Privacy-Preserving Outsourced Certificate Validation

Tarek Galal
Hasso-Plattner-Institute, University of Potsdam
mail@tgalal.com

Anja Lehmann
Hasso-Plattner-Institute, University of Potsdam
anja.lehmann@hpi.de

ABSTRACT

Digital Covid certificates are the first widely deployed end-user cryptographic certificates. For service providers, such as airlines or event ticket vendors, that needed to check that their (global) customers satisfy certain health policies, the verification of such Covid certificates was challenging though — not because of the cryptography involved, but due to the multitude of issuers, different certificate types and the evolving nature of country-specific policies that had to be supported. As Covid certificates contain sensitive health information, their (online) presentation to non-health related entities also poses clear privacy risk. To address both challenges, the EU proposed a specification for outsourcing the verification process to a *validator service*, that executes the process and informs service providers of the result. The WHO announced to adapt this approach for general vaccination credentials beyond Covid-19. While being beneficial to improve security and privacy for service providers, their solution requires strong trust assumption for the (central) validation service that learns all health-related details of the users.

In our work, we propose and formally model a privacy-preserving variant of such an outsourced validation service. Therein the validator learns the attributes it is supposed to verify, but not the users identity. Still, the validator's assertion is blindly bound to the user's identity to ensure the desired user-binding. We analyze the EU specification in our model and show that it only meets a subset of those goals. Our analysis further shows that the EU protocol is unnecessarily complex and can be significantly simplified while maintaining the same (weak) level of security. Finally, we propose a new construction for privacy-preserving certificate validation that provably satisfies all desired goals.

KEYWORDS

digital certificates, authentication, privacy

1 INTRODUCTION

Digital certificates such as X.509 certificates are the backbone of security in communication with servers over the internet. They provide guarantees regarding the identity of servers as well as the authenticity of transmitted data. Although applications for digital certificates are not limited to server-side authentication, this is by far the most prominent application, whereas user-controlled certificates have hardly seen any adoption yet.

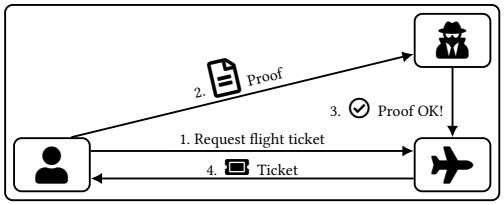

Figure 1: Outsourced Validation Example. An airline requires passengers to prove vaccination status before it issues tickets.

Covid Certificates – the first major deployment of user certificates. The first major roll-out of end-user digital certificates were the so-called Covid-certificates, attesting users that they had been tested, received a vaccination, or recovered from Covid-19. Up to date, more than 1.7 billion [13] certificates have been issued world-wide, and in several countries presenting these credentials even became mandatory for participating in events or traveling [18, 25, 32].

A notable effort for realizing a federated infrastructure is the *EU Digital Covid Certificate* by the European Commission [12]. It specifies a common certificate structure that had to be implemented by the EU member states and requires interoperability of certificates across countries [14]. These are typically simple RSA/ECDSA signatures by one or several trusted issuance authorities per country over a set of user-specific attributes that were encoded in form of QR-codes to facilitate offline-presentation of such certificates.

The Challenge of Federation and Evolving Policies. Comparing the handling of such Covid-certificates with the long-established X.509 server certificates reveals interesting differences: Server certificates contain clear attributes, such as their public key, expiry date, URL and name of organisation, that are verified against a set of predefined and static rules (e.g., expiry date is in the future). In contrast, despite the efforts for setting up a common certificate specification, the exact country-specific instantiations varied greatly and different countries had different policies, both for issuing these credentials as well as what a correct "proof" meant. The latter refers to the question of when a certificate is considered "valid", which could depend on the number (and/or time) of vaccinations a user had received, or whether a recovery status (country-dependent) was sufficient as well. In the case of Covid, these regulations also changed regularly, reflecting the current nationwide efforts to contain the pandemic.

This presents a challenge for verifying parties that had to handle a multitude of certificates from many countries and ensure compliance with a continuously changing set of policies.

Privacy Challenges in Online Scenarios. Another challenge are the privacy implications of such certificates, in particular when used in online scenarios, e.g,. when booking flight or concert tickets, or proving to an employer that certain public health policies were

Proceedings on Privacy Enhancing Technologies 2023(4), 322–340

https://doi.org/10.56553/popets-2023-0113

satisfied in order to get access rights to office facilities or universities [33, 37]. As Covid-certificates are mainly plain ECDSA/RSA signatures, this required sending the full certificate, containing all attributes and sensitive medical data, to private companies for verification. This is clearly not desirable from a privacy perspective, as information of how often (or not all) one is vaccinated, or whether (and when) one had certain infections, is not information that airlines, employers or ticket vendors should process and store. It is also a security risk: as these certificates are not bound to any user keys, sending the original certificate enables misuse.

Outsourced Certificate Validation. To address these challenges in the context of Digital Covid Certificates, an official proposal was set out by the European Commission to enable secure and privacy-friendly verification in online "booking" scenarios [15]. Their solution proposes to outsource the delicate verification to a dedicated new party: the *Validation Service*. Instead of receiving and verifying health certificates themselves, service providers delegate the verification procedures to a trusted validation service which checks the authenticity of certificates and that the contained attributes fulfill a desired policy, and communicates the result back. This offloads the complex management of the verification process to an external party, that can offer its validation service to many relying parties. It also addresses the privacy risk, as the validator only reveals whether a certain user satisfied the policy, but not *how*.

While the validation service proposed by the EU certainly has security and privacy benefits for relying parties, it also introduces new risks: if only one or few validation services exist, then it introduces a single-point of failure. For privacy in particular, the concept of having a single (or few) central entities that collect sensitive health data of millions of citizens is not a desirable prospect.

On the Importance of Validation Services. We stress that the complex verification of user-certificates is not a problem specific to Covid-certificates, but will be a challenge for all user-centric certificates of "soft" attributes that get issued (in a non-standard form) by many issuers and gets validated against flexible policies. With the inevitable rise of digital health services, and efforts such as e-prescription, and general electronic vaccination passports [22, 38], we believe that the verification problem and outsourced validation approach will become even more important. In fact, the WHO already announced that it wants to offer such a vaccination validation service which is "intended to serve as a standard procedure for other vaccinations such as polio or yellow fever after COVID-19." [40]. The company chosen to develop such a service, is the same (and only one) that already announced to offer a service compliant with the EU's specification in Germany [39], which underpins our concern that there will be only very few such providers.

1.1 Contributions

In this work, we introduce the new concept of privacy-friendly validation services for outsourced verification. Their goal is to offload the verification to an external (semi-trusted) entity, to check whether a user owns a certificate that matches a certain policy — but now without the validator learning the user's identity. We formally define this system, analyze the current EU proposal in our model, and propose a new protocol that provides better security and privacy.

Formal Security Model. We start by formally defining such Certificates with Outsourced Validation system (COV) and the desired security and privacy guarantees. Our system model is inspired by the EU specification [15], and includes four entities: issuers, users, relying parties $\mathcal{RP}$ and validation services $\mathcal{VS}$. To express the different information that relying parties and validators are interested in, we consider the issuers' certificates to contain two type of values: user identifying values uid (such as full name and date of birth), and additional attributes $attr$ (which could be the type of vaccination). The validator should only learn the attributes $attr$ and assert whether they satisfy a (public) policy f, whereas the relying party must learn uid and whether this user satisfies the policy (by relying on the result of the validator). In terms of privacy, we consequently want to achieve the following two properties:

VS-Privacy: The validator only learns $attr$ and whether the user owns a valid certificate on it, but does not learn uid or even whether two verification requests belong to the same user.

RP-Privacy: The relying party only learns whether uid satisfies the desired policy, but nothing beyond $f(attr)$ about the user attributes $attr$.

While formalizing these properties is rather simple, the challenge is in capturing the desired unforgeability properties despite the blindness imposed by our two privacy guarantees. That is, despite validation being performed in a partially-blind way, we still want the validation result to be (blindly) bound to the underlying user uid and allow a non-blind verification by the relying party:

Unforgeability: It is infeasible to forge a validation result. If the $\mathcal{RP}$ outsources the certificate validation of a user uid to a trusted $\mathcal{VS}$, and receives positive notification from $\mathcal{VS}$, it is ensured that the user indeed owns a certificate $cert$ on $(uid, attr)$ with $f(attr) = 1$, and this user also initiated the verification request. This property must also hold in the presence of other corrupt validators, which an honest user might have used before.

Without VS- or RP-privacy, formulating and achieving unforgeability is trivial, the challenge is guaranteeing all three properties at the same time.

Analysis of the EU Proposal. We then present the EU proposal for such validation services in our framework and analyze its security. We show that the protocol neither achieves VS-Privacy nor Unforgeability. The former is not surprising, as such privacy was not intended. The lack of (strong) unforgeability stems from the fact that users send their full certificate to the $\mathcal{VS}$, i.e., a corrupt $\mathcal{VS}$ can collect and misuse certificates of honest users (which is considered an attack in our system). We prove that a weaker form of unforgeability, assuming *all* validators to be honest, is achieved, and so is RP-privacy. Our analysis reveals that the EU proposal is overly complicated though, and requires several message flows and cryptographic building blocks that can be omitted without impacting the security of the protocol. Most notably, it lets the user encrypt her certificate and sign the ciphertext under a freshly generated signature key pair for "maximum data privacy" [15]. But the only party ever seeing the ciphertext also knows the decryption

key, and the pseudonymously signed ciphertext contains the users' identifying data – thus neither have an effect in our model.

*Privacy-preserving Protocol (*PP-COV*).* Finally, we present a new protocol that provides outsourced validation and provably achieves all three properties. Our protocol follows the common approach of privacy-preserving authentication and relies for its certificates on a signature scheme that allows selective disclosure of attributes and (non-interactive) proving of ownership of a signature on partially hidden attributes. We stress that these signatures are only used for the issuers' certificates that get validated through the $\mathcal{VS}$. The validators return their result in form of standard signatures, i.e., the relying party will only have to consume standard signatures from the $\mathcal{VS}$ it trusts. The desired unforgeability and user-binding are established by combining such a signature with a commitment scheme, where the commitment (on *uid*) remains closed toward the $\mathcal{VS}$, but the $\mathcal{RP}$ can open and verify that the $\mathcal{VS}$'s validation result belong to the same user that $\mathcal{RP}$ is communicating with.

1.2 Related Work

Outsourcing cryptographic computations to external helpers is not a new idea, and has been extensively studied e.g., in [20, 24, 28]. However, therein the goal is to shift demanding tasks from resource-restricted devices to a more powerful, but potentially malicious entity. While we do outsource signature verification, this is only because we want to outsource the (logical) complexities of policy validation, rather than due to resource constraints.

A closer line of work are Attribute-Based Signatures (ABS) [23, 27, 29], for which outsourced verification has been proposed in [11, 41]. In ABS, a signer owning a set of certified attributes from an attribute authority, can sign a message with a predicate that is satisfied by her attributes, such that the signature only reveals if the predicate is satisfied, but not how. The work on outsourced ABS verification again focuses on shifting the computation-heavy parts to an untrusted helper, but not on splitting the verification over two entities that are supposed to verify different parts of the signature/certificate – which is what our work aims at.

Our privacy-preserving protocol PP-COV shares similarities with protocols known as anonymous or privacy-enhancing credentials [2, 3, 6–10, 19, 34]. Both utilize attribute-based signatures that the user receives from an issuer, which allow for unlinkable presentations of selectively disclosed attributes. Recent works have extended anonymous credentials with an outsourced computation capability [21, 26], albeit for the user's part and not verification, which is our primary focus.

In anonymous credentials, the verifier receives a user's presentation of certified attributes for verification against the issuer's public key. In PP-COV, on the other hand, verification is split among two parties: $\mathcal{RP}$ and $\mathcal{VS}$, each verifying a different subset of the attributes, and the verification performed by $\mathcal{RP}$ mostly relies on the outsourced validation result, which needs to be blindly bound to the underlying user identity (*uid*) that the $\mathcal{RP}$ receives directly. Our main challenge is to formally capture the security and privacy properties of such two-party verification.

In fact, our PP-COV protocol can be seen as an indirect extension of anonymous credentials with an outsourced verification capability. By demonstrating how the verification process can be outsourced in a secure and privacy-friendly manner, we believe that our approach can contribute to a wider adoption of anonymous credentials. However, we stress that our main motivation is to model and analyse the validation system proposed by the EU [15] and supported by the WHO [40], with PP-COV serving as a privacy-enhancing alternative to the existing solution.

2 SYSTEM AND SECURITY MODEL

In this section we introduce our formal model for a Certificates with Outsourced Validation system (COV). Our system model is inspired by the EU specification that introduced such an outsourced validation service for the verification process of Digital COVID Certificates [15]. We start by defining the generic syntax and intended functional behaviour of the system, and then formalize the desired security guarantees through game-based definitions.

High-Level Description. There are four types of entities in a COV system: issuers $\mathcal{I}$, users $\mathcal{U}$, relying parties $\mathcal{RP}$ and validation services $\mathcal{VS}$. A user $\mathcal{U}$ can receive a certificate *cert* from an issuer $\mathcal{I}$ on some unique user identifying information *uid* and a set of attributes *attr*. Later, $\mathcal{U}$ can access a service from a relying party $\mathcal{RP}$ that needs to ensure its users satisfy a certain (public) policy f, i.e., the user identified via *uid* owns a certificate *cert* from a trusted issuer *ipk* on attributes *attr* such that $f(attr, ipk) = 1$. The $\mathcal{RP}$ does not want to verify the certificate itself, but relies on a validation service $\mathcal{VS}$ to learn whether *attr* belonging to *uid* satisfies the policy or not. The $\mathcal{VS}$ ensures the user has a valid certificate, runs $f(attr, ipk)$ on behalf of $\mathcal{RP}$, and communicates the result to it. Note that the result of the policy verification depends on the combination of the certified attributes *attr* and who certified them (*ipk*). This allows us to express that the same attribute can have different meanings depending on the issuer, e.g., an attribute `booster` = 1 might have different meanings in different countries/issuer domains.

Having different parties ($\mathcal{RP}$, $\mathcal{VS}$) interested in different types of information contained in *cert* is what motivates the explicit separation of *uid* and *attr*: $\mathcal{RP}$ wants to know *uid* and the result of $f(attr, ipk)$, whereas $\mathcal{VS}$ only needs to know *attr* such that it can assert if the policy is satisfied, but does not need to know *uid*.

To correctly capture real-world usage, our system supports multiple issuers and multiple validation services. In terms of trust relations, a $\mathcal{VS}$ trusts a certain subset of issuers, denoted as PK, and we assume that trust relation to be publicly known. For simplicity we use the slightly overloaded notation $f(attr, ipk \in PK) = 1$ to express that the $\mathcal{VS}$ does not only check that the combination of $(attr, ipk)$ satisfies the policy f, but also that *ipk* is in its (publicly known) set of trusted issuers PK.

2.1 Syntax

We now define the generic syntax of a COV system. We start by informally describing the algorithms and their use, and then provide the formal definition.

Setup & Policy. Our system is initialized by the Setup algorithm that generates public parameters *pp* for a security parameter 1^λ that will be considered as implicit input to all algorithms. Both, issuers

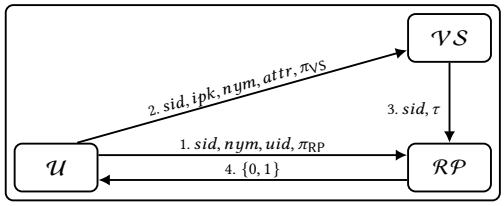

Figure 2: Flow of messages in a joint validation session *sid* of a user proving ownership of *cert* on $(uid, attr)$ by issuer *ipk*.

and validators have dedicated key pairs, generated via IKGen as (ipk, isk) and via VKGen as (vpk, vsk) respectively.

We assume that each validator is publicly known to handle a specific policy f and trusts a set of issuers without making that explicit. That is, we assume that each validator public key vpk is implicitly bound to particular policy f and a set of trusted issuers' public keys PK that are known to all parties involved. In a real deployment, $\mathcal{VS}$'s public keys are managed over a PKI, and the certificates on these keys would most likely also include these information.

Certificates. An issuer generates a certificate over a particular $(uid, attr)$ belonging to some user by running CertIssue with their isk, and passes $cert$ to the user who can verify its correctness using CertVerify. We assume uid to be globally unique.

Outsourced Validation Process. Users in possession of a certificate can then prove to some $\mathcal{RP}$ that they have attributes certified by a trusted issuer $\mathcal{I}$ with $f(attr, ipk \in PK) = 1$, using a validator $\mathcal{VS}$ identified through vpk (that is known to verify f for trusted issuer public keys PK) for the policy check. As the user might request several of these showings from the $\mathcal{VS}$, we use a unique *session identifier sid* in all algorithms of this process to distinguish different sessions. More precisely, we assume the session id to be of the form $sid = (\mathcal{VS}, \mathcal{RP}, sid')$ where $\mathcal{VS}, \mathcal{RP}$ identify the involved parties and sid' is a fresh and unique string for each session.

The validation process is initiated by a user $\mathcal{U}$ who chooses a $\mathcal{VS}$ and runs CertPresent on her certificate and the public keys of that $\mathcal{VS}$ and the certificate's issuer, obtaining three output values: nym, π_{VS}, π_{RP}. The value nym can be seen as a pseudonym that is given to both the $\mathcal{RP}$ and $\mathcal{VS}$ and (blindly) binds the session to $\mathcal{U}$, whereas π_{VS} is a *presentation token* for the $\mathcal{VS}$ and π_{RP} is $\mathcal{RP}$-specific verification information.

$\mathcal{U}$ passes the produced presentation token π_{VS}, pseudonym nym and her attributes $attr$ to $\mathcal{VS}$. The $\mathcal{VS}$ processes the received input via $\mathsf{ValTokGen}_{f,PK}$, which internally checks if $f(attr, ipk \in PK) = 1$. Depending on the result of its check, it returns a *validation token* τ, and sends it along with sid to $\mathcal{RP}$. Finally, $\mathcal{RP}$ can combine this token with the information (uid, nym, π_{RP}) it received from $\mathcal{U}$ for sid. It runs ValTokVerify to verify whether the user uniquely identified by uid, and in this session sid associated with pseudonym nym, has been attested by the validator to own a certificate that satisfies the desired policy f (implicitly defined via vpk). Figure 2 visualizes the exchanged messages during a validation session.

Formally, a COV scheme for $(\lambda, sid) \in \mathbb{Z}^2$ and attribute space $\mathcal{D}$ consists of the following algorithms:

$\mathsf{Setup}(1^\lambda)$**:** On input of security parameter 1^λ outputs public parameters pp, implicitly made available to all algorithms.

$\mathsf{IKGen}(pp) \to (ipk, isk)$**:** Outputs issuer's public key ipk and secret key isk.

$\mathsf{VKGen}(pp) \to (vpk, vsk)$**:** Output $\mathcal{VS}$'s key pair. We assume every $\mathcal{VS}$ key pair implicitly defines a policy verification function f and a set of trusted issuers' public keys PK.

$\mathsf{CertIssue}(isk, uid, attr) \to cert$**:** Run by $\mathcal{I}$ to generate a $cert$ for the given $uid, attr$.

$\mathsf{CertVerify}(ipk, uid, attr, cert) \to \{0, 1\}$**:** Run by $\mathcal{U}$ to verify a $cert$ relative to the given $ipk, uid, attr$.

$\mathsf{CertPresent}(ipk, vpk, uid, attr, sid, cert) \to (\pi_{VS}, \pi_{RP}, nym)$**:** Run by $\mathcal{U}$, outputs a pseudonym nym for the given uid, pseudonym verification info π_{RP}, and a presentation token π_{VS} for the given $ipk, vpk, uid, attr, sid$, and nym.

$\mathsf{ValTokGen}_{f,PK}(vsk, ipk, attr, sid, nym, \pi_{VS}) \to \{\tau, \perp\}$**:** This is run by $\mathcal{VS}$ to verify π_{VS} w.r.t. the given ipk, $attr$, sid, nym and outputs a validation token τ or $\perp$. f and PK denote the (non-cryptographic) policy that is implicitly defined via vpk and asserted through a token τ.

$\mathsf{ValTokVerify}(vpk, uid, sid, nym, \pi_{RP}, \tau) \to \{0, 1\}$**:** Run by $\mathcal{RP}$, outputs 1 if τ is valid w.r.t. to vpk for the given uid, sid, nym, π_{RP}, or 0 otherwise.

2.2 Security Model

In this section we formally define our three security goals: VS-Privacy (a corrupt $\mathcal{VS}$ learns nothing about uid), RP-Privacy (a corrupt $\mathcal{RP}$ learns nothing about $attr$), and Unforgeability.

We model the security goals as games where an adversary $\mathcal{A}$ interacts with a challenger in control of the honest parties in every game. In the following, we give an intuition for each goal and then argue how this is captured in our formal model.

Notation & Conventions. We use the shortcut $(key, -) \in Q$ and $key \in Q$ synonymously to check if any tuple in the set Q contains a given value key. Similarly, we write $(key, val) \in Q$ for a given input key to look up the corresponding values val stored for this key, which implicitly also checks that $key \in Q$.

Further, all our games require multi-stage adversaries $\mathcal{A}$, and we assume $\mathcal{A}$ keeps state across all stages without making it explicit.

Finally, recall that we identify all session through unique $sid = (\mathcal{VS}, \mathcal{RP}, sid')$ where the first part identifies the involved $\mathcal{VS}$ and $\mathcal{RP}$ and sid' must be unique (for that combination of parties). In practice, sid' will simply be a concatenation of nonces provided by both parties. For simplicity, we assume in our model (and construction), that all sid's are well-formed and unique without making that explicit in all oracles and algorithms.

2.2.1 *Unforgeability.*

This is the core property of our scheme, and it guarantees unforgeable validation tokens if the $\mathcal{RP}$ relies on an honest validator that in turn trusts only honest issuers. Note that both trust assumptions are inherent requirements for any meaningful unforgeability guarantee, as a corrupt validator could generate any tokens it wants, and an honest validator relying on corrupt issuers could also be fed arbitrary information that it would validate.

We stress that while the exact validator the $\mathcal{RP}$ relies on must be honest, our security model also captures corrupt users and validators. It models that corrupt users get certificates from honest issuers, and honest users present certificates to corrupts validators, and ensures security in the presence of these corrupt entities.

Roughly, unforgeability guarantees that $\mathcal{RP}$ accepts the final validation token τ for a user uid only if the user legitimately obtained a certificate from an honest issuer with policy-passing $attr$ and intended to have a session identified through (sid, nym) with an honest $\mathcal{VS}$ who trusts that issuer. Thus, this property is expressed for the final verification where the input from the user and $\mathcal{VS}$ are used together in $\mathsf{ValTokVerify}(vpk, uid, sid, nym, \pi_{\mathsf{RP}}, \tau)$. This combination of distributedly and partially blindly derived information is what makes defining unforgeability challenging. In fact, unforgeability encapsulates several security aspects of our system:

(1) infeasibility to create a valid τ for a session identified by (sid, nym) that was never approved by the honest $\mathcal{VS}$ *(direct unforgeability)*,
(2) infeasibility to get the honest $\mathcal{VS}$ to create τ for user uid who does not possess the necessary certificate with policy passing attributes from a trusted issuer *(indirect unforgeability)*,
(3) if the $\mathcal{VS}$ in a session (sid, nym) with $\mathcal{RP}$ has checked certain attributes $(attr, ipk)$ (that satisfy the policy) and generated a token τ, then only a user uid owning a certificate for $attr$ from ipk can use τ for (sid, nym) towards $\mathcal{RP}$ *(user binding)*,

Note that *user binding* is stronger than the *indirect forgery* requirement: it guarantees that when a $\mathcal{VS}$ has validated $(attr, ipk)$ in a session for (sid, nym), then the $\mathcal{RP}$ is ensured that the uid it interacts with in the same session indeed owns a valid certificate. More precisely, the following attack is captured through user-binding and prevented in any scheme satisfying our unforgeability notion: let uid_1 and uid_2 be two corrupt users where only uid_1 has a policy-compliant certificate. Then it is impossible that uid_1 shows her certificate (blindly) to $\mathcal{VS}$ in a session (sid, nym), but the returned validation token τ verifies correctly for uid_2.

For *honest* users uid we want even stronger guarantees, that ensure that any final validation token τ for sid, nym and uid does indeed stem from a validation session initiated by that particular user. This comprises the following two aspects, where the second captures security in the presence of corrupt $\mathcal{VS}$'s.

(4) if an honest user uid and honest $\mathcal{VS}$ engage in session (sid, nym), where uid has a correct certificate for $(attr, ipk)$ and the validator generates a token τ, then only this exact user uid can use τ for (sid, nym) *(strict user binding)*,
(5) an adversary (being a corrupt $\mathcal{VS}$) receiving presentation tokens from an honest user for session (sid, nym), cannot reuse that information to impersonate the honest user towards an honest $\mathcal{VS}$ *(impersonation resistance)*.

User Binding vs. Strict User Binding. One might wonder why there is a difference for honest and corrupt users regarding the binding guarantees, and why we have only a weaker form for corrupt users. This is in fact an inherent limitation that stems from certificate validation being done *partially-blindly* via the $\mathcal{VS}$. When a corrupt user presents her certificate towards the honest $\mathcal{VS}$, our algorithms (and oracles) do not reveal any information about who that user is. All we know are the (sid, nym) she used in that session and her information $attr, ipk$ to be verified. Thus, due to the blindness of that process we simply cannot express who the exact user is. This is similar to the situation of blind signatures, that cannot achieve standard unforgeability, but only the weaker form, one-more-unforgeability. The unforgeability we ensure for corrupt users is significantly stronger than classic one-more-unforgeability though, and leverages the fact that during both, issuance and final verification, the user uid is known again. More precisely, we know $(sid, nym, attr, ipk)$ from the presentation towards $\mathcal{VS}$, we know what information $(attr', ipk')$ for uid got certified from an honest issuer, and in the final verification run by $\mathcal{RP}$ we know the user identity uid again, and we require all information to be consistent.

We already explained above that this prevents corrupt users uid_1 and uid_2 to "pool" their certificates, i.e., a corrupt user uid_2 not having the required certificate cannot exploit the blind validation to falsely present a certificate from uid_1 to $\mathcal{VS}$. What is not considered an attack though, is that two corrupt users uid_1 and uid_2 both owning valid certificates on the same policy-compliant values $(attr, ipk)$ can mix their sessions. That is, uid_1 shows her certificate to $\mathcal{VS}$, but the returned validation token is used by uid_2 towards $\mathcal{RP}$. As uid_2 has the same certificate that would have also passed the validation, we consider this a benign attack and avoid the use of extractors or constrained adversaries to exclude them in the model.

The reason why we can ensure stronger security for honest users is that they are controlled by the challenger in the security model, i.e., we know exactly which user has engaged in which session and can express that only this user can present the validation token, i.e., even users with the same certificates cannot "hijack" their sessions.

Oracles. We model corrupt users via O_{CU} that the adversary can use to receive certificates from an honest issuer ipk_i over $uid, attr$ of its choosing. The adversary can request issuance of honest user certificates using O_{HU} which stores the generated certificate internally, and allows subsequent use of them in form of presentation tokens. These two oracles ensure global uid uniqueness by not issuing more than one certificate per uid.

To interact with the honest "challenger" $\mathcal{VS}$ (with vpk), there are two different oracles depending on whether $\mathcal{A}$ uses $\mathcal{VS}$ directly as a corrupt user (via O_τ) or requests a token for an honest user (via $O_{\pi,\tau}$). In the latter case, this internally runs the honest presentation token generation and verification, and returns the validation token τ as well as nym, π_{RP} to the adversary (posing as corrupt $\mathcal{RP}$).

Finally, we use O_π to model malicious validators in the system and capture the desired impersonation resistance. That is, this oracle can be used to ask honest users (generated via O_{HU}) to create presentation tokens for validator keys $vpk_j \neq vpk$ of $\mathcal{A}$'s choice.

Unforgeability Game. Our formal unforgeability game assumes a maximal number ℓ of honest issuers, and starts by generating key pairs for all of them (defined as L_{honest}), as well as a key pair for the honest $\mathcal{VS}$. The adversary gets all public issuer keys and oracle access to trigger certificate issuance (via O_{CU} and O_{HU}). For honest users, he can already request presentation tokens for malicious validators of his choice (via O_π).

To start the challenge phase, we allow the adversary to choose any subset $PK \subseteq L_{honest}$ of issuers that will be considered trusted,

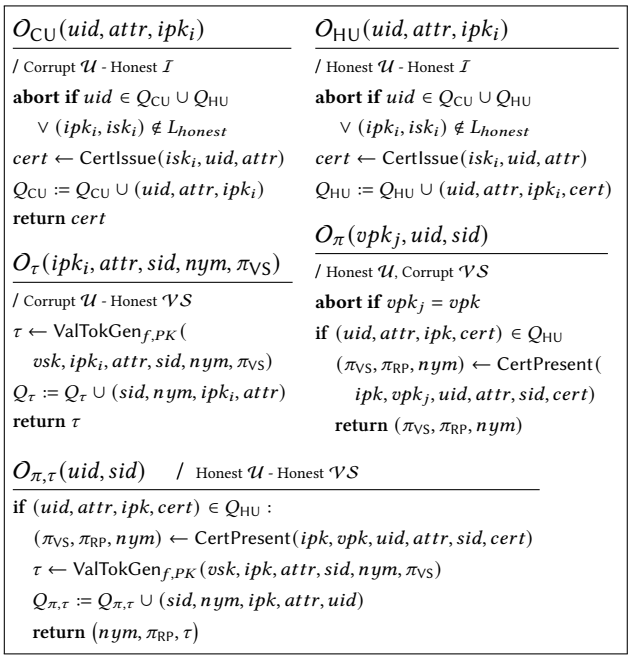

Figure 3: Oracles for our security definitions. If an oracle receives input uid, the line $(uid, attr, cert, ipk_i) \in Q_{\mathsf{HU}}$ expresses that we retrieve $(attr, cert, ipk_i)$ for that uid from Q_{HU}, or abort if no such entry is found.

as well as a policy f. We now implicitly bind the honest validator's key pair to PK and f, by giving $\mathcal{A}$ additional oracle access to the honest validator (via O_τ and $O_{\pi,\tau}$) that internally runs f for PK. The task of the adversary is to output $(sid^*, uid^*, nym^*, \pi^*_{\mathsf{RP}})$ together with a token τ^* that verifies correctly under vpk but is a non-trivial forgery. What non-trivial means is captured through four winning conditions out of which at least one must be satisfied.

Case 1: The forgery is for a user uid^* that has never received any certificate from any honest issuer. This can happen either through a direct or indirect forgery.

Case 2: Here the forgery is for a *corrupt* user uid^* that has a certificate for $(attr, ipk)$, but there is no recorded validation session via the honest $\mathcal{VS}$ for $(sid^*, nym^*, ipk, attr)$. This could either mean that there was no query for (sid^*, nym^*) at all (direct forgery), or there was a validated session $(sid^*, nym^*, attr', ipk')$ but for $(attr', ipk') \neq (attr, ipk)$ (user binding).

Case 3: Now the forgery is for an *honest* user uid^* that has a certificate for $(attr, ipk)$, but there is no recorded validation session via the honest $\mathcal{VS}$ for $(sid^*, nym^*, ipk, attr, uid^*)$ (note the extra uid^* in the tuple). This could again mean that there was no query for (sid^*, nym^*) at all (direct forgery), or there was a validated session $(sid^*, nym^*, ipk', attr', uid')$ but for $(attr', ipk', uid') \neq (attr, ipk, uid^*)$ (strict user binding). Note that we do not require any conditions on O_π here, i.e., if $(sid^*, uid^*, nym^*, \pi^*_{\mathsf{RP}})$ are from a presentation of the honest user towards a corrupt validator, this is a valid forgery too (impersonation resistance).

Case 4: This is for a user uid^* (honest or corrupt) that owns a certificate on $(attr, ipk)$ which does not satisfy the policy. This can again happen through direct or indirect forgery.

Definition 2.1 (Unforgeability). COV is unforgeable if for all PPT adversaries $\mathcal{A}$, $\Pr[Exp^{\mathsf{UF}}_{\mathcal{A},\mathsf{COV}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

$Exp^{\mathsf{UF}}_{\mathcal{A},\mathsf{COV}}(\lambda) \rightarrow \{0,1\}$ / For max. number of honest issuers ℓ

$(pp) \leftarrow \mathsf{Setup}(1^\lambda)$; $(vpk, vsk) \leftarrow \mathsf{VKGen}(pp)$

for $i := 1, ..., \ell$: $(ipk_i, isk_i) \leftarrow \mathsf{IKGen}(pp)$, $L_{honest} := \{ipk_i, isk_i\}_1^\ell$

$PK, f \leftarrow \mathcal{A}^{O_{\mathsf{CU}}, O_{\mathsf{HU}}, O_\pi}(pp, ipk_1, ..., ipk_\ell)$

abort if $PK \notin L_{honest}$, initialize $O_{\pi,\tau}, O_\tau$ with PK, f

$(sid^*, uid^*, nym^*, \pi^*_{\mathsf{RP}}, \tau^*) \leftarrow \mathcal{A}^{O_{\mathsf{CU}}, O_{\mathsf{HU}}, O_\pi, O_{\pi,\tau}, O_\tau}(vpk)$

return 1 if $\mathsf{ValTokVerify}(vpk, uid^*, sid^*, nym^*, \pi^*_{\mathsf{RP}}, \tau^*) = 1$

and at least one the following holds:

Case 1 : $(uid^*, -, -) \notin Q_{\mathsf{CU}} \cup Q_{\mathsf{HU}}$

Case 2 : $(uid^*, attr, ipk) \in Q_{\mathsf{CU}} \wedge (sid^*, nym^*, ipk, attr) \notin Q_\tau$

Case 3 : $(uid^*, attr, ipk, -) \in Q_{\mathsf{HU}} \wedge (sid^*, nym^*, ipk, attr, uid^*) \notin Q_{\pi_{\mathsf{VS}},\tau}$

Case 4 : $(uid^*, attr, ipk, -) \in Q_{\mathsf{CU}} \cup Q_{\mathsf{HU}} \wedge f(attr, ipk \in PK) = 0$

Weak Unforgeability. For the EU specification, we will only be able to show a weaker form of unforgeability in which re-using presentation tokens received by a corrupt validator is not considered a forgery, i.e., impersonation resistance is no longer guaranteed. This is what we denote as *weak unforgeability*, and can be easily derived from the definition above, by simply forbidding the adversary to receive presentations from honest users for malicious $\mathcal{VS}$'s. We denote $Exp^{\mathsf{weakUF}}_{\mathcal{A},\mathsf{COV}}$ to be the game that is equivalent to $Exp^{\mathsf{UF}}_{\mathcal{A},\mathsf{COV}}$ except that $\mathcal{A}$ therein is not getting access to the O_π oracle.

2.2.2 VS-Privacy. This property captures the privacy guarantees towards a corrupt $\mathcal{VS}$: a corrupt $\mathcal{VS}$ receiving $(ipk, attr, nym, \pi_{\mathsf{VS}})$ only learns whether $f(attr, ipk \in PK) = 1$, indicating that the user owns a certificate $cert$ from ipk on policy-compliant attributes $attr$, but does not learn uid (also contained in $cert$).

In this game, both the validator and issuer(s) can be corrupt, but the users for which we want to ensure privacy are honest as well as the $\mathcal{RP}$ she authenticates to. Recall that the $\mathcal{VS}$ must learn $(attr, ipk)$ to perform its policy check, but shouldn't learn anything about uid or the concrete certificate $cert$ the user owns. The latter might be a bit surprising, given that we want the validator to check that the user owns an appropriate certificate. However, requiring $cert$ to remain secret is necessary, as the certificate would otherwise serve as a unique identifier each user has, which would immediately ruin any hope for privacy (when both $\mathcal{VS}$ and $\mathcal{I}$ can be corrupt).

The game $Exp^{\mathsf{PRIV}\text{-}\mathcal{VS}}_{\mathcal{A},\mathsf{COV}}$ follows a classic indistinguishability experiment where $\mathcal{A}$ outputs two pairs $(uid_i, cert_i)_{i\in\{0,1\}}$ and the common values $(ipk, vpk, attr, sid)$, then receives the presentation (π_{VS}, nym) for either of the tuples, and must tell the origin.

The requirement that our game does not allow $\mathcal{A}$ to specify different $(ipk, vpk, attr, sid)$ for each user is necessary, as these values are required inputs for the validator's algorithm $\mathsf{ValTokGen}_{f,PK}$, and thus user-specific values would immediately allow $\mathcal{A}$ to distinguish the users. Note that this game captures the aforementioned unlinkability even though we don't provide any user oracle: $\mathcal{A}$ knows the certificates of both users and can generate presentations for the concrete users himself.

Definition 2.2 (VS-Privacy). COV satisfies VS-Privacy if for all PPT adversaries $\mathcal{A}$: $\Pr[Exp^{\text{PRIV-}\mathcal{VS}}_{\mathcal{A},\text{COV}}(\lambda) = 1] - \frac{1}{2} \leq \text{negl}(\lambda)$.

$Exp^{\text{PRIV-}\mathcal{VS}}_{\mathcal{A},\text{COV}}(\lambda) \rightarrow \{0,1\}$

$pp \leftarrow \text{Setup}(1^\lambda)$; $\quad b \leftarrow_R \{0,1\}$

$(ipk, vpk, attr, sid, (uid_i, cert_i)_{i\in 0,1}) \leftarrow \mathcal{A}(pp)$

return 0 **if** $\text{CertVerify}(ipk, uid_0, attr, cert_0) = 0 \vee$
$\text{CertVerify}(ipk, uid_1, attr, cert_1) = 0$

$(\pi_{\text{VS}}, \pi_{\text{RP}}, nym) \leftarrow \text{CertPresent}(ipk, vpk, uid_b, attr, sid, cert_b)$

$b^* \leftarrow \mathcal{A}(\pi_{\text{VS}}, nym)$

return 1 **if** $b = b^*$ **else** 0

2.2.3 RP-Privacy. Our second privacy property captures the privacy of honest users towards a corrupt $\mathcal{RP}$. We again allow the issuer to be corrupt, but now the validator must be honest (as there is no privacy when all central entities are corrupt and collude).

RP-Privacy requires that none of the values the $\mathcal{RP}$ receives, reveal any information about $attr, ipk$ (beyond $f(attr, ipk \in PK) = 1$) of the user's underlying certificate. As the $\mathcal{RP}$ receives $(vpk, uid, sid, nym, \pi_{\text{RP}}, \tau)$ where π_{RP}, nym and τ stem from algorithms that received the $(attr, ipk)$ (or even the user's certificate) as input, we formulate the indistinguishability challenge for them.

The game initializes the adversary with an honest $\mathcal{VS}$'s public key and gives it access to O_τ (as defined in Fig.3). The adversary outputs common values (uid, sid), and two pairs $(ipk_i, attr_i, cert_i)_{i\in\{0,1\}}$, again modelling the corrupt issuer. The challenger picks one of the tuples randomly, first generates a presentation token $(\pi_{\text{VS}}, \pi_{\text{RP}}, nym)$, and then also derives a validation token τ for vsk and the user's token. To prevent trivial wins, we require that both pairs output by $\mathcal{A}$ must either pass the validators policy check, or not, but cannot lead to different responses. The adversary in the role of the corrupt $\mathcal{RP}$ receives $(\pi_{\text{RP}}, nym, \tau)$, and wins if it correctly determines the used tuple.

Note that here the uid must be the same for both challenge requests, whereas the issuer can be different. This reflects the different setting when compared with $Exp^{\text{PRIV-}\mathcal{VS}}_{\mathcal{A},\text{COV}}$: here the $\mathcal{RP}$ learns uid (and thus there is no privacy for the user's identity), but must not learn anything about the user's concrete credential (which includes the identity of the issuer – which was required information in $\mathcal{VS}$'s operation though).

Definition 2.3 (RP-Privacy). COV satisfies RP-Privacy if for all PPT adversaries $\mathcal{A}$: $\Pr[Exp^{\text{PRIV-}\mathcal{RP}}_{\mathcal{A},\text{COV}}(\lambda) = 1] - \frac{1}{2} \leq \text{negl}(\lambda)$.

$Exp^{\text{PRIV-}\mathcal{RP}}_{\mathcal{A},\text{COV}}(\lambda) \rightarrow \{0,1\}$

$(vpk, vsk) \leftarrow \text{VKGen}(pp)$, $pp \leftarrow \text{Setup}(1^\lambda)$; $\quad b \leftarrow_R \{0,1\}$

$(uid, sid, (ipk_i, attr_i, cert_i)_{i\in 0,1}) \leftarrow \mathcal{A}^{O_\tau}(pp, vpk)$

return 0 **if** $f(attr_0, ipk_0 \in PK) \neq f(attr_1, ipk_1 \in PK) \vee$
$\text{CertVerify}(ipk_0, uid, attr_0, cert_0) = 0 \vee$
$\text{CertVerify}(ipk_1, uid, attr_1, cert_1) = 0$

$(\pi_{\text{VS}}, \pi_{\text{RP}}, nym) \leftarrow \text{CertPresent}(ipk_b, vpk, uid, attr_b, sid, cert_b)$

$\tau \leftarrow \text{ValTokGen}_{f,PK}(vsk, ipk_b, attr_b, sid, nym, \pi_{\text{VS}})$

$b' \leftarrow \mathcal{A}^{O_\tau}(\pi_{\text{RP}}, nym, \tau)$

return 1 **if** $b = b'$ **else** 0

3 SECURITY ANALYSIS OF EU-COV

In this section we present EU-COV, which is the EU specification for outsourced validation (EU-Spec, [15]), adapted to our syntax and overall system model. We analyze EU-COV in our model, which shows that it satisfies RP-Privacy but not VS-Privacy and only a weaker form of unforgeability (assuming all validators to be honest). At the end of this section, we also explain where EU-COV deviates from the actual specification EU-Spec, and argue what our results mean for EU-Spec. Our analysis reveals that several cryptographic building blocks have no impact on security, and allow to simplify the EU-COV protocol without harming security or privacy.

3.1 Standard Building Blocks

Before describing the EU-COV construction we briefly introduce the generic building blocks it is based on. For consistency with our new protocol we state both building blocks of signatures and encryption to be instantiated for public parameters pp instead of a security parameter 1^λ, but in the rest of this section this can simply be understood as $pp = 1^\lambda$.

Signature Schemes. The construction uses standard signature schemes $\mathsf{S} = (\text{KGen}, \text{Sign}, \text{Verify})$ with key generation $(pk, sk) \leftarrow \text{KGen}(pp)$, sign algorithm $\sigma \leftarrow \text{Sign}(sk, m)$, and verify function $d \leftarrow \text{Verify}(pk, m, \sigma)$. The standard security property for signatures is *existential unforgeability under chosen message attacks* (EUF-CMA). It requires that an adversary, knowing pk and after seeing several signatures for messages of his choice, is not able to create a valid signature for a message that was never signed by sk's owner.

Public-key Encryption. The second building block is a public-key encryption scheme $\mathsf{E} = (\text{KGen}, \text{Enc}, \text{Dec})$, comprising of key generation $(epk, esk) \leftarrow \text{KGen}(pp)$, an encryption algorithm $c \leftarrow \text{Enc}(epk, m)$, and $m \leftarrow \text{Dec}(esk, c)$ for decryption. The typical security properties of encryption schemes are either IND-CPA or IND-CCA security, but we omit their definitions as our security analysis will not require any property from E beyond correctness.

3.2 EU-COV Construction

We now present the EU-COV construction adapted to our syntax, the full protocol is shown in Figure 4 which uses $\text{Setup}(1^\lambda)$ that simply sets $pp := 1^\lambda$. It uses an encryption scheme $\mathsf{E} = (\text{KGen}, \text{Enc}, \text{Dec})$, and three signature schemes, which we make explicit by denoting them S_I, S_VS and S_U.

Key Generation and Certificate Issuance. The issuer's key simply consists of a key pair of the signature scheme S_I. Consequently, certificates are standard signatures on the given $(uid, attr)$ under the issuer's secret key. The validator's key consists of two parts: a signing key pair from S_VS and an encryption key pair from E.

Outsourced Validation. When a user wants to present her certificate $cert$ towards a validator, she encrypts her certificate and $uid, attr$ under the public key of the targeted $\mathcal{VS}$. In addition, she generates a fresh and ephemeral key pair of the signature scheme S_U and signs the ciphertext and session id sid with her ephemeral secret key. The public key serves as her pseudonym nym as well as $\mathcal{RP}$-specific authenticator π_{RP}. The presentation token π_{VS} for the $\mathcal{VS}$ consists of the ciphertext and her signature on it.

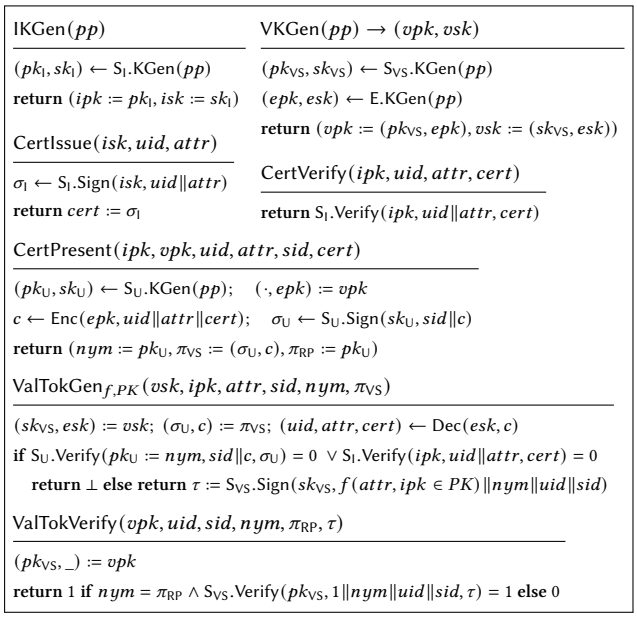

Figure 4: Generic Construction for EU-COV. We assume that every $\mathcal{VS}$ key pair implicitly defines a policy verification function f and a set of trusted issuers' public keys PK.

The validator (knowing to assert the policy f for trusted issuers PK) takes the presentation $\pi_{VS} = (\sigma_U, c)$, and first verifies the user's signature under the provided (pseudonym) key $nym = pk_U$. It then decrypts the ciphertext to obtain $cert$ and $uid, attr$ and verifies that $cert$ is a valid signature on $(uid, attr)$ under the issuer's public key. If both signatures passed, the $\mathcal{VS}$ then computes $f(attr, ipk \in PK)$ and signs the result ("1" if the policy is satisfied, and "0" otherwise) along with pk_U, uid, sid.

Finally, ValTokVerify simply verifies the $\mathcal{VS}$ signature τ w.r.t. $(1\|pk_U\|uid\|sid)$ (where 1 denotes that the policy was satisfied).

3.3 Security Analysis

In this section we analyze EU-COV w.r.t. to the security goals we defined in Section 2.2. We show that it neither achieves VS-Privacy nor Unforgeability, but RP-Privacy as well as a weaker form of unforgeability. Full proofs are available in Appendix C.

Unforgeability. EU-COV does not satisfy our Unforgeability notion as it does not guarantee the desired *impersonation resistance*. Recall that the adversary $\mathcal{A}$ in our game can request honest users to send presentation tokens to malicious validators (via O_π), and $\mathcal{A}$ wins if it can re-use such a presentation towards the honest $\mathcal{VS}$. This attack is covered in Case 3 of $Exp^{\mathsf{UF}}_{\mathcal{A},\mathsf{COV}}$, which $\mathcal{A}$ can trivially win for the EU-COV construction, as the user sends her original certificate and all attributes to the malicious validator. Thus, after having received one presentation from an honest user uid, the adversary can now fully impersonate her towards the honest $\mathcal{VS}$.

The user signature might have had the intention to prevent such attack, but as the user's signature is done via a fresh ephemeral key, there is nothing that binds her key pair to her certificate.

3.3.1 Weak Unforgeability. Despite not satisfying Unforgeability, we show that EU-COV provides Weak Unforgeability. Recall that it is a weaker version where the adversary does not get O_π. In practice, this means that EU-COV has no impersonation resistance and unforgeability is only guaranteed in a setting where *all* validators are honest.

Theorem 3.1. *If the schemes* $\mathsf{S_I}$ *and* $\mathsf{S_{VS}}$ *are EUF-CMA secure, then* EU-COV *is weakly-unforgeable according to* $Exp^{\mathsf{weakUF}}_{\mathcal{A},\mathsf{COV}}$.

Note that in the above theorem we do not require any property from $\mathsf{S_U}$ or E. The latter is easy to see, as the encryption scheme is only used in π_{VS} which is never exposed to $\mathcal{A}$ in the weak unforgeability game. The user's signature scheme $\mathsf{S_U}$ also does not serve any purpose, as there is no relation between her key pair and her certificate. The only important part is that her ephemeral public pk_U, which also serves as pseudonym, is signed by the $\mathcal{VS}$, to bind the final validation token to nym as required by our definition. But this fully relies on $\mathsf{S_{VS}}$, not on $\mathsf{S_U}$.

Proof (Sketch). Assume an adversary wins in $Exp^{\mathsf{weakUF}}_{\mathcal{A},\mathsf{COV}}$, i.e., it outputs $(uid^*, sid^*, nym^*, \pi^*_{RP}, \tau^*)$ such that ValTokVerify$(vpk, uid^*, sid^*, nym^*, \pi^*_{RP}, \tau^*) = 1$ and one of the Cases 1-4 of the game is satisfied. Note that ValTokVerify outputs 1 only if τ^* is a valid $\mathsf{S_{VS}}$ signature over $(1\|nym^*\|uid^*\|sid^*)$.

We show that winning $Exp^{\mathsf{weakUF}}_{\mathcal{A},\mathsf{COV}}$ translates to a forgery either for $\mathsf{S_I}$ or $\mathsf{S_{VS}}$. In all four cases we have the same two sub-conditions:

(1) either τ^* validates a session (sid^*, nym^*) that was indeed certified by the honest $\mathcal{VS}$,
(2) or τ^* is for a session (sid^*, nym^*) that was never approved by $\mathcal{VS}$ (which implies a direct forgery for $\mathsf{S_{VS}}$)

For the first sub-condition, the concrete argumentation is different in each case, but always concludes that the $\mathcal{VS}$ must have received a valid certificate $(uid^*, attr)$ from a trusted issuer ipk as part of π_{VS}, that is inconsistent with the information that got certified by the honest issuers through O_{CU} and O_{HU}. We then show that this yields a forgery for $\mathsf{S_I}$. The full proof is in Appendix C.1. □

VS-Privacy. It is obvious that EU-COV cannot satisfy VS-Privacy, which guarantees that the validator does not learn anything about uid or $cert$. Here the validator receives both. This information is encrypted in π_{VS}, but under the key of the validator, so this isn't providing any privacy when the validator is corrupt. Thus, the use of the encryption scheme does not serve any purpose, but could lead to the misconception that EU-COV is privacy friendly.

RP-Privacy. It is easy to see that EU-COV guarantees privacy towards corrupt $\mathcal{RP}$'s and does so information-theoretically.

Theorem 3.2. EU-COV *satisfies RP-Privacy according to Def. 2.3.*

Proof. In the RP-Privacy game, $\mathcal{A}$ receives π_{RP}, nym, τ derived for $(ipk_b, attr_b, cert_b)$ and must determine b. Both nym and π_{RP} are the user's ephemeral session public key pk_U, generated independently from any of her long-term values. The validation token is merely a signature on $1\|nym\|uid\|sid$ and thus does not depend on the challenge values or b either. Hence, all information the adversary sees is entirely independent of b, and thus he can only win by guessing, which has probability $\frac{1}{2}$ to be correct. □

3.4 Difference to the Original Specification

Note that EU-COV is not fully compliant with the original EU Specification (EU-Spec) [15]. We made a number of syntactical changes that were necessary to fit our system model and syntax, while preserving as much of the original protocol as possible. We provide a visual comparison of both protocols, EU-COV and EU-Spec, and their message flows during a validation session in Figure 5. In this section we explain where EU-COV deviates from EU-Spec and justify what our results for EU-COV mean for EU-Spec.

An immediate question is why our model is not generic enough to fit the original specification. The reason is that EU-Spec has an unnecessarily complex communication scheme, which would have required to model the $\mathcal{VS}$'s part ($\text{ValTokGen}_{f,PK}$) as an interactive multi-party protocol instead of a non-interactive algorithm: In EU-Spec, the $\mathcal{VS}$ simultaneously receives inputs from the user and $\mathcal{RP}$, and the input of the user partially stems from $\mathcal{RP}$. None of that message complexity is necessary though, but would significantly complicate our formal model, and the unforgeability notion in particular. Thus, we rather opted for a simpler model and stripped out just enough redundancies to make EU-Spec fit our model.

We argue in the following sections why these modifications do not affect the security and privacy intended by EU-Spec. A nice side effect is that EU-COV serves as an easy optimization upgrade for EU-Spec that remains compatible with existing certificates.

3.4.1 Description of EU-Spec. The validation process begins with a user generating a fresh signature key pair (pk_U, sk_U) and sending uid, pk_U along with a session identifier sid to $\mathcal{RP}$ (Step 1).

Then, $\mathcal{RP}$ forwards this sid, pk_U to the designated $\mathcal{VS}$ (Step 2), which informs $\mathcal{VS}$ to be contacted by the owner of pk_U. $\mathcal{RP}$ also generates a signed token γ containing the uid whose certificate it wants validated in this session, and sends it to the user (Step 3).

The user then encrypts her certificate, along with $uid, attr$ under $\mathcal{VS}$'s public key, signs the ciphertext using sk_U, and sends $\mathcal{RP}$'s token γ, ciphertext, and signature to $\mathcal{VS}$ (Step 4). The $\mathcal{VS}$ checks that the user signature verifies under the pk_U it received from $\mathcal{RP}$. If so, it decrypts the ciphertext, and verifies the enclosed certificate. It also checks that $\mathcal{RP}$'s token γ signs the same uid that is contained in the certificate. If all is correct, the $\mathcal{VS}$ signs the policy validation result along with sid and transmits it to $\mathcal{RP}$ (Step 5).

3.4.2 Comparison between EU-COV *and* EU-Spec. The main difference between the original specification EU-Spec and our analysed variant EU-COV is how the user binding is handled. We explain our modifications in two steps:

Change 1: Omitting γ & Step 3. The EU-Spec protocol assumes that the $\mathcal{RP}$ has a signature key pair. This key pair is used to indirectly deliver authenticated information to $\mathcal{VS}$ by generating the signed token γ and having the user $\mathcal{U}$ forward it (Step 3+4). At the same time, the protocol (in Step 2 and 5) requires $\mathcal{RP}$ and $\mathcal{VS}$ to communicate over a secure, mutually authenticated channel (e.g., mTLS). One can eliminate Message 3 in EU-Spec and instead let $\mathcal{RP}$ deliver the signed information directly to $\mathcal{VS}$ over their secure channel. This information is (uid, sid), which can be sent along with pk_U in Step 2. Omitting γ also removes the need for $\mathcal{RP}$'s key pair, without changing the security of the protocol.

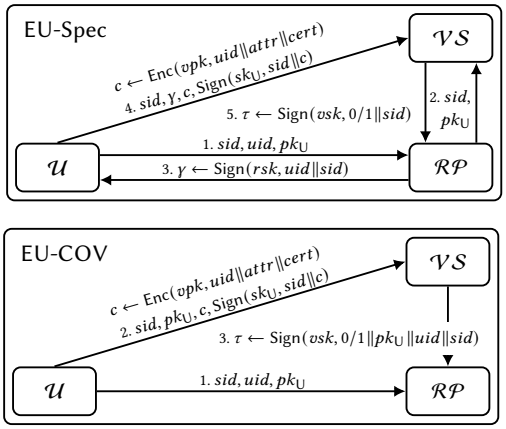

Figure 5: Comparison between the flow of messages in the original EU specification and our adaptation of it as EU-COV.

Change 2: Omitting Step 2. We now argue that sending uid, pk_U, sid from $\mathcal{RP}$ to $\mathcal{VS}$ can be omitted entirely, if one shifts the checks for user-binding from $\mathcal{VS}$ to $\mathcal{RP}$. Recall that a core security goal of EU-Spec is to guarantee that $\mathcal{RP}$ and $\mathcal{VS}$ are talking to the same user. EU-Spec achieves this by letting the $\mathcal{RP}$ tell the $\mathcal{VS}$ for which uid and pk_U it expects the policy check. The $\mathcal{VS}$ then only issues a validation token $\tau \leftarrow \text{Sign}(vsk, 1 \| sid)$ if 1) the uid in the verified $cert$ matches the one received from $\mathcal{RP}$, and 2) it receives a valid signature from the user that verifies under the pk_U it received from $\mathcal{RP}$.

We simplify this by letting the user send pk_U directly to $\mathcal{VS}$. The validator now checks the user's signature and her certificate for uid, and if both are valid, it signs uid, pk_U along with the policy result d as $\tau \leftarrow \text{Sign}(vsk, d \| pk_U \| uid \| sid)$. The $\mathcal{RP}$ verifies τ and that pk_U and uid are the same values it received from the user.

Thus, we maintain the binding of validation sessions to a particular uid, pk_U while transmitting less messages among the parties. In particular, our EU-COV interpretation does not require the $\mathcal{VS}$ to wait for matching inputs from $\mathcal{RP}$ and the user, but simply performs the validation after a single message from the user.

3.4.3 Security Results. Due to the additional interactions among $\mathcal{VS}$ and $\mathcal{RP}$ in the EU-Spec variant, neither our system nor security model can be applied to EU-Spec, but we argue why we believe that our shown security results for EU-COV apply (informally) to EU-Spec too: In both protocols, the $\mathcal{VS}$ learns the uid and full certificate of the user, and thus cannot satisfy any form of VS-Privacy. Regarding RP-Privacy, all the $\mathcal{RP}$ learns is an ephemeral public key pk_U, the uid, and a signature on the policy result (and uid, pk_U in EU-COV). Thus, in both protocols, we reveal the same amount of information about $attr, cert$ – namely none – to $\mathcal{RP}$.

When it comes to unforgeability, note that we only shift the matching of pk_U to uid from $\mathcal{VS}$ to $\mathcal{RP}$. Both are honest in the context of unforgeability, so the arguments from EU-COV apply to EU-Spec: unforgeability is not satisfied due to the lack of impersonation resistance (as $\mathcal{VS}$ learns the full certificate); but a weaker form of unforgeability where *all* validators are honest holds.

Optimizations. Our security analysis did not require any assumptions for the signature scheme S_U. Further, our model implicitly assumes that honest users and honest validators communicate over a secure channel, e.g., via TLS. Thus, the encryption scheme E does not provide any added security either. Therefore, both S_U and E can be omitted from EU-COV's construction (and consequently EU-Spec too) without impacting the security in our model. We discuss in Appendix B what security properties could have motivated S_U and E, and why they are not affected by the optimization.

4 OUR CONSTRUCTION: PP-COV

In this section we present our construction PP-COV, which provably achieves all security and privacy goals defined in Section 2.2. We first introduce the additional building blocks needed for our privacy-preserving scheme, and then describe the PP-COV construction, and prove its security. A comparison between PP-COV and EU-COV is given in Section 5.1.

4.1 Additional Building Blocks

In addition to a standard signature scheme S as defined in Section 3.1, PP-COV requires non-interactive zero-knowledge proofs, commitments and a signature scheme that enables privacy-friendly presentations.

Non-Interactive Zero-Knowledge Proofs of Knowledge. We use the notation $\pi \leftarrow \mathsf{NIZK}\{(w) : statement(w)\}(cxt)$ to describe a non-interactive zero-knowledge proof of knowledge (NIZK) of a witness w such that $statement(w) = 1$ and the proof π is bound to a context cxt. We require the proofs to be simulation-sound extractable and zero-knowledge. This enables us to simulate proofs that we give to adversaries in security experiments, and use an extractor algorithm $\mathcal{E}$ to extract witnesses from a new valid proof we get back.

Commitments. A commitment scheme Com = (Commit, OpenVf) for public parameters pp consists of a probabilistic function $(com, o) \leftarrow \mathsf{Commit}(m)$ that takes a message m and outputs a commitment com and opener o; and a verification function $0/1 \leftarrow \mathsf{OpenVf}(m, com, o)$ that outputs 1 if the given com opens to m using the opening o. We require Com to be hiding and binding.

Multi-Message Signature (MMS) with Efficient NIZK. A multi-message signature scheme MMS = (MMS.Setup, MMS.KGen, MMS.Sign, MMS.Verify) extends standard signatures by producing and verifying signatures for n-sized vectors of messages rather than just one. In our protocol we will handle exactly two message blocks per signature, and thus tailor the definition accordingly for simplicity. The algorithms are a setup $pp \leftarrow \mathsf{MMS.Setup}(1^\lambda)$ that outputs public parameters, and a key generation function $(pk, sk) \leftarrow \mathsf{MMS.KGen}(pp, n)$ where n denotes the number of signed messages blocks, which we set to $n := 2$. Accordingly, we have a signature algorithm $\sigma \leftarrow \mathsf{MMS.Sign}(sk, (m_1, m_2))$, and a verification function $\{0, 1\} \leftarrow \mathsf{MMS.Verify}(pk, (m_1, m_2), \sigma)$. We require MMS to satisfy MMS-EUF-CMA which translates the standard unforgeability game to the multi-message setting. For completeness, the definition is given in Appendix A.1.

Most importantly, we require the MMS to support efficient NIZKs of signatures over partially committed messages.

4.2 Detailed Protocol

The idea of our protocol is in its core similar to anonymous credential and privacy-enhancing authentication [7, 9], but makes an additional twist to ensure the required unforgeability property in this multi-party verification setting. In the following we give the high-level intuition of our protocol, and present the detailed PP-COV construction in Figure 6.

In PP-COV, the user's certificate on $(uid, attr)$ is a MMS signature (under the issuer's key), where both values are encoded as individual message blocks. To convince a validator that she has a policy-compliant certificate on $attr$ from ipk, she now merely proves possession of such a certificate via a NIZK, but does not send the original $cert$ anymore. This is crucial to achieve the desired impersonation resistance toward malicious $\mathcal{VS}$'s, which EU-COV was lacking. In the NIZK proof we further hide uid from the certificate, which yields the VS-Privacy.

This solution would not provide any user binding though. This is added by letting the user also compute a commitment com_{uid} on her uid which serves as session-specific pseudonym nym. The opening o_{uid} for that commitments becomes the authentication value π_{RP} for the $\mathcal{RP}$. In her NIZK proof π_{VS} towards $\mathcal{VS}$, the user then also proves that the commitment com_{uid} is for the uid that is also contained in $cert$. To ensure that the proof is bound to that particular session, all public and session-specific information are included as context to the NIZK. This also comprises the validator's public key, which ensures that a malicious validator can't relay the received proof to another honest $\mathcal{VS}$.

The $\mathcal{VS}$ then verifies the NIZK proof for $(attr, ipk, nym, sid)$ and that it is the intended recipient. If the proof is correct and $f(attr, ipk \in PK) = 1$, the $\mathcal{VS}$ signs (sid, nym) with a standard signature scheme as τ.

The $\mathcal{RP}$ receives (sid, nym, uid, o_{uid}) from the user, and (sid, τ) from the validator. It verifies that the signature τ under the $\mathcal{VS}$'s key signs the same (sid, nym) received from the user, and that the commitment $nym = com_{uid}$ opens to the expected uid using π_{RP}.

4.3 Security Analysis

We have informally argued how the different building blocks contribute to the desired security and privacy goals and now formally prove that PP-COV satisfies all properties defined in Section 2.2.

Theorem 4.1. *If the signature schemes* S *and* MMS *are respectively* EUF-CMA *and* MMS-EUF-CMA *secure,* NIZK *is simulation-sound extractable and zero knowledge, and the commitment scheme* Com *is binding, then* PP-COV *is unforgeable w.r.t. Def. 2.1.*

Proof (Sketch). Assume $\mathcal{A}$ wins $Exp^{\mathsf{UF}}_{\mathcal{A},\mathsf{COV}}$ against PP-COV, thus outputs $(uid^*, sid^*, nym^*, \pi^*_{\mathsf{RP}}, \tau^*)$ such that $\mathsf{ValTokVerify}(vpk, uid^*, sid^*, nym^*, \pi^*_{\mathsf{RP}}, \tau^*) = 1$ and at least one of the conditions given in Case 1-4 is fulfilled. Verification in our scheme only returns 1 when τ^* is a valid (standard) signature under vpk on (sid^*, nym^*) and nym^* correctly opens to uid^*.

Similar to EU-COV, the proof branches into two sub-conditions:

(1) (sid^*, nym^*) has not been signed by $\mathcal{VS}$ (which yields a direct forgery for S),
(2) (sid^*, nym^*) has been signed by $\mathcal{VS}$ but the rest of $\mathcal{A}$'s output is not consistent with the state of the game.

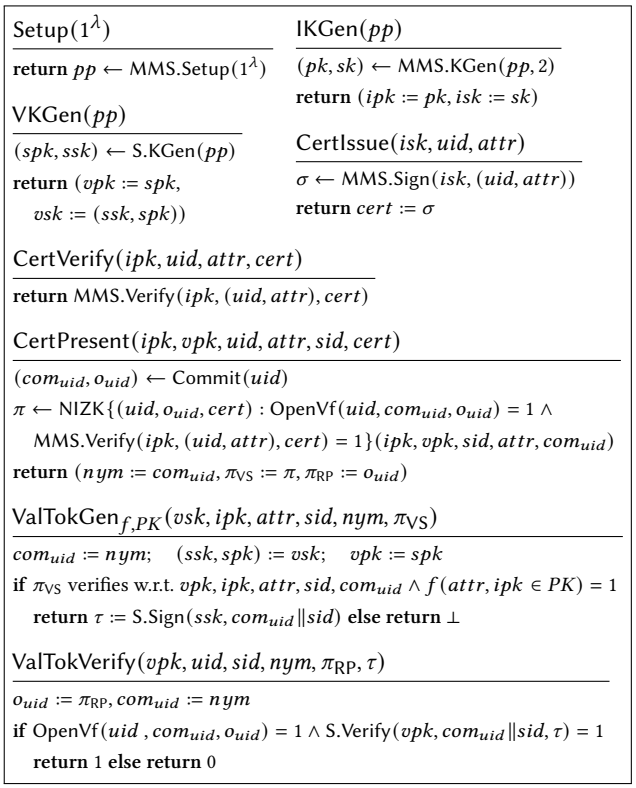

Figure 6: Generic Construction for PP-COV.

For (2), the proof then further branches for each of Case 1-4. In summary we show that $\mathcal{A}$ either managed to re-use a validation token for a different user by opening nym^* to an incorrect uid^* (which isn't possible due to the binding property of Com), or the $\mathcal{VS}$ must have received some forged input. This must have either been through a forged proof π_{VS} (which isn't possible due to the soundness of NIZK), or a correct proof for a forged certificate (which isn't possible due to the unforgeability of the MMS). For proving the latter, we rely on the (simulation-)soundness of the NIZK to extract $cert$ from π_{VS} and use this as forgery for the MMS scheme.

While most of the reductions are rather straight-forward, handling Case 3 (unforgeability for honest users) requires special care: To prove the desired impersonation resistance (which is part of Case 3), and be able to extract a valid forgery from π_{VS}, we no longer create certificates for honest users in O_{HU}, but only keep state of which honest user has which attributes. When the adversary requests an honest user to create a presentation for a corrupt $\mathcal{VS}$ via O_π, we then simulate π_{VS} (which is why we need the zero-knowledge property here too). If $\mathcal{A}$ manages to come up with a valid proof π_{VS} for an honest user that he presents to the honest $\mathcal{VS}$ via O_τ, we extract a valid MMS forgery from there. This step relies on the simulation-soundness of the NIZK, and on the fact that we bind each proof to its specific context, i.e., the adversary can't replay any of our simulated proofs. As we never issued any certificates for honest users, we know that the extracted MMS signature is a valid and fresh forgery in the reduction.

The full proof is given in Appendix C.2. □

Theorem 4.2. *If* NIZK *is zero-knowledge and* Com *is hiding, then* PP-COV *satisfies VS-Privacy according to Def. 2.2.*

Proof. Assume an adversary wins, that is, it tells with non-negligible probability which $uid_b, cert_b$ were used in creating the NIZK π_{VS} and the uid_b commitment nym, for uniform $b \in \{0, 1\}$.

Because the values $ipk, vpk, attr, sid$ are the same regardless of b, the adversary cannot use them for winning. Thus, either π_{VS} or nym must be leaking information about uid_b or $cert_b$. But as we assume nym to be hiding, and π_{VS} to be zero-knowledge, this can only happen with negligible probability. □

Theorem 4.3. PP-COV *satisfies RP-Privacy according to Def. 2.3.*

Proof. In $Exp^{\text{PRIV-}\mathcal{RP}}_{\mathcal{A},\text{COV}}$, the adversary chooses some uid, sid, and a pair $(ipk_i, attr_i, cert_i)_{i\in 0,1}$. Then it receives (π_{RP}, nym, τ) derived for $(ipk_b, attr_b, cert_b)$ and wins if it correctly guesses the random bit b. Since nym, π_{RP} are a commitment and opening over the uid which is independent of b or the different challenge sets, they cannot contain any information that helps $\mathcal{A}$ in determining b. The game also requires $sid, f(attr_b, ipk_b \in PK)$ to be equivalent for $b \in \{0, 1\}$, so τ will also be a standard signature independent of the challenge values. Overall, the adversary does not receive any information that depends on b, and thus can only guess the bit. □

5 DISCUSSION & COMPARISON

In this section we compare the efficiency of PP-COV to EU-COV and discuss the information-leakage through attributes (and how that can be contained), and what can (not) be achieved by constructions relying on standard signatures only.

5.1 Comparison between PP-COV & EU-COV

To compare both protocols, we first need to choose instantiations for all generic building blocks they are composed of.

Instantiations for EU-COV. For EU-COV, we rely on the schemes proposed by the actual specification [15], and the concrete instantiations used in their implementation [16]. Both suggest to use ECDSA (with SHA-256) for all signature schemes S_{VS}, S_U, S_I, and the encryption is realized via hybrid encryption using RSA-OAEP[1] (with at least 3072 bits) and AES-256-GCM or AES-256-CBC.

Instantiations for PP-COV. For the standard signature scheme S used by the $\mathcal{VS}$ in our PP-COV construction, we rely on the same ECDSA signatures as EU-COV for better comparability. The MMS scheme is instantiated with PS signatures [36] which has the shortest signature sizes, and works in a pairing group $(\mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e, p) \leftarrow \text{PGen}(1^\lambda)$. Pointcheval and Sanders already showed that their scheme allows for efficient randomization and proofs of knowledge via generalized Schnorr-signature proofs of knowledge [4] which can be made non-interactive via the Fiat-Shamir heuristic [17], which we use too. For the commitment scheme we rely on Pedersen

[1] The specification [15] actually suggests to protect the AES key either with RSA-OAEP or ECDSA; which we hope to be a typo. However, even if someone implements it by signing the secret AES key instead of encrypting it, this wouldn't create much harm, as we prove the scheme to be secure regardless of any security of the encryption scheme.

	CertPresent	ValTokGen$_{f,PK}$	ValTokVerify	Unforgeability	RP-Privacy	VS-Privacy
PP-COV	$3\mathbb{G} + 5\tilde{\mathbb{G}} + 2\mathbb{G}_T + 4P$	$4\tilde{\mathbb{G}} + 2\mathbb{G}_T + 4P + 1\bar{\mathbb{G}}$	$2\bar{\mathbb{G}} + 2\tilde{\mathbb{G}}$	✔	✔	✔
EU-COV	$1\mathbb{Z}_n + 2\bar{\mathbb{G}}$	$5\bar{\mathbb{G}} + 1\mathbb{Z}_n$	$2\bar{\mathbb{G}}$	Weak Unforgeability	✔	-

Figure 7: Number of modular exponentiations and satisfied properties in PP-COV and EU-COV. In PP-COV, $\mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T$ are groups of a pairing-friendly curve. EU-COV uses the RSA group $\mathbb{Z}_n$. Both constructions use an ECDSA group $\bar{\mathbb{G}}$.

commitments [35], that we use in $\tilde{\mathbb{G}}$ for two generators $\hat{g}, \hat{h}$, chosen at random during the Setup of our scheme, and which become part of our public parameters. For completeness, we present the instantiation of PP-COV with the aforementioned building blocks in Appendix D.

Efficiency Comparison. In Figure 7, we compare the instantiations in terms of the number of modular exponentiations and pairing operations involved— these bear the highest cost in the algorithms. We particularly focus on the algorithms that will be executed frequently: CertPresent, ValTokGen$_{f,PK}$, and ValTokVerify.

To create a presentation token, the user computes 10 exponentiations and 4 pairings in PP-COV, which is more than the 3 exponentiations required by EU-COV (but some in RSA groups). By running benchmarks from the open source cryptographic library mcl [2], we found that the exponentiations in $\mathbb{G}$, $\tilde{\mathbb{G}}$, $\mathbb{G}_T$, and $\bar{\mathbb{G}}$ require 0.4ms, 0.9ms, 1.7ms, and 0.04ms respectively, while pairing costs 1.5ms. Accordingly, for PP-COV we estimate execution times of 15ms, 13ms, 1.9ms for CertPresent, ValTokGen$_{f,PK}$, ValTokVerify which is clearly efficient enough for real-world deployment.

5.2 Privacy Considerations

Our security model for VS-Privacy guarantees that the user's presentation does not reveal anything beyond the user's attributes to the validator, and in particular does not reveal anything (beyond *attr*) that allows to identify *uid* or link different presentations. This means that one needs to ensure that the attribute values themselves are not already unique and identifiable enough (as this is outside the security model). The more attributes within one certificate and the more detailed they are (e.g., very precise issuance/expiration dates), the more identifiable their presentation will be.

Minimizing Attribute Leakage. One can already steer the information leakage in the certificate setup, making sure to issue dedicated certificates for each attribute instead of one *cert* for many (e.g., one certificate per vaccination, instead of one cert containing all), and avoid fine-grained time information as attributes (which often is not needed, usually month/year is sufficient). Another approach is to reduce the amount of information the user has to show during its presentation. Currently, we assume that the user reveals all attributes in the certificate to $\mathcal{VS}$, which can easily be minimized by encoding every attribute in *cert* individually and only reveal the ones required by the policy via an adapted NIZK. In fact, this also demonstrates another advantage of our PP-COV solution – one reason for the lack of adoption of anonymous credentials is the increased complexity in verifying these credentials. Our validation system allows to securely outsource this verification.

[2]https://github.com/herumi/mcl using AMD Ryzen 5 2.9GHz processor.

Unlinkability and Revocation. Assuming attributes to *not* be unique per user does not match reality, if one considers classic revocation. Therein, each certificate contains a globally unique *cid* that is checked against a black-list of revoked certificates. We do not believe that classic revocation is the best approach for such (low-value) user certificates; instead short-lived certificates that get renewed regularly is a more suitable solution. In fact, also the EU's validation service currently does not consider or support revocation of user-specific certificates [15]. Revocation is more likely to happen on the level of issuer keys (with each country maintaining a hierarchy of issuer keys), which is also better for privacy.

However, *if* user-specific revocation should be supported, one needs to deploy privacy-preserving revocation such as [1, 5, 30] in order to still achieve our strong VS-Privacy notion. If standard revocation is used, i.e., every presentation reveals the unique *cid* to the $\mathcal{VS}$, one needs to adjust and weaken the security model to capture a meaningful guarantee *despite* the linkability imposed by revealing a user-specific *cid* in every presentation.

Weaker VS-Privacy for Classic Revocation. We sketch the weaker VS-Privacy notion that incorporates the inherent privacy loss via classic revocation (and that would be achieved by PP-COV with classic revocation, but not EU-COV). First, one needs to assume the issuer to be honest, as he knows which user is associated with which *cid*. In the VS-Privacy game, this would be reflected by giving $\mathcal{A}$ oracle access to the honest users and their certificates (instead of letting $\mathcal{A}$ compute the *cert*s). This actually complicates the model significantly, as we'd now have to offer several oracles to steer the honest users to engage in sessions with the corrupt $\mathcal{VS}$ as well as with corrupt $\mathcal{RP}$s (this was not necessary in our definition, as therein $\mathcal{A}$ knows the certificates and can run all algorithms himself). Second, one no longer requires *attr* to be the same for uid_0 and uid_1 but allow different values, as long as they are both policy compliant. This weaker model would no longer capture unlinkability of π_{VS}, but still ensure that a corrupt $\mathcal{VS}$ learns nothing beyond that attribute linkage – in particular it cannot use information learned in sessions where it received presentations in the role of a corrupt $\mathcal{RP}$, to identify the user.

Privacy with Standard Certificates? If one considers such a weaker notion of VS-Privacy that no longer requires unlinkability of π_{VS}, this raises the question of whether a construction solely relying on standard certificates could satisfy it. A clear disadvantage of PP-COV is that it is not compatible with classic certificates based on ECDSA and RSA-signatures like EU-COV is. However, we believe that no meaningful privacy can be achieved by standard signatures, and sketch a solution that can be constructed from standard signatures S and explain why it does not satisfy the weaker VS-Privacy above.

Certificates: The issuer uses a standard signature S and creates certificates $cert' \leftarrow \mathsf{Sign}(sk, (\mathsf{H}(r, uid) \| attr))$ where $r \leftarrow \{0,1\}^\lambda$ is chosen at random and becomes part of the certificate sent to the user, i.e., $cert \leftarrow (cert', r)$.

Presentation to $\mathcal{VS}$: When a user wants to engage in a validation session, it sets $nym \leftarrow \mathsf{H}(r, uid)$, $\pi_{\mathsf{VS}} \leftarrow cert'$ and $\pi_{\mathsf{RP}} \leftarrow r$. The $\mathcal{VS}$ verifies whether $cert'$ is a valid signature on $(nym \| attr)$ and if so, signs (sid, nym) as τ.

Verification by $\mathcal{RP}$: $\mathcal{RP}$ receives $sid, uid, nym, \pi_{\mathsf{RP}} (= r)$ from the user, receives sid, τ from $\mathcal{VS}$, and verifies that τ correctly signs (sid, nym) and $nym = \mathsf{H}(r, uid)$.

On the first glance, this might appear as a privacy-preserving solution, as the $\mathcal{VS}$ only learns the hash value $nym = \mathsf{H}(r, uid)$ which is known to be a secure commitment if r is random. However, this pseudonym is static in every usage of the certificate, which in turn allows linkage of all sessions that a user ever makes. As the user is never anonymous towards $\mathcal{RP}$, this allows the following trivial attack even when the issuer is honest: the user runs a single validation session (possibly via an honest $\mathcal{VS}_{\mathsf{hon}}$) towards a corrupt $\mathcal{RP}_{\mathsf{cor}}$, i.e., the corrupt $\mathcal{RP}_{\mathsf{cor}}$ learns the (static) nym that belongs to user uid. If the user runs another validation session via a corrupt $\mathcal{VS}_{\mathsf{cor}}$ towards an honest $\mathcal{RP}_{\mathsf{hon}}$, she sends the same nym. The adversary can then re-use its knowledge from the other session to derive the user's identity. This protocol sketched above would only satisfy a very weak form of user hiding, if (in addition to the issuer) *all* relying parties are honest, which clearly is not realistic.

Overall, we strongly believe that privacy-preserving certificates are necessary for any meaningful privacy in such system. And while they are not compatible with the existing infrastructure of health certificates at the moment, this should not be taken as an excuse to discard privacy-friendly solutions, but rather as (yet another) reason to move user-centric certificates to solutions that have built-in privacy as a core requirement and feature.

ACKNOWLEDGMENTS

This research was partially funded by the HPI Research School on Data Science and Engineering. It was also supported by the German Federal Ministry of Education and Research (BMBF) through funding of the ATLAS project under reference number 16KISA037.

REFERENCES

[1] Foteini Baldimtsi, Jan Camenisch, Maria Dubovitskaya, Anna Lysyanskaya, Leonid Reyzin, Kai Samelin, and Sophia Yakoubov. 2017. Accumulators with Applications to Anonymity-Preserving Revocation. In *2017 IEEE European Symposium on Security and Privacy*. IEEE Computer Society Press, Paris, France, 301–315. https://doi.org/10.1109/EuroSP.2017.13

[2] Stefan A. Brands. 2000. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA. https://doi.org/10.7551/mitpress/5931.001.0001

[3] Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. 2015. Composable and Modular Anonymous Credentials: Definitions and Practical Constructions. In *ASIACRYPT 2015, Part II (LNCS, Vol. 9453)*, Tetsu Iwata and Jung Hee Cheon (Eds.). Springer, Heidelberg, Germany, Auckland, New Zealand, 262–288. https://doi.org/10.1007/978-3-662-48800-3_11

[4] Jan Camenisch, Aggelos Kiayias, and Moti Yung. 2009. On the Portability of Generalized Schnorr Proofs. In *EUROCRYPT 2009 (LNCS, Vol. 5479)*, Antoine Joux (Ed.). Springer, Heidelberg, Germany, Cologne, Germany, 425–442. https://doi.org/10.1007/978-3-642-01001-9_25

[5] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. 2009. An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In *PKC 2009 (LNCS, Vol. 5443)*, Stanislaw Jarecki and Gene Tsudik (Eds.). Springer, Heidelberg, Germany, Irvine, CA, USA, 481–500. https://doi.org/10.1007/978-3-642-00468-1_27

[6] Jan Camenisch, Stephan Krenn, Anja Lehmann, Gert Læssøe Mikkelsen, Gregory Neven, and Michael Østergaard Pedersen. 2016. Formal Treatment of Privacy-Enhancing Credential Systems. In *SAC 2015 (LNCS, Vol. 9566)*, Orr Dunkelman and Liam Keliher (Eds.). Springer, Heidelberg, Germany, Sackville, NB, Canada, 3–24. https://doi.org/10.1007/978-3-319-31301-6_1

[7] Jan Camenisch and Anna Lysyanskaya. 2001. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *EUROCRYPT 2001 (LNCS, Vol. 2045)*, Birgit Pfitzmann (Ed.). Springer, Heidelberg, Germany, Innsbruck, Austria, 93–118. https://doi.org/10.1007/3-540-44987-6_7

[8] Jan Camenisch and Anna Lysyanskaya. 2004. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *CRYPTO 2004 (LNCS, Vol. 3152)*, Matthew Franklin (Ed.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 56–72. https://doi.org/10.1007/978-3-540-28628-8_4

[9] David Chaum. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the Association for Computing Machinery* 24, 2 (1981), 84–90. https://doi.org/10.1145/358549.358563

[10] David Chaum. 1985. Security without identification: Transaction systems to make big brother obsolete. *Communications of the Association for Computing Machinery* 28, 10 (1985), 1030–1044. https://doi.org/10.1145/4372.4373

[11] Yu Chen, Jiguo Li, Chengdong Liu, Jinguang Han, Yichen Zhang, and Peng Yi. 2022. Efficient Attribute Based Server-Aided Verification Signature. *IEEE Transactions on Services Computing* 15, 6 (2022), 3224–3232. https://doi.org/10.1109/TSC.2021.3096420

[12] The European Commission. 2021. Commission Implementing Decision (EU) 2021/1073 of 28 June 2021 laying down technical specifications and rules for the implementation of the trust framework for the EU Digital COVID Certificate established by Regulation (EU) 2021/953 of the European Parliament and of the Council. https://eur-lex.europa.eu/eli/dec_impl/2021/1073/oj

[13] The European Commission. 2022. EU Digital COVID Certificate: EU launches new revocation feature. https://digital-strategy.ec.europa.eu/en/news/eu-digital-covid-certificate-eu-launches-new-revocation-feature.

[14] The European Commission. 2022. Regulation (EU) 2021/953 of the European Parliament and of the Council of 14 June 2021 on a framework for the issuance, verification and acceptance of interoperable COVID-19 vaccination, test and recovery certificates (EU Digital COVID Certificate) to facilitate free movement during the COVID-19 pandemic. http://data.europa.eu/eli/reg/2021/953/2022-03-31/eng

[15] eHealth Network. 2021. Guidlines on the use of Digital Covid Certificates in traveller and online booking scenarios. https://ec.europa.eu/health/sites/default/files/ehealth/docs/covid-certificate_traveller-onlinebooking_en.pdf.

[16] eHealth Network. 2022. EU Digital COVID Certificate Validation Service. https://github.com/eu-digital-green-certificates/dgca-validation-service/tree/2f047cd.

[17] Amos Fiat and Adi Shamir. 1987. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO'86 (LNCS, Vol. 263)*, Andrew M. Odlyzko (Ed.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 186–194. https://doi.org/10.1007/3-540-47721-7_12

[18] Centers for Disease Control and Prevention. 2021. Requirement for Proof of COVID-19 Vaccination for Air Passengers. https://web.archive.org/web/20211101003745/https://www.cdc.gov/coronavirus/2019-ncov/travelers/proof-of-vaccination.html.

[19] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. 2019. Structure-Preserving Signatures on Equivalence Classes and Constant-Size Anonymous Credentials. *Journal of Cryptology* 32, 2 (April 2019), 498–546. https://doi.org/10.1007/s00145-018-9281-4

[20] Marc Girault and David Lefranc. 2005. Server-Aided Verification: Theory and Practice. In *ASIACRYPT 2005 (LNCS, Vol. 3788)*, Bimal K. Roy (Ed.). Springer, Heidelberg, Germany, Chennai, India, 605–623. https://doi.org/10.1007/11593447_33

[21] Ulrich Haböck and Stephan Krenn. 2019. Breaking and Fixing Anonymous Credentials for the Cloud. In *CANS 19 (LNCS, Vol. 11829)*, Yi Mu, Robert H. Deng, and Xinyi Huang (Eds.). Springer, Heidelberg, Germany, Fuzhou, China, 249–269. https://doi.org/10.1007/978-3-030-31578-8_14

[22] Mark A. Hall and David M. Studdert. 2021. "Vaccine Passport" Certification - Policy and Ethical Considerations. *New England Journal of Medicine* 385, 11 (2021), e32. https://doi.org/10.1056/NEJMp2104289

[23] Javier Herranz, Fabien Laguillaumie, Benoît Libert, and Carla Ràfols. 2012. Short Attribute-Based Signatures for Threshold Predicates. In *CT-RSA 2012 (LNCS, Vol. 7178)*, Orr Dunkelman (Ed.). Springer, Heidelberg, Germany, San Francisco, CA, USA, 51–67. https://doi.org/10.1007/978-3-642-27954-6_4

[24] Susan Hohenberger and Anna Lysyanskaya. 2005. How to Securely Outsource Cryptographic Computations. In *TCC 2005 (LNCS, Vol. 3378)*, Joe Kilian (Ed.). Springer, Heidelberg, Germany, Cambridge, MA, USA, 264–282. https://doi.org/10.1007/978-3-540-30576-7_15

[25] Georgios Karopoulos, Jose L. Hernandez-Ramos, Vasileios Kouliaridis, and Georgios Kambourakis. 2021. A Survey on Digital Certificates Approaches for the COVID-19 Pandemic. *IEEE Access* 9 (2021), 138003–138025. https://doi.org/10.1109/ACCESS.2021.3117781

[26] Stephan Krenn, Thomas Lorünser, Anja Salzer, and Christoph Striecks. 2017. Towards Attribute-Based Credentials in the Cloud. In *CANS 17 (LNCS, Vol. 11261)*, Srdjan Capkun and Sherman S. M. Chow (Eds.). Springer, Heidelberg, Germany, Hong Kong, China, 179–202. https://doi.org/10.1007/978-3-030-02641-7_9

[27] Jin Li, Man Ho Au, Willy Susilo, Dongqing Xie, and Kui Ren. 2010. Attribute-Based Signature and Its Applications. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security* (Beijing, China) *(ASIACCS '10)*. Association for Computing Machinery, New York, NY, USA, 60–69. https://doi.org/10.1145/1755688.1755697

[28] Chae Hoon Lim and Pil Joong Lee. 1995. Server (Prover/Signer)-Aided Verification of Identity Proofs and Signatures. In *EUROCRYPT'95 (LNCS, Vol. 921)*, Louis C. Guillou and Jean-Jacques Quisquater (Eds.). Springer, Heidelberg, Germany, Saint-Malo, France, 64–78. https://doi.org/10.1007/3-540-49264-X_6

[29] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. 2011. Attribute-Based Signatures. In *CT-RSA 2011 (LNCS, Vol. 6558)*, Aggelos Kiayias (Ed.). Springer, Heidelberg, Germany, San Francisco, CA, USA, 376–392. https://doi.org/10.1007/978-3-642-19074-2_24

[30] Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki. 2009. Revocable Group Signature Schemes with Constant Costs for Signing and Verifying. In *PKC 2009 (LNCS, Vol. 5443)*, Stanislaw Jarecki and Gene Tsudik (Eds.). Springer, Heidelberg, Germany, Irvine, CA, USA, 463–480. https://doi.org/10.1007/978-3-642-00468-1_26

[31] NIST. 2013. Digital Signature Standard (DSS). National Institute of Standards and Technology, NIST FIPS PUB 186-5. https://doi.org/10.6028/NIST.FIPS.186-5

[32] Ministry of Foreign Affairs of Japan. 2023. Border measures to prevent the spread of novel coronavirus (COVID-19). https://www.mofa.go.jp/ca/fna/page4e_001053.html.

[33] Israel Ministry of Health. 2021. What is a Green Pass. https://web.archive.org/web/20210213060744/https://corona.health.gov.il/en/directives/green-pass-info/.

[34] Christian Paquin and Greg Zaverucha. 2013. U-Prove Cryptographic Specification V1.1 (Revision 3). https://www.microsoft.com/en-us/research/publication/u-prove-cryptographic-specification-v1-1-revision-3/

[35] Torben P. Pedersen. 1992. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *CRYPTO'91 (LNCS, Vol. 576)*, Joan Feigenbaum (Ed.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 129–140. https://doi.org/10.1007/3-540-46766-1_9

[36] David Pointcheval and Olivier Sanders. 2016. Short Randomizable Signatures. In *CT-RSA 2016 (LNCS, Vol. 9610)*, Kazue Sako (Ed.). Springer, Heidelberg, Germany, San Francisco, CA, USA, 111–126. https://doi.org/10.1007/978-3-319-29485-8_7

[37] The Washington Post. 2020. Chile's 'immunity passport' will allow recovered coronavirus patients to break free from lockdown, get back to work. https://www.washingtonpost.com/world/the_americas/chile-coronavirus-immunity-passport-antibody-testing-card/2020/04/20/8daef326-826d-11ea-81a3-9690c9881111_story.html.

[38] Abhishek Singh, Ramesh Raskar, and Anna Lysyanskaya. 2021. Safepaths: Vaccine Diary Protocol and Decentralized Vaccine Coordination System using a Privacy Preserving User Centric Experience. *CoRR* abs/2103.01754 (2021), 13 pages. https://doi.org/10.48550/arXiv.2103.01754

[39] T-Systems. 2021. Corona Validation Service. https://www.t-systems.com/de/en/newsroom/news/corona-validation-service-475486.

[40] T-Systems. 2022. World Health Organization selects T-Systems as industry partner. https://www.telekom.com/en/media/media-information/archive/covid-19-who-commissions-t-systems-648634.

[41] Zhiwei Wang, Ruirui Xie, and Shaohui Wang. 2014. Attribute-based Server-Aided Verification Signature. *Applied Mathematics & Information Sciences* 8 (11 2014), 3183–3190. https://doi.org/10.12785/amis/080660

A DEFINITIONS

In this section we include the formal definition of MMS-EUF-CMA that we used in our construction, as well as of correctness of the COV system.

A.1 MMS-EUF-CMA

It is infeasible for $\mathcal{A}$ in the game $Exp^{\text{MMS-EUF-CMA}}_{\mathcal{A},\text{MMS}}$ to find a valid signature σ w.r.t. pk for a message vector with size n that was never queried to O_{MMS}. Here we are only interested in $n = 2$, so without loss of generality, we adjust the definition to operate on exactly two messages m_1 and m_2 directly.

Definition A.1 (MMS-EUF-CMA). MMS is unforgeable if for all PPT adversaries $\mathcal{A}$, $\Pr[Exp^{\text{MMS-EUF-CMA}}_{\mathcal{A},\text{MMS}}(\lambda) = 1] \leq \text{negl}(\lambda)$.

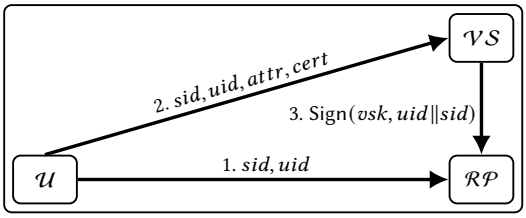

Figure 8: Optimized EU-COV, omitting the unnecessary user signature and certificate encryption.

Experiment $Exp^{\text{MMS-EUF-CMA}}_{\mathcal{A},\text{MMS}}(\lambda)$

Define $O_{\text{MMS}}(m_1, m_2)$
$Q_{\text{MMS}} := Q_{\text{MMS}} \cup (m_1, m_2)$
return $\text{MMS.Sign}(sk, (m_1, m_2))$

Initialize Q_{MMS} as empty array.
$pp \leftarrow \text{MMS.Setup}(1^\lambda)$
$(pk, sk) \leftarrow \text{MMS.KGen}(pp, n := 2)$
$(\sigma, m_1, m_2) \leftarrow \mathcal{A}^{O_{\text{MMS}}}(pk)$
return $(m_1, m_2) \notin Q_{\text{MMS}} \wedge$
$\text{MMS.Verify}(pk, (m_1, m_2), \sigma) = 1$

A.2 COV Correctness

A COV system is correct if for all $\lambda \in \mathbb{N}$, all $pp \leftarrow \text{Setup}(1^\lambda)$, all issuer keys $(ipk, isk) \leftarrow \text{IKGen}(pp)$, all $(uid, attr) \in \mathcal{D}^2$, and all $cert \leftarrow \text{CertIssue}(isk, uid, attr)$, it holds that

$$\text{CertVerify}(ipk, uid, attr, cert) = 1$$

and for all $sid \in \mathbb{Z}$, all validator keys $(vpk, vsk) \leftarrow \text{VKGen}(pp)$ that implicitly define policy f and trusted issuers PK, all presentations $(\pi_{\text{VS}}, \pi_{\text{RP}}, nym) \leftarrow \text{CertPresent}(ipk, vpk, uid, attr, sid, cert)$, and all validations $\tau \leftarrow \text{ValTokGen}_{f,PK}(vsk, ipk, attr, sid, nym, \pi_{\text{VS}})$, the following holds:

$$\text{ValTokVerify}(vpk, uid, sid, nym, \pi_{\text{RP}}, \tau) = f(attr, ipk \in PK)$$

B EU-COV OPTIMIZATIONS

Our security analysis did not require any assumptions on the encryption scheme E and the user's signature scheme S_U, and thus can simply be omitted from EU-COV's construction (and consequently for EU-Spec too) without impacting the security in our model. The optimized version of EU-COV is depicted in Figure 8, and can be seen as a version that implicitly uses $nym = \pi_{\text{RP}} = \perp$ everywhere.

In the following we briefly discuss what security properties (possibly not captured by our model) could have been the motivation for S_U and E, and why they are not affected by the optimization.

Omitting Signature Scheme S_U. The EU-Spec motivates the user's signature key pair with two properties: user-binding and data privacy. We now argue that it doesn't contribute to either.

No impact on user-binding: One reason for S_U could be to ensure that the same user talks to $\mathcal{VS}$ and $\mathcal{RP}$: the user sends a signature under pk_U to $\mathcal{VS}$, and sends pk_U to the $\mathcal{RP}$, who forwards the key to $\mathcal{VS}$. The validator then checks that they match, i.e., it verifies the signature it receives from the user, under the key it gets from the $\mathcal{RP}$. This is unnecessary though, as the $\mathcal{VS}$ already verifies that the expected uid from $\mathcal{RP}$ matches the uid in

cert (or signs the verified *uid* and shifts the check whether this is the expected user to $\mathcal{RP}$, which is what we do in EU-COV).

What we do not know though, is whether the user showing the certificate and getting a token for *uid* is indeed that user. The underlying problem is that users don't own any long-term secret keys, i.e., there is nothing that defines who a user is, except the certificate *cert* containing the user's unique *uid*. Now anyone having the certificate can pose as *uid*. This is unavoidable and cannot be fixed by sending an unrelated ephemeral public key and signature with the certificate.

For the targeted application this is not a problem though, as the final assertion is still bound to the *uid* that had a proper certificate and passed the check by the $\mathcal{VS}$. This already prevents the main threat, namely that a user *not* owning the necessary certificate can convince $\mathcal{RP}$ of the opposite.

No impact on data-privacy: The EU-Spec argues that the "[user] public key must be unique for each subject for maximum data privacy" [15]. As the key relates to a signature and not an encryption scheme, there is no data it can protect other than the user's identity. Thus, we expect that the unique public key is meant as some form of a pseudonym. While having pseudonymous authentication is an admirable goal, it is impossible to achieve here as the user sends her unique identity *uid* to both the $\mathcal{VS}$ and $\mathcal{RP}$.

Overall, the $\mathsf{S_U}$ scheme neither provides any additional guarantees regarding user binding, nor is there any hope for user privacy (which is what we cover as VS-Privacy) in the scheme altogether, and thus $\mathsf{S_U}$ can be fully omitted from the protocol.

Omitting Encryption Scheme E. In a validation session, the user encrypts her certificate with $\mathcal{VS}$'s public key before transmitting it, however, we did not require any guarantees from E. This is because our model implicitly assumes that honest users and honest validators communicate over a secure channel, which aligns with EU-Spec's assumption of a TLS connection between those parties. Hence, the additional encryption via E does not contribute to privacy in our model and can be omitted.

In a broader perspective though, not only must the communication channel itself be secure, but the whole environment (e.g., smartphone software) that handles the user's certificate, until it finally gets transmitted. An additional encryption performed by the user would provide extra protection that alleviates the amount of *effective trust* required in practice.

Validation Oracle. In the EU-Spec (and EU-COV) construction, $\mathsf{ValTokGen}_{f,PK}$ distinguishes between invalid inputs (e.g., unverifiable signatures) and policy check failures. For the former it outputs $\perp$, for the latter it creates a validation token τ, but with a signed 0 as the verification result. This has a side effect of creating an oracle that leaks unwanted information (e.g., signature checks passed, policy validation failed). It also requires careful handling of the different possible outcomes when realizing the function in practice. Thus we suggest to generate τ only if both the certificate and the policy validation succeeds.

C FULL PROOFS

In this section we give the full proofs for EU-COV satisfying Weak Unforgeability, and PP-COV satisfying Unforgeability.

C.1 EU-COV Weak Unforgeability

Recall that this is the weaker notion of Unforgeability where the adversary $\mathcal{A}$ in the game $Exp^{\mathsf{weakUF}}_{\mathcal{A},\mathsf{COV}}$ is not provided access to O_π. $\mathcal{A}$ outputs uid^*, sid^*, nym^*, π^*_{RP}, τ^* and wins if $\mathsf{ValTokVerify}(vpk, uid^*, sid^*, nym^*, \pi^*_{\mathsf{RP}}, \tau^*) = 1$ and one of Case 1-4 annotated in the game holds. For each of the cases, we use $\mathcal{A}$ to construct efficient adversaries $\mathcal{B}$ and C against EUF-CMA of $\mathsf{S_{VS}}$ and $\mathsf{S_I}$ respectively.

Proof. The proof branches into addressing indirect and direct forgeries, depending on whether the honest $\mathcal{VS}$ has seen a query for (sid^*, nym^*) or not. If $(sid^*, nym^*) \notin Q_\tau \cup Q_{\pi,\tau}$ we have an immediate forgery against $\mathsf{S_{VS}}$ (direct forgery), and in Figure 9, we use $\mathcal{A}$ to construct the adversary $\mathcal{B}$ against the EUF-CMA game. $\mathcal{B}$ is initialized with a public key from the scheme $\mathsf{S_{VS}}$, which becomes the honest $\mathcal{VS}$'s public key, and we modify $O_\tau, O_{\pi,\tau}$ to forward requests to O_{SIG} if the given certificate verification succeeds. We do not need to alter the other oracles $O_{\mathsf{CU}}, O_{\mathsf{HU}}$ as their operation is independent of what $O_{\pi,\tau}$ and O_τ do. With the oracles perfectly simulated, $\mathcal{B}$ runs $\mathcal{A}$ and outputs $(\sigma := \tau^*, m := 1 \| nym^* \| uid^* \| sid^*$ as a forgery in the EUF-CMA game.

The rest of the proof is now for the case when $(sid^*, nym^*) \in Q_\tau \cup Q_{\pi,\tau}$. For this, we will use $\mathcal{A}$ to construct the other adversary C against EUF-CMA of $\mathsf{S_I}$, shown in Figure 9. C is initialized with a public key from the scheme $\mathsf{S_I}$, which it inserts as an issuer's public key into a random index in the list of public keys of honest issuers given to $\mathcal{A}$. We modify the oracles $O_{\mathsf{CU}}, O_{\mathsf{HU}}$ to forward queries to O_{SIG} if the input ipk_i equals pk, and the oracle O_τ to store π_{VS} from the queries it receives. We do not require similar modification to $O_{\pi,\tau}$ as we will show that the adversary's forgeries could not have originated from it. Further, as our changes perfectly simulate the original oracles, the unchanged $O_{\pi,\tau}$ – which accesses O_{HU}'s Q_{HU} – also remains perfectly simulated. C runs $\mathcal{A}$, and with probability $\geq \frac{1}{l}$, the query that $\mathcal{A}$ won with, used $ipk_i = pk$. In this case, we retrieve the respective *attr* and π_{VS}, decrypt the enclosed ciphertext to obtain *cert*, and output $(\sigma := cert, m := uid^* \| attr)$ as forgery for EUF-CMA. Note that we only look up this query in Q_τ. In the following, we address each of the four cases of the definition individually, which will reveal that the query cannot exist in $Q_{\pi,\tau}$.

Case 1 - $(uid^*, -, -) \notin Q_{\mathsf{CU}} \cup Q_{\mathsf{HU}}$: The condition requires that uid^* was never certified by any honest issuer. Since uid^* does not exist, there cannot be any successful query to the honest user oracle $O_{\pi,\tau}$ for that uid^*, i.e,. we know there is no matching tuple in $Q_{\pi,\tau}$. To get a validation token from O_τ, the adversary must provide it with a certificate *cert* from an honest issuer (this follows from the policy bit 1 signed in τ^*), that was never produced by that honest issuer. This gives an immediate forgery for $\mathsf{S_I}$.

Case 2 - $(uid^*, attr, ipk) \in Q_{\mathsf{CU}} \wedge (sid^*, nym^*, ipk, attr) \notin Q_\tau$: A certificate was issued to a corrupt user uid^* from an honest issuer ipk over *attr*, but O_τ did not see a query with $(sid^*, nym^*, ipk, attr)$. Because uid^* is corrupt, we know no there is no matching tuple in the state maintained by $O_{\pi,\tau}$ either. Then a query to O_τ must be w.r.t. to some $attr' \neq attr$ or honest issuer's public key $ipk' \neq ipk$, that is, not matching what the honest issuer ipk originally certified for this uid^*. If $attr' \neq attr$, but $ipk' = ipk$, this is a forgery for the honest ipk. If $ipk' \neq ipk$, we know by the game's definition that ipk' must also belong to an honest issuer. We also assumed uid's

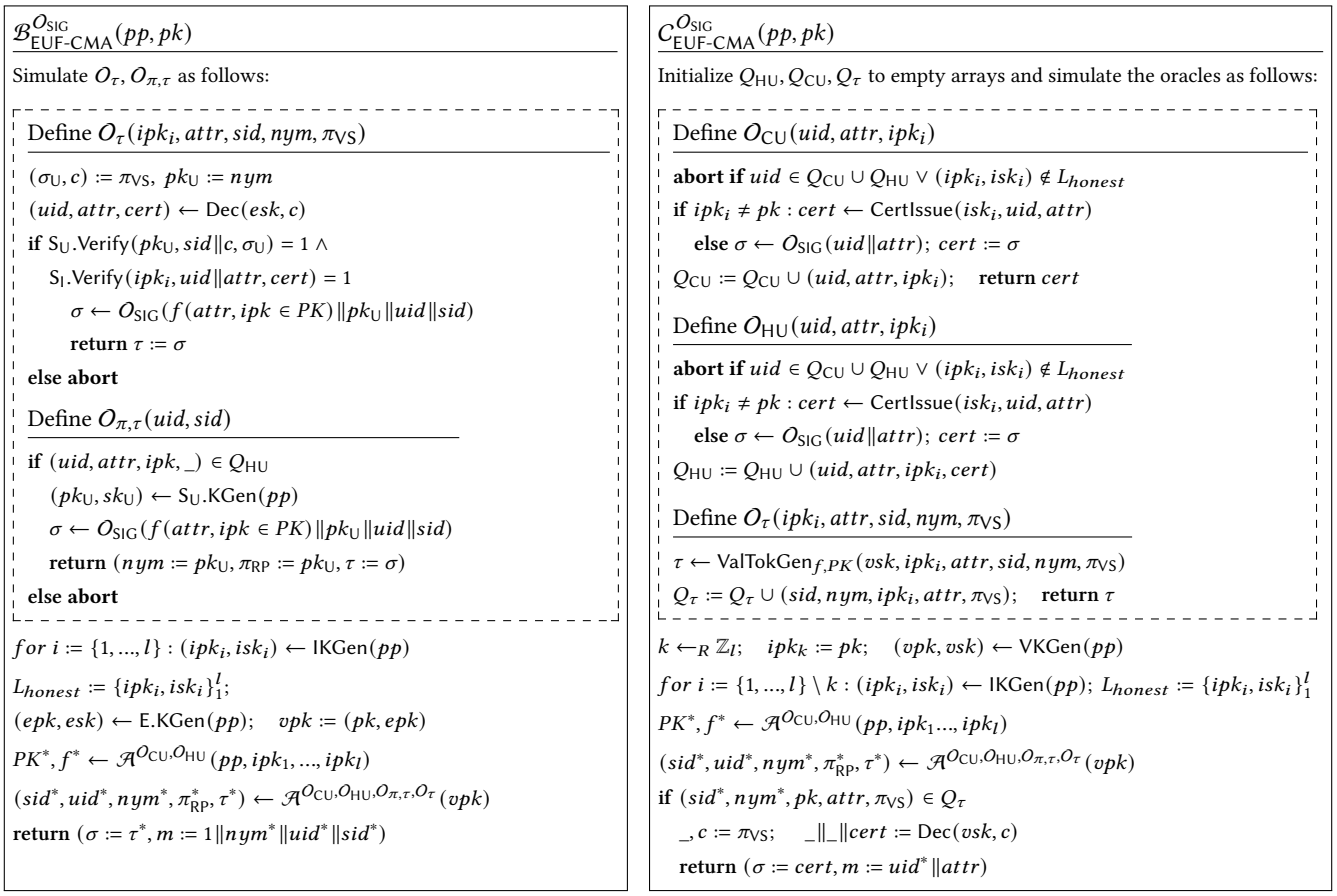

Figure 9: EU-COV's adversaries against EUF-CMA of the schemes S_{VS} and S_I. The adversaries $\mathcal{B}, C$ use the adversary $\mathcal{A}$ who wins in $Exp^{weakUF}_{\mathcal{A},COV}$ with direct and indirect forgeries respectively to win in EUF-CMA.

to be globally unique, and ensured that uid^* was signed only by ipk. This yields a forgery for S_I under ipk'.

Case 3 - $(uid^*, attr, ipk) \in Q_{HU} \wedge (sid^*, nym^*, ipk, attr, uid^*) \notin Q_{\pi_{VS},\tau}$: A certificate was issued for an honest user uid^*, but $O_{\pi,\tau}$ did not see a query with $sid^*, nym^*, ipk, attr, uid^*$. To get a validation token from O_τ, the adversary must present a valid π_{VS} to it, i.e., one that contains a valid certificate for uid^*. But because certificates of honest users are never revealed to the adversary, it must have forged $cert$. However, this would not be a fresh forgery for EUF-CMA as there is already a certificate issued by the honest issuer ipk for $(uid^*, attr)$, i.e., O_{SIG} has already seen a query for this message. Therefore, we make the reduction work by not generating a certificate for uid^* and "simulate" it instead. We do this by adjusting O_{HU} to not issue certificates, but only store requests for them:

Define $O_{HU}(uid, attr, ipk_i)$

abort if $uid \in Q_{CU} \cup Q_{HU} \vee (ipk_i, isk_i) \notin L_{honest}$
if $ipk_i \neq pk$
$cert \leftarrow CertIssue(isk_i, uid, attr)$
else $cert := \perp$
$Q_{HU} := Q_{HU} \cup (uid, attr, ipk_i, cert)$

To maintain the indistinguishability of our changes w.r.t. the adversary's perspective, we additionally change $O_{\pi,\tau}$ to not run $ValTokGen_{f,PK}$ but directly create the signature for honest users:

Define $O_{\pi,\tau}(uid, sid)$

if $(uid, attr, ipk, \_) \in Q_{HU}$
$(pk_U, sk_U) \leftarrow S_U.KGen(pp)$
$(sk_{VS}, \_) := vsk$
$\sigma_{VS} \leftarrow S_{VS}.Sign(sk_{VS}, f(attr, ipk \in PK)\|pk_U\|uid\|sid)$
return $(nym := pk_U, \pi_{RP} := pk_U, \tau := \sigma_{VS},)$
else abort

We stress that this kind of simulation does not require any zero-knowledge-like behaviour from the signature scheme, but fully relies on the fact that we are in the *weak* unforgeability game. This means that $\mathcal{A}$ does not have access to O_π, and thus never sees the user's certificates. We then simply keep track of the signed information internally.

With that, we are perfectly simulating the oracles and at the same time, C extracts a fresh forgery for S_I from $\mathcal{A}$'s interaction with O_τ.

Case 4 - $(uid^*, attr, ipk) \in Q_{\mathsf{CU}} \cup Q_{\mathsf{HU}} \wedge f(attr, ipk \in PK) = 0$: This case requires that the forgery is for a user uid^* (corrupt or honest) that has a proper certificate on $attr$ from an honest issuer ipk, but with $(ipk, attr)$ not satisfying the policy f. Any matching query to either O_τ or $O_{\pi,\tau}$, must be with some $attr' \neq attr$ or $ipk' \neq ipk$, i.e., different than what has been originally certified, such that f produces 1. This is impossible in $O_{\pi,\tau}$, which internally always uses the correct certificate, thus must have happened with O_τ. Like in Case 2, this results in a forgery for $\mathsf{S_I}$ under ipk'.

Note that unlike Case 3, we do not require oracle adjustments for the reduction to work when uid^* is honest. This is because we have proven that the adversary's forged $cert$ is w.r.t. $ipk', attr'$ that were never used together in certifying uid^*, making $(\sigma := cert, m := uid^* \| attr')$ a fresh forgery in the EUF-CMA game of $\mathsf{S_I}$. □

C.2 PP-COV Unforgeability

Recall that the adversary $\mathcal{A}$ in the game $Exp^{\mathsf{UF}}_{\mathcal{A},\mathsf{COV}}$ outputs uid^*, sid^*, nym^*, π^*_{RP}, τ^* and wins if $\mathsf{ValTokVerify}(vpk, uid^*, sid^*, nym^*, \pi^*_{\mathsf{RP}}, \tau^*) = 1$ and one of Case 1-4 annotated in the game holds. In the following, we use the adversary $\mathcal{A}$ that wins in each case to construct efficient adversaries $\mathcal{B}$ and $\mathcal{C}$ against EUF-CMA and MMS-EUF-CMA of the schemes S and MMS respectively.

The proof branches into direct forgery against S when we have $(sid^*, nym^*) \notin Q_\tau \cup Q_{\pi,\tau}$, and into indirect forgeries. For the former, we use the adversary $\mathcal{A}$ to construct the adversary $\mathcal{B}$ in the EUF-CMA game. Shown in Figure 10, $\mathcal{B}$ is initialized with a public key from the scheme S, which is the honest $\mathcal{VS}$'s public key, and we modify O_τ and $O_{\pi,\tau}$ to forward requests to O_{SIG} if certificate verification succeeds. We do not alter the other oracles $O_{\mathsf{CU}}, O_{\mathsf{HU}}, O_\pi$ as their operation is independent of what $O_{\pi,\tau}$ and O_τ do. With the oracles given to $\mathcal{A}$ perfectly simulated, $\mathcal{B}$ runs $\mathcal{A}$ and outputs $(\sigma := \tau^*, m := nym^* \| sid^*)$ as a forgery in the EUF-CMA game.

The rest of the proof is for indirect forgeries, i.e., $(sid^*, nym^*) \in Q_\tau \cup Q_{\pi,\tau}$. In this case, we will use $\mathcal{A}$ to construct the adversary $\mathcal{C}$ in the MMS-EUF-CMA game. In Figure 10, $\mathcal{C}$ is initialized with a public key pk from the scheme MMS, which it inserts as an issuer's public key into a random location in the list of public keys of honest issuers that $\mathcal{A}$ takes. The oracles $O_{\mathsf{CU}}, O_{\mathsf{HU}}$ are modified to forward queries to O_{MMS} if the input ipk_i equals pk, and O_τ is modified to store π_{VS} that it gets in queries. We do not require such modification to $O_{\pi,\tau}$ as we will show that the adversary's forgeries could not have originated from it. Further, as our changes perfectly simulate the original oracles, the unchanged $O_{\pi,\tau}, O_\pi$ – which accesss O_{HU}'s Q_{HU} – remain perfectly simulated. $\mathcal{C}$ runs $\mathcal{A}$, then with probability $\geq \frac{1}{l}$, the query in $Q_\tau \cup Q_{\pi_{\mathsf{VS}},\tau}$ used $ipk_i = pk$. In this case, we retrieve the respective $attr, \pi_{\mathsf{VS}}$ and use an extractor algorithm $\mathcal{E}$ to obtain the witness $cert$ from π_{VS}, and output $(\sigma := cert, (m_1 := uid^*, m_2 := attr))$ as forgery in the MMS-EUF-CMA game. Note that $\mathcal{C}$ looks up the query in Q_τ but not $Q_{\pi_{\mathsf{VS}},\tau}$ – we will show in the proofs that it can only exist in Q_τ.

Lemma C.1. *(Case 1) If* MMS *and* S *are unforgeable according to* MMS-EUF-CMA *and* EUF-CMA *respectively,* NIZK *is special sound (which is implied by simulation soundness), and* Com *is binding, then* PP-COV *is unforgeable when* $(uid^*, -, -) \notin Q_{\mathsf{CU}} \cup Q_{\mathsf{HU}}$.

Proof. The adversary's forgery is for a user uid^* that has never received any certificate. Thus, we know the session validation cannot stem from $O_{\pi,\tau}$, as this only returns tokens for honest users. So $(sid^*, nym^*, ipk, attr) \in Q_\tau$ and the honest validator must have accepted the provided proof π_{VS} in that session. To get a session validation from O_τ, it must be provided with valid values for the commitment nym^* and the NIZK π_{VS}. In this case, the NIZK soundness requires knowledge of the witnesses $cert$ and uid.

Furthermore, the binding property of nym^* ensures that this uid must be the same one output by the adversary (i.e., $uid = uid^*$).

As both O_{CU} and O_{HU} have not seen any queries for uid^*, then the adversary used a forged $cert$. This makes $(\sigma := cert, m_1 := uid^*, m_2 := attr)$ a valid forgery for MMS. In the reduction, C uses the special soundness of the NIZK to extract $cert$ from π_{VS} and wins against MMS-EUF-CMA if $pk = ipk$. □

Lemma C.2. *(Case 2) If* MMS *and* S *are unforgeable according to* MMS-EUF-CMA *and* EUF-CMA *respectively,* NIZK *is special sound, and* Com *is binding, then* PP-COV *is unforgeable when* $(uid^*, attr, ipk) \in Q_{\mathsf{CU}} \wedge (sid^*, nym^*, ipk, attr) \notin Q_\tau$.

Proof. The condition requires O_τ to not have seen a query involving sid^*, nym^* and the $ipk, attr$ used in certifying the corrupt uid^*. It is also not possible to receive τ^* from $O_{\pi,\tau}$ using a corrupt uid^*. Thus, the adversary must have queried O_τ using some $ipk' \neq ipk$ or $attr' \neq attr$. If $attr' \neq attr$, but $ipk' = ipk$, this is a forgery for the honest ipk. If $ipk' \neq ipk$, we know by the game's definition that ipk' must also belong to an honest issuer. As nym^* is bound to uid^*, uid^* is assumed to be globally unique, and we ensured that uid^* was signed only by ipk, we must have a forgery for MMS under ipk'. By NIZK soundness, C extracts the forged certificate and wins in the MMS-EUF-CMA game. □

Lemma C.3. *(Case 3) If* S *and* MMS *are unforgeable in* EUF-CMA *and* MMS-EUF-CMA, NIZK *is simulation-sound extractable and zero knowledge, and* Com *is binding, then* PP-COV *is unforgeable when* $(uid^*, attr, ipk) \in Q_{\mathsf{HU}} \wedge (sid^*, nym^*, ipk, attr, uid^*) \notin Q_{\pi_{\mathsf{VS}},\tau}$.

Proof. The condition forbids having a query to $O_{\pi,\tau}$ that involves sid^*, uid^*, and nym^*, thus a query to O_τ must exist. This query must contain a π_{VS} bound to nym^*, that in turn is bound to uid^* due to Com binding. As the adversary has no way of obtaining π_{VS} or the underlying $cert$ for the honest uid^* through any of the oracles, it must have a forged π_{VS}, and – by NIZK soundness – the underlying $cert$.

However, this forgery would be for some $(\sigma := cert, m_1 := uid^*, m_2 := attr)$ that is not fresh w.r.t. to O_{MMS}. To be able to carry out the reduction, we modify O_{HU} to not issue certificates at all, but only store requests for them:

Define $O_{\mathsf{HU}}(uid, attr, ipk_i)$

abort if $uid \in Q_{\mathsf{CU}} \cup Q_{\mathsf{HU}} \vee (ipk_i, isk_i) \notin L_{honest}$

if $ipk_i \neq pk : cert \leftarrow \mathsf{CertIssue}(isk_i, uid, attr)$ **else** $cert := \perp$

$Q_{\mathsf{HU}} := Q_{\mathsf{HU}} \cup (uid, attr, ipk_i, cert)$

We accommodate this change within $O_{\pi,\tau}$ and O_π by not running CertPresent, instead, we simulate the NIZK proof when O_{HU} has seen the given uid^* with $ipk_i = pk$ before. This is possible because the NIZK is zero knowledge:

$\mathcal{B}^{O_{\mathsf{SIG}}}_{\mathsf{EUF\text{-}CMA}}(pp, pk)$:

Simulate $O_\tau, O_{\pi,\tau}$ as follows:

Define $O_\tau(ipk_i, attr, sid, nym, \pi_{\mathsf{VS}})$

if π_{VS} verifies w.r.t. $spk, ipk_i, attr, sid, nym \wedge$
  $f(attr, ipk_i \in PK) = 1$
  $\sigma \leftarrow O_{\mathsf{SIG}}(nym \| sid)$
  return $\tau := \sigma$
else abort

Define $O_{\pi,\tau}(uid, sid)$

if $(uid, attr, ipk, \_) \in Q_{\mathsf{HU}}$
  $(com_{uid}, o_{uid}) \leftarrow \mathsf{Commit}(uid)$
  if $f(attr, ipk \in PK) = 1$
    $\sigma \leftarrow O_{\mathsf{SIG}}(nym \| sid)$
    return $(nym := com_{uid}, \pi_{\mathsf{RP}} := o_{uid}, \tau := \sigma)$
  else
    return $(nym := com_{uid}, \pi_{\mathsf{RP}} := o_{uid}, \perp)$

$for\ i := \{1, ..., l\} : (ipk_i, isk_i) \leftarrow \mathsf{IKGen}(pp)$
$vpk := pk; \quad L_{honest} := \{ipk_i, isk_i\}_1^l$
$PK^*, f^* \leftarrow \mathcal{A}^{O_{\mathsf{CU}}, O_{\mathsf{HU}}, O_\pi}(pp, ipk_1, ..., ipk_l)$
$(sid^*, uid^*, nym^*, \pi^*_{\mathsf{RP}}, \tau^*) \leftarrow \mathcal{A}^{O_{\mathsf{CU}}, O_{\mathsf{HU}}, O_\pi, O_{\pi,\tau}, O_\tau}(vpk)$
return $(\sigma := \tau^*, m := nym^* \| sid^*)$

$\mathcal{C}^{O_{\mathsf{MMS}}}_{\mathsf{MMS\text{-}EUF\text{-}CMA}}(pp, pk)$:

Initialize $Q_{\mathsf{HU}}, Q_{\mathsf{CU}}, Q_\tau$ to empty arrays and simulate the oracles as follows:

Define $O_{\mathsf{CU}}(uid, attr, ipk_i)$

abort if $uid \in Q_{\mathsf{CU}} \cup Q_{\mathsf{HU}} \vee \quad (ipk_i, isk_i) \notin L_{honest}$
if $ipk_i \neq pk : cert \leftarrow \mathsf{CertIssue}(isk_i, uid, attr)$
  else $\sigma \leftarrow O_{\mathsf{MMS}}(uid, attr); \quad cert := \sigma$
$Q_{\mathsf{CU}} := Q_{\mathsf{CU}} \cup (uid, attr, ipk_i); \quad$ **return** $cert$

Define $O_{\mathsf{HU}}(uid, attr, ipk_i)$

abort if $uid \in Q_{\mathsf{CU}} \cup Q_{\mathsf{HU}} \vee (ipk_i, isk_i) \notin L_{honest}$
if $ipk_i \neq pk : cert \leftarrow \mathsf{CertIssue}(isk_i, uid, attr)$
  else $\sigma \leftarrow O_{\mathsf{MMS}}(uid, attr); \quad cert := \sigma$
$Q_{\mathsf{HU}} := Q_{\mathsf{HU}} \cup (uid, attr, ipk_i, cert)$

Define $O_\tau(ipk_i, attr, sid, nym, \pi_{\mathsf{VS}})$

$\tau \leftarrow \mathsf{ValTokGen}_{f,PK}(vsk, ipk_i, attr, sid, nym, \pi_{\mathsf{VS}})$
$Q_\tau := Q_\tau \cup (sid, nym, ipk_i, attr, \pi_{\mathsf{VS}}); \quad$ **return** τ

$k \leftarrow_R \mathbb{Z}_l; \quad ipk_k := pk; \quad (vpk, vsk) \leftarrow \mathsf{VKGen}(pp)$
$for\ i := \{1, ..., l\} \setminus k : (ipk_i, isk_i) \leftarrow \mathsf{IKGen}(pp);\ L_{honest} := \{ipk_i, isk_i\}_1^l$
$PK^*, f^* \leftarrow \mathcal{A}^{O_{\mathsf{CU}}, O_{\mathsf{HU}}, O_\pi}(pp, ipk_1 ..., ipk_l)$
$(sid^*, uid^*, nym^*, \pi^*_{\mathsf{RP}}, \tau^*) \leftarrow \mathcal{A}^{O_{\mathsf{CU}}, O_{\mathsf{HU}}, O_\pi, O_{\pi,\tau}, O_\tau}(vpk)$
if $(sid^*, nym^*, pk, attr, \pi_{\mathsf{VS}}) \in Q_\tau$
  $cert := \mathcal{E}(\pi_{\mathsf{VS}})$
  return $(\sigma := cert, (m_1 := uid^*, m_2 := attr))$

Figure 10: PP-COV's adversaries against EUF-CMA and MMS-EUF-CMA of S and MMS. The adversaries $\mathcal{B}, C$ use the adversary $\mathcal{A}$ who wins in $Exp^{\mathsf{UF}}_{\mathcal{A},\mathbf{COV}}$ with direct and indirect forgeries to win in EUF-CMA and MMS-EUF-CMA respectively.

Define $O_{\pi,\tau}(uid, sid)$

if $(uid, attr, ipk, cert) \in Q_{\mathsf{HU}}$
  if $ipk = pk : (nym, \pi_{\mathsf{RP}}) \leftarrow \mathsf{Commit}(uid)$
    $\pi_{\mathsf{VS}} \leftarrow \mathcal{S}(ipk, vpk, sid, attr, nym)$
  else $(\pi_{\mathsf{VS}}, \pi_{\mathsf{RP}}, nym) \leftarrow \mathsf{CertPresent}(ipk, vpk, uid, attr, sid, cert)$
  $\tau \leftarrow \mathsf{ValTokGen}_{f,PK}(vsk, ipk, attr, sid, nym, \pi_{\mathsf{VS}})$
  return $(nym, \pi_{\mathsf{RP}}, \tau)$

Define $O_\pi(vpk_j, uid, sid)$

if $vpk_j \neq vpk \wedge (uid, attr, ipk, cert) \in Q_{\mathsf{HU}}$:
  if $ipk = pk : (nym, \pi_{\mathsf{RP}}) \leftarrow \mathsf{Commit}(uid)$
    $\pi_{\mathsf{VS}} \leftarrow \mathcal{S}(ipk, vpk_j, sid, attr, nym)$
  else $(\pi_{\mathsf{VS}}, \pi_{\mathsf{RP}}, nym) \leftarrow \mathsf{CertPresent}(ipk, vpk_j, uid, attr, sid, cert)$
  return $(\pi_{\mathsf{VS}}, \pi_{\mathsf{RP}}, nym)$

With that, we still perfectly simulate the oracles given to the adversary in $Exp^{\mathsf{UF}}_{\mathcal{A},\mathsf{COV}}$ and at the same time extract a fresh forgery against MMS-EUF-CMA. □

Lemma C.4. *(Case 4) If* MMS *is unforgeable in* MMS-EUF-CMA, S *is unforgeable in* EUF-CMA, NIZK *is simulation-sound extractable and zero knowledge, and* Com *is binding, then* PP-COV *is unforgeable when* $(uid^*, attr, ipk) \in Q_{\mathsf{CU}} \cup Q_{\mathsf{HU}} \wedge f(attr, ipk \in PK) = 0$.

Proof. Any query to O_τ or $O_{\pi,\tau}$ that produces τ^* must be w.r.t. to some $attr'$ and ipk' such that $f(attr', ipk' \in PK) = 1$. Further, satisfying the second condition of Case 4 requires that either $attr' \neq attr$ or $ipk' \neq ipk$. This is impossible in $O_{\pi,\tau}$ which internally always uses the ipk and $attr$ certified with uid^*, thus the query took place towards O_τ. Like in Case 2, this results in a forgery for MMS under ipk'.

In contrast to Case 3, as this forgery is w.r.t. $ipk', attr'$ that were never used together in certifying uid^*, we do not require oracle adjustments for the reduction to work when uid^* is honest. □

D PP-COV INSTANTIATION

We include the full instantiation details of all algorithms under the PP-COV construction in Figure 11. PS Signatures [36] are used for MMS, Pedersen Commitments [35] for Com, ECDSA [31] for the signature scheme S, and generalized Schnorr-signature proofs of knowledge [4] for NIZK, which are made non-interactive via Fiat-Shamir heuristic [17].

$\mathsf{Setup}(1^\lambda) \to pp$

- $(\mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e, p) \leftarrow \mathsf{PGen}(1^\lambda)$ generates three cyclic groups with prime order p and type-3 pairing with $e : \mathbb{G} \times \tilde{\mathbb{G}} \to \mathbb{G}_T$, for use in PS Signatures. In practice, already established and publicly known pairing groups are used.
- $(\hat{g}, \hat{h}) \leftarrow_R \tilde{\mathbb{G}}^2$ are random generators of the pairing's source group $\tilde{\mathbb{G}}$ for use in Pedersen commitments.
- $(\bar{\mathbb{G}}, q, \bar{g}) \leftarrow \mathsf{GGen}(1^\lambda)$ generates a secure elliptic curve group with prime order q and generator $\bar{g}$, for use in ECDSA. In practice, already established and publicly known groups and generators are used.

return $pp := (\mathbb{G}, \tilde{\mathbb{G}}, \mathbb{G}_T, e, p, \hat{g}, \hat{h}, \bar{\mathbb{G}}, q, \bar{g})$

$\mathsf{IKGen}(pp)$

$\tilde{g} \leftarrow_R \tilde{\mathbb{G}}; \quad (x, y_1, y_2) \leftarrow_R \mathbb{Z}_p^3$

$(\tilde{X}, \tilde{Y}_1, \tilde{Y}_2) := \tilde{g}^x, \tilde{g}^{y_1}, \tilde{g}^{y_2}$

return $(ipk := (\tilde{g}, \tilde{X}, \tilde{Y}_1, \tilde{Y}_2), isk := (x, y_1, y_2))$

$\mathsf{VKGen}(pp)$

$(pk, sk) \leftarrow \mathsf{ECDSA.KGen}(\bar{\mathbb{G}}, q, \bar{g})$

return $(vpk := pk, vsk := sk)$

$\mathsf{CertIssue}(isk, uid, attr)$

$(x, y_1, y_2) := isk$

$h \leftarrow_R \mathbb{G} \backslash \{1_\mathbb{G}\}$

$s := h^{x + uid \cdot y_1 + attr \cdot y_2}$

return $cert := (h, s)$

$\mathsf{CertVerify}(ipk, uid, attr, cert)$

$(h, s) := cert; \quad (\tilde{g}, \tilde{X}, \tilde{Y}_1, \tilde{Y}_2) := ipk$

return 1 **if** $h \neq 1_\mathbb{G} \wedge e(h, \tilde{X} \cdot \tilde{Y}_1^{uid} \cdot \tilde{Y}_2^{attr}) = e(s, \tilde{g})$

$\mathsf{CertPresent}(ipk, vpk, uid, attr, sid, cert)$

$(\tilde{g}, \tilde{X}, \tilde{Y}_1, \tilde{Y}_2) := ipk; \quad (h, s) := cert; \quad (o_{uid}, r, t) \leftarrow_R \mathbb{Z}_p^3$

$com_{uid} := \hat{g}^{uid} \hat{h}^{o_{uid}}; \quad (h', s') := (h^t, (s^t \cdot h^{tr}))$

$\pi \leftarrow \mathsf{NIZK}\{(uid, o_{uid}, r) :$

$e(h', \tilde{Y}_1)^{uid} \cdot e(h', \tilde{g})^r = \dfrac{e(s', \tilde{g})}{e(h', \tilde{X} \cdot \tilde{Y}_2^{attr})} \wedge$

$\hat{g}^{uid} \cdot \hat{h}^{o_{uid}} = com_{uid}\}(ipk, vpk, sid, attr, com_{uid})$

return $(\pi_{\mathsf{VS}} := (h', s', \pi), nym := com_{uid}, \pi_{\mathsf{RP}} := o_{uid})$

$\mathsf{ValTokGen}_{f,PK}(vsk, ipk, attr, sid, nym, \pi_{\mathsf{VS}})$

$(\tilde{g}, \tilde{X}, \tilde{Y}_1, \tilde{Y}_2) := ipk; \quad com_{uid} := nym; \quad (h', s', \pi) := \pi_{\mathsf{VS}}$

if $h' \neq 1_\mathbb{G} \wedge \pi$ verifies w.r.t. $(ipk, vpk, sid, attr, com_{uid}) \wedge$

$f(attr, ipk \in PK) = 1$:

$\sigma \leftarrow \mathsf{ECDSA.Sign}(sk := vsk, m := com_{uid} \| sid)$

return $\tau := \sigma$

else return $\perp$

$\mathsf{ValTokVerify}(vpk, uid, sid, nym, \pi_{\mathsf{RP}}, \tau)$

$o_{uid} := \pi_{\mathsf{RP}}; \quad com_{uid} := nym$

return 1 **if** $\mathsf{ECDSA.Verify}(pk := vpk, m := com_{uid} \| sid, \sigma := \tau) = 1 \wedge com_{uid} = \hat{g}^{uid} \hat{h}^{o_{uid}}$ **else** 0

Figure 11: Secure Instantiation of PP-COV