Delegatable Keyed-Verification Anonymous Credentials

Omid Mirzamohammadi[1], Varesh Mishra[1], Jan Bobolz[2], Emad Heydari Beni[1,3], Aysajan Abidin[1], Dave Singelée[4], and Bart Preneel[1]

[1] COSIC, KU Leuven, Leuven, Belgium
firstname.lastname@esat.kuleuven.be
[2] Hasso Plattner Institute, University of Potsdam, Germany
jan.bobolz@hpi.de
[3] Nokia Bell Labs, Antwerp, Belgium
[4] DistriNet-Campus Group T, KU Leuven, Leuven, Belgium
dave.singelee@kuleuven.be

Abstract. Keyed-verification anonymous credentials (KVACs) are a variant of anonymous credentials (ACs) in which the issuer and the verifier are either the same entity or mutually trusted entities. Due to this design, KVACs can avoid some of the practical burdens of publicly verifiable ACs, such as requiring bilinear pairings, making them more practical in certain settings. Delegation of credentials is a powerful feature that enables applications where users, holding a credential, can delegate a restricted version of their credential to another user. This paper proposes the first delegatable keyed-verification anonymous credentials (DKVAC). We provide formal definitions of DKVAC, we design two pairing-free DKVAC constructions with delegation capability, we prove them secure, and we demonstrate their practicality with implementation benchmarks. Our two constructions present trade-offs between expressiveness and performance.

Keywords: Delegatable Anonymous Credentials, Keyed-Verification Anonymous Credentials, Zero-Knowledge Proofs

1 Introduction

Anonymous credential (AC) systems [Cha82, CL01] enable users to obtain credentials on a set of attributes and subsequently prove possession of these credentials while revealing only selected information. Unlike traditional authentication mechanisms, ACs provide strong privacy guarantees such as selective disclosure and unlinkability. A user can prove possession of attributes required for an interaction without revealing unrelated attributes and without allowing different credential presentations to be linked together. These properties make anonymous credentials attractive for a wide range of applications, including access control, digital identity, transportation systems, healthcare and e-government services.

Delegatable credentials. A desirable feature for anonymous credentials is that of delegation: A user holding a credential Cred should be able to *delegate* an attenuated version of their credential to another user. In early influential works [CL06, BCC+09, CL19], attenuation was modeled via credential levels. Holding a level L credential, one can only delegate credentials with higher levels $L' \geq L$. We instead follow an *attribute-based* notion of delegatable credentials (e.g., [BB18, MSBM23, PM23]), where a credential with attributes $\mathbf{M}$ can only delegate credentials with "attenuated" attributes $\mathbf{M}' \preceq \mathbf{M}$ for some relation $\preceq$. For example the user may be able to delegate subsets $\mathbf{M}' \subseteq \mathbf{M}$ of their set $\mathbf{M}$ of attributes.

Keyed-verification anonymous credentials. A particularly efficient class of anonymous credentials is formed by keyed-verification anonymous credentials (KVACs) [CMZ14]. In a KVAC system, the issuer and verifier share a secret key. This differs from publicly verifiable AC, where verification can be performed using only public information. While this additional trust assumption restricts the deployment model, it is natural in many practical settings where issuers and verifiers already belong to the same administrative domain or are operated by mutually trusted entities. Examples include university access-control systems, transportation providers, and enterprise identity management systems. In such environments, KVACs offer a compelling alternative because they can avoid expensive cryptographic tools commonly used in publicly verifiable credentials. Importantly, we can construct KVACs in groups *without pairings*. This often leads to more efficient constructions, especially for verifiers, which may operate on constrained hardware and must securely store secret verification material. Because of these desirable properties, KVACs have been deployed in practice [CPZ20, TG23, Orr25].

Delegatable KVACs. Despite the extensive literature on both delegatable and keyed-verification anonymous credentials (see extended related work in Appendix A), there is no prior work on the combination, i.e., delegatable keyed-verification anonymous credentials (DKVAC).[5] These are delegatable credential systems that do not rely on pairings. As a trade-off, credential verification requires the issuer's secret key (as in KVAC). We initiate their study. Our main research question is the following.

"To what extent can delegatable anonymous credentials be realized without public verification and without pairings?"

Scenarios for DKVACs. First, we discuss what applications DKVACs are used for. We observe that there is an unusual imbalance in such systems: On the one hand, great care is taken to restrict credential holders from delegating more access rights than they have themselves. On the other hand, the verifier holds the issuer's key and hence can issue arbitrary credentials. This means that the

[5] The closest work, DkvSSO [XYH+25], supports delegation of verification tokens rather than credentials, and its delegation requires the issuer's key; see Appendix A.

verifier is trusted *more* than a delegator, which is different from the publicly verifiable setting, where we often trust the issuer the most, the delegators less, and the verifier the least. While that rules out some scenarios, there are still many use cases for such systems.

Consider a building access system. A university administration may issue a credential that grants access to all buildings and facilities. A department administrator should then be able to delegate a restricted credential to a visitor, contractor, or temporary employee. The delegated credential may authorize access only to a subset of locations and only during a specified time interval. Importantly, the delegatee should still benefit from the privacy guarantees of anonymous credentials. When presenting the delegated credential at different access points, the resulting presentations should remain unlinkable so that the delegatee's movements cannot be tracked by the access-control infrastructure.

A similar scenario arises in public transport systems. A transportation provider may hold a credential that authorizes access to all stations within a network. Upon ticket purchase, the provider should be able to delegate a credential corresponding only to a particular geographical zone and validity period. The passenger should subsequently be able to present this delegated credential to access gates throughout the network without enabling the operator to link different uses of the credential.

In general, these examples share several common requirements: (1) the verifier is often operated by the same organization as the issuer, (2) credential presentations must remain unlinkable, (3) delegation must support privilege attenuation, and (4) delegated credentials should themselves remain further delegatable.

Observation: key-parameter consistency vs delegator anonymity. An important property of KVACs is *key-parameter consistency*, which guarantees that credentials received by honest users are valid under the issuer's (committed) secret key. Without this property, we cannot guarantee meaningful privacy: A corrupt issuer could issue credentials under a different key for each issuance and then the corrupt verifier can simply try verification under all possible issuer keys to link the credential's presentation to its issuance. On the delegatable credential side, a desirable property is *delegator privacy*, which hides the delegator identity, attributes, and any possible linking information from delegatees.

This leads to our first observation around delegatable KVACs: *We cannot hope to achieve both key-parameter consistency and delegator privacy at the same time.* To see this, assume we had a construction that achieves both. The user can simply delegate attributes $\mathbf{M}'$ to the verifier in order to prove ownership of these attributes (note that the verifier knows the issuer's key, so it does not gain anything new from this). The verifier simply checks that the received credential is valid via key-parameter consistency. Due to delegator privacy, this kind of presentation is privacy-preserving. However, in this presentation protocol, the verifier does not use its secret key. Such a possibility would be highly surprising in a KVAC and suggests a fundamental tension between these two properties.

Faced with the choice of either key-parameter consistency or delegator privacy, we choose to keep key-parameter consistency, as it is a crucial property

for basic user privacy against the issuer/verifier. Previous works have explored delegatable credentials without delegator privacy [CDD17], noting that usually, users already know the parties they receive a credential from.

The challenges of constructing DKVACs. Our goal is to construct a delegatable keyed-verification anonymous credential system without pairings. At first, this goal seems unattainable: Most delegatable credential constructions [CL19, CL21, PM23, MSBM23] are based on variations of structure-preserving signatures on equivalence classes (SPS-EQ). Unfortunately, there seems to be little hope to construct SPS-EQ *without pairings*, even when restricting ourselves to keyed verification [MBS+25].

The delegatable credential system of Camenisch et al. [CDD17] is not built on message-malleable signatures, which comes at the cost of delegator privacy (which we do not plan to support anyway). However, their approach is that the delegator signs the public key of the delegatee, hence a credential is a chain of signatures. To present, the user proves knowledge of that chain of signatures. This approach seems to be inherently incompatible with keyed-verification: the user cannot know the (secret) verification keys of their chain of MACs (and hence cannot prove the chain's validity). Additionally, in our DKVAC scenario, an important goal is verifier efficiency, ideally independent of the delegation chain length.

Besides finding a good delegation mechanism and MAC construction to base our construction on, another challenge is ensuring key-parameter consistency (and hence privacy). As discussed above, key-parameter consistency ensures that credentials held by users are valid under the issuer's secret key. Since users cannot check credential validity themselves, the issuer in KVAC constructions generally provides a zero-knowledge proof that the issued credential is valid. How can we maintain this guarantee when delegating credentials? The delegator, not knowing the issuer's key, cannot create a new proof that the delegated credential is valid. The issuer's original proof can only be verified using the original credential, but the delegatee must not learn the original credential (only an attenuated version).

Our contributions. We formally define delegatable keyed-verification anonymous credentials and their security, design two constructions, and prove them secure in the generic group model.

In our first construction, a credential authenticates a set $\mathbf{M} \subset \mathbb{Z}_p$ of attributes. Users can delegate subsets $\mathbf{M}' \subseteq \mathbf{M}$ of their attributes to other users. Anyone with a credential or delegated credential can present it (partial disclosure) or further delegate it. Perhaps surprisingly, presentations consist of only two group elements, independent of the number of attributes or delegation chain length. In the (even non-delegatable) KVAC setting, this was previously only achieved by [MBS+25], using different techniques.

Our second construction is based on a (publicly verifiable) credential system [BB18], which uses ("dynamically") malleable signatures. We show that this specific kind of malleable signatures (in unexpected contrast to SPS-EQ) can be

made to work without pairings in the keyed verification setting. In this construction, a credential authenticates a vector $\mathbf{M} \in \mathbb{Z}_p^t$ of attributes $(s_1, \ldots, s_t)$, where some of the s_i are *malleable*. When users delegate $(s'_1, \ldots, s'_t)$, they can change the malleable s_i to s'_i, but they cannot change the values of non-malleable s_j. Furthermore, the user can restrict the set $\mathbf{L} \subseteq [t]$ of malleable indices during delegation to any subset $\mathbf{L}' \subseteq \mathbf{L}$. This enables some more flexibility, e.g., allowing a delegator to change an expiration timestamp as an attribute during delegation. The cost of this additional flexibility in the second construction is that presentations are no longer constant-size.

Depending on the application, one can choose one of the two constructions. The first offers faster presentation and verification in our benchmarks, with constant-size presentations, and supports applications where attenuation consists of removing attributes. The second construction has a more complex presentation protocol but a more expressive attenuation notion, supporting additional applications (such as expiration dates), with delegation objects of similar size in the tested configurations (see Table 1).

In both constructions, we solve the key-parameter consistency problem described above by carefully encrypting parts of the credential. The issuer proves, in zero-knowledge, that the encrypted values encode a valid credential. The user also gets the contents of the encryption, making the credential usable. When delegating, the delegatee receives the original encrypted credential and the proof, and hence can check the issuer's proof without being able to use the original credential. The delegator also attaches a proof that they correctly delegated the encrypted credential. Because the issuer's proof does not change between delegations, delegatees learn some linkability information about the credential used by the delegator. This is where we sacrifice delegator privacy, which is inherently unavoidable, as discussed above.

Our constructions emphasize performance, and, like many efficient KVAC constructions (e.g., [CMZ14, MBS+25]) and delegatable credential constructions (e.g., [BB18, CL19, MSBM23]), our security proofs rely on the generic group model. We implemented both of our DKVAC constructions and performed performance benchmarks to demonstrate that these schemes are highly practical.

Overall, our constructions demonstrate that delegation does not inherently require public verification. While many existing delegatable credential systems derive their functionality from publicly verifiable signatures and pairing-based techniques, our results show that meaningful forms of delegation can be realized in pairing-free KVAC systems. At the same time, the design choices required to achieve this reveal fundamental differences between delegation in publicly verifiable credentials and delegation in keyed-verification credentials.

2 Preliminaries

The security parameter is λ, and PPT means probabilistic polynomial time. Vectors and sets are written in bold. We write $u \xleftarrow{\$} U$ for uniform sampling and $u \leftarrow \mathcal{A}(x)$ for a randomized algorithm's output. Let $[n] = \{1, \ldots, n\}$ for an

integer n. A function $\mathsf{negl}(\lambda)$ is called *negligible* if, for every positive polynomial $p(\cdot)$, there exists λ_0 such that, for all $\lambda > \lambda_0$, it holds that $\mathsf{negl}(\lambda) < \frac{1}{p(\lambda)}$. We use additive notation for a cyclic group $\mathbb{G}$ of prime order $p > 2^\lambda$, with generator G and identity $0_\mathbb{G}$. Scalars lie in $\mathbb{Z}_p$, and $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$. For a relation $\mathcal{R}$, a proof system $\mathsf{NIP} = (\mathsf{Prove}, \mathsf{Verify})$ has algorithms $\pi \leftarrow \mathsf{Prove}^{\mathcal{O}_{\mathrm{RO}}}(\mathbb{x}, \mathbb{w})$ and $\mathsf{Verify}^{\mathcal{O}_{\mathrm{RO}}}(\mathbb{x}, \pi) \in \{0,1\}$, where $\mathcal{O}_{\mathrm{RO}}$ is a random oracle (with domain separation in case of multiple proof systems). We require completeness, soundness, and zero knowledge. We use Camenisch-Stadler notation [CS97] to specify the relations. Appendix B provides formal definitions and the Fiat-Shamir generalized-Schnorr construction.

3 Delegatable Keyed-Verification Anonymous Credentials

We introduce several terms that parameterize the construction.

Message and Message Space. A credential authenticates a message $\mathbf{M} \in \mathcal{M}$, where $\mathcal{M}$ denotes the *message space*. The message $\mathbf{M}$ represents the full authenticated object. It is known to the credential holder, but it need not be fully revealed during presentation.

Presentation Policy. During presentation, the verifier specifies a public predicate, called a *policy*, $\phi \in \Phi$, where $\phi : \mathcal{M} \to \{0,1\}$. A presentation convinces the verifier that the credential holder has a valid credential on some authenticated message $\mathbf{M}$ satisfying $\phi(\mathbf{M}) = 1$, while revealing no information about $\mathbf{M}$ beyond what is intentionally disclosed by ϕ. Thus, the disclosed information is encoded as part of the policy. Since the policy is public and explicitly specified by the verifier, it is given as public input to the showing and verification algorithms.

Delegation Relation. Delegating a credential on a message $\mathbf{M}$ yields a credential on a new message $\mathbf{M}'$. This delegation is admissible exactly when $M' \preceq M$, for a fixed relation $\preceq$ over the message space $\mathcal{M}$. We call $\preceq$ the *delegation relation*.

Delegation Safety. The delegation relation $\preceq$ must be reflexive and transitive. Reflexivity captures trivial delegation: a credential can always be regarded as a delegation of itself, namely $\mathbf{M} \preceq \mathbf{M}$. Transitivity captures the safety of delegation chains. If a credential on $\mathbf{M}$ can be delegated to $\mathbf{M}'$, and a credential on $\mathbf{M}'$ can further be delegated to $\mathbf{M}''$, then $\mathbf{M}''$ must also be a permissible delegation of the original message $\mathbf{M}$:

$$(\mathbf{M}'' \preceq \mathbf{M}') \wedge (\mathbf{M}' \preceq \mathbf{M}) \implies \mathbf{M}'' \preceq \mathbf{M}$$

Thus, delegation can only restrict the authority represented by a credential, or leave it unchanged.

Syntax. A DKVAC scheme for message space, policy class, and delegation relation $(\mathcal{M}, \Phi, \preceq)$ consists of the following algorithms. We first describe the syntax of a KVAC scheme.

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: On input a security parameter λ, the setup algorithm outputs public parameters pp.
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{isk}, \mathsf{ipar}, \mathcal{M})$: On input the public parameters pp, the key-generation algorithm outputs an issuer secret key isk and public issuer parameters ipar. Here $\mathcal{M}$ specifies the admissible message domain associated with the key; only ipar is published. The notation does not require publishing a separate description of $\mathcal{M}$ that exposes secret-key information.
- $\mathsf{IssueCred}(\mathsf{isk}, \mathsf{ipar}, \mathbf{M}) \to (\mathsf{Cred}, \pi)/\perp$: On input the issuer secret key isk, issuer parameters ipar, and a message $\mathbf{M}$, the issuance algorithm returns $\perp$ if $\mathbf{M} \notin \mathcal{M}$. Otherwise, it outputs a credential Cred together with a proof of well-formedness π.
- $\mathsf{ObtainCred}(\mathsf{ipar}, \mathbf{M}, \mathsf{Cred}, \pi) \to \mathsf{Cred}/\perp$: On input issuer parameters ipar, a message $\mathbf{M}$, a credential Cred, and a proof π, the obtain algorithm verifies π. If the proof is valid, it outputs Cred; otherwise, it outputs $\perp$.
- $\mathsf{ShowCred}(\mathsf{ipar}, \mathbf{M}, \phi, \mathsf{Cred}) \to \mathsf{Show}/\perp$: On input issuer parameters ipar, a message $\mathbf{M}$, a policy $\phi \in \Phi$, and a credential Cred, the showing algorithm checks whether $\phi(\mathbf{M}) = 1$. If so, it outputs a presentation Show of Cred with respect to ϕ, using fresh randomness on each invocation; otherwise, it outputs $\perp$.
- $\mathsf{VerifyShow}(\mathsf{isk}, \phi, \mathsf{Show}) \to \{0, 1\}$: On input the secret verification key isk, a policy ϕ, and a presentation Show, the verification algorithm checks whether Show is a valid presentation of a credential on some message satisfying ϕ. It outputs 1 if verification succeeds, and 0 otherwise.

We now describe the three additional algorithms that extend the syntax of a KVAC scheme to a DKVAC scheme.

- $\mathsf{IssueDel}(\mathsf{isk}, \mathsf{ipar}, \mathbf{M}) \to \mathbf{Del}/\perp$: On input the issuer secret key isk, issuer parameters ipar, and a message $\mathbf{M}$, the algorithm returns $\perp$ if $\mathbf{M} \notin \mathcal{M}$. Otherwise, it issues a *delegatable* credential on $\mathbf{M}$ and outputs a delegation object $\mathbf{Del}$.
- $\mathsf{Delegate}(\mathbf{Del}, \mathbf{M}') \to \mathbf{Del}'/\perp$: On input a delegation object $\mathbf{Del}$ and a target message $\mathbf{M}'$, the delegation algorithm outputs new $\mathbf{Del}'$ authenticating only the delegated message $\mathbf{M}'$. It may output $\perp$ in error cases (e.g., $M' \not\preceq M$).
- $\mathsf{ObtainDel}(\mathsf{ipar}, \mathbf{Del}) \to (\mathsf{Cred}, \mathbf{M})/\perp$: On input issuer parameters ipar and a delegation object $\mathbf{Del}$, the obtain-delegated-credential algorithm generates a credential Cred on the final message $\mathbf{M}$ and outputs $(\mathsf{Cred}, \mathbf{M})$, or returns $\perp$ if validation fails. This algorithm is deterministic: for fixed inputs and a fixed random oracle, it always returns the same output, including rejection. The obtained credential Cred has the same structure as a directly obtained credential from $\mathsf{ObtainCred}$ algorithm, and therefore it can be presented using $\mathsf{ShowCred}$ in the same way.

A DKVAC scheme extends the underlying KVAC scheme by replacing IssueCred and ObtainCred with their delegatable counterparts IssueDel and ObtainDel, and by adding a new algorithm Delegate for transferring and attenuating credentials. The algorithm IssueDel outputs a delegation object, which may be updated through Delegate. The algorithm ObtainDel converts a valid delegation object into an ordinary credential. Consequently, the obtained credential has the same form as one produced by ObtainCred and can be presented and verified using the standard ShowCred and VerifyShow algorithms of the underlying KVAC scheme.

DKVAC Security. A secure DKVAC should have the following properties.

Definition 1 (Correctness). *A DKVAC scheme is correct if, for every security parameter λ, every* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(\mathsf{isk}, \mathsf{ipar}, \mathcal{M}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*, every sequence* $\mathbf{M}_1 \succeq \cdots \succeq \mathbf{M}_n$ *of messages in* $\mathcal{M}$ *and delegation objects* $\mathbf{Del}_1 \leftarrow \mathsf{IssueDel}(\mathsf{isk}, \mathsf{ipar}, \mathbf{M}_1)$ *and* $\mathbf{Del}_i \leftarrow \mathsf{Delegate}(\mathbf{Del}_{i-1}, \mathbf{M}_i)$ *for* $1 < i \leq n$*, and any policy* $\phi \in \Phi$ *such that* $\phi(\mathbf{M}_n) = 1$*,*

$$\Pr\left[\begin{array}{l}(\mathsf{Cred}, \mathbf{M}) \leftarrow \mathsf{ObtainDel}(\mathsf{ipar}, \mathbf{Del}_n); \\ \mathsf{Show} \leftarrow \mathsf{ShowCred}(\mathsf{ipar}, \mathbf{M}, \phi, \mathsf{Cred}); \\ b \leftarrow \mathsf{VerifyShow}(\mathsf{isk}, \phi, \mathsf{Show})\end{array} : \begin{array}{r} b = 1 \\ \wedge\ \mathbf{M} = \mathbf{M}_n \end{array}\right] = 1\ .$$

Unforgeability requires that no probabilistic polynomial-time (PPT) adversary can produce a valid presentation Show for a policy $\phi \in \Phi$, unless it has previously obtained a valid credential on some message $\mathbf{M}$ such that $\phi(\mathbf{M}') = 1$ for some $\mathbf{M}' \preceq \mathbf{M}$. For this, the adversary can generate (delegated) credentials via $\mathcal{O}_{\mathsf{Cred}}$ (and $\mathcal{O}_{\mathsf{Delegate}}$), which models operations for honest users. The adversary can receive a credential through $\mathcal{O}_{\mathsf{Reveal}}$.

Definition 2 (Unforgeability). *A DKVAC scheme is unforgeable if for every PPT adversary $\mathcal{A}$,*

$$\Pr\left[\mathsf{Exp}^{\mathsf{Unforge}}_{\mathcal{A},\mathsf{DKVAC}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,,$$

where $\mathsf{Exp}^{\mathsf{Unforge}}_{\mathcal{A},\mathsf{DKVAC}}$ *is defined in Figure 1.*

Multi-show unlinkability requires that a presentation Show reveals nothing about the underlying credential beyond the information disclosed by the policy $\phi \in \Phi$. We model this as a bit guessing indistinguishability game. The adversary plays the role of corrupted issuers, delegators, and verifiers. It is provided a left-or-right oracle $\mathcal{O}_{\mathsf{LoR}}$, which takes two delegation objects and a common policy, and either always generates presentations from the first object or always from the second. The adversary may repeat delegation objects across queries. Since ObtainDel is deterministic, repeating an object yields the same obtained credential; only the presentation randomness is refreshed. Thus the game also covers repeated presentations of a credential obtained once and then stored by its holder.

$\underline{\mathsf{Exp}^{\mathsf{Unforge}}_{\mathcal{A},\mathsf{DKVAC}}(\lambda)}$
$\mathsf{ctr} := 0$
Initiate two empty lists $\mathcal{E}$ and $\mathcal{Q}$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{isk}, \mathsf{ipar}, \mathcal{M}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$
$(\mathsf{Show}^*, \phi^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{ipar})$
return $\begin{pmatrix}\mathsf{VerifyShow}(\mathsf{isk}, \phi^*, \mathsf{Show}^*) \\ \wedge(\nexists \mathbf{M} \in \mathcal{Q} : \phi^*(\mathbf{M}) = 1)\end{pmatrix}$

$\underline{\mathcal{O}_{\mathsf{Cred}}(\mathbf{M})}$
$\mathbf{Del} \leftarrow \mathsf{IssueDel}(\mathsf{isk}, \mathsf{ipar}, \mathbf{M})$
if $\mathbf{Del} = \bot$: **return** $\bot$
$\mathsf{ctr} := \mathsf{ctr} + 1$
$\mathcal{E}[\mathsf{ctr}] := (\mathbf{M}, \mathbf{Del})$
return $\mathsf{Handle} := \mathsf{ctr}$

$\underline{\mathcal{O}_{\mathsf{Delegate}}(\mathbf{M}', \mathsf{Handle})}$
parse $\mathcal{E}[\mathsf{Handle}]$ **as** $(\mathbf{M}, \mathbf{Del})$
require $\mathbf{M}' \preceq \mathbf{M}$
$\mathbf{Del}' \leftarrow \mathsf{Delegate}(\mathbf{Del}, \mathbf{M}')$
$\mathsf{ctr} := \mathsf{ctr} + 1$
$\mathcal{E}[\mathsf{ctr}] := (\mathbf{M}', \mathbf{Del}')$
return $\mathsf{Handle} := \mathsf{ctr}$

$\underline{\mathcal{O}_{\mathsf{Reveal}}(\mathsf{Handle})}$
parse $\mathcal{E}[\mathsf{Handle}]$ **as** $(\mathbf{M}, \mathbf{Del})$
$\mathcal{Q} := \mathcal{Q} \cup \{\mathbf{M}' | \mathbf{M}' \preceq \mathbf{M}\}$
return $\mathbf{Del}$

$\underline{\mathcal{O}_{\mathsf{VerifyShow}}(\mathsf{Show}, \phi)}$
return $\mathsf{VerifyShow}(\mathsf{isk}, \phi, \mathsf{Show})$

Fig. 1: Unforgeability game for DKVAC

$\underline{\mathsf{Exp}^{\mathsf{Unlink}}_{\mathcal{A},\mathsf{DKVAC}}(\lambda)}$
$b \xleftarrow{\$} \{0,1\}, \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{ipar} \leftarrow \mathcal{A}(\mathsf{pp})$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{LoR}}}$
return $b = b'$

$\underline{\mathcal{O}_{\mathsf{LoR}}(\mathbf{Del}_0, \mathbf{Del}_1, \phi)}$
$(\mathsf{Cred}_0, \mathbf{M}_0) \leftarrow \mathsf{ObtainDel}(\mathsf{ipar}, \mathbf{Del}_0)$
$(\mathsf{Cred}_1, \mathbf{M}_1) \leftarrow \mathsf{ObtainDel}(\mathsf{ipar}, \mathbf{Del}_1)$
require $\forall i \in \{0,1\} : \ \phi(\mathbf{M}_i) = 1 \wedge \mathsf{Cred}_i \neq \bot$
return $\mathsf{Show} \leftarrow \mathsf{ShowCred}(\mathsf{ipar}, \mathbf{M}_b, \phi, \mathsf{Cred}_b)$

Fig. 2: Multi-Show Unlinkability game for DKVAC

Definition 3 (Multi-Show Unlinkability). *A DKVAC scheme satisfies multi-show unlinkability if, for every PPT adversary $\mathcal{A}$, its probability of winning the unlinkability game is at most negligibly better than random guessing, namely*

$$\Pr\left[\mathsf{Exp}^{\mathsf{Unlink}}_{\mathcal{A},\mathsf{DKVAC}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)\,,$$

where $\mathsf{Exp}^{\mathsf{Unlink}}_{\mathcal{A},\mathsf{DKVAC}}$ *is defined in Figure 2.*

4 Subset DKVAC Construction

In this section, we present our subset DKVAC construction. For clarity, we first describe the underlying subset KVAC construction, and then explain how delegation can be incorporated into the construction.

4.1 Subset KVAC Construction

Our first KVAC construction, presented in Figure 3, uses an algebraic MAC whose authentication components follow the Boneh-Boyen signature [BB04]. We

$\underline{\mathsf{Setup}(1^\lambda)}$

Generate a cyclic group $\mathbb{G} = \langle \mathsf{G} \rangle$ of prime order $p > 2^\lambda$

return $\mathsf{pp} := (\mathbb{G}, \mathsf{G}, p, \mathsf{H} \xleftarrow{\$} \mathbb{G} \setminus \{0_\mathbb{G}\})$

$\underline{\mathsf{KeyGen}(\mathsf{pp})}$

$(x, y) \xleftarrow{\$} (\mathbb{Z}_p^*)^2$

$\mathsf{isk} := (x, y), \mathsf{ipar} := (X, Y) := (x\mathsf{G}, y\mathsf{G})$

$\mathcal{M} := 2^{\mathbb{Z}_p \setminus \{p-y\}}$

return $(\mathsf{isk}, \mathsf{ipar}, \mathcal{M})$

$\underline{\mathsf{IssueCred}(\mathsf{isk}, \mathsf{ipar}, \mathbf{M} := \mathbf{S})}$

parse isk **as** (x, y)

require $\forall s \in \mathbf{S} : \ y + s \neq 0$

$v \xleftarrow{\$} \mathbb{Z}_p^*$ // Tag randomness

$\tau := (V, (C_s)_{s \in \mathbf{S}}) := \left(vx\mathsf{G}, (\frac{v\mathsf{G}}{y+s})_{s \in \mathbf{S}} \right)$

// Well-formedness proof for τ

$\mathbb{x} := (\mathsf{ipar}, \tau, \mathbf{S}), \mathbb{w} := (y, v)$

$\pi \leftarrow \mathsf{NIP}_{\mathsf{SCrlss}}.\mathsf{Prove}(\mathbb{x}, \mathbb{w})$

return $(\mathsf{Cred} := \tau, \pi)$

$\underline{\mathsf{ObtainCred}(\mathsf{ipar}, \mathbf{S}, \mathsf{Cred}, \pi)}$

require $\mathsf{NIP}_{\mathsf{SCrlss}}.\mathsf{Verify}\,((\mathsf{ipar}, \mathsf{Cred}, \mathbf{S}), \pi)$

return Cred

$\underline{\mathsf{ShowCred}(\mathsf{ipar}, \mathbf{M} := \mathbf{S}, \phi_\mathbf{D}, \mathsf{Cred})}$

parse Cred **as** $\tau := (V, (C_s)_{s \in \mathbf{S}})$

$\mu \xleftarrow{\$} \mathbb{Z}_p^*$ // Randomness

require $\phi_\mathbf{D}(\mathbf{S}) = 1$

parse $\mathbf{D} \neq \emptyset$ from $\phi_\mathbf{D}$

$\tau' := (V', C') := \left(\mu V, \mu \sum_{d \in \mathbf{D}} (C_d) \right)$

return $\mathsf{Show} := \tau'$

$\underline{\mathsf{VerifyShow}(\mathsf{isk}, \phi_\mathbf{D}, \mathsf{Show})}$

parse isk **as** (x, y) and $\mathbf{D} \neq \emptyset$ from $\phi_\mathbf{D}$

parse Show **as** $\tau' := (V', C')$

if $(\exists d \in \mathbf{D} : \ y + d = 0)$: **return** 0

return $(\sum_{d \in \mathbf{D}} \frac{1}{y+d} V' = xC') \wedge (V' \neq 0_\mathbb{G})$

Fig. 3: Subset KVAC

adapt this algebraic structure to authenticate unordered sets of attributes and to support subset presentations without pairings.

Message and Message Space. A message is an unordered set of $\mathbb{Z}_p$ attributes. For an issuer key (x, y), the admissible message domain is $\mathcal{M} := 2^{\mathbb{Z}_p \setminus \{p-y\}}$. Excluding $-y \bmod p$ ensures that $(y+s)^{-1}$ is well-defined for every attribute s of an admissible message. The domain $\mathcal{M}$ is defined this way, but it is encoded without revealing y; only $\mathsf{ipar} = (X, Y)$ is published. Membership is nevertheless publicly testable: $\mathbf{S} \in \mathcal{M}$ if and only if $\mathbf{S} \subseteq \mathbb{Z}_p$ and $Y + s\mathsf{G} \neq 0_\mathbb{G}$ for every $s \in \mathbf{S}$. We write a message $\mathbf{M} \in \mathcal{M}$ simply as a set $\mathbf{M} = \mathbf{S} \subseteq \mathbb{Z}_p \setminus \{p-y\}$ of attributes. Both issuance algorithms reject malformed inputs or sets outside this domain with $\perp$, before sampling credential randomness or computing inverses.

Setup and Key Generation. On input 1^λ, the Setup algorithm generates a cyclic group $\mathbb{G}$ of prime order $p > 2^\lambda$ with generator G, and independently samples $\mathsf{H} \xleftarrow{\$} \mathbb{G} \setminus \{0_\mathbb{G}\}$, which is later used for encrypting credentials. Since $\mathbb{G}$ has prime order, every such H is a generator. It outputs the public parameters containing $(\mathbb{G}, \mathsf{G}, p, \mathsf{H})$. The issuer then runs KeyGen to sample two secret scalars

$(x, y) \xleftarrow{\$} (\mathbb{Z}_p^*)^2$. The scalar x serves as the MAC key used to authenticate credentials, while y is used to derive the per-attribute credential components. The issuer publishes $\mathsf{ipar} := (X, Y) = (x\mathsf{G}, y\mathsf{G})$. These public issuer parameters allow credential recipients to verify the well-formedness of generated credentials and to check *key-parameter consistency*, namely that the issuer uses a single secret key pair (x, y) across credentials. This prevents the issuer from using inconsistent keys in a way that could compromise privacy. The $\mathcal{M}$ output records the admissible domain defined above; neither y nor the excluded scalar $-y$ is disclosed.

Credential Issuance. When a user requests a credential on an attribute set $\mathbf{S}$, the issuer first checks that $y + s \neq 0$ for every $s \in \mathbf{S}$ and returns $\perp$ if this check fails. Otherwise, $\mathsf{IssueCred}$ samples a fresh blinding factor $v \xleftarrow{\$} \mathbb{Z}_p^*$ and computes the credential token

$$\tau := \big(V,\ (C_s)_{s\in\mathbf{S}}\big) := \left(vx\mathsf{G},\ \left(\frac{v\mathsf{G}}{y+s}\right)_{s\in\mathbf{S}}\right).$$

Each component $C_s = \frac{v\mathsf{G}}{y+s}$ is a Boneh-Boyen–style evaluation encoding the corresponding attribute value s. Intuitively, different credentials use different v, which prevents combining attributes between different credentials. The issuer also computes a proof π showing the well-formedness of τ with respect to ipar:

$$\mathcal{R}_{\mathsf{SCrIss}} := \{(y, v) : V \neq 0_{\mathbb{G}} \ \wedge\ V = vX \ \wedge\ (C_s(y + s) = v\mathsf{G})_{s\in\mathbf{S}} \ \wedge\ Y = y\mathsf{G}\}.$$

Finally, the issuer sends $(\mathsf{Cred} := \tau,\ \pi)$ to the user.

Obtaining a Credential. The user runs $\mathsf{ObtainCred}$ to verify the well-formedness proof π with respect to the public issuer parameters ipar and the attribute set $\mathbf{S}$. If the proof is valid, the user stores the credential $\mathsf{Cred} = \tau$; otherwise, it outputs $\perp$ and aborts.

Policy. For this construction, presentation is selective disclosure of attributes. A presentation policy is a nonempty set of attributes $\mathbf{D} \subseteq \mathbb{Z}_p$, and is denoted by $\phi_{\mathbf{D}}$. The policy checks that the presented credential contains all disclosed attributes, i.e., $\phi_{\mathbf{D}}(\mathbf{S}) = 1 \iff \mathbf{D} \subseteq \mathbf{S}$.

Credential Presentation with Selective Disclosure. To present a credential under a verifier-chosen policy $\phi_{\mathbf{D}}$ ($\mathbf{D} \neq \emptyset$), the holder runs $\mathsf{ShowCred}$. It selects the credential components C_d corresponding to the attributes $d \in \mathbf{D}$, samples fresh randomness $\mu \xleftarrow{\$} \mathbb{Z}_p^*$, and re-randomizes and aggregates these components as follows: $V' = \mu V$ and $C' = \mu \sum_{d\in\mathbf{D}} C_d$. The aggregation step combines the Boneh-Boyen components for the disclosed attributes into a single group element. The holder sends the presentation $\mathsf{Show} := (V', C')$ to the verifier.

Presentation Verification. The verifier runs $\mathsf{VerifyShow}$ to check whether the presentation is valid. It uses its secret key (x, y) to verify the presentation $\mathsf{Show} = (V', C')$ by checking whether $\sum_{d \in \mathbf{D}} \frac{1}{y+d} \cdot V' = x \cdot C'$. The verifier also checks that $V' \neq 0_{\mathbb{G}}$ in order to rule out trivial forgeries.

4.2 Adding Delegation to Subset KVAC

Figure 4 illustrates how delegation works in Subset DKVAC. We now describe the algorithms in detail. A delegation object is a pair consisting of a nonempty ordered chain and the decryption key for the final node of the chain:

$$\mathbf{Del} = (\mathbf{T}, \mathsf{dk}), \qquad \mathbf{T} = (\mathsf{Del}_1, \ldots, \mathsf{Del}_n), \qquad \mathsf{Del}_i = (\mathsf{EC}_i, \pi_i, \mathbf{S}_i).$$

Here $n \geq 1$ counts the chain nodes, including the issuance node. Each chain node Del_i contains an encrypted credential EC_i, a well-formedness proof π_i, and the message $\mathbf{S}_i$ that is authenticated by the node. The decryption key dk is able to decrypt the final encrypted credential EC_n, allowing the holder to recover the last node's credential Cred but not any of the other credentials on the delegation chain. Encrypting the chain allows the holder to verify the well-formedness proofs π_i without learning the credentials whose validity these proofs guarantee. Algorithms receiving a delegation object reject malformed encodings or an empty chain.

Delegation relation. For this construction, the delegation relation is $\mathbf{S}' \preceq \mathbf{S} \iff \mathbf{S}' \subseteq \mathbf{S}$. Thus, delegation can only remove attributes from the credential; it cannot add new attributes.

Delegation issuance. To enable delegation, the issuer runs $\mathsf{IssueDel}$, first checking $y + s \neq 0$ for every $s \in \mathbf{S}$ and returning $\perp$ if this check fails. For an admissible set, it generates an encrypted version EC of the root credential and sends it to the user together with its decryption key dk. To encrypt the credential, the issuer encrypts only the tag component V by computing the ElGamal encryption $(\mathsf{E}, \mathsf{EZ}) := (v x \mathsf{G} + z \mathsf{H},\ z\mathsf{G})$ of V. This makes the attribute components $(C_s)_{s \in \mathbf{S}}$ unusable without the decryption key $\mathsf{dk} := z$. The issuer also sends $\mathsf{EV} = v\mathsf{G}$ to the user, which lets the well-formedness proof enforce a nonzero credential randomizer. The relation for the well-formedness proof π is defined as follows:

$$\mathcal{R}_{\mathsf{Slss}} := \{(y, v, z) : \mathsf{E} = vX + z\mathsf{H} \ \wedge\ \mathsf{EV} \neq 0_{\mathbb{G}} \ \wedge\ \mathsf{EV} = v\mathsf{G} \ \wedge\ \mathsf{EZ} = z\mathsf{G} \ \wedge\ (C_s(y+s) = v\mathsf{G})_{s \in \mathbf{S}} \ \wedge\ Y = y\mathsf{G}\}.$$

It effectively ensures that $(\mathsf{E}, \mathsf{EZ})$ is a valid ElGamal encryption of $V = vX$ and that each C_s is well-formed with respect to that v. The condition $\mathsf{EV} \neq 0_{\mathbb{G}}$ forces $v \neq 0$ and, together with $C_s(y + s) = v\mathsf{G}$, also forces $y + s \neq 0$ for every $s \in \mathbf{S}$. This nonzero check is essential for unlinkability against malicious issuers: allowing $v = 0$ would let an issuer create a zero credential whose presentations are always $(0_{\mathbb{G}}, 0_{\mathbb{G}})$, immediately distinguishable from those with $v \neq 0$ when $X \neq 0_{\mathbb{G}}$.

$\underline{\mathsf{IssueDel}(\mathsf{isk}, \mathsf{ipar}, \mathbf{M} := \mathbf{S})}$

parse isk **as** (x, y)

require $\forall s \in \mathbf{S} : \; y + s \neq 0$

$v \xleftarrow{\$} \mathbb{Z}_p^*$ // Randomness for tag

$\mathsf{dk} := z \xleftarrow{\$} \mathbb{Z}_p$ // Decryption Key

// Encryption of the credential:

$\mathsf{EC} := (\mathsf{E}, \mathsf{EV}, \mathsf{EZ}, (C_s)_{s\in\mathbf{S}}) :=$

$\left(vx\mathsf{G} + z\mathsf{H}, v\mathsf{G}, z\mathsf{G}, (\frac{v\mathsf{G}}{y+s})_{s\in\mathbf{S}}\right)$

// Well-formedness proof for EC

$\mathbb{x} := (\mathsf{ipar}, \mathsf{EC}, \mathbf{S}), \mathbb{w} := (y, v, z)$

$\pi \leftarrow \mathsf{NIP}_{\mathsf{Slss}}.\mathsf{Prove}(\mathbb{x}, \mathbb{w})$

$\mathsf{Del}_1 := (\mathsf{EC}, \pi, \mathbf{S})$

return $\mathbf{Del} := \big((\mathsf{Del}_1), \mathsf{dk}\big)$

$\underline{\mathsf{Delegate}(\mathbf{Del}, \mathbf{M}' := \mathbf{S}')}$

parse $\mathbf{Del}$ **as** $\big((\mathsf{Del}_i)_{i\in[n]}, \mathsf{dk}\big)$

parse Del_n **as** $(\mathsf{EC}, \pi, \mathbf{S})$

require $n \geq 1 \wedge \mathbf{S}' \preceq \mathbf{S}$ // $\mathbf{S}'$: To be delegated

parse EC **as** $(\mathsf{E}, \mathsf{EV}, \mathsf{EZ}, (C_s)_{s\in\mathbf{S}})$

$(\mu, z') \xleftarrow{\$} \mathbb{Z}_p^* \times \mathbb{Z}_p$ // Randomness for delegation

$\mathsf{EC}' := \big(\mathsf{E}', \mathsf{EV}', \mathsf{EZ}', (C'_s)_{s\in\mathbf{S}'}\big)$

$:= \big(\mu\mathsf{E} + z'\mathsf{H}, \mu\mathsf{EV}, \mu\mathsf{EZ} + z'\mathsf{G}, \mu(C_s)_{s\in\mathbf{S}'}\big)$

$\mathbb{x} := (\mathsf{EC}, \mathsf{EC}', \mathbf{S}'), \mathbb{w} := (\mu, z')$

$\pi' \leftarrow \mathsf{NIP}_{\mathsf{SDel}}.\mathsf{Prove}(\mathbb{x}, \mathbb{w})$

$\mathsf{dk}' := \mu \cdot \mathsf{dk} + z'$ // New decryption key

$\mathbf{T}' := \big(\mathsf{Del}_1, \ldots, \mathsf{Del}_n, \mathsf{Del}' := (\mathsf{EC}', \pi', \mathbf{S}')\big)$

return $\mathbf{Del}' := (\mathbf{T}', \mathsf{dk}')$

$\underline{\mathsf{ObtainDel}(\mathsf{ipar}, \mathbf{Del})}$

parse $\mathbf{Del}$ **as** $\big((\mathsf{EC}_i, \pi_i, \mathbf{S}_i)_{i\in[n]}, \mathsf{dk}\big)$ and EC_n **as** $(\mathsf{E}, \mathsf{EV}, \mathsf{EZ}, (C_s)_{s\in\mathbf{S}_n})$

require $n \geq 1$

$\mathbb{x}_1 := (\mathsf{ipar}, \mathsf{EC}_1, \mathbf{S}_1), \forall i \in \{2, \ldots, n\} : \mathbb{x}_i := (\mathsf{EC}_{i-1}, \mathsf{EC}_i, \mathbf{S}_i)$

require $(\mathbf{S}_{i+1} \preceq \mathbf{S}_i)_{i\in[n-1]} \;\wedge\; \mathsf{EZ} = \mathsf{dkG} \wedge \mathsf{NIP}_{\mathsf{Slss}}.\mathsf{Verify}(\mathbb{x}_1, \pi_1) \wedge$

$\bigwedge_{i=2}^{n} \mathsf{NIP}_{\mathsf{SDel}}.\mathsf{Verify}(\mathbb{x}_i, \pi_i)$

return $(\mathsf{Cred} := (\mathsf{E} - \mathsf{dkH}, (C_s)_{s\in\mathbf{S}_n}), \mathbf{M} := \mathbf{S}_n)$

Fig. 4: Delegation Algorithms in Subset DKVAC.

Delegation. A credential holder, whether it received the credential directly from the issuer or through a previous delegation, can further delegate the credential by running the $\mathsf{Delegate}$ algorithm.

Consider first the case where the holder received the encrypted credential directly from the issuer. Suppose the holder has a credential on an attribute set $\mathbf{S}$ and wants to delegate a smaller attribute set $\mathbf{S}'$ such that $\mathbf{S}' \subseteq \mathbf{S}$. For this, the holder re-randomizes all components of the credential, and keeps only the credential components C_s corresponding to the remaining attributes $s \in \mathbf{S}'$. More concretely, it computes

$$\mathsf{EC}' = (\mathsf{E}', \mathsf{EV}', \mathsf{EZ}', (C'_s)_{s\in\mathbf{S}'}) = \big(\mu\mathsf{E} + z'\mathsf{H},\ \mu\mathsf{EV},\ \mu\mathsf{EZ} + z'\mathsf{G},\ (\mu C_s)_{s\in\mathbf{S}'}\big)$$

for random $(\mu, z') \xleftarrow{\$} \mathbb{Z}_p^* \times \mathbb{Z}_p$. Intuitively, if EC decrypts to $\big(vx\mathsf{G}, (v\mathsf{G}/(y+s))_{s\in\mathbf{S}}\big)$ under dk, then the new EC' decrypts to

$\big((\mu v)x\mathsf{G}, ((\mu v)\mathsf{G}/(y+s))_{s\in\mathbf{S}'}\big)$ under $\mathsf{dk}' = \mu\mathsf{dk} + z'$. As a result, the randomized credential is distributed like a newly issued credential on $\mathbf{S}'$.

For key-parameter consistency, the delegator also provides a proof π' showing that the new encrypted credential EC' was obtained by correctly re-randomizing the previous encrypted credential and restricting the attribute set to $\mathbf{S}'$ as described above. The corresponding relation is

$$\mathcal{R}_{\mathsf{SDel}} := \{(\mu, z') : \mathsf{E}' = \mu\mathsf{E} + z'\mathsf{H} \ \wedge \ \mathsf{EV}' \neq 0_{\mathbb{G}} \ \wedge \ \mathsf{EV}' = \mu\mathsf{EV} \\ \wedge \ \mathsf{EZ}' = \mu\mathsf{EZ} + z'\mathsf{G} \ \wedge \ (C'_s = \mu C_s)_{s\in\mathbf{S}'}\}\,.$$

The delegator outputs the new delegation chain, consisting of the issuer's original $\mathsf{Del} := (\mathsf{EC}, \pi, \mathbf{S})$, the newly re-randomized and attenuated $\mathsf{Del}' := (\mathsf{EC}', \pi', \mathbf{S}')$, and the decryption key dk' for EC'. Notably, the original encrypted credential EC is useless without its decryption key dk (which is not included in the delegator's output).

If the delegator itself received the credential from another user, it performs the procedure described above on the last node of the chain. Thus, each delegation step forms a re-randomized and attenuated copy $\mathsf{Del}' := (\mathsf{EC}', \pi', \mathbf{S}')$ of the last node Del, appends this copy to the ordered chain, and outputs $\mathbf{Del}' = ((\mathsf{Del}_1, \ldots, \mathsf{Del}_n, \mathsf{Del}'), \mathsf{dk}')$. The existing chain nodes are retained, but the old decryption key is not carried forward.

Obtaining a Delegated Credential. The delegatee runs $\mathsf{ObtainDel}$ to obtain the delegated credential, parsing $\mathbf{Del}$ into its ordered chain and final decryption key. It first verifies all proofs provided by the delegator for the encrypted credentials in the delegation chain, checking both key-parameter consistency and the well-formedness of each delegation step. This effectively ensures that all encrypted credentials in the chain are valid w.r.t. the issuer's keys. Then, the delegatee uses the decryption key to decrypt the last chain node. The resulting credential can then be used in the same way as a directly issued credential. For subsequent delegation, the delegatee retains the complete delegation object, including the ordered chain and its current decryption key.

4.3 Security of Subset DKVAC

Theorem 1. *If the* NIP*s are complete, then Subset DKVAC is correct.*

The proof is given in Appendix C.1.

Theorem 2. *If* $\mathsf{NIP}_{\mathsf{SIss}}$ *and* $\mathsf{NIP}_{\mathsf{SDel}}$ *are zero-knowledge (Appendix B.1), then Subset DKVAC is unforgeable in the generic group and random oracle models.*

Proof (Sketch). We replace honest issuance and delegation proofs by simulated ones. Repeated-proof zero knowledge bounds this change. In the generic group model, the simulators are generic algorithms, so their group outputs remain in the linear span of the group elements already present in the game.

It remains to analyze the adversary's view in the generic group model. Using standard Schwartz-Zippel arguments (Lemma 1), we view the unforgeability game as a symbolic experiment, where the random values chosen by the game's oracles are represented by formal polynomial variables. In this setting, the adversary is restricted to linearly combining polynomials (representing group elements) it receives from the experiment (including requested credentials and their encrypted chains). Careful analysis of these polynomials shows that the only presentations accepted by $\mathsf{VerifyShow}$ correspond to policies satisfied by a message whose credential the adversary has requested to receive. Such presentations do not constitute valid forgeries. □

The full proof is given in Appendix D.2.

Theorem 3. *If* $\mathsf{NIP}_{\mathsf{Slss}}$ *and* $\mathsf{NIP}_{\mathsf{SDel}}$ *are sound, then the Subset DKVAC is multi-show unlinkable.*

Proof (Sketch). By soundness of the proof system, the obtained credential is well-formed, and its presentation is $\mathsf{Show} = (\rho x\mathsf{G}, \rho w_{\mathbf{D}}\mathsf{G})$, where $w_{\mathbf{D}} = \sum_{d\in\mathbf{D}}(y + d)^{-1}$ and $\rho = \mu v$ is uniform in $\mathbb{Z}_p^*$. This distribution depends only on the fixed issuer parameters ipar and the common disclosure set $\mathbf{D}$, so fresh randomness makes all adaptive presentations independent of the challenge bit. □

The full proof is given in Appendix E.1.

5 Vector DKVAC Construction

In this section, we present our Vector DKVAC construction. For clarity, we first describe the underlying vector KVAC construction, and then explain how delegation can be incorporated into the construction. The construction is parameterized by a public attribute length $t = t(\lambda) \geq 1$, computable in polynomial time and polynomially bounded in λ. Setup records t in pp, which is an implicit input to the other algorithms unless displayed explicitly.

5.1 Vector KVAC Construction

Our second KVAC construction, presented in Figure 5, is based on an algebraic MAC related to the CMZ MAC family of [CMZ14] and the Pointcheval-Sanders signature family [PS16, BB18]. In what follows, we describe the algorithms of the construction.

Message and Message Space. A message consists of an ordered attribute vector together with a set of *malleable* index positions. For the attribute length t specified in pp, the issuer defines the message space as $\mathcal{M} := \mathbb{Z}_p^t \times 2^{[t]}$. We write a message as a pair $\mathbf{M} = (\mathbf{S}, \mathbf{L})$, where $\mathbf{S} = (s_1, \ldots, s_t)$ is the ordered attribute vector and $\mathbf{L} \subseteq [t]$ is the set of malleable indices. For these indices, the credential holder receives malleability keys that allow it to modify the corresponding attributes when delegating the credential.

$\underline{\mathsf{Setup}(1^\lambda)}$

Generate a cyclic group $\mathbb{G} = \langle \mathsf{G} \rangle$

of prime order $p > 2^\lambda$

return $\mathsf{pp} := (\mathbb{G}, \mathsf{G}, p, \mathsf{H} \xleftarrow{\$} \mathbb{G} \setminus \{0_\mathbb{G}\}, t)$

$\underline{\mathsf{KeyGen}(\mathsf{pp})}$

parse pp **as** $(\mathbb{G}, \mathsf{G}, p, \mathsf{H}, t)$

$(r, x, y_1, \ldots, y_t) \xleftarrow{\$} (\mathbb{Z}_p^*)^{t+2}$

$\mathsf{isk} := (r, x, (y_i)_{i\in[t]})$

$\mathsf{ipar} := \big(R, X', (Y_i')_{i\in[t]}\big)$

$\quad := \big(r\mathsf{H}, rx\mathsf{G}, (ry_i\mathsf{G})_{i\in[t]}\big)$

$\mathcal{M} := \mathbb{Z}_p^t \times 2^{[t]}$

return $(\mathsf{isk}, \mathsf{ipar}, \mathcal{M})$

$\underline{\mathsf{IssueCred}(\mathsf{isk}, \mathsf{ipar}, \mathbf{M} := (\mathbf{S}, \mathbf{L}) \in \mathcal{M})}$

parse isk **as** $(r, x, (y_i)_{i\in[t]})$

$v \xleftarrow{\$} \mathbb{Z}_p^*$ // Tag randomness

$$\tau := (V, C) := \left(v\mathsf{G}, (x + \sum_{i\in[t]} y_i s_i)v\mathsf{G}\right)$$

$\mathbf{MK} := (vy_i\mathsf{G})_{i\in\mathbf{L}}$ // malleability keys

// Well-formedness proof for τ and $\mathbf{MK}$

$\mathbb{x} := (\mathsf{ipar}, \tau, \mathbf{MK}, \mathbf{M})$

$\mathbb{w} := \big(r^{-1}, r, v\big)$

$\pi \leftarrow \mathsf{NIP}_{\mathsf{VCrIss}}.\mathsf{Prove}(\mathbb{x}, \mathbb{w})$

return $(\mathsf{Cred} := (\tau, \mathbf{MK}), \pi)$

$\underline{\mathsf{ObtainCred}(\mathsf{ipar}, \mathbf{M} := (\mathbf{S}, \mathbf{L}), \mathsf{Cred}, \pi)}$

parse Cred **as** $(\tau, \mathbf{MK})$

$\mathbb{x} := (\mathsf{ipar}, \tau, \mathbf{MK}, \mathbf{M})$

require $\mathsf{NIP}_{\mathsf{VCrIss}}.\mathsf{Verify}(\mathbb{x}, \pi)$

return Cred

$\underline{\mathsf{ShowCred}(\mathsf{ipar}, \mathbf{M} := (\mathbf{S}, \mathbf{L}), \phi, \mathsf{Cred})}$

require $\phi(\mathbf{M}) = 1$

parse $\mathbf{I}$ from ϕ. **Set** $\bar{\mathbf{I}} := [t] \setminus \mathbf{I}$.

// $\mathbf{I}$ / $\bar{\mathbf{I}}$: Indices of disclosed / hidden attributes

parse ipar **as** $\big(R, X', (Y_i')_{i\in[t]}\big)$

parse Cred **as** $(\tau := (V, C), \mathbf{MK})$

$\mu \xleftarrow{\$} \mathbb{Z}_p^*$

$\big(\mu', (\beta_i)_{i\in\bar{\mathbf{I}}}\big) \xleftarrow{\$} \mathbb{Z}_p^{|\bar{\mathbf{I}}|+1}$

$(V', C') := (\mu V, \mu C)$ // Randomizing τ

// Mask C' and commit to hidden attributes:

$W := C' + \mu'\mathsf{H}, (Q_i := s_i V' + \beta_i \mathsf{G})_{i\in\bar{\mathbf{I}}}$

$$P := \sum_{i\in\bar{\mathbf{I}}} \beta_i Y_i' - \mu' R$$

// Well-formedness proof for the commitments

$\mathbb{x} := \big(\mathsf{ipar}, V', P, (Q_i)_{i\in\bar{\mathbf{I}}}\big)$

$\mathbb{w} := \big(\mu', (\beta_i, s_i)_{i\in\bar{\mathbf{I}}}\big)$

$\pi \leftarrow \mathsf{NIP}_{\mathsf{VPres}}.\mathsf{Prove}(\mathbb{x}, \mathbb{w})$

return $\mathsf{Show} := \big(V', W, (Q_i)_{i\in\bar{\mathbf{I}}}, \pi\big)$

$\underline{\mathsf{VerifyShow}(\mathsf{isk}, \phi, \mathsf{Show})}$

parse isk **as** $(r, x, (y_i)_{i\in[t]})$

$\mathsf{ipar} := \big(R, X', (Y_i')_{i\in[t]}\big)$

$\quad := \big(r\mathsf{H}, rx\mathsf{G}, (ry_i\mathsf{G})_{i\in[t]}\big)$

parse $(\mathbf{I}, \mathbf{D} := (d_i)_{i\in\mathbf{I}})$ from ϕ

$\bar{\mathbf{I}} := [t] \setminus \mathbf{I}$

parse Show **as** $\big(V', W, (Q_i)_{i\in\bar{\mathbf{I}}}, \pi\big)$

$$P := r(xV' + \sum_{i\in\bar{\mathbf{I}}} y_i Q_i - W + \sum_{i\in\mathbf{I}} y_i d_i V')$$

$\mathbb{x} := \big(\mathsf{ipar}, V', P, (Q_i)_{i\in\bar{\mathbf{I}}}\big)$

return $\mathsf{NIP}_{\mathsf{VPres}}.\mathsf{Verify}(\mathbb{x}, \pi) \wedge (V' \neq 0_\mathbb{G})$

Fig. 5: Vector KVAC

Setup and Key Generation. The Setup algorithm generates the group and samples H uniformly from $\mathbb{G} \setminus \{0_\mathbb{G}\}$ as in the first construction, and includes the fixed attribute length t in pp. The KeyGen algorithm reads t from pp and samples $(r, x, y_1, \ldots, y_t) \xleftarrow{\$} (\mathbb{Z}_p^*)^{t+2}$ independently, where x is the MAC key and y_i is the secret weight of the i-th attribute. The issuer secret key is $\mathsf{isk} = (r, x, (y_i)_{i\in[t]})$. The issuer publishes $\mathsf{ipar} := \big(R, X', (Y_i')_{i\in[t]}\big) := \big(r\mathsf{H}, rx\mathsf{G}, (ry_i\mathsf{G})_{i\in[t]}\big)$, which

commits the issuer to isk, which we need for key-parameter consistency. Publishing the unblinded elements $x\mathsf{G}$ and $(y_i\mathsf{G})_{i\in[t]}$ would compromise unforgeability: anyone could form the credential tag $(\mathsf{G}, x\mathsf{G} + \sum_{i\in[t]} s_i(y_i\mathsf{G}))$ for an arbitrary attribute vector. For this reason, the secret r is used to blind $x\mathsf{G}$ in the public issuer parameters.

Credential Issuance. The issuer runs $\mathsf{IssueCred}$ to generate a credential on a message $\mathbf{M} := (\mathbf{S} = (s_1, \ldots, s_t), \mathbf{L})$ as follows. It samples fresh randomness $v \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $\tau := (V, C) := \left(v\mathsf{G},\ \left(x + \sum_{i\in[t]} y_i s_i\right) v\mathsf{G}\right)$. The issuer also computes, for each index $i \in \mathbf{L}$, a *malleability key*: $\mathbf{MK} := (vy_i\mathsf{G})_{i\in\mathbf{L}}$. Using $\mathbf{MK}$, a holder can update the attribute at position $i \in \mathbf{L}$ without knowing either v or y_i. For positions $i \notin \mathbf{L}$, such updates are not possible.

The issuer then creates a proof π showing the well-formedness of $\tau = (V, C)$ and the malleability keys $\mathbf{MK}$ with the following relation:

$$\mathcal{R}_{\mathsf{VCrIss}} := \{\left(r^{-1}, r, v\right) : V \neq 0_{\mathbb{G}} \ \wedge\ V = v\mathsf{G} \ \wedge\ rC = v(X' + \sum_{i\in[t]} Y_i' s_i) \\ \wedge\ r\mathbf{MK} = (vY_i')_{i\in\mathbf{L}} \ \wedge\ R = r\mathsf{H} \ \wedge\ r^{-1}R = \mathsf{H}\}\,.$$

In this relation, r, r^{-1} are separate variable names, so all equations are linear in the witness. Since $\mathsf{H} \neq 0_{\mathbb{G}}$, the equations $R = r\mathsf{H}$ and $r^{-1}R = \mathsf{H}$ fix a nonzero r and force the variable named r^{-1} to be its multiplicative inverse. The V and rC equations ensure well-formedness of the credential, the $r\mathbf{MK}$ equation ensures well-formedness of the malleability key (note that r on the left hand side cancels with r in $X' = rx\mathsf{G}$ and $Y_i' = ry_i\mathsf{G}$ on the right hand side). Finally, the issuer outputs $(\mathsf{Cred} := (\tau, \mathbf{MK}),\ \pi)$.

Obtaining a Credential. The user runs $\mathsf{ObtainCred}$ to verify the proof π with respect to ipar and the message $\mathbf{M}$. If the verification succeeds, the user stores Cred together with $\mathbf{M}$; otherwise, it outputs $\perp$.

Policy. Presentation implements partial disclosure. A policy specifies a set of indices $\mathbf{I}$ and target values $\mathbf{D} = (d_i)_{i\in\mathbf{I}}$. We denote such a policy by $\phi_{\mathbf{I},\mathbf{D}}$. The policy is satisfied by a message $\mathbf{M} = (\mathbf{S}, \mathbf{L})$ if $\phi_{\mathbf{I},\mathbf{D}}(\mathbf{S}, \mathbf{L}) = 1 \iff \forall i \in \mathbf{I} : s_i = d_i$, i.e., the message indeed corresponds to $\mathbf{D}$ at indices $\mathbf{I}$. The set $\mathbf{L}$ of malleable indices is ignored by presentation and only plays a role for restricting malleability for delegation.

Credential Presentation with Selective Disclosure. Our presentation and verification algorithms follow the $\mathsf{MAC}_{\mathsf{GGM}}$-based protocol of Chase, Meiklejohn, and Zaverucha [CMZ14], adapted to our r-blinded issuer parameters. To present a credential under a verifier-chosen policy $\phi_{\mathbf{I},\mathbf{D}}$, the holder runs $\mathsf{ShowCred}$. The algorithm first checks that $\phi_{\mathbf{I},\mathbf{D}}(\mathbf{M}) = 1$ and recovers the disclosed index set $\mathbf{I}$. The hidden index set is then defined as $\bar{\mathbf{I}} := [t] \setminus \mathbf{I}$.

Step 1: Randomization. The holder samples $\mu \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $(V', C') := (\mu V, \mu C)$. This randomization guarantees unlinkability across presentations.

Step 2: Commit to hidden attributes. For each hidden index $i \in \bar{\mathbf{I}}$, the holder samples fresh randomness $\beta_i \xleftarrow{\$} \mathbb{Z}_p$ and computes the Pedersen commitment $Q_i := s_i V' + \beta_i \mathsf{G}$. Using V' as the commitment base will later enable the verifier's algebraic check to combine the committed hidden values with the MAC verification equation.

Step 3: Mask the MAC. The holder independently samples $\mu' \xleftarrow{\$} \mathbb{Z}_p$ and computes $W := C' + \mu' \mathsf{H}$. This step is essential for privacy. From C' alone, a verifier knowing $(r, x, (y_i)_{i\in[t]})$ and the disclosed values could subtract the disclosed part and recover $\sum_{i\in\bar{\mathbf{I}}} y_i s_i \cdot V'$, which leaks information about the hidden attributes. The mask $\mu'\mathsf{H}$ prevents this by adding a hiding term that the verifier cannot remove.

Step 4: Bind and Prove. The holder computes $P := \sum_{i\in\bar{\mathbf{I}}} \beta_i Y'_i - \mu' R$, thereby binding the commitment randomness to the issuer parameters $Y'_i = r y_i \mathsf{G}$ and $R = r\mathsf{H}$. It then produces a non-interactive zero-knowledge proof π for the relation $\mathcal{R}_{\mathsf{VPres}}$, proving the well-formedness of P and the commitments $(Q_i)_{i\in\bar{\mathbf{I}}}$. The holder sends the presentation $\mathsf{Show} := (V', W, (Q_i)_{i\in\bar{\mathbf{I}}}, \pi)$. The relation is defined as:

$$\mathcal{R}_{\mathsf{VPres}} := \{(\mu', (\beta_i, s_i)_{i\in\bar{\mathbf{I}}}) : P = \sum_{i\in\bar{\mathbf{I}}} \beta_i Y'_i - \mu' R \wedge \bigwedge_{i\in\bar{\mathbf{I}}} (Q_i = s_i V' + \beta_i \mathsf{G})\}. \quad (1)$$

Verification. The verifier runs $\mathsf{VerifyShow}$ as follows. From $\mathsf{isk} = (r, x, (y_i)_{i\in[t]})$ and the public setup parameters, it reconstructs $\mathsf{ipar} = (r\mathsf{H}, rx\mathsf{G}, (ry_i\mathsf{G})_{i\in[t]})$ for the proof statement; these values may be computed once and cached with the verification key. It recovers the index set $\mathbf{I}$ and the target values $\mathbf{D} = (d_i)_{i\in\mathbf{I}}$ from the policy $\phi_{\mathbf{I},\mathbf{D}}$ and sets $\bar{\mathbf{I}} = [t] \setminus \mathbf{I}$. It then computes

$$P := r(xV' + \sum_{i\in\bar{\mathbf{I}}} y_i Q_i - W + \sum_{i\in\mathbf{I}} y_i d_i \cdot V').$$

For an honestly generated presentation, this value equals the value P computed by the holder. The verifier then checks the proof π and verifies that $V' \neq 0_{\mathbb{G}}$.

5.2 Adding Delegation to Vector KVAC

Figure 6 illustrates how delegation works in Vector DKVAC. The core idea is the same as in our Subset DKVAC construction: the delegator attenuates and re-randomizes the encrypted credential for which it holds a decryption key. It then sends the resulting encrypted credential to the delegatee, together with the updated decryption key, a proof of correct re-randomization, and all encrypted credentials and associated proofs received from previous delegators and from the issuer. We now describe the algorithms in detail. As in the subset construction, the delegation object consists of an ordered chain and its final decryption key:

$$\mathbf{Del} = (\mathbf{T}, \mathsf{dk}), \qquad \mathbf{T} = (\mathsf{Del}_1, \ldots, \mathsf{Del}_n), \qquad \mathsf{Del}_i = (\mathsf{EC}_i, \mathbf{MK}_i, \mathbf{M}_i, \pi_i).$$

$\underline{\mathsf{IssueDel}(\mathsf{isk}, \mathsf{ipar}, \mathbf{M} := (\mathbf{S}, \mathbf{L}) \in \mathcal{M})}$
parse isk **as** $(r, x, (y_i)_{i\in[t]})$
$v \xleftarrow{\$} \mathbb{Z}_p^*$ // Randomness for tag
$\mathsf{dk} := z \xleftarrow{\$} \mathbb{Z}_p$ // Decryption Key
// Encryption of the credential:
$\mathsf{EC} := (\mathsf{EV}, \mathsf{EZ}, C)$
$\quad := \big(v\mathsf{G} + z\mathsf{H}, z\mathsf{G}, (x + \sum_{i\in[t]} y_i s_i)v\mathsf{G}\big)$
// malleability keys for a set $\mathbf{L}$
$\mathbf{MK} := (vy_i\mathsf{G})_{i\in\mathbf{L}}$
// Well-formedness proof for EC and $\mathbf{MK}$
$\mathbb{x} := (\mathsf{ipar}, \mathsf{EC}, \mathbf{M}, \mathbf{MK})$
$\mathbb{w} := \big(r^{-1}, r, v, z\big)$
$\pi \leftarrow \mathsf{NIP}_{\mathsf{VIss}}.\mathsf{Prove}(\mathbb{x}, \mathbb{w})$
$\mathsf{Del}_1 := (\mathsf{EC}, \mathbf{MK}, \mathbf{M}, \pi)$
$\mathbf{Del} := \big((\mathsf{Del}_1), \mathsf{dk}\big)$
return $\mathbf{Del}$

$\underline{\mathsf{Delegate}(\mathbf{Del}, \mathbf{M}' := (\mathbf{S}', \mathbf{L}'))}$
parse $\mathbf{Del}$ **as** $\big((\mathsf{Del}_i)_{i\in[n]}, \mathsf{dk}\big)$
parse Del_n **as** $(\mathsf{EC}, \mathbf{MK}, \mathbf{M} = (\mathbf{S}, \mathbf{L}), \pi)$
require $n \geq 1 \wedge \mathbf{M}' \preceq \mathbf{M}$
parse EC **as** $(\mathsf{EV}, \mathsf{EZ}, C)$
$(\mu, z') \xleftarrow{\$} \mathbb{Z}_p^* \times \mathbb{Z}_p$ // Delegation randomness
$\mathsf{EC}' := (\mathsf{EV}', \mathsf{EZ}', C')$
$\quad := \big(\mu\mathsf{EV} + z'\mathsf{H}, \mu\mathsf{EZ} + z'\mathsf{G},$
$\qquad \mu(C + \sum_{i\in\mathbf{L}}(s_i' - s_i)\mathsf{MK}_i)\big)$
$\mathbf{MK}' := \mu(\mathsf{MK}_i)_{i\in\mathbf{L}'}$ // Retained keys
// Delegation proof from $(\mathsf{EC}, \mathbf{MK})$ to $(\mathsf{EC}', \mathbf{MK}')$
$\mathbb{x} := (\mathsf{EC}, \mathsf{EC}', \mathbf{MK}, \mathbf{MK}', \mathbf{M}, \mathbf{M}')$
$\mathbb{w} := (\mu, z')$
$\pi' \leftarrow \mathsf{NIP}_{\mathsf{VDel}}.\mathsf{Prove}(\mathbb{x}, \mathbb{w})$
$\mathsf{dk}' := \mu \cdot \mathsf{dk} + z'$ // New decryption key
$\mathsf{Del}' := (\mathsf{EC}', \mathbf{MK}', \mathbf{M}', \pi')$
$\mathbf{T}' := \big(\mathsf{Del}_1, \ldots, \mathsf{Del}_n, \mathsf{Del}'\big)$
return $\mathbf{Del}' := (\mathbf{T}', \mathsf{dk}')$

$\underline{\mathsf{ObtainDel}(\mathsf{ipar}, \mathbf{Del})}$
parse $\mathbf{Del}$ **as** $\big((\mathsf{EC}_i, \mathbf{MK}_i, \mathbf{M}_i, \pi_i)_{i\in[n]}, \mathsf{dk}\big)$ and EC_n **as** $(\mathsf{EV}, \mathsf{EZ}, C)$
require $n \geq 1$
$\mathbb{x}_1 := (\mathsf{ipar}, \mathsf{EC}_1, \mathbf{M}_1, \mathbf{MK}_1)$
$\forall i \in \{2, \ldots, n\} : \ \mathbb{x}_i := (\mathsf{EC}_{i-1}, \mathsf{EC}_i, \mathbf{MK}_{i-1}, \mathbf{MK}_i, \mathbf{M}_{i-1}, \mathbf{M}_i)$
require $(\mathbf{M}_{i+1} \preceq \mathbf{M}_i)_{i\in[n-1]} \ \wedge \ \mathsf{EV} - \mathsf{dkH} \neq 0_{\mathbb{G}} \ \wedge \ \mathsf{EZ} = \mathsf{dkG}$
$\qquad \wedge \mathsf{NIP}_{\mathsf{VIss}}.\mathsf{Verify}(\mathbb{x}_1, \pi_1) \wedge \bigwedge_{i=2}^{n} \mathsf{NIP}_{\mathsf{VDel}}.\mathsf{Verify}(\mathbb{x}_i, \pi_i)$
return $\big(\mathsf{Cred} := \big((\mathsf{EV} - \mathsf{dkH}, C), \mathbf{MK}_n\big), \mathbf{M}_n\big)$

Fig. 6: Delegation Algorithms in Vector DKVAC

Delegation Relation. For this construction, the delegation relation is defined as $(\mathbf{S}', \mathbf{L}') \preceq (\mathbf{S}, \mathbf{L}) \iff (\mathbf{L}' \subseteq \mathbf{L}) \wedge (\forall i \in \bar{\mathbf{L}} : s_i' = s_i)$, where $\bar{\mathbf{L}} = [t] \setminus \mathbf{L}$. This relation is reflexive and transitive: if $\mathbf{M}'' \preceq \mathbf{M}' \preceq \mathbf{M}$, then $\mathbf{L}'' \subseteq \mathbf{L}' \subseteq \mathbf{L}$ and $s_i'' = s_i' = s_i$ for every $i \notin \mathbf{L}$. This means that a delegator may modify attributes at positions that were marked as malleable ($i \in \mathbf{L}$), and restrict the set of malleable indices for the delegatee. For example, the delegator may encode a new expiration date and make it unmodifiable for the delegatee.

Delegation Issuance. The issuer runs $\mathsf{IssueDel}$ to issue a delegatable credential. Conceptually, the issuer computes a credential $\tau = (V, C) = \Big(v\mathsf{G}, \Big(x + \sum_{i\in[t]} y_i s_i\Big) v\mathsf{G}\Big)$ and $\mathbf{MK} = (\mathsf{MK}_i)_{i\in\mathbf{L}} = (vy_i\mathsf{G})_{i\in\mathbf{L}}$ on the mes-

sage $\mathbf{M} = (\mathbf{S}, \mathbf{L})$. Then the issuer ElGamal-encrypts $V = v\mathsf{G}$, as $(\mathsf{EV}, \mathsf{EZ}) := (v\mathsf{G} + z\mathsf{H}, z\mathsf{G})$. The encrypted credential is unusable without the decryption key $\mathsf{dk} := z$, which is needed to recover V. The issuer outputs the single delegation object $\mathbf{Del} = ((\mathsf{Del}_1), \mathsf{dk})$, where $\mathsf{Del}_1 = (\mathsf{EC}, \mathbf{MK}, \mathbf{M}, \pi)$ contains the encrypted credential, malleability keys, message, and well-formedness proof. The corresponding proof relation for the well-formedness proof π is

$$\begin{aligned}\mathcal{R}_{\mathsf{VIss}} := \{\big(r^{-1}, r, v, z\big) : \mathsf{EV} = v\mathsf{G} + z\mathsf{H} \ \wedge \ \mathsf{EZ} = z\mathsf{G} \\ \wedge \ rC = v(X' + \sum_{i\in[t]} Y'_i s_i) \ \wedge \ r\mathbf{MK} = (vY'_i)_{i\in\mathbf{L}} \\ \wedge \ R = r\mathsf{H} \ \wedge \ r^{-1}R = \mathsf{H}\}\,.\end{aligned}$$

This relation is analogous to $\mathcal{R}_{\mathsf{VCrIss}}$ (from the non-delegatable issuance above), except that it accounts for the encrypted V.

Delegation. To delegate from $\mathbf{M} = (\mathbf{S}, \mathbf{L})$ to $\mathbf{M}' = (\mathbf{S}', \mathbf{L}')$ such that $\mathbf{M}' \preceq \mathbf{M}$, the holder parses $\mathbf{Del}$ into its chain and current key, then reads the last chain node $\mathsf{Del}_n = (\mathsf{EC}, \mathbf{MK}, \mathbf{M}, \pi)$. It samples fresh randomness $(\mu, z') \xleftarrow{\$} \mathbb{Z}_p^* \times \mathbb{Z}_p$ and computes:

$$\begin{aligned}\mathsf{EC}' &= \big(\mu\mathsf{EV} + z'\mathsf{H},\ \mu\mathsf{EZ} + z'\mathsf{G},\ \mu(C + \textstyle\sum_{i\in\mathbf{L}}(s'_i - s_i)\mathsf{MK}_i)\big), \\ \mathbf{MK}' &= \mu(\mathsf{MK}_i)_{i\in\mathbf{L}'}\,.\end{aligned}$$

The term $(s'_i - s_i)\mathsf{MK}_i = (s'_i - s_i)vy_i\mathsf{G}$ replaces the contribution $s_i \cdot vy_i\mathsf{G}$ of attribute $i \in \mathbf{L}$ in C with $s'_i \cdot vy_i\mathsf{G}$. Thus, the updated component C' encodes the modified attribute value s'_i. The malleability keys for indices in $\mathbf{L} \setminus \mathbf{L}'$ are not retained in $\mathbf{MK}'$, which restricts the malleable set to $\mathbf{L}'$. The decryption key is updated to $\mathsf{dk}' := \mu \cdot \mathsf{dk} + z'$. The result is an encrypted credential EC' that is distributed like a freshly issued one on message $\mathbf{M}'$ with decryption key dk'.

The delegator also produces a proof π' for the relation $\mathcal{R}_{\mathsf{VDel}}$, showing that EC' and $\mathbf{MK}'$ are obtained by a valid re-randomization and restriction of the previous encrypted credential:

$$\begin{aligned}\mathcal{R}_{\mathsf{VDel}} := \{(\mu, z') : \mathsf{EV}' = \mu\mathsf{EV} + z'\mathsf{H} \ \wedge \ \mathsf{EZ}' = \mu\mathsf{EZ} + z'\mathsf{G} \\ \wedge \ C' = \mu(C + \sum_{i\in\mathbf{L}}(s'_i - s_i)\mathsf{MK}_i) \ \wedge \ \mathbf{MK}' = (\mu\mathbf{MK}_i)_{i\in\mathbf{L}'}\}\,.\end{aligned}$$

The delegator appends the node $\mathsf{Del}' := (\mathsf{EC}', \mathbf{MK}', \mathbf{M}', \pi')$ to the ordered chain and outputs $\mathbf{Del}' = ((\mathsf{Del}_1, \ldots, \mathsf{Del}_n, \mathsf{Del}'), \mathsf{dk}')$. This single object contains the updated key and all chain nodes, but not the preceding decryption key.

Obtaining a Delegated Credential. The $\mathsf{ObtainDel}$ algorithm parses the delegation object into its ordered chain and final decryption key. It first verifies the entire delegation chain. The first proof is the issuance proof for $\mathcal{R}_{\mathsf{VIss}}$, attached to the initial encrypted credential. Each subsequent proof is a delegation proof for

$\mathcal{R}_{\mathsf{VDel}}$, showing that the next encrypted credential and malleability keys follow the specified algebraic update. Separately, ObtainDel checks $\mathbf{M}_{j+1} \preceq \mathbf{M}_j$ at each link. These public checks enforce $\mathbf{L}_{j+1} \subseteq \mathbf{L}_j$ and $s_{j+1,i} = s_{j,i}$ for every $i \notin \mathbf{L}_j$; the proof relation alone does not enforce these restrictions. Overall, this guarantees that all credentials in the delegation chain are valid w.r.t. the issuer's parameters ipar. ObtainDel uses the final decryption key dk to decrypt the (V component of) the last credential in the chain. Since the decrypted Cred has the same form as a directly issued credential, it can be presented using ShowCred.

5.3 Security of Vector DKVAC

Theorem 4. *If the* NIP*s are complete, then Vector DKVAC is correct.*

The proof is given in Appendix C.2.

Theorem 5. *If* $\mathsf{NIP}_{\mathsf{VIss}}$ *and* $\mathsf{NIP}_{\mathsf{VDel}}$ *are zero-knowledge and* $\mathsf{NIP}_{\mathsf{VPres}}$ *is instantiated by the Fiat–Shamir generalized-Schnorr construction, both as in Appendix B.1, then Vector DKVAC is unforgeable in the generic group and random oracle models.*

Proof (Sketch). We replace honest issuance and delegation proofs by simulated ones. Repeated-proof zero knowledge bounds this change. In the generic group model, the simulators are generic algorithms, so their group outputs remain in the linear span of the group elements already present in the game.

Similarly to Theorem 2, we apply generic group model analysis and switch to a symbolic experiment where the adversary is restricted to linearly combining group elements it has seen, and group elements are represented by polynomials. We show that the generalized Schnorr protocol guarantees that the equations in the proven relation must hold over the formal polynomials (). Careful analysis of the GGM polynomials shows that the only presentations accepted by VerifyShow correspond to policies that can be trivially satisfied given some credential the adversary has requested to receive. Such a presentation does not constitute a valid forgery. □

The full proof is given in Appendix D.3.

Theorem 6. *If* $\mathsf{NIP}_{\mathsf{VIss}}$ *and* $\mathsf{NIP}_{\mathsf{VDel}}$ *are sound, and* $\mathsf{NIP}_{\mathsf{VPres}}$ *is zero-knowledge, then the Vector DKVAC is multi-show unlinkable.*

Proof (Sketch). We first simulate the presentation proofs using zero knowledge. Using proof soundness, we abort on an accepted false issuance or delegation statement. The obtained credentials are then well formed, V' is uniform over the nonzero group elements, and the independent uniform masks $\mu', \beta_i \in \mathbb{Z}_p$ make $W, (Q_i)_{i \in \bar{\mathbf{I}}}$ independent and uniform over $\mathbb{G}$ conditioned on V'. These values determine the complete presentation statement for fixed issuer parameters and policy, so the simulated view is independent of the challenge bit even over adaptive queries. □

The full proof is given in Appendix E.2.

Table 1: Reported runtimes and sizes (1 KiB = 1024 bytes). Here n is the attribute count and d the number of delegation steps after issuance. Runtime columns name the measured algorithms. Size columns give the implementation-reported sizes for credentials (Cred), presentations (Show), the initial delegation object ($\mathbf{Del}^{(0)}$, output of IssueDel), and the delegation object after d steps ($\mathbf{Del}^{(d)}$). Presentations disclose half the attributes; delegation retains all attributes. Sizes count 32-byte group elements and scalars only; messages (attribute values and, for Vector, index sets) are excluded. For Vector DKVAC, the Cred size also excludes the malleability keys **MK**, which are needed only for delegation, not for presentation.

			Runtime (ms)					**Size (KiB)**			
Scheme	n	d	IssueDel	ShowCred	VerifyShow	Delegate	ObtainDel	Cred	Show	$\mathbf{Del}^{(0)}$	$\mathbf{Del}^{(d)}$
Subset DKVAC	4	3	0.49	0.06	0.08	0.68	2.55	0.16	0.06	0.59	2.09
	32	3	2.75	0.08	0.23	2.87	11.53	1.03	0.06	2.34	9.09
Vector DKVAC	4	3	0.84	0.62	0.58	0.68	2.62	0.06	0.38	0.66	2.16
	32	3	4.54	3.65	4.04	2.91	11.69	0.06	2.13	2.41	9.16

6 Performance Evaluation

We open-source our implementation[6]. We used Rust and `curve25519-dalek` (Ristretto255). The benchmarks in Table 1 use credentials with $n \in \{4, 32\}$ attributes and delegation depth $d = 3$ on a MacBook Pro M1 Max. Here n counts attributes ($n = t$ for Vector), and d counts delegation steps after issuance, so depth 3 corresponds to four chain nodes. Presentations disclose half the attributes and each delegation retains all attributes. For Vector, all indices are malleable ($\mathbf{L} = [t]$), and delegation keeps $\mathbf{L}' = \mathbf{L}$.

The measurements support the practicality of both schemes in these configurations: presentation and verification take 0.06–4.04 ms, while ObtainDel takes 2.55–11.69 ms. The Subset construction prioritizes efficient presentations and verification, with constant-size Show. At 32 attributes, its presentation and verification take 0.08 and 0.23 ms, respectively, compared with 3.65 and 4.04 ms for Vector. Delegation and ObtainDel have similar measured runtimes in both schemes. Vector supports more expressive delegation. Obtaining a delegated credential verifies the entire chain; subsequent presentations and their verification do not process the chain.

Acknowledgments

This work was supported by SolidLab Vlaanderen (Flemish Government, EWI), the VLAIO–ICON project SCAMPER with project number HBC.2024.0622, and the Flemish Government's Cybersecurity Research Program under grant number VOEWICS02.

[6] https://github.com/emad7105/dkvac

Bibliography

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Berlin, Heidelberg, May 2004.

[BB18] Johannes Blömer and Jan Bobolz. Delegatable attribute-based anonymous credentials from dynamically malleable signatures. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 2018*, volume 10892 of *LNCS*, pages 221–239. Springer, Cham, July 2018.

[BCC+09] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, Berlin, Heidelberg, August 2009.

[BPE+14] Arnar Birgisson, Joe Gibbs Politz, Ulfar Erlingsson, Ankur Taly, Michael Vrable, and Mark Lentczner. Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014*, 2014.

[CDD17] Jan Camenisch, Manu Drijvers, and Maria Dubovitskaya. Practical UC-secure delegatable credentials with attributes and their application to blockchain. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 683–699. ACM Press, October / November 2017.

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Berlin, Heidelberg, May 2001.

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Berlin, Heidelberg, August 2006.

[CL19] Elizabeth C. Crites and Anna Lysyanskaya. Delegatable anonymous credentials from mercurial signatures. In Mitsuru Matsui, editor, *CT-RSA 2019*, volume 11405 of *LNCS*, pages 535–555. Springer, Cham, March 2019.

[CL21] Elizabeth C. Crites and Anna Lysyanskaya. Mercurial signatures for variable-length messages. *PoPETs*, 2021(4):441–463, October 2021.

[CMZ14] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In Gail-Joon

Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 1205–1216. ACM Press, November 2014.

[CPZ20] Melissa Chase, Trevor Perrin, and Greg Zaverucha. The Signal private group system and anonymous credentials supporting efficient verifiable encryption. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1445–1459. ACM Press, November 2020.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Burton S. Kaliski, Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 410–424. Springer, Berlin, Heidelberg, August 1997.

[Fuc11] Georg Fuchsbauer. Commuting signatures and verifiable encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 224–245. Springer, Berlin, Heidelberg, May 2011.

[MBS+25] Omid Mirzamohammadi, Jan Bobolz, Mahdi Sedaghat, Emad Heydari Beni, Aysajan Abidin, Dave Singelée, and Bart Preneel. Keyed-verification anonymous credentials with highly efficient partial disclosure. *CiC*, 2(3):15, 2025.

[MSBM23] Omid Mir, Daniel Slamanig, Balthazar Bauer, and René Mayrhofer. Practical delegatable anonymous credentials from equivalence class signatures. *PoPETs*, 2023(3):488–513, July 2023.

[Orr25] Michele Orrù. Revisiting keyed-verification anonymous credentials. In Chun-Ying Huang, Jyh-Cheng Chen, Shiuh-Pyng Shieh, David Lie, and Véronique Cortier, editors, *ACM CCS 2025*, pages 1188–1199. ACM Press, October 2025.

[PM23] Colin Putman and Keith M. Martin. Selective delegation of attributes in mercurial signature credentials. In Elizabeth A. Quaglia, editor, *19th IMA International Conference on Cryptography and Coding*, volume 14421 of *LNCS*, pages 181–196. Springer, Cham, December 2023.

[PS16] David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126. Springer, Cham, February / March 2016.

[TG23] Lindsey Tulloch and Ian Goldberg. Lox: Protecting the social graph in bridge distribution. *PoPETs*, 2023(1):494–509, January 2023.

[XYH+25] Wenyi Xue, Yang Yang, Minming Huang, Yingjiu Li, Hwee Hwa Pang, and Robert H. Deng. DkvSSO: Delegatable keyed-verification credentials for efficient anonymous single sign-on. *IEEE Transactions on Information Forensics and Security*, 20:4196–4211, 2025.

A Related Work

Anonymous credentials were introduced by Chaum [Cha82] and later formalized and refined by Camenisch and Lysyanskaya [CL01]. Since then, a large

body of work has explored privacy-preserving authentication, selective disclosure, unlinkability, and efficient credential presentation. Prominent examples include Idemix,[7] U-Prove,[8] CL-signature-based credentials [CL01], and numerous pairing-based anonymous credential systems.

Delegation has been studied extensively in the context of publicly verifiable anonymous credentials. We have mentioned important known approaches above. In group-based constructions, all known delegatable credential systems require pairings. Public verifiability is used extensively in those systems.

Existing approaches can roughly be divided into several categories. Early constructions relied on malleable proofs and delegatable proof systems [BCC+09, Fuc11]. Later works explored delegation using structure-preserving signatures, mercurial signatures, equivalence-class signatures, and related primitives [CL19, CL21, PM23, MSBM23]. Blömer and Bobolz [BB18] introduced a delegatable credential construction based on dynamically malleable signatures. Camenisch, Drijvers, and Dubovitskaya [CDD17] presented practical UC-secure delegatable credentials with attributes. These systems provide powerful delegation functionality but typically rely on public verification and often employ bilinear pairings.

Keyed-verification anonymous credentials were introduced by Chase, Meiklejohn, and Zaverucha [CMZ14] and subsequently attracted significant interest due to their efficiency advantages over publicly verifiable credentials. Subsequent works improved the efficiency and functionality of KVAC systems, particularly in the areas of selective disclosure and presentation efficiency. Most existing KVAC constructions, however, focus on issuance, selective disclosure, unlinkability, and presentation efficiency rather than delegation.

Delegation of credentials has received no attention in the KVAC setting. Works that broadly treat delegation in a KVAC setting only support delegation of verification tokens [XYH+25] (instead of credentials), and delegation requires the issuer's key.

Our work is also related to Macaroons [BPE+14], which introduced the notion of attenuation through chained caveats. While highly influential, Macaroons do not provide anonymity, unlinkability, or privacy-preserving presentation and therefore do not satisfy the goals typically associated with anonymous credential systems.

B Additional Preliminaries

This appendix gives the formal proof-system definitions used in our constructions and security analyses. We use the notation of Section 2 and work in the random oracle model. In particular, we specify completeness, ordinary soundness, and zero knowledge for repeated, adaptively chosen proof requests.

[7] https://github.com/IBM/idemix
[8] https://www.microsoft.com/en-us/research/project/u-prove/

B.1 Non-interactive proof systems

Let pp denote the public parameters already supplied by the surrounding DKVAC experiment. The proof system generates no additional parameters and has no separate setup algorithm. Let $\mathcal{R}_{\mathsf{pp}}$ be a polynomial-time decidable relation between statements and polynomial-length witnesses, and define

$$\mathcal{L}(\mathcal{R}_{\mathsf{pp}}) = \{\mathbb{x} : \exists \mathbb{w},\, (\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{pp}}\}.$$

A proof system $\mathsf{NIP} = (\mathsf{Prove}, \mathsf{Verify})$ consists of a randomized polynomial-time prover Prove and a polynomial-time verifier Verify, with interfaces

$$\pi \leftarrow \mathsf{Prove}^{\mathcal{O}_{\mathrm{RO}}}(\mathbb{x}, \mathbb{w}), \qquad \mathsf{Verify}^{\mathcal{O}_{\mathrm{RO}}}(\mathbb{x}, \pi) \in \{0, 1\}.$$

The security parameter and the existing public parameters pp are available to these algorithms but suppressed in their inputs. Issuer parameters and all other varying public inputs are part of the statement. Statements and proof requests may be chosen adaptively after earlier oracle answers.

Random oracle. For each proof type, $\mathcal{O}_{\mathrm{RO}}$ maps encoded inputs to $\mathbb{Z}_p$. Each new input receives an independent uniform output, and repeated inputs receive the same output. Different proof types use distinct tags. All encodings are unambiguous and include the parameters, indices, and group elements that define the statement. Real executions use this oracle. A zero-knowledge simulation instead uses the simulator's oracle interface defined below.

Completeness. For every valid pp and $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathsf{pp}}$, we require

$$\Pr\Big[\mathsf{Verify}^{\mathcal{O}_{\mathrm{RO}}}\Big(\mathbb{x}, \mathsf{Prove}^{\mathcal{O}_{\mathrm{RO}}}(\mathbb{x}, \mathbb{w})\Big) = 1\Big] = 1.$$

The probability is over the prover's randomness and the random oracle. The requirement also holds after earlier oracle queries, with all previously assigned oracle outputs preserved.

Ordinary soundness for repeated proofs. For every PPT adversary $\mathcal{A}$, let $(\mathbb{x}_i, \pi_i)_{i \in [q]} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{RO}}}(\mathsf{pp})$ be its output list of polynomially many statement-proof pairs. The pairs may be chosen adaptively, including after running the public verifier with the same random oracle. We require

$$\mathsf{Adv}^{\mathrm{Sound}}_{\mathsf{NIP},\mathcal{A}}(\lambda) := \Pr\left[\exists i \in [q] : \begin{array}{l} \mathsf{Verify}^{\mathcal{O}_{\mathrm{RO}}}(\mathbb{x}_i, \pi_i) = 1 \\ \wedge\ \ \mathbb{x}_i \notin \mathcal{L}(\mathcal{R}_{\mathsf{pp}}) \end{array}\right] \le \mathsf{negl}(\lambda).$$

All verifier evaluations use the same random oracle as the adversary. The probability is over the adversary's and oracle's randomness and pp. Thus, the advantage bounds the probability that any proof in the list is accepted for a false statement. This formulation is equivalent to single-proof soundness up to a polynomial loss, by selecting a uniformly random pair from the list. The advantage already accounts for all proof attempts, so no additional proof-count factor is needed when it is used below.

Zero knowledge for repeated proofs. There must exist a stateful PPT simulator Sim, given the existing public parameters pp, with interfaces $\mathsf{Sim}.\mathcal{O}_{\mathrm{RO}}$ and $\mathsf{Sim.Prove}$. For every PPT adversary $\mathcal{A}$, require

$$\mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIP},\mathsf{Sim},\mathcal{A}}(\lambda) := \Big| \Pr[\mathcal{A}^{\mathcal{O}_{\mathrm{RO}},\mathcal{O}_{\mathrm{Real}}}(\mathsf{pp}) = 1] - \Pr[\mathcal{A}^{\mathsf{Sim}.\mathcal{O}_{\mathrm{RO}},\mathcal{O}_{\mathrm{Sim}}}(\mathsf{pp}) = 1] \Big| \leq \mathsf{negl}(\lambda).$$

Both proof oracles take $(\mathbb{x}, \mathbb{w})$ and return $\perp$ if $(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}_{\mathsf{pp}}$. Otherwise, $\mathcal{O}_{\mathrm{Real}}$ returns $\mathsf{Prove}^{\mathcal{O}_{\mathrm{RO}}}(\mathbb{x}, \mathbb{w})$, whereas $\mathcal{O}_{\mathrm{Sim}}$ returns $\mathsf{Sim.Prove}(\mathbb{x})$. Only the proof oracle's validity check uses the supplied witness in the simulated experiment and Sim receives no witness.

The adversary may make polynomially many proof and random-oracle queries. The simulator retains state across calls and may program previously unanswered oracle inputs. The probabilities are over all algorithm and oracle randomness and pp (both experiments use the same distribution of pp).

Generalized Schnorr and Fiat-Shamir. We instantiate the proof relations in our constructions by generalized Schnorr followed by Fiat-Shamir. Write their group equations as $\mathbf{X} = \Phi\mathbb{w}$, where $\mathbf{X} \in \mathbb{G}^m$, $\Phi \in \mathbb{G}^{m\times n}$ and $\mathbb{w} \in \mathbb{Z}_p^n$ and additional public conditions are checked directly. The full statement $\mathbb{x}$ includes these group elements, the public parameters and all relevant indices and messages. The prover samples $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_p^n$ and computes

$$\mathbf{A} = \Phi\mathbf{a}, \qquad c = \mathcal{O}_{\mathrm{RO}}(\mathsf{tag}, \mathbb{x}, \mathbf{A}), \qquad \mathbf{u} = \mathbf{a} + c\mathbb{w}.$$

It returns $\pi = (\mathbf{A}, \mathbf{u})$. The verifier rejects malformed inputs or failed public conditions, recomputes c, and accepts exactly when $\Phi\mathbf{u} = \mathbf{A} + c\mathbf{X}$. The tag identifies the proof type. All group operations are generic and the construction uses the existing parameters and requires no additional setup. For $\mathsf{NIP}_{\mathsf{VPres}}$, we include the challenge explicitly, writing $\pi = (c, \mathbf{A}, \mathbf{u})$; the verifier additionally checks that the supplied c equals the recomputed hash. Its simulator includes c in the same way.

Generic simulation. Each proof type has an independent, stateful random-oracle table. Its simulator answers new queries uniformly, preserves previous answers, and reads or programs only its own tagged domain. To simulate a proof with $\Phi \neq 0$, it samples $c \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^n$, and computes

$$\mathbf{A} = \Phi\mathbf{u} - c\mathbf{X}.$$

If $(\mathsf{tag}, \mathbb{x}, \mathbf{A})$ is already in its table, it returns $\perp$. In the other case, it stores the answer c at this input and returns $(\mathbf{A}, \mathbf{u})$. If $\Phi = 0$, it returns $\perp$ unless $\mathbf{X} = 0$. In the latter case, it uses $\mathbf{A} = 0$, samples $\mathbf{u}$ uniformly, and queries its oracle normally without programming. Public conditions are checked as by the verifier.

For a valid statement with $\Phi \neq 0$, the announcement is uniform in the image of the map $\mathbf{a} \mapsto \Phi\mathbf{a}$, which has at least p elements. A table with at most

ℓ previous entries is therefore hit with probability at most ℓ/p. Until such a collision, the simulated proof and oracle transcript can be coupled identically to the real execution, using $\mathbf{a} = \mathbf{u} - c\mathbb{w}$. A union bound gives negligible failure for polynomially many adaptive proof requests and table entries. This failure is already included in the repeated-proof zero-knowledge advantage above. The simulator uses no witness and only known-scalar linear combinations of statement elements, so its group outputs remain in their linear span. Unsimulated proof domains retain independent random oracles.

C Correctness Proofs

C.1 Proof of Theorem 1

Proof. Fix pp, $(\mathsf{isk}, \mathsf{ipar}, \mathcal{M})$ with $\mathsf{isk} = (x, y)$, messages $\mathbf{S}_1 \supseteq \cdots \supseteq \mathbf{S}_n$ in $\mathcal{M}$, and the honestly generated $\mathbf{Del}_1, \ldots, \mathbf{Del}_n$ as in Definition 1. Since $\mathbf{S}_1 \in \mathcal{M}$, the issuance check $y + s \neq 0$ holds for every $s \in \mathbf{S}_1$; subset delegation preserves this condition. We first show by induction on n that the last node of $\mathbf{Del}_n$ has the form

$$\mathsf{EC}_n = \left(vx\mathsf{G} + z\mathsf{H},\ v\mathsf{G},\ z\mathsf{G},\ \left(\frac{v\mathsf{G}}{y+s}\right)_{s\in\mathbf{S}_n} \right)$$

for some $v \in \mathbb{Z}_p^*$, where z is the decryption key output with $\mathbf{Del}_n$, and that all proofs in the chain are honestly generated for true statements. For $n = 1$, this holds by the definition of $\mathsf{IssueDel}$, and (y, v, z) is a witness for $\mathcal{R}_{\mathsf{Slss}}$; in particular, $\mathsf{EV} = v\mathsf{G} \neq 0_{\mathbb{G}}$. For the step, $\mathsf{Delegate}$ is run on $\mathbf{S}_{n+1} \subseteq \mathbf{S}_n$, so its check passes, and it samples $(\eta, z') \in \mathbb{Z}_p^* \times \mathbb{Z}_p$, where η denotes the delegation randomizer called μ in the construction. It outputs

$$\mathsf{EC}' = \left((\eta v)x\mathsf{G} + (\eta z + z')\mathsf{H},\ (\eta v)\mathsf{G},\ (\eta z + z')\mathsf{G},\ \left(\frac{(\eta v)\mathsf{G}}{y+s}\right)_{s\in\mathbf{S}_{n+1}} \right)$$

and $\mathsf{dk}' = \eta z + z'$. This is of the required form with $\eta v \in \mathbb{Z}_p^*$, and (η, z') is a witness for $\mathcal{R}_{\mathsf{SDel}}$, since $\mathsf{EV}' = \eta v\mathsf{G} \neq 0_{\mathbb{G}}$.

Now consider $\mathsf{ObtainDel}(\mathsf{ipar}, \mathbf{Del}_n)$. The chain satisfies $\mathbf{S}_{i+1} \subseteq \mathbf{S}_i$, the check $\mathsf{EZ} = \mathsf{dkG}$ holds, and by completeness every proof in the chain verifies. Hence $\mathsf{ObtainDel}$ outputs $\mathbf{M} = \mathbf{S}_n$ and $\mathsf{Cred} = (V, (C_s)_{s\in\mathbf{S}_n})$ with $V = \mathsf{E} - z\mathsf{H} = vx\mathsf{G}$ and $C_s = \frac{v\mathsf{G}}{y+s}$. Since $\phi_{\mathbf{D}}(\mathbf{S}_n) = 1$, we have $\emptyset \neq \mathbf{D} \subseteq \mathbf{S}_n$, so $\mathsf{ShowCred}$ outputs $V' = \mu vx\mathsf{G}$ and $C' = \mu v \sum_{d\in\mathbf{D}} \frac{\mathsf{G}}{y+d}$ using fresh presentation randomness $\mu \in \mathbb{Z}_p^*$. Since $\mathbf{S}_n \in \mathcal{M}$, we have $y + d \neq 0$ for all $d \in \mathbf{D}$, so $\mathsf{VerifyShow}$ does not reject early. Moreover, $V' \neq 0_{\mathbb{G}}$ because $\mu, v, x \in \mathbb{Z}_p^*$, and

$$\sum_{d\in\mathbf{D}} \frac{1}{y+d} V' = \mu vx \sum_{d\in\mathbf{D}} \frac{\mathsf{G}}{y+d} = xC'.$$

Thus $\mathsf{VerifyShow}$ outputs 1 with probability 1. □

C.2 Proof of Theorem 4

Proof. Fix pp, $(\mathsf{isk}, \mathsf{ipar}, \mathcal{M})$ with $\mathsf{isk} = (r, x, (y_i)_{i\in[t]})$, messages $\mathbf{M}_1 \succeq \cdots \succeq \mathbf{M}_n$ with $\mathbf{M}_k = (\mathbf{S}_k, \mathbf{L}_k)$ and $\mathbf{S}_k = (s_{k,1}, \ldots, s_{k,t})$, and the honestly generated $\mathbf{Del}_1, \ldots, \mathbf{Del}_n$ as in Definition 1. For $\mathbf{S} = (s_1, \ldots, s_t)$, write $c(\mathbf{S}) := x + \sum_{i\in[t]} y_i s_i$. We first show by induction on n that the last node of $\mathbf{Del}_n$ has the form

$$\mathsf{EC}_n = (v\mathsf{G} + z\mathsf{H},\ z\mathsf{G},\ c(\mathbf{S}_n)v\mathsf{G})\,, \qquad \mathbf{MK}_n = (vy_i\mathsf{G})_{i\in\mathbf{L}_n}$$

for some $v \in \mathbb{Z}_p^*$, where z is the decryption key output with $\mathbf{Del}_n$, and that all proofs in the chain are honestly generated for true statements. For $n = 1$, this holds by the definition of $\mathsf{IssueDel}$, and (r^{-1}, r, v, z) is a witness for $\mathcal{R}_{\mathsf{VIss}}$; in particular, $rC = c(\mathbf{S}_1)v \cdot r\mathsf{G} = v(X' + \sum_{i\in[t]} s_{1,i}Y'_i)$. For the step, $\mathsf{Delegate}$ is run on $\mathbf{M}_{n+1} \preceq \mathbf{M}_n$, so its check passes, and it samples $(\eta, z') \in \mathbb{Z}_p^* \times \mathbb{Z}_p$, where η denotes the delegation randomizer called μ in the construction. Since $s_{n+1,i} = s_{n,i}$ for all $i \notin \mathbf{L}_n$,

$$\eta\Big(c(\mathbf{S}_n)v\mathsf{G} + \sum_{i\in\mathbf{L}_n}(s_{n+1,i} - s_{n,i})vy_i\mathsf{G}\Big) = c(\mathbf{S}_{n+1})(\eta v)\mathsf{G}.$$

Hence $\mathsf{Delegate}$ outputs $\mathsf{EC}' = ((\eta v)\mathsf{G} + (\eta z + z')\mathsf{H},\ (\eta z + z')\mathsf{G},\ c(\mathbf{S}_{n+1})(\eta v)\mathsf{G})$, $\mathbf{MK}' = ((\eta v)y_i\mathsf{G})_{i\in\mathbf{L}_{n+1}}$ (well defined because $\mathbf{L}_{n+1} \subseteq \mathbf{L}_n$) and $\mathsf{dk}' = \eta z + z'$. This is of the required form with $\eta v \in \mathbb{Z}_p^*$, and (η, z') is a witness for $\mathcal{R}_{\mathsf{VDel}}$.

Now consider $\mathsf{ObtainDel}(\mathsf{ipar}, \mathbf{Del}_n)$. The chain satisfies $\mathbf{M}_{i+1} \preceq \mathbf{M}_i$, we have $\mathsf{EZ} = \mathsf{dk}\mathsf{G}$ and $\mathsf{EV} - \mathsf{dk}\mathsf{H} = v\mathsf{G} \neq 0_{\mathbb{G}}$, and by completeness every proof in the chain verifies. Hence $\mathsf{ObtainDel}$ outputs $\mathbf{M} = \mathbf{M}_n$ and $\mathsf{Cred} = ((V, C), \mathbf{MK}_n)$ with $V = v\mathsf{G}$ and $C = c(\mathbf{S}_n)V$. Let $\phi = \phi_{\mathbf{I},\mathbf{D}}$ with $\phi(\mathbf{M}_n) = 1$, i.e., $s_{n,i} = d_i$ for all $i \in \mathbf{I}$, and write $s_i := s_{n,i}$. Then $\mathsf{ShowCred}$ outputs $V' = \mu V$, $W = c(\mathbf{S}_n)V' + \mu'\mathsf{H}$ and $Q_i = s_i V' + \beta_i \mathsf{G}$ for $i \in \bar{\mathbf{I}}$, using fresh presentation randomness $\mu \in \mathbb{Z}_p^*$ and $\mu', \beta_i \in \mathbb{Z}_p$, and proves $\mathcal{R}_{\mathsf{VPres}}$ for its value $P = \sum_{i\in\bar{\mathbf{I}}} \beta_i Y'_i - \mu' R$ with the witness $(\mu', (\beta_i, s_i)_{i\in\bar{\mathbf{I}}})$. The verifier reconstructs the same ipar from isk and pp. Its reconstruction of P gives the same value:

$$\begin{aligned} r\Big(xV' + \sum_{i\in\bar{\mathbf{I}}} y_i Q_i - W + \sum_{i\in\mathbf{I}} y_i d_i V'\Big) &= r\Big(\sum_{i\in\bar{\mathbf{I}}} \beta_i y_i \mathsf{G} - \mu'\mathsf{H} + \sum_{i\in\mathbf{I}} y_i(d_i - s_i)V'\Big) \\ &= \sum_{i\in\bar{\mathbf{I}}} \beta_i Y'_i - \mu' R. \end{aligned}$$

Thus the verifier checks π on exactly the statement for which it was honestly generated, so it verifies by completeness. Finally, $V' = \mu v\mathsf{G} \neq 0_{\mathbb{G}}$. Thus $\mathsf{VerifyShow}$ outputs 1 with probability 1. □

D GGM Analyses

We analyze our constructions in the generic group model. Effectively, the adversary is limited to linearly combining the elements it sees and treating secret

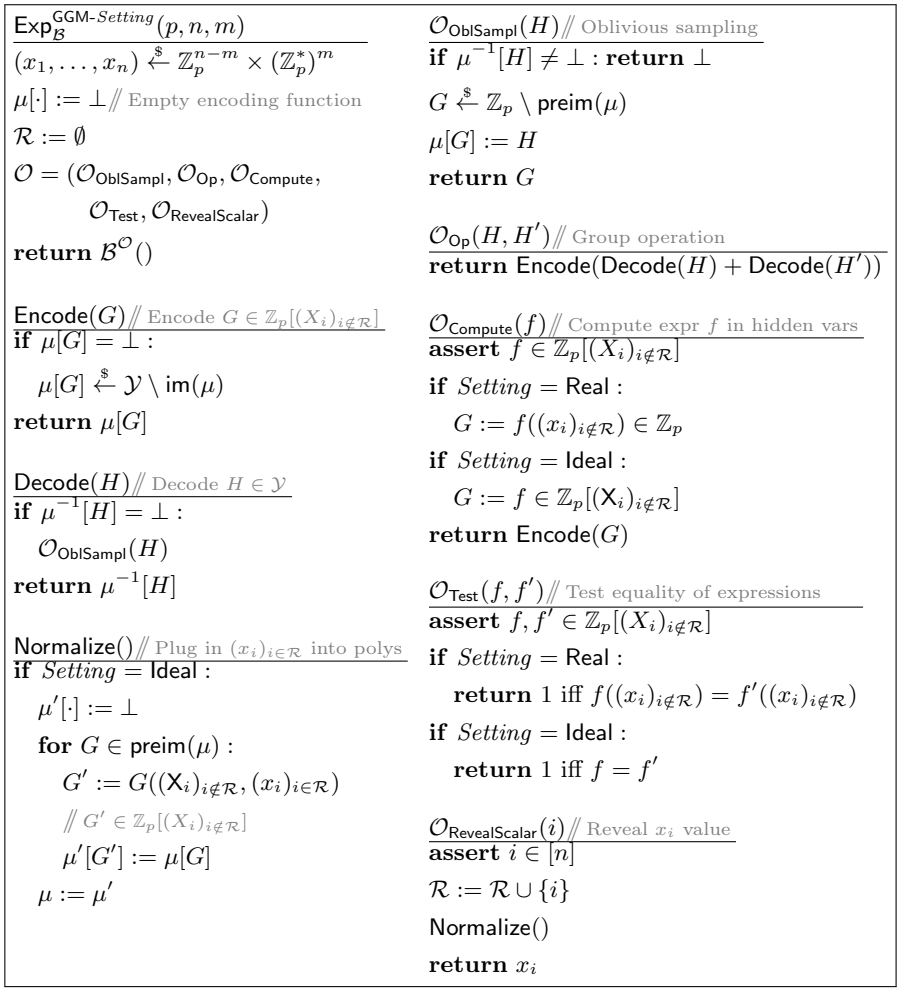


Fig. 7: The generic group model. The values $x_i \xleftarrow{\$} S_i$ are random secrets. In the Real setting, the encoding function encodes elements of $\mathbb{Z}_p$. In the Ideal setting, it encodes elements of $\mathbb{Z}_p[(X_i)_{i\notin\mathcal{R}}]$, i.e., polynomials with corresponding formal variables X_i for all x_i that were not revealed.

exponents (e.g., secret keys, randomness) as a symbol (like a polynomial variable) rather than a scalar number.

We model the generic group model using the framework depicted in Figure 7. For $Setting = \mathsf{Real}$, the model describes the ability of some arbitrary game $\mathcal{B}$ that interacts with the generic group through its oracles. We take $|\mathcal{Y}| = p$ and encode group elements using a lazily sampled random bijection $\mu : \mathbb{Z}_p \to \mathcal{Y}$. We call the encodings $H \in \mathcal{Y}$ *labels*. $\mathcal{B}$ performs group operations with the $\mathcal{O}_{\mathsf{Op}}$ oracle. Oblivious sampling is supported: $\mathcal{B}$ can simply choose any encoding $H \in \mathcal{Y}$ in group operations, which will decode to some random element $G \in \mathbb{Z}_p$. There are

hidden (secret) variables $x_1, \ldots, x_n \in \mathbb{Z}_p$, which $\mathcal{B}$ does not have direct access to. $\mathcal{B}$ can ask for polynomial functions $f \in \mathbb{Z}_p[(X_i)_{i \notin \mathcal{R}}]$ to be computed by the $\mathcal{O}_{\mathsf{Compute}}$ oracle, which returns the encoding of $f(x_1, \ldots, x_n)$. Additionally, $\mathcal{B}$ can adaptively ask for the value of any x_i to be revealed through the $\mathcal{O}_{\mathsf{RevealScalar}}$ oracle; repeated queries return the same scalar. The standard setup is that $\mathcal{B}$ is a security game that internally runs a generic adversary $\mathcal{A}$. The adversary $\mathcal{A}$ gets access to $\mathcal{O}_{\mathsf{Op}}$, while the game $\mathcal{B}$ uses $\mathcal{O}_{\mathsf{Compute}}$ and $\mathcal{O}_{\mathsf{RevealScalar}}$ to simulate the oracles of the real-world protocol for $\mathcal{A}$. Figures 8 and 9 give these games for our two constructions.

With *Setting* = Ideal, Figure 7 models a setting where the variables x_i are treated symbolically rather than using their actual scalar values. In this setting, μ is a partial injective map from the stored polynomials in $\mathbb{Z}_p[(X_i)_{i \notin \mathcal{R}}]$ to $\mathcal{Y}$. Group operations and computations are performed symbolically, i.e., on $\mathbb{Z}_p[(X_i)_{i \notin \mathcal{R}}]$. If $\mathcal{B}$ decides to reveal a variable x_i, the Normalize function is called, which substitutes all occurrences of X_i in the polynomials with the actual value x_i. This is a nonstandard operation in the generic group model, but it will allow us to adaptively switch the encryption randomness variables in our construction from symbolic to concrete scalars. During Normalize, it may happen that two polynomials G and G' in $\mathsf{preim}(\mu)$ with different encodings become equal after substituting the revealed values (and hence would have had the same encoding in the real world). Such normalization collisions are included in the full-evaluation collision event collision bounded below.

The main invariant of the symbolic setting is that all stored group elements are linear combinations of the values queried to $\mathcal{O}_{\mathsf{Compute}}$ and the constant 1. More precisely, let $\mathcal{F}$ contain all polynomials queried to $\mathcal{O}_{\mathsf{Compute}}$, with the currently revealed values substituted. Then every $G \in \mathsf{preim}(\mu)$ lies in $\mathrm{span}_{\mathbb{Z}_p}(\mathcal{F} \cup \{1\})$: constants introduced by $\mathcal{O}_{\mathsf{OblSampl}}$ are scalar multiples of 1.

Lemma 1 (GGM symbolic execution). *Let $p \in \mathbb{N}$ be prime, $n \in \mathbb{N}$, and $m \in [n]$. Let $\mathcal{B}$ make at most q oracle queries and let $d_{\max}$ upper-bound the total degree of every polynomial queried to $\mathcal{O}_{\mathsf{Compute}}$ and $\mathcal{O}_{\mathsf{Test}}$. Then*

$$\left|\Pr[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{GGM\text{-}Real}}(p,n,m)=1] - \Pr[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{GGM\text{-}Ideal}}(p,n,m)=1]\right| \leq \frac{12q^2}{p} + n\binom{3q}{2}\frac{d_{\max}}{p-1}.$$

In particular, if $n, q, d_{\max}$ are polynomially bounded and p is exponential in the security parameter, the difference is negligible.

Proof. We first introduce intermediate experiments hybrid-Real and hybrid-Ideal, which work like Real and Ideal, respectively, except that during $\mathcal{O}_{\mathsf{OblSampl}}(H)$, the value G is sampled from $\mathbb{Z}_p$ instead of $\mathbb{Z}_p \setminus \mathsf{preim}(\mu)$. This may lead to inconsistently reassigning values, which we tolerate in the hybrid games. Given that $|\mathsf{preim}(\mu)| \leq 3q$ and $\mathcal{O}_{\mathsf{OblSampl}}$ is invoked at most $2q$ times, we get that $|\Pr[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{GGM\text{-}}\mathit{Setting}}(p,n,m)=1] - \Pr[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{GGM\text{-hybrid-}}\mathit{Setting}}(p,n,m)=1]| \leq 2q \cdot 3q/p$ for both *Setting* $\in \{\mathsf{Real}, \mathsf{Ideal}\}$.

We now consider the difference between hybrid-Real and hybrid-Ideal. We partition the ideal hybrid into phases separated by $\mathcal{O}_{\mathsf{RevealScalar}}$ queries. Without loss of generality, we assume that $\mathcal{O}_{\mathsf{RevealScalar}}$ is never called twice for the same input i. Let $\mathbf{G}_0$ be the set of G such that $\mu(G) \neq \bot$ at any point before the first call to $\mathcal{O}_{\mathsf{RevealScalar}}$ (this includes values that are later overwritten by $\mathcal{O}_{\mathsf{OblSampl}}$) in an execution of hybrid-Ideal. Additionally, $\mathbf{G}_0$ contains 0 and $f - f'$ for all $\mathcal{O}_{\mathsf{Test}}(f, f')$ queries during this time. For $j > 0$, let $\mathbf{G}_j$ be the set of G such that $\mu(G) \neq \bot$ at any point after the jth call and before the $j+1$st call (if that exists) to $\mathcal{O}_{\mathsf{RevealScalar}}$ (this includes values that are later overwritten by $\mathcal{O}_{\mathsf{OblSampl}}$). Additionally, $\mathbf{G}_j$ contains 0 and $f - f'$ for all $\mathcal{O}_{\mathsf{Test}}(f, f')$ queries during this time. Let $\mathcal{R}_0 = \emptyset$ and let $\mathcal{R}_j$ be the set $\mathcal{R}$ of revealed indices i after the jth query to $\mathcal{O}_{\mathsf{RevealScalar}}$. We say that there is a *collision* in $\mathbf{G}_j \subset \mathbb{Z}_p[(X_i)_{i \notin \mathcal{R}_j}]$ if for some $G, G' \in \mathbf{G}_j$ with $G \neq G'$, it holds that $G((x_i)_{i \notin \mathcal{R}_j}) = G'((x_i)_{i \notin \mathcal{R}_j})$. If there is no collision in any $\mathbf{G}_j$, then hybrid-Ideal behaves exactly like hybrid-Real, i.e., $\Pr[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{GGM\text{-}hybrid\text{-}Ideal}}(p,n,m) = 1 \wedge \neg\mathsf{collision}] = \Pr[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{GGM\text{-}hybrid\text{-}Real}}(p,n,m) = 1 \wedge \neg\mathsf{collision}]$.

We can bound the probability of a collision in $\mathbf{G}_j$ using Schwartz-Zippel as usual. More specifically, because $|\mathbf{G}_j| \leq 3q$, there are at most $\binom{3q}{2}$ pairs $G \neq G'$, each of which collides with probability at most $d_{\max}/(p-1)$ due to Schwartz-Zippel. Applying the union bound, we get that

$$\Pr[\mathsf{collision}] \leq n\binom{3q}{2}\frac{d_{\max}}{p-1}.$$

Overall, this implies $|\Pr[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{GGM\text{-}hybrid\text{-}Real}}(p,n,m) = 1] - \Pr[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{GGM\text{-}hybrid\text{-}Ideal}}(p,n,m) = 1]| \leq n \cdot \binom{3q}{2} \cdot d_{\max}/(p-1)$ and hence $|\Pr[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{GGM\text{-}Real}}(p,n,m) = 1] - \Pr[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{GGM\text{-}Ideal}}(p,n,m) = 1]| \leq 2 \cdot 2q \cdot 3q/p + n \cdot \binom{3q}{2} \cdot d_{\max}/(p-1)$. □

Since the hybrids agree outside collision and each hybrid step changes the execution only with probability $6q^2/p$, the proof shows more precisely that

$$\Pr[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{GGM\text{-}Real}}(p,n,m) = 1] \leq \Pr[\mathsf{Exp}_{\mathcal{B}}^{\mathsf{GGM\text{-}Ideal}}(p,n,m) = 1 \wedge \neg\mathsf{collision}] + \frac{12q^2}{p} + n\binom{3q}{2}\frac{d_{\max}}{p-1},$$

with collision evaluated on the sampled x_i in the ideal execution. We use this form whenever a symbolic argument excludes collision.

D.1 Shared setup for the unforgeability proofs

Both unforgeability proofs follow the same three steps.

1. We write down the game $\mathcal{B}$ that we analyze in the framework of Figure 7. It runs the adversary $\mathcal{A}$ of Figure 1 and answers all of $\mathcal{A}$'s queries using the GGM oracles. Figures 8 and 9 give this game for the Subset and the Vector construction, respectively.

2. We argue that $\mathcal{A}$'s winning probability in $\mathcal{B}$, run in the Real setting, differs from that in the unforgeability experiment by at most the zero-knowledge advantage of the issuance and delegation proofs, plus a random-oracle error for the Vector construction.
3. We switch $\mathcal{B}$ to the Ideal setting using Lemma 1 (for the Vector construction, via **??**) and show that there, $\mathcal{A}$ cannot output a fresh accepting presentation (for the Vector construction, outside the exceptional events bounded in **??**).

Nodes, handles, and revealed keys. In Figure 1, every successful $\mathcal{O}_{\mathsf{Cred}}$ or $\mathcal{O}_{\mathsf{Delegate}}$ query creates one new chain node, i.e., one new encrypted credential, and returns a new handle. Rejected queries create neither a node nor a handle. We number nodes in creation order, so node j is the last node of the chain stored under handle j. We write $\mathbf{M}_j$ for the message of node j, and N bounds the number of nodes. An $\mathcal{O}_{\mathsf{Reveal}}(j)$ call returns the chain $\mathsf{Del}_1, \ldots, \mathsf{Del}_\ell$ stored under handle j and only the decryption key dk for the last node Del_ℓ, which is node j. Let $\mathcal{J}_R$ be the set of nodes whose decryption key was returned by some $\mathcal{O}_{\mathsf{Reveal}}$ call, and $\mathcal{J}_H$ the set of all other nodes; unless stated otherwise, both sets are taken at the end of the game. The list of messages for which $\mathcal{A}$ obtained a credential is then

$$\mathcal{Q} = \bigcup_{j \in \mathcal{J}_R} \{\mathbf{M} : \mathbf{M} \preceq \mathbf{M}_j\},$$

and $\mathcal{A}$ wins if its final presentation is accepted for a policy ϕ^* that no $\mathbf{M} \in \mathcal{Q}$ satisfies. We call such a presentation *fresh*.

How $\mathcal{B}$ uses the generic group. We identify each group element with its discrete logarithm with respect to G. Thus G corresponds to the polynomial 1, and H to the variable h. The game $\mathcal{B}$ creates every group element of the issuer and the honest users with $\mathcal{O}_{\mathsf{Compute}}$, exactly as listed in the figures. It forwards every group operation of $\mathcal{A}$, of the proof simulators (see below), and of the Vector presentation verifier to $\mathcal{O}_{\mathsf{Op}}$. For every label L it handles, $\mathcal{B}$ records the polynomial $[\![L]\!]$ that L encodes, i.e., the linear combination of 1 and of its $\mathcal{O}_{\mathsf{Compute}}$ queries from which L was obtained. A label that $\mathcal{B}$ has not seen before is first given to $\mathcal{O}_{\mathsf{OblSampl}}$, which assigns it a constant polynomial. (In the Real setting, all polynomials recorded for the same label evaluate to the same scalar, so it does not matter which one is used.) $\mathcal{B}$ calls $\mathcal{O}_{\mathsf{RevealScalar}}$ only inside $\mathcal{O}_{\mathsf{Reveal}}(j)$, and only for the decryption key of node j. It answers $\mathcal{O}_{\mathsf{VerifyShow}}$ queries and checks the final presentation using the GGM oracles, returning only the resulting bit.

Why $\mathcal{B}$ behaves like the unforgeability experiment. The game $\mathcal{B}$ (Figures 8 and 9) differs from Figure 1 in the following three ways.

– *Simulated proofs.* $\mathcal{B}$ replaces the honest issuance and delegation proofs by the stateful, witness-free simulators $\mathsf{Sim}_{\mathrm{iss}}$ and $\mathsf{Sim}_{\mathrm{del}}$ given by the zero-knowledge property of these proof systems (Appendix B.1). As generic algorithms, they access the group only through operations that $\mathcal{B}$ forwards to $\mathcal{O}_{\mathsf{Op}}$. Each simulator reads and programs only its own tagged random-oracle

domain. The Vector presentation-challenge random oracle remains independent and is answered by lazy sampling. We first replace the issuance proofs and then the delegation proofs. Let $\mathcal{C}_{\text{iss}}$ (resp. $\mathcal{C}_{\text{del}}$) be the zero-knowledge adversary that runs the experiment with $\mathcal{A}$ and obtains the issuance (resp. delegation) proofs from its proof oracle. Repeated-proof zero knowledge, which covers adaptive proof requests, bounds the resulting change in $\mathcal{A}$'s winning probability by the sum of their advantages.

– *Fresh randomness for delegated nodes.* Honest delegation replaces the randomness (v, z) of the parent's encrypted credential by $(\mu v, \mu z + z')$, where $(\mu, z') \xleftarrow{\$} \mathbb{Z}_p^* \times \mathbb{Z}_p$ is fresh. For a fixed parent (v, z) with $v \neq 0$, the map

$$(\mu, z') \longmapsto (\mu v, \mu z + z')$$

is a bijection of $\mathbb{Z}_p^* \times \mathbb{Z}_p$. Hence, conditioned on everything sampled before the delegation query (in particular the parent's randomness) and on the delegated message, the new pair $(\mu v, \mu z + z')$ is uniform in $\mathbb{Z}_p^* \times \mathbb{Z}_p$ (once the delegation proof is simulated, (μ, z') enters only through this pair). As we check for each construction below, its encrypted credential also has exactly the form of a fresh $\mathsf{IssueDel}$ output on the delegated message, with decryption key $\mathsf{dk}' = \mu z + z'$. Thus $\mathcal{B}$ creates every node, whether issued or delegated, with fresh variables and only checks the delegation condition $\mathbf{M}' \preceq \mathbf{M}$. The delegation statement remains true (with witness $\mu = v_j / v_i$ and $z' = z_j - \mu z_i$ for parent i and child j), and its proof is simulated without a witness.

– *GGM oracles.* In the Real setting, $\mathcal{O}_{\mathsf{Compute}}$ returns exactly the group elements of the construction, and the GGM group operations and tests evaluate the verification equations on the actual secrets. The Vector game additionally rejects unless the supplied challenge has a unique matching preimage in the random-oracle query table; we bound the resulting error below.

In addition, the Subset game writes its credential randomness as a product with the same distribution; see Appendix D.2.

Span invariant. In the Ideal setting, every recorded polynomial is a linear combination of 1 and of the polynomials that $\mathcal{B}$ queried to $\mathcal{O}_{\mathsf{Compute}}$, with revealed keys substituted (see the discussion before Lemma 1). The proof simulators make no $\mathcal{O}_{\mathsf{Compute}}$ queries; they only combine labels they have received (including those in random-oracle queries), which already lie in this span, with obliviously sampled constants. Hence every group element that $\mathcal{A}$ submits, in particular its final presentation, is such a linear combination, and analyzing $\mathcal{B}$ amounts to reading its $\mathcal{O}_{\mathsf{Compute}}$ queries off the figures. (For the analysis, we over-approximate and pretend that $\mathcal{A}$ has access to all node elements: $\mathcal{B}$ computes the elements of node j when the node is created, whereas $\mathcal{A}$ sees them only after revealing a handle whose chain contains node j. So several of the coefficients below must in fact be zero, but we do not rely on this fact.) The coefficients are scalars that may depend on the whole transcript, including revealed keys and random-oracle answers; unrevealed variables remain indeterminates.

Query count. In each proof, Q bounds all GGM oracle calls made by $\mathcal{B}$. This includes the forwarded group operations of $\mathcal{A}$ and of the proof simulators, label bookkeeping, revelations, issuance checks, auxiliary tests, and every verification, including that of the final presentation.

D.2 GGM Analysis for Theorem 2

Figure 8 gives the game $\mathcal{B}_{\text{sub}}$ for Subset DKVAC (Figures 3 and 4) in the experiment of Figure 1.

The game. For a set $\mathbf{S}$, put $f_{\mathbf{S}}(y) = \prod_{s\in\mathbf{S}}(y+s)$, with $f_{\emptyset} = 1$, so that $f_{\mathbf{S}}(y)/(y+s) = f_{\mathbf{S}\setminus\{s\}}(y)$ for $s \in \mathbf{S}$. The hidden variables of $\mathcal{B}_{\text{sub}}$ are h, x, y for H and the issuer key, and u_j, z_j for each node $j \in [N]$. The GGM experiment of Figure 7 samples $z_1, \ldots, z_N \xleftarrow{\$} \mathbb{Z}_p$ and $h, x, y, u_1, \ldots, u_N \xleftarrow{\$} \mathbb{Z}_p^*$, so $n = 2N + 3$ and $m = N + 3$; $\mathcal{B}_{\text{sub}}$ accesses them only through the GGM oracles.

When $\mathcal{B}_{\text{sub}}$ creates node j on a set $\mathbf{S}_j$, it computes the encrypted credential of Figure 4 with credential randomness $v_j := u_j f_{\mathbf{S}_j}(y)$ and decryption key z_j:

$$\begin{aligned} \mathsf{E}_j &= \big(x u_j f_{\mathbf{S}_j}(y) + z_j h\big)\mathsf{G}, & \mathsf{EV}_j &= u_j f_{\mathbf{S}_j}(y)\mathsf{G}, \\ \mathsf{EZ}_j &= z_j \mathsf{G}, & C_{j,s} &= u_j f_{\mathbf{S}_j\setminus\{s\}}(y)\mathsf{G}. \end{aligned}$$

In other words, the game sets $C_{j,s} = u_j f_{\mathbf{S}_j}(y)\mathsf{G}/(y+s)$ instead of $C_{j,s} = v_j\mathsf{G}/(y+s)$. This way, every component is a polynomial in the hidden variables, as $\mathcal{O}_{\mathsf{Compute}}$ requires; with v_j as a variable, $C_{j,s}$ would contain the inverse of $y+s$. The change does not affect the distribution. $\mathcal{O}_{\mathsf{Cred}}$ creates a node only if $y + s \neq 0$ for all $s \in \mathbf{S}$, and $\mathcal{O}_{\mathsf{Delegate}}$ only for subsets of such sets, so $f_{\mathbf{S}_j}(y) \neq 0$. Multiplication by this nonzero scalar is a bijection of $\mathbb{Z}_p^*$, so for every fixed y and transcript so far, v_j is uniform in $\mathbb{Z}_p^*$, exactly as in the construction.

For delegation from node i with randomness (v_i, z_i) to a set $\mathbf{S}' \subseteq \mathbf{S}_i$, the construction computes

$$\begin{aligned} \mu\mathsf{E}_i + z'\mathsf{H} &= (\mu v_i)x\mathsf{G} + (\mu z_i + z')\mathsf{H}, & \mu\mathsf{EV}_i &= (\mu v_i)\mathsf{G}, \\ \mu\mathsf{EZ}_i + z'\mathsf{G} &= (\mu z_i + z')\mathsf{G}, & \mu C_{i,s} &= (\mu v_i)\mathsf{G}/(y+s) \quad (s \in \mathbf{S}'). \end{aligned}$$

This is a fresh encrypted credential on $\mathbf{S}'$ with randomness $(\mu v_i, \mu z_i + z')$ and decryption key $\mathsf{dk}' = \mu z_i + z'$. By Appendix D.1, the unforgeability experiment and $\mathcal{B}_{\text{sub}}$ in the Real setting therefore differ only by the proof simulation. Let ϵ_{ZK} denote the cost of simulating the issuance and delegation proofs:

$$\begin{aligned} \epsilon_{\mathsf{ZK}} := {} & \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIP}_{\mathsf{Slss}},\mathsf{Sim}_{\text{iss}},\mathcal{C}_{\text{iss}}}(\lambda) \\ & + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIP}_{\mathsf{SDel}},\mathsf{Sim}_{\text{del}},\mathcal{C}_{\text{del}}}(\lambda). \end{aligned}$$

Then

$$\Pr[\mathsf{Exp}^{\mathsf{Unforge}}_{\mathcal{A},\mathsf{DKVAC}}(\lambda) = 1] \leq \epsilon_{\mathsf{ZK}} + \Pr[\mathsf{Exp}^{\mathsf{GGM\text{-}Real}}_{\mathcal{B}_{\text{sub}}}(p, 2N+3, N+3) = 1].$$

$\underline{\mathcal{B}_{\mathrm{sub}}()}$
// Variables $h, x, y, (u_j, z_j)_{j\in[N]}$
$\mathsf{G} := \mathcal{O}_{\mathsf{Compute}}(1)$; $\mathsf{H} := \mathcal{O}_{\mathsf{Compute}}(h)$
$X := \mathcal{O}_{\mathsf{Compute}}(x)$; $Y := \mathcal{O}_{\mathsf{Compute}}(y)$
$\mathsf{ctr} := 0$; $\mathcal{Q} := \emptyset$
$(\mathsf{Show}^*, \phi^*) \leftarrow \mathcal{A}^{\mathcal{O}}((\mathsf{G}, \mathsf{H}), (X, Y))$
return $\mathsf{VerifyShow}(\phi^*, \mathsf{Show}^*) \wedge$
$(\nexists \mathbf{M} \in \mathcal{Q} : \phi^*(\mathbf{M}) = 1)$

$\underline{\mathsf{NewNode}(\mathbf{S})}$
$\mathsf{ctr} := \mathsf{ctr} + 1$; $j := \mathsf{ctr}$; $\mathbf{S}_j := \mathbf{S}$
$\mathsf{E}_j := \mathcal{O}_{\mathsf{Compute}}(x u_j f_{\mathbf{S}}(y) + z_j h)$
$\mathsf{EV}_j := \mathcal{O}_{\mathsf{Compute}}(u_j f_{\mathbf{S}}(y))$
$\mathsf{EZ}_j := \mathcal{O}_{\mathsf{Compute}}(z_j)$
foreach $s \in \mathbf{S}$:
$C_{j,s} := \mathcal{O}_{\mathsf{Compute}}(u_j f_{\mathbf{S}\setminus\{s\}}(y))$
$\mathsf{EC}_j := (\mathsf{E}_j, \mathsf{EV}_j, \mathsf{EZ}_j, (C_{j,s})_{s\in\mathbf{S}})$
return j

$\underline{\mathcal{O}_{\mathsf{Cred}}(\mathbf{S})}$
if $\mathcal{O}_{\mathsf{Test}}(f_{\mathbf{S}}(y), 0) = 1$: **return** $\perp$
$j := \mathsf{NewNode}(\mathbf{S})$
$\pi_j \leftarrow \mathsf{Sim}_{\mathrm{iss}}.\mathsf{Prove}((X, Y), \mathsf{EC}_j, \mathbf{S})$
$\mathbf{T}_j := \big((\mathsf{EC}_j, \pi_j, \mathbf{S})\big)$
return j

$\underline{\mathcal{O}_{\mathsf{Delegate}}(\mathbf{S}', i)}$
require $i \le \mathsf{ctr} \wedge \mathbf{S}' \subseteq \mathbf{S}_i$
$j := \mathsf{NewNode}(\mathbf{S}')$
$\pi_j \leftarrow \mathsf{Sim}_{\mathrm{del}}.\mathsf{Prove}(\mathsf{EC}_i, \mathsf{EC}_j, \mathbf{S}')$
$\mathbf{T}_j := \mathbf{T}_i \,\|\, (\mathsf{EC}_j, \pi_j, \mathbf{S}')$
return j

$\underline{\mathcal{O}_{\mathsf{Reveal}}(j)}$
require $j \le \mathsf{ctr}$
$\mathsf{dk} := \mathcal{O}_{\mathsf{RevealScalar}}(z_j)$
$\mathcal{Q} := \mathcal{Q} \cup \{\mathbf{S} : \mathbf{S} \subseteq \mathbf{S}_j\}$
return $(\mathbf{T}_j, \mathsf{dk})$

$\underline{\mathsf{VerifyShow}(\phi_{\mathbf{D}}, \mathsf{Show})}$
parse Show **as** (V', C')
$P_V := [\![V']\!]$; $P_C := [\![C']\!]$
$A_{\mathbf{D}}(y) := \sum_{d\in\mathbf{D}} f_{\mathbf{D}\setminus\{d\}}(y)$
return $\mathcal{O}_{\mathsf{Test}}(P_V, 0) = 0 \wedge$
$\mathcal{O}_{\mathsf{Test}}(A_{\mathbf{D}}(y) P_V - x f_{\mathbf{D}}(y) P_C, 0) = 1$

Fig. 8: The game $\mathcal{B}_{\mathrm{sub}}$ analyzed for the unforgeability of Subset DKVAC, where $f_{\mathbf{S}}(y) = \prod_{s\in\mathbf{S}}(y + s)$ and $\mathbf{T}_j$ is the chain stored under handle j. $\mathcal{O}_{\mathsf{VerifyShow}}(\mathsf{Show}, \phi)$ returns $\mathsf{VerifyShow}(\phi, \mathsf{Show})$, and malformed inputs and $\mathbf{D} = \emptyset$ are rejected. Group operations of $\mathcal{A}$ and of the simulators are forwarded to $\mathcal{O}_{\mathsf{Op}}$, and $[\![L]\!]$ is the polynomial recorded for label L (Appendix D.1). The simulators answer random-oracle queries in their domains. $\mathcal{O}_{\mathsf{RevealScalar}}(z_j)$ reveals the variable z_j.

Issuance check and verification. $\mathcal{O}_{\mathsf{Cred}}$ rejects malformed requests directly. For a well-formed set $\mathbf{S}$, $\mathcal{O}_{\mathsf{Test}}(f_{\mathbf{S}}(y), 0) = 1$ holds exactly when $y + s = 0$ for some $s \in \mathbf{S}$. In this case, $\mathcal{B}_{\mathrm{sub}}$ returns $\perp$ without creating a node, proof, or handle, as in Figures 3 and 4. For verification, $\mathcal{B}_{\mathrm{sub}}$ rejects malformed inputs and $\mathbf{D} = \emptyset$. With $A_{\mathbf{D}}(y) = \sum_{d\in\mathbf{D}} f_{\mathbf{D}\setminus\{d\}}(y)$, it accepts exactly when

$$\mathcal{O}_{\mathsf{Test}}(P_V, 0) = 0 \quad \text{and} \quad \mathcal{O}_{\mathsf{Test}}(A_{\mathbf{D}}(y) P_V - x f_{\mathbf{D}}(y) P_C, 0) = 1.$$

When $f_{\mathbf{D}}(y) \neq 0$, the second test is the verification equation $\sum_{d\in\mathbf{D}}(y+d)^{-1}V' = xC'$ multiplied by $f_{\mathbf{D}}(y)$. If $y = -d$ for some $d \in \mathbf{D}$, then $A_{\mathbf{D}}(-d) = \prod_{e\in\mathbf{D}\setminus\{d\}}(e-d) \neq 0$, so the second test forces $P_V = 0$ and the first test rejects, as the construction does. These tests create no encodings.

Switching to the symbolic setting. Let a bound all requested issuance-set sizes, including rejected requests, and b all disclosure-set sizes. The polynomials queried to $\mathcal{O}_{\mathsf{Compute}}$ have degree at most $a+2$, the issuance tests degree at most a, and the verification tests degree at most $a+b+3$. Hence Lemma 1 bounds the cost of switching $\mathcal{B}_{\mathrm{sub}}$ to the Ideal setting by

$$\epsilon_{\mathsf{GGM}} = \frac{12Q^2}{p} + (2N+3)\binom{3Q}{2}\frac{a+b+3}{p-1}. \tag{2}$$

We now show that $\mathcal{B}_{\mathrm{sub}}$ never outputs 1 in the Ideal setting.

What the adversary can compute. Reading off the $\mathcal{O}_{\mathsf{Compute}}$ queries in Figure 8, the span invariant shows that the polynomial of every label that $\mathcal{A}$ submits has the form

$$\begin{aligned}&\gamma_g + \gamma_h h + \gamma_x x + \gamma_y y + \sum_{i\in\mathcal{J}_R} u_i f_{\mathbf{S}_i}(y)\Big(\gamma_{i,E}\,x + \gamma_{i,V} + \sum_{s\in\mathbf{S}_i}\frac{\gamma_{i,s}}{y+s}\Big)\\ &+ \sum_{j\in\mathcal{J}_H}\Big(\gamma_{j,E}\big(x u_j f_{\mathbf{S}_j}(y) + z_j h\big) + \gamma_{j,Z}\,z_j + u_j f_{\mathbf{S}_j}(y)\Big(\gamma_{j,V} + \sum_{s\in\mathbf{S}_j}\frac{\gamma_{j,s}}{y+s}\Big)\Big)\end{aligned} \tag{3}$$

with scalars $\gamma \in \mathbb{Z}_p$. Here $\gamma_{k,E}, \gamma_{k,V}, \gamma_{k,Z}, \gamma_{k,s}$ are the coefficients of $\mathsf{E}_k, \mathsf{EV}_k, \mathsf{EZ}_k, C_{k,s}$, and each $u_k f_{\mathbf{S}_k}(y)/(y+s) = u_k f_{\mathbf{S}_k\setminus\{s\}}(y)$ is a polynomial. For a revealed node i, the key z_i is a scalar, so the terms $z_i h$ and z_i of E_i and EZ_i are absorbed into γ_h and γ_g.

Symbolic acceptance. Suppose that $\mathcal{A}$ outputs a fresh presentation (V^*, C^*) for a nonempty set $\mathbf{D}^*$ that is accepted in the Ideal setting. Let $P_V = [\![V^*]\!]$ and $P_C = [\![C^*]\!]$ have the form (3) with coefficients α and β, respectively. Since $\mathcal{Q}$ contains exactly the subsets of revealed sets, freshness means $\mathbf{D}^* \not\subseteq \mathbf{S}_i$ for every $i \in \mathcal{J}_R$. In the Ideal setting, acceptance means $P_V \neq 0$ and $A_{\mathbf{D}^*}(y)P_V = x f_{\mathbf{D}^*}(y)P_C$ as polynomials. We divide by $f_{\mathbf{D}^*}(y)$ and regard both sides as polynomials in $x, h, (u_k)_k, (z_j)_{j\in\mathcal{J}_H}$ with coefficients in the field $\mathbb{Z}_p(y)$ of rational functions in y. With

$$w_{\mathbf{D}^*}(y) := \frac{A_{\mathbf{D}^*}(y)}{f_{\mathbf{D}^*}(y)} = \sum_{d\in\mathbf{D}^*}\frac{1}{y+d},$$

acceptance gives

$$w_{\mathbf{D}^*}(y)\,P_V = x\,P_C. \tag{4}$$

We compare the coefficients of the monomials in $x, h, (u_k)_k, (z_j)_{j\in\mathcal{J}_H}$ on both sides. We also use that 1, y, and the fractions $(y+s)^{-1}$ for distinct $s \in \mathbb{Z}_p$ are linearly independent over $\mathbb{Z}_p$. Indeed, multiplying a vanishing linear combination by $y+s$ and evaluating at $y = -s$ shows that the coefficient of $(y+s)^{-1}$ is zero; the remaining polynomial part then vanishes as well.

Coefficient comparison. The right side of (4) is a multiple of x and $w_{\mathbf{D}^*}(y) \neq 0$, so the part of P_V without x vanishes. For $j \in \mathcal{J}_H$, its monomials $z_j h$ and z_j give $\alpha_{j,E} = \alpha_{j,Z} = 0$, so hidden encrypted tags E_j do not contribute. The monomials h and 1 give $\alpha_h = 0$ and $\alpha_g + \alpha_y y = 0$, i.e., $\alpha_g = \alpha_y = 0$. For every node k, the monomial u_k gives, after dividing by $f_{\mathbf{S}_k}(y) \neq 0$, $\alpha_{k,V} + \sum_{s \in \mathbf{S}_k} \alpha_{k,s}(y+s)^{-1} = 0$, so these coefficients vanish as well. Cancelling x in (4), we obtain

$$P_V = x\Big(\alpha_x + \sum_{i \in \mathcal{J}_R} \alpha_{i,E}\, u_i f_{\mathbf{S}_i}(y)\Big), \qquad P_C = w_{\mathbf{D}^*}(y)\Big(\alpha_x + \sum_{i \in \mathcal{J}_R} \alpha_{i,E}\, u_i f_{\mathbf{S}_i}(y)\Big).$$

The right side of the second identity contains no monomial with x, h, z_j, or u_j for $j \in \mathcal{J}_H$, so the corresponding coefficients of P_C vanish. Comparing the remaining monomials 1 and u_i for $i \in \mathcal{J}_R$, and dividing the latter by $f_{\mathbf{S}_i}(y)$, gives

$$\beta_g + \beta_y y = \alpha_x\, w_{\mathbf{D}^*}(y), \qquad \beta_{i,V} + \sum_{s \in \mathbf{S}_i} \frac{\beta_{i,s}}{y+s} = \alpha_{i,E} \sum_{d \in \mathbf{D}^*} \frac{1}{y+d} \quad (i \in \mathcal{J}_R).$$

By linear independence, the first identity gives $\beta_g = \beta_y = 0$ and, since $\mathbf{D}^* \neq \emptyset$, also $\alpha_x = 0$. In the second identity, $\beta_{i,V} = 0$, so the elements EV_i cannot contribute to an accepting C^*. Moreover, $\alpha_{i,E} \neq 0$ implies $\mathbf{D}^* \subseteq \mathbf{S}_i$, because for $d \in \mathbf{D}^*$ the fraction $(y+d)^{-1}$ occurs on the left only if $d \in \mathbf{S}_i$. Since $P_V \neq 0$ and $\alpha_x = 0$, some $i^* \in \mathcal{J}_R$ has $\alpha_{i^*,E} \neq 0$. Hence $\mathbf{D}^* \subseteq \mathbf{S}_{i^*}$, i.e., $\mathbf{S}_{i^*} \in \mathcal{Q}$ satisfies $\phi_{\mathbf{D}^*}$, contradicting freshness.

Thus $\mathcal{B}_{\mathrm{sub}}$ never outputs 1 in the Ideal setting. Combining the proof simulation with Lemma 1 yields

$$\Pr[\mathsf{Exp}^{\mathsf{Unforge}}_{\mathcal{A},\mathsf{DKVAC}}(\lambda) = 1] \leq \epsilon_{\mathsf{ZK}} + \epsilon_{\mathsf{GGM}}. \tag{5}$$

Both terms are negligible for polynomially bounded N, a, b, Q and exponential p, proving Theorem 2.

D.3 GGM Analysis for Theorem 5

Figure 9 gives the game $\mathcal{B}_{\mathrm{vec}}$ for Vector DKVAC (Figures 5 and 6) in the experiment of Figure 1, with $\mathsf{NIP}_{\mathsf{VPres}}$ instantiated by Fiat–Shamir generalized Schnorr for Equation (1). Its hidden variables are $h, x, r, y_1, \ldots, y_t$ for H and the issuer key, and v_j, z_j for each node $j \in [N]$. The GGM experiment of Figure 7 samples $z_1, \ldots, z_N \xleftarrow{\$} \mathbb{Z}_p$ and $h, x, r, y_1, \ldots, y_t, v_1, \ldots, v_N \xleftarrow{\$} \mathbb{Z}_p^*$, so $n = 2N + t + 3$ and $m = N + t + 3$; $\mathcal{B}_{\mathrm{vec}}$ accesses them only through the GGM oracles. For node k with message $(\mathbf{S}_k, \mathbf{L}_k)$, write $a_k = x + \sum_{i \in [t]} y_i s_{k,i}$. Unlike in the Subset game, all components of the construction are already polynomials in these variables, so $\mathcal{B}_{\mathrm{vec}}$ uses the credential randomness v_j directly.

Why $\mathcal{B}_{\mathrm{vec}}$ behaves like the experiment. We first check that a delegated node has the form of a fresh one. Let node i have message $(\mathbf{S}, \mathbf{L})$ and randomness (v_i, z_i),

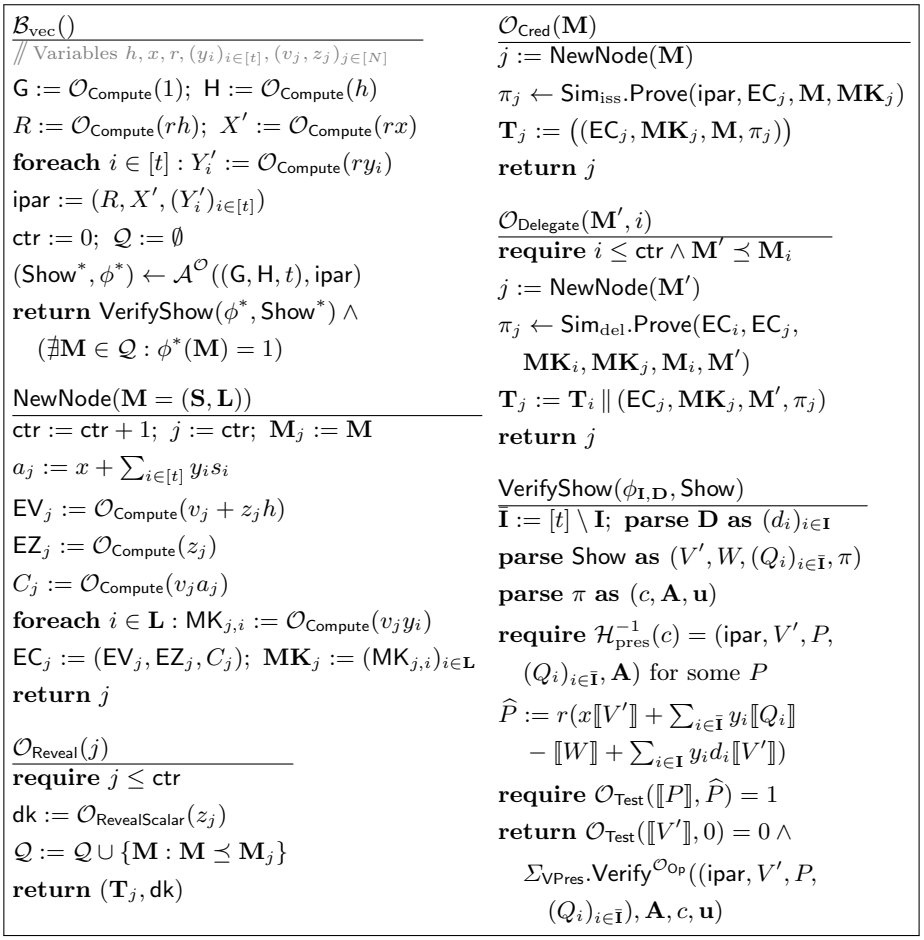


Fig. 9: The game $\mathcal{B}_{\mathrm{vec}}$ analyzed for the unforgeability of Vector DKVAC, with conventions as in Figure 8. $[\![L]\!]$ denotes the polynomial recorded for group-element label L (Appendix D.1). The presentation random oracle $\mathcal{O}_{\mathrm{RO}}(\mathsf{tag}_{\mathsf{VPres}}, \cdot)$ is answered by lazy sampling and never programmed; $\mathcal{H}_{\mathrm{pres}}$ records its queried well-formed inputs and answers. Here $\mathcal{H}^{-1}_{\mathrm{pres}}(c)$ returns the unique recorded input with answer c, written as a flat tuple, or $\perp$ if there is no unique preimage. $\Sigma_{\mathsf{VPres}}.\mathsf{Verify}^{\mathcal{O}_{\mathsf{Op}}}$ runs the generalized-Schnorr Sigma protocol verifier for Equation (1) on the statement and transcript $(\mathbf{A}, c, \mathbf{u})$, using GGM group-operation queries to $\mathcal{O}_{\mathsf{Op}}$.

and delegate it to $(\mathbf{S}', \mathbf{L}') \preceq (\mathbf{S}, \mathbf{L})$. Since $s'_l = s_l$ for $l \notin \mathbf{L}$, the construction

computes

$$\begin{aligned}
\mu\mathsf{EV}_i + z'\mathsf{H} &= \big(\mu v_i + (\mu z_i + z')h\big)\mathsf{G},\\
\mu\mathsf{EZ}_i + z'\mathsf{G} &= (\mu z_i + z')\mathsf{G},\\
\mu\Big(C_i + \sum_{l\in\mathbf{L}}(s'_l - s_l)\mathsf{MK}_{i,l}\Big) &= (\mu v_i)\Big(x + \sum_{l\in[t]} y_l s'_l\Big)\mathsf{G},\\
\mu\,\mathsf{MK}_{i,l} &= (\mu v_i) y_l \mathsf{G} \quad (l \in \mathbf{L}').
\end{aligned}$$

This is a fresh encrypted credential on $(\mathbf{S}', \mathbf{L}')$ with randomness $(\mu v_i, \mu z_i + z')$ and decryption key $\mathsf{dk}' = \mu z_i + z'$. As for the Subset game, with

$$\begin{aligned}
\epsilon_{\mathsf{ZK}} := {} & \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIP}_{\mathsf{Vlss}},\mathsf{Sim}_{\mathrm{iss}},\mathcal{C}_{\mathrm{iss}}}(\lambda)\\
& + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIP}_{\mathsf{VDel}},\mathsf{Sim}_{\mathrm{del}},\mathcal{C}_{\mathrm{del}}}(\lambda),
\end{aligned}$$

we obtain

$$\Pr[\mathsf{Exp}^{\mathsf{Unforge}}_{\mathcal{A},\mathsf{DKVAC}}(\lambda) = 1] \le \epsilon_{\mathsf{ZK}} + \epsilon_{\mathrm{RO}} + \Pr[\mathsf{Exp}^{\mathsf{GGM\text{-}Real}}_{\mathcal{B}_{\mathrm{vec}}}(p, 2N+t+3, N+t+3) = 1].$$

Here let q_H bound the total number of adversarial presentation-challenge queries and presentation verifications, including repeated attempts and final verification, and set

$$\epsilon_{\mathrm{RO}} := \frac{q_H + \binom{q_H}{2}}{p}.$$

To justify this error, first couple the real verifier to rejection whenever its full challenge input has not been queried by $\mathcal{A}$. In the rejection-coupled transcript, the corresponding oracle answer remains hidden and uniform, so the supplied c matches it with probability at most $1/p$ per attempt, even if earlier attempts used the same input. A union bound costs at most q_H/p. Next, distinct queried presentation inputs have the same answer with probability at most $\binom{q_H}{2}/p$. Outside these events, every potentially accepting proof has a unique matching queried preimage. The game retrieves its P label and checks (6) with $\mathcal{O}_{\mathsf{Test}}$, which in the Real setting is equivalent to the real verifier's reconstruction. Thus the remaining verification checks agree.

Verification. For a policy $\phi_{\mathbf{I},\mathbf{D}}$ with $\bar{\mathbf{I}} = [t]\backslash\mathbf{I}$, the verifier in Figure 5 reconstructs

$$P = r\left(xV' + \sum_{i\in\bar{\mathbf{I}}} y_i Q_i - W + \sum_{i\in\mathbf{I}} y_i d_i V'\right). \tag{6}$$

Instead, $\mathcal{B}_{\mathrm{vec}}$ reads c from π and searches $\mathcal{H}_{\mathrm{pres}}$ for queried well-formed presentation inputs with answer c. It rejects unless exactly one such input exists and it has the form $(\mathsf{tag}_{\mathsf{VPres}}, (\mathsf{ipar}, V', P, (Q_i)_{i\in\bar{\mathbf{I}}}), \mathbf{A})$ with the prescribed parameters, indices, and all fields other than P matching the presentation. It retrieves P and its recorded polynomial from this input, and uses $\mathcal{O}_{\mathsf{Test}}$ to check that the polynomial equals the right-hand side of (6). For the announcement $\mathbf{A} = (A_0, (A_i)_{i\in\bar{\mathbf{I}}})$,

the responses $\mathbf{u} = (u_\mu, (u_{\beta,i}, u_{s,i})_{i\in\bar{\mathbf{I}}})$, and the challenge c, the Schnorr verification equations hold exactly when the residuals

$$\begin{aligned} F_0 &= A_0 + cP - \sum_{i\in\bar{\mathbf{I}}} u_{\beta,i} Y'_i + u_\mu R, \\ F_i &= A_i + cQ_i - u_{s,i} V' - u_{\beta,i}\mathsf{G} \quad (i \in \bar{\mathbf{I}}) \end{aligned} \tag{7}$$

vanish as group elements; in the Ideal setting, this means equality of their recorded polynomials. After the syntax, unique-preimage, and reconstruction checks, $\mathcal{B}_{\text{vec}}$ accepts exactly when $\mathcal{O}_{\mathsf{Test}}(V', 0) = 0$ and $\Sigma_{\mathsf{VPres}}.\mathsf{Verify}^{\mathcal{O}_{\mathsf{Op}}}((\mathsf{ipar}, V', P, (Q_i)_{i\in\bar{\mathbf{I}}}), \mathbf{A}, c, \mathbf{u}) = 1$.

What the adversary can compute. Reading off the $\mathcal{O}_{\mathsf{Compute}}$ queries in Figure 9, the public elements have polynomials $1, h, rh, rx, (ry_i)_{i\in[t]}$, and node k has $v_k + z_k h$, z_k, $v_k a_k$, and $(v_k y_i)_{i\in\mathbf{L}_k}$. For a revealed node j, the key z_j is a scalar, so EV_j contributes v_j, while the terms $z_j h$ and z_j are absorbed into the coefficients of h and 1. Every linear combination of these public and node polynomials, and hence every polynomial recorded for a label obtained from them, has the following form with scalar coefficients γ:

$$\begin{aligned} \mathcal{E}_{\text{vec}}(\gamma) = \gamma_g + \gamma_h h + r\left(\gamma_r h + \gamma_x x + \sum_{i\in[t]} \gamma_{y,i} y_i\right) \\ + \sum_{j\in\mathcal{J}_R} v_j \left(\gamma_{v,j} + \gamma_{c,j} a_j + \sum_{i\in\mathbf{L}_j} \gamma_{j,i} y_i\right) \\ + \sum_{k\in\mathcal{J}_H} \Big(\gamma_{ev,k}(v_k + z_k h) + \gamma_{ez,k} z_k \\ + v_k\big(\gamma_{c,k} a_k + \sum_{i\in\mathbf{L}_k} \gamma_{k,i} y_i\big)\Big). \end{aligned} \tag{8}$$

There are no other $\mathcal{O}_{\mathsf{Compute}}$ queries. Fresh labels supplied by $\mathcal{A}$, including in well-formed random-oracle inputs, are registered with $\mathcal{O}_{\mathsf{OblSampl}}$ and contribute only constants. Thus every element submitted by $\mathcal{A}$, including the retrieved P, lies in (8).

Lemma 2 shows that accepting presentations satisfy the formal relations proved by $\mathsf{NIP}_{\mathsf{VPres}}$.

Lemma 2 (Schnorr guarantees relation in GGM). *Consider the game $\mathcal{B}_{\text{vec}}$ of Figure 9 running in the Ideal GGM setting. Let $\mathsf{Show}^* = (V', W, (Q_i)_{i\in\bar{\mathbf{I}}}, \pi), \phi^*$ be the adversary's candidate forgery. If $\mathsf{VerifyShow}(\phi^*, \mathsf{Show}^*)$ accepts, then with overwhelming probability, there exist scalars $\mu', \beta_i, s_i \in \mathbb{Z}_p$ such that*

$$P = \sum_{i\in\bar{\mathbf{I}}} \beta_i Y'_i - \mu' R, \qquad Q_i = s_i V' + \beta_i \mathsf{G} \quad (\textit{for all } i \in \bar{\mathbf{I}}). \tag{9}$$

The proof is an easy exercise.

From acceptance to the MAC identity. An accepting final presentation has $V'^{*} \neq 0$ and witnesses (9). Denote its hidden-attribute witnesses by s_i^*, set $s_i^* = d_i^*$ for disclosed positions, and write $a^* = x + \sum_{i\in[t]} y_i s_i^*$. Substitute the witnesses into (6). Since $Y_i' = r y_i \mathsf{G}$ and $R = r\mathsf{H}$, the β_i terms cancel; cancelling nonzero r gives

$$W^* = \mu'\mathsf{H} + a^* V'^{*}. \tag{10}$$

Write the scalar representations as $W^* = \mathcal{E}_{\mathrm{vec}}(\alpha)$ and $V'^{*} = \mathcal{E}_{\mathrm{vec}}(\beta)$. Their coefficients may depend on the transcript, but $x, r, h, (y_i)_i, (v_k)_k$ and hidden keys z_k remain indeterminates.

Coefficient comparison. For a hidden node k, comparing z_k in (10) gives

$$\alpha_{ev,k} h + \alpha_{ez,k} = a^*(\beta_{ev,k} h + \beta_{ez,k}).$$

The xh, x terms force $\beta_{ev,k} = \beta_{ez,k} = 0$, then $\alpha_{ev,k} = \alpha_{ez,k} = 0$. Comparing v_k now yields

$$\alpha_{c,k} a_k + \sum_{i\in\mathbf{L}_k} \alpha_{k,i} y_i = a^* \left(\beta_{c,k} a_k + \sum_{i\in\mathbf{L}_k} \beta_{k,i} y_i \right).$$

The x^2, xy_i terms force $\beta_{c,k} = \beta_{k,i} = 0$, followed by $\alpha_{c,k} = \alpha_{k,i} = 0$. Thus hidden nodes contribute to neither V'^{*} nor W^*.

The coefficient of r gives

$$\alpha_r h + \alpha_x x + \sum_i \alpha_{y,i} y_i = a^* \left(\beta_r h + \beta_x x + \sum_i \beta_{y,i} y_i \right).$$

Comparing xh, x^2, xy_i eliminates the β coefficients, then all α coefficients. The part independent of r, the v_k, and the hidden keys z_k satisfies

$$\alpha_g + \alpha_h h = \mu' h + a^*(\beta_g + \beta_h h).$$

Its x, xh terms imply $\beta_g = \beta_h = 0$, then $\alpha_g = 0$ and $\alpha_h = \mu'$.

For each revealed node j, comparison of v_j gives

$$\begin{aligned} &\alpha_{v,j} + \alpha_{c,j} a_j + \sum_{i\in\mathbf{L}_j} \alpha_{j,i} y_i \\ &\qquad = a^* \left(\beta_{v,j} + \beta_{c,j} a_j + \sum_{i\in\mathbf{L}_j} \beta_{j,i} y_i \right). \end{aligned}$$

Comparing $x^2, xy_i, 1, x$ gives, respectively, $\beta_{c,j} = 0$, $\beta_{j,i} = 0$, $\alpha_{v,j} = 0$, and $\alpha_{c,j} = \beta_{v,j}$. Therefore

$$V'^{*} = \sum_{j\in\mathcal{J}_R} \beta_{v,j} v_j \mathsf{G},$$

$$\begin{cases} \alpha_{c,j} s_{j,i} + \alpha_{j,i} = \beta_{v,j} s_i^*, & i \in \mathbf{L}_j, \\ \alpha_{c,j} s_{j,i} = \beta_{v,j} s_i^*, & i \notin \mathbf{L}_j. \end{cases}$$

Since $V'^{*} \neq 0$, some $j^* \in \mathcal{J}_R$ has $\beta_{v,j^*} \neq 0$. Hence $s_i^* = s_{j^*,i}$ outside $\mathbf{L}_{j^*}$. The message $(\mathbf{S}^*, \emptyset)$ is consequently a descendant of $\mathbf{M}_{j^*}$, belongs to $\mathcal{Q}$, and satisfies the disclosed policy because $s_i^* = d_i^*$ for $i \in \mathbf{I}^*$. This contradicts freshness.

Hence overall, $\Pr[\mathsf{Exp}^{\mathsf{Unforge}}_{\mathcal{A},\mathsf{DKVAC}}(\lambda) = 1]$ is negligible in the generic group and random oracle models for polynomially bounded N, t, Q, q_H and exponential p, proving Theorem 5.

E Multi-Show Unlinkability Proofs

E.1 Proof of Theorem 3

Proof. Let $\mathsf{Game} := \mathsf{Exp}^{\mathsf{Unlink}}_{\mathcal{A},\mathsf{SubsetDKVAC}}$. Let $\mathsf{failsound}$ be the event that some proof accepted inside $\mathsf{ObtainDel}$ is for a false statement for $\mathcal{R}_{\mathsf{Slss}}$ or $\mathcal{R}_{\mathsf{SDel}}$. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ run this game and output all statement-proof pairs checked by $\mathsf{ObtainDel}$ in both branches, for issuance and delegation, respectively. By the repeated-proof soundness definition in Appendix B.1, $\Pr[\mathsf{failsound}]$ is bounded by the sum of their soundness advantages and is negligible.

Let Game' be Game stopped with output 0 at the first occurrence of $\mathsf{failsound}$. We show that its view is independent of b. Write $\mathsf{ipar} = (x\mathsf{G}, y\mathsf{G})$. For every query that reaches presentation, the true issuance relation imposes $\mathsf{EV} = v\mathsf{G} \neq 0_{\mathbb{G}}$ and $C_s(y+s) = v\mathsf{G}$, forcing $v \neq 0$ and $y+s \neq 0$ for every issued attribute. Each true delegation relation imposes $\mathsf{EV}' = \mu\mathsf{EV} \neq 0_{\mathbb{G}}$, preserving a nonzero randomizer; the public subset checks preserve admissibility of the retained attributes. Together with the final check $\mathsf{EZ} = \mathsf{dkG}$, these conditions imply by induction that the obtained credential is well formed with some $v \in \mathbb{Z}_p^*$. The nonzero checks are essential: allowing $v = 0$ would let a malicious issuer distinguish zero credentials, whose presentations are $(0_{\mathbb{G}}, 0_{\mathbb{G}})$, from those with $v \neq 0$ whenever $X \neq 0_{\mathbb{G}}$. The oracle draws $\mu \xleftarrow{\$} \mathbb{Z}_p^*$ and returns $\mathsf{Show} = (V', C')$ with

$$V' = \mu V = \mu v x\mathsf{G} \quad \text{and} \quad C' = \mu \sum_{d \in \mathbf{D}} C_d = \mu v \Big(\textstyle\sum_{d \in \mathbf{D}} \frac{1}{y+d} \Big) \mathsf{G}\,.$$

Let $\rho := \mu v$. Since μ is uniform on $\mathbb{Z}_p^*$, so is ρ, independently of v. In particular, repeating a delegation object yields the same credential under deterministic $\mathsf{ObtainDel}$, but fresh μ gives this same presentation distribution even when v is fixed across queries. Let $w_{\mathbf{D}} = \Big(\sum_{d \in \mathbf{D}} \frac{1}{y+d} \Big)$. The distribution of $\mathsf{Show} = \big(\rho x\mathsf{G}, \rho w_{\mathbf{D}} \mathsf{G} \big)$ depends only on ipar and $\mathbf{D}$ and therefore is independent of b. The checks and stopping rule examine both branches without using b. Fresh randomness for each response extends this argument to all adaptive queries. Thus $\Pr[\mathsf{Game}' = 1] \leq \frac{1}{2}$. Finally, the games agree until $\mathsf{failsound}$, so

$$\begin{aligned} \Pr[\mathsf{Game} = 1] &\leq \Pr[\mathsf{Game}' = 1] + \Pr[\mathsf{failsound}] \\ &\leq \frac{1}{2} + \mathsf{Adv}^{\mathrm{Sound}}_{\mathsf{NIP}_{\mathsf{Slss}},\mathcal{C}_1}(\lambda) + \mathsf{Adv}^{\mathrm{Sound}}_{\mathsf{NIP}_{\mathsf{SDel}},\mathcal{C}_2}(\lambda). \end{aligned}$$

□

E.2 Proof of Theorem 6

Proof. We use three games. Let $\mathbf{W}_i$ denote the event that the adversary $\mathcal{A}$ wins in Game_i.

Game_0: The real game as defined in the experiment $\mathsf{Exp}^{\mathsf{Unlink}}_{\mathcal{A},\mathsf{VectorDKVAC}}$.

Game_1: Same as Game_0, but presentation proofs and their random-oracle domain are simulated by one stateful Sim from the start, as in Appendix B.1. The witness $(\mu', (\beta_i, s_i)_{i\in\bar{\mathbf{I}}})$ satisfies $\mathcal{R}_{\mathsf{VPres}}$ for the statement constructed by $\mathsf{ShowCred}$. By repeated-proof zero knowledge,

$$|\Pr[\mathbf{W}_0] - \Pr[\mathbf{W}_1]| \le \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIP}_{\mathsf{VPres}},\mathsf{Sim},\mathcal{B}}(\lambda)\,.$$

Game_2: Same as Game_1, but abort with output 0 if an issuance or delegation proof accepted inside $\mathsf{ObtainDel}$ in either branch is for a false statement. Let $\mathcal{C}_1, \mathcal{C}_2$ run Game_1 and output all checked issuance and delegation statement-proof pairs, respectively. By repeated-proof soundness,

$$|\Pr[\mathbf{W}_1] - \Pr[\mathbf{W}_2]| \le \mathsf{Adv}^{\mathrm{Sound}}_{\mathsf{NIP}_{\mathsf{VIss}},\mathcal{C}_1}(\lambda) + \mathsf{Adv}^{\mathrm{Sound}}_{\mathsf{NIP}_{\mathsf{VDel}},\mathcal{C}_2}(\lambda)\,.$$

We now show that Game_2 reveals no information about b. Fix the previous transcript, including the simulator and random-oracle states, and consider the next accepted $\mathcal{O}_{\mathsf{LoR}}$ query. Deterministic $\mathsf{ObtainDel}$ may return a credential used in earlier queries; the argument below fixes that credential and uses only fresh presentation randomness. Both branches disclose the same $\mathbf{I}, \mathbf{D}$. Since $\mathsf{H} \neq 0_{\mathbb{G}}$, the issuance equations $R = r\mathsf{H}$ and $r^{-1}R = \mathsf{H}$ fix the same nonzero r in both branches. For this r, define the unique scalars $x, (y_i)_{i\in[t]}$ by $X' = rx\mathsf{G}$ and $Y'_i = ry_i\mathsf{G}$. These scalars are fixed by ipar, although they are not part of the issuance witness. At every link, $\mathsf{ObtainDel}$ also checks $\mathbf{M}_{j+1} \preceq \mathbf{M}_j$ publicly. For $\mathbf{M}_j = (\mathbf{S}_j, \mathbf{L}_j)$, this check supplies $\mathbf{L}_{j+1} \subseteq \mathbf{L}_j$ and $s_{j+1,i} = s_{j,i}$ for every $i \notin \mathbf{L}_j$; $\mathcal{R}_{\mathsf{VDel}}$ alone does not enforce these restrictions. Applying the true issuance and delegation relations along the chain, together with these public checks and the final decryption checks, gives $V = v\mathsf{G}$ and $C = v(x + \sum_{i\in[t]} y_i s_i)\mathsf{G}$ for some $v \in \mathbb{Z}_p^*$. The challenger returns

$$\mathsf{Show} = (V', W, (Q_i)_{i\in\bar{\mathbf{I}}}, \pi),\ V' = \mu v\mathsf{G},\quad W = \mu C + \mu'\mathsf{H},\quad Q_i = s_i V' + \beta_i\mathsf{G},$$

with π simulated from the statement $\mathbb{x} = (\mathsf{ipar}, V', P, (Q_i)_{i\in\bar{\mathbf{I}}})$. From the verification identity, $P = r\big(xV' + \sum_{i\in\bar{\mathbf{I}}} y_i Q_i - W + \sum_{i\in\mathbf{I}} y_i s_i V'\big)$. For the fixed history, ipar and $\mathbf{D}$, the statement is determined by $(V', W, (Q_i)_{i\in\bar{\mathbf{I}}})$. Thus, if this tuple is independent of b, so are the statement and its simulated proof. Set $\rho := \mu v$. Since μ is uniform on $\mathbb{Z}_p^*$ and $v \neq 0$, so is ρ, making $V' = \rho\mathsf{G}$ independent of b. The scalars $\mu', (\beta_i)_{i\in\bar{\mathbf{I}}}$ are sampled independently and uniformly from $\mathbb{Z}_p$ by $\mathsf{ShowCred}$. Since $\mathsf{H}, \mathsf{G} \neq 0_{\mathbb{G}}$, conditioned on V' the masks $\mu'\mathsf{H}, \beta_i\mathsf{G}$ make $W, (Q_i)_{i\in\bar{\mathbf{I}}}$ jointly uniform irrespective of b. The checks and aborts examine both branches without using b. The same argument applies after every query, including the simulator's oracle answers and possible failure. Since an abort returns 0,

we obtain $\Pr[\mathbf{W}_2] \leq \frac{1}{2}$. Finally,

$$\begin{aligned}\Pr\left[\mathsf{Exp}^{\mathsf{Unlink}}_{\mathcal{A},\mathsf{VectorDKVAC}}(\lambda) = 1\right] &= \Pr[\mathbf{W}_0] \\ &\leq \Pr[\mathbf{W}_2] + |\Pr[\mathbf{W}_0] - \Pr[\mathbf{W}_1]| + |\Pr[\mathbf{W}_1] - \Pr[\mathbf{W}_2]| \\ &\leq \frac{1}{2} + \mathsf{Adv}^{\mathrm{Sound}}_{\mathsf{NIP}_{\mathsf{VIss}},\mathcal{C}_1}(\lambda) + \mathsf{Adv}^{\mathrm{Sound}}_{\mathsf{NIP}_{\mathsf{VDel}},\mathcal{C}_2}(\lambda) \\ &\qquad + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIP}_{\mathsf{VPres}},\mathsf{Sim},\mathcal{B}}(\lambda)\,,\end{aligned}$$

which is $\frac{1}{2} + \mathsf{negl}(\lambda)$, matching the unlinkability definition. □