Achieving Higher Level of Assurance in Privacy Preserving Identity Wallets

***Abstract*—Recent advances in decentralized digital identity based on Verifiable Credentials utilize identity wallets to ensure that the identity data control remains with the user. However, they still lack higher Level of Assurance guarantees, restricting their full potential. In this paper, we design and showcase DOOR, a scheme that enables identity wallets to utilize a hardware root of trust and bring them in alignment with emerging regulations and standards that require higher level of assurances for services (e.g. eIDAS). At the same time, we make sure that privacy-enhancing properties like selective-disclosure are fully supported, in order to make the wallet compliant with privacy regulations (e.g. GDPR). To achieve all these we have designed an enhanced variant of DAA-A crypto protocol to offer anonymity, unlinkability, and unforgeability, while being the first to offer strong guarantees on the Wallet's integrity when constructing attribute attestations. We formally prove the security properties of DOOR and evaluate the performance of its implementation for every phase of the credential management.**

***Index Terms*—selective disclosure, anonymous credentials, identity wallet, trusted computing**

I. INTRODUCTION

Currently, there is an increasing shift to decentralized digital identity models, where there is no single governing organization that has control over identity data origination. Instead, participants produce and manage their own identifiers and credentials without deference or permission from any other administrative organization. The World Wide Web Consortium (W3C) is currently developing two new standards to realize this emerging model, namely, Decentralized Identifiers (DIDs) [1] and Verifiable Credentials (VCs) [2].

In the ecosystem of Verifiable Credentials, the Issuer issues a credential containing a set of claims on a Subject and transfers it to a Holder, who is typically the same entity. The Holder stores the VCs in a storage called the Identity Wallet. In response to a request from the Verifier, the Holder retrieves one or more stored VCs from her Wallet and presents them to the Verifier. Alternatively, the Holder can construct Verifiable Presentations (VPs), i.e. a collection of claims that a Holder can contract from different VCs issued by varying entities. Then, a Holder can prove to a Verifier that it owns a VC or VP with certain attributes. This is usually achieved through a unique identifier (e.g., public key), owned by the Holder that enables her to generate proof of possession on specific claims (e.g., a digital signature with the corresponding private key).

VCs can be combined with anonymous credentials [3], [4] to enable the Holder to manage her privacy by choosing the level of information disclosure. That is, the Holder can select only some of the attributes in the credentials she owns and prove that they are certified by a trusted Issuer, without revealing any further information; i.e., a signature from the Holder's unique identifier or other remaining attributes. This property is called *selective disclosure*.

One core challenge is the verification of the integrity and origin of the presented VCs or VPs: *How can someone be sure that they really belong to the claimed entity?* On a technical level, this translates into Holders having control of their own VCs and DIDs through their Wallets, which can ensure that credentials and (private) keys can only become available to this specific Holder as the actual owner of the issued wallet credentials. Since it is only the Holder (as the Identity Owner) that knows the (private) key associated with a DID, the level of control and credential management assurance relies solely on possessing and controlling the private key, which in current designs is a software-based key. However, the use of a software keystore introduces many security risks and raises trustworthiness issues [5], [6]. So the above challenge translates into the more technical question: *How can a Verifier be sure that the respective key of the Holder presenting a VC remains under her control, and cannot be used by any other unauthorized entity?*

This question is particularly important because it relates to the requirement towards achieving a certain Level of Assurance (LoA) behind credential management [7]. The LoA characterizes the degree of confidence in the electronic identification means, thus providing assurance that the person claiming a particular identity is, in fact, the intended recipient to which that identity is assigned. For example, in the case of Europe, the eIDAS regulation [8] clearly defines the requirement for multiple authentication factors to achieve an LoA classified as "substantial" (e.g., fingerprints and secret key). Bare proof-of-possession of a SW-based (private) key does not achieve even the lowest LoA in eIDAS, since it involves only a single authentication factor.

One of the necessary measures to solve such security gaps, and reach a "high" LoA, is to isolate the keys from the Holder while still being stored in the user's domain. There are several types of isolation defined in the literature that can be achieved through the incorporation of trusted computing technologies [9], i.e., Hardware Secure Module (HSM), Trusted Platform Module (TPM), or Trusted Execution Environment (TEE). All such trusted components provide reliable, tamper-evident, and secure processing units (including support for crypto operations, key storage, and authentication) that offer a higher level of trust for the executed applications.

Another requirement to achieve LoA "high" is the binding of identity data to the Holder (*Holder Binding*). This binding is based on a unique identifier representing the Holder, i.e., a secret key. One way to have high confidence is to make sure that the secret key is bound to the Wallet managing the identity data. For instance, DIF defines this property as Device

Binding, that is, “a building block that enables a differential credential security model by anchoring a hardware-generated key (e.g., TPM Key) to the credential.” [10]. This sets the challenge ahead: *How can we achieve both requirements for higher LoA while empowering the user to control the level of her privacy by selectively disclosing only those attributes needed, for accessing a service, in a verifiable manner?* This requires the Wallet to not only be equipped with a HW-based Root-of-Trust but to use this trust anchor for securely managing attribute keys and creating attribute attestations that provide proof-of-possession about the unique Holder identifiers, but without disclosing any further information, thus, enabling the property of selective disclosure.

To this end, if we want to consider the core feature of selective disclosure, we have to be able to use zero-knowledge (ZK) proofs for each attribute separately. This essentially boils down in been able to represent each attribute with a separate key and present the necessary commitments as proof-of-possession. However, this further aggravates the problem of assurance since such schemes require the use of additional software-based keys resulting into a difficulty to achieve balance between safeguarding user privacy (anonymity and selective disclosure) while guaranteeing integrity and unforgeability of the produced ZK attribute attestations.

All in all, the challenge of building a solution facilitating hardware-based keys becomes more pressing and extends to not only binding a credential to the Wallet but also binding each Holder identifying attribute to the credential and, in turn, to the host Wallet, thus, achieving a “chain-of-trust” when presenting verifiable attribute attestations. What is needed are new mechanisms and security controls for managing attribute-based credentials, safeguarded through HW-based keys, providing an efficient way to disclose a Holder’s personal attributes, while minimizing risk of sensitive data revelation and thus granting anonymity, unforgeability and unlinkability. Indeed, the privacy at the attribute level has been investigated by other approaches [11], [12], but combining this with the device binding of attribute keys, thus, enabling hardware-bound attribute-based credentials, has never been studied before.

Contribution: To solve this open problem, this paper is the first to propose a protocol leveraging an enhanced variant of Direct Anonymous Attestation (DAA) [13] where each one of the attributes can be represented as a key, bound to the Holder’s unique identifier, which in turn is bound to the underlying trusted component. This allows the user to create privacy-preserving attribute claims (disclosing only those attributes needed to be checked against service access control policies) with strong trust guarantees on the correctness and origin of the attributes. More specifically, we propose DOOR, a new scheme that showcases how hardware-based keys can achieve the envisioned property of higher LoA for credential management, while enabling privacy preservation via selective disclosure. DOOR proposes an enhanced variant of an anonymous signature scheme, namely Attribute-based DAA (DAA-A) [14], to achieve this. It enables the construction of VPs with selective disclosure to satisfy legal requirements of privacy protection (e.g., GDPR). DAA-A is a strong privacy-preserving authentication scheme that enables the representation of attributes as keys, hence enabling the encoding of complex attribute structures as key hierarchies. This, in turn, allows arbitrary policies to be checked against individual attributes without complicating or overwhelming the credential. Verifiers can dynamically define policies, as high-level programs, that can be verifiably executed by Holders for constructing the necessary VPs as claims anchored to their credentials.

To overcome the current limitation of traditional crypto schemes that do not consider VC/VP linkability issues leading to Holder profiling, we have designed an enhanced variant of DAA-A with “credential blinding” capabilities. It ensures Holder anonymity and VC/VP unlinkability, and unforgeability while being the first to offer strong guarantees on the Wallet’s integrity when constructing attribute attestations. DOOR also ensures the binding of the identity data, at the attribute level, to the Holder by cryptographically binding the Wallet to the intended owner. Through this way, we offer higher levels of confidence to the authentication and electronic identification service of the Wallet - hence a higher LoA as required by emerging regulations. In this paper, we further propose a formal definition of the security properties that a protocol should offers to achieve LoA ”high”, and then we provide a mathematical definition of these properties and proof of the correctness and soundness of our scheme. We also present a performance analysis and evaluation of DOOR based on real-world implementation.

A. Wallet Properties & Requirements

Our design follows the concepts and roles defined by W3C for the ecosystem of VCs [2]. The main actors include the Issuer, Subject, Holder, and Verifier. This model should satisfy additional security properties, in order to allow any Wallet to achieve the highest LoA, extending the set of requirements defined in ISO 29115 and the eIDAS implementation act [7]. Based on that, we come up with specific security properties, following the simulation-based security model of the UC framework, based on *composable proofs*, which allows our scheme properties to fit any wallet design.

Definition 1: (Holder Binding) *It must be ensured that the issued identity data are delivered only to the intended Holder.* ■

The intention of this definition is to safeguard against adversaries that try to construct VPs without having access or being the intended recipient of the issued credentials. This might occur, for instance when an adversary gets access to a Holder’s VCs (but not her unique identifier - secret key) and constructs VPs that would be accepted by a Verifier. We differentiate this from the scenario where a legitimate Holder is acting on behalf of another user (e.g., parents attesting to attributes of their children).

Definition 2: (Device Binding) *Issued VCs should be* bound *to the Holder’s unique identifier (i.e. secret key) and no one should be able to use or show this credential without proof of possession of this unique identifier.*■

In continuation to the Holder Binding property, this definition further ensures the issuance of credentials *bound* to the

Holder's secret key, so that no one can show this credential without such secret key. This requires the anchoring of the public part of the Holder's secret key to the credential. The key needs to be a hardware-based key originating from a Trusted Component (TC), hosted on the Holder, so that additional security policies can be enforced for protecting against key leakage and ensuring that only the Holder's authenticated Wallet can securely contract the key for creating signed attribute attestations.

Definition 3: (Selective Disclosure) *VPs should constitute collections of claims that the Holder can construct (from different VCs) disclosing only those attributes needed for verification without revealing further information on the claims, such as the signature of the VC Issuer or other remaining attributes.*■

Selective disclosure guarantees that if all I credentials with any K attributes each are correctly issued to a Holder by L honest VC Issuers, then any presentation with selective disclosure and proof of possession of the original credential (indicated by c) as well as proof of Wallet integrity (indicated by signature σ_D), correctly computed by the Holder will be accepted by the Verifier.

Definition 4: (Full Anonymity) *Shared VCs and/or VPs are considered* anonymous *when no adversary or (single) "honest-but-curious" infrastructure entity can identify the Holder presenting a claim (based on a set of issued attributes x) or learn anything about the Holder except to the extent that it is trivially learned from the VC Issuers' public key required to verify the claim. Full anonymity also includes unlinkability dictating that no Issuer or Verifier should be able to link VPs back to their Holders; cannot keep track of the use of attributes they issue and verify.* ■

The above definition is twofold: On one hand, it means that no adversary can extract any knowledge from a constructed VP signature (σ) that helps identify who presents σ and which credential is being used to construct σ, except for the level of identification that can be performed from the disclosed attributes and VC Issuers' public keys. On the other hand, unlinkability of the credentials and presentations is needed so that the Holder's actions cannot be tracked between Issuers, Verifiers, or even between Issuers and Verifiers. While the latter has also been highlighted as one core property for all EU Identity Wallets [15], existing cryptographic schemes, including the SD-JWT [16] and Mobile Security Object [17], specified in ISO 18013, all support only linkable signatures.

Definition 5: (Unforgeability) *It should not be possible for any adversary to construct a forgery VP/VC, based on "non-valid credentials", that will be accepted by a Verifier (V_i).*■

With an unforgeable anonymous credential, for an honest Issuer (at least one of either the DAA or VC Issuer) and a group of honest Holders, no adversary can create a valid signature (σ_D) on a claim that will be presented and accepted by a Verifier (V_i). Here, forgery should be non-trivial, that is, forgery should not be feasible when the Holder does not have access to the signing key (protected by the Device Binding property) nor when the Holder is not the intended recipient of the used credential based on which presented claims were disclosed (protected by the Holder Binding property).

Definition 6: (Wallet Correctness) *It must be ensured that only authenticated and non-compromised Wallets can access a Holder's unique identifier for creating attribute attestations.* ■

This definition ensures that a Verifier will accept a presented claim *if and only if* the Wallet can provide verifiable evidence that its integrity has not been altered (from the time of credential issuance) in an unauthenticated manner. This basically necessitates the enforcement of key restriction usage policies for governing the credential management, leveraging a TC's policy-based safeguards.

B. System Model

In this work, we present an enhanced version of DAA-A [14], in order to satisfy all of the above properties for decentralized identity wallets. Direct Anonymous Attestation (DAA) [13] is an anonymous signature scheme, which allows a Trusted Component (TC) to attest to the state of the host system while preserving the privacy of the Holder. While we do not build our solution around a specific type of TC, in our implementation we have leveraged the functionality of the TPM as the underlying root-of-trust for providing support on cryptographic operations and secure key storage. A DAA scheme consists of an Issuer (DAA Issuer in Figure 1) and a set of signers (Holders). It includes five algorithms: Setup, Join, Sign, Verify and Link. The DAA Issuer produces a DAA membership credential for each Holder, which corresponds to a signature on the Holder's unique identifier. This credential furthermore authorizes the use of the HW-based DAA Key which is stored inside the TPM and its usage is safeguarded through a number of policy regulations.

DAA-A construction [14] appeared later as a variant of DAA, with the difference that the public key does not correspond to a single secret key but is the result of a discrete logarithmic representation of multiple attributes. On one side, this feature enables us to build VPs with selective disclosure (Definitions 3) by encoding each attribute as a separate key, and on the other side it provides controlled anonymity (Definition 4) by allowing the representation of the identity as a separate attribute key to be hidden. The authenticity of the hidden attributes is proven by the integrated zero-knowledge (ZK) protocol.

Relying on these strong privacy guarantees, we build our DOOR protocol on top of DAA-A by adding extra layers of security; i.e., constructing policy regulations to govern the usage of the DAA Key (by the Holder) to sign attribute claims. The component responsible for the enforcement of these policies is the TC Bridge, which acts as the mediator between the Wallet and the underlying TPM. One such policy can ensure the binding of the DAA Key to the Holder's authenticated Wallet (Definition 2), which in turn enables the binding of the issued identity data to the Holder as the intended recipient (Definition 1). This is done by the VC Issuer through binding issued attributes to the anonymized part of the DAA credential. We also contract an additional policy to restrict the usage of the DAA Key, for creating attribute attestations, *if and only if* the Wallet integrity has not been altered in an unauthenticated manner (Definition 6).

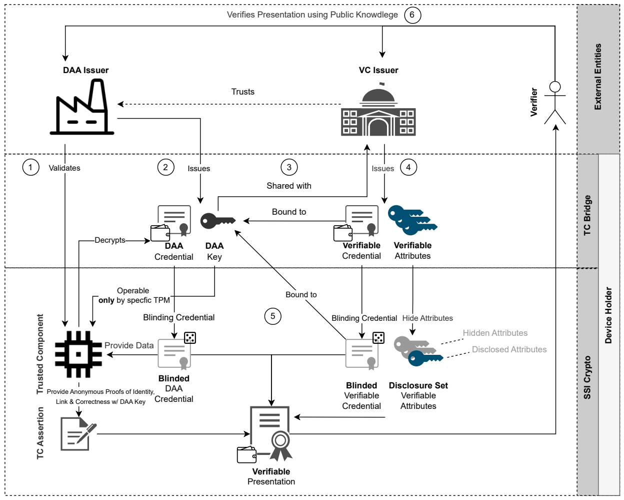

Fig. 1: DOOR High-level Architectural Overview and Credential Management Functionality

We also highlight that our notion of unforgeability (Definition 5) is stronger than what is required in existing VC management schemes [12] where only one Issuer is assumed in the system model: the VC Issuer can forge credentials and, hence, create forged signatures on presented claims. In our design, we have prompted to adopt the *separation-of-duties* principle where each Issuer is given the minimum amount of information required to execute its respective task; i.e., Device Binding and Key Restriction (DAA Issuer) and VC Issuance (VC Issuer), as detailed in Section II-A. Therefore, our unforgeability definition provides stronger guarantees that no single Issuer entity can forge signatures.

C. Related Work

The latest advancement in the area of Verifiable Credentials (VC) have their base in Anonymous Credential (AC) systems. The first practical approach of AC was from Camenisch and Lysyanskaya known as the CL-signatures [18] that use RSA groups and facilitate to efficiently do the proof of knowledge of a signature. They extended the work with CL-signatures from bilinear groups [3], which significantly improved on the efficiency of the scheme as it reduced the size of the keys. This was followed by a series of works related to AC schemes such as [19]–[21] that comes with different trade-offs related to efficiency, privacy and security.

AC schemes allow for the construction of efficient VCs that comes in different assertion formats, popular once being SD-JWT [22] and JSON-LD [23] using Linked Data (LD) Proofs [24]. The assertion formats could factor towards significant impact on the security and privacy of these AC schemes, so it is important to choose the right assertion format. The LDP-BBS+ [11] scheme is one of the most popular one in the community and under standardization efforts, e.g. in ETSI [25]. This scheme applies the BBS+ signature [26] to VC or VP based on JSON-LD to allow selective disclosure of attributes. However, BBS+ Signatures are not capable of predicates, which might be required for specific use cases, as well as it is hard to achieve selective disclosure with a VP that is based on VCs from different issuers. To address this, LDP-BBS+ [11] was enhanced by Yamamoto et al. [12] that allows to securely manage credentials using BBS+ signatures for achieving selective disclosure. However, relying on LD has an inherent limitation on the anonymity level due to the ordering of the attributes (based on the lexicographic order used by the canonicalization algorithm) which leaks information for the non-disclosed attributes. On the other side, the use of LD helps to link multiple credentials from different credential issuers and improve credential interoperability by enabling the credential to be anchored to a specific trust framework, e.g. Gaia-X [27].

The AC schemes described above cover the privacy aspects, but they don't consider the requirement of providing a high level of assurance that is required to develop a fully compliant digital wallet. So far, there has not been work that focuses on satisfying both requirements. There is a separate line of work that focuses on protecting cryptographic key material with hardware-based measures, however, this is mostly related to access control and not to protecting the wallet itself. For instance, Abraham et al. proposed a scheme to ensure user authentication to the Wallet, utilizing the secure element of the mobile phone, as well as a second key on a FIDO2 hardware token [7]. Moreover, Hanzlik and Slamanig presents a highly efficient core/helper anonymous credentials scheme (CHAC) [28] using a combination of signatures with flexible public keys (SFPK) and the novel notion of aggregatable attribute-based equivalence class signatures (AAEQ). However, this might not be the perfect solution in the context of digital wallets, since it would not support holder binding and is not efficient to run in devices with fewer computational resources.

To the best of our knowledge, DOOR is the first complete protocol capable of providing secure credential management with selective disclosure and with high LoA, thus, achieving all requirements listed in Section I-A. Our work can be integrated into the OpenID Connect for SSI specifications [29] as well as the technical Architecture and Reference Framework (AFR) for implementing the European Digital Identity [15].

II. Conceptual Protocol Overview

A. High-Level Overview

This section presents the high-level flows and functionalities of DOOR implemented to support the requirements formulated in Section I-A. The overall flow begins with a party wishing to acquire a credential, i.e., a virtual driver license. This party, the Holder, must first acquire the software necessary to handle credentials; the Wallet. The device the Wallet is installed on must have a Trusted Component such as a TPM (discrete or virtualized in a TEE) because of the following step. The application contacts the DAA Issuer, who will validate the Trusted Component and issue a DAA Credential and Key - a special cryptographic. The DAA Key can only be read and used internally inside this particular TC, therefore binding it to the device; we call this Device Binding. Furthermore, the key is only operable if certain negotiated conditions are met; i.e. the state of the device. We call these conditions Key Restriction Policies. Now the Holder can use the Wallet to acquire a Verifiable Credential (VC) from a VC Issuer, such as the DMV, school, government, etc. To do so, the Wallet shares the DAA Key with the VC Issuer, which verifies that it is genuine. If so (and assuming the Holder is verified through some authentic means) a VC is crafted and bound to this particular DAA Key, meaning it can only be used on the Holder's device. When the Holder wishes to present the credential, for example, to show that the holder's age is above 21, he uses the Wallet application, selects the credential, and decides what attributes to present, thereby only revealing the necessary attributes. The Wallet then constructs a presentation of the credential called a Verifiable Presentation (VP) using both the DAA Key and the VC. This can be shared with another party, the Verifier, who can verify the authenticity of the presented attributes using only the relevant VC Issuer and DAA Issuer public keys. Figure 1 shows the conceptual flows between the actors. Steps (1,2,3,4) concern Credential Management and (5,6) Attribute Authentication, all covered in a high-level manner in this section.

Device Binding and Key Restriction: To support Device Binding of VCs, we use a hardware-protected key. It takes on the role of a DAA key, built by the Holders' Trusted Component (TC). This guarantees that only this particular component can read and interact with the key, ensuring that cryptographic outcomes are unforgeable. Before creating the key, an exchange could take place between the key-certifying entity, i.e., the DAA Issuer ($\mathsf{I}_{\mathsf{DAA}}$), and the VC Issuer ($\mathsf{I}_{\mathsf{VC}}$) in order to negotiate a TC-enforced key restriction policy, that is requirements regarding when the key can be used. Upon agreement on requirements, $\mathsf{I}_{\mathsf{DAA}}$ develops a key restriction policy that captures the requirements and sends the policy to the TC Bridge. The TC builds the key and releases the public part of the key, containing the public key PK and integrity-protected information, such as the key restriction policy. This information is shared with $\mathsf{I}_{\mathsf{DAA}}$ to verify the key produced and validate the TC (§ 1). If both checks succeed, it releases a DAA Credential for the now certified key (§ 2).

Obtaining a VC: When requesting a VC, the TC Bridge shares the DAA public key with I_{VC} (§ 3) (and authenticates the Holder, along with a DAA signature to prove device ownership). I_{VC} constructs the relevant attributes for the VC and includes the DAA public key as an "identity attribute". Therefore, this binds the VC to the DAA key, resulting in the VC being bound to the physucal device. The credential is returned (§ 4) to the Holder and stored in the Wallet.

Generating a Verifiable Presentation: A VP is a representation of a subset of the attributes issued as part of the Verifiable Credential. It can be verified by any Verifier knowing the respective VC and DAA credentials. To protect against VP linkability, we are contracting a "blinding" or "randomization" operation to the credentials. This process does not negate the Device Binding property (Definition 2), but ensures Full Anonymity and Unlinkability (Definition 4) against both the VC Issuer (not keep track of the use of the attributes they have issued) and the Verifier (conducting attribute validity) (cf.

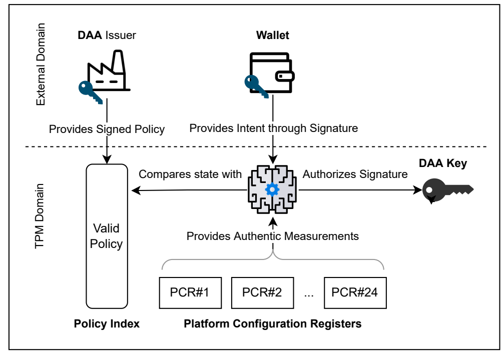

Fig. 2: DAA signing key usage requires both authentication tokens from Wallet, and internal PCRs to be in a trusted state.

Section III). Following this blinding, the Holder then decides which attributes to disclose. All non-disclosed attributes are hidden by using a cryptographic operation that attests to their equivalence in the original VC, thus, providing the necessary proof that the VP has been correctly computed by the Holder which owns the Wallet of credentials (creates a bound signature associated to the attribute values). The "identity attribute" (the DAA private key) will always be hidden by the TC, to ensure anonymity. The final presentation blob is now constructed as a set of disclosed and non-disclosed attributes accompanied by the two blinded credentials and can be shared with any Verifier. To verify that both hidden and revealed attributes were certified in the VC, the Verifier can use the mechanisms provided by the DAA-A scheme to confirm attribute validity and that the Holder also controls the rest of the (undisclosed) attributes. Furthermore, she can also assert that the correct key restriction usage policy has been implemented to ensure Wallet Correctness (Definition 6).

B. *Principles of Secure Wallet Construction*

It should now be evident that DOOR protects the HW-based DAA key with a set of policies for ensuring Wallet Correctness and verifying that only the authenticated (intended) Wallet can interact with the TC through the TC-Bridge software stack. A Verifier trusts the DAA Issuer to validate the DAA key and the TPM, which implies trust in any assertions produced by that key. An asymmetric key pair created by the TPM is de facto bound to it. Keys are created as Primary Keys (re-creatable under a secret internal unique seed) or as Childre Keys of a Primary key (encrypted by the parent). As private keys *never* leave the TPM, the produced keys are restricted to the physical chip. TPMs also provide a policy-enforcing functionality that can bind an integrity-preserved policy to a key, thereby restricting the use until the policy is satisfied; we call this a key restriction policy. To satisfy a policy, one or more policy commands [30] are executed on the TPM, each providing distinctive evidence. Therefore, if a policy-protected TPM key provides a signature, the Verifier can be sure that the policy has been satisfied. In the rest of the section, we cover how the TPM is used to protect the DAA key from untrusted device configurations and unauthorized Holders using policies.

Restrict to Trusted Configuration The TPM includes a set of internal *extendable* registers called Platform Configuration

Registers (PCRs). These store measurements of the residing platform (the Holder device) as chained hashes originating from a root of trust for measurements (e.g., CPU microcode, TEEs, or similar). We can build a policy that can be satisfied, if a selection of PCRs matches a predetermined value, referencing a trusted state. Using `PolicyPCR`, we ensure that the DAA key is inoperable, if the integrity of the device state is compromised. Since policies are immutable, updating the reference values (the policy itself) will require a new key with an updated policy to be issued. To avoid this, the DAA key is instead bound to the contents of an internal non-volatile register through a policy called `PolicyAuthorizeNV` (Algorithm **??**). Any policy (e.g. `PolicyPCR`) residing in the register, the Policy Index (PI), must be satisfied before the TPM allows the use of the key. To protect the PI from malevolent changes, it is itself protected by a policy (Algorithm **??**). This policy (`PolicySigned`) requires that to write a policy to the PI, $\mathsf{I}_{\mathsf{DAA}}$ must sign it. This policy also protects the deletion of the index, to prevent recreation, and is resistant to replay attacks by including a TPM session-nonce in the authorization (Algorithm **??**).

$Wallet\ (PK)$		$\mathsf{I}_{\mathsf{DAA}}\ ((x, y), \rho)$
	$\xrightarrow{request}$	
	$\xleftarrow{\rho}$	$\rho \leftarrow \{0,1\}^{\lambda}$
TPM Calculate $PK = x_0 G_0$ ①		
	$\xrightarrow{(\rho, PK)}$	
		$\forall\ x_0{}^* \in$ `KRL` , abort if $PK = x_0^* G_0$ $r \in_R \mathbb{Z}_q$ $A = rG_0, B = yA$ $C = xA + rxyPK$ $D = ryPK$ ② $\omega \in_R \mathbb{Z}_q$; $\hat{c} = H_1\Big(\omega G_0 \vert \omega PK \vert \rho\Big)$ ③ $\hat{s} = \omega - \hat{c}ry$
	$\xleftarrow{(A,B,C,D,\hat{c},\hat{s})}$	
$\tau(A, \tilde{Y}) \stackrel{?}{=} \tau(B, \tilde{G_0})$ ④ $\tau(A + D, \tilde{X}) \stackrel{?}{=} \tau(C, \tilde{G_0})$ $\hat{c} \stackrel{?}{=} H_1\Big(\hat{c}B + \hat{s}G_0 \vert \hat{c}D + \hat{s}PK \vert \rho\Big)$ ⑤		

Fig. 3: The Join Protocol with $\mathsf{I}_{\mathsf{DAA}}$. Note: x_0 is private to TPM

Key Ownership and Usage As the TPM is not uniquely accessible from the TC Bridge, the Verifier cannot determine who ordered the signature. For instance, if a trusted application was misused to produce signatures. To appoint ownership of the DAA key, we again use the TPM policy functionality. Policies are not restricted to singular commands, but can be built as multiple policies in an order-restrictive format; hence, we can add more policies to the already described `PolicyPCR`. The wallet comes with a preinstalled key, the "Wallet key" (WK), which we can use in `PolicySigned`, as we did to protect the PI (require a signature to authorize operation). Any trusted applications cannot access the WK, and untrusted applications are locked out by `PolicyPCR`. A signature of the DAA key provides the Verifier with evidence of intention and Holder correctness, as seen in Figure 2. However, a DAA signature requires an additional cryptographic operation, namely the `Commit` operation. As this operation is not used to provide evidence to a Verifier, it should not be restricted. Furthermore, since the DAA key is used to certify the PI using `NV_Certify`, this should be allowed without using resources on the other policies. TPMs provide the functionality to allow multiple policies to be valid, using a special policy command `PolicyOR`. This allows the key to be operable if the sign policy is satisfied (proof of intent and correctness) *or* if the next command is authorized (Commit and CeritfyNV). Both can be allowed using individual policies using the policy command `PolicyCommandCode`. This can be added to one large policy, which is authorized by $\mathsf{I}_{\mathsf{DAA}}$. The final result is a device-bound key.

III. Architectural Details & Protocols

Notation Let $\mathbb{F}$ be a finite base field and $\tilde{\mathbb{F}}$ be a finite extension field of $\mathbb{F}$. Let $\mathbb{E}$ be an elliptic curve defined over $\mathbb{F}$ with a base point G_0. Let $\tilde{\mathbb{E}}$ denote the points of $\mathbb{E}$ over the extension field $\tilde{\mathbb{F}}$ and $\tilde{G_0}$ be a base point of $\tilde{\mathbb{E}}$. $\tilde{G_0}$ is used to generate the Issuers' public keys in $\tilde{\mathbb{E}}$, whereas G_0 is used to generate the public part of the TPM's DAA Key in $\mathbb{E}$. The curve $\mathbb{E}$ is equipped with a type III pairing $\tau : \mathbb{E} \times \tilde{\mathbb{E}} \to \tilde{\mathbb{F}}$. τ is used to verify the DAA and VC credentials under the Issuers' public keys. The operation on $\mathbb{E}$ (resp. $\tilde{\mathbb{E}}$) is written with additive notation. Multiplication by scalars is always written on the left. Scalars are always defined on $\mathbb{Z}_q$ (from where the secret keys are sampled), where q is a prime number that represents the order of the subgroup $\langle G_0 \rangle$ in $\mathbb{E}$. Arithmetic has to be understood in the respective finite fields. Uppercase Latin or Greek letters always indicate EC points on the curve $\mathbb{E}$. Uppercase Latin or Greek letters with a tilde on top will denote elements on the curve $\tilde{\mathbb{E}}$.

Setup: The public group elements $G, G_1, \ldots, G_n \in \mathbb{E}$ and $\tilde{G}, \tilde{G}_1, \ldots$
$, \tilde{G}_n \in \tilde{\mathbb{E}}$ are generated from G_0 and $\tilde{G_0}$ respectively, where $G = r_G G_0$, $G_k = r_k G_0$, $\tilde{G} = r_G \tilde{G_0}$ and $\tilde{G}_k = r_k \tilde{G_0}$ for $k = 1 \ldots n$ and $r_G, r_k \in_R \mathbb{Z}_q$, where $\in_R$ indicates that the elements are chosen randomly. $G_1, \ldots, G_n$ will be used in our scheme to generate the attribute tokens in $\mathbb{E}$. $\tilde{G}, \tilde{G}_1, \ldots, \tilde{G}_n$ are used in the verification phase in the batch proof trick presented later in the section. It is required that the values of r_G and r_k for $k = 1 \ldots n$ are generated by the setup system and erased after the setup process, such that there is no known discrete logarithm relation between any G_k and G_j (for some $j \neq k$) and between any G_k and G. The hash function: $\mathsf{H}_1 : \{0,1\}^* \to \mathbb{Z}_q$ is used in our scheme to output the challenge c used when creating Schnorr Non-interactive Zero-Knowledge Proofs. $\mathsf{I}_{\mathsf{DAA}}$'s signing secret key consists of two integers $x, y \in \mathbb{Z}_q$. $\tilde{X} = x\tilde{G_0}$ and $\tilde{Y} = y\tilde{G_0}$ correspond to $\mathsf{I}_{\mathsf{DAA}}$'s public key. Let $u, v \in \mathbb{Z}_q$ be the VC Issuer's private key. $\tilde{U} = u\tilde{G_0}$ and $\tilde{V} = v\tilde{G_0}$ correspond to the VC Issuer's public key. Let us highlight that $\mathsf{I}_{\mathsf{DAA}}$ creates a proof of knowledge π_{ipk}^{DAA} to prove that the relation between

$Wallet\ (PK)$ | $\mathsf{I}_{\mathsf{VC}}\ ((u,v),\rho)$

$\xrightarrow{request_{VC}(PK)}$

$\xleftarrow{Key_w=\{x_1,x_2,\ldots,x_n\},\ n\in_R Z}$ (1)

TPM Calculate σ^{DAA} on (Key_W, n) (2)

$\xrightarrow{\sigma^{DAA}}$

Verifies σ^{DAA} on (Key_W, n)
$\Gamma = PK + \sum_{k=1}^{k=n} x_k G_k$ (3)
$t \in_R \mathbb{Z}_q$ (4)
$A_w = tG, B_w = vA_w$
$C_w = uA_w + tuv\Gamma$
$D_w = tv\Gamma$ (5)
$E_{W_k} = tvG_k\ \forall\ k \in [0,n]$ (6)
$\gamma \in_R \mathbb{Z}_q$
$\hat{c}_w = H_1\Big(\gamma G|\gamma G_0|\ldots|\gamma G_n|\gamma\Gamma|\rho\Big)$
$\hat{s}_w = \gamma - \hat{c}_w tv$ (7)

$\xleftarrow{(A_w, B_w, C_w, D_w, E_{W_k}, \hat{c}_w, \hat{s}_w, \rho)}$

$\tau(A_w, \tilde{V}) \stackrel{?}{=} \tau(B_w, \tilde{G_0})$
$\tau(A_w + D_w, \tilde{U}) \stackrel{?}{=} \tau(C_w, \tilde{G_0})$ (8)
$\hat{c'}_w = H_1\Big(\hat{c}_w B_w + \hat{s}_w G|\hat{c}_w E_{W_0} + \hat{s_w} G_0|\hat{c}_w E_{W_1} + \hat{s_w} G_1|\ldots|\hat{c}_w E_{W_n} + \hat{s}_w G_n|\hat{c}_w D_w + \hat{s}_w \Gamma|\rho\Big)$
$\hat{c'}_w \stackrel{?}{=} \hat{c}_w$ (9)
$cre = (cre_{PK}, cre_w)$

Fig. 4: The Join Protocol with I_{VC} (Issue VC)

(x, y) and $(\tilde{X}, \tilde{Y})$ is well established (i.e. $\tilde{X} = x\tilde{G_0}$ and $\tilde{Y} = y\tilde{G_0}$). This proof correctly binds the public key $(\tilde{X}, \tilde{Y})$ to its corresponding secret key (x, y). This step is crucial for the correctness of the protocol, which states that honestly generated signatures should successfully verify, hence verified under the same Issuer's public key who initially created the user credential. Similarly, I_{VC} proves that his key is well-formed by providing a proof of knowledge π_{ipk}^{VC} of (u, v).

A. Credential Management

In our architecture, VCs are enhanced with hardware-based keys issued from a Trusted Component (e.g., TPM). This key (DAA Key) must be accompanied by a credential (DAA Credential) certifying its properties and enabling the key to work. All VCs issued are, therefore, bound to a particular DAA Key, hence the creation and certification of this must happen prior to the issuance of a VC. Each issuer is assumed to have an authentic copy of the TPM's endorsement key, which is used to establish a secure and authenticated channel between the TPM and the issuer. In the join protocol description, it is assumed the existence of a secure authentication channel between the TPM and the DAA/ VC Issuer, the reader is recommended to find the detail regarding how to establish such a channel from [31].

Issue DAA Credential: Before establishing contact with $\mathsf{I}_{\mathsf{DAA}}$, the TC Bridge configures the TPM to enable safe storage of the Issuer-generated key restriction policy. It does so by creating the TPM-protected Policy Index (PI) with safety mechanisms that only allow the $\mathsf{I}_{\mathsf{DAA}}$ Issuer to modify it (Figure **??**). The TC Bridge computes a policy for the upcoming DAA key that makes the key usable only if the policy stored in the PI can be satisfied. To do so, it acquires the unique index name $\mathcal{N}$ and calculates the policy digest according to the TPM standard. With the newly created policy, the TC Bridge sends it to the TPM with instructions to generate a new DAA Key. Internally, the TPM chooses the secret DAA Key $x_0 \leftarrow \mathbb{Z}_q$ and sets its public key $PK = x_0 G_0$ (1). It returns the cryptographic PK, alongside other parameters (i.e., policy) in an integrity-protected data structure.

An authorization session is started with the TPM, returning a nonce n to the TC Bridge. A registration package can now be assembled, consisting of the DAA Key data structure, nonce n, index name $\mathcal{N}$, the public TPM Endorsement Key (EK), and the Wallets' software-based public key WK and then sent to $\mathsf{I}_{\mathsf{DAA}}$.

$\mathsf{I}_{\mathsf{DAA}}$ verifies that the DAA key policy ensures the contents of the PI are satisfied as a policy, and computes the key-restriction policy to be written to the PI. This policy, $\mathcal{K}$, can only be satisfied by proof of intent from the WK and if the integrity of the Wallet is not compromised. It then computes the write-authorization by computing $a_c = H(n|0|c_c|00_{16})$, where $c_c = H(\texttt{CC\_NV\_Write}|\mathcal{N}|\mathcal{N}|\mathcal{K})$ and signs it with $\mathsf{I}_{\mathsf{DAA}}$ private key to produce σ_a. This authorization allows $\mathcal{K}$ to be written to an index with the name $\mathcal{N}$ in a session with the nonce n.

$\mathsf{I}_{\mathsf{DAA}}$ then creates a challenge using the `make_credential` functionality of the DAA scheme. The challenge, authorization, and policy are returned to the TC Bridge, which satisfies the PI policy using the provided authorization to write the policy $\mathcal{K}$, authorized by $\mathsf{I}_{\mathsf{DAA}}$, enabling operations of the key. By using the `activate_credential` functionality, the TPM computes the challenge response. It then computes $\pi^{\mathcal{M}}$ as proof of construction of PK. To provide evidence of the creation and contents of the PI, the TPM provides an Index Certificate signed by the DAA Key.

The resulting certificate and the challenge response are sent to $\mathsf{I}_{\mathsf{DAA}}$, who first verifies $\pi^{\mathcal{M}}$ to check whether the TPM Wallet is eligible to join, i.e., the DAA Key has not been previously certified. If this validation succeeds, $\mathsf{I}_{\mathsf{DAA}}$ verifies that the current contents of the PI match the previously computed policy and that writing and deleting the PI requires $\mathsf{I}_{\mathsf{DAA}}$ authorization. If this verification succeeds, it will now compute the credential (Figure 3). A random $r \in_R \mathbb{Z}_q$ is chosen and used to calculate the four points (A, B, C, D) (2).

To provide authenticity, $\mathsf{I}_{\mathsf{DAA}}$ performs a Schnorr ZK proof written as $(\hat{c}, \hat{s})$, which shows that the discrete logarithms are equivalent. To do this, $\mathsf{I}_{\mathsf{DAA}}$ chooses a random $\omega \in_R \mathbb{Z}_q$; and calculates the challenge $\hat{c}$ and signature $\hat{s}$ (3), where ρ is a message for freshness agreed by $\mathsf{I}_{\mathsf{DAA}}$, the VC Issuer and the signer.

$\mathsf{I}_{\mathsf{DAA}}$ sends the PK-credential $cre_{PK} : (A, B, C, D, \hat{c}, \hat{s})$ to the Wallet (TC Bridge), which represents the credential that corresponds to the Holder with embedded TPM with Public DAA Key $PK = x_0 G_0$. Upon receiving cre_{PK}, the Wallet verifies the credential under $\mathsf{I}_{\mathsf{DAA}}$'s public keys $\tilde{X}$ and $\tilde{Y}$ by

$TPM\ (x_0)$		$Wallet\ (PK, Key_W)$
		$a \in_R \mathbb{Z}_q$ ① $A' = aA, B' = aB, C' = aC\ D' = aD$ $A'_w = aA_w; B'_w = aB_w$ $C'_w = aC_w, D'_w = aD_w$ $E'_{W_k} = aE_{W_k}\ \forall\ k \in [0, n]$ ②
	$\xleftarrow{\texttt{TPM2\_Commit}(B' + E'_{W_0})③}$	
$\omega_0 \in_R \mathbb{Z}_q$ $R_0 = \omega_0(B' + E'_{W_0})$ ④		
	$\xrightarrow{R_0}$	
		$\{\omega_1, \ldots, \omega_p\} \in_R \mathbb{Z}_q$ $R_{W_k} = \omega_k E'_{W_k}\ \forall\ k \in \mathcal{P}$ ⑤ $c = H_1(A'\|B'\|C'\|D'\|A'_w\|B'_w\|C'_w\|D'_w\|E'_{W_0}\|E'_{W_1}\|\ldots\|E'_{W_n}\|R_0 + \sum_{k\in\mathcal{P}} R_{W_k}\|m')$ ⑥
	$\xleftarrow{\texttt{TPM2\_Sign}(c)}$	
$s_0 = \omega_0 + cx_0$ ⑦	$\xrightarrow{s_0}$	
		$s_k = \omega_k + cx_k\ \forall\ k \in \mathcal{P}$ $\sigma = (A', B', C', D', A'_w, B'_w C'_w, D'_w, E'_{W_0}, \ldots, E'_{W_n}, s_k, s_0, c)$

Fig. 5: Creating Verifiable Presentations

checking the pairings ④ and verify the discrete logarithm equivalence via $(\hat{c}, \hat{s})$ ⑤. If the above verification passes are successful, the Wallet stores cre_{PK} and can now acquire VCs. Note that this DAA Key and credential is usable with multiple VC Issuers as long as they assert the authorized policy from the DAA Issuer.

Issue Verifiable Credential: To obtain a Verifiable Credential (Figure 4), the Wallet sends a request to $\mathsf{I_{VC}}$ to issue a credential for a set of attribute keys (e.g. all attributes for a drivers license). This request contains the DAA Key PK and any other authenticating information. $\mathsf{I_{VC}}$ authenticates the Wallet and defines the Holders' attribute space U, covering the set of attributes $x_1, \ldots x_n$ that correspond to the attributes related to the Holder, now identified by the DAA Key (PK).

$\mathsf{I_{VC}}$ sets the Wallet attribute keys (attributes) and sends it to the TPM Wallet with a nonce n ①. Recall that an attribute key is just an encoded attribute using the hashing function H_1. The Wallet stores the attribute keys $(x_1, \ldots, x_n)$ along with cre_{PK} and must now prove to $\mathsf{I_{VC}}$ that it controls the DAA Key provided. To do so, the TC Bridge satisfies both the commit and sign policy and uses the TPM to generate a DAA signature σ^{DAA} ②, using the standard DAA signature scheme, and sends it to $\mathsf{I_{VC}}$. $\mathsf{I_{VC}}$ verifies the signature; If the verification passes, $\mathsf{I_{VC}}$ generates a verifiable credential cre_w, which contains a signature on all the attribute keys by performing the following steps. $\mathsf{I_{VC}}$ calculates Γ ② which now represents the Holders' public key for the attributes, later used for verification. It then chooses a random value ③ and calculates the points A_w, B_w, C_w, D_w ⑤. Following this the values E_w can be calculated as shown in ⑥. These values demonstrate that key x_k that was used to provide signature s_k, indeed was a certified attribute key. $\mathsf{I_{VC}}$ then chooses a random value and calculates the challenge $\hat{c}_w$ and $\hat{s}_w$ ⑦, where ρ is a message of freshness agreed by $\mathsf{I_{VC}}$, $\mathsf{I_{DAA}}$ and the Holder, just as the credential for the DAA key.

$\mathsf{I_{VC}}$ sends the credential $cre_w = (A_w, B_w, C_w, D_w, E_{W_k}, \hat{c}_w, \hat{s}_w)$ back to the Wallet. Upon receiving cre_w, the Wallet verifies the signatures. First, it check the pairings ⑧ and it this check is successful, it then validates the Schnorr signature ⑨. If this check also succeeds, the credential (VC) is stored in the wallet.

Storing Both types of credentials can be stored and managed by an SSI Wallet, but to use the credentials, the Wallet must be extended with our TC Bridge. Even if credentials are lost due to theft or data leaks, this does not raise any concern regarding the misuse of credentials. Since a VC depends on a unique hardware key and this hardware key can *only* be used by a particular TPM, stolen credentials cannot provide any true verifiable presentation.

B. Attribute Authentication

To verify a set of attributes from a credential, the Wallet uses the TC Bridge to compute a VP. With our architecture, the TC Bridge can only create a VP on the device to which the corresponding credential was issued, with a proven intent of the requesting Wallet. Additionally, due to the extended hardware support, the Trusted Component will only provide the necessary assertions if the integrity of the Wallet and TC bridge is not compromised.

Creating Verifiable Presentations: The TPM checks if the policy is satisfied, i.e. the device is in the correct state. If so, the TPM creates a DAA signature σ_{DAA} using its DAA credential cre_{PK} and its corresponding DAA key. Using the basename $\mathsf{bsn} = \perp$ the signature becomes unlinkable. The TC Bridge creates a Proof of Knowledge (DAA Signature) that, as depicted in Figure 5, enables the Holder to present a valid credential for the attribute key $Key_w = (x_1, \ldots, x_n)$ and such that the overall DAA signature is only verified under a certified public key of the device $\Gamma = PK + \sum_{k=1}^{n} x_k G_k$, where PK is a certified TPM key from $\mathsf{I}_{\mathsf{DAA}}$. The flow of this is as follows.

1) **Blind**: The TC Bridge creates a random number and uses it as a blinding factor ①. Then it blinds both the DAA- and Verifiable Credential by multiplying the blinding factor upon the eight respective points and E-values ②, this step is crucial for unlinkability.
2) **TPM Commit**: The TC Bridge commits the B point from the DAA Credential and the first E-point from the VC using the TPM ③. At this point the TPM choses a random value ω_0 and multiplies that upon the committed value ④, this random value is safely stored in the TPM, to be used later.
3) **TC Bridge Commit**: The next step is to commit all the attributes we *do not* wish to disclose, within the host. Let $\mathcal{D}$ be the set of indices of the disclosed attributes needed for a specific service, and let $\mathcal{P} = \{1, \ldots, n\} \setminus \mathcal{D}$ represent the set of indices of all other committed (hidden) attributes. We can represent $\mathcal{P}$ by the set $\{1, \ldots p\}$ that denotes the indices of the committed attributes with $p \leq n$. For each committed attributes, the Wallet selects a random value and multiply it upon the respective E-points ⑤. The random values are stored within the host.
4) **TPM Sign**: The TC Bridge calculates the hash value to be signed, c, ⑥ and satisfies the signing policy and executes `TPM2_Sign` to sign c. TPM then signs the hash using the same ω_0 as used in the TPM Commit phase and the DAA private key ⑦.
5) **TC Bridge Sign**: The Wallet signs each of the committed attributes using the respective ω and outputs $s_k = \omega_k + cx_k \ \forall \ k \in \mathcal{P}$, using the attribute keys. The Wallet sends $\sigma_D = (A', B', C', D', A'_w, B'_w, C'_w, D'_w, E'_{W_0}, \ldots, E'_{W_n}, s_0, s_{k\in\mathcal{P}}, x_{k\in\mathcal{D}}, c)$ to the verifier.

Verification: The verifier checks the attributes and verifies the DAA signature as follows.

1) Verify the modified CL certificate by checking the pairings on both the blinded DAA- and Verifiable Credential: $\tau(A', \tilde{Y}) \stackrel{?}{=} \tau(B', \tilde{G}_0)$ and $\tau(A' + D', \tilde{X}) \stackrel{?}{=} \tau(C', \tilde{G}_0)$. $\tau(A'_w, \tilde{V}) \stackrel{?}{=} \tau(B'_w, \tilde{G}_0)$ and $\tau(A'_w + D'_w, \tilde{U}) \stackrel{?}{=} \tau(C'_w, \tilde{G}_0)$.
2) Verify the equivalence of the discrete logarithm using the batch proof trick from [13]: $t_0, t_1, \ldots, t_n \in \mathbb{Z}$;
$$\tau(t_0 E'_{W_0} + \ldots + t_n E'_{W_n}, \tilde{G}) \stackrel{?}{=} \tau(B'_w, t_0\tilde{G}_0 + \ldots + t_n\tilde{G}_n)$$
3) Verify the Schnorr ZK proof of knowledge of the hidden attributes:
$\mu_W = \sum_{k\in\mathcal{P}} s_k E'_{W_k} + s_0(B' + E'_{W_0}) - c(D' + D'_w - \sum_{k\in\mathcal{D}} x_k E'_{W_k})$
$c \stackrel{?}{=} H_1(A'|B'|C'|D'|A'_w|B'_w|C'_w|D'_w|E'_{W_0}|\ldots|E'_{W_n}|\mu_W|m')$
4) Outputs Valid if all checks and verification pass.

As in [12], our protocol can offer linkability of the committed attributes even when are not issued by the same VC Issuer. This is done by the TC Bridge by adding attribute link tokens in the form of $J_k = x_k H_2(bsn_k)$ for each committed attribute x_k for some verifier's input bsn_k and $H_2 : \{0,1\}^* \rightarrow \mathbb{E}$. If any two signatures that are signed under the same bsn_k contain the same J_k, then the verifier is convinced that the signers share a common attribute x_k without learning anything about x_k.

IV. Performance Evaluation

This section presents a performance analysis and evaluation of DOOR's capabilities on creating and verifying VPs, as these are the more time-sensitive operations. Offline commands, such as initial setup and initialization are not presented here but have been analyzed as well. We implemented DOOR in the C language and timed each component using real-time measurements for both the TC as well as the Holder device, showcased with a TPM and Raspberry Pi, respectively.

Experimental Setup & TPM Timings The physical setup consists of a Raspberry Pi 4 Model B with an Infineon SLI Iridium 9670 TPM. The tests executed consists of three scenarios: 1) No attributes are disclosed, 2) Half of all attributes are disclosed, and 3) All attributes are exposed. For each of these three scenarios, we executed one thousand tests with 8, 16, 32, 64, and 128 attributes, per protocol, and stored the average time. In Table I we show the time it takes to complete the TPM commands concerning enforcing the previously described trust requirements. In case of no trust requirements, this can be reduced to `TPM2_Commit` and `TPM2_Sign` which is summarized to 298.25 ms, leaving the overhead for our complete trust assurance at only 1026.11 ms.

TABLE I: Create Verifiable Presentation - TPM timings

Command	Mean	± (95% CI)
TPM2_StartAuthSession	52.48 ms	2.67 ms
TPM2_PolicyCommandCode	1.51 ms	0.03 ms
TPM2_PolicyOR	3.11 ms	0.09 ms
TPM2_PolicyAuthorizeNV	326.28 ms	7.41 ms
TPM2_Commit	176.63 ms	3.23 ms
TPM2_Hash	119.27 ms	9.41 ms
TPM2_StartAuthSession	51.32 ms	2.64 ms
TPM2_PolicyPCR	2.57 ms	0.07 ms
TPM2_PolicySigned	141.02 ms	2.06 ms
TPM2_PolicyOR	3.18 ms	0.10 ms
TPM2_PolicyAuthorizeNV	325.37 ms	7.74 ms
TPM2_Sign	121.62 ms	9.35 ms
Total TPM Time	1324.36 ms	44.80 ms

Performance Comparison It is obvious to compare DOOR with BBS+ as they are similar by both providing selective disclosure. However, while similar, DOOR provides additional security with key-restricting policies, making DOOR directly incomparable to BBS+. To fairly compare the protocols, we stripped DOOR for all policies, only requiring the TPM to provide Commit- and Sign capabilities, as described above. To achieve comparable performances, we used a BBS+ library [32] written in Rust. For anonymity, we tested against BLS signatures, as they are prominently used in Wallets as well. The results showed that BLS signatures are, as expected, much faster than DAA with TPMs, but they also do not provide controlled linkability features, as DAA does.

Creating Verifiable Presentations In Figure 6, we see how BBS+ are much more affected by an increase in attributes than DOOR. It's also notable how BBS+ VP generation is more affected by the number of hidden attributes - the less you disclose, the longer time it takes. While DOOR is also affected by this, the difference in the worst case (128 attributes) between full- and no disclosure is $55.72ms$, an

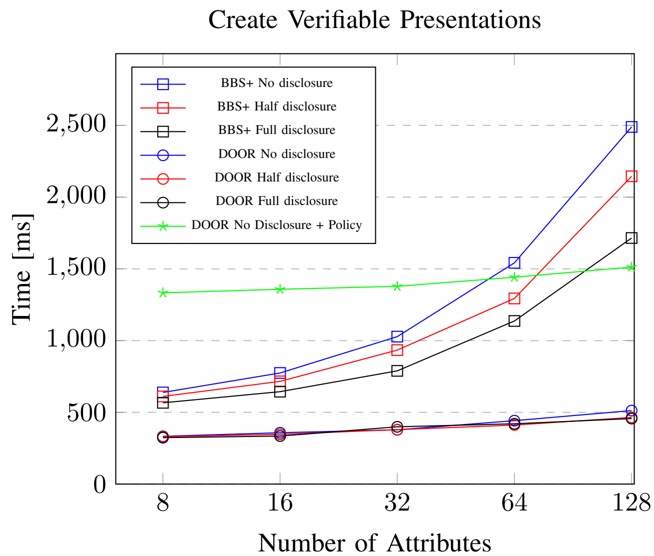

Fig. 6: Comparison between BBS+ and DOOR for creating Verifiable Presentations. Note that the X-axis is of a logarithmic nature.

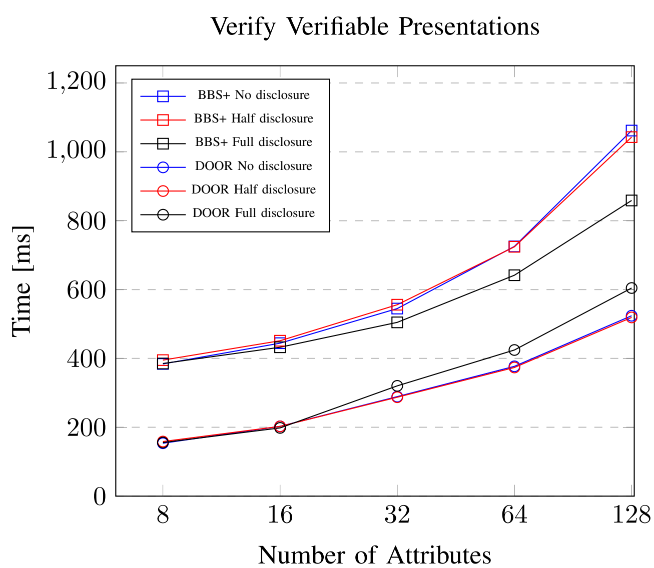

Fig. 7: Comparison between BBS+ and DOOR for verifying Verifiable Presentations. Note that the X-axis is of a logarithmic nature.

increase of $\ 12\%$, compared to BBS+ $774.17ms$, an increase of $\ 45\%$. However, if we add the policies described in this paper to DOOR, we will add around 1000 ms to the time it takes to create a VP (shown with the star-marked line). In this case, BBS+ is significantly faster for low-count attributes, however, BBS+ does not provide comparable trust assurances in this case.

Verifying Verifiable Presentations In verification, seen in Figure 7, we see how DOOR is generally faster than BBS+. This operation is indifferent whether policies have been used or not, making this operation easily comparable. Both algorithms generally increase similarly in time, as the number of attributes increases, and overall DOOR is double as fast as BBS+. However, it's interestingly noteworthy that BBS+ and DOOR are mirrored s.t. DOOR is generally faster in verifying Half- and No Disclosure whereas this is the slowest case for BBS+.

V. Conclusion

DOOR offers higher levels of confidence to the authentication and electronic identification service of digital identity Wallets - hence a higher LoA. It also enables construction of Verifiable Presentations that selectively disclose only those attributes needed for verification, ensuring at the same time that anonymity and unlinkability is preserved. The implementation and evaluation of the performance of DOOR showed the effectiveness of the design of our protocol. As future work we plan to add the functionality of constructing one VP for combining attributes not only based on bound and public credentials but also multiple credentials issued from different Issuers. This could open up new application scenarios.

References

[1] W3C Recommendation, "Decentralized Identifiers (DIDs) v1.0 Core architecture, data model, and representations," 2021. https://www.w3.org/TR/did-core/.

[2] M. Sporny, D. Longley, and D. Chadwick, "Verifiable Credentials Data Model 1.1." https://www.w3.org/TR/vc-data-model/, 2022.

[3] J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *Advances in Cryptology*, 2004.

[4] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Communications of the ACM*, vol. 28, no. 10, 1985.

[5] J. Camenisch et. al., "D2.1 Architecture for Attribute-based Credential Technologies - Version 1," tech. rep., ABC4Trust Deliverable, 2011.

[6] H. B. Debes and T. Giannetsos, "Segregating keys from noncense: Timely exfil of ephemeral keys from embedded systems," in *Int. Conf. on Distributed Computing in Sensor Systems (DCOSS)*, pp. 92–101, 2021.

[7] A. Abraham et al., "SSI Strong Authentication using a Mobile-Phone based Identity Wallet reaching a High Level of Assurance," in *SECRYPT*, 2021.

[8] E. Union, "Council regulation (EU) no 1502/2015," 2015.

[9] Trusted Computing Group, "Trusted Computing Platform Alliance $TCPA$ main specification," 2003.

[10] D. I. Foundation, "Device binding work item," 2023.

[11] T. Looker and O. Steele, "BBS+ Signatures 2020," 2022. https://w3c-ccg.github.io/ldp-bbs2020/.

[12] D. Yamamoto, Y. Suga, and K. Sako, "Formalising linked-data based verifiable credentials for selective disclosure," in *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 52–65, IEEE, 2022.

[13] L. Chen, "A daa scheme using batch proof and verification.," *TRUST*, vol. 10, pp. 166–180, 2010.

[14] L. Chen and R. Urian, "DAA-A: Direct anonymous attestation with attributes," in *Int. Conf. on Trust and Trustworthy Computing*, pp. 228–245, Springer, 2015.

[15] E. Commission, "The European Digital Identity Wallet Architecture and Reference Framework," tech. rep., 2013.

[16] IETF, "Selective Disclosure for JWTs (SD-JWT)," tech. rep., 2022.

[17] S. T. Alliance, "The Mobile Driver's License (mDL) and Ecosystem," tech. rep., 2020.

[18] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *Advances in Cryptology—EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques Innsbruck, Austria, May 6–10, 2001 Proceedings 20*, pp. 93–118, Springer, 2001.

[19] O. Sanders, "Efficient redactable signature and application to anonymous credentials," in *Public-Key Cryptography–PKC 2020: 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4–7, 2020, Proceedings, Part II*, pp. 628–656, Springer, 2020.

[20] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya, "P-signatures and noninteractive anonymous credentials," in *Theory of Cryptography: Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008. Proceedings 5*, pp. 356–374, Springer, 2008.

[21] D. Boneh and X. Boyen, "Short signatures without random oracles," in *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*, pp. 56–73, Springer, 2004.

[22] IETF, "SD-JWT," 2023. https://www.ietf.org/archive/id/draft-fett-oauth-selective-disclosure-jwt-02.html.

[23] G. Kellogg, D. Longley, and P.-A. Champin, "JSON-LD 1.1. A JSON-based Serialization for Linked Data (W3C Recommendation)," July 2020. https://www.w3.org/TR/json-ld11/ [Online].

[24] C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data - The Story So Far," *Int. Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 5, no. 3, pp. 1–22, 2009.

[25] ETSI, "ETSI GR CIM 018 V1.1.1 Enabling chain of trust from the content sources to content consumers," 2022.

[26] J. Camenisch, M. Drijvers, and A. Lehmann, "Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited," in *Trusted Computing*, pp. 1–20, 2016.

[27] Gaia-X, "Gaia-X Compliance Service," 2023. https://gaia-x.gitlab.io/policy-rules-committee/trust-framework/gaia-x$_{t}rust_{f}ramework/$.

[28] L. Hanzlik and D. Slamanig, "With a little help from my friends: Constructing practical anonymous credentials," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2004–2023, 2021.

[29] K. Yasuda, T. Lodderstedt, D. Chadwick, K. Nakamura, and J. Vercammen, "OpenID for Verifiable Credentials," tech. rep., 2022.

[30] T. C. Group, "Trusted Platform Module Library Part 3: Commands," tech. rep., 2019.

[31] L. Chen, N. El Kassem, and C. J. Newton, "How to bind a tpm's attestation keys with its endorsement key," *The Computer Journal*, p. bxad037, 2023.

[32] M. Lodder, "The bbs+ signature scheme," 2023.