} = (\mathsf{OCG.Set}, \mathsf{OCG.KG}, \mathsf{OCG.AKG}, \mathsf{OCG.User}, \mathsf{OCG.Aut}, \mathsf{OCG.AVer}, \mathsf{OCG.Man}, \mathsf{OCG.Derive}, \mathsf{OCG.Ver})$ be an OCG protocol and $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Ver})$ an NIZK proof system for $\mathcal{L}$, where

$$\mathcal{L} = \{(C, \mathsf{pk}) \mid \exists(\mathsf{attr}, \mathsf{cert}_{\mathsf{OCG}}) \text{ s.t. } C(\mathsf{attr}) = 1 \wedge \mathsf{OCG.Ver}(\mathsf{pk}, \mathsf{attr}, \mathsf{cert}_{\mathsf{OCG}}) = 1\}.$$

$\mathsf{Setup}(1^\lambda)$: It computes $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ and $\mathsf{pp}_{\mathsf{OCG}} \leftarrow \mathsf{OCG.Set}(1^\lambda)$ and outputs $\mathsf{pp} := (\mathsf{crs}, \mathsf{pp}_{\mathsf{OCG}})$.

$\mathsf{Gen}_\mathsf{A}(1^\lambda)$: It computes $(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{OCG.AKG}(1^\lambda)$ and outputs $(\mathsf{apk}, \mathsf{ask})$.

[2] While we provide only the generic construction of our SCA protocol here, we leave to give an concrete instantiation and its efficiency analysis as an important open problem.

$$
\begin{array}{l}
\underline{\textbf{Experiment } \mathsf{Exp}^{\mathsf{priv}_1,\mathsf{coin}}_{\mathsf{SCA},\mathcal{A}}(\lambda)} \\
\quad \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\
\quad (\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{Gen_A}(1^\lambda) \\
\quad (\mathsf{ipk}, \mathsf{attr}_0, \mathsf{attr}_1) \leftarrow \mathcal{A}^{\mathsf{OIssue_A}}(\mathsf{pp}, \mathsf{apk}) \\
\quad \forall b \in \{0,1\} : \\
\quad\quad (\mathsf{ureq}_b, \mathsf{st}^b_{\mathcal{U}}) \leftarrow \mathsf{UReq}(\mathsf{apk}, \mathsf{ipk}) \\
\quad\quad \mathsf{areq}_b \leftarrow \mathsf{AReq}(\mathsf{ask}, \mathsf{ipk}, \mathsf{ureq}_b, \mathsf{attr}_b) \\
\quad (\mathsf{resp}_{\mathsf{coin}}, \mathsf{resp}_{1-\mathsf{coin}}) \leftarrow \mathcal{A}^{\mathsf{OIssue_A}}(\mathsf{areq}_{\mathsf{coin}}, \mathsf{areq}_{1-\mathsf{coin}}) \\
\quad \forall b \in \{0,1\} : \\
\quad\quad \mathsf{cert}_b \leftarrow \mathsf{Derive}(\mathsf{apk}, \mathsf{ipk}, \mathsf{st}^b_{\mathcal{U}}, \mathsf{resp}_b) \\
\quad \textbf{if } \mathsf{cert}_0 = \bot \vee \mathsf{cert}_1 = \bot \textbf{ then } (\mathsf{cert}_0, \mathsf{cert}_1) := (\bot, \bot) \\
\quad \mathsf{coin}' \leftarrow \mathcal{A}^{\mathsf{OIssue_A}}(\mathsf{cert}_0, \mathsf{cert}_1) \\
\quad \textbf{return } \mathsf{coin}' \\
\\
\quad \underline{\textbf{Oracle } \mathsf{OIssue_A}(\mathsf{ipk}, \mathsf{ureq}, \mathsf{attr})} \\
\quad\quad \mathsf{areq} \leftarrow \mathsf{AReq}(\mathsf{ask}, \mathsf{ipk}, \mathsf{ureq}, \mathsf{attr}) \\
\quad\quad \textbf{return } \mathsf{areq}
\end{array}
$$

Figure 4: The experiment for defining privacy w.r.t. areq.

$$
\begin{array}{l}
\underline{\textbf{Experiment } \mathsf{Exp}^{\mathsf{priv}_2}_{\mathsf{SCA},\mathcal{A}}(\lambda)} \\
\quad \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\
\quad \mathsf{coin} \xleftarrow{\$} \{0,1\} \\
\quad (\mathsf{apk}, \mathsf{ask}, \mathsf{ipk}, \mathsf{isk}, C, \mathsf{attr}_0, \mathsf{attr}_1) \leftarrow \mathcal{A}(\mathsf{pp}) \\
\quad \textbf{if } C(\mathsf{attr}_0) \neq 1 \vee C(\mathsf{attr}_1) \neq 1 \textbf{ then} \\
\quad\quad \textbf{return } \mathsf{coin}' \xleftarrow{\$} \{0,1\} \\
\quad \forall b \in \{0,1\} : \\
\quad\quad (\mathsf{ureq}_b, \mathsf{st}^b_{\mathcal{U}}) \leftarrow \mathsf{UReq}(\mathsf{apk}, \mathsf{ipk}) \\
\quad\quad \mathsf{areq}_b \leftarrow \mathsf{AReq}(\mathsf{ask}, \mathsf{ipk}, \mathsf{ureq}_b, \mathsf{attr}_b) \\
\quad\quad \mathsf{resp}_b \leftarrow \mathsf{Issue}(\mathsf{isk}, \mathsf{apk}, \mathsf{areq}_b) \\
\quad\quad \mathsf{cert}_b \leftarrow \mathsf{Derive}(\mathsf{apk}, \mathsf{ipk}, \mathsf{st}^b_{\mathcal{U}}, \mathsf{resp}_b) \\
\quad\quad \pi_{\mathsf{coin}} \leftarrow \mathsf{Sign}(\mathsf{apk}, \mathsf{ipk}, \mathsf{attr}_{\mathsf{coin}}, \mathsf{cert}_{\mathsf{coin}}, C) \\
\quad \mathsf{coin}' \leftarrow \mathcal{A}\left(\begin{array}{c} (\mathsf{ureq}_0, \mathsf{areq}_0, \mathsf{resp}_0, \mathsf{cert}_0), \\ (\mathsf{ureq}_1, \mathsf{areq}_1, \mathsf{resp}_1, \mathsf{cert}_1), \pi_{\mathsf{coin}} \end{array} \right) \\
\quad \textbf{if } \mathsf{coin} = \mathsf{coin}' \textbf{ then return } 1 \\
\quad \textbf{else return } 0
\end{array}
$$

Figure 5: The experiment for defining privacy w.r.t. π.

$\mathsf{Gen_I}(1^\lambda)$: It computes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{OCG.KG}(1^\lambda)$ and outputs $(\mathsf{ipk}, \mathsf{isk}) := (\mathsf{pk}, \mathsf{sk})$.

$\mathsf{UReq}(\mathsf{apk}, \mathsf{ipk})$: It computes $(\mu, \mathsf{st}_{\mathcal{U}}) \leftarrow \mathsf{OCG.User}(\mathsf{pk})$ and outputs $(\mathsf{ureq}, \mathsf{st}_{\mathcal{U}}) := (\mu, \mathsf{st}_{\mathcal{U}})$.

$\mathsf{AReq}(\mathsf{ask}, \mathsf{ipk}, \mathsf{ureq}, \mathsf{attr})$: It computes $\nu \leftarrow \mathsf{OCG.Aut}(\mathsf{ask}, \mu, \mathsf{attr})$ and outputs $\mathsf{areq} := \nu$.

$\mathsf{AVer}(\mathsf{apk}, \mathsf{areq})$: It computes $b \leftarrow \mathsf{OCG.AVer}(\mathsf{apk}, \mathsf{areq})$ and outputs b.

$\mathsf{Issue}(\mathsf{isk}, \mathsf{apk}, \mathsf{areq})$: It computes $\tau \leftarrow \mathsf{OCG.Man}(\mathsf{sk}, \nu)$ and outputs $\mathsf{resp} := \tau$.

$$
\begin{array}{l}
\underline{\textbf{Experiment } \mathsf{Expt}^{\mathsf{imp}}_{\mathsf{SCA},\mathcal{A}}(\lambda)} \\
\quad \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\
\quad L_{\mathsf{Gen_A}}, L_\sigma, L_{\mathsf{CorA}} := \emptyset \\
\quad (\mathsf{apk}^*, \mathsf{ureq}^*, \mathsf{attr}^*, \mathsf{areq}^*) \leftarrow \mathcal{A}^{\mathsf{OGen_A}, \mathsf{CorA}, \mathsf{OIssue_A}}(\mathsf{pp}) \\
\quad \textbf{if } (\mathsf{apk}^*, \cdot) \notin L_{\mathsf{Gen_A}} \lor \mathsf{apk}^* \in L_{\mathsf{CorA}} \textbf{ then return } 0 \\
\quad \textbf{if } (\mathsf{apk}^*, \mathsf{areq}^*) \in L_\sigma \textbf{ then return } 0 \\
\quad \textbf{if } \mathsf{AVer}(\mathsf{apk}^*, \mathsf{areq}^*) = 1 \textbf{ then return } 1 \\
\quad \textbf{return } 0 \\
\\
\quad \underline{\textbf{Oracle } \mathsf{OGen_A}()} \\
\quad\quad (\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{Gen_A}(\mathsf{pp}) \\
\quad\quad L_{\mathsf{Gen_A}} \leftarrow L_{\mathsf{Gen_A}} \cup \{(\mathsf{apk}, \mathsf{ask})\} \\
\quad\quad \textbf{return } \mathsf{apk} \\
\\
\quad \underline{\textbf{Oracle } \mathsf{CorA}(\mathsf{apk})} \\
\quad\quad \textbf{if } (\mathsf{apk}, \cdot) \notin L_{\mathsf{Gen_A}} \textbf{ then return } \bot \\
\quad\quad \textbf{find } \mathsf{ask} \text{ s.t. } (\mathsf{apk}, \mathsf{ask}) \in L_{\mathsf{Gen_A}} \\
\quad\quad L_{\mathsf{CorA}} \leftarrow L_{\mathsf{CorA}} \cup \{\mathsf{apk}\} \\
\quad\quad \textbf{return } \mathsf{ask} \\
\\
\quad \underline{\textbf{Oracle } \mathsf{OIssue_A}(\mathsf{apk}, \mathsf{ipk}, \mathsf{ureq}, \mathsf{attr})} \\
\quad\quad \textbf{if } (\mathsf{apk}, \cdot) \notin L_{\mathsf{Gen_A}} \textbf{ then return } \bot \\
\quad\quad \textbf{find } \mathsf{ask} \text{ s.t. } (\mathsf{apk}, \mathsf{ask}) \in L_{\mathsf{Gen_A}} \\
\quad\quad \mathsf{areq} \leftarrow \mathsf{AReq}(\mathsf{ask}, \mathsf{ipk}, \mathsf{ureq}, \mathsf{attr}) \\
\quad\quad L_\sigma \leftarrow L_\sigma \cup \{(\mathsf{apk}, \mathsf{areq})\} \\
\quad\quad \textbf{return } \mathsf{areq}
\end{array}
$$

Figure 6: The experiment for defining impersonation resilience against authority.

$\mathsf{Derive}(\mathsf{apk}, \mathsf{ipk}, \mathsf{st}_{\mathcal{U}}, \mathsf{resp})$: It computes $\mathsf{cert}_{\mathsf{OCG}} \leftarrow \mathsf{OCG.Derive}(\mathsf{st}_{\mathcal{U}}, \tau)$ and outputs $\mathsf{cert} := \mathsf{cert}_{\mathsf{OCG}}$.

$\mathsf{Sign}(\mathsf{apk}, \mathsf{ipk}, \mathsf{attr}, \mathsf{cert}, C)$: It computes $\pi_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (C, \mathsf{pk}), (\mathsf{attr}, \mathsf{cert}_{\mathsf{OCG}}))$ and outputs $\pi := \pi_{\mathsf{NIZK}}$.

$\mathsf{Verify}(\mathsf{apk}, \mathsf{ipk}, C, \pi)$: It computes $b \leftarrow \mathsf{NIZK.Ver}(\mathsf{crs}, (C, \mathsf{pk}), \pi_{\mathsf{NIZK}})$ and outputs b.

Due to the correctness of OCG and NIZK, it is easy to see that the correctness of SCA holds. Moreover, impersonation resilience against authority and privacy w.r.t. areq are immediately followed from the impersonation resilience of OCG and the attribute hiding of OCG, respectively. Then, in the following, we show that our SCA protocol SCA satisfies unforgeability and privacy w.r.t. π.

Theorem 4.6. *If* OCG *satisfies unforgeability and* NIZK *satisfies extractability, then* SCA *satisfies unforgeability.*

Proof of Theorem 4.6. Let $\mathcal{A}$ be any PPT adversary that attacks the unforgeability of SCA. We proceed the proof via a sequence of games. We introduce the following two games: Game_i for $i \in \{0, 1\}$.

Game_0 : This is the original unforgeability game with respect to SCA. The detailed description is as follows.

1. The challenger $\mathcal{CH}$ firstly generates $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$, $\mathsf{pp_{OCG}} \leftarrow \mathsf{OCG.Set}(1^\lambda)$, and $(\mathsf{ipk}, \mathsf{isk}) \leftarrow \mathsf{OCG.KG}(1^\lambda)$, sets $\mathsf{pp} := (\mathsf{crs}, \mathsf{pp_{OCG}})$ and $i := 0, L_{\mathsf{cert}}, L_{\mathsf{SIG}}, L_{\mathsf{CorC}} := \emptyset$, and gives $(\mathsf{pp}, \mathsf{ipk})$ to $\mathcal{A}$.
2. When $\mathcal{A}$ makes queries to $\mathsf{OCert}, \mathsf{CorC}, \mathsf{OSign}$, $\mathcal{CH}$ proceeds as follows:

   $\mathsf{OCert}(\mathsf{apk}, \mathsf{ask}, \mathsf{attr})$: When $\mathcal{A}$ makes a query $(\mathsf{apk}, \mathsf{ask}, \mathsf{attr})$, $\mathcal{CH}$ updates $i := i + 1$, computes $(\mu, \mathsf{st}_{\mathcal{U}}) \leftarrow \mathsf{OCG.User}(\mathsf{ipk})$, $\nu \leftarrow \mathsf{OCG.Aut}(\mathsf{ask}, \mu, \mathsf{attr})$, $\tau \leftarrow \mathsf{OCG.Man}(\mathsf{isk}, \nu)$, and $\mathsf{cert_{OCG}} \leftarrow \mathsf{OCG.Derive}(\mathsf{st}_{\mathcal{U}}, \tau)$, and sets $L_{\mathsf{cert}} \leftarrow L_{\mathsf{cert}} \cup \{(i, \mathsf{apk}, \mathsf{attr}, \mathsf{cert})\}$.

   $\mathsf{CorC}(j)$: When $\mathcal{A}$ makes a query j, $\mathcal{CH}$ checks whether $(j, -, -, -) \notin L_{\mathsf{cert}}$ holds. If this is the case, then $\mathcal{CH}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{CH}$ searches cert such that $(j, \mathsf{apk}, \mathsf{attr}, \mathsf{cert}) \in L_{\mathsf{cert}}$, sets $L_{\mathsf{CorC}} \leftarrow L_{\mathsf{CorC}} \cup \{(\mathsf{attr}, \mathsf{cert})\}$, and returns cert to $\mathcal{A}$.

   $\mathsf{OSign}(j, \mathsf{attr}, C)$: When $\mathcal{A}$ makes a query (j, attr, C), $\mathcal{CH}$ checks whether $C(\mathsf{attr}) = 0$ or $(j, -, \mathsf{attr}, -) \notin L_{\mathsf{cert}}$ holds. If this is the case, then $\mathcal{CH}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{CH}$ searches apk and cert such that $(j, \mathsf{apk}, \mathsf{attr}, \mathsf{cert}) \in L_{\mathsf{cert}}$, computes $\pi_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (C, \mathsf{ipk}), (\mathsf{attr}, \mathsf{cert_{OCG}}))$, sets $L_{\mathsf{SIG}} \leftarrow L_{\mathsf{SIG}} \cup \{(\mathsf{apk}, C, \pi)\}$, and returns π_{NIZK} to $\mathcal{A}$.
3. When $\mathcal{A}$ outputs $(\mathsf{apk}^*, C^*, \pi^*)$ and terminates, $\mathcal{CH}$ checks whether $\exists(\mathsf{attr}, -) \in L_{\mathsf{CorC}}$ such that $C^*(\mathsf{attr}) = 1$ or $\exists(\mathsf{apk}^*, C^*, \pi^*) \in L_{\mathsf{SIG}}$ holds. If this is the case, then $\mathcal{CH}$ returns 0 and terminates. Otherwise, $\mathcal{CH}$ checks whether $\mathsf{Verify}(\mathsf{apk}^*, \mathsf{ipk}, C^*, \pi^*) = 1$ holds. If this is the case, then $\mathcal{CH}$ returns 1 and terminates. Otherwise, $\mathcal{CH}$ returns 0 and terminates.

Game_1 : This game is identical to Game_0 except that we compute $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Ext}_0(1^\lambda)$ instead of $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$.

Let Succ_i be an event that $\mathcal{CH}$ returns 1 in Game_i for $i \in \{0, 1\}$. Let Bad be an event that $(\mathsf{X}^*, \mathsf{W}^*) \notin \mathcal{R}$ in Game_1, where $\mathsf{X}^* = (C^*, \mathsf{ipk})$ and $\mathsf{W}^* \leftarrow \mathsf{Ext}_1(\mathsf{crs}, \mathsf{td}, \mathsf{X}^*)$. By using the triangle inequality, we have

$$\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SCA},\mathcal{A}}(\lambda) = \Pr[\mathsf{Succ}_0] \leq |\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1]| + \Pr[\mathsf{Succ}_1].$$

Moreover, we have

$$\begin{aligned} \Pr[\mathsf{Succ}_1] &= \Pr[\mathsf{Bad}]\Pr[\mathsf{Succ}_1 \wedge \mathsf{Bad}] + \Pr[\neg\mathsf{Bad}]\Pr[\mathsf{Succ}_1 \wedge \neg\mathsf{Bad}] \\ &\leq \Pr[\mathsf{Bad}] + \Pr[\mathsf{Succ}_1 \wedge \neg\mathsf{Bad}]. \end{aligned}$$

It remains to show how each $|\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1]|$, $\Pr[\mathsf{Bad}]$, and $\Pr[\mathsf{Succ}_1 \wedge \neg\mathsf{Bad}]$ are upper-bounded. In the following, we show that there exist an adversary $\mathcal{B}_1$ against the CRS indistinguishability of NIZK such that $|\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1]| = \mathsf{Adv}^{\mathsf{crs}}_{\mathsf{NIZK},\mathcal{B}_1}(\lambda)$, an adversary $\mathcal{B}_2$ against the extractability of NIZK such that $\Pr[\mathsf{Bad}] = \mathsf{Adv}^{\mathsf{ext}}_{\mathsf{NIZK},\mathcal{B}_2}(\lambda)$, and an adversary $\mathcal{B}_3$ against the unforgeability of OCG such that $\Pr[\neg\mathsf{Bad} \wedge \mathsf{Succ}_1] = \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{OCG},\mathcal{B}_3}(\lambda)$.

Lemma 4.7. *There exists an adversary $\mathcal{B}_1$ against the CRS indistinguishability of* NIZK *such that* $|\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1]| = \mathsf{Adv}^{\mathsf{crs}}_{\mathsf{NIZK},\mathcal{B}_1}(\lambda)$.

Proof of Lemma 4.7. We construct an adversary $\mathcal{B}_1$ that attacks the CRS indistinguishability of NIZK so that $|\Pr[\mathsf{Succ}_0]-\Pr[\mathsf{Succ}_1]| = \mathsf{Adv}^{\mathsf{crs}}_{\mathsf{NIZK},\mathcal{B}_1}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving crs from the challenger, $\mathcal{B}_1$ generates $\mathsf{pp}_{\mathsf{OCG}} \leftarrow \mathsf{OCG.Set}(1^\lambda)$ and $(\mathsf{ipk},\mathsf{isk}) \leftarrow \mathsf{OCG.KG}(1^\lambda)$, sets $i := 0$, gives $\mathsf{pp} := (\mathsf{pp}_{\mathsf{OCG}},\mathsf{ipk})$ to $\mathcal{A}$.
2. When $\mathcal{A}$ makes queries to $\mathsf{OCert},\mathsf{CorC},\mathsf{OSign}$, $\mathcal{B}_1$ answers in the same way as $\mathcal{CH}$ does in Game_0.
3. When $\mathcal{A}$ outputs $(\mathsf{apk}^*, C^*, \pi^*)$ and terminates, $\mathcal{B}_1$ checks whether Succ_0 (or equally Succ_1) occurs. If this is the case, then $\mathcal{B}_1$ returns 1 to its challenger and terminates.

We can see that $\mathcal{B}_1$ perfectly simulates Game_0 for $\mathcal{A}$ if it receives a real CRS from its challenger. This ensures that the probability that $\mathcal{B}_1$ outputs 1 when it receives a real CRS is exactly the same as the probability that $\mathcal{A}$ outputs 1 in Game_0. On the other hand, $\mathcal{B}_1$ perfectly simulates Game_1 for $\mathcal{A}$ if it receives a simulated CRS from its challenger. This ensures that the probability that $\mathcal{B}_1$ outputs 1 when it receives a simulated CRS from its challenger is exactly the same as the probability that $\mathcal{A}$ outputs 1 in Game_1. Hence, we have $|\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1]| = \mathsf{Adv}^{\mathsf{crs}}_{\mathsf{NIZK},\mathcal{B}_1}(\lambda)$. $\square$ (**Lemma 4.7**)

Lemma 4.8. *There exists an adversary $\mathcal{B}_2$ against the extractability of* NIZK *such that* $\Pr[\mathsf{Bad}] = \mathsf{Adv}^{\mathsf{ext}}_{\mathsf{NIZK},\mathcal{B}_2}(\lambda)$.

Proof of Lemma 4.8. We construct an adversary $\mathcal{B}_2$ that attacks the extractability of NIZK so that $\Pr[\mathsf{Bad}] = \mathsf{Adv}^{\mathsf{ext}}_{\mathsf{NIZK},\mathcal{B}_2}(\lambda)$, using the adversary $\mathcal{A}$ as follows.

1. Upon receiving crs from the challenger, $\mathcal{B}_2$ generates $\mathsf{pp}_{\mathsf{OCG}} \leftarrow \mathsf{OCG.Set}(1^\lambda)$ and $(\mathsf{ipk},\mathsf{isk}) \leftarrow \mathsf{OCG.KG}(1^\lambda)$, sets $i := 0$, gives $\mathsf{pp} := (\mathsf{pp}_{\mathsf{OCG}},\mathsf{ipk})$ to $\mathcal{A}$.
2. When $\mathcal{A}$ makes queries to $\mathsf{OCert},\mathsf{CorC},\mathsf{OSign}$, $\mathcal{B}_2$ answers in the same way as $\mathcal{CH}$ does in Game_0.
3. When $\mathcal{A}$ outputs $(\mathsf{apk}^*, C^*, \pi^*)$ and terminates, $\mathcal{B}_2$ sets $\mathsf{X}^* := (C^*, \mathsf{ipk})$ and returns (X^*, π^*) to its challenger and terminates.

It is easy to see that $\mathcal{B}_2$ perfectly simulates Game_1 for $\mathcal{A}$. Recall that the success condition of $\mathcal{B}_2$ is to output a tuple (X^*, π^*) satisfying $(\mathsf{X}^*, \mathsf{W}^*) \notin \mathcal{R} \wedge 1 = \mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{X}^*, \pi^*)$, where $\mathsf{W}^* \leftarrow \mathsf{Ext}_1(\mathsf{crs},\mathsf{td},\mathsf{X}^*)$. If the event Bad occurs, then $(\mathsf{X}^*, \mathsf{W}^*) \notin \mathcal{R} \wedge 1 = \mathsf{Verify}(\mathsf{apk}^*, \mathsf{ipk}, C^*, \pi^*)$ holds. Due to the construction of SCA, the condition $1 = \mathsf{Verify}(\mathsf{apk}^*, \mathsf{ipk}, C^*, \pi^*)$ implies the condition $1 = \mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{X}^*, \pi^*)$. Thus, when Bad occurs, $\mathcal{B}_2$ achieves its success condition by returning (X^*, π^*) to its challenger. Hence, we have $\Pr[\mathsf{Bad}] = \mathsf{Adv}^{\mathsf{ext}}_{\mathsf{NIZK},\mathcal{B}_2}(\lambda)$.
$\square$ (**Lemma 4.8**)

Lemma 4.9. *There exists an adversary $\mathcal{B}_3$ against the unforgeability of* OCG *such that* $\Pr[\neg\mathsf{Bad} \wedge \mathsf{Succ}_1] = \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{OCG},\mathcal{B}_3}(\lambda)$.

Proof of Lemma 4.9. We construct an adversary $\mathcal{B}_3$ that attacks the unforgeability of OCG so that $\Pr[\mathsf{Succ}_1] = \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{OCG},\mathcal{B}_3}(\lambda)$ as follows.

1. Upon receiving $(\mathsf{pp}_{\mathsf{OCG}}, \mathsf{pk})$ from the challenger, $\mathcal{B}_3$ generates $(\mathsf{crs},\mathsf{td}) \leftarrow \mathsf{Sim}_0(1^\lambda)$, sets $\mathsf{pp} := (\mathsf{pp}_{\mathsf{OCG}}, \mathsf{crs})$ and $i := 0$, and gives pp to $\mathcal{A}$.

2. When $\mathcal{A}$ makes queries to $\mathsf{OCert}, \mathsf{CorC}, \mathsf{OSign}$, $\mathcal{B}_3$ answers as follows.

   $\mathsf{OCert}(\mathsf{apk}, \mathsf{ask}, \mathsf{attr})$: When $\mathcal{A}$ makes a query $(\mathsf{apk}, \mathsf{ask}, \mathsf{attr})$, $\mathcal{B}_3$ updates $i := i+1$, computes $(\mu_i, \mathsf{st}_{\mathcal{U},i}) \leftarrow \mathsf{OCG.User}(\mathsf{pk})$ and $\nu_i \leftarrow \mathsf{OCG.Aut}(\mathsf{ask}, \mu_i, \mathsf{attr}_i)$, and makes a query ν_i to its oracle $\mathcal{O}_{\mathsf{Man}}$. Upon receiving τ_i, $\mathcal{B}_3$ computes $\mathsf{cert}_{\mathsf{OCG},i} \leftarrow \mathsf{OCG.Derive}(\mathsf{st}_{\mathcal{U}}, \tau)$ and sets $L_{\mathsf{cert}} := L_{\mathsf{cert}} \cup \{(i, \mathsf{apk}, \mathsf{attr}_i := \mathsf{attr}, \mathsf{cert}_{\mathsf{OCG},i})\}$.

   $\mathsf{CorC}(j)$: When $\mathcal{A}$ makes a query j, $\mathcal{B}_3$ proceeds in the same way as $\mathcal{CH}$ does in Game_1.

   $\mathsf{OSign}(j, \mathsf{attr}, C)$: When $\mathcal{A}$ makes a query (j, attr, C), $\mathcal{B}_3$ proceeds in the same way as $\mathcal{CH}$ does in Game_1.

3. When $\mathcal{A}$ outputs $(\mathsf{apk}^*, C^*, \pi^*)$ and terminates, $\mathcal{B}_3$ computes $(\mathsf{attr}^*, \mathsf{cert}^*_{\mathsf{OCG}}) \leftarrow \mathsf{Sim}_1(\mathsf{crs}, \mathsf{td}, (C^*, \mathsf{pk}))$, sets $\mathsf{attr}_{i+1} := \mathsf{attr}^*$ and $\mathsf{cert}_{\mathsf{OCG},i+1} := \mathsf{cert}^*_{\mathsf{OCG}}$, and returns $\{(\mathsf{attr}_k, \mathsf{cert}_{\mathsf{OCG},k})\}_{k\in[i+1]}$ to its challenger.

It is easy to see that $\mathcal{B}_3$ perfectly simulates Game_1 for $\mathcal{A}$. Recall that the success condition of $\mathcal{B}_3$ is to output a tuple $\{(\mathsf{attr}_k, \mathsf{cert}_{\mathsf{OCG},k})\}_{k\in[i+1]}$ satisfying $\mathsf{OCG.Ver}(\mathsf{pk}, \mathsf{attr}_k, \mathsf{cert}_k) = 1$ for all $k \in [i+1]$ and $\{\mathsf{attr}_k\}_{k\in[i+1]}$ is pairwise distinct.

If Succ_1 occurs, we have $(\mathsf{apk}^*, C^*, \pi^*) \notin L_{\mathsf{SIG}}$, $\forall(\mathsf{attr}, \cdot) \in L_{\mathsf{CorC}} : C^*(\mathsf{attr}) = 0$, and $\mathsf{Verify}(\mathsf{apk}^*, \mathsf{ipk}, C^*, \pi^*) = 1$. Due to the construction of SCA, the condition $1 = \mathsf{Verify}(\mathsf{apk}^*, \mathsf{ipk}, C^*, \pi^*)$ implies the condition $1 = \mathsf{NIZK.Ver}(\mathsf{crs}, \mathsf{X}^*, \pi^*)$.

Moreover, if $\neg\mathsf{Bad}$ occurs, $((C^*, \mathsf{pk}), (\mathsf{attr}^*, \mathsf{cert}^*_{\mathsf{OCG}})) \in \mathcal{R}$ holds for $(\mathsf{attr}^*, \mathsf{cert}^*_{\mathsf{OCG}}) \leftarrow \mathsf{Sim}_1(\mathsf{crs}, \mathsf{td}, (C^*, \mathsf{pk}))$. That is, we have $C^*(\mathsf{attr}^*) = 1$ and $\mathsf{OCG.Ver}(\mathsf{pk}, \mathsf{attr}^*, \mathsf{cert}^*_{\mathsf{OCG}}) = 1$. From the conditions $(\mathsf{apk}^*, C^*, \pi^*) \notin L_{\mathsf{SIG}}$, $C^*(\mathsf{attr}^*) = 1$, and $\forall(\mathsf{attr}, \cdot) \in L_{\mathsf{CorC}} : C^*(\mathsf{attr}) = 0$, $\mathsf{attr}^*(= \mathsf{attr}_{i+1})$ is distinct from attr_k for any $k \in [i]$.

Thus, when Succ_1 occurs, $\mathcal{B}_3$ achieves its success condition by returning $\{(\mathsf{attr}_k, \mathsf{cert}_{\mathsf{OCG},k})\}_{k\in[i+1]}$ to its challenger. Hence, we have $\Pr[\neg\mathsf{Bad} \wedge \mathsf{Succ}_1] = \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{OCG},\mathcal{B}_3}(\lambda)$. □ (**Lemma 4.9**)

Putting everything together, we obtain

$$\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SCA},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{crs}}_{\mathsf{NIZK},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{ext}}_{\mathsf{NIZK},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{unf}}_{\mathsf{OCG},\mathcal{B}_3}(\lambda)$$

Since OCG satisfies unforgeability and NIZK satisfies CRS indistinguishability and extractability, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{unf}}_{\mathsf{SCA},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds. Therefore, SCA satisfies unforgeability. □ (**Theorem 4.6**)

Theorem 4.10. *If* NIZK *satisfies zero-knowledge, then* SCA *satisfies privacy w.r.t.* π.

Proof of Theorem 4.10. Let $\mathcal{A}$ be any PPT adversary that attacks the privacy w.r.t. π of SCA. We introduce the following two games: Game_i for $i \in \{0,1\}$.

Game_0 : This is the original privacy game w.r.t. π. The detailed description is as follows.

1. The challenger $\mathcal{CH}$ samples $\mathsf{coin} \xleftarrow{\$} \{0,1\}$, generates $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ and $\mathsf{pp}_{\mathsf{OCG}} \leftarrow \mathsf{OCG.Set}(1^\lambda)$, sets $\mathsf{pp} := (\mathsf{crs}, \mathsf{pp}_{\mathsf{OCG}})$, and sends pp to $\mathcal{A}$.
2. Upon receiving $(\mathsf{apk}, \mathsf{ask}, \mathsf{ipk}, \mathsf{isk}, C, \mathsf{attr}_0, \mathsf{attr}_1)$ from $\mathcal{A}$, $\mathcal{CH}$ checks whether $C(\mathsf{attr}_0) \neq 1 \vee C(\mathsf{attr}_1) \neq 1$ holds. If this is the case, then $\mathcal{CH}$ outputs $\mathsf{coin}' \xleftarrow{\$} \{0,1\}$ and terminates. Otherwise, it computes $(\mu_b, \mathsf{st}^b_{\mathcal{U}}) \leftarrow \mathsf{OCG.User}(\mathsf{ipk})$, $\nu_b \leftarrow \mathsf{OCG.Aut}(\mathsf{ask}, \mu_b, \mathsf{attr}_b)$, $\tau_b \leftarrow \mathsf{OCG.Man}(\mathsf{isk}, \nu_b)$, and $\mathsf{cert}_{\mathsf{OCG},b} \leftarrow \mathsf{OCG.Derive}(\mathsf{st}^b_{\mathcal{U}}, \tau_b)$ for $b \in \{0,1\}$, generates $\pi_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (C, \mathsf{ipk}), (\mathsf{attr}_b, \mathsf{cert}_{\mathsf{OCG},b}))$, and sends $((\mu_0, \nu_0, \tau_0, \mathsf{cert}_{\mathsf{OCG},0}), (\mu_1, \nu_1, \tau_1, \mathsf{cert}_{\mathsf{OCG},1}), \pi_{\mathsf{NIZK}})$ to $\mathcal{A}$.

3. Upon receiving coin' from $\mathcal{A}$, $\mathcal{CH}$ checks whether $\mathsf{coin} = \mathsf{coin}'$ holds. If this is the case, then $\mathcal{CH}$ returns 1 and terminates. Otherwise, $\mathcal{CH}$ returns 0 and terminates

Game_1 : This game is identical to Game_0 except that, when receiving $(\mathsf{apk}, \mathsf{ask}, \mathsf{ipk}, \mathsf{isk}, C, \mathsf{attr}_0, \mathsf{attr}_1)$ from $\mathcal{A}$, we compute $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}_0(1^\lambda)$ and $\pi_{\mathsf{NIZK}} \leftarrow \mathsf{Sim}_1(\mathsf{crs}, \mathsf{td}, (C, \mathsf{ipk}))$ instead of $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ and $\pi_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, (C, \mathsf{ipk}), (\mathsf{attr}_b, \mathsf{cert}_{\mathsf{OCG},b}))$, respectively.

Let Succ_i be an event that $\mathcal{CH}$ returns 1 in Game_i for $i \in \{0, 1\}$. By using the triangle inequality, we have

$$\mathsf{Adv}^{\mathsf{priv}_2}_{\mathsf{SCA},\mathcal{A}}(\lambda) = |\Pr[\mathsf{Succ}_0] - \frac{1}{2}| \le |\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1]| + |\Pr[\mathsf{Succ}_1] - \frac{1}{2}|.$$

It remains to show how each $|\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1]|$ and $|\Pr[\mathsf{Succ}_1] - \frac{1}{2}|$ are upper-bounded. In the following, we show that there exists an adversary $\mathcal{B}$ against the zero-knowledge of NIZK such that $|\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1]| = \mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathcal{B}}(\lambda)$ and $|\Pr[\mathsf{Succ}_1] - \frac{1}{2}| = 0$ holds.

Lemma 4.11. *There exists an adversary $\mathcal{B}$ against the zero-knowledge of* NIZK *such that* $|\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1]| = \mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathcal{B}}(\lambda)$.

Proof of Lemma 4.11. We show that there exists a PPT adversary $\mathcal{B}$ against the zero-knowledge of NIZK such that $|\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1]| = \mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathcal{B}}(\lambda)$.

1. Upon receiving crs from the challenger, $\mathcal{B}$ samples $\mathsf{coin} \xleftarrow{\$} \{0, 1\}$, generates $\mathsf{pp}_{\mathsf{OCG}} \leftarrow \mathsf{OCG.Set}(1^\lambda)$, sets $\mathsf{pp} := (\mathsf{crs}, \mathsf{pp}_{\mathsf{OCG}})$, sends pp to $\mathcal{A}$.
2. Upon receiving $(\mathsf{apk}, \mathsf{ask}, \mathsf{ipk}, \mathsf{isk}, C, \mathsf{attr}_0, \mathsf{attr}_1)$ from $\mathcal{A}$, $\mathcal{B}$ proceeds in the same way as $\mathcal{CH}$ does in Game_0 except that it makes a query $((C, \mathsf{ipk}), (\mathsf{attr}_{\mathsf{coin}}, \mathsf{cert}_{\mathsf{OCG},\mathsf{coin}}))$ to its oracle instead that it computes π_{NIZK} by itself.
3. Upon receiving π_{NIZK} from the challenger, $\mathcal{B}$ sends $((r_0, \nu_0, \tau_0, \mathsf{cert}_{\mathsf{OCG},0}), (r_1, \nu_1, \tau_1, \mathsf{cert}_{\mathsf{OCG},1}), \pi_{\mathsf{NIZK}})$ to $\mathcal{A}$.
4. When $\mathcal{A}$ outputs coin' and terminates, $\mathcal{B}$ returns 1 to its challenger if $\mathsf{coin} = \mathsf{coin}'$ holds. Otherwise, $\mathcal{B}$ returns 0 to its challenger.

We can see that $\mathcal{B}$ perfectly simulates Game_0 for $\mathcal{A}$ if it receives a real CRS and a real proof from its challenger. This ensures that the probability that $\mathcal{B}$ outputs 1 when it receives a real CRS and a real proof is exactly the same as the probability that $\mathcal{A}$ outputs 1 in Game_0. On the other hand, $\mathcal{B}$ perfectly simulates Game_1 for $\mathcal{A}$ if it receives a simulated CRS and a simulated proof from its challenger. This ensures that the probability that $\mathcal{B}$ outputs 1 when it receives a simulated CRS and a simulated proof from its challenger is exactly the same as the probability that $\mathcal{A}$ outputs 1 in Game_1. Hence, we have $|\Pr[\mathsf{Succ}_0] - \Pr[\mathsf{Succ}_1]| = \mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathcal{B}}(\lambda)$.
□ (**Lemma 4.12**)

Lemma 4.12. $|\Pr[\mathsf{Succ}_1] - \frac{1}{2}| = 0$ *holds.*

Proof of Lemma 4.12 Due to the change in Game_1, the information of coin is information-theoretically hidden for $\mathcal{A}$ in Game_1. Thus, $|\Pr[\mathsf{Succ}_1] - \frac{1}{2}| = 0$ holds. □ (**Lemma 4.12**)

Since NIZK satisfies zero-knowledge, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{priv}_2}_{\mathsf{SCA},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ holds. Therefore, SCA satisfies privacy w.r.t. π. □ (**Theorem 4.10**)

Table 1: Use Case Matrix for the Anonymous Credential Model

Domain	**Use Case**	**Disclosed Attribute Type**	**Selective Disclosure**	**Notes**
Public Services	Access to public transportation discount	Boolean	Proves eligibility without revealing reason	Eligibility (e.g., disability) confirmed without disclosing type or medical condition
Education	Request for reasonable accommodation	Boolean	Proves need without showing specific circumstance	Diagnostic or personal reasons remain confidential
Healthcare	priority service at pharmacy	Category	Indicates condition category	User can prove eligibility (e.g., chronic or episodic condition) without disclosing specifics
Private Sector	Use of disability-based membership benefits	Boolean	Proof of a valid certificate only	Presence of valid certificate is shown; contents or reason remain hidden
Administrative Procedures	Proof of eligibility for disability hiring quota	Boolean	Eligibility for quota only	Used to show eligibility for disability employment quota

5 Application

In this section, we discuss the importance and applications of our proposed SCA.

5.1 Social Importance of Privacy in Disability Contexts

The proposed anonymous credential model, which separates the role of the attribute management and the credential issuance, offers significant practical value in enabling the issuance and use of anonymous credentials that contain sensitive attributes. In particular, attribute information concerning persons with disabilities is closely tied to medical care, welfare, mental health, and other domains, and is deeply connected to individual dignity and social standing. Such information is legally recognized worldwide as requiring especially careful handling. For instance, Article 9 of the EU General Data Protection Regulation (GDPR) [1] explicitly designates information related to disability, alongside health status, biometric data, and sexual orientation, as "special category personal data" that may not be collected or processed without the data subject's explicit consent. Similarly, the United States' HIPAA [28] and Australia's Privacy Act [3] identify disability-related information as sensitive personal data requiring special safeguards.

According to the World Health Organization, over 1.3 billion people, or approximately 16% of the global population, live with some form of disability [32]. While many support services are designed to assist persons with disabilities across healthcare, education, welfare, and employment domains, international evidence reveals that a substantial portion of this population deliberately avoids using these services due to concerns about stigma, discrimination, and violations of privacy. Global comparative studies have shown that persons with disabilities are, for example, up to four times more likely to report being treated badly by healthcare providers, and nearly three times more likely to be denied care [30]. Given the global disabled population, this suggests that approximately 500 to 700 million people may be at elevated risk of such treatment. These experiences lead many to avoid or delay accessing health services entirely [32]. In the context of mental health, more than 50% of individuals with serious conditions worldwide—amounting to over 650 million people—do not receive treatment, particularly in low- and middle-income countries, where stigma and fear of disclosure are cited as key deterrents [31]. This avoidance behavior extends beyond healthcare. According to UNICEF, an estimated 60 million children with disabilities lack access to early childhood services, and over 110 million may never attend school at all [27]. These gaps are attributed not only to structural inaccessibility, but also to cultural stigma and parental fears of discrimination. In the workplace, a cross-national survey found that 43% of persons with disabilities did not feel safe disclosing their status, and more than half avoided requesting job accommodations out of concern for prejudice or career disadvantage [24]. This means that more than 400 million people may face disclosure-related barriers at work. These findings underscore the urgent need for authentication mechanisms that support selective disclosure—allowing individuals to prove eligibility for services without revealing detailed or stigmatized information. In many institutional contexts, eligibility is determined not by the precise nature of one's condition but by whether it falls within a defined set of acceptable categories. This makes it possible for users to present proof of eligibility without disclosing the specific underlying attribute.

5.2 Application of Proposed Model in Real-World Settings

In the proposed model, the authority is responsible for verifying the correctness of the user's attributes, while the issuer performs credential issuance solely based on the verified result. This separation enables a clearer division of institutional responsibilities. Specifically, in real-world institutional contexts, the entities responsible for evaluating attributes (e.g., municipalities, medical institutions, welfare agencies) and those responsible for cryptographic credential issuance (e.g., certification authorities) are often distinct. The proposed model supports this distinction by allowing each party to focus on their domain-specific expertise without requiring unified trust anchors for both functions. A crucial benefit of this separation is that it structurally prevents users from unilaterally asserting arbitrary attributes. Since the issuer only issues credentials based on verification received from an authorized authority, the model enforces a binding relationship between verification and issuance. This design mitigates a key limitation of prior anonymous credential schemes, which users could potentially fabricate unsupported claims.

An additional benefit of this separation is that it structurally facilitates the incorporation of attribute anonymity into the credential issuance process. By allowing the authority to delegate issuance to the issuer without disclosing the attribute content through our proposed protocol, the system ensures that even a possibly malicious issuer cannot learn the user's sensitive information. This makes it possible to design systems where attribute confidentiality is preserved not only through a policy, but also by cryptographic design principles. This structural safeguard becomes especially important in cases where the issuer may be legally or operationally distinct from the authority.

Anonymous credentials issued under this model are applicable in various services targeting persons with disabilities. For instance, in public transportation systems offering disability discounts, users can prove that they meet eligibility criteria without disclosing the specific type of their disabilities. In such systems, a transport operator can validate a user's credential—e.g., "eligible for reduced fare"—without learning whether the underlying reason is visual impairment, chronic illness, or mental health status. Similarly, in healthcare settings, a person with a panic disorder may wish to access priority services at a pharmacy. Rather than disclosing their exact diagnosis, they could present a credential attesting that they qualify for such priority under broadly defined criteria (e.g., certain chronic or episodic conditions). This prevents unintended exposure of psychiatric or neurological conditions while still fulfilling the eligibility check. Also, in educational settings, students requesting accommodations—such as flexible attendance or extended deadlines—are often not required to disclose whether their situation involves illness, bereavement, or disability, as long as it falls within the institution's accommodation policy. Our model supports issuing anonymized proofs of eligibility through pre-approved institutional workflows, thereby streamlining administrative burden. Other applications include employment contexts involving reasonable accommodations, or private services that offer preferential rates or benefits to individuals with certified disabilities. See TABLE 1 for a summary.

5.3 Estimated Reach and Broader Social Impact

As discussed in Section 5.1, global studies estimate that 15% to 20% of persons with disabilities avoid accessing health, welfare, or education services due to concerns about privacy, stigma, or past discrimination. Given the estimated 1.3 billion persons with disabilities worldwide, this corresponds to approximately 195 to 260 million individuals currently excluded from essential services.

While not all cases can be resolved by authentication design alone, a system that enables individuals to prove eligibility without revealing sensitive attributes could directly remove one of the most commonly cited barriers to service utilization.

This estimate reflects only the primary effect among those already eligible but disengaged due to disclosure concerns. It does not account for secondary impacts, such as reduced institutional processing burden, greater legal compliance, or improved long-term inclusion for newly applying individuals. Even under conservative assumptions, the deployment of such a system presents a clear and measurable pathway to restore autonomy and access for tens of millions globally.

6 Conclusion

In this paper, we propose a new cryptographic protocol called a split credential authentication (SCA) protocol, in which the roles of managing users' attributes and issuing certificates are split into distinct authorities. To this end, as an intermediate tool, we also introduce a new cryptographic primitive called oblivious certificate generation (OCG) and give its construction based on a digital signature scheme and a (two-round) splittable blind signature scheme. Then, we provide a generic construction of SCA based on an OCG protocol and an NIZK proof system. Finally, we explain the practical importance of our SCA protocol from the viewpoint of real-world applications including sensitive attribute verification.

References

[1] Article 9 gdpr – processing of special categories of personal data. https://gdpr-info.eu/art-9-gdpr/. Accessed: 2025-05-23.

[2] World wide web consortium (w3c). verifiable credentials data model v1.1, w3c recommendation 03 march 2022. https://www.w3.org/TR/vc-data-model.

[3] Australian Law Reform Commission. Sensitive information - alrc. https://www.alrc.gov.au/publication/for-your-information-australian-privacy-law-and-practice-alrc-report-108/6-the-privacy-act-some-important-definitions/sensitive-information/. Accessed: 2025-05-23.

[4] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, Heidelberg, August 2009.

[5] Johannes Blömer and Jan Bobolz. Delegatable attribute-based anonymous credentials from dynamically malleable signatures. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18*, volume 10892 of *LNCS*, pages 221–239. Springer, Heidelberg, July 2018.

[6] Johannes Blömer, Jan Bobolz, Denis Diemert, and Fabian Eidens. Updatable anonymous credentials and applications to incentive systems. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 1671–1685. ACM Press, November 2019.

[7] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988.

[8] Jan Bobolz, Fabian Eidens, Stephan Krenn, Sebastian Ramacher, and Kai Samelin. Issuer-hiding attribute-based credentials. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *CANS 21*, volume 13099 of *LNCS*, pages 158–178. Springer, Heidelberg, December 2021.

[9] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, January 2003.

[10] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

[11] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, May 2001.

[12] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Heidelberg, August 2004.

[13] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 1205–1216. ACM Press, November 2014.

[14] Melissa Chase, Trevor Perrin, and Greg Zaverucha. The Signal private group system and anonymous credentials supporting efficient verifiable encryption. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1445–1459. ACM Press, November 2020.

[15] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

[16] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, Heidelberg, August 2006.

[17] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 233–253. Springer, Heidelberg, August 2015.

[18] Sanjam Garg and Divya Gupta. Efficient round optimal blind signatures. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 477–495. Springer, Heidelberg, May 2014.

[19] Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh. Round optimal blind signatures. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 630–648. Springer, Heidelberg, August 2011.

[20] Essam Ghadafi. Efficient round-optimal blind signatures in the standard model. In Aggelos Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 455–473. Springer, Heidelberg, April 2017.

[21] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006.

[22] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

[23] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Round-optimal blind signatures in the plain model from classical and quantum standard assumptions. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 404–434. Springer, Heidelberg, October 2021.

[24] Organisation for Economic Co-operation and Development (OECD). Disability, work and inclusion: Mainstreaming in all policies and practices. https://www.oecd.org/employment/disability-work-and-inclusion/, 2022. Accessed: 2025-05-23.

[25] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 89–114. Springer, Heidelberg, August 2019.

[26] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

[27] UNICEF. Seen, counted, included: Using data to shed light on the well-being of children with disabilities. https://data.unicef.org/resources/children-with-disabilities-report-2021/, 2021. Accessed: 2025-05-23.

[28] U.S. Department of Health and Human Services. Standards for privacy of individually identifiable health information. https://www.hhs.gov/hipaa/for-professionals/privacy/guidance/standards-privacy-individually-identifiable-health-information/index.html. Accessed: 2025-05-23.

[29] Eric R. Verheul. Self-blindable credential certificates from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 533–551. Springer, Heidelberg, December 2001.

[30] World Health Organization. World report on disability. https://www.who.int/publications/i/item/9789241564182, 2011. Accessed: 2025-05-23.

[31] World Health Organization. mhgap: Mental health gap action programme – scaling up care for mental, neurological, and substance use disorders. https://www.who.int/publications/i/item/9789241549790, 2017. Accessed: 2025-05-23.

[32] World Health Organization. Global report on health equity for persons with disabilities. https:

//www.who.int/publications/i/item/9789240063600, 2022. Accessed: 2025-05-23.