How to Bind Anonymous Credentials to Humans

Julia Hesse[*]
IBM Research Europe - Zurich
jhs@zurich.ibm.com

Nitin Singh
IBM Research India - Bangalore
nitisin1@in.ibm.com

Alessandro Sorniotti
IBM Research Europe - Zurich
aso@zurich.ibm.com

Abstract

Digital and paper-based authentication are the two predominant mechanisms that have been deployed in the real world to authenticate end-users. When verification of a digital credential is performed in person (e.g. the authentication that was often required to access facilities at the peak of the COVID global pandemic), the two mechanisms are often deployed together: the verifier checks government-issued ID to match the picture on the ID to the individual holding it, and then checks the digital credential to see that the personal details on it match those on the ID, and to discover additional attributes of the holder. This pattern is extremely common and very likely to remain in place for the foreseeable future. However it poses an interesting problem: if the digital credential is privacy-preserving (e.g. based on BBS+ on CL signatures), but the holder is still forced to show an ID card or a passport to verify that the presented credential was indeed issued to the holder, what is the point of deploying privacy-preserving digital credential? In this paper we address this problem by redefining what an ID card should show, and force a minimal but mandatory involvement of the card in the digital interaction. Our approach permits verifiers to successfully authenticate holders and to determine that they are the rightful owners of the digital credential. At the same time, optimal privacy guarantees are preserved. We design our scheme, formally define and analyse its security in the Universal Composability (UC) framework, and implement the card component, showing the running time to be below 200*ms* irrespective of the number of certified attributes.

1 Introduction

In the past decade, digital authentication has made considerable progress from academic proposal to real-world viability, owing to several factors. A first factor is the cryptographic community, which has provided secure, efficient and privacy-preserving tools for *issuers* to issue *credentials* to *holders*, and let them prove their possession to *verifiers*. These tools, predominantly under the umbrella of anonymous credentials, offer guarantees such as untraceability, anonymity and data minimisation which are ideally suited to provide adequate privacy for holders.

A second factor is the blockchain revolution, which has provided a catalyst for taking those academic tools and bringing them to fruition as fully-fledged Self-Sovereign Identity (SSI) platforms. An example of such a platform is Hyperledger Indy [41] which provides a full-stack SSI solution. Other examples include Veramo [34] or Okapi [60].

Another factor is constituted by the COVID global pandemic, which in large parts of the globe required the introduction of digital passports that citizens had to show, in person, to prove facts about their health status to human verifiers. The latter could perform the verification aided by mobile phones and verifier apps. This has sparked awareness in the public about digital authentication, has fostered the creation of usable wallets for digital credentials and somewhat normalised the fact that citizens carry and present digital credentials to human verifiers. The pandemic has also seen a drastic increase in the level of public-sector investment in digital authentication [5].

In spite of all of these technological as well as behaviour advancements, authentication based on government-issued physical ID is still prevalent and shows no sign of giving way to its digital counterpart. This is true because long-term incumbent authentication paradigms tend to be favoured [13], and also because government-issued physical ID is still to this day the fail-safe way to address the following problem:

If a person verifies a digital credential, how can they be sure that it was issued to the person who presents it?

This problem is typically addressed in one of two ways: either by extending the digital credential with biometric information (e.g. a picture of its rightful owner); or by requiring the person presenting the digital credential to show government-issued physical ID to the person verifying the digital cre-

[*]The author was supported by the Swiss National Science Foundation (SNSF) under the AMBIZIONE grant "Cryptographic Protocols for Human Authentication and the IoT.

dential (as mandated for instance for the verification of the European Digital Green certificate [39]).

Both approaches however fall short when it comes to their privacy guarantees. Indeed, the verification performed in person between prover and verifier typically involves a physical (e.g. looking at a face or inspecting a passport) and a digital (e.g. receiving a credential or scanning a QR code) component. While it is fair to assume transcripts of the physical verification to be transient (e.g. the verifier will forget faces), digital transcripts can be easily stored, shared, stolen and abused. Consequently, we posit that whenever a solution creates digital traces that reveal "too much", untraceability is lost because whatever is revealed in digital form may be abused to enable the construction of full user profiles, where attributes disclosed over time by the holder can be accumulated by a set of honest-but-curious or malicious verifiers.

In this work we address this problem and propose a scheme where digital authentication enjoys ideal privacy guarantees, and where human verifiers are able to securely establish a match between the digital credential and the individual standing in front of them. We achieve this by relying on a new authentication token embodied by a smartcard. The card is issued to the holder, in person, by a trusted issuing authority. The card must be physically presented to verifiers upon verification of the digital credential to act as a binding between the digital (credential) and physical (individual) world. The card plays a similar role as the one played by government-issued physical ID today, with two important distinctions: i) the card only displays a picture of the holder to allow the verifier to match the holder, together with the necessary security features to determine the card's authenticity: crucially, no other information is present on the card, since any and all attributes of the holder will be disclosed digitally; ii) the card must have computation and transmission capabilities (typical of smartcards) to run a small yet mandatory part of the authentication protocol to prevent mix-and-match attacks.

Our contributions include:

- The formalisation of the concept of *Anonymous Credentials with Visual Holder Authentication*, a system that solves the problem described above;
- The proposal of a card-based Anonymous Credential (cbAC) scheme, a novel cryptographic primitive that minimally and efficiently extends the BBS+ [22] signature suite widely used in the SSI community [42, 51] and being currently standardised at the IETF [48]; our scheme achieves several desirable properties: i) it forces the involvement of the card in the authentication step; ii) it respects the asymmetry of resources between card and holder; iii) the card always runs the same program, irrespective of who scans it.
- A formal security modeling and analysis of the primitives in the UC framework;
- A proof-of-concept of the viability of our approach, where we show that the slowest entity in our system – the smartcard – can generate its contribution in the order of hundreds of milliseconds on commodity hardware, irrespective of the number of attributes that are certified in the credential.

The rest of this paper is organised as follows: Section 1.1 discusses the related work. In Section 2 we describe the problem this work solves in more detail, outline our solution and detail system and threat model. Section 3 gives preliminaries on proof systems and BBS+ signatures. Section 4 describes joint proofs of knowledge for BBS+ signatures, a core building block of our solution. In Section 5 we describe and define the cryptographic primitive that we identify to be sufficient for anonymous credentials with visual holder authentication. Section 6 describes the formal security model. Section 7 presents our construction and its security analysis. A performance evaluation is given in Section 8. The Appendices contain additional preliminaries on proofs in the ROM (Appendix A), the full formal protocol description (Appendix B), and the full proofs of our theorems (Appendix C).

1.1 Related Work

Biometric authentication One approach to bind digital credentials to the appearance of the holder is to integrate biometric authentication into the presentation of the credential. The idea of fuzzy cryptography [46] is to derive a high-entropy secret from fuzzy authentication data, such as biometric readings. Such primitives are useful to integrate biometrics of the credential holder into the holder's authentication actions, in order to bind these actions to their appearance. PrivBioMTAuth [43] combines fuzzy extractors [37] and authentication: users can authenticate to remote services using their biometrics sampled by their own mobile phones: a high-entropy secret is generated using the fuzzy extractor, after successful match of a picture of the holder against a pre-generated template, and this secret is later used to produce a zero-knowledge proof that achieves authentication.

Rila et al. [57] propose a system where cardholders authenticate to cards using biometric data (and fingerprints in particular). The authentication occurs between the smartcard and the smartcard reader in adversarial setups, considering for instance replay attacks and active adversaries. All these biometric-based approaches have drawbacks that we aim to avoid in this work. They either encourage digital hubs of biometric information (targets for theft/leaks), or rely on holder devices to perform biometric matching.

Anonymous Credentials Introduced by Chaum [31] and Lysanskaya et al. [50], anonymous credentials (AC) have been refined and augmented with privacy enhancements such as unlinkability across multiple presentation, selective disclosure of attributes (or predicates on them), and support for revocation. The cryptographic primitives that underpin anonymous credentials include Camenisch-Lysanskaya signatures [24–26], the U-Prove protocol suite [16] and BBS+ signatures [22].

Direct Anonymous Attestation (DAA) An immediate application of the concepts that underpin anonymous credentials is direct anonymous attestation (DAA). DAA allows a platform consisting of a secure element (a Trusted Platform Module (TPM) or Trusted Execution Environment (TEE)) and a host to create anonymous attestations and prove that the attestation was generated using an authorized secure element. DAA was initially proposed in [19] and further refined in [17, 18, 22, 23]. The DAA scheme provides a *join* interface through with an issuer binds a host and a secure element as a platform, and certifies the platform by issuing a credential. Later the platform can create valid signatures on messages using *sign* interface. A full featured DAA scheme (e.g. [23]) also features a *verify* interface allowing parties to verify that signatures are created by certified platform, and a *link* interface to determine if two signatures were generated by the same platform. While our definitions resemble the ones in [23], there are several important differences. Most importantly, a DAA protocol outputs an attestation object which can be stored, transferred, and repeatedly (and locally) verified. The goal of credential verification in this work is however not to output such an object, but instead to convince the verifier of a certain "ad hoc" statement (i.e., let them output a bit) with the help of an interactive credential presentation procedure.

Anonymous Credentials on Smart Cards A straightforward approach to bind digital credentials to physical appearance is to delegate the presentation of the credential to a smartcard, which could also embed a tamper-resistant picture of the credential holder. Recent efforts target the real-time applicability of ACs on resource-constrained devices such as smartphones and smart cards. [52, 53] present a smartcard implementation of the U-Prove [54] AC system. Similarly, Idemix AC system [28] on a smartcard is presented in [9, 36, 61]. Batina et al. [4] propose a pairing-based AC system to be implemented on Java cards. A promising line of work [3, 21, 30] for smart card friendly anonymous credentials is *Keyed-Verification Anonymous Credentials* (KVAC) introduced by Chase et. al in [30], where the issuer is also the verifier. Intuitively, the setting in KVAC allows one to replace signatures with simpler message authentication codes.

Multi Device Anonymous Credentials A separate line of work to combine smartcards with attribute-based credentials does not attempt to run the entire protocol on the card: the card is instead used in conjunction with a prover app so that card and prover can jointly authenticate to a verifier. This approach is motivated either by attempts to respect the asymmetry of resources between user and card (e.g. by trying to keep the card's computational load independent of the number of attributes), or to leverage the secure element that is present in smartcards to prevent credential cloning or theft. U-prove's design [16] proposes to split the certified attributes between card and holder to force the card's involvement: our work efficiently translates this approach to the BBS+ setting in a provably secure manner while permitting multi-show credentials, neither of which is supported by the original work. Lueks et al. [49] propose to leverage a central server to assist a user in anonymously presenting a BBS+ credential [12]. The idea is to share the BBS+ signing key between the user device and the server, and use a threshold version of BBS+ for presentation of a credential. Several other works on thresholdizing AC systems exist, however, they put equal load on each proving device [38], or require a majority of them to be honest [59]. Our system puts only minimal computation load on the card, and tolerates corruption of even both card and holder. Closest to our work is a system by Hanzlik and Slamanig [45], which leverages smartphones in conjunction with smart cards, to let both jointly present shared credentials. Their scheme is shown to be efficient in practice, and in particular ensures that the computational overhead of the core device is independent of the number of attributes in the credential. Our system achieves the same independency. However, their solution involves fairly recent cryptographic primitives such as signatures with flexible public keys (SFPK [2]) and signatures on equivalence classes (SPS-EQ [33, 44, 47]), which need to be coupled in a non-trivial manner: the message space of SPS-EQ scheme should match with the key space of the SFPK scheme. Additionally, the choice of the primitives makes it non-trivial to augment the scheme in [45] to support *blind signing* and predicates other than selective disclosure. Our work is based on more established primitives such as BBS+ signature [12] and Schnorr proofs of knowledge; in particular, our credentials *are* BBS+ signatures and thus compatible with existing wallet implementations. BBS+ is backed up by mature implementations [42, 51] and is currently undergoing standardization [48] by the Internet Research Task Force (IRTF). For concrete exposition, we describe our solution for the case of selective disclosure of attributes, though it can be trivially extended to support predicates, which can be efficiently verified using Schnorr proofs. Another notable difference to our work is that [45] demand a weaker notion of anonymity: in their system, cards only communicate with smartphones, and hence they only consider the joint privacy of card and smartphone facing potentially malicious third parties. In our work, we allow more general communication patterns and hence demand the anonymity of cards already stand-alone, meaning that an adversary getting read access to the card should not be able to detect whether he already talked to the same card before. This stronger anonymity guarantee allows the deployment of our scheme in crowded environments such as, e.g., airports, where users have no control over which other devices are within, say, NFC connection range to their card.

2 Problem statement

Our work focuses on scenarios where an individual is required to authenticate, in person, to another individual. We will refer to the former as the *holder* (of a credential) and to the latter as the *verifier* (of that credential). Further, we focus on scenarios where this authentication uses digital means: the credential is thus not a physical artefact but a digital one. Digital credentials are flexible and convenient; furthermore, in contrast with their physical counterparts, digital credentials lend themselves well to the creation of advanced authentication schemes that preserve the anonymity and unlinkability of the holder, and minimise the amount of information a holder has to disclose. For example, it is possible to build a scheme where the owner of a liquor store only learns the value of the boolean associated to the "purchaser is above 18 years of age" predicate when performing the age checks required by law.

Whenever this scenario occurs, the verifier faces a problem, namely, that of determining whether the credential they verify was issued to the person presenting it, as opposed to someone else – a colluding malicious entity or the perpetrator of identity theft. The problem is often resolved by requiring the holder to also produce a physical piece of identification (e.g. an ID card): matching the personal details on the digital credentials with those on the ID card, and visually verifying that the picture on the ID card matches the individual provides the missing link in the verification chain.

This however all but thwarts any privacy ambition, since it forces the release of a full digital fingerprint (and not just the boolean value from the example above), defeating anonymity, unlinkability and data minimisation.

Design principle 1: Matching digital credentials to humans by requiring traditional physical identification means (e.g. passport, ID card) violates the privacy of the holder.

One way to avoid this unwanted release of personal information would be to embed the picture of the holder as part of the digital credential. In the above example, the merchant would seemingly just learn the necessary boolean together with a digital picture needed to match the individual. This strategy poses two challenges: the first is that if the digital picture is sent to the verifier, anonymity, unlinkability and data minimisation are violated as before. This is true since the digital picture acts as a unique identifier for the user (violating anonymity), permits the linking of different presentations to a specific holder (breaking unlinkability) and creates the basis for tracing and profiling, since verifiers may decide to pool information learned about a specific holder from numerous interactions.

Design principle 2: It is not desirable from a privacy standpoint to send digital visual information about the holder to the verifier.

The shortcomings of this approach may be avoided by not sending the digital version of the picture to any verifier-controlled device. This leaves keeping it on holder-controlled devices or sending it to third-party devices. In the former case, the system must ensure the integrity of the displayed picture, to protect against attempts by a malicious holder to authenticate with somebody else's credentials whilst displaying their own picture. This problem might be solvable by resorting to TEEs[1] but does not seem to have easy solutions otherwise, since the holder might create a rogue presentation app that displays any picture of their choice. Even if the picture is cryptographically protected as part of the credential, a rogue holder app will just skip any verification.

Design principle 3: Holder devices should not be trusted to correctly display and/or verify visual information of the holder.

The problem has an easy solution if we assume the existence of a trusted third party that has rolled out trusted devices to which the holder can send visual information, alongside any authentication/integrity information that is appropriately used to determine its correctness. Aside from the fact that this assumption might not be very realistic in several settings, we also contend that it is a bad design from a privacy perspective since its design accepts a collection point for personal identifiable information (PII) that may either be exploited by adversaries, leading to serious privacy breaches, or that may later suffer from benign or malicious function creep.

Design principle 4: Involving third parties to handle digital visual information is not desirable.

2.1 Solution overview

This paper focuses on the design, analysis and implementation of a system for *anonymous credentials with visual holder authentication*. The design will respect the principles established in this section, introducing visual holder authentication to the well-known authentication paradigm of anonymous credentials without compromising on any privacy objective.

Given the discussion in the previous section, we require the visual authentication of the holder to solely relay on physical means, and to be conducted personally by the verifier. We

[1]We have not investigated this research direction which we leave as future work.

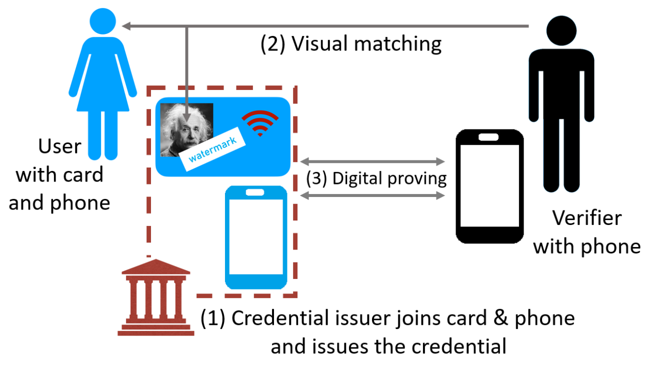


Figure 1: Overview of our anonymous credential system that binds digital proving (3) to visual authentication of the phone holder (2), with the help of picture-showing smartcards that are bound to phones by credential issueing authorities (1).

however know that – as per design principle 1 – we must not relay on existing physical identification artefacts, since they force the (digital) disclosure of additional information about the holder, and in so doing violate anonymity and unlinkability.

We therefore propose a solution where a picture of the holder is displayed on a new physical medium which we shall henceforth refer to as *card* or *smartcard* interchangeably. The card is issued by a trusted *card issuer*, which could be a government or an authority of similar standing. An overview of the entities and flows of our solution is depicted in Figure 1.

Similarly to other physical identification artefacts, this card must be hard to forge and must embed security features and markings that help credential issuing authorities to reliably ascertain its authenticity, however, we do not demand the same capability of the verifiers who may face adversarially programmed cards. Contrary to existing physical authentication artefacts such as identity cards or passports, we require that cards *do not* contain any information other than the picture of the holder: in particular, no personal data such as name, date of birth etc. must be displayed on the card. This way we achieve data minimisation: digitally, the holder is able to disclose a minimum subset of attributes (using the selective disclosure property of attribute-based credentials); physically, the card only discloses a picture of the holder, whom the verifier anyway sees in person.

Helped by the card, our system enables two types of verification: i) an in-person, physical verification carried out by the verifier to check that the holder and their on-card picture match; ii) a digital verification of the credential. Note that the card must also take part in the latter verification in order to prevent mix-and-match attacks, where a malicious holder presents somebody else's credential and their own card. The card is thus required to possess an embedded electronic microprocessor and contactless smart card technology.

The card must additionally not be required to behave differently with different parties (e.g. reveal secrets only if it talks to the holder). Aside from the additional complexity, such a design would also require the deployment of system-wide access control which may be exploited to violate the privacy of users.

2.2 High-level scheme design

An *anonymous credentials with visual holder authentication* system has the following actors: the *card*, the *holder*, the *verifier*, the *card issuer* and the *credential issuer*. The system can be comprised of arbitrarily many issuers, cards, holders, and verifiers. There are three types of interactions between the entities. First, a holder receives a card from a card issuer. Second, a card and a holder can jointly obtain an attribute-based credential from a credential issuer. We refer to this as the "join phase", during which the credential issuer verifies in person that the card matches its holder and is not a forgery, to then issue the credential. Third, a holder and a card can jointly convince any verifier of the possession of a credential over specific attributes. We call this the "presentation phase". We describe these three interactions here at a high-level:

- at first the holder authenticates, in person, to the card issuer and – upon success – obtains a smartcard from the latter; the card displays a picture of the holder, and other necessary markings to determine the card's authenticity. The card's digital infrastructure is equipped with a secret identifier uid (and other secret values) that will enable the minimal but mandatory contribution of the card in the digital authentication protocol; note that the identifier uid is *not* known by the holder.
- as in the previous phase, the holder is required to authenticate, in person, to a credential issuer, and produce a genuine card whose picture matches their visual traits. Then the holder obtains a digital credential from a credential issuer, wherein the credential issuer certifies a set of attributes of the holder. One of the attributes that is (blindly) certified is the card's identifier (uid). This requires the participation of the card: the credential issuer scans the card, obtaining the blind signature request component for the uid, which the credential issuer proceeds to sign, alongside the other attributes in the credential.
- the holder presents a digital credentials to a verifier. During a presentation the holder may choose to disclose certain attributes while keeping others secret (still proving their knowledge). This step is analogous to the traditional presentation step of anonymous credentials, but for one crucial difference: given that the uid is not known by the holder, it can neither be disclosed to the verifier, nor can its knowledge be proven by the holder. As a consequence, the card must be present during the authentication protocol, playing the following dual role:

i) serve as reference for the visual authentication of the holder performed in person by the verifier; ii) be scanned by the verifier and contribute to the cryptographic authentication protocol with the proof of knowledge of the undisclosed attribute uid.

2.3 Threat model

We describe here the threat model and objectives of the system.

2.3.1 Issuers

Card issuers are assumed to be honest and thus not corruptible by the adversary: i) individuals receive from them only cards with matching pictures (which also implies in-person verification and issuance); ii) uid values and other secrets are unique per card and are not shared with anyone else; iii) card issuers do not collude with any other entity in the system to violate, e.g., the privacy of holders or the anonymity of cards.

Credential issuers on the other hand need not be trusted. Formally, we allow credential issuers to be maliciously corrupted. The effect of such corruption is that the adversary fully controls the credential issuer. We note that, although a credential system with such a corrupt credential issuer cannot ensure unforgeability of credentials anymore, our modeling of issuers allows to still evaluate whether, e.g., anonymity or privacy of holders is still guaranteed in the worst case of leaked issueing keys.

2.3.2 Holders and verifiers

Both holders and verifiers can arbitrarily deviate from the protocol. For example, corrupt holders could collude with each other to combine their credentials. Corrupt verifiers could for example enter the system with the sole purpose of learning holder's attributes, or of stealing their credentials: they will collude with other verifiers in order to trace holders across multiple interactions and build holder profiles. Formally, this means we allow for static malicious corruptions of holders and verifiers. Contrary to credential issuers, we do not assume verifiers to be capable of detecting counterfeit cards. This means that verifiers could be presented with cards which the adversary programmed arbitrarily.

As discussed in the previous subsection, verifiers perform two verifications, a physical one to match holders and their picture, and a digital one to establish integrity and provenance of the credential. Our scheme guarantees anonymity and unlinkability of holders for the digital verification. Concerning the physical verification, a malicious verifier could try to remember the facial features of a holder they see often and generate an offline attribute profile based on that. They could also surreptitiously take pictures of holders and create a database where pictures (acting as primary keys) are linked to attributes. We consider these threats outside of the scope of our work: we will restrict our guarantees to the digital interactions, where biometric information is not digitally exchanged/recorded as part of routine processing, and where any and all transcripts of the physical verification are transient (humans forget, pictures are not taken, cctv tapes are eventually destroyed etc.). Our work also supports honest verifiers who want to avoid handling biometrics to prevent leaks/theft/liabilities. This is in contrast to protocols that rely on machine verifiers, where biometric data must be sent to the verifier digitally to perform the authentication successfully.

2.3.3 Smartcards

Smartcards are assumed to have computing/storage facility and NFC capabilities. Smartcards' local storage is expected to be tamper-resistant: secrets stored on the card are inaccessible to all actors. Smartcards are also assumed to guarantee the integrity of the processing logic: cards cannot be forced or tricked into deviating from the original program, and cannot be reprogrammed. These assumptions are consistent with current smartcard technology.

We assume that credential issuers are capable of detecting counterfeit cards, i.e., cards that have not been issued by the card issuer. On the other hand, verifiers are not necessarily required to be able to detect counterfeit cards: this means that verifiers could be presented with cards which the adversary programmed arbitrarily. Summarizing, we disallow corrupt cards to enter the join phase, but we allow malicious cards to enter the presentation phase. Crucially, such card corruptions are *static*, which means that the adversary cannot reprogram any smartcard issued by the card issuer, and it also does not get access to the internal values of any such card. We also assume that cards cannot (and must not) verify the identity of credential issuers or holders, and hence the adversary is allowed to interact with cards during all protocol phases.

We envision our system to be used in settings where a holder presents a smartcard to the issuer and to verifiers in person, i.e., we can assume them to be in physical proximity. Due to the physical proximity, we can rule out the presence of network attackers, and hence we formally assume the availability of secure channels between all entities, in all three phases. We must also assume that relay/man-in-the-middle attacks can be prevented by the local proximity scanning settings, to ensure that the card being visually inspected is also generating the protocol messages: it should therefore be impossible for malicious holders to collude in order to show a counterfeit card and have it relay messages that are forwarded to and from the genuine card of another holder.

3 Preliminaries and Notation

3.1 Bilinear groups

An asymmetric bilinear type-3 group generator is a PPT algorithm BGen that takes as input the security parameter λ and outputs a tuple $\mathsf{BG} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, g_T, e, p)$, where

- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of order p, where p is an λ-bit prime.
- $\mathbb{G}_1 = \langle g_1 \rangle, \mathbb{G}_2 = \langle g_2 \rangle, \mathbb{G}_T = \langle g_T \rangle$.
- $e : \mathbb{G}_1 \times G_2 \to \mathbb{G}_T$ is an efficiently computable non-degenerate bilinear map.
- There is no efficiently computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$.

3.2 Signature of Knowledge

We define signatures of knowledge, and later use their instantiations for discrete-log like relations in the random oracle model [7, 40].

Definition 3.1 (Signature of Knowledge)**.** *Let $\mathcal{U}_\lambda$ denote the set of functions from $\{0,1\}^*$ to $\{0,1\}^\lambda$. A pair of PPT algorithms* $\mathsf{SoK} = (\mathsf{Prove}, \mathsf{Verify})$ *are called a* signature of knowledge *for NP relation $\mathcal{R}$ if all of the following hold, where A^X means that algorithm A has oracle access to function or algorithm X.*

- **Completeness***: For all $(x,w) \in \mathcal{R}$, $m \in \{0,1\}^*$, $\mathsf{H} \in \mathcal{U}_\lambda$, and $\pi \leftarrow \mathsf{SoK.Prove}^{\mathsf{H}}(x,m,w)$ we have $\mathsf{SoK.Verify}^{\mathsf{H}}(x,m,\pi) = 1$.*

- **Zero-Knowledge** [8]*: There exists a PPT simulator $\mathsf{SoK.Sim}$ which emulates a random-oracle $\mathsf{SoK.Sim.H}$ such that the below holds for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.*

$$\Pr\left[\begin{array}{c|r} \mathcal{A}_2^{\mathsf{H}_b}(\mathsf{state}, \pi_b) = b \wedge & (x,m,w,\mathsf{state}) \leftarrow \mathcal{A}_1(1^\lambda) \\ \mathcal{R}(x,w) = 1 & \pi_0 \leftarrow \mathsf{SoK.Prove}^{\mathsf{H}}(x,m,w) \\ & \pi_1 \leftarrow \mathsf{SoK.Sim}(x,m) \\ & b \leftarrow \{0,1\} \end{array}\right]$$

is negligible in λ, where $\mathsf{H}_0() = \mathsf{H}()$ for $\mathsf{H} \leftarrow \mathcal{U}_\lambda$ and $\mathsf{H}_1() = \mathsf{SoK.Sim.H}()$.

- **Argument-of-Knowledge***: There exists a PPT extractor $\mathsf{SoK}.\mathcal{E}$ such that for any PPT adversary $\mathcal{A}$ the following holds with overwhelming probability (over random choices of PPT algorithms and random oracle H): $(x,m,\pi) \leftarrow \mathcal{A}^{\mathsf{H}}$, $w \leftarrow \mathsf{SoK}.\mathcal{E}^{\mathcal{A},\mathsf{H}}(x,m,\pi)$, $(x,w) \in \mathcal{R} \vee \neg\mathsf{SoK.Verify}^{\mathsf{H}}(x,m,\pi)$. Here the notation $\mathsf{SoK}.\mathcal{E}^{\mathcal{A},\mathsf{H}}$ denotes that $\mathsf{SoK}.\mathcal{E}$ can access queries (and answers) made by $\mathcal{A}$ to H as well as query H itself. Additionally $\mathsf{SoK}.\mathcal{E}$ has access to copies of initial state of $\mathcal{A}$ to which it can simulate its own random oracle.*

More details on above notions in the random oracle setting appear in Appendix A and references therein.

3.2.1 Signatures of Knowledge for discrete log

In this paper, relations of interest to us are over cyclic groups of prime order. Let $\mathbb{G}$ be a cyclic group of prime order p. For integers $s,k,n \geq 1$, we consider relations $\mathcal{R}_{s,k,n}$ consisting of pairs (x,w) with $x = (y_1, \ldots, y_s, g_1, \ldots, g_k) \in \mathbb{G}^{s+k}$, $w = (\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}_p^n$ such that:

$$(x,w) \in \mathcal{R}_{s,k,n} \Leftrightarrow \bigwedge_{i=1}^{s} \mathcal{L}_i(x,w) \qquad (1)$$

where each $\mathcal{L}_i(x,w)$ is of the form $\prod_{j=1}^{t_i} g_{ij}^{\alpha_{ij}} = y_i$ with $\{g_{i1}, \ldots, g_{it_i}\} \subseteq \{g_1, \ldots, g_k\}$ and $\{\alpha_{i1}, \ldots, \alpha_{it_i}\} \subseteq \{\alpha_1, \ldots, \alpha_n\}$. We call the elements $(y_1, \ldots, y_\ell)$ in the statement x as "commitments", while $(g_1, \ldots, g_k)$ are referred to as "generators". Constructions of signatures of knowledge for relations $\mathcal{R}_{s,k,n}$ as defined above are presented in [27] and we recap it below.

Lemma 3.1 ([27])**.** *Let $s,k,n \in \mathbb{N}$ and $\mathbb{G}$ denote a cyclic group. There exist a signature of knowledge* $\mathsf{SDL} = (\mathsf{Prove}, \mathsf{Verify})$ *for relation $\mathcal{R}_{s,k,n}$ in the random oracle model, assuming the hardness of computing discrete logarithms in $\mathbb{G}$.*

These signatures SDL are also proven to be *existentially unforgeable* [56] under the same assumptions. Following the notation introduced in [27], we use

$$\pi \leftarrow \mathsf{SDL}\{(\alpha_1, \ldots, \alpha_n) : \bigwedge_{i=1}^{s} \mathcal{L}_i(x,w)\}(m)$$

to denote the output of $\mathsf{SDL.Prove}^{\mathsf{H}}(x,m,w)$ where $(x,w) \in \mathcal{R}_{s,k,n}$ and $m \in \{0,1\}^*$. When m is the empty string $\perp$, we omit (m) in the above notation, and call π a *proof of knowledge*.
Notation: All constructions of signatures of knowledge as defined in Definition 3.1 use a concrete hash function $\mathcal{H}$, which models the access to random-oracle H in the definition. Thus, notation for algorithms $\mathsf{SDL.Prove}$, $\mathsf{SDL.Verify}$ will not specify oracle access.

3.3 BBS+ Signature Scheme

BBS+ signatures were introduced in [1], building upon the BBS signatures introduced by [12]. Subsequently, the construction in [1] was adapted to asymmetric bilinear groups by Camenisch et al. [22].

Definition 3.2 (BBS+ Signatures [22])**.** *Let* $\mathsf{BG} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, g_T, e, p) \leftarrow \mathsf{BGen}(1^\lambda)$ *denote a bilinear group and $\ell \in \mathbb{N}$. Then the BBS+ signature scheme over* BG *with dimension ℓ is described by the algorithms* $\mathsf{BBS}^+ := (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *as below:*

– KeyGen*: Sample* $h_0,\dots,h_\ell \leftarrow \mathbb{G}_1^{\ell+1}$, $x \leftarrow \mathbb{Z}_p$, $w \leftarrow g_2^x$, $\bar{g}_1 \leftarrow \mathbb{G}_1$, $\bar{g}_2 \leftarrow \bar{g}_1^x$. *Set* $sk = x$ *and* $vk = (w,\bar{g}_1,\bar{g}_2,h_0,\dots,h_L)$.

– Sign*: On input message* $\mathbf{m} = (m_1,\dots,m_\ell) \in \mathbb{Z}_p^\ell$ *and secret key x, sample* $e \leftarrow \mathbb{Z}_p\backslash\{x\}$, $s \leftarrow \mathbb{Z}_p$, *compute* $A = \left(g_1 h_0^s \prod_{i=1}^{\ell} h_i^{m_i}\right)^{1/(e+x)}$. *Output* (A,e,s) *as the signature on* $\mathbf{m}$.

– Verify*: On input a public key* $(w,h_0,\dots,h_\ell)$*, message* $\mathbf{m} = (m_1,\dots,m_\ell)$ *and signature* $\sigma = (A,e,s)$*, output* $e(A,wg_2^e) \stackrel{?}{=} e(g_1 h_0^s \prod_{i=1}^{\ell} h_i^{m_i}, g_2)$.

The above signature scheme is proven to be *existentially unforgeable under chosen message attack* (EUF-CMA) under the q-Strong Diffie-Hellman Assumption (qSDH) in BG [11], which demands that no efficient adversary given the $q+3$ tuple $(g_1,g_1^x,g_1^{x^2},\dots,g_1^{x^q},g_2,g_2^x) \in \mathbb{G}_1^{q+1}\times\mathbb{G}_2^2$ can output $(c,g_1^{1/(x+c)}) \in \mathbb{Z}_p\setminus\{-x\}\times\mathbb{G}_1$, except with negligible probability.

3.4 Proof of Knowledge for BBS+ Signatures

We describe the proof of knowledge of BBS+ signature as presented in [22]. The prover in possession of a BBS+ signature (A,e,s) on attributes $(m_1,\dots,m_\ell)$, selectively discloses the attributes $\mathbf{a}_V = \{(i,m_i) : i \in V\}$ to a verifier $\mathcal{V}$ as follows: The prover chooses $r_1 \leftarrow \mathbb{Z}_p^*$, $r_2 \leftarrow \mathbb{Z}_p$ and computes $r_3 = 1/r_1$. Next it computes $A' = A^{r_1}$, $b = g_1 h_0^s \prod_{i=1}^{\ell} h_i^{m_i}$, $\overline{A} = A'^{-e} b^{r_1}$, $d = b^{r_1} h_0^{-r_2}$. Finally, the prover computes proof π as:

$$\pi \leftarrow \mathsf{SDL}\{(e,s,r_2,r_3,\{m_i\}_{i\in H}) : \\ A'^{-e} h_0^{r_2} = \overline{A}/d \wedge d^{-r_3} h_0^s \prod_{i\in H} h_i^{m_i} = g_1^{-1} \prod_{i\in V} h_i^{-m_i}\}$$

In the above, the set $H = L\setminus V$ corresponds to undisclosed attributes $\mathbf{a}_H = \{(i,m_i) : i \in H\}$. The prover sends $(A',\overline{A},d,\pi)$ to the verifier, who checks $e(A',w) = e(\overline{A},g_2)$ and verifies the proof π against the statement computed from $A',\overline{A},d,V$. For proof of completeness, zero knowledge and argument-of-knowledge we refer the reader to Section 4 in [22].

3.5 Signatures over committed messages

The BBS+ signature scheme outlined above allows an issuer to sign attributes while only knowing a commitment over them. This feature of the BBS+ signature scheme is used in constructions (Section B.3) for realising the version of our scheme featuring *blind issuance* property (Section 6). The protocol below due to Au et al. [1] allows a holder to obtain signature over message vector $(m_1,\dots,m_\ell)$ where $\{m_i : i \in H\}$ are hidden from the issuer for some publicly known sets H,L with $H \subseteq L$.

- Holder computes commitment $C = h_0^{s'} \prod_{i\in H} h_i^{m_i}$ and the proof $\pi \leftarrow \mathsf{SDL}\{(s',\{m_i\}_{i\in H}) : h_0^{s'} \prod_{i\in H} h_i^{m_i} = C\}$. It sends (C,π) to the issuer.
- The issuer verifies $b \leftarrow \mathsf{SDL.Verify}((C,\{h_i\}_{i\in H}),\pi)$. The issuer aborts if the verification fails. Otherwise, it computes the signature as: $e \leftarrow \mathbb{Z}_p^*\backslash\{x\}$, $s \leftarrow \mathbb{Z}_p$, $A = (g_1 h_0^s \cdot C \cdot \prod_{i\in L\setminus H} h_i^{m_i})^{1/(e+x)}$. It sends $\mathbf{a}_I,(A,e,s)$ to the holder.
- The holder sets $\sigma = (A,e,s+s')$ as the signature over the message vector $(m_1,\dots,m_\ell)$.

Signatures over committed messages have applications in scenarios where users must get their cryptographic secrets (such as signing keys) certified by an issuer without revealing the secrets to the issuer.

4 Joint Proof of Knowledge for BBS+

In this section, we present a novel proof of knowledge scheme for BBS+ signatures that requires two parties to contribute. This new primitive is necessary to construct our scheme since we require credential presentations that require the joint participation of holder and card. In our design we strive to remain as compatible as possible to BBS+ signatures in order to preserve as much of the existing ecosystem (components, code, formats, standards) as possible.

The main idea is as follows. Instead of storing all attributes $(m_1,\dots,m_\ell)$ on the holder, we “shave off" the attribute m_1 from the holder storage, and instead store m_1 on a smart card $\mathcal{C}$. Next, we modify the protocol for proving knowledge of a valid BBS+ message-signature pair in Section 3.4 to enable the holder and the card to construct this proof jointly with *minimal* but *crucial* involvement of the card. Specifically, we decompose the prover algorithm $\mathcal{P}$ for the BBS+ proof of knowledge from Section 3.4, into two PPT algorithms $(\mathsf{Prove}_{\mathcal{H}}(\tau,\cdot),\mathsf{Prove}_{\mathcal{C}}(\tau,\cdot))$ which share the state τ. Here $\mathsf{Prove}_{\mathcal{C}}$ is lightweight and executed by the smart-card $\mathcal{C}$ while $\mathsf{Prove}_{\mathcal{H}}$ is executed by the holder device $\mathcal{H}$. We provide details of the decomposition in Figure 2 and the overall protocol in Figure 3. The decomposition roughly works as follows: proving knowledge of BBS+ signature involves showing knowledge of exponents $(s,m_1,\dots,m_\ell)$ over generators $h_0,\dots,h_\ell$ which satisfies $h_0^s h_1^{m_1}\cdots h_\ell^{m_\ell} = P$ for publicly known P. In the above decomposition, card and holder generate shared randomness r using the PRF key K and then the card proves $h_1^{m_1} h_0^r = B$ and the holder proves $h_0^{s'} h_2^{m_2}\cdots h_\ell^{m_\ell} = PB^{-1}$ using $s' = s-r$, which convinces the verifier that they together know the entire vector. The shared state τ consists of a PRF key K and a non-hiding commitment $Q = h_1^{m_1}$ to message m_1 contributed by the card. All algorithms implicitly have public parameters as input. We state the security of our scheme in the following theorem and provide a full proof in Appendix C.

Theorem 4.1. *Let* $q,k,n \in \mathbb{N}$ *and* $\mathsf{BBS}^+ := (\mathsf{KeyGen},\mathsf{Sign},\mathsf{Verify})$ *denote the BBS+ signature scheme with dimension* ℓ *over bilinear groups. If computation of discrete logarithms in* $\mathbb{G}_1$ *is hard,* SDL *is a signature of*

```
Prove_C(K, m_1, n_1, n): // Executed by card
  1. $r = \mathsf{PRF}_K(n)$.
  2. $B = h_1^{m_1} h_0^r$. // $h_0, h_1$ from public params
  3. $\pi \leftarrow \mathsf{SDL}\{(\alpha, \beta) : h_1^{\alpha} h_0^{\beta} = B\}(n_1)$.
  4. return $(B, \pi)$.
```

```
Prove_H(K, Q, (m_2, ..., m_ℓ), (A, e, s), a_V, n, B, n_2): // Executed by holder. All generators come from public parameters
  1. Parse $\mathbf{a}_V = \{(i, v_i) : i \in V\}$. Output $\perp$ if $m_i \neq v_i$ for some $i \in V$.
  2. Set $r = \mathsf{PRF}_K(n)$. Output $\perp$ if $Qh_0^r \neq B$.
  3. Set $H = \{2, \ldots, \ell\} \setminus V$.
  4. $r_1 \leftarrow \mathbb{Z}_p^*$, $r_2 \leftarrow \mathbb{Z}_p$, $r_3 = r_1^{-1}$, $s' = s - r_2 r_3 - r$.
  5. $A' = A^{r_1}$, $b = g_1 h_0^s Q \prod_{i=2}^{\ell} h_i^{m_i}$, $\bar{A} = A'^{-e} b^{r_1}$, $d = b^{r_1} h_0^{-r_2}$.
  6. $\pi' \leftarrow \mathsf{SDL}\{(e, s', r_2, r_3, \{m_i\}_{i \in H}) : A'^{-e} h_0^{r_2} = \bar{A}/d \wedge d^{-r_3} h_0^{s'} \prod_{i \in H} h_i^{m_i} = g_1^{-1} B^{-1} \prod_{i \in V} h_i^{-m_i}\}(n_1)$.
  7. return $(A', \bar{A}, d, \pi')$
```

Figure 2: Splitting a BBS+ proof of knowledge between card and holder

knowledge for relation $\mathcal{R}_{q,k,n}$ (1), and PRF *is a pseudorandom function, the protocol presented in Figure 3 satisfies* completeness, soundness *and* zero-knowledge *as defined below with respect to semi-honest verifier corruption and malicious card and holder corruption in the random-oracle model.*

- **Completeness***: the verifier $\mathcal{V}$ outputs 1 in the honest execution of the protocol whenever* $\mathsf{BBS}^+.\mathsf{Verify}(\mathsf{vk}, (m_1, \ldots, m_\ell), (A, e, s)) = 1$.
- **Soundness***: There exists an efficient extractor $\mathcal{E}$ such that whenever a colluding card $\mathcal{C}$ and holder $\mathcal{H}$ succeed against an honest verifier $\mathcal{V}$ ($\mathcal{V}$ outputs 1), $\mathcal{E}^{\mathcal{A}}(\mathbf{a}_V)$ outputs $(\mathbf{m}, \sigma)$ where σ is a verifying signature on $\mathbf{m} \in \mathbb{Z}_p^{\ell}$ with respect to public key w. Here $\mathcal{A}$ denotes the adversary corrupting $\mathcal{H}$ and $\mathcal{C}$.*
- **Zero Knowledge***: There exists a simulator $\mathcal{S}$ which simulates the view of a semi-honest verifier $\mathcal{V}$ in the protocol with honest $\mathcal{C}$ and $\mathcal{H}$.*

Remark: We only prove the above theorem assuming $\mathcal{V}$ is semi-honest, as it is sufficient for proving the security of our overall solution later in Sections 7 and 7.1. In the full protocol in Section B.4, malicious behavior of $\mathcal{V}$ can be detected by the honest holder, which then outputs a “dummy" proof $\perp$.

- **Setup**: Generate a bilinear group $\mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, g_T, ep)$ and obtain $(\mathsf{sk}, \mathsf{vk}) := (x, (w, \bar{g}_1, \bar{g}_2, h_0, \ldots, h_\ell)) \leftarrow \mathsf{BBS}^+.\mathsf{KeyGen}$. Sample a PRF key $K \leftarrow \mathcal{K}$. Everything except x, K constitutes public parameters, denoted by pp.
- **Card's Inputs**: $0 \neq m_1 \in \mathbb{Z}_p$, PRF key K.
- **Holder's Inputs**: PRF key K, $Q = h_1^{m_1}$, message $m_2, \ldots, m_\ell \in \mathbb{Z}_p$, BBS+ signature $(A, e, s) \leftarrow \mathsf{BBS}^+.\mathsf{Sign}((m_1, m_2, \ldots, m_\ell), x)$.
- **Verifier's Inputs**: Attributes $\mathbf{a}_V = \{(i, v_i) : i \in V\}$, where V denotes the indices of attributes to be disclosed and v_i being the corresponding target values. We assume $1 \notin V$.
- **Protocol**: We denote card, holder and verifier by $\mathcal{C}$, $\mathcal{H}$ and $\mathcal{V}$ respectively.
  - $\mathcal{H} \to \mathcal{V}$: Nonce $n_{\mathcal{H}} \leftarrow \{0,1\}^{\lambda}$.
  - $\mathcal{V} \to \mathcal{C}$: $(n_{\mathcal{H}}, n_{\mathcal{V}})$ where $n_{\mathcal{V}} \leftarrow \{0,1\}^{\lambda}$.
  - $\mathcal{C} \to \mathcal{V}$: $n_{\mathcal{C}}, B, \pi$ where $n_{\mathcal{C}} \leftarrow \{0,1\}^{\lambda}$, $(B, \pi) \leftarrow \mathsf{Prove}_{\mathcal{C}}(K, m_1, n_{\mathcal{V}}, n_{\mathcal{C}} || n_{\mathcal{H}})$ (See Figure 2).
  - $\mathcal{V} \to \mathcal{H}$: $n_{\mathcal{C}}, n_{\mathcal{V}}, \mathbf{a}_V, B$.
  - $\mathcal{H}$: $\mathsf{proof} \leftarrow \mathsf{Prove}_{\mathcal{H}}(K, Q, (m_i)_{i=2}^{\ell}, (A, e, s), \mathbf{a}_V, n, B, n_{\mathcal{V}})$ for $n = n_{\mathcal{C}} || n_{\mathcal{H}}$ (an instantiation of $\mathsf{Prove}_{\mathcal{H}}$ is described in Figure 2).
  - $\mathcal{H} \to \mathcal{V}$: proof.
  - $\mathcal{V}$: If $\mathsf{proof} = \perp$ output 0, else parse proof as $(A', \bar{A}, d, \pi')$.
  - $\mathcal{V}$: Checks (i) $e(\bar{A}, g_2) = e(A', w)$ (ii) Proofs π and π' are valid. It outputs 1 if all the checks pass, and outputs 0 otherwise.

Figure 3: An interactive protocol for a card-based proof of BBS+ signature

5 A Model for Secure card-based Anonymous Credentials

We now present the core cryptographic building block of our anonymous credential scheme with visual holder authentication, which we call *card-based anonymous credentials* (cbAC). cbAC formally defines the interactions between holders, verifiers and credential issuers introduced in Section 2.2. We choose not to include card issuers as part of cbAC since visual verification and pictures of holders are no cryptographic procedures or objects.

Before proceeding, we introduce some notation regarding attribute formats. Throughout the paper, we assume attributes to sort into ℓ “categories” (e.g., birthdate, citizenship, or hair color). We denote by $L := \{1, \ldots, \ell\}$ the full set of indices, and call any set $V \subseteq L$ an *index set*. Attributes from the category with index i, $i \in L$, can take values in *universe* $\mathcal{U}_i$. We often use shorthand notation $\mathbf{a}_V := \{(i, m_i) : i \in V\}$ for some $m_i \in$

$\mathcal{U}_i$ for each $i \in V$, in contexts where the concrete values of m_i are not relevant. We denote the "merge" of two attribute sets by $\{(i,m_i) : i \in V\} \oplus \{(j,m_j) : j \in H\} := \{(i,m_i) : i \in L\}$ which is only well-defined if the index sets V and H are disjoint, i.e., if the original attribute sets do not both contain attributes from the same category.

We start by giving an algorithmic description for cbAC. We use notation $\mathsf{Alg}(A : x, B : y) \to (B : z)$ to denote a (potentially interactive) procedure Alg where party A has input x, party B has input y, and party B outputs z.

Definition 5.1. *Let $\ell \in \mathbb{N}$, $L := \{0,\dots,\ell\}$ and $\mathcal{U}_i$ denote attribute universes for all $i \in L$. A card-based anonymous credential (cbAC) system for $(\mathcal{U}_i)_{i\in L}$ is a set of three interactive procedures* $(\mathsf{Setup}, \mathsf{Join}, \mathsf{Present})$, *executed between an issuer I, and arbitrary cards $\mathcal{C}$, holders $\mathcal{H}$ and verifiers $\mathcal{V}$ as follows.*

> $\mathsf{Setup}(I : \lambda) \to \mathsf{pp}$*: The setup algorithm is executed by the credential issuer and results in public parameters* pp *which are made available to all entities in the system.*
>
> $\mathsf{Join}(\mathcal{H} : \mathbf{a}_H, \mathcal{C} : \bot, I : \mathbf{a}_I) \to (\mathcal{H} : b)$*: The join procedure is executed between one holder, one card, and the credential issuer, where both the holder and the credential issuer contribute attributes $\mathbf{a}_H \in (\mathcal{U}_i)_{i\in H}$ for $H \subseteq L$, $\mathbf{a}_I \in (\mathcal{U}_i)_{i\in I}$ for $I \subseteq L$. The output of the procedure is a bit b signaling either success or failure to the holder, and we require $b = 0$ if $L \setminus H \neq I$, i.e., ambiguity in attributes is not allowed.*
>
> $\mathsf{Present}(\mathcal{H} : \bot, \mathcal{C} : \bot, \mathcal{V} : \mathbf{a}_V) \to (\mathcal{V} : f)$*: A presentation is executed between one verifier, one card, and one holder. Card and holder receive no input in this phase, but the verifier provides a set of attributes $\mathbf{a}_V \in (\mathcal{U}_i)_{i\in V}$ for some $V \subseteq L$. The result of the procedure is that the verifier outputs a bit f.*

We expect the following properties from a cbAC scheme.

Correctness/Completeness with selective disclosure Let $\mathbf{a}_H, \mathbf{a}_I$ denote two sets of attributes wrt. "index sets" H, I with $L \setminus H = I$. We say that a cbAC scheme satisfies *correctness with selective disclosure* if the following holds. Assume

$$\mathsf{Join}(\mathcal{H} : \mathbf{a}_H, \mathcal{C} : \bot, I : \mathbf{a}_I) = 1$$

for some holder $\mathcal{H}$, some card $\mathcal{C}$, and some attribute set $\mathbf{a}_H \in (\mathcal{U}_i)_{i\in H}, \mathbf{a}_I \in (\mathcal{U}_i)_{i\in I}$. Then it holds that for any $\mathbf{a}_V \subseteq (\mathbf{a}_H \oplus \mathbf{a}_I)$

$$\mathsf{Present}(\mathcal{H} : \bot, \mathcal{C} : \bot, \mathcal{V} : \mathbf{a}_V) = 1$$

Unforgeability We next demand unforgeability of credentials, namely that it be computationally infeasible for a holder $\mathcal{H}'$ and card $\mathcal{C}$ to convince any verifier of the possession of attributes that were not jointly issued to them.

More formally, assume $\mathsf{Present}(\mathcal{H}' : \bot, \mathcal{C} : \bot, \mathcal{V} : \mathbf{a}_V) = 1$ for $\mathbf{a}_V \in (\mathcal{U}_i)_{i\in V}$ for some $V \subseteq L$, some holder $\mathcal{H}'$, and some card $\mathcal{C}$. Then we say that the cbAC scheme is *unforgeable* if there exists a holder $\mathcal{H}$ with inputs $\mathbf{a}_H \in (\mathcal{U}_i)_{i\in H}$ for some $H \subseteq L$ who participated in a join procedure with $\mathcal{C}$ resulting in $\mathsf{Join}(\mathcal{H} : \mathbf{a}_H, \mathcal{C}' : \bot, I : \mathbf{a}_I) = 1$, where $\mathcal{H} = \mathcal{H}'$ or both $\mathcal{H}, \mathcal{H}'$ are corrupt, and $\mathbf{a}_V \subseteq \mathbf{a}_I \oplus \mathbf{a}_H$.

Anonymity From anonymity we understand the inability to recognize the repeated participation of an entity in the digital part of the protocol. We consider anonymity of both cards and holders, while previous works consider only joint anonymity [45], and we consider them from both the perspective of honest-but-curious credential issuers and from the perspective of malicious verifiers. However, as common in the AC literature (e.g., [4, 45, 54]), we only guarantee such anonymity for cards and holders that were joined by *the same* credential issuer. This restriction is natural since credentials are not expected to hide the public key of the credential issuer. Naturally, anonymity can only be guaranteed provided that the set of disclosed attributes does not trivially deanonymize their holder, i.e. the set of disclosed attributes must be identical.

More detailed, we require cards and holders that were joined by the same credential issuer to remain anonymous during a presentation, i.e., a potentially malicious verifier cannot detect which card or holder participated. These guarantees must even hold if the verifier has access to the issuance transcripts, and the internal state of the issuer, e.g., its signing keys.

More formally, we call a cbAC scheme *anonymous during presentation* if the following holds. Consider executions of

$$\mathsf{Join}(\mathcal{H}^0 : \mathbf{a}_H^0, \mathcal{C}^0 : \bot, I : \mathbf{a}_I^0) = 1 \text{ and}$$
$$\mathsf{Join}(\mathcal{H}^1 : \mathbf{a}_H^1, \mathcal{C}^1 : \bot, I : \mathbf{a}_I^1) = 1$$

for inputs $(\mathbf{a}_H^0, \mathbf{a}_H^1, \mathbf{a}_I^0, \mathbf{a}_I^1, \mathcal{C}^0, \mathcal{C}^1, \mathcal{H}^0, \mathcal{H}^1) \leftarrow \mathcal{V}$ provided by any PPT adversary $\mathcal{V}^I$, who has access to the internal state and incoming/outgoing messages of the semi-honest issuer I. Let b denote a random bit. Then $\mathcal{V}^I$ participating in

$$\mathsf{Present}(\mathcal{H}^b : \bot, \mathcal{C}^b : \bot, \mathcal{V}^I : \mathbf{a}_V)$$

where $\mathbf{a}_V \subseteq ((\mathbf{a}_H^0 \oplus \mathbf{a}_I^0) \cap (\mathbf{a}_H^1 \oplus \mathbf{a}_I^1))$ outputs b with advantage negligibly close to $1/2$.

Holder privacy during presentation We next demand strong privacy properties for the holder when presenting attributes. Namely, even a malicious verifier does not learn more information about the attributes of the holder than what is revealed by the outcome of the presentation.

More formally, consider a PPT adversary $\mathcal{V}$ outputting $\mathbf{a}_L^0, \mathbf{a}_L^1, \mathbf{a}_V$ such that $\mathbf{a}_V \subseteq \mathbf{a}_L^0$ and $\mathbf{a}_V \subseteq \mathbf{a}_L^1$, and the identity of some honest $\mathcal{C}$. Let b denote a randomly chosen bit upon which we execute procedures

$$\mathsf{Join}(\mathcal{H} : \mathbf{a}_L^b, \mathcal{C} : \bot, I : \bot) = 1$$

and

$$\mathsf{Present}(\mathcal{H}:\perp,\mathcal{C}:\perp,\mathcal{V}:\mathbf{a}_V)=1.$$

We say that a cbAC scheme has *holder privacy* if the advantage of any such PPT $\mathcal{V}$ outputting b is negligibly close to $1/2$ over the random coins of all the involved entities in the execution.

Unlinkability of presentations Unlinkability of presentations demands that a malicious verifier cannot link two presentations, i.e., tell whether the same card or the same holder was involved in them. Unlinkability implies anonymity and we can hence define it by strengthening the adversary in the anonymity definition above.

More formally, unlinkability is defined as anonymity but where the adversary $\mathcal{V}$ additionally gets take part in arbitrarily many executions of

$$\mathsf{Present}(\mathcal{H}^i:\perp,\mathcal{C}^i:\perp,\mathcal{V}:\mathbf{a}_V)$$

for $i\in\{0,1\}$ before being challenged and making his decision.

Blind issuance of attributes We finally define a property that we consider optional for cbAC in general but useful for some applications. Blind issuance of attributes demands that the holder be able to contribute attributes to the credential that are not seen by the credential issuer.

More formally, let $\mathcal{C},\mathcal{H},H,I\subseteq L$ with $L\setminus I=H$ and $\mathbf{a}^0_H,\mathbf{a}^1_H,\mathbf{a}_I$ all be given by any PPT adversary $\mathcal{A}$. Let b denote a uniformly sampled bit. Consider a run of

$$\mathsf{Join}(\mathcal{H}:\mathbf{a}^b_H,\mathcal{C}:\perp,I:\mathbf{a}_I)=1$$

where $\mathcal{A}$ can observe the internal state of I and sees all messages that I receives. We say that a cbAC scheme supports *blind issuance of attributes* if the advantage of any such PPT $\mathcal{A}$ outputting b is negligibly close to $1/2$.

Use cases for blind issuance include issuance of sensitive attributes such as gender, or protection against credential leakage (e.g., holder's device stores attribute sk in secure storage, such that without knowledge of sk a captured credential is rendered useless).

Interpreting the formal properties for the real-world system. As explained in the beginning of this Section, our formal cbAC model captures the guarantees of the *cryptographic* part of our card-based credential system. Visual verification of pictures on smartcards, and entities such as holders and verifiers being humans who meet in person, are not part of the cryptographic protocol. Consequently, e.g., the anonymity guarantees described above only capture that cards leave no *digital* traces of their identity during a run of the protocol. That said, a verifier can always attempt to somehow *remember* a picture on a smartcard, to deanonymize that card in future presentations. The same obviously holds for employees working at an issueing authority, who can attempt to remember faces of the smartphone owners. Such threats need to be taken into account when deploying the actual system, for example verifiers should perform verification in front of the user, to ensure that no pictures of the smartcard are taken.

Further, care needs to be taken when translating the above guarantees into practice. For example, anonymity only holds for card-holder pairs that were joined by the same credential issuer, such that the anonymity set corresponds to all "customers" of a credential issuer. Consequently, our cbAC system does not provide any meaningful anonymity guarantees in settings where each card-holder pair receives their credential from a different credential issuer. It is an interesting avenue for future work whether techniques to hide the identity of the credential issuer [10, 15, 20] could be applied to our work.

6 Ideal functionality for cbAC

We give a formal definition of card-based anonymous credentials cbAC in terms of an ideal functionality in the Universal Composability [29] framework in Figure 4. The functionality supports three sets interfaces; *Setup* for initialization, *Join* to allow a pair consisting of a card and a holder $(\mathcal{C}_i,\mathcal{H}_j)$ to be jointly issued a set of attributes, and finally *Presentation* to allow a pair $(\mathcal{C}_i,\mathcal{H}_j)$ to present specific attributes towards a verifier $\mathcal{V}$. We start with an honest walk-through of how $\mathcal{F}_{\mathsf{cbAC}}$ is used to join a card and a holder who subsequently present attributes to a verifier. We then detail which attacks it admits[2]. Along the way, we argue how $\mathcal{F}_{\mathsf{cbAC}}$ ensures the security properties of cbAC described above.

Figure 4 presents two functionalities, namely functionality $\mathcal{F}_{\mathsf{cbAC}}$ where the credential issuer determines which attributes a holder gets issued, and $\mathcal{F}^{\mathsf{blind}}_{\mathsf{cbAC}}$ where the holder is allowed to contribute "blind" attributes to the credential (i.e., which are not seen by the issuer). The functionalities only differ in their join phases, where $\mathcal{F}^{\mathsf{blind}}_{\mathsf{cbAC}}$ collects holder attributes $\mathbf{a}_H$ that combine to a "full" ℓ-fold attribute sets together with the credential issuer's attributes $\mathbf{a}_I$, while $\mathcal{F}_{\mathsf{cbAC}}$ expects the issuer attributes $\mathbf{a}_I$ to already contain all ℓ attributes.

The Setup phase. The SETUP interface is called by a credential issuer I, whose identity is encoded in the session identifier of $\mathcal{F}_{\mathsf{cbAC}}$. This modeling enforces that the identity of the issuer is publicly known and nobody can impersonate this entity (e.g., the digital signature verification key of the issuer is reliably known to any other entity in the system). While our modeling assumes only one such issuing party S.1,S2 , in practice the task performed by I can be distributed over many physical institutions who, e.g., all sign under their own keys.

[2]While it is desirable to not allow any attacks from a security perspective, practically-efficient schemes are usually only in reach if we enter tradeoffs and relax the security (e.g., allow for DoS attacks on the scheme). The adversarial interface of $\mathcal{F}_{\mathsf{cbAC}}$ precisely describes these relaxations.

$\mathcal{F}_{\mathsf{cbAC}}$ is instantiated with session identifier $\mathsf{sid} = (\mathcal{I}, sid')$ for some sid', which we omit from all interfaces. $\mathcal{F}_{\mathsf{cbAC}}$ maintains join session records $\mathsf{JR}(\mathsf{jid}) = (\mathtt{C}, \mathtt{H}, \boxed{\mathtt{attH}}, \mathtt{attI})$ and presentation session records $\mathsf{SR}(\mathsf{vid}) = (\mathtt{C}, \mathtt{H}, \mathtt{attV})$. These records are initialized with all values set to $\perp$ when accessed for the first time. Let creds denote an initially empty list. Interfaces of Join and Show can only be called after Setup was completed. We assume $\mathcal{F}_{\mathsf{cbAC}}$ to ignore malformed inputs. $\mathcal{F}_{\mathsf{cbAC}}$ is parametrized by $\ell \in \mathbb{N}$ and we denote $L := \{1, \dots, \ell\}$.

Setup Phase: Initialize the functionality instance.
On input (SETUP) from a party $\mathcal{I}$

- S.1 Ignore if this is not the first SETUP query
- S.2 From now on, use $\mathcal{I}$ to denote the unique party that is allowed to call interface $\mathsf{JOINISSUE}$.
- S.3 Send (SETUP) to $\mathcal{S}$ and a delayed output $(\mathsf{SETUPDONE}, \mathsf{sid})$ to $\mathcal{I}$.

Join Phase: $\boxed{\text{Holder inputs attributes } \mathbf{a}_H = \{(i, m_i) : i \in H\},}$ issuer inputs attributes $\mathbf{a}_I = \{(i, m_i) : i \in I\}$ where $\boxed{I = L \setminus H}$ $\dbox{I = L}$. The card-holder pair is coupled to the attribute vector $\mathbf{m} = (m_1, \dots, m_\ell)$ contained in $\boxed{\mathbf{a}_H}$ and $\mathbf{a}_I$.

`Holder Requests`: On input $(\mathsf{JOIN}, \mathsf{jid}, \mathcal{C}, \boxed{\mathbf{a}_H})$ from $\mathcal{H}$

- J-H.1 Drop the query if $\perp \neq \mathsf{JR}(\mathsf{jid}).\mathtt{C} \neq \mathcal{C}$ or if $\mathsf{JR}(\mathsf{jid}).\mathtt{H} \neq \perp$. Otherwise set $\mathsf{JR}(\mathsf{jid}).\mathtt{H} \leftarrow \mathcal{H}, \mathsf{JR}.(\mathsf{jid}).\mathtt{C} \leftarrow \mathcal{C}$, $\boxed{\text{and } \mathsf{JR}(\mathsf{jid}).\mathtt{attH} \leftarrow \mathbf{a}_H}$.
- J-H.2 Output $(\mathsf{JOIN}, \mathsf{jid}, H, \mathcal{H})$ to $\mathcal{S}$. // No anonymity for holder in join phase.

`Card joins`: On input $(\mathsf{JOINID}, \mathsf{jid})$ from $\mathcal{C}$

- J-C.1 Drop the query if $\perp \neq \mathsf{JR}(\mathsf{jid}).\mathtt{C} \neq \mathcal{C}$. Otherwise set $\mathsf{JR}.(\mathsf{jid}).\mathtt{C} \leftarrow \mathcal{C}$.
- J-C.2 Output $(\mathsf{JOINID}, \mathsf{jid}, \mathcal{C})$ to $\mathcal{S}$. // No anonymity for card in the join phase.

`Issuer Agrees`: On input $(\mathsf{JOINISSUE}, \mathsf{jid}, \mathbf{a}_I)$ from $\mathcal{I}$

- J-I.1 $\dbox{\text{Drop the query if the index set } I \text{ of } \mathbf{a}_I \text{ is not equal to } L}$.
- J-I.2 Create record $\mathsf{JR}(\mathsf{jid})$ with $\mathtt{C} \leftarrow \perp$, $\mathtt{H} \leftarrow \perp$, $\boxed{\mathtt{attH} \leftarrow \perp}$, $\mathtt{attI} \leftarrow \mathbf{a}_I$ if no such record exists. Otherwise, set $\mathsf{JR}(\mathsf{jid}).\mathtt{attI} \leftarrow \mathbf{a}_I$ if $\mathsf{JR}(\mathsf{jid}).\mathtt{attI} = \perp$. // Add attributes contributed by issuer to the record.
- J-I.3 Send $(\mathsf{JOINISSUE}, \mathsf{jid})$ to $\mathcal{S}$ and send a delayed output $(\mathsf{JOINISSUE}, \mathsf{jid}, \mathbf{a}_I)$ to $\mathcal{H}$.

`Finalize the join`: On input $(\mathsf{JOINCOMPLETE}, \mathsf{jid})$ from $\mathcal{S}$

- J.1 Ignore if there is no record $\mathsf{JR}(\mathsf{jid})$, or any of its variables is $\perp$.
- J.2 $\boxed{\text{Parse } \mathsf{JR}(\mathsf{jid}).\mathtt{attH} \text{ as } \{(i, p_i) : i \in H\},\ \mathsf{JR}(\mathsf{jid}).\mathtt{attI} \text{ as } \{(i, q_i) : i \in I\} \text{ for some } H, I \subseteq L \text{ and drop the query if } H \neq L \setminus I.}$
- J.3 Construct $\mathbf{m} = (m_1, \dots, m_\ell)$ as follows: Set $m_i = q_i$ for $i \in I$, $\boxed{m_i = p_i \text{ for } i \in H.}$
- J.4 Add $(\mathsf{JR}(\mathsf{jid}).\mathtt{H}, \mathsf{JR}(\mathsf{jid}).\mathtt{C}, \mathbf{m})$ to creds. // Credential installed: $\mathcal{H}$ and $\mathcal{C}$ can from now on show $\mathbf{m}$.
- J.5 Delete record $\mathsf{JR}(\mathsf{jid})$ and send a delayed output $(\mathsf{JOINED}, \mathsf{jid})$ to $\mathcal{H}$.

Presentation Phase: During a presentation phase, the card-holder pair authenticates against a set of attributes $\mathbf{a}_V = \{(i, m_i) : i \in V\}$ specified by the verifier $\mathcal{V}$. They succeed if they have been previously coupled to vector $\mathbf{m} \in \mathbb{Z}_p^\ell$ such that $m_i = \mathbf{m}[i]$ for $i \in V$.

`Set Attributes`: On input $(\mathsf{SETATTRS}, \mathsf{vid}, \mathbf{a}_V)$ from $\mathcal{V}$

- P-V.1 Create record $\mathsf{SR}(\mathsf{vid}) = (\perp, \perp, \mathbf{a}_V)$ if no such record exists. Otherwise, set $\mathsf{SR}(\mathsf{vid}).\mathtt{attV} \leftarrow \mathbf{a}_V$ if $\mathsf{SR}(\mathsf{vid}).\mathtt{attV} = \perp$.
- P-V.2 Output $(\mathsf{SETATTRS}, \mathsf{vid}, \mathcal{V}, \mathbf{a}_V)$ to $\mathcal{S}$. // Verifier's identity and attributes are public.

`Set Card`: On input $(\mathsf{SETID}, \mathsf{vid})$ from $\mathcal{C}$

- P-C.1 Drop the query if $\mathcal{C}$ is honest and $(*, \mathcal{C}, *) \notin \mathsf{creds}$. // Uninitialized card.
- P-C.2 Create record $\mathsf{SR}(\mathsf{vid})$ with $\mathtt{C} \leftarrow \mathcal{C}$, $\mathtt{H} \leftarrow \perp$, $\mathtt{attV} \leftarrow \perp$ if no such record exists. Otherwise, set $\mathsf{SR}(\mathsf{vid}).\mathtt{C} \leftarrow \mathcal{C}$ if $\mathsf{SR}(\mathsf{vid}).\mathtt{C} = \perp$.
- P-C.3 If $\mathcal{H}$ is corrupt, and $(\mathcal{H}', \mathcal{C}, *) \in \mathsf{creds}$ for corrupt $\mathcal{H}'$, output $(\mathsf{SETID}, \mathsf{vid}, \mathcal{C})$ to $\mathcal{S}$, else output $(\mathsf{SETID}, \mathsf{vid})$ to $\mathcal{S}$. // Card remains anonymous as long as holder is not corrupt and has already used that card during issuance.

`Set Credential`: On input $(\mathsf{SETCRED}, \mathsf{vid})$ from $\mathcal{H}$

- P-H.1 Create record $\mathsf{SR}(\mathsf{vid}) = (\perp, \mathcal{H}, \perp)$ if no such record exists. Otherwise, set $\mathsf{SR}(\mathsf{vid}).\mathtt{H} \leftarrow \mathcal{H}$ if $\mathsf{SR}(\mathsf{vid}).\mathtt{H} = \perp$.
- P-H.2 Output $(\mathsf{SETCRED}, \mathsf{vid})$ to $\mathcal{S}$. // Holder remains anonymous.

`Verify`: On input $(\mathsf{VERIFYCOMPLETE}, \mathsf{vid}, b)$ from $\mathcal{S}$

- V.1 Ignore if there is no record $\mathsf{SR}(\mathsf{vid})$, or any of its variables is $\perp$. // Card or holder missing.
- V.2 Parse $\mathsf{SR}(\mathsf{vid}).\mathtt{C}$ as $\mathcal{C}$, $\mathsf{SR}(\mathsf{vid}).\mathtt{H}$ as $\mathcal{H}$, $\mathsf{SR}(\mathsf{vid}).\mathtt{attV}$ as $\{(i, m_i) : i \in V\}$ for some $V \subseteq L$.
- V.3 Set $f = 0$. // Unforgeability: only change this to 1 below if there is a corresponding credential in creds.
- V.4 If $\mathcal{H}$ is honest and $b = 1$, set $f = 1$ if $(\mathcal{H}, \mathcal{C}, \mathbf{m}) \in \mathsf{creds}$ such that $\mathbf{m}[i] = m_i \ \forall\, i \in V$. // Completeness
- V.5 If $\mathcal{H}$ is corrupt and $b = 1$, set $f = 1$ if $(\mathcal{H}', \mathcal{C}, \mathbf{m}) \in \mathsf{creds}$ such that $\mathbf{m}[i] = m_i \ \forall\, i \in V$ and some corrupt party $\mathcal{H}'$. // Corrupt holders can exchange their credentials.
- V.6 If $\mathcal{H}$ and $\mathcal{I}$ are both corrupt and $b = 1$, set $f = 1$. // No unforgeability if issuer and holder collude.
- V.7 Delete record $\mathsf{SR}(\mathsf{vid})$ and send a delayed output $(\mathsf{VERIFIED}, \mathsf{vid}, f)$ to $\mathcal{V}$.

Figure 4: Functionalities $\mathcal{F}_{\mathsf{cbAC}}^{\mathsf{blind}}$ and $\mathcal{F}_{\mathsf{cbAC}}$ for card-based anonymous credentials. Instructions in $\boxed{\text{boxes}}$ appear only in $\mathcal{F}_{\mathsf{cbAC}}^{\mathsf{blind}}$ which allows the holder to contribute (blind) attributes during join. Instructions in $\dbox{\text{dashed boxes}}$ appear only in $\mathcal{F}_{\mathsf{cbAC}}$ where holders do not contribute attributes during join.

The Join phase. The join phase involves three parties: the issuer I, a holder $\mathcal{H}$, and a card $\mathcal{C}$. To initiate an attribute issuance, all three parties need to call their corresponding interfaces (JOIN for $\mathcal{H}$, JOINID for $\mathcal{C}$, JOINISSUE for I) with the same session identifier jid, and $\mathcal{F}_{\mathsf{cbAC}}$ does not enforce any order in which these calls happen J-H.1, J-C.1, J-I.2. Our modeling leaves the agreement on session identifiers jid to the application, but we only put minimal requirements to these identifiers: they are publicly known, and need not be unique ($\mathcal{F}_{\mathsf{cbAC}}$ allows the reuse of jid identifiers after the completion of join session jid J.5. To proceed, $\mathcal{F}_{\mathsf{cbAC}}$ records the identities of the holder and the card J-H.1, J-C.1 in form of a JR record for join session jid. In this record, which maintains the state of a single join session, $\mathcal{F}^{\mathsf{blind}}_{\mathsf{cbAC}}$ also collects attributes from the holder J-H.1 ($\mathcal{F}_{\mathsf{cbAC}}$ skips this step) and the issuer J-I.2. While the holder learns the attributes that the issuer suggests to issue J-I.3, in $\mathcal{F}^{\mathsf{blind}}_{\mathsf{cbAC}}$ the issuer is oblivious of the attributes contributed by the holder (this can be seen from the absence of any output towards the issuer), enabling **blind issuance of attributes**. Finally, the issuance of the attributes is completed by the adversary calling JOINCOMPLETE with identifier jid indicating which join session to complete. $\mathcal{F}_{\mathsf{cbAC}}$ now assembles the ordered attribute vector $(m_1, \ldots, m_\ell)$ from the attributes provided by holder and issuer in join session jid (by retrieving them from the corresponding JR record), and aborts if any attribute is missing, or both the holder and the issuer gave an attribute m_i for the same index $i \in [\ell]$. Otherwise, $\mathcal{F}_{\mathsf{cbAC}}$ adds a tuple $(\mathcal{H}, \mathcal{C}, \mathbf{m})$ to a list called creds, which keeps track of successful join requests. The holder then receives a confirmation message.

The presentation phase. The presentation phase, which allows for attribute-based authentication, involves a holder $\mathcal{H}$, a card $\mathcal{C}$, and a verifier $\mathcal{V}$. To initiate a presentation, all three parties need to call the corresponding interfaces (SETATTRS for $\mathcal{V}$, SETID for $\mathcal{C}$, and SETCRED for $\mathcal{H}$) with the same session identifier vid. $\mathcal{F}_{\mathsf{cbAC}}$ does not enforce any order in which these interfaces must be called P-V.1, P-C.1, P-H.1, and again vid can be public and reused after successful completion of the presentation V.7. $\mathcal{F}_{\mathsf{cbAC}}$ records the identities of all three participants in a record SR for presentation session vid P-V.1, P-C.1, P-H.1. $\mathcal{F}_{\mathsf{cbAC}}$ also adds to this record the attributes $\mathbf{a}_V$ that the verifier $\mathcal{V}$ contributed P-V.1. $\mathbf{a}_V$ may contain less than ℓ attributes V.2, which enables **selective disclosure** of certain attributes. With this, the parameters of the presentation session vid are complete, and the question is now whether $\mathcal{C}$ and $\mathcal{H}$ are capable of presenting the attributes $\mathbf{a}_V$ or not. To resolve this question, $\mathcal{F}_{\mathsf{cbAC}}$ first waits for the adversary to signal completion of the presentation session by calling VERIFYCOMPLETE. Then, $\mathcal{F}_{\mathsf{cbAC}}$ outputs 1 to the verifier $\mathcal{V}$ if there is a record $(\mathcal{H}, \mathcal{C}, \mathbf{m})$ in list creds where $\mathbf{a}_V \subseteq \mathbf{m}$ V.4,V.5, or if V.6 holder and issuer collude (are both corrupt) and can hence present any credential. Since existence of this record means that $\mathcal{H}$ and $\mathcal{C}$ successfully completed a join session for an attribute set $\mathbf{m}$ which contains $\mathbf{a}_V$, **completeness** holds. If no such record is found, $\mathcal{F}_{\mathsf{cbAC}}$ gives the verifier a 0 V.3,V.7, ensuring **unforgeability of presentations** for attributes that were not previously issued. The fact that the verifier only learns one bit from each presentation session implies that $\mathcal{F}_{\mathsf{cbAC}}$ guarantees **holder privacy during presentation** and **unlinkability of presentations**. This concludes the honest walk-through of $\mathcal{F}_{\mathsf{cbAC}}$.

Showcasing formal implication of game-based properties. In Section 5 we presented game-based properties of a *cbAC* scheme. We showcase here how to formally verify that $\mathcal{F}_{\mathsf{cbAC}}$ implies these properties, for the case of unforgeability. The claim is that a protocol that UC-emulates $\mathcal{F}_{\mathsf{cbAC}}$ satisfies unforgeability as defined in Section 5. To prove this claim, we assume that the protocol is not unforgeable, and construct a distinguisher $\mathcal{Z}$ between a protocol realizing $\mathcal{F}_{\mathsf{cbAC}}$ and $\mathcal{F}_{\mathsf{cbAC}}$ together with some simulator $\mathcal{S}$. If the protocol is not unforgeable, there is a presentation of attributes for which there exist no corresponding (honest: same, corrupt: different) holder $\mathcal{H}$ who got issued a credential for these attributes. In the real protocol execution, the forgeability implies that the verifier outputs 1. In the ideal world, however, the non-existence of the appropriate inputs to the holder $\mathcal{H}$ imply that $\mathcal{F}_{\mathsf{cbAC}}$ does not have a corresponding entry in the creds table, and hence due to V.3 and the issuer I being honest, we have $f = 0$. Hence, the real protocol run (verifier outputs 1) is distinguishable from the ideal run ($f = 0$) in case the protocol does not satisfy unforgeability.

Adversarial leakage and influence. The adversary is allowed to learn the identities of the card and the holder during the issuing of attributes J-H.2, J-C.2 as well as the identity of the verifier during presentation P-V.2, but remains oblivious of the identities of the card and the holder in any presentation phase P-C.3, P-H.2. This means $\mathcal{F}_{\mathsf{cbAC}}$ cannot be realized by protocols where the presentation comprises the sending of, e.g., signatures under a long-term public key held by the card, or other identifying information. $\mathcal{F}_{\mathsf{cbAC}}$ hence guarantees **anonymity during presentation**. However, $\mathcal{F}_{\mathsf{cbAC}}$ does leak the identity of a card participating in a presentation session to a corrupt holder in case the holder has already used this card in a previous join session. The reason is that we cannot expect this information to remain hidden from a corrupt holder, as holders can always recognize "their" cards by running the verification protocol with the card and all credentials that they have been previously issued.

The adversary further has full control over when a party receives outputs due to its ability to delay outputs S.3, J-I.3, J.5, V.7. The adversary can cause any join session jid to fail by not calling (JOINCOMPLETE, jid). The adversary can make any presentation session vid fail (i.e., cause the verifier to output 0) by sending (VERIFYCOMPLETE, vid, 0)

V.3-V.5 . These attacks correspond to Denial-of-Service attacks due to, e.g., message tampering.

Functionality $\mathcal{F}_{\mathsf{cbAC}}$ does not have separate interfaces for party corruption. Instead, we use the corruption mechanism built within the UC framework [29], which lets the adversary corrupt parties by sending them a special corruption message. $\mathcal{F}_{\mathsf{cbAC}}$ is aware of such corruption, and can hence act upon it. While corrupt cards or verifiers are not treated differently from honest ones, $\mathcal{F}_{\mathsf{cbAC}}$ provides corrupt holders with more freedom than honest holders. Namely, corrupt holders may exchange their credentials among each other, and hence $\mathcal{F}_{\mathsf{cbAC}}$ allows corrupt holders to "use" a credential list entry of any other corrupt holder V.5 . Further, in case the credential issuer and the holder are both corrupt, $\mathcal{F}_{\mathsf{cbAC}}$ does not guarantee unforgeability anymore, which is reflected in V.6 where arbitrary attributes can be presented *if* holders and credential issuers team up.

7 Our card-based Anonymous Credential scheme

In this section we describe how we realise the three interactions described in Section 2.2: credentials received during the join phase are BBS+ signature from the credential issuer on a message vector $\mathbf{m} = (\mathsf{uid}, m_1, \ldots, m_\ell)$, where uid is a unique secret identifier of the card, and $m_1, \ldots, m_\ell$ represent the attributes of the holder. Issuing the credential in practice amounts to generating a blind signature over a set of shown and hidden messages. The set of hidden messages will contain at least the term uid, and possibly, but not mandatorily, other sensitive attributes that the holder does not want to reveal to the credential issuer.

During the presentation phase the holder discloses a set of certified attributes and proves knowledge of the complement of that set. This phase instantiates the joint proof of knowledge of a BBS+ signature of Figure 3, to let a holder who ignores the term uid generate a valid proof of knowledge of a BBS+ signature that contains it. This is enabled by the presence of some card $\mathcal{C}$.

One feature of our protocol is that cards never need to communicate with holders directly. This greatly simplifies the card's interface during a presentation session, and avoids the deployment of expensive authentication mechanisms that let cards distinguish between holders and verifiers.

We depict our protocol in Figures 5 (Join) and 6 (Presentation), where card and holder call the joint BBS+ proving algorithms from Figure 2. Note that Figure 6 is essentially a visualization of Figure 3. These figures do not show the setup phase and interaction with trusted parties, and they are cleaned from any "cluttering" that is introduced by the UC framework. We state the full formal protocol description as executed in the UC framework (i.e., using the same interfaces of functionality $\mathcal{F}_{\mathsf{cbAC}}$, including UC artifacts such as session identifiers) in the appendix, B.2 (Setup), B.3 (Join), and B.4 (Presentation).

7.1 Security

Theorem 7.1 (cbAC security without blind issuance)**.** *The cbAC construction of Section 7 UC-emulates the functionality $\mathcal{F}_{\mathsf{cbAC}}$ parametrized with $\ell \in \mathbb{N}$ in the $(\mathcal{F}_{\mathsf{crs}}, \mathcal{F}_{\mathsf{cardAuth}})$-hybrid model under the following assumptions:*

- *The adversary does not corrupt any $\mathcal{C}$ that enters the join phase. All other corruptions are static and malicious.*
- *Holder inputs are restricted to $\mathbf{a}_H = \emptyset$ in* JOIN.
- *All channels are secure. Channels during presentation are additionally holder- and card-anonymous.*
- PRF *is a pseudorandom function with key space $\{0,1\}^\lambda$, and* SDL *is a signature of knowledge for relation $\mathcal{R}_{s,k,n}$ (see 3.2.1).*
- *Computation of discrete logarithms is hard in $\mathbb{G}_1$ and the qSDH assumption holds in* BG.

Our construction supports blind issuance of attributes under a stronger assumption on the proof system SDL, namely *online extractability*, which we describe in Appendix A.

Theorem 7.2 (cbAC security with blind issuance)**.** *Theorem 7.1 holds for $\mathcal{F}^{\mathsf{blind}}_{\mathsf{cbAC}}$ if additionally* SDL *is online extractable, and the restriction on holder inputs is dropped.*

The full formal proof and simulator code are deferred to Appendix C.2.

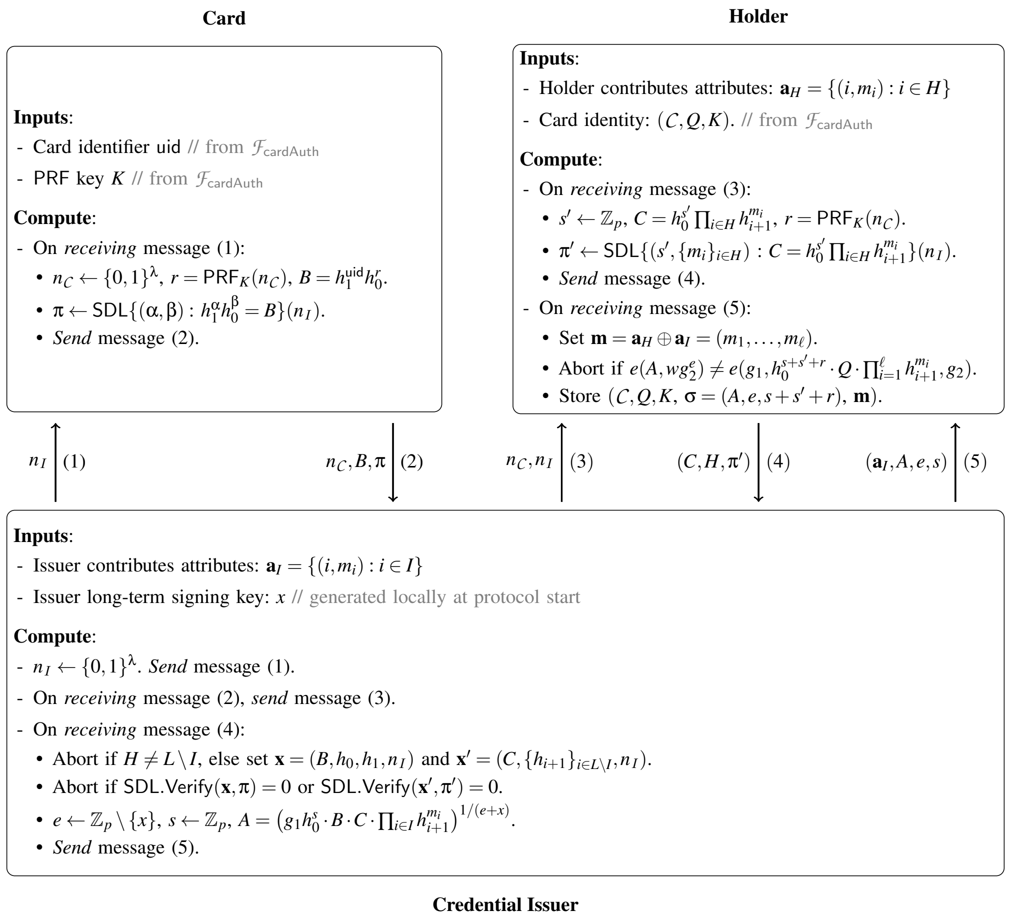
8 Evaluation

We implement the card-specific part of the scheme and test it on real smartcards to establish the scheme's practicality and determine its performance. We also implement a simplified verifier to determine whether the cards' messages are properly constructed and can be verified successfully[3].

We choose to implement the card as a Javacard [35] applet. Javacard is a Java framework for smartcards supporting a subset of the Java runtime. The applet supports byte-level I/O through smartcard application protocol data unit (APDUs). APDUs can contain selectors for different functions, and the applet is structured to handle the different functions. The Javacard framework supports operations on elliptic curves. Points on various elliptic curves might be built by selecting the curve type ($\mathbf{F}_p$ or $\mathbf{F}_{2^m}$) and specifying the relevant parameters. We structure our applet with the 5 following functions (with reference to Figure 2):

- a set of initialisation functions: SETUP, where the invoker sets the parameters of the curve. We choose to test

[3]We refrain from implementing issuer, holder and verifier since their practicality and performance has already been established by prominent open source projects such as Hyperledger Ursa [42]



Card

```
Inputs:
- Card identifier uid // from $\mathcal{F}_{\mathsf{cardAuth}}$
- PRF key $K$ // from $\mathcal{F}_{\mathsf{cardAuth}}$
Compute:
- On receiving message (1):
    • $n_C \leftarrow \{0,1\}^\lambda$, $r = \mathsf{PRF}_K(n_C)$, $B = h_1^{\mathsf{uid}} h_0^r$.
    • $\pi \leftarrow \mathsf{SDL}\{(\alpha,\beta) : h_1^\alpha h_0^\beta = B\}(n_I)$.
    • Send message (2).
```

Holder

```
Inputs:
- Holder contributes attributes: $\mathbf{a}_H = \{(i, m_i) : i \in H\}$
- Card identity: $(\mathcal{C}, Q, K)$. // from $\mathcal{F}_{\mathsf{cardAuth}}$
Compute:
- On receiving message (3):
    • $s' \leftarrow \mathbb{Z}_p$, $C = h_0^{s'} \prod_{i \in H} h_{i+1}^{m_i}$, $r = \mathsf{PRF}_K(n_C)$.
    • $\pi' \leftarrow \mathsf{SDL}\{(s', \{m_i\}_{i \in H}) : C = h_0^{s'} \prod_{i \in H} h_{i+1}^{m_i}\}(n_I)$.
    • Send message (4).
- On receiving message (5):
    • Set $\mathbf{m} = \mathbf{a}_H \oplus \mathbf{a}_I = (m_1, \ldots, m_\ell)$.
    • Abort if $e(A, wg_2^e) \neq e(g_1, h_0^{s+s'+r} \cdot Q \cdot \prod_{i=1}^{\ell} h_{i+1}^{m_i}, g_2)$.
    • Store $(\mathcal{C}, Q, K, \sigma = (A, e, s + s' + r), \mathbf{m})$.
```

Credential Issuer

```
Inputs:
- Issuer contributes attributes: $\mathbf{a}_I = \{(i, m_i) : i \in I\}$
- Issuer long-term signing key: $x$ // generated locally at protocol start
Compute:
- $n_I \leftarrow \{0,1\}^\lambda$. Send message (1).
- On receiving message (2), send message (3).
- On receiving message (4):
    • Abort if $H \neq L \setminus I$, else set $\mathbf{x} = (B, h_0, h_1, n_I)$ and $\mathbf{x}' = (C, \{h_{i+1}\}_{i \in L \setminus I}, n_I)$.
    • Abort if $\mathsf{SDL.Verify}(\mathbf{x}, \pi) = 0$ or $\mathsf{SDL.Verify}(\mathbf{x}', \pi') = 0$.
    • $e \leftarrow \mathbb{Z}_p \setminus \{x\}$, $s \leftarrow \mathbb{Z}_p$, $A = \left(g_1 h_0^s \cdot B \cdot C \cdot \prod_{i \in I} h_{i+1}^{m_i}\right)^{1/(e+x)}$.
    • Send message (5).
```

Figure 5: Outline of join protocol detailed in Section B.3. The protocol delivers a credential certifying attributes assembled from holder input $\mathbf{a}_H$, issuer's input $\mathbf{a}_I$ for holder and the card identified by its uid.

on Fp256BN curve [32][4] and so the input APDU for the setup contains the value of the a and b coefficients, the value of field and order of the curve, and coordinates and cofactor of the generator; SETBASE where the invoker sets the public bases h_0 and h_1; SETUID where the invoker sets the secret value of m_1; SETSEED where the invoker sets the value of K, the PRF seed used by the card. We use AES in ECB mode to instantiate a PRF with domain and codomain of all 128-bit strings.

- a RUN function that executes $\mathsf{Prove}_{\mathcal{C}}(K, m_1, n_1, n)$; K and m_1 are already set by SETSEED and SETUID, respectively, so the input of the invocation are the verifier nonce n_1 and the PRF input n.

We assume that the initialisation functions can be invoked once before the card is issued. RUN can instead be invoked arbitrarily many times by whoever is in proximity of the card. RUN requires no access control and always responds in the same way, irrespective of the identity of the invoker.

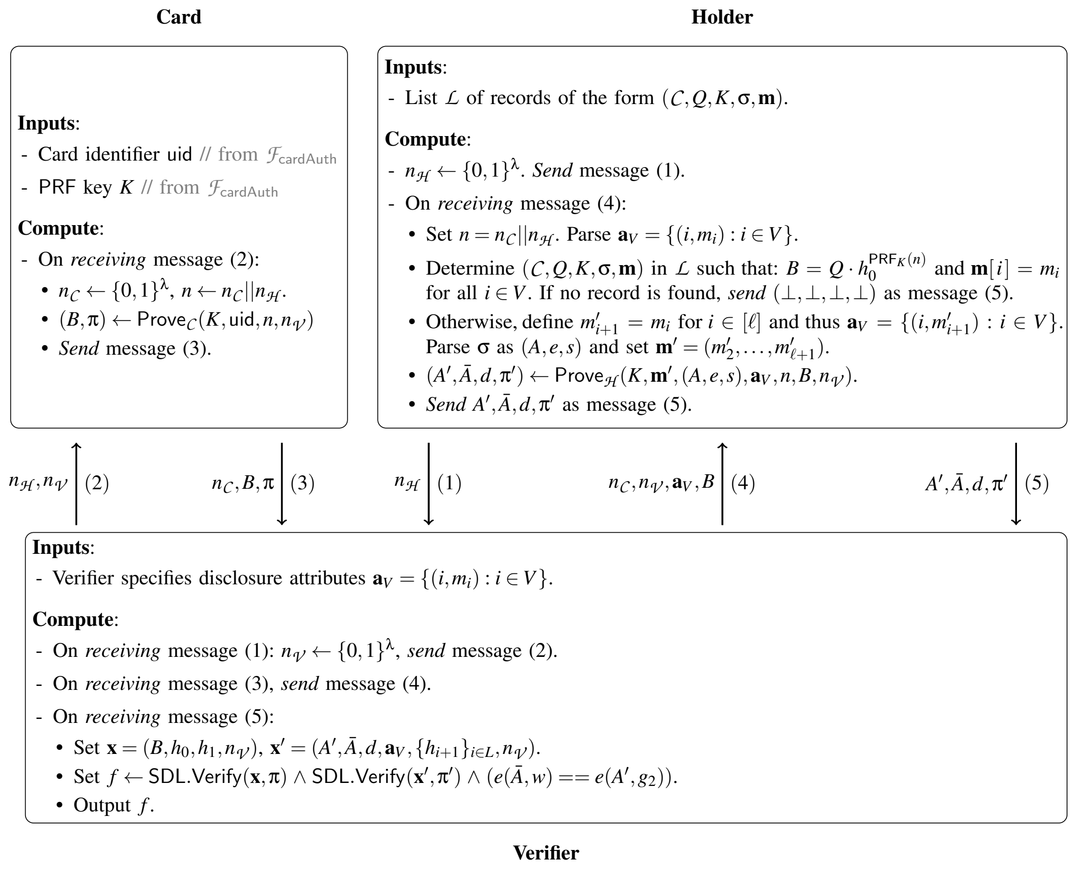
For the deployment, we choose NXP 1ID white plastic cards with a Smart MX D600 chip (400KiB of available memory) running JCOP 4.5 OS with NXP's JCOPx extensions at version 1.1.4[5]. Cards have a dual interface (6 PIN contact, 1ID inlay 56pf contactless on input CAP). We use the contactless communication channel and rely on a uTrust 3700 F as a PCSC reader to program the cards, communicate with them and benchmark them. Alternative designs may employ the native NFC capabilities available in most modern mobile platforms.

We benchmark the RUN algorithm by executing it 100 times and determining average and standard deviation of all samples. To determine the breakdown of the running time we also benchmark a no-op version of the RUN algorithm where only the I/O part is implemented.

The RUN function completes on average in 185.08ms with a standard deviation of 4.06ms. This is an end-to-end mea-

[4]We choose Fp256BN given the sundry available implementations, even though this curve no longer offers 128 bits of security.

[5]The extensions are needed to directly access the low-level API to perform scalar point multiplication.


Card

Inputs:
- Card identifier uid // from $\mathcal{F}_{\mathsf{cardAuth}}$
- PRF key K // from $\mathcal{F}_{\mathsf{cardAuth}}$

Compute:
- On *receiving* message (2):
  - $n_{\mathcal{C}} \leftarrow \{0,1\}^{\lambda}$, $n \leftarrow n_{\mathcal{C}} || n_{\mathcal{H}}$.
  - $(B,\pi) \leftarrow \mathsf{Prove}_{\mathcal{C}}(K,\mathsf{uid},n,n_{\mathcal{V}})$
  - *Send* message (3).

Holder

Inputs:
- List $\mathcal{L}$ of records of the form $(\mathcal{C},Q,K,\sigma,\mathbf{m})$.

Compute:
- $n_{\mathcal{H}} \leftarrow \{0,1\}^{\lambda}$. *Send* message (1).
- On *receiving* message (4):
  - Set $n = n_{\mathcal{C}} || n_{\mathcal{H}}$. Parse $\mathbf{a}_V = \{(i,m_i) : i \in V\}$.
  - Determine $(\mathcal{C},Q,K,\sigma,\mathbf{m})$ in $\mathcal{L}$ such that: $B = Q \cdot h_0^{\mathsf{PRF}_K(n)}$ and $\mathbf{m}[i] = m_i$ for all $i \in V$. If no record is found, *send* $(\perp,\perp,\perp,\perp)$ as message (5).
  - Otherwise, define $m'_{i+1} = m_i$ for $i \in [\ell]$ and thus $\mathbf{a}_V = \{(i,m'_{i+1}) : i \in V\}$. Parse σ as (A,e,s) and set $\mathbf{m}' = (m'_2,\ldots,m'_{\ell+1})$.
  - $(A',\bar{A},d,\pi') \leftarrow \mathsf{Prove}_{\mathcal{H}}(K,\mathbf{m}',(A,e,s),\mathbf{a}_V,n,B,n_{\mathcal{V}})$.
  - *Send* $A',\bar{A},d,\pi'$ as message (5).

$n_{\mathcal{H}},n_{\mathcal{V}}$ (2)
$n_{\mathcal{C}},B,\pi$ (3)
$n_{\mathcal{H}}$ (1)
$n_{\mathcal{C}},n_{\mathcal{V}},\mathbf{a}_V,B$ (4)
$A',\bar{A},d,\pi'$ (5)

Inputs:
- Verifier specifies disclosure attributes $\mathbf{a}_V = \{(i,m_i) : i \in V\}$.

Compute:
- On *receiving* message (1): $n_{\mathcal{V}} \leftarrow \{0,1\}^{\lambda}$, *send* message (2).
- On *receiving* message (3), *send* message (4).
- On *receiving* message (5):
  - Set $\mathbf{x} = (B,h_0,h_1,n_{\mathcal{V}})$, $\mathbf{x}' = (A',\bar{A},d,\mathbf{a}_V,\{h_{i+1}\}_{i\in L},n_{\mathcal{V}})$.
  - Set $f \leftarrow \mathsf{SDL.Verify}(\mathbf{x},\pi) \wedge \mathsf{SDL.Verify}(\mathbf{x}',\pi') \wedge (e(\bar{A},w) == e(A',g_2))$.
  - Output f.

Verifier

Figure 6: Outline of presentation protocol detailed in Section B.4. The algorithms $\mathsf{Prove}_{\mathcal{C}}$ and $\mathsf{Prove}_{\mathcal{H}}$ are as in Figure 2. In the above protocol, verifier specifies presentation predicate as $\mathbf{a}_V$. The holder generates proof of possession of a credential issued against the participating card.

surement that includes APDU I/O, parsing, crypto and receipt of the response APDU. To determine a breakdown of the running time we construct a no-op version of the RUN function which is identical to the original but for the fact that all cryptographic processing is removed, thus resulting only in the APDU I/O, parsing and producing a response message of identical length to the original. The no-op version completes on average in 24.53ms with a standard deviation of 4.3ms.

These result confirm that the performance of the scheme is perfectly in line with that of other NFC-driven interactions users engage in on a daily basis (e.g. contactless payments), thus confirming the viability of our approach and its deployment readiness.

9 Conclusion

We present *Anonymous Credentials with Visual Holder Authentication*, a system that permits secure and privacy preserving verification of digital credentials. This system enhances digital anonymous credential system by permitting verifiers to determine whether the holder presenting the credential is its legitimate owner. This determination can be performed without any deterioration of the privacy guarantees offered by the underlying anonymous credential system. The key idea is to introduce plastic cards from a trusted issuer (e.g. a government), playing role in both the physical (verifier inspects the picture on the card) and digital (verification requires contributions from both card and holder) side of the authentication.

To realise this system we present a primitive for joint proof of knowledge for BBS+ signatures, which is both central for our construction and of independent interest. We then formally define card-based Anonymous Credentials, our cryptographic building block, and analyse its security in the Universal Composability (UC) framework. We further implement the performance-sensitive aspects of our system, namely all interactions of the card, to determine real-world viability.

Our system achieves several desirable properties: i) it is compatible with BBS+ public keys and signatures, so that implementers are able to leverage the vast body of open-source projects handling those artefacts; ii) it keeps the design of the card simple and minimalistic, avoiding complex access control on the card; iii) it maintains the familiar pattern of authentication with phone and physical ID; iv) it achieves ideal privacy by not forcing the disclosure of unnecessary holder attributes to determine a match with physical ID.

Acknowledgements

The authors would like to thank: the anonymous reviewers for the helpful and constructive comments; Ilie Circiumaru, Jim Coon, Todd Nuzum and Nick Nurse for their help in setting up the evaluation environment. Alessandro wishes to say *"arrivederci, mio meraviglioso papà, e grazie per tutto"*.

References

[1] M. H. Au, W. Susilo, and Y. Mu. Constant-size dynamic k-TAA. In R. D. Prisco and M. Yung, editors, *SCN 06: 5th International Conference on Security in Communication Networks*, volume 4116 of *Lecture Notes in Computer Science*, pages 111–125, Maiori, Italy, Sept. 6–8, 2006. Springer, Heidelberg, Germany.

[2] M. Backes, L. Hanzlik, K. Kluczniak, and J. Schneider. Signatures with flexible public key: Introducing equivalence classes for public keys. In T. Peyrin and S. Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 405–434, Brisbane, Queensland, Australia, Dec. 2–6, 2018. Springer, Heidelberg, Germany.

[3] A. Barki, S. Brunet, N. Desmoulins, S. Gambs, S. Gharout, and J. Traoré. Private eCash in practice (short paper). In J. Grossklags and B. Preneel, editors, *FC 2016: 20th International Conference on Financial Cryptography and Data Security*, volume 9603 of *Lecture Notes in Computer Science*, pages 99–109, Christ Church, Barbados, Feb. 22–26, 2016. Springer, Heidelberg, Germany.

[4] L. Batina, J. Hoepman, B. Jacobs, W. Mostowski, and P. Vullers. Developing efficient blinded attribute certificates on smart cards via pairings. In D. Gollmann, J. Lanet, and J. Iguchi-Cartigny, editors, *Smart Card Research and Advanced Application, 9th IFIP WG 8.8/11.2 International Conference, CARDIS 2010, Passau, Germany, April 14-16, 2010. Proceedings*, volume 6035 of *Lecture Notes in Computer Science*, pages 209–222. Springer, 2010.

[5] A. Beduschi. Rethinking digital identity for post-covid-19 societies: Data privacy and human rights considerations. *Data & Policy*, 3, 2021.

[6] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications Security*, pages 390–399, Alexandria, Virginia, USA, Oct. 30 – Nov. 3, 2006. ACM Press.

[7] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, Nov. 3–5, 1993. ACM Press.

[8] D. Bernhard, M. Fischlin, and B. Warinschi. Adaptive proofs of knowledge in the random oracle model. In J. Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 629–649, Gaithersburg, MD, USA, Mar. 30 – Apr. 1, 2015. Springer, Heidelberg, Germany.

[9] P. Bichsel, J. Camenisch, T. Groß, and V. Shoup. Anonymous credentials on a standard java card. In E. Al-Shaer, S. Jha, and A. D. Keromytis, editors, *ACM CCS 2009: 16th Conference on Computer and Communications Security*, pages 600–610, Chicago, Illinois, USA, Nov. 9–13, 2009. ACM Press.

[10] J. Bobolz, F. Eidens, S. Krenn, S. Ramacher, and K. Samelin. Issuer-hiding attribute-based credentials. *IACR Cryptol. ePrint Arch.*, page 213, 2022.

[11] D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, Apr. 2008.

[12] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55, Santa Barbara, CA, USA, Aug. 15–19, 2004. Springer, Heidelberg, Germany.

[13] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *2012 IEEE Symposium on Security and Privacy*, pages 553–567, San Francisco, CA, USA, May 21–23, 2012. IEEE Computer Society Press.

[14] D. Bosk, D. Frey, M. Gestin, and G. Piolle. Hidden issuer anonymous credential. *Proc. Priv. Enhancing Technol.*, 2022(4):571–607, 2022.

[15] D. Bosk, D. Frey, M. Gestin, and G. Piolle. Hidden issuer anonymous credential. *Proc. Priv. Enhancing Technol.*, 2022(4):571–607, 2022.

[16] S. A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000.

[17] E. Brickell, L. Chen, and J. Li. Simplified security notions of direct anonymous attestation and a concrete scheme from pairings. Cryptology ePrint Archive, Report 2008/104, 2008. https://eprint.iacr.org/2008/104.

[18] E. Brickell and J. Li. Enhanced privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. Cryptology ePrint Archive, Report 2007/194, 2007. https://eprint.iacr.org/2007/194.

[19] E. F. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In V. Atluri, B. Pfitzmann, and P. McDaniel, editors, *ACM CCS 2004: 11th Conference on Computer and Communications Security*, pages 132–145, Washington, DC, USA, Oct. 25–29, 2004. ACM Press.

[20] J. Camenisch, M. Drijvers, and M. Dubovitskaya. Practical uc-secure delegatable credentials with attributes and their application to blockchain. In B. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 683–699. ACM, 2017.

[21] J. Camenisch, M. Drijvers, P. Dzurenda, and J. Hajny. Fast keyed-verification anonymous credentials on standard smart cards. Cryptology ePrint Archive, Report 2019/460, 2019. https://eprint.iacr.org/2019/460.

[22] J. Camenisch, M. Drijvers, and A. Lehmann. Anonymous attestation using the strong Diffie Hellman assumption revisited. Cryptology ePrint Archive, Report 2016/663, 2016. https://eprint.iacr.org/2016/663.

[23] J. Camenisch, M. Drijvers, and A. Lehmann. Universally composable direct anonymous attestation. In C.-M. Cheng, K.-M. Chung, G. Persiano, and B.-Y. Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 9615 of *Lecture Notes in Computer Science*, pages 234–264, Taipei, Taiwan, Mar. 6–9, 2016. Springer, Heidelberg, Germany.

[24] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany.

[25] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In S. Cimato, C. Galdi, and G. Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289, Amalfi, Italy, Sept. 12–13, 2003. Springer, Heidelberg, Germany.

[26] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72, Santa Barbara, CA, USA, Aug. 15–19, 2004. Springer, Heidelberg, Germany.

[27] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In B. S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424, Santa Barbara, CA, USA, Aug. 17–21, 1997. Springer, Heidelberg, Germany.

[28] J. Camenisch and E. Van Herreweghen. Design and implementation of the idemix anonymous credential system. In V. Atluri, editor, *ACM CCS 2002: 9th Conference on Computer and Communications Security*, pages 21–30, Washington, DC, USA, Nov. 18–22, 2002. ACM Press.

[29] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, Oct. 14–17, 2001. IEEE Computer Society Press.

[30] M. Chase, S. Meiklejohn, and G. Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In G.-J. Ahn, M. Yung, and N. Li, editors, *ACM CCS 2014: 21st Conference on Computer and Communications Security*, pages 1205–1216, Scottsdale, AZ, USA, Nov. 3–7, 2014. ACM Press.

[31] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. 28(10), 1985.

[32] I. J. S. . T. Committee. ISO/IEC 15946-5:2022: Cryptographic techniques based on elliptic curves. https://www.iso.org/standard/80241.html, 2022.

[33] A. Connolly, P. Lafourcade, and O. P. Kempner. Improved constructions of anonymous credentials from structure-preserving signatures on equivalence classes. Cryptology ePrint Archive, Report 2021/1680, 2021. https://eprint.iacr.org/2021/1680.

[34] V. core development. Veramo. https://github.com/uport-project/veramo, 2023.

[35] O. Corporation. Oracle java card technology. 2023.

[36] A. de la Piedra, J. Hoepman, and P. Vullers. Towards a full-featured implementation of attribute based credentials on smart cards. volume 8813 of *Lecture Notes in Computer Science*, pages 270–289. Springer, 2014.

[37] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

[38] J. Doerner, Y. Kondi, E. Lee, abhi shelat, and L. Tyner. Threshold bbs+ signatures for distributed anonymous credential issuance. Cryptology ePrint Archive, Paper 2023/602, 2023. https://eprint.iacr.org/2023/602.

[39] eHealth Network. Interoperability of health certificates: Trust framework. 2021.

[40] M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 152–168, Santa Barbara, CA, USA, Aug. 14–18, 2005. Springer, Heidelberg, Germany.

[41] H. Foundation. Hyperledger indy. https://github.com/hyperledger/indy-node, 2023.

[42] H. Foundation. Hyperledger ursa. https://github.com/hyperledger/ursa, 2023.

[43] H. Gunasinghe and E. Bertino. Privbiomtauth: Privacy preserving biometrics-based and user centric protocol for user authentication from mobile phones. *IEEE Trans. Inf. Forensics Secur.*, 13(4):1042–1057, 2018.

[44] C. Hanser and D. Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In P. Sarkar and T. Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 491–511, Kaoshiung, Taiwan, R.O.C., Dec. 7–11, 2014. Springer, Heidelberg, Germany.

[45] L. Hanzlik and D. Slamanig. With a little help from my friends: Constructing practical anonymous credentials. In G. Vigna and E. Shi, editors, *ACM CCS 2021: 28th Conference on Computer and Communications Security*, pages 2004–2023, Virtual Event, Republic of Korea, Nov. 15–19, 2021. ACM Press.

[46] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In J. Motiwalla and G. Tsudik, editors, *ACM CCS 99: 6th Conference on Computer and Communications Security*, pages 28–36, Singapore, Nov. 1–4, 1999. ACM Press.

[47] M. Khalili, D. Slamanig, and M. Dakhilalian. Structure-preserving signatures on equivalence classes from standard assumptions. Cryptology ePrint Archive, Report 2019/1120, 2019. https://eprint.iacr.org/2019/1120.

[48] T. Looker, V. Kalos, A. Whitehead, and M. Lodder. The BBS Signature Scheme. Internet-Draft draft-irtf-cfrg-bbs-signatures-01, Internet Engineering Task Force, Oct. 2022. Work in Progress.

[49] W. Lueks, B. Hampiholi, G. Alpár, and C. Troncoso. Tandem: Securing keys by using a central server while preserving privacy. *Proc. Priv. Enhancing Technol.*, 2020(3):327–355, 2020.

[50] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H. M. Heys and C. M. Adams, editors, *SAC 1999: 6th Annual International Workshop on Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199, Kingston, Ontario, Canada, Aug. 9–10, 1999. Springer, Heidelberg, Germany.

[51] MATTR. bbs-signatures. *GitHub repository*, 2023.

[52] W. Mostowski and P. Vullers. Efficient u-prove implementation for anonymous credentials on smart cards. In M. Rajarajan, F. Piper, H. Wang, and G. Kesidis, editors, *Security and Privacy in Communication Networks - 7th International ICST Conference, SecureComm 2011, London, UK, September 7-9, 2011, Revised Selected Papers*, volume 96 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 243–260. Springer, 2011.

[53] W. Mostowski and P. Vullers. Efficient u-prove implementation for anonymous credentials on smart cards. In *Security and Privacy in Communication Networks*, pages 243–260. Springer Berlin Heidelberg, 2012.

[54] C. Paquin and G. Zaverucha. U-prove cryptographic specification v1.1 (revision 3), December 2013.

[55] R. Pass. On deniability in the common reference string and random oracle model. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 316–337, Santa Barbara, CA, USA, Aug. 17–21, 2003. Springer, Heidelberg, Germany.

[56] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

[57] L. Rila and C. J. Mitchell. Security protocols for biometrics-based cardholder authentication in smartcards. In J. Zhou, M. Yung, and Y. Han, editors, *ACNS 03: 1st International Conference on Applied Cryptography and Network Security*, volume 2846 of *Lecture Notes in Computer Science*, pages 254–264, Kunming, China, Oct. 16–19, 2003. Springer, Heidelberg, Germany.

[58] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In K. Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 1–16, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany.

[59] A. Sonnino, M. Al-Bassam, S. Bano, S. Meiklejohn, and G. Danezis. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*.

[60] Trinsic. Okapi. https://github.com/trinsic-id/okapi, 2023.

[61] P. Vullers and G. Alpár. Efficient selective disclosure on smart cards using idemix. In S. Fischer-Hübner, E. de Leeuw, and C. J. Mitchell, editors, *Policies and Research in Identity Management - Third IFIP WG 11.6 Working Conference, IDMAN 2013, London, UK, April 8-9, 2013. Proceedings*, volume 396 of *IFIP Advances in Information and Communication Technology*, pages 53–67. Springer, 2013.

A Non Interactive Random Oracle Arguments

In this section we formally define non-interactive zero-knowledge proofs of knowledge in the random oracle model. Our game based definition of the primitive is taken from [8]. For security parameter λ, a random oracle H provides "oracle" access to a random function $\{0,1\}^* \to \{0,1\}^\lambda$. Alternatively, a random oracle H when queried on previously unqueried input x returns $\mathsf{H}(x)$ as a uniformly sampled value from $\{0,1\}^\lambda$, and returns the same value when x is queried subsequently.

Definition A.1. *A pair $(\mathcal{P}, \mathcal{V})$ of PPT algorithms, where $\mathcal{V}$ is deterministic is called a non-interactive zero-knowledge proof of knowledge for NP relation $\mathcal{R}$ in the random oracle model if it satisfies* completeness, zero-knowledge *and* proof-of-knowledge *as defined below.*

We view $\mathcal{R}$ as $\cup_{\lambda\in\mathbb{N}}\mathcal{R}_\lambda$ where $\mathcal{R}_\lambda$ consists of $(x,w)\in\mathcal{R}$ such that $|x|\leq\lambda$ and thus, $|w|\leq p(\lambda)$ for some polynomial p. We now define the properties mentioned in Definition A.1.

- *Completeness*: We say that the scheme $(\mathcal{P}, \mathcal{V})$ defined above is complete if for all $(x,w)\in\mathcal{R}_\lambda$, $\pi\leftarrow\mathcal{P}^{\mathsf{H}}(x,w)$, $\Pr\left[\mathcal{V}^{\mathsf{H}}(x,\pi)=1\right]\geq 1-negl(\lambda)$ where H is a random oracle. Here the probability is taken over the choices of H and $\mathcal{P}$'s randomness. Both $\mathcal{P}$ and $\mathcal{V}$ can query the random oracle H.

- *Zero-Knowledge*: We say that the scheme $(\mathcal{P}, \mathcal{V})$ defined above is zero-knowledge if there exists a simulator $\mathcal{S}$ such that any PPT adversary $\mathcal{A}$ has negligible advantage in distinguishing between Games G_1 and G_2 as defined below:

$$\Pr\left[\begin{array}{c|r} & (x,w,\mathsf{state})\leftarrow\mathcal{A}_1(1^\lambda) \\ \mathcal{A}_2^{\mathsf{H}_b}(\mathsf{state},\pi_b)=b\,\wedge & \pi_0\leftarrow\mathcal{P}^{\mathsf{H}}(x,w) \\ \mathcal{R}(x,w)=1 & \pi_1\leftarrow\mathcal{S}(x) \\ & b\leftarrow\{0,1\}\end{array}\right]$$

  is negligible in λ, where $\mathsf{H}_0()=\mathsf{H}()$ for $\mathsf{H}\leftarrow\mathcal{U}_\lambda$ and $\mathsf{H}_1()=\mathcal{S}.\mathsf{H}()$.

- *Proof-of-Knowledge* (Online): We say that the scheme $(\mathcal{P}, \mathcal{V})$ is an online proof of knowledge if there exists an extractor $\mathcal{K}$ such that for any prover $\widehat{\mathcal{P}}$ the game G_{online} involving $\mathcal{K}$ and $\widehat{\mathcal{P}}$ as described below outputs 1 with overwhelming probability.

$$\Pr\left[\begin{array}{c|c} w\leftarrow\mathcal{K}^{\mathsf{H},O_{\mathsf{q}}}\,\wedge & \\ \mathcal{R}(x,w)=1\,\wedge & (x,\pi))\leftarrow\widehat{\mathcal{P}}^{\mathsf{H}}(1^\lambda) \\ \mathcal{V}^{\mathsf{H}}(x,\pi)=1 & \end{array}\right]$$

  is overwhelming in λ, where $\mathsf{H}\leftarrow\mathcal{U}_\lambda$ and O_{q} replies with a list of H replies given already to $\widehat{\mathcal{P}}$.

- *Proof-of-Knowledge* (Rewinding): We say that the scheme $(\mathcal{P}, \mathcal{V})$ is a rewinding proof of knowledge if there is an efficient extractor $\mathcal{K}$ such that the Game G_{online} as defined for online proof of knowledge outputs 1 with overwhelming probability for every efficient prover $\widehat{\mathcal{P}}$. Here, the extractor $\mathcal{K}$ is additionally provided "rewound copies" $\widehat{\mathcal{P}}'$ of $\widehat{\mathcal{P}}$ initialized with the same random tape, as the $\widehat{\mathcal{P}}$ in the game G_{online}. This allows $\mathcal{K}$ implement "rewinding" technique to extract witness by answering the random oracle queries of rewound provers.

A.0.1 Random Oracle Proofs of Knowledge of Discrete Log

Let $\ell\in\mathbb{N}$ be fixed. For security parameter λ, let $\mathbb{G}_\lambda$ be a group whose order is a λ-bit prime. Define $\mathcal{R}_\lambda$ to be the relation consisting of pairs $(\mathbf{x},\mathbf{w})$ where $\mathbf{x}=(x,g_0,\ldots,g_\ell)\in\mathbb{G}_\lambda^{\ell+1}$ and $\mathbf{w}=(w_0,\ldots,w_\ell)\in\mathbb{Z}^{\ell+1}$ satisfying $x=\prod_{i=0}^{\ell}g_0^{w_0}$, which we henceforth abbreviate simply as $x=\mathbf{g}^{\mathbf{w}}$ where $\mathbf{g}=(g_0,\ldots,g_\ell)$. Let $\mathcal{R}$ denote the relation $\bigcup_\lambda\mathcal{R}_\lambda$.

Rewinding Proofs of Knowledge: The protocol $(\mathcal{P},\mathcal{V})$ given below is folklore rewinding proof of knowledge protocol for the relation $\mathcal{R}$ in the random oracle model:

- $\mathcal{P}$'s Inputs: $(\mathbf{x},\mathbf{w}) \in \mathcal{R}$, where $\mathbf{x} = (x,\mathbf{g})$ as above.
- $\mathcal{V}$'s Inputs: $\mathbf{x}$.
- Additionally, both $\mathcal{P}$ and $\mathcal{V}$ can query random oracle H.
- Proof Generation: $\mathcal{P}$ computes $\mathbf{r} \leftarrow \mathbb{Z}^{\ell+1}$, $a = \mathbf{g}^{\mathbf{r}}$, $c = \mathsf{H}(\mathbf{x},a)$, $\mathbf{z} = c\mathbf{w}+\mathbf{r}$ and outputs $\pi = (\mathbf{x},a,c,z)$.
- Verification: $\mathcal{V}$ verifies $\pi = (\mathbf{x},a,c,z)$ as (i) check $c = \mathsf{H}(\mathbf{x},a)$ and (ii) $\mathbf{g}^{\mathbf{z}} = x^c \cdot a$.

The proof of knowledge (via rewinding) follows from the fact that underlying sigma protocol is 2-*special sound* (i.e, witness can be computed from two valid transcripts with the same first message, and different challenges), and Forking Lemma ([6, 56]) which guarantees that two such transcripts can be efficiently found for a prover that succeeds with non-negligible probability.

Online Proofs of Knowledge: Transformations of three message sigma protocols, to non-interactive proofs in the random oracle model with online extractors were presented in [40, 55]. While the transformation of [55] applies to any three message sigma protocol, it incurs $O(\lambda^2)$ multiplicative overhead in the proof size. The overhead is avoided using Fischlin's transformation [40], however the transformation only applies to sigma protocols with *quasi-unique responses* (i.e, an efficient adversary produces valid transcripts of the form (x,a,c,z) and (x,a,c,z') with $z \neq z'$ with negligible probability).

B Full protocol description

B.1 Public parameters and writing conventions

We assume all parties have access to an ideal functionality $\mathcal{F}_{\mathsf{crs}}$, which can be queried to obtain the public parameters for the protocol. For security parameter λ, $\mathcal{F}_{\mathsf{crs}}$ generates public parameters consisting of:

- $\mathsf{BG} := (\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,g_1,g_2,g_T,e,p) \leftarrow \mathsf{BGen}(1^\lambda)$
- Generators $g_1 \leftarrow \mathbb{G}_1, h_0,\ldots,h_{\ell+1} \leftarrow \mathbb{G}_1$ for $\ell = poly(\lambda)$, $g_2 \leftarrow \mathbb{G}_2$. Set $L := \{1,\ldots,\ell\}$.
- Session identifier sid'.

We additionally assume that parties have access to a trusted card issuer $\mathcal{F}_{\mathsf{cardAuth}}$ which, on input jid from a card $\mathcal{C}$ and $(\mathsf{jid},\mathcal{C})$ from a holder $\mathcal{H}$, does the following:

- Replies to both parties with $\perp$ if $\mathcal{C}$ has previously received values uid,K
- Replies to both parties with $\perp$ if card and holder do not belong to each other (e.g., by verifying physical appearance against a picture)
- Samples $\mathsf{uid} \leftarrow \mathbb{Z}_p$, $K \leftarrow \{0,1\}^\lambda$ and sends $(\mathsf{jid},\mathsf{uid},K)$ to $\mathcal{C}$ and $(\mathsf{jid},h_1^{\mathsf{uid}},K,\mathcal{C})$ to the holder.

To simplify the protocol presentation, we assume parties to ignore inputs that do not correspond to the format specified by $\mathcal{F}_{\mathsf{cbAC}}$. We also assume that parties drop repeated messages for the same jid,vid, e.g., an issuer I drops a message $(\mathsf{sid},\mathsf{jid},\mathcal{C}',\pi_2')$ from a holder $\mathcal{H}'$ in case he already received a $(\mathsf{sid},\mathsf{jid},\mathcal{C},\pi_2)$ from another holder $\mathcal{H}$. In practise, accidential or adversarial repetition of jid and vid can often be avoided by proximity requirements, e.g., a card and a holder's phone receive vid while being placed on a verification terminal's sensor.

We use the sets $H,V,I \subseteq L$ to denote the indices of attributes specified by the holder, verifier, and issuer.

B.2 Setup

The issuer I generates a BBS+ keypair and shares the verification key and other public parameters with all participants during setup phase.

1. On input $(\mathsf{SETUP},\mathsf{sid})$ the issuer I proceeds as follows:
   - Check that $\mathsf{sid} = (I,\mathsf{sid}')$.
   - Choose $x \leftarrow \mathbb{Z}_p$, $\bar{g}_1 \leftarrow \mathbb{G}_1$. Compute $w \leftarrow g_2^x$.
   - Set $vk := (\bar{g}_1,\bar{g}_1^x,w)$.
   - Output $(\mathsf{SETUPDONE},\mathsf{sid})$.

B.3 Join

The join protocol is between a card $\mathcal{C}$, a holder $\mathcal{H}$ and issuer I and proceeds as (see also Figure 5 for an overview of the protocol flow):

1. On input $(\mathsf{JOINISSUE},\mathsf{sid},\mathsf{jid},\mathbf{a}_I)$ the issuer I:
   - Samples $n_I \leftarrow \{0,1\}^\lambda$ and sends $(\mathsf{sid},\mathsf{jid},n_I)$ to a card.
   - Create a record $\langle \mathsf{sid},\mathsf{jid},\mathbf{a}_I,n_I \rangle$
2. On input $(\mathsf{JOINID},\mathsf{sid},\mathsf{jid})$ the card $\mathcal{C}$:
   - Retrieve tuple (uid,K). If there is no such tuple stored yet, then send jid to $\mathcal{F}_{\mathsf{cardAuth}}$, abort if the reply is $\perp$, otherwise recieve back $(\mathsf{jid},\mathsf{uid},K)$ and store the two latter values.
   - Samples $n_C \leftarrow \{0,1\}^\lambda$ and computes $r = \mathsf{PRF}_K(n_C)$.
   - Computes $B = h_1^{\mathsf{uid}} h_0^r$.
   - On receiving $(\mathsf{sid},\mathsf{jid},n_I)$ from I, $\mathcal{C}$ computes signature proof $\pi \leftarrow \mathsf{SDL}\{(\alpha,\beta) : h_1^\alpha h_0^\beta = B\}(n_I)$.

- Sends $(\mathsf{sid},\mathsf{jid},(n_{\mathcal{C}},B,\pi))$ to I.

3. On receiving message $(\mathsf{sid},\mathsf{jid},(n_{\mathcal{C}},B,\pi))$, the issuer I:
   - Retrieves record $\langle\mathsf{sid},\mathsf{jid},\mathbf{a}_I,n_I\rangle$ and appends $n_{\mathcal{C}},B,\pi$ to it.
   - Sends $(\mathsf{sid},\mathsf{jid},n_{\mathcal{C}},n_I)$ to a holder.

4. On input $(\mathsf{JOIN},\mathsf{sid},\mathsf{jid},\mathcal{C},\mathbf{a}_H)$ the holder $\mathcal{H}$:
   - Retrieve record $(\mathcal{C},Q,K)$. If there is no such tuple, send jid to $\mathcal{F}_{\mathsf{cardAuth}}$. Upon receiving back $\perp$, the holder drops the query. Otherwise, upon receipt of $(\mathsf{jid},Q,K,\mathcal{C})$, the holder stores record $(\mathcal{C},Q,K)$.
   - Parses $\mathbf{a}_H$ as $\{(i,m_i):i\in H\}$
   - On receiving $(\mathsf{sid},\mathsf{jid},n_{\mathcal{C}},n_I)$ from I, $\mathcal{H}$ computes:
     – Samples $s'\leftarrow\mathbb{Z}_p$.
     – Computes $C=h_0^{s'}\prod_{i\in H}h_{i+1}^{m_i}$.
     – Computes signature proof $\pi'\leftarrow\mathsf{SDL}\{(s',\{m_i\}_{i\in H}):C=h_0^{s'}\prod_{i\in H}h_{i+1}^{m_i}\}(n_I)$.
     – Creates record $\langle\mathsf{sid},\mathsf{jid},\mathbf{a}_H,s',n_{\mathcal{C}},n_{\mathcal{V}},C,H,\pi'\rangle$.
     – Sends $(\mathsf{sid},\mathsf{jid},(C,H,\pi'))$ to I.

5. On receiving message $(\mathsf{sid},\mathsf{jid},(C,H,\pi'))$ from $\mathcal{H}$, the issuer I:
   - Retrieves record $\langle\mathsf{sid},\mathsf{jid},\mathbf{a}_I,\mathcal{C},\mathcal{H},n_I,n_{\mathcal{C}},B,\pi\rangle$ and appends C,H,π' to it.
   - Parses $\mathbf{a}_I$ as $\{(i,m_i):i\in I\}$
   - Abort if $H\neq L\setminus I$.
   - Verifies π' as $\mathsf{SDL.Verify}(\mathbf{x}',\pi')\neq 0$ for $\mathbf{x}'=(C,\{h_{i+1}\}_{i\in H},n_I)$. Abort if the proof fails.
   - Sample $e\leftarrow\mathbb{Z}_p\setminus\{x\}$, $s\leftarrow\mathbb{Z}_p$.
   - Compute $A=\left(g_1h_0^s\cdot B\cdot C\cdot\prod_{i\in I}h_{i+1}^{m_i}\right)^{1/(e+x)}$.
   - Send $(\mathsf{sid},\mathsf{jid},(\mathbf{a}_I,A,e,s))$ to $\mathcal{H}$.

6. On receiving message $(\mathsf{sid},\mathsf{jid},(\mathbf{a}_I,A,e,s))$ from issuer I, the holder $\mathcal{H}$:
   - Retrieves record $\langle\mathsf{sid},\mathsf{jid},\mathbf{a}_H,s',n_{\mathcal{C}},n_I,C,H,\pi'\rangle$
   - Set $r=\mathsf{PRF}_K(n_{\mathcal{C}})$.
   - Sets $\sigma=(A,e,s+s'+r)$.
   - Sets $\mathbf{m}$ to be the message vector contained in $\mathbf{a}_H,\mathbf{a}_I$.
   - Continue below only if the received credential verifies, i.e., if $e(A,wg_2^e)=e(g_1,h_0^{s+s'+r}\prod_{i=1}^{\ell}h_i^{\mathbf{m}_i},g_2)$.
   - Stores $(\mathcal{C},Q,K,\mathbf{m},\sigma)$.
   - Outputs $(\mathsf{JOINED},\mathsf{sid},\mathsf{jid})$.

B.4 Presentation

The presentation phase involves a card $\mathcal{C}$, a holder $\mathcal{H}$ and verifier $\mathcal{V}$. The verifier specifies attributes to be disclosed $\mathbf{a}_V$ as $\{(i,m_i):i\in V\}$. The protocol involves $\mathcal{H}$ showing possession of attributes (undisclosed) $\mathbf{a}_H=\{(i,m_i):i\in H\}$ for $H=L\setminus V$, such that the complete message vector $\mathbf{m}=(m_1,\ldots,m_\ell)$ was used to pair $\mathcal{H}$ and $\mathcal{C}$. We now describe the detailed protocol (See also Figure 6 for an overview of the protocol flow).

1. On input $(\mathsf{SETATTRS},\mathsf{sid},\mathsf{vid},\mathbf{a}_V)$, the verifier $\mathcal{V}$:
   - Samples $n_{\mathcal{V}}\leftarrow\{0,1\}^\lambda$ and creates record $\langle\mathsf{sid},\mathsf{vid},\mathbf{a}_V,n_{\mathcal{V}}\rangle$.
   - On receiving $(\mathsf{sid},\mathsf{vid},n_{\mathcal{H}})$ from holder $\mathcal{H}$, add $n_{\mathcal{H}}$ to the record and send $(\mathsf{sid},\mathsf{vid},n_{\mathcal{H}},n_{\mathcal{V}})$ to a card.

2. On input $(\mathsf{SETID},\mathsf{sid},\mathsf{vid})$, the card $\mathcal{C}$:
   - Retrieves record (uid,K) and aborts if it contains $(\perp,\perp)$.
   - On receiving message $(\mathsf{sid},\mathsf{vid},n_{\mathcal{H}},n_{\mathcal{V}})$ from $\mathcal{V}$, the card $\mathcal{C}$:
     – Samples $n_{\mathcal{C}}\leftarrow\{0,1\}^\lambda$. Sets $n=n_{\mathcal{C}}||n_{\mathcal{H}}$.
     – Computes $r=\mathsf{PRF}_K(n)$.
     – Computes $B=h_1^{\mathsf{uid}}h_0^r$.
     – Computes signature proof: $\pi\leftarrow\mathsf{SDL}\{(\alpha,\beta):h_1^\alpha h_0^\beta=B\}(n_{\mathcal{V}})$.
     – Anonymously sends $(\mathsf{sid},\mathsf{vid},(n_{\mathcal{C}},B,\pi))$ to $\mathcal{V}$.

3. On receiving $(\mathsf{sid},\mathsf{vid},(n_{\mathcal{C}},B,\pi))$ from some card, the verifier $\mathcal{V}$:
   - Retrieves record $\langle\mathsf{sid},\mathsf{vid},\mathbf{a}_V,n_{\mathcal{V}},n_{\mathcal{H}}\rangle$ and appends $n_{\mathcal{C}},B,\pi$ to it.
   - Sends $(\mathsf{sid},\mathsf{vid},n_{\mathcal{C}},n_{\mathcal{V}},\mathbf{a}_V,B)$ to a holder.

4. On input $(\mathsf{SETCRED},\mathsf{sid},\mathsf{vid})$, the holder $\mathcal{H}$:
   - Samples $n_{\mathcal{H}}\leftarrow\{0,1\}^\lambda$ and sends $(\mathsf{sid},\mathsf{vid},n_{\mathcal{H}})$ to $\mathcal{V}$.
   - Upon receiving $(\mathsf{sid},\mathsf{vid},n_{\mathcal{C}},n_{\mathcal{V}},\mathbf{a}_V,B)$ from $\mathcal{V}$, $\mathcal{H}$ parses $\mathbf{a}_V$ as $\{(i,m_i):i\in V\}$ and sets $n=n_{\mathcal{C}}||n_{\mathcal{H}}$.
   - Retrieves record $(Q,K,\mathbf{m},\sigma)$ such that $B=Q\cdot h_0^{\mathsf{PRF}_K(n)}$ and $\mathbf{m}[i]=m_i$ for all $i\in V$. If no such record is found send $(\mathsf{sid},\mathsf{vid},\perp,\perp,\perp,\perp)$ to $\mathcal{V}$. The $\perp$ values here indicate that the holder does not have required attributes with any paired card.
   - Define $m'_{i+1}=m[i]$ for $i=1,\ldots,\ell$. Parse σ as (A,e,s).
   - Computes $(A',\bar{A},d,\pi')\leftarrow\mathsf{Prove}_{\mathcal{H}}(K,Q,(m'_2,\ldots,m'_{\ell+1}),(A,e,s),\mathbf{a}_V,n,B,n_{\mathcal{V}})$.
   - Sends $(\mathsf{sid},\mathsf{vid},(A',\bar{A},d,\pi'))$ to $\mathcal{V}$.

5. On receiving $(\mathsf{sid},\mathsf{vid},A',\bar{A},d,\pi')$ from some holder, the verifier $\mathcal{V}$:
   - Retrieves record $\langle \mathsf{sid},\mathsf{vid},\mathbf{a}_V,n_{\mathcal{V}},n_{\mathcal{C}},B,\pi\rangle$
   - Sets $\mathbf{x} = (B,h_0,h_1,n_{\mathcal{V}})$, $\mathbf{x}' = (A',\bar{A},d,\mathbf{a}_V,\{h_{i+1}\}_{i\in L},n_{\mathcal{V}})$.
   - If $\pi' = \perp$, outputs $(\mathsf{VERIFIED},\mathsf{sid},\mathsf{vid},0)$.
   - If $\mathsf{SDL.Verify}(\mathbf{x}',\pi') = 0 \vee \mathsf{SDL.Verify}(\mathbf{x},\pi) = 0$ outputs $(\mathsf{VERIFIED},\mathsf{sid},\mathsf{vid},0)$.
   - Sets $f = 1$ if $e(\bar{A},g_2) = e(A',w)$ else sets $f = 0$.
   - Outputs $(\mathsf{VERIFIED},\mathsf{sid},\mathsf{vid},f)$.

C Full proofs

C.1 Proof of Theorem 4.1

Proof. Since we are analysing security in the random-oracle model, we assume that parties have access to a random oracle H. We will skip completeness, as it follows from direct calculation, and the completeness of signature proofs of knowledge. For soundness, we consider the case when adversary $\mathcal{A}$ corrupts $\mathcal{C}$ and $\mathcal{H}$ and describe the extractor $\mathcal{E}^{\mathcal{A}}$. Since $\mathcal{V}$ outputs 1, we have $\mathsf{SDL.Verify}(\mathbf{x},\pi) = 1$ and $\mathsf{SDL.Verify}(\mathbf{x}',\pi') = 1$ for $\mathbf{x} = (B,h_0,h_1,n_{\mathcal{V}})$ and $\mathbf{x}' = (A',\bar{A},d,\{m_i\}_{i\in V},\{h_i\}_{i=0}^{\ell},n_{\mathcal{V}})$ respectively. From the argument-of-knowledge property of signature proofs (Lemma 3.1), with overwhelming probability we extract $(\alpha,\beta) \leftarrow \mathsf{SDL}.\mathcal{E}^{\mathcal{A}}(\mathbf{x},\pi)$ and $(e,s',r_2,r_3,\{m_i\}_{i\in H}) \leftarrow \mathsf{SDL}.\mathcal{E}^{\mathcal{A}}(\mathbf{x}',\pi')$ satisfying the equations:

$$h_1^{\alpha}h_0^{\beta} = B, \quad A'^{-e}h_0^{r_2} = \bar{A}/d,$$
$$d'^{-r_3}h_0^{s'}\prod_{i\in H}h_i^{m_i} = g_1^{-1}B^{-1}\prod_{i\in V}h_i^{-m_i}$$

Assuming the extractor outputs a non-trivial discrete log relation amongst the generators in pp with negligible probability, above equations imply $r_3 \neq 0$. From the success of the pairing check $e(\bar{A},g_2) = e(A',w)$ we conclude that $\bar{A} = (A')^x$. From above equations we get:

$$A'^{e+x} = dh_0^{r_2} = \big(g_1h_0^{s'}h_1^{\alpha}h_0^{\beta}\prod_{i=2}^{\ell}h_i^{m_i}\big)^{1/r_3}h_0^{r_2}$$
$$\Rightarrow (A'^{r_3})^{e+x} = g_1h_0^{s'+\beta+r_2r_3}h_1^{\alpha}\prod_{i=2}^{\ell}h_i^{m_i}.$$

which implies that $(A'^{r_3},e,s'+\beta+r_2r_3)$ is a valid signature on $(\alpha,m_2,\ldots,m_\ell)$. The extractor $\mathcal{E}^{\mathcal{A}}$ outputs the extracted message-signature pair.

To prove zero knowledge, we describe the simulator $\mathcal{S}$ which proceeds as follows: $\mathcal{S}$ simulates a random oracle H' where it responds to query x by uniformly choosing $\mathsf{H}'(x)$ from $\{0,1\}^{\lambda}$ if x has not been queried before, otherwise it responds with previously chosen value of $\mathsf{H}'(x)$. The simulator $\mathcal{S}$ simulates $n_{\mathcal{H}} \leftarrow \{0,1\}^{\lambda}$ as the first message from the holder to $\mathcal{V}$. When $\mathcal{V}$ sends the message $n_{\mathcal{H}},n_{\mathcal{V}}$ to $\mathcal{C}$, $\mathcal{S}$ simulates $(n_{\mathcal{C}},B,\pi)$ towards $\mathcal{V}$ by picking $n_{\mathcal{C}} \leftarrow \{0,1\}^{\lambda}$, $B \leftarrow \mathbb{G}_1$ and $\pi \leftarrow \mathsf{SDL.Sim}(\mathbf{x})$ for $\mathbf{x} = (B,h_0,h_1,n_{\mathcal{V}})$. Later when $\mathcal{V}$ sends the message $(n_{\mathcal{C}},n_{\mathcal{V}},\mathbf{a}_V,B)$ to $\mathcal{H}$, $\mathcal{S}$ simulates the message $(A',\bar{A},d,\pi')$ towards $\mathcal{V}$ by choosing $\rho \leftarrow \mathbb{Z}_p$, $A' \leftarrow \bar{g}_1^{\rho}, \bar{A} \leftarrow \bar{g}_2^{\rho}$, $d \leftarrow \mathbb{G}_1$ and $\pi' \leftarrow \mathsf{SDL.Sim}'(\mathbf{x}')$ where $\mathbf{x}' = (A',\bar{A},d,\{m_i\}_{i\in V},\{h_i\}_{i=0}^{\ell},n_{\mathcal{V}})$. Here $\mathsf{SDL.Sim}$ and $\mathsf{SDL.Sim}'$ denote the simulators for relations $\mathcal{R}$ and $\mathcal{R}'$ proved by $\mathcal{C}$ and $\mathcal{H}$ respectively. We remark that both simulators $\mathsf{SDL.Sim}$ and $\mathsf{SDL.Sim}'$ output simulated proofs using H' as their (shared) random-oracle, i.e $\mathsf{SDL.Sim.H} = \mathsf{SDL.Sim}'.\mathsf{H} = \mathsf{H}'$ as in Definition 3.1.

We argue indistinguishability of simulator $\mathcal{S}$'s output from verifier's veiw in the protocol via following games

Game $\mathbf{G}_0$: (**Real Protocol**): This game runs the real protocol described in Figure 3, with honest $\mathcal{C}$ and $\mathcal{H}$ with correct inputs.

Game $\mathbf{G}_1$: (**$\mathcal{C}$ and $\mathcal{H}$ query** PRF): We re-structure the above game as follows: We no longer provide the PRF key K as input to $\mathcal{C}$ and $\mathcal{H}$. Instead we introduce a challenger $\mathcal{B}$ which runs the PRF functionality. The challenger $\mathcal{B}$ samples $K \leftarrow \mathcal{K}$ when instantiated, and returns $\mathsf{PRF}_K(n)$ when queried with input n. Accordingly, we modify $\mathcal{C}$ and $\mathcal{H}$ to obtain $r = \mathsf{PRF}_K(n)$ by querying $\mathcal{B}$. As this change is only sytanctical, it is indistinguishable to $\mathcal{V}$.

Game $\mathbf{G}_2$: (**$\mathcal{C}$ and $\mathcal{H}$ query random function**): This game is identical to previous game except that $\mathcal{B}$ when queried on an input n, returns uniformly sampled $r \leftarrow \mathbb{Z}_p^*$ if n has not been queried before. For subsequent queries to n, the same value is returned. Thus $\mathcal{B}$ simulates a random function. This change is indistinguishable to $\mathcal{V}$, as a distinguishing $\mathcal{V}$ can be used to construct a distinguisher $\mathcal{D}$ between outputs of PRF and random function as follows: Let $\mathcal{B}$ be a challenger in the PRF game, i.e., $\mathcal{B}$ samples $b \leftarrow \{0,1\}$ and emulates the challenger in $\mathbf{G}_1$ when $b = 0$, and emulates the challenger in $\mathbf{G}_2$ when $b = 1$. The distinguisher $\mathcal{D}$ runs the re-structured game querying $\mathcal{B}$ on input n to obtain r_1. When $b = 0$, the game is identical to $\mathbf{G}_1$ while for $b = 1$ it is identical to $\mathbf{G}_2$. The distinguisher invokes $\mathcal{V}$ to distinguish between the two.

Game $\mathbf{G}_3$: (**$\mathcal{S}$ runs random-oracle**): In this game, we replace random oracle H by the random-oracle H' which is simulated by $\mathcal{S}$. This game is clearly indistinguishable from the previous game.

Game $\mathbf{G}_4$: (**$\mathcal{C}$ simulates proofs**): In this game, instead of generating proof π in $\mathsf{Prove}_{\mathcal{C}}$ using the algorithm $\mathsf{SDL.Prove}$, $\mathcal{C}$ generates it using zero-knowledge simulator $\mathsf{SDL.Sim}$, i.e. $\pi \leftarrow \mathsf{SDL.Sim}(\mathbf{x})$, where $\mathbf{x}$ is the statement for the proof as defined earlier. Here, $\mathsf{SDL.Sim}$ uses the random oracle H' simulated by $\mathcal{S}$ as introduced in the previous game. We prove that $\mathbf{G}_4$ is indistinguishable from $\mathbf{G}_3$. We show that a distinguisher $\mathcal{D}$ between $\mathbf{G}_4$ and $\mathbf{G}_3$ can be used to construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ which has a non-negligible success probability in the game describing zero-knowledge property in Definition 3.1.

We construct a challenger $\mathcal{C}$ as follows: The challenger runs a random oracle H'. Initially it waits to receive $(\mathbf{x}, \mathbf{w})$ from $\mathcal{B}$. Subsequently it samples $b \leftarrow \{0,1\}$ uniformly and waits for further queries from $\mathcal{B}$. On receiving a random oracle query x from $\mathcal{B}$, the challenger $\mathcal{C}$ returns $\mathsf{H}'(x)$. On receiving a query $\mathsf{prove}(\mathbf{x}, \mathbf{w})$ from $\mathcal{B}$, $\mathcal{C}$ computes $\pi \leftarrow \mathsf{SDL.Prove}^{\mathsf{H}'}(\mathbf{x}, \mathbf{w})$ if $b = 0$ and $\pi \leftarrow \mathsf{SDL.Sim}(\mathbf{x})$ if $b = 1$, where $\mathsf{SDL.Sim.H}() = \mathsf{H}'()$.

We now describe the adversary $\mathcal{B}$. The adversary $\mathcal{B}$ runs the game $\mathbf{G}_3$ with following re-structuring: It forwards all queries to the random oracle to $\mathcal{C}$, which answers them using its own simulated random-oracle as described above. Similarly, proof π is obtained by querying the challenger $\mathcal{C}$ using query $\mathsf{prove}(\mathbf{x}, \mathbf{w})$ instead of computing $\pi \leftarrow \mathsf{SDL.Prove}(\mathbf{x}, \mathbf{w})$. Here $\mathbf{w}$ denote the witness used to generate proof for the statement $\mathbf{x}$. When $b = 0$, the re-structured game is identical to $\mathbf{G}_3$, while for $b = 1$ it is identical $\mathbf{G}_4$. Thus $\mathcal{B}$ can use the distinguisher $\mathcal{D}$ to succeed against $\mathcal{C}$ with non-negligible probability, contradicting the zero-knowledge property of signature proofs in Definition 3.1.

Game $\mathbf{G}_5$: (**$\mathcal{H}$ simulates proofs**): This game is same as previous game, except we also generate the proof π' by the holder as $\pi' \leftarrow \mathsf{SDL.Sim}'(\mathbf{x}')$ where $\mathsf{SDL.Sim}'$ also uses the random oracle H' run by $\mathcal{S}$. Additionally, we abort if $\mathsf{SDL.Sim}'$ attempts to program a query x' in H' which has been previously queried or programmed. This happens with negligible probability because x' is distributed uniformly over a set of size $\Omega(2^{\lambda})$. This game can be shown to be indistinguishable from the previous game using zero-knowledge property of signature proofs as earlier. Moreover, when the game does not abort, its output is identical to the output of $\mathcal{S}$, as the proofs π and π' are distributed indentically to the case when $\mathsf{SDL.Sim}$ and $\mathsf{SDL.Sim}'$ use a shared random oracle H' as used by the simulator.

Game $\mathbf{G}_6$: (**Simulate statements for card and holder proofs**): Finally, we simulate the statements for card and holder proofs by choosing $n_{\mathcal{H}}, n_{\mathcal{C}}, B, A', \bar{A}, d$ as $n_{\mathcal{H}} \leftarrow \{0,1\}^{\lambda}$, $n_{\mathcal{C}} \leftarrow \{0,1\}^{\lambda}$, $B \leftarrow \mathbb{G}_1$, $\rho \leftarrow \mathbb{Z}_p$, $A' \leftarrow \bar{g}_1^{\rho}$, $\bar{A} = \bar{g}_2^{\rho}$ $(= A'^{x})$, $d \leftarrow \mathbb{G}_1$, which is identical to their distribution output by the simulator $\mathcal{S}$. The distribution corresponds to uniformly and independently sampling randomness r inside algorithms $\mathsf{Prove}_{\mathcal{C}}$ and Prove_{H}, instead of querying the random function as in $\mathbf{G}_5$. Since all the queries to the random function are distinct with overwhelming probability, this change is indistinguishable. This completes the proof of zero-knowledge.

□

C.2 Proof of Theorems 7.1 and 7.2

Proof Sketch for both above Theorems. We show the construction of simulator $\mathcal{S}$ such that no PPT environment $\mathcal{E}$ can distinguish between the interaction of real world adversary $\mathcal{A}$ with the real-world protocol π_{cbAA} and that of the ideal-world adversary $\mathcal{S}$ interacting with ideal functionality $\mathcal{F}_{\mathsf{cbAC}}$. We can further assume (without loss of generality) that $\mathcal{A}$ is a dummy adversary that relays messages between $\mathcal{E}$ and real protocol/simulator.

The proof strategy is to start with the game where environment $\mathcal{E}$ interacts with the real protocol, and via a sequence of games which are indistinguishable, arrive at a game where $\mathcal{E}$ interacts with the functionality $\mathcal{F}_{\mathsf{cbAC}}$ and a simulator $\mathcal{S}$. The challenge for $\mathcal{S}$ is hence to produce a view indistinguishable from the real protocol run, but without knowing the information that is kept secret by $\mathcal{F}_{\mathsf{cbAC}}$, which is:

- Attribute sets $\mathbf{a}_H, \mathbf{a}_I$ in the join phase.
- Party identifiers $\mathcal{C}$ and $\mathcal{H}$ in the presentation phase.

We describe the sequence of games below and start with a sketch.

- **The real protocol**: In this game, we have an empty functionality $\mathcal{F}$ and a simulator $\mathcal{S}$ which runs all the honest parties according to the real-world protocol. To enable this, the functionality $\mathcal{F}$ simply passes all the inputs from the environment to $\mathcal{S}$. The outputs of honest parties in $\mathcal{S}$ are sent back to $\mathcal{E}$, again via the functionality $\mathcal{F}$. Clearly, this game is simply a restructuring of the real-world protocol.

- **Add setup to $\mathcal{F}$**: In this game, we add setup interface of $\mathcal{F}_{\mathsf{cbAC}}$ to $\mathcal{F}$, but otherwise continue to forward inputs from $\mathcal{E}$ to $\mathcal{S}$. After the simulated issuer completes the setup, $\mathcal{S}$ indicates the same the to $\mathcal{F}$, which also runs the setup interface of $\mathcal{F}_{\mathsf{cbAC}}$. These changes being internal to $\mathcal{F}$ and $\mathcal{S}$ are indistinguishable to $\mathcal{E}$.

- **Add join phase to $\mathcal{F}$**: We beef up the functionality $\mathcal{F}$ further by incorporating the code of join interface of $\mathcal{F}_{\mathsf{cbAC}}$, with a slight difference of still forwarding inputs from the environment to $\mathcal{S}$. This allows simulator $\mathcal{S}$ to continue simulating the real world protocol as in previous games, but in parallel we start populating attributes in $\mathcal{F}$ also. Specifically, whenever the issuer in simulated real world protocol with $\mathcal{C}$ and $\mathcal{H}$ outputs a credential (A, e, s) for attribute vector $\mathbf{m}$, the simulator adds the corresponding pair and attributes $(\mathcal{C}, \mathcal{H}, \mathbf{m})$ in $\mathcal{F}$ using $\mathsf{JOINCOMPLETE}$ interface. In case of corrupt $\mathcal{H}$, the part of vector $\mathbf{m}$ not specified by the issuer is "extracted" from the commitment and proof (C, π') sent by the holder. In nutshell, the game ensures that corresponding to each issued credential (A, e, s) in the real protocol, the functionality adds a record $(\mathcal{C}, \mathcal{H}, \mathbf{m})$. Finally, we notice that the messages (2) and (4) to the corrupt issuer can be simulated by uniformly choosing outputs of the PRF and using the simulators for zero-knowledge proofs to simulate proofs on the resulting (simulated) statements.

- **Add presentation phase to $\mathcal{F}$**: We now replicate presentation phase of $\mathcal{F}_{\mathsf{cbAC}}$ in $\mathcal{F}$. This means that $\mathcal{F}$ hides information which it gets from $\mathcal{E}$ from $\mathcal{S}$. In particular, it hides identities of honest $\mathcal{C}$ and $\mathcal{H}$ from $\mathcal{S}$ (as required by anonymity). Thus, $\mathcal{S}$ can no longer run the real-world protocol, but must deliver identical output in the VERIFIED interface while simulating messages to the adversary as in the real protocol. The most non-trivial simulation is that of messages to the verifier in the real protocol since they depend on the secrets maintained within the card (uid) and the holder (attributes $\mathbf{m}$, signature (A,e,s)). For simulating these messages we primarily lean on the simulation in Theorem 4.1. The consistency of VERIFIED output between the two games relies on the fact that for each credential issued against a holder and a card for certain attributes in the real protocol, there is a corresponding record within $\mathcal{F}$, which can be used to determine the output of VERIFIED by calling VERIFYCOMPLETE. In all honest case, the consistency of outputs is obvious: Honest $\mathcal{C}$ and $\mathcal{H}$ are granted credentials (A,e,s) in the real-world join protocol for attributes $\mathbf{m}$, if and only if the corresponding record $(\mathcal{C},\mathcal{H},\mathbf{m})$ is also present in $\mathcal{F}$. Moreover, honest holder only generates proofs if has the credential for specified attributes and card (which it recognizes by checking $B = Q \cdot h_0^{\mathsf{PRF}_K(n)}$ against its list of (Q,K) pairs), otherwise it outputs a dummy proof $\perp$ which is invalid by convention. By completeness of the protocol in Theorem 4.1, the real protocol to determine the VERIFIED output can be replaced by looking for the appropriate record in $\mathcal{F}$, which is accomplished by querying VERIFYCOMPLETE interface of $\mathcal{F}$. If the real world-protocol succeeds for corrupt parties, we can extract valid uid for card, attributes $\mathbf{m}$ and signature (A,e,s) which if not present in $\mathcal{F}$ would constitute a forgery of the issuer's signature. More details of this reduction appear in the detailed proof that follows this overview. This ensures consistency of outputs even in presence of corrupt parties.

- **Stop passing inputs to simulator in join**: To make $\mathcal{F}$ identical to $\mathcal{F}_{\mathsf{cbAC}}$, we need to "undo" the extravagance of passing inputs from $\mathcal{E}$ to $\mathcal{S}$ in the join interface. Since the presentation phase now no longer depends on the protocol execution in the join phase, and only on the records $(\mathcal{C},\mathcal{H},\mathbf{m})$ present in $\mathcal{F}$, we no longer need to run the real-world protocol in join phase, as long as we still add records in $\mathcal{F}$ as in the previous games. The JOINCOMPLETE interface is sufficient to add the records, as long as we know that the real protocol issues a credential. For the all honest case this is always true. For corrupt holders, the simulator can make checks identical to those made by the issuer in the real protocol before issuing the credential. It can also provide "extracted" attributes to $\mathcal{F}$ on behalf of the corrupt holder as earlier. We also change the simulator to explicitly maintain the state of simulated parties, as it no longer needs to execute them. The simulation of messages to the adversary (holder) is trivial as simulator has access to the required inputs $\mathbf{a}_I$ (leaked to $\mathcal{S}$ in this case) and state of honest parties (e.g uid for cards, signing key x for issuer) to generate the messages.

  The above completes the bird's eye view of the full proof that follows.

We give an overview of the most important cases to simulate in Figure 7 (corrupt card in presentation phase), Figure 8 (corrupt holder), and Figure 9 (corrupt verifier). The full indistinguishability argument follows.

Game G_0: The real protocol execution. In this game, we consider a dummy functionality $\mathcal{F}$ and a simulator $\mathcal{S}$. The simulator $\mathcal{S}$ runs all the honest parties, according to the real-world protocol described in Section 7. For discreteness, we use "$\mathcal{P}$" to denote the simulated copy of $\mathcal{P}$ run by the simulator. As $\mathcal{S}$ runs all the honest parties, it also receives all the messages sent by the adversary $\mathcal{A}$ on behalf of the corrupt parties to honest parties. The simulator $\mathcal{S}$ also relays the outputs of honest parties back to $\mathcal{F}$. The functionality $\mathcal{F}$ in this game acts as a relay between the environment $\mathcal{E}$ and the real world protocol run by $\mathcal{S}$. Specifically, $\mathcal{F}$ accepts inputs from $\mathcal{E}$ for the honest party $\mathcal{P}$ according to the interface specified by $\mathcal{F}_{\mathsf{cbAC}}$, and relays the same to $\mathcal{S}$ which uses them as inputs to the corresponding simulated party "$\mathcal{P}$". In the other direction, on receiving an output for a simulated party "$\mathcal{P}$" from $\mathcal{S}$, $\mathcal{F}$ outputs the same to $\mathcal{P}$. By construction, this game is identical to $\mathcal{E}$ interacting with the real world protocol.

Game G_1: $\mathcal{F}$ incorporates Setup interface. In this game $\mathcal{F}$ additionally incorporates the code of the SETUP interface of $\mathcal{F}_{\mathsf{cbAC}}$. We modify $\mathcal{F}$ from the previous game as: On receiving $(\mathsf{SETUPDONE},\mathsf{sid})$ from $\mathcal{S}$ as output of "$\mathcal{I}$", it executes the SETUP interface of $\mathcal{F}_{\mathsf{cbAC}}$ (which outputs $(\mathsf{SETUPDONE},\mathsf{sid})$ to $\mathcal{E}$). Clearly, this change is indistinguishable for $\mathcal{E}$, as additional messsages are internal to $\mathcal{F}$ and $\mathcal{S}$.

For corrupt issuer we change $\mathcal{S}$ as follows: When $\mathcal{A}$ delivers the same message (sid,vk) to all honest parties, the simulator calls $(\mathsf{SETUP},\mathsf{sid})$ on $\mathcal{F}$, and subsequently on receiving $(\mathsf{SETUP},\mathsf{sid})$ from $\mathcal{F}$ it calls $(\mathsf{SETUPDONE},\mathsf{sid})$ on $\mathcal{F}$. Otherwise $\mathcal{S}$ aborts.

Game G_2: $\mathcal{F}$ incorporates Join interface but forwards inputs. In this game $\mathcal{F}$ additionally incorporates the join interface of $\mathcal{F}_{\mathsf{cbAC}}$ with following differences:

- In step (J-H.2) output $(\mathsf{JOIN},\mathsf{jid},\mathcal{H},\mathbf{a}_H)$ to $\mathcal{S}$.
- In step (J-I.2) output $(\mathsf{JOINISSUE},\mathsf{jid},\mathbf{a}_I)$ to $\mathcal{S}$.

Accordingly we change $\mathcal{S}$ for join related queries as:

Join Phase $\mathcal{S}$ simulates an honest interaction between $\mathcal{C}, \mathcal{H}$ and I over secret channels, meaning that no message contents will be revealed to $\mathcal{A}$ but $\mathcal{A}$ will learn that a message was sent. For brevity we use the message numbering from Figures 5 and 6 in the below. $\mathcal{S}$ initializes records $\mathsf{JR}(x)$ with attributes $(\mathtt{C} \leftarrow \bot, \mathtt{H} \leftarrow \bot)$ in case they do not exist when updated.

On message $(\mathsf{JOINID}, \mathsf{vid}, \mathcal{C})$ from $\mathcal{F}_{\mathsf{cbAC}}$:

- If there is no tuple $(\mathcal{C}, *, *)$ stored yet, choose $\mathsf{uid} \leftarrow \mathbb{Z}_p$, $K \leftarrow \{0,1\}^\lambda$ and store $(\mathcal{C}, \mathsf{uid}, K)$.
- $\mathcal{S}$ updates $\mathsf{JR}(\mathsf{jid}).\mathtt{C} \leftarrow \mathcal{C}$.

On message $(\mathsf{JOIN}, \mathsf{jid}, \mathcal{H})$ from $\mathcal{F}_{\mathsf{cbAC}}$:

- $\mathcal{S}$ updates $\mathsf{JR}(\mathsf{jid}).\mathtt{H} \leftarrow \mathcal{H}$.

On message $(\mathsf{JOINISSUE}, \mathsf{jid})$ from $\mathcal{F}_{\mathsf{cbAC}}$:

- $\mathcal{S}$ signals the sending of consecutive messages (1)-(5) between issuer I, card $\mathsf{JR}(\mathsf{jid}).\mathtt{C}$ and holder $\mathsf{JR}(\mathsf{jid}).\mathtt{H}$ of the join phase to $\mathcal{A}$, where message (2) is only signaled if $\mathsf{JR}(\mathsf{jid}).\mathtt{C} \neq \bot$, and message (4) is only signaled if $\mathsf{JR}(\mathsf{jid}).\mathtt{H} \neq \bot$.
- If $\mathcal{A}$ delivered all messages unchanged, $\mathcal{S}$ sends $(\mathsf{JOINCOMPLETE}, \mathsf{jid})$ to $\mathcal{F}_{\mathsf{cbAC}}$.

Presentation Phase Let $\mathcal{C}^*$ denote the identifier of a corrupt card who is interacting with the verifier.

On message $(\mathsf{SETATTRS}, \mathsf{vid}, \mathcal{V}, \mathbf{a}_V)$ from $\mathcal{F}_{\mathsf{cbAC}}$:

- Sample $n_{\mathcal{H}}, n_{\mathcal{V}} \leftarrow \{0,1\}^\lambda$ and send $n_{\mathcal{H}}, n_{\mathcal{V}}$ to the corrupt $\mathcal{C}^*$ after message (1) has been signalled.

On message (B, π) from the corrupt $\mathcal{C}^*$:

- Signal the sending of message (4) on the channel to the holder.
- Let $\mathcal{C}^*$ denote any corrupt card. Send $(\mathsf{SETID}, \mathsf{vid})$ to $\mathcal{F}_{\mathsf{cbAC}}$ as input from $\mathcal{C}^*$.

On message $(\mathsf{SETCRED}, \mathsf{vid})$ from $\mathcal{F}_{\mathsf{cbAC}}$:

- Signal sending of message (1) to verifier, if it has not been signalled.
- If message (4) has already been signaled, signal now the sending of message (5) on the channel between verifier and holder.
- Send $(\mathsf{VERIFYCOMPLETE}, \mathsf{vid}, 0)$ to $\mathcal{F}_{\mathsf{cbAC}}$. // Presentations with corrupt cards always fail.

Figure 7: Simulator for an honest join phase, and a presentation phase with a corrupt card.

Join Phase

On message $(\mathsf{JOINISSUE}, \mathsf{jid}, \mathbf{a}_I)$ from $\mathcal{F}_{\mathsf{cbAC}}$:

- Signal the sending of messages (1) on the channel to the card

On message $(\mathsf{JOINID}, \mathsf{jid}, \mathcal{C})$ from $\mathcal{F}_{\mathsf{cbAC}}$:

- If message (1) has already been signaled, signal sending of message (2) by $\mathcal{C}$.
- If there is no tuple $(\mathcal{C}, *, *)$ stored yet, choose $\mathsf{uid} \leftarrow \mathbb{Z}_p$, $K \leftarrow \{0,1\}^\lambda$ and store $(\mathcal{C}, \mathsf{uid}, K)$.
- Choose $n_{\mathcal{C}}, n_I \leftarrow \{0,1\}^\lambda$, $r \leftarrow \mathsf{PRF}_K(n_{\mathcal{C}})$. Set $B \leftarrow h_1^{\mathsf{uid}} h_0^r$ and $\pi \leftarrow \mathsf{SDL}\{(\alpha, \beta) : h_1^\alpha h_0^\beta = B\}(n_I)$.
- Send $n_{\mathcal{C}}, n_I$ to the corrupt holder.

On message (C, π') from $\mathcal{A}$ to the issuer:

- [Thm. 7.2: $\{m_i\}_{i \in H} \leftarrow \mathcal{E}^{\mathcal{A}}((C, \{h_{i+1}\}_{i \in H}), \pi')$]
- Send $(\mathsf{JOIN}, \mathsf{vid},$ [Thm. 7.2: $\{(i, m_i) : i \in H\}$]$)$ to $\mathcal{F}_{\mathsf{cbAC}}$.
- Run the issuer's code to produce (A, e, s) and send $(\mathbf{a}_I, A, e, s)$ to the corrupt holder.
- Send $(\mathsf{JOINCOMPLETE}, \mathsf{vid})$ to $\mathcal{F}_{\mathsf{cbAC}}$.

Presentation Phase

On message $(\mathsf{SETATTRS}, \mathsf{vid}, \mathcal{V}, \mathbf{a}_V)$ from $\mathcal{F}_{\mathsf{cbAC}}$:

- $\mathcal{S}$ signals the sending of message (2) to the card after receiving $(\mathsf{SETCRED}, \mathsf{vid})$ from $\mathcal{F}_{\mathsf{cbAC}}$.

On message $(\mathsf{SETID}, \mathsf{vid}, [\mathcal{C}'])$ from $\mathcal{F}_{\mathsf{cbAC}}$:

- Signal the sending of message (3) from the card to the verifier, but only after message (2) was signaled.
- If the message does *not* have a card identifier, choose a fresh pair $\mathsf{uid} \leftarrow \mathbb{Z}_p$, $K \leftarrow \{0,1\}^\lambda$
- If the message contains a card identifier $\mathcal{C}'$, retrieve $(\mathcal{C}', \mathsf{uid}, K)$.
- Run the card and the verifier protocol with these values on input $\mathbf{a}_V$, to obtain $n_{\mathcal{C}}, n_{\mathcal{V}}, B$. Send $(n_{\mathcal{C}}, n_{\mathcal{V}}, B, \mathbf{a}_V)$ to $\mathcal{A}$.

On message $n_{\mathcal{H}}$ from $\mathcal{A}$ to the verifier:

- Send input $(\mathsf{SETCRED}, \mathsf{vid})$ to $\mathcal{F}_{\mathsf{cbAC}}$ on behalf of any corrupt holder.

On message $(A', \hat{A}, d', \pi')$ from $\mathcal{A}$ to the issuer:

- Run the verifier code on the message and let b denote its output.
- Send $(\mathsf{VERIFYCOMPLETE}, \mathsf{vid}, b)$ to $\mathcal{F}_{\mathsf{cbAC}}$.

Figure 8: Simulator for a *corrupt holder*.

Presentation Phase

On message $(\mathsf{SETCRED}, \mathsf{vid})$ from $\mathcal{F}_{\mathsf{cbAC}}$:

- Choose $n_{\mathcal{H}} \leftarrow \{0,1\}^{\lambda}$ and send $n_{\mathcal{H}}$ to $\mathcal{A}$.

On message $(\mathsf{SETID}, \mathsf{vid}, [\mathcal{C}'])$ from $\mathcal{F}_{\mathsf{cbAC}}$:

On message $(\mathsf{SETID}, \mathsf{vid})$ from $\mathcal{F}_{\mathsf{cbAC}}$ and $n'_{\mathcal{H}}, n_{\mathcal{V}}$ from $\mathcal{A}$:

- Invoke the ZK $\mathcal{S}$ of Thm. 4.1 on $n'_{\mathcal{H}}, n_{\mathcal{V}}$ to produce $n_{\mathcal{C}}, B, \pi$. Send $(n_{\mathcal{C}}, B, \pi)$ to $\mathcal{A}$.

On messages $(n'_{\mathcal{C}}, n'_{\mathcal{V}}, B', \mathbf{a}_V)$ from $\mathcal{A}$ and $(\mathsf{SETCRED}, \mathsf{vid})$ from $\mathcal{F}_{\mathsf{cbAC}}$:

- If $n_{\mathcal{C}}, n_{\mathcal{V}}, B \neq n'_{\mathcal{C}}, n'_{\mathcal{V}}, B'$ then send $(\perp,\perp,\perp,\perp)$ to $\mathcal{A}$.
- Send $(\mathsf{SETATTRS}, \mathsf{vid}, \mathbf{a}_V)$ to $\mathcal{F}_{\mathsf{cbAC}}$ on behalf of the corrupt $\mathcal{V}$, send $(\mathsf{VERIFYCOMPLETE}, \mathsf{vid}, 1)$ to $\mathcal{F}_{\mathsf{cbAC}}$ and wait to receive back $(\mathsf{VERIFIED}, \mathsf{vid}, f)$.
- If $f = 0$, then send message $(\perp,\perp,\perp,\perp)$ to $\mathcal{A}$.
- If $f = 1$, invoke again the ZK $\mathcal{S}$ (Thm. 4.1) on the verifier's messages to create message $(A', \bar{A}, d, \pi')$ and send it to $\mathcal{A}$.

Figure 9: Simulator for a *corrupt verifier* appearing only in the presentation phase.

Join Phase

On message $(\mathsf{JOIN}, \mathsf{jid}, \mathcal{H}, H')$ from $\mathcal{F}_{\mathsf{cbAC}}$:

- Set $\mathsf{JR}(\mathsf{jid}).\mathtt{setH} \leftarrow H'$, $\mathsf{JR}(\mathsf{jid}).\mathtt{H} \leftarrow \mathcal{H}$.

On message $(\mathsf{JOINID}, \mathsf{jid}, \mathcal{C})$ from $\mathcal{F}_{\mathsf{cbAC}}$ and n_I from $\mathcal{A}$:

- If no record $(\mathcal{C}, *, *)$ exists, sample $\mathsf{uid} \leftarrow \mathbb{Z}_p$, $K \leftarrow \mathcal{K}$ and store $(\mathcal{C}, \mathsf{uid}, K)$; otherwise fetch uid, K from the record $(\mathcal{C}, \mathsf{uid}, K)$.
- Simulate message $(n_{\mathcal{C}}, B, \pi)$ to the adversary by choosing $n_{\mathcal{C}} \leftarrow \{0,1\}^{\lambda}$, $r \leftarrow \mathsf{PRF}_K(n_{\mathcal{C}})$, $B \leftarrow h_1^{\mathsf{uid}} h_0^r$ and $\pi \leftarrow \mathsf{SDL.Sim}(B, h_0, h_1, n_I)$.

On message $(n_{\mathcal{C}}, n_I)$ from $\mathcal{A}$:

- Simulate message (C, H, π') by choosing $C \leftarrow \mathbb{G}_1$, $H = \mathsf{JR}(\mathsf{jid}).\mathtt{setH}$ and $\pi' \leftarrow \mathsf{SDL.Sim}(C, \{h_{i+1}\}_{i\in H}, n_I)$ but only after receiving $(\mathsf{JOIN}, \mathsf{jid}, *)$ from $\mathcal{F}_{\mathsf{cbAC}}$.

On message $(\mathbf{a}_I, A, e, s)$ from $\mathcal{A}$:

- Abort if $e(A, wg_2^e) \neq e(g_1 h_0^s \cdot B \cdot C \cdot \prod_{i\in I} h_{i+1}^{m_i}, g_2)$
- Send $(\mathsf{JOINISSUE}, \mathsf{jid}, \mathbf{a}_I)$ to $\mathcal{F}_{\mathsf{cbAC}}$ and on immediate output $(\mathsf{JOINISSUE}, \mathsf{jid})$ send $(\mathsf{JOINCOMPLETE}, \mathsf{jid})$ to $\mathcal{F}_{\mathsf{cbAC}}$.

Presentation Phase: Same as that for corrupt verifier in Fig 9.

Figure 10: Simulation for *corrupt issuer* appearing in the join phase, and as a verifier in the presentation phase

- Honest $\mathcal{H}$: We first consider the case when issuer is honest. On receiving messages $(\mathsf{JOIN}, \mathsf{jid}, \mathcal{H}, \mathbf{a}_H)$, $(\mathsf{JOINID}, \mathsf{jid}, \mathcal{C})$ and $(\mathsf{JOINISSUE}, \mathsf{jid}, \mathbf{a}_I)$ from $\mathcal{F}$, $\mathcal{S}$ runs the corresponding simulated parties "$\mathcal{H}$", "$\mathcal{C}$" and "$\mathcal{I}$" with the respective inputs. When simulated holder "$\mathcal{H}$" outputs $(\mathsf{JOINED}, \mathsf{jid})$, $\mathcal{S}$ calls $(\mathsf{JOINCOMPLETE}, \mathsf{jid})$ interface of $\mathcal{F}$. When the issuer is corrupt, the simulator $\mathcal{S}$ calls $(\mathsf{JOINISSUE}, \mathsf{jid}, \mathbf{a}_I)$ interface of $\mathcal{F}$ after receiving messages (jid, n_I), $(\mathsf{jid}, n'_{\mathcal{C}}, n'_I)$ and $(\mathsf{jid}, \mathbf{a}_I, A, e, s)$ from the adversary. The simulator then calls $(\mathsf{JOINCOMPLETE}, \mathsf{jid})$ when simulated holder outputs $(\mathsf{JOINED}, \mathsf{jid})$.
- Corrupt $\mathcal{H}$: When issuer is honest, we proceed as: When adversary $\mathcal{A}$ delivers the message $(\mathsf{jid}, C, H, \pi')$ to "$\mathcal{I}$" on behalf of a corrupt $\mathcal{H}$, and $\mathsf{SDL.Verify}((C, \{h_{i+1}\}_{i\in H}, n_2), \pi) = 1$, $\mathcal{S}$ extracts holder attributes $\{m_i\}_{i\in H} \leftarrow \mathsf{SDL}.\mathcal{E}^{\mathcal{A}}((C, \{h_{i+1}\}_{i\in H}), n_I, \pi')$. Here n_I is the nonce sent by issuer to the corrupt holder. As we remark later while proving indistinguishability of this game to the previous game, the proof π' here needs to be *online-extractable* (see Appendix A and references therein for details). If the proof π' does not verify for the statement $(C, \{h_{i+1}\}_{i\in H}, n_I)$, $\mathcal{S}$ chooses $m_i : i \in H$ arbitrarily. In this case they will never be delivered to $\mathcal{F}$ via $\mathsf{JOINCOMPLETE}$. Finally $\mathcal{S}$ sets $\mathbf{a}_H = \{(i, m_i) : i \in H\}$ and calls $(\mathsf{JOIN}, \mathsf{jid}, \mathbf{a}_H)$ interface of $\mathcal{F}$ on behalf of some corrupt holder $\mathcal{H}'$. Later, when "$\mathcal{I}$" produces final message $(\mathsf{jid}, \mathbf{a}_I, A, e, s)$ (which happens only if holder proof verifies) $\mathcal{S}$ calls the $(\mathsf{JOINCOMPLETE}, \mathsf{jid})$ interface of $\mathcal{F}$ to install the attributes in $\mathcal{F}$. When the issuer is also corrupt, $\mathcal{S}$ simply runs the real-wolrd protocol for the honest card $\mathcal{C}$.

We argue that the changes are indistinguishable to $\mathcal{E}$. Since $\mathcal{F}$ always delivers inputs from $\mathcal{E}$ to the corresponding simulated parties in $\mathcal{S}$, honest parties run with identical inputs between this and the previous game. As a consequence, all the messages to the adversary are distributed identically in the two games. It only remains to argue that both the games deliver the output $(\mathsf{JOINED}, \mathsf{jid})$ to $\mathcal{E}$ identically. Since $\mathcal{S}$ in current game calls $\mathsf{JOINCOMPLETE}$ to deliver the output, only if the real-world protocol outputs the same, output in $\mathbf{G}_2$ implies output in $\mathbf{G}_1$. In the other direction, the only additional check made in $\mathsf{JOINCOMPLETE}$ interface of $\mathcal{F}$ is the check $H = L \setminus I$ (J2). This must hold for real-world protocol to output as well and thus does not prevent $\mathbf{G}_2$ from delivering the output. Hence we conclude that $\mathbf{G}_2$ outputs whenever $\mathbf{G}_1$ outputs. This completes the proof of indistinguishability between the two games.

Game $\mathbf{G}_3$: $\mathcal{F}$ replicates the presentation phase of $\mathcal{F}_{\mathsf{cbAC}}$. In this game, we modify the presentation phase of $\mathcal{F}$ to be identical to that of $\mathcal{F}_{\mathsf{cbAC}}$. We note that $\mathcal{F}$ is still not identical to $\mathcal{F}_{\mathsf{cbAC}}$ as the join interface in $\mathcal{F}$ still forwards $\mathcal{E}$'s inputs to honest parties to $\mathcal{S}$. We modify $\mathcal{S}$ from the previous game as follows:

Join Phase: Same as in Game $\mathbf{G}_2$.

Presentation Phase: We change the presentation phase to deliver the output to honest verifier using VERIFYCOMPLETE instead of forwarding the output of simulated verifier "$\mathcal{V}$" like in previous game.

- All honest: This case does not require $\mathcal{S}$ to simulate any protocol messages. It just delivers the output using VERIFYCOMPLETE after all required interfaces of $\mathcal{F}$ have been called.
  - Call $(\mathsf{VERIFYCOMPLETE}, \mathsf{vid}, 1)$ interface of $\mathcal{F}$ to deliver the output to $\mathcal{E}$ after receiving $(\mathsf{SETATTRS}, \mathsf{vid}, *)$, $(\mathsf{SETID}, \mathsf{vid})$ and $(\mathsf{SETCRED}, *)$ from $\mathcal{F}$.
- Corrupt $\mathcal{C}$: In this case, $\mathcal{S}$ simulates the second message in Figure 6 from issuer to the card, and calls the SETID interface of $\mathcal{F}$ on behalf of $\mathcal{C}$ using an arbitrary corrupt card $\mathcal{C}'$.
  - On input $(\mathsf{SETATTRS}, \mathsf{vid}, \mathcal{V}, \mathbf{a}_V)$ from $\mathcal{F}$: sample $n_{\mathcal{H}}, n_{\mathcal{V}} \leftarrow \{0,1\}^{\lambda}$ and send $(\mathsf{vid}, n_{\mathcal{H}}, n_{\mathcal{V}})$ to $\mathcal{A}$ after receiving $(\mathsf{SETCRED}, \mathsf{vid})$ from $\mathcal{F}$.
  - On message $(\mathsf{vid}, n_{\mathcal{C}}, B, \pi)$ from $\mathcal{A}$:
    - ∗ Call $(\mathsf{SETID}, \mathsf{vid})$ interface of $\mathcal{F}$ using a corrupt card $\mathcal{C}'$ and on immediate message $(\mathsf{SETID}, \mathsf{vid})$ from $\mathcal{F}$, call $(\mathsf{VERIFYCOMPLETE}, \mathsf{vid}, 0)$ interface to deliver the output to $\mathcal{E}$.
- Corrupt $\mathcal{H}$: In this case $\mathcal{S}$ needs to simulate the fourth message $(\mathsf{vid}, n_{\mathcal{C}}, n_{\mathcal{V}}, B, \mathbf{a}_V)$ in Figure 6 in addition to delivering the VERIFIED output. On receiving the message $(\mathsf{vid}, n_{\mathcal{H}})$ from $\mathcal{A}$, the simulator chooses $n_{\mathcal{C}}, n_{\mathcal{V}} \leftarrow \{0,1\}^{\lambda}$ uniformly at random, while distribution of B depends on whether the card $\mathcal{C}$ was previously joined to a corrupt holder or not. Moreover, the simulator can determine whether the presentation would succeed or fail: In case $\mathcal{C}$ is not paired to a corrupt holder, it will always fail. If the card was previously paired, $\mathcal{S}$ learns the card, and can check its credentials issued against the card, to determine if one of them satisfies the presentation predicate and that the corrupt holder is using such a credential. The latter is essentially accomplished by verifying the messages from the adversary like an honest verifier would. The simulator appropriately sets $b = 0, 1$ to indicate whether VERIFYCOMPLETE should deliver success or failure. We provide the details below:
  - On input $(\mathsf{SETATTRS}, \mathsf{vid}, \mathcal{V}, \mathbf{a}_V)$ from $\mathcal{F}$: create a record $(\mathsf{vid}, \mathcal{V}, \mathbf{a}_V)$ to pull up $\mathbf{a}_V$ later.
  - On input $(\mathsf{SETID}, \mathsf{vid})$ from $\mathcal{F}$: This is the case when card was not previously joined with any corrupt $\mathcal{H}'$.
    - ∗ Sample $n_{\mathcal{C}}, n_{\mathcal{V}} \leftarrow \{0,1\}^{\lambda}$, $B \leftarrow \mathbb{G}_1$ and simulate message $(\mathsf{vid}, n_{\mathcal{C}}, n_{\mathcal{V}}, B, \mathbf{a}_V)$ towards $\mathcal{A}$, but only after $(\mathsf{SETATTRS}, \mathsf{vid}, \mathcal{V}, \mathbf{a}_V)$ is received from $\mathcal{F}$ and $(\mathsf{vid}, n_{\mathcal{H}})$ is delivered by the adversary. Note that we pick B which depends on PRF output uniformly and independently at random, as with overwhleming probability, the adversary sees messages corresponding to PRF outputs on *distinct* queries.
  - On input $(\mathsf{SETID}, \mathsf{vid}, \mathcal{C})$ from $\mathcal{F}$: This is the case when card was previously joined with some corrupt $\mathcal{H}'$.
    - ∗ Fetch Q, K corresponding to simulated card "$\mathcal{C}$".
    - ∗ Sample $n_{\mathcal{C}}, n_{\mathcal{V}} \leftarrow \{0,1\}^{\lambda}$, set $n = n_{\mathcal{C}} || n_{\mathcal{H}}$, $B = Q \cdot h_0^{\mathsf{PRF}_K(n)}$ and simulate message $(\mathsf{vid}, n_{\mathcal{C}}, n_{\mathcal{V}}, \mathbf{a}_V, B)$ towards $\mathcal{A}$, but only after $(\mathsf{SETATTRS}, \mathsf{vid}, \mathcal{V}, \mathbf{a}_V)$ is received from $\mathcal{F}$ and $(\mathsf{vid}, n_{\mathcal{H}})$ is delivered by the adversary.
  - On message $(\mathsf{vid}, A', \bar{A}, d, \pi')$ from $\mathcal{A}$:
    - ∗ Parse $\mathbf{a}_V = \{(i, v_i) : i \in V\}$.
    - ∗ Set $\mathbf{x}' = (A', \bar{A}, d, \mathbf{a}_V, \{h_{i+1}\}_{i \in L \setminus V}, n_{\mathcal{V}})$.
    - ∗ If $\mathsf{SDL.Verify}(\mathbf{x}', \pi') = 1 \wedge e(\bar{A}, g_2) = e(A', w)$ set $b \leftarrow 1$, otherwise set $b \leftarrow 0$.
    - ∗ Call $(\mathsf{SETCRED}, \mathsf{vid})$ interface of $\mathcal{F}$ on behalf of some corrupt holder $\mathcal{H}'$ and then call $(\mathsf{VERIFYCOMPLETE}, \mathsf{vid}, b)$ interface to deliver the output.
- Corrupt $\mathcal{C}, \mathcal{H}$: On input $(\mathsf{SETATTRS}, \mathsf{vid}, \mathcal{V}, \mathbf{a}_V)$ from $\mathcal{F}$, simulate the message $(\mathsf{vid}, n_{\mathcal{H}}, n_{\mathcal{V}})$ with $n_1 \leftarrow \{0,1\}^{\lambda}$ towards the adversary $\mathcal{A}$ but only after it delivers $(\mathsf{vid}, n_{\mathcal{H}})$. Thereafter simulate $(\mathsf{vid}, n_{\mathcal{C}}, n_{\mathcal{V}}, B, \mathbf{a}_V)$ towards the adversary with after it delivers the message $(\mathsf{vid}, n_{\mathcal{C}}, B, \pi)$. On receiving $(\mathsf{vid}, A', \bar{A}, d, \pi')$ from $\mathcal{A}$, $\mathcal{S}$ calls $(\mathsf{SETCRED}, \mathsf{vid})$ and on subsequent immediate message $(\mathsf{SETCRED}, \mathsf{vid})$ from $\mathcal{F}$, calls $(\mathsf{VERIFYCOMPLETE}, \mathsf{vid}, 0)$ to deliver the output.
- Corrupt $\mathcal{V}$:We only need to simulate protocol messages to $\mathcal{V}$ in this case. Note that the verifier forwards the holder's part of PRF input $n_{\mathcal{H}}$ to $\mathcal{C}$ which is used by the card (with its own part $n_{\mathcal{C}}$ to determine the PRF input) to output a randomized commitment B to uid. Similarly the holder combines the forwarded $n_{\mathcal{C}}$ with its own part $n_{\mathcal{H}}$ to reconstruct the PRF query $n = n_{\mathcal{C}} || n_{\mathcal{H}}$ and uses it to de-randomize the commitment B to obtain the non-hiding commitment $Q = B.h_0^{-\mathsf{PRF}_K(n)}$. It can then use Q to identify one of the joined cards (if any). We let the simulator simulate $(\mathsf{vid}, \perp, \ldots, \perp)$ as the message (5) whenever $\mathcal{A}$ incorrectly forwards $n_{\mathcal{H}}, n_{\mathcal{C}}$ or B. If the $\mathcal{A}$ relays the messages honestly the simulator simulates message (5) as $(\mathsf{vid}, \perp, \ldots, \perp)$ if $\mathcal{F}$ returns $(\mathsf{vid}, \mathsf{VERIFIED}, 0)$ when queried using VERIFYCOMPLETE interface, and according to simulation in Theorem 4.1 when VERIFYCOMPLETE query returns $(\mathsf{vid}, \mathsf{VERIFIED}, 1)$ as this case meets the requirements for the zero knowledge property in the theorem which honest $\mathcal{C}$ and $\mathcal{H}$ participate with the correct inputs. We now provide details of the simulation in this case, and later argue the correctness.
  - On input $(\mathsf{SETID}, \mathsf{vid})$ from $\mathcal{F}$: Simulate message $(\mathsf{vid}, n_{\mathcal{C}}, B, \pi)$ towards $\mathcal{A}$ by choosing $n_{\mathcal{C}} \leftarrow \{0,1\}^{\lambda}$,

$B \leftarrow \mathbb{G}_1$ and $\pi \leftarrow \mathsf{SDL.Sim}(B, h_0, h_1, n_{\mathcal{V}})$ but only after $\mathcal{A}$ delivers $(\mathsf{vid}, n'_{\mathcal{H}}, n_{\mathcal{V}})$. Again, B is sampled independent of PRF inputs $n_{\mathcal{C}}, n_{\mathcal{H}}$ as these are distinct with overwhelming probability.

– On $\mathcal{A}$ delivering message $(\mathsf{vid}, n'_{\mathcal{C}}, n'_{\mathcal{V}}, \mathbf{a}_V, B')$: Call $(\mathsf{SETATTRS}, \mathsf{vid}, \mathbf{a}_V)$ of $\mathcal{F}$ using any corrupt party $\mathcal{V}'$.

– On input $(\mathsf{SETCRED}, \mathsf{vid})$ from $\mathcal{F}$: Simulate $(\mathsf{vid}, n_{\mathcal{H}})$ towards the adversary where $n_{\mathcal{H}} \leftarrow \{0,1\}^{\lambda}$. Only after $\mathcal{A}$ delivering messages (2) and (4) in Figure 6 which we denote by $(\mathsf{vid}, n'_{\mathcal{H}}, n_{\mathcal{V}})$ and $(\mathsf{vid}, n'_{\mathcal{C}}, n'_{\mathcal{V}}, \mathbf{a}_V, B')$ respectively simulate towards $\mathcal{A}$ as follows: If $(n'_{\mathcal{C}}, n'_{\mathcal{H}}, B') \neq (n_{\mathcal{C}}, n_{\mathcal{H}}, B)$ simulate the message $(\mathsf{vid}, \perp, \perp, \perp, \perp)$ towards $\mathcal{A}$. Otherwise, call $(\mathsf{VERIFYCOMPLETE}, \mathsf{vid}, 1)$ to obtain $(\mathsf{VERIFIED}, \mathsf{vid}, f)$. If $f = 0$ simulate the message $(\mathsf{vid}, \perp, \perp, \perp, \perp)$ towards $\mathcal{A}$ else simulate $(A', \bar{A}, d, \pi')$ according to simulator $\mathcal{S}$ in Theorem 4.1.

- Corrupt $\mathcal{V}, \mathcal{H}$: On input $(\mathsf{SETID}, \mathsf{vid}, \mathcal{C})$ from $\mathcal{F}$ (i.e, the card is not anonymous) simulate the message $(n_{\mathcal{C}}, B, \pi)$ towards $\mathcal{A}$ by setting $B = Q \cdot h_0^{\mathsf{PRF}_K(n)}$ and $\pi \leftarrow \mathsf{SDL.Sim}(B, h_0, h_1, n_{\mathcal{V}})$ for $n = n_{\mathcal{C}} || n_{\mathcal{H}}$ but only after $\mathcal{A}$ has delivered the message $(\mathsf{vid}, n_{\mathcal{H}}, n_{\mathcal{V}})$. The pair (Q, K) in the above simulation corresponds to honest (simulated) card "$\mathcal{C}$".

  On input $(\mathsf{SETID}, \mathsf{vid})$ from $\mathcal{F}$ (i.e, card is anonymous), simulate as in the previous case of corrupt $\mathcal{V}$. .

- Corrupt $\mathcal{V}, \mathcal{C}$: On input $(\mathsf{SETCRED}, \mathsf{vid})$ from $\mathcal{F}$, simulate the message $(\mathsf{vid}, \perp, \perp, \perp, \perp, 0)$ towards $\mathcal{A}$ only after $\mathcal{A}$ has delivered message $(\mathsf{vid}, n_{\mathcal{C}}, n_{\mathcal{V}}, \mathbf{a}_V, B)$. .

- Corrupt $I, \mathcal{H}$ (and optionally $\mathcal{C}$): In this case we cannot guarantee unforgeability which is reflected in the functionality $\mathcal{F}_{\mathsf{cbAC}}$ by allowing $\mathcal{S}$ to determine the output of the $\mathsf{VERIFYCOMPLETE}$ interface (V-6). We leverage the same: In this case, the simulator $\mathcal{S}$ sees all the messages that an honest verifier needs to output the bit f. This allows the simulator to determine the outcome of the real-world protocol and set b appropriately in the $\mathsf{VERIFYCOMPLETE}$ interface to $\mathcal{F}_{\mathsf{cbAC}}$ to make the simulation consistent. The messages to the adversary are simulated as in the case when I is honest.

We argue that $\mathbf{G}_3$ is indistinguishable from $\mathbf{G}_2$. We show the following: (i) This game produces output if and only if previous game produces output (ii) Outputs are identical when produced and (iii) The messages received by adversary are identically distributed in both games. We note that $\mathbf{G}_2$ does not output only if an uninitialized honest card is used in SETID interface (check P-C.1). In this case, game $\mathbf{G}_3$ also does not output as $\mathcal{S}$ never calls $\mathsf{VERIFYCOMPLETE}$ interface on $\mathcal{F}$ (since it never receives message $(\mathsf{SETID}, \mathsf{vid})$ from $\mathcal{F}$). The only additional case where game $\mathbf{G}_3$ aborts is when $\mathsf{VERIFYCOMPLETE}$ is delivered out of order (V.1) on $\mathcal{F}$ which is appropriately avoided by $\mathcal{S}$ calling $\mathsf{VERIFYCOMPLETE}$ only after other presentation phases have been called on $\mathcal{F}$.

Next we argue that the outputs are identical in the games $\mathbf{G}_2$ and $\mathbf{G}_3$. Let $\mathcal{C}$, $\mathcal{H}$ and $\mathcal{V}$ denote the parties involved with $\mathcal{V}$ specifying attributes $\mathbf{a}_V = \{(i, m_i) : i \in V\}$. Assume that $\mathbf{G}_3$ outputs $(\mathsf{VERIFIED}, \mathsf{vid}, 1)$. If $\mathcal{H}$ is honest, check V.4 implies that there exists $\mathbf{m}$ such that $(\mathcal{C}, \mathcal{H}, \mathbf{m}) \in \mathsf{creds}$ and $\mathbf{m}[i] = m_i$ for all $i \in V$. Thus, $\mathcal{C}$ and $\mathcal{H}$ were joined with attributes $\mathbf{m}$, and in particular $\mathcal{H}$ must have a record $(Q, K, \mathbf{m}, \sigma)$ where (Q, K) corresponds to the card $\mathcal{C}$ and $\sigma = (A, e, s)$ is a verifying signature on $\mathbf{m}$ (note that an honest holder only records verifying signatures). From the completeness property in Theorem 4.1, we conclude that $\mathbf{G}_2$, which runs the real-world protocol with honest $\mathcal{C}$ and $\mathcal{H}$ with valid inputs also outputs 1 in this case. If $\mathcal{H}$ is corrupt, the simulator calls $\mathsf{VERIFYCOMPLETE}$ with $b = 1$ only after the checks made by the verifier in real-world protocol are satisfied. Thus the game $\mathbf{G}_2$ also outputs 1 in this case. Now consider the case when $\mathbf{G}_2$ outputs $(\mathsf{VERIFIED}, \mathsf{vid}, 1)$. We show that game $\mathbf{G}_3$ also outputs $(\mathsf{VERIFIED}, \mathsf{vid}, 1)$. This is trivial when $\mathcal{H}$ is honest, as honest holder generates a verifying proof only when it has a record $Q, K, \mathbf{m}, \sigma$ where (Q, K) corresponds to the card $\mathcal{C}$, $\mathbf{m}$ is the attribute vector satisfying $\mathbf{m}[i] = m_i$ for $i \in V$ and $\sigma = (A, e, s)$ is a verifying signature certifying attributes $\mathbf{m}$. Equivalently, the record $(\mathcal{C}, \mathcal{H}, \mathbf{m}) \in \mathsf{creds}$ in $\mathcal{F}$, and hence $\mathbf{G}_3$ also outputs 1. We now prove that $\mathbf{G}_3$ outputs 1 whenever $\mathbf{G}_2$ outputs 1, except with negligible probability even when $\mathcal{H}$ is corrupt. This is clearly true when the issuer I is also corrupt, as in this case the simulator explicitly determines the output of $\mathbf{G}_3$ by setting it identical to the output obtained according to $\mathbf{G}_2$ (specifying the output as b in the $\mathsf{VERIFYCOMPLETE}$ interface, and leveraging the check V-6 to ensure it is the delivered output). Thus, we assume that issuer is honest, and that with probability ε $\mathbf{G}_3$ outputs 0 when $\mathbf{G}_2$ outputs 1. We will construct a forger $\mathcal{B}$ that succeeds against a challenger $\mathcal{D}$ in the unforgeability game for signature scheme with probability $\varepsilon - \varepsilon_0$, where ε_0 is negligible in λ. In this experiment, the forger $\mathcal{B}$ runs $\mathcal{F}$ and $\mathcal{S}$ from Game $\mathbf{G}_2$ with slight re-structuring as follows: Instead of generating the signature secret key x and public key w, the simulated issuer "I" obtains w from the challenger $\mathcal{D}$ which simulates a signature oracle with signing key x. We let $\mathcal{A}$ denote the adversary consisting of PPT environment $\mathcal{E}$ and the protocol adversary. The protocol adversary can be considered a "dummy" (or part of $\mathcal{E}$) when analysing in the UC framework, while we consider it seperate from $\mathcal{E}$ for standalone security. We also change the issuer to respond to signature request message (C, H, π') (message (4) in Figure 5) by $\mathcal{A}$ on behalf of a corrupt holder in the following way:

- When using $\mathcal{F}_{\mathsf{cbAC}}$ without blind attributes from holder: The forger $\mathcal{B}$ assembles message vector $\mathbf{m} = (m_1, \ldots, m_\ell)$ from $\mathbf{a}_I$.
- When using $\mathcal{F}_{\mathsf{cbAC}}$ with blind attributes from

holder: The forger $\mathcal{B}$ runs the extractor $(m_i)_{i\in H} \leftarrow \mathsf{SDL}.\mathcal{E}^{\mathcal{A}}((C,H,\{h_{i+1}\}_{i\in H},n_2),\pi')$ and obtains holder attributes $\mathbf{a}_H = \{(i,m_i) : i \in H\}$. It then assembles message vector $\mathbf{m} = (m_1,\ldots,m_\ell)$ from $\mathbf{a}_H$ and $\mathbf{a}_I$ (available to $\mathcal{B}$ from $\mathcal{A}$).

To ensure that this step runs in polynomial time, we need the proof π' to be online extractable (e.g, those described by Fischlin in [40]). While UC framework does not permit extraction by rewinding the adversary, even in the standalone setting, extraction by rewinding may require exponential time in the number of signing requests. We refer the interested reader to the discussion in ([58], Section 2.7) regarding extraction by rewinding from several adaptively generated statement-proof pairs in the random oracle model (our setting in this paper). To illustrate the issue broadly, assume that random oracle query for proof π'_i for commitment C_i is made before the query for proof π'_j for commitment C_j, and C_j depends on the random oracles's answer to the first query. Now, if the adversary uses C_j in the signing request before C_i, to extract the message from C_i, the extractor would need to rewind $\mathcal{A}$ to the state when it queried random oracle for C_i, and replay it with fresh random oracle simulation. This will however result in $\mathcal{A}$ making a new signing request for C'_j (the changed commitment C_j) thus making it re-do the earlier extraction for C_j. Over n extractions, the extractor can potentially incur 2^n executions.

- $\mathcal{B}$ fetches the uid for the honest card $\mathcal{C}$ specified by $\mathcal{A}$.
- Obtains signature (A,e,s) by querying the challenger $\mathcal{D}$ on message $(\mathsf{uid},m_1,\ldots,m_\ell)$.
- Sends $(\mathbf{a}_I,,A,e,s)$ to $\mathcal{A}$. Note that $(\mathcal{C},*,\mathbf{m})$ is simultaneously installed in $\mathcal{F}$.

We note that if extraction succeeds, the signature (A,e,s) computed above is distributed identically to the one output by issuer in $\mathbf{G}_2$. Let ε' denote the negligible probability, that one of the extractions does not output a valid witness. Then $\mathcal{A}$ cannot distinguish between the experiment with the forger $\mathcal{B}$ and the game $\mathbf{G}_2$ except with probability ε'. Thus, with probability $\varepsilon - \varepsilon'$, $\mathcal{A}$ outputs a proof π' on behalf of a corrupt holder in the presentation phase which satisfies all verification constraints, but the subsequent $(\mathsf{VERIFYCOMPLETE},\mathsf{vid},1)$ as in game $\mathbf{G}_3$ would have output 0. Invoking the extractor as in in Theorem 4.1, $\mathcal{B}$ extracts a signature (A,e,s) over message $(\mathsf{uid}',\mathbf{m}')$ such that $\mathbf{m}'[i] = m_i$ for all $i \in V$. We note that the proof π' in the presentation phase is not required to be online extractable, as in the indistinguishability argument, we only need to extract from one such proof.

Now uid' equals the uid of an honest card not joined to $\mathcal{A}$ with negligible probability (say ε''). If uid' corresponds to card joined to $\mathcal{A}$ but $\mathbf{G}_3$ outputs 0, it implies $\mathbf{m}'$ is not present in the list of signed messages maintained by $\mathcal{F}$ (as part of $\mathcal{B}$). Thus $\mathcal{B}$ outputs the message $(\mathsf{uid}',\mathbf{m}')$ and (A,e,s) as the forgery with probability at least $\varepsilon - \varepsilon_0$ where $\varepsilon_0 = \varepsilon' + \varepsilon''$. Since the signature scheme is EUF-CMA secure we conclude that ε must be negligible. This completes the argument that both $\mathbf{G}_2$ and $\mathbf{G}_3$ produce identical outputs.

Finally, we need to show that messages to the adversary $\mathcal{A}$ are indistinguishable between $\mathbf{G}_2$ and $\mathbf{G}_3$. When $\mathcal{H}$ is corrupt, the message $(n_{\mathcal{C}},n_{\mathcal{V}},B,\mathbf{a}_V)$ from $\mathcal{V}$ is indistinguishable to $\mathcal{A}$ when $n_{\mathcal{C}},n_{\mathcal{V}},B$ are chosen as $n_{\mathcal{C}},n_{\mathcal{V}} \leftarrow \{0,1\}^\lambda$, $B \leftarrow \mathbb{G}_1$ for the case where the card $\mathcal{C}$ is not joined to $\mathcal{H}$. This is because $\mathcal{A}$ does not know the PRF seed K used by honest card $\mathcal{C}$ to generate $B = Q \cdot h_0^{\mathsf{PRF}_K(n)}$ with $n = n_{\mathcal{C}}||n_{\mathcal{H}}$ and so B appears independent of n as with overwhelming probability the values n are distinct throughout the protocol (and PRF is indistinguishable from a random function). Otherwise it is simulated as $n_{\mathcal{C}},n_{\mathcal{V}} \leftarrow \{0,1\}^\lambda$, $B = Q \cdot h_0^{\mathsf{PRF}_K(n)}$ with $n = n_{\mathcal{C}}||n_{\mathcal{H}}$ for (Q,K) corresponding to $\mathcal{C}$ (which is leaked to $\mathcal{H}$ in this case). Indistinguishability of simulation for the case of corrupt $\mathcal{V}$ (and also corrupt $\mathcal{V},\mathcal{C}$) follows from observing: Whenever $\mathcal{V}$ modifies $n_{\mathcal{H}},n_{\mathcal{C}}$ or B, the real protocol outputs $(\mathsf{vid},\perp,\ldots,\perp)$ with overwhelming probability, and so does the simulator. Similarly, when corrupt $\mathcal{V}$ truthfully forwards the values the real protocol outputs $(\mathsf{vid},\perp,\ldots,\perp)$ as message (5), when either $\mathcal{C}$ and $\mathcal{H}$ have not been joined (their PRF seeds are different), or they have not been joined with attributes satisfying the predicate $\mathbf{a}_V$. These cases result in output $(\mathsf{VERIFIED},\mathsf{vid},0)$ for $\mathcal{S}$ which accordingly simulates the message as $(\mathsf{vid},\perp,\ldots,\perp)$. In the remaining case when the card and holder have been joined with the satisfying attributes, $\mathcal{F}$ outputs $(\mathsf{VERIFIED},f,1)$ to the simulator $\mathcal{S}$ which the simulates according to the simulator in zero-knowledge property in Theorem 4.1. This proves that messages to the adversary are indistinguishable across both the games. In other words, when the verifier truthfully forwards $n_{\mathcal{C}},n_{\mathcal{H}}$ and B, the simulator can determine if it needs to simulate the proof $(\mathsf{vid},\perp,\ldots,\perp)$ or an honest proof using Theorem 4.1 by querying $\mathcal{F}$.

Game $\mathbf{G}_4$: $\mathcal{F}$ stops forwarding inputs in Join In this game, $\mathcal{F}$ stops forwarding the inputs of honest parties in the join interface to $\mathcal{S}$, and instead interfaces with $\mathcal{E}$ and $\mathcal{S}$ according to the join interface of $\mathcal{F}_{\mathsf{cbAC}}$. This makes $\mathcal{F}$ in this game identical to $\mathcal{F}_{\mathsf{cbAC}}$. We introduce following changes to the simulator: $\mathcal{S}$ no longer runs the honest parties in setup and join phases. Instead $\mathcal{S}$ explicitly maintains the state of honest parties and uses it to simulate the messages from honest parties to the adversary. The following code describes the final simulator $\mathcal{S}$. In the code below $\mathcal{S}$ initializes records $\mathsf{JR}(x)$ with attributes $(\mathtt{C} \leftarrow \perp, \mathtt{H} \leftarrow \perp)$ in case they do not exist when being updated.

- On input $(\mathsf{SETUP}, \mathsf{sid})$ from $\mathcal{F}_{\mathsf{cbAC}}$
  - Check that $\mathsf{sid} = (I, \mathsf{sid}')$.
  - Choose $x \leftarrow \mathbb{Z}_p$, $\bar{g}_1 \leftarrow \mathbb{G}_1$. Compute $w \leftarrow g_2^x$.
  - Set $vk := (\bar{g}_1, \bar{g}_1^x, w)$.
  - Output $(\mathsf{SETUPDONE}, \mathsf{sid})$.

- On message $(\mathsf{JOINID}, \mathsf{jid}, \mathcal{C})$ from $\mathcal{F}_{\mathsf{cbAC}}$:
  - If entry $(\mathcal{C}, *, *)$ does not exist, samples $\mathsf{uid} \leftarrow \mathbb{Z}_p$, $K \leftarrow \mathcal{K}$ and stores $(\mathcal{C}, \mathsf{uid}, K)$.
  - Updates $\mathsf{JR}(\mathsf{jid}).\mathtt{C} \leftarrow \mathcal{C}$.
- On message $(\mathsf{JOIN}, \mathsf{jid}, H')$ from $\mathcal{F}_{\mathsf{cbAC}}$:
  - $\mathcal{S}$ updates $\mathsf{JR}(\mathsf{jid}).\mathtt{H} \leftarrow \mathcal{H}$, $\mathsf{JR}(\mathsf{jid}).\mathtt{setH} \leftarrow H'$.
- When $\mathcal{A}$ delivers the message $(\mathsf{jid}, C, \pi', H)$ on behalf of corrupt $\mathcal{H}$:
  - $\mathcal{S}$ extracts $s', \{m_i\}_{i \in H} \leftarrow \mathcal{E}^{\mathcal{A}}((C, \{h_{i+1}\}_{i \in H}, n_I), \pi')$ if the proof π' verifies for the statement $(C, \{h_{i+1}\}_{i \in H}, n_I)$, else it aborts. Here n_I is the nonce generated by issuer as part of message (3) in the join protocol in Figure 5.
  - It sets $\mathbf{a}_H = \{(i, m_i) : i \in H\}$, stores $(\mathsf{jid}, \mathbf{a}_H, s')$ and calls $(\mathsf{JOIN}, \mathsf{jid}, \mathbf{a}_H)$ interface of $\mathcal{F}$ on behalf of some corrupt holder $\mathcal{H}'$.
- On message $(\mathsf{JOINISSUE}, \mathsf{jid})$ from $\mathcal{F}_{\mathsf{cbAC}}$:
  - Honest $\mathcal{H}$
    * Call $(\mathsf{JOINCOMPLETE}, \mathsf{jid})$ on $\mathcal{F}$ after recieving $(\mathsf{JOINID}, \mathsf{jid})$ and $(\mathsf{JOIN}, \mathsf{jid})$ messages from $\mathcal{F}$.
  - Corrupt $\mathcal{H}$
    * Simulates message $(\mathsf{jid}, n_{\mathcal{C}}, n_I)$ to $\mathcal{A}$ after receiving $(\mathsf{JOINID}, \mathsf{jid}, \mathcal{C})$ from $\mathcal{F}$.
    * Deliver the credential to holder and install the attributes in $\mathcal{F}$ after receiving $(\mathsf{JOINID}, \mathsf{jid}, *)$ and $(\mathsf{JOIN}, \mathsf{jid}, *)$ from $\mathcal{F}$:
      · Fetch (uid, K) from the record $(\mathcal{C}, \mathsf{uid}, K)$, and $\mathbf{a}_H, s'$ from record $(\mathsf{jid}, \mathbf{a}_H, s')$.
      · Abort if $H \neq L \setminus I$.
      · Parse $\mathbf{a}_H = \{(i, m_i) : i \in H\}$. Parse $\mathbf{a}_I = \{(i, m_i) : i \in I\}$. Let $(m_1, \ldots, m_\ell)$ be message vector contained in $\mathbf{a}_H$ and $\mathbf{a}_I$.
      · Compute $B = h_1^{\mathsf{uid}} h_0^{\mathsf{PRF}_K(n_{\mathcal{C}})}$, $C = h_0^{s'} \prod_{i \in H} h_{i+1}^{m_i}$.
      · Compute $e \leftarrow \mathbb{Z}_p \setminus \{x\}$, $s \leftarrow \mathbb{Z}_p$, $A \leftarrow \left(g_1 h_0^s h_1^{\mathsf{uid}} \prod_{i=1}^{\ell} h_{i+1}^{m_i}\right)^{1/(e+x)}$. Here the simulator takes the signing key x from simulated issuer "I".
      · Simulate $(\mathsf{jid}, \mathbf{a}_I, A, e, s)$ towards $\mathcal{A}$.
      · Call $(\mathsf{JOINCOMPLETE}, \mathsf{jid})$ on $\mathcal{F}$.
- Corrupt I: The messages to the corrupt issuer are simulated as follows, where the last two steps apply only if the holder $\mathcal{H}$ is honest.
  - On message $(\mathsf{JOINID}, \mathsf{jid}, \mathcal{C})$ from $\mathcal{F}_{\mathsf{cbAC}}$:
    * If no record $(\mathcal{C}, *, *)$ exists, sample $\mathsf{uid} \leftarrow \mathbb{Z}_p$, $K \leftarrow \mathcal{K}$ and store $(\mathcal{C}, \mathsf{uid}, K)$;
    * Set $\mathsf{JR}(\mathsf{jid}).\mathtt{C} = \mathcal{C}$.
  - On message (jid, n_I) from $\mathcal{A}$:
    * Simulate message $(\mathsf{jid}, n_{\mathcal{C}}, B, \pi)$ but only after $(\mathsf{JOINID}, \mathsf{jid}, \mathcal{C})$ is received from $\mathcal{F}_{\mathsf{cbAC}}$. The message is simulated by choosing $n_{\mathcal{C}} \leftarrow \{0,1\}^{\lambda}$, $r \leftarrow \mathsf{PRF}_K(n_{\mathcal{C}})$, $B \leftarrow h_1^{\mathsf{uid}} h_0^r$ and $\pi \leftarrow \mathsf{SDL.Sim}(B, h_0, h_1, n_I)$ where (uid, K) are fetched from the record $(\mathcal{C}, \mathsf{uid}, K)$.
  - On message $(\mathsf{jid}, n'_{\mathcal{C}}, n'_I)$ from $\mathcal{A}$:
    * Simulate message (C, H, π') from $\mathcal{H}$ by choosing $C \leftarrow \mathbb{G}_1$, $H = \mathsf{JR}(\mathsf{jid}).\mathtt{setH}$ and $\pi' \leftarrow \mathsf{SDL.Sim}(C, \{h_{i+1}\}_{i \in H}, n_I)$ but only after receiving $(\mathsf{JOIN}, \mathsf{jid}, *)$ from $\mathcal{F}_{\mathsf{cbAC}}$.
  - On message $(\mathsf{jid}, \mathbf{a}_I, A, e, s)$ from $\mathcal{A}$:
    * Abort if $e(A, w g_2^e) \neq e(g_1 h_0^s \cdot B \cdot C \cdot \prod_{i \in I} h_{i+1}^{m_i}, g_2)$. Here we assume $\mathbf{a}_I = \{(i, m_i) : i \in I\}$.
    * Send $(\mathsf{JOINISSUE}, \mathsf{jid}, \mathbf{a}_I)$ to $\mathcal{F}_{\mathsf{cbAC}}$ and on immediate output $(\mathsf{JOINISSUE}, \mathsf{jid})$ send $(\mathsf{JOINCOMPLETE}, \mathsf{jid})$ to $\mathcal{F}_{\mathsf{cbAC}}$.

We now argue that game $\mathbf{G}_4$ is indistinguishable from game $\mathbf{G}_3$. We first notice that $\mathcal{S}$ explicitly maintains the state of honest cards, instead of implicitly having that as part of simulated party "$\mathcal{C}$". This change is only syntactic. We now show that a card $\mathcal{C}$ and a holder $\mathcal{H}$ are joined with attributes $\mathbf{m}$ in $\mathbf{G}_4$ if and only if they are joined with the same attributes in $\mathbf{G}_3$. This is obvious in the case of honest $\mathcal{H}$ as both the real protocol in $\mathbf{G}_3$ and simulator in $\mathbf{G}_4$ abort only if the pre-signature (A, e, s) obtained from the issuer is incorrect. Hence both the games install identical attributes by calling $\mathsf{JOINCOMPLETE}$.

For the corrupt holder, as long as the issuer is honest, the checks made in $\mathbf{G}_4$ before calling $\mathsf{JOINCOMPLETE}$ are identical to the checks made by simulated issuer in $\mathbf{G}_3$, and hence identical set of attributes are installed in both the games. When both the holder and the issuer are corrupt, no credentials are installed in $\mathcal{F}_{\mathsf{cbAC}}$ in both the games. Finally, we notice that for the case of corrupt holder, the simulator has the state of all honest parties, to simulate the messages towards the adversary in the join phase. Further, both games are identical during presentation phase as its only dependence on the join phase is the list creds of installed attributes which is identical in both the games. This proves the indistinguishability of $\mathbf{G}_3$ and $\mathbf{G}_4$ and completes the proof of the theorem.

□