Keyed-Verification Anonymous Credentials with Highly Efficient Partial Disclosure

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

[1] COSIC, KU Leuven, Leuven, Belgium
[2] University of Edinburgh, Edinburgh, UK
[3] Nokia Bell Labs, Antwerp, Belgium

Abstract.
An anonymous credential (AC) system with partial disclosure allows users to prove possession of a credential issued by an issuer while selectively disclosing a subset of their attributes to a verifier in a privacy-preserving manner. In keyed-verification AC (KVAC) systems, the issuer and verifier share a secret key. Existing KVAC schemes rely on computationally expensive zero-knowledge proofs during credential presentation, with the presentation size growing linearly with the number of attributes. In this work, we propose two highly efficient KVAC constructions that eliminate the need for zero-knowledge proofs during the credential presentation and achieve constant-size presentations.

Our first construction adapts the approach of Fuchsbauer, Hanser and Slamanig (JoC'19), which achieved constant-size credential presentation in a publicly verifiable setting using their proposed structure-preserving signatures on equivalence classes (SPS-EQ) and set commitment schemes, to the KVAC setting. We introduce structure-preserving message authentication codes on equivalence classes (SP-MAC-EQ) and designated-verifier set commitments (DVSC), resulting in a KVAC system with constant-size credentials (2 group elements) and presentations (5 group elements). To avoid the bilinear groups and pairing operations required by SP-MAC-EQ, our second construction uses a homomorphic MAC with a simplified DVSC. While this sacrifices constant-size credentials ($n + 2$ group elements, where n is the number of attributes), it retains constant-size presentations (2 group elements) in a pairingless setting.

We formally prove the security of both constructions and provide open-source implementation results demonstrating their practicality. We extensively benchmarked our KVAC protocols and, additionally, bechmarked the efficiency of our SP-MAC-EQ scheme against the original SPS-EQ scheme, showcasing significant performance improvements.

Keywords: Keyed-Verification Anonymous Credentials · Structure-Preserving MAC on Equivalence Classes · Designated-Verifier Set Commitment

1 Introduction

An anonymous credential system (AC) [Cha82, CL01] allows issuers to issue credentials to users. A credential attests to a set of attributes, encoding properties of the user (such

E-mail: omid.mirzamohammadi@esat.kuleuven.be (Omid Mirzamohammadi), jan.bobolz@ed.ac.uk (Jan Bobolz), ssedagha@esat.kuleuven.be (Mahdi Sedaghat), emad.heydaribeni@kuleuven.be (Emad Heydari Beni), aabidin@esat.kuleuven.be (Aysajan Abidin), dave.singelee@esat.kuleuven.be (Dave Singelée), bart.preneel@esat.kuleuven.be (Bart Preneel)

Date of this document: 2025-10-19.

as name, address, job, ...) or access control information (such as "user has access to the main building"). The user can disclose some of his attributes to a verifier and demonstrate that he is in possession of a credential attesting to those attributes, without revealing his other attributes. For instance, if a verifier needs to check whether a user is a janitor and that he has access to the main building, these protocols allow the user to unlinkably prove possession of these specific attributes without revealing his other attributes, like his name, address, or what other buildings he may access. In a *keyed-verification* anonymous credential system (KVAC) [CMZ14], the issuer and verifier share the same secret key. This is a restriction compared to general (publicly verifiable) anonymous credentials, where the issuer does not have to trust verifiers. However, in many scenarios, it is reasonable to assume that issuers trust verifiers. Oftentimes, they are even the same entity. For instance, a university managing access to its buildings would be both the issuer and the verifier. If the scenario admits using KVAC, their significantly better performance makes them preferable over publicly verifiable anonymous credentials.

While an anonymous credential is typically a (zero-knowledge friendly) digital signature on the user's attributes, prior work on KVAC replaces the signature with a (zero-knowledge friendly) MAC tag. This change introduces new challenges. For example, the user cannot easily check that the received MAC tag is valid, leading to potential privacy issues if the issuer can use different MAC keys for different users undetected. Furthermore, to present a credential in the digital signature setting, the user can simply create a zero-knowledge proof of knowledge of his valid signature on certain (partially hidden) attributes. If the user only holds a MAC tag instead of a signature, his inability to check the MAC tag prevents him from creating such a proof in a straightforward way.

Prior work [CMZ14, CPZ20, BBDT16, CR19, CDDH19] has found elegant solutions to these challenges, with the resulting MAC-based KVACs being significantly more efficient than their signature-based equivalents. When it comes to presenting a credential, these solutions still involve zero-knowledge proofs in some form to hide attributes from the verifier. On the one hand, this structure is extensible: in principle, it supports more powerful access policies than partial disclosure of attributes. On the other hand, these zero-knowledge proofs tend to account for the lion's share of the verification cost with respect to both communication complexity and computation cost.[1]
In this paper, we ask the following question.

Can one design efficient KVAC constructions with selective disclosure where presentation does not rely on expensive zero-knowledge proofs?

We answer this question affirmatively by providing two constructions with highly efficient selective disclosure: one in the pairing setting with constant-size credentials, and one without pairings with linear-size credentials. Both constructions compare very favorably to state of the art KVAC constructions, see Table 1.

Our $\mathsf{KVAC}_{\mathsf{MEQ}}$ construction from SP-MAC-EQ. Our approach for avoiding costly zero-knowledge proofs during credential presentation takes heavy inspiration from the work of Fuchsbauer, Hanser, and Slamanig on constant-size anonymous credentials [FHS19]. Their construction of (publicly verifiable) anonymous credentials is built around *structure-preserving signatures on equivalence classes* (SPS-EQ) [FHS19] and suitable *set commitments* [Ngu05]. An SPS-EQ is a signature σ on messages $(M_1, \ldots, M_\ell) \in (\mathbb{G}_1^*)^\ell$ that can be efficiently adapted (without the secret key) to a signature σ' on the message $(\mu M_1, \ldots, \mu M_\ell)$ for any $\mu \in \mathbb{Z}_p^*$. A set commitment C to a set $\mathbf{S}$ has partial opening capabilities, i.e. there are short (constant size) witnesses $W_{\mathbf{D}}$ that attest to $\mathbf{D} \subseteq \mathbf{S}$. Roughly

[1] Using techniques such as modern SNARKs or compressed Sigma protocols [AC20], one can drastically reduce communication complexity, but only at the price of significantly increased concrete user computation cost or significantly increased verifier computation cost, respectively.

Table 1: Attribute-based multi-show unlinkable anonymous credential schemes. n denotes the number of attributes possessed by a user. The bit length of groups $\mathbb{G}$, $\mathbb{G}_1$ and $\mathbb{G}_2$ and scalars are denoted by $|\mathbb{G}|$, $|\mathbb{G}_1|$, $|\mathbb{G}_2|$ and $|\mathbb{Z}_p|$, respectively. SetCom stands for set commitment, O-DVNIZK stands for oblivious designated verifier non-interactive zero-knowledge. SCDHI stands for Strong Computational Diffie-Hellman Inversion Problem and se-DL stands for short-exponent discrete logarithm assumption. GGM and AGM stand for Generic Group Model and Algebraic Group Model, respectively. Statistical anonymity refers to anonymity holding unconditionally (no assumption). All constructions except [CR19, FHS19] rely on the random oracle model, which we do not list explicitly.

Scheme	**Pairing-less**	**Credential size**	**Presentation size**	**Security** (Unforgeability/ Anonymity)
SPS-EQ + SetCom [FHS19]†	✗	$2\|\mathbb{G}_1\| + 1\|\mathbb{G}_2\|$ $+1\|\mathbb{Z}_p\|$	$\geq 6\|\mathbb{G}_1\| + 1\|\mathbb{G}_2\|$ $+3\|\mathbb{Z}_p\|$	GGM/ DDH
$\mathsf{MAC}_{\mathrm{GGM}}$ + Schnorr [CMZ14]	✓	$2\|\mathbb{G}\|$	$(n+2)\|\mathbb{G}\|$ $+(2n+2)\|\mathbb{Z}_p\|$	GGM/ DDH
$\mathsf{MAC}_{\mathrm{BBS}}$ + Schnorr [BBDT16]	✓	$2\|\mathbb{G}\| + 2\|\mathbb{Z}_p\|$	$3\|\mathbb{G}\|$ $+(n+7)\|\mathbb{Z}_p\|$	q-SDH/ Statistical
$\mathsf{MAC}_{\mathrm{wBB}}$ + Optimized Schnorr [CDDH19]	✓	$(n+1)\|\mathbb{G}\|$	$\leq 2\|\mathbb{G}\|$ $+(n+1)\|\mathbb{Z}_p\|$	n-SCDHI/ Statistical
$\mathsf{MAC}_{\mathrm{GGM}}$ + O-DVNIZK [CR19]‡	✓	$2n\|\mathbb{G}_{pq}\|$	$(n+2)\|\mathbb{G}_{pq}\|$	GGM + IND-CPA + se-DL/Statistical
μCMZ + Schnorr [Orr24]	✓	$2\|\mathbb{G}\|$	$(n+2)\|\mathbb{G}\|$ $+(2n+2)\|\mathbb{Z}_p\|$	AGM + 3-DL/ Statistical
μBBS + Schnorr [Orr24]	✓	$1\|\mathbb{G}\| + 1\|\mathbb{Z}_p\|$	$2\|\mathbb{G}\|$ $+(n+4)\|\mathbb{Z}_p\|$	AGM + q-DL/ Statistical
SP-MAC-EQ + DVSC (Figure 1)	✗	$1\|\mathbb{G}_1\| + 1\|\mathbb{G}_2\|$	$4\|\mathbb{G}_1\| + 1\|\mathbb{G}_2\|$	GGM/ DDH
Pairingless construction (Figure 2)	✓	$(n+2)\|\mathbb{G}\|$	$2\|\mathbb{G}\|$	GGM/ Statistical

† This scheme is a *publicly verifiable* anonymous credential system.
‡ This scheme requires a large-order group, where the order must match the plaintext space of a DVNIZK-friendly encryption scheme.

speaking, a credential in [FHS19] is a signature σ on a set commitment C to the user's attributes $\mathbf{S}$. To present a credential disclosing attributes $\mathbf{D}$ but hiding attributes $\mathbf{S} \setminus \mathbf{D}$, the user sends (1) a randomized version μC of his set commitment, (2) an adapted SPS-EQ to match μC, effectively proving his randomized set commitment is valid, and (3) a partial opening witness $\mu W_{\mathbf{D}}$ showing that μC opens to some hidden $\mathbf{S}$ with $\mathbf{D} \subseteq \mathbf{S}$. This process provides partial disclosure guarantees without expensive zero-knowledge proofs.[2] Indeed, the protocol's communication cost is constant, independent of the number of attributes.

While [FHS19] is quite efficient as-is, it has been designed with public verification in mind and has not been considered for the keyed verification scenario before. For our $\mathsf{KVAC}_{\mathsf{MEQ}}$ construction, we adapt the approach above to the KVAC setting. For this, we replace both building blocks with keyed-verification equivalents. To replace the SPS-EQ, we define and construct structure-preserving *MACs* on equivalence classes (SP-MAC-EQ), the MAC equivalent to SPS-EQ, which may be of independent interest (e.g., to replace SPS-EQ in [BEK+20]). Our SP-MAC-EQ construction is based on the SPS-EQ in [FHS19], but through careful optimizations, our SP-MAC-EQ consists of only two group elements (down from three for SPS-EQ) and verification only requires two pairing operations (compared to $\ell + 3$ pairings to verify an SPS-EQ on ℓ messages, cf. Table 5). To replace the set

[2]For technical reasons, the original construction [FHS19] does actually employ a small zero-knowledge proof for credential presentation. However, that proof does not include statements about the user's attributes, making it constant-size and practically efficient.

commitment scheme, we define a *designated verifier* set commitment scheme (DVSC). A DVSC is easily constructed based on an existing set commitment [Ngu05], that we adapt to the designated-verifier setting. In the resulting DVSC, the verifier can check (a partial opening of) the set commitment much more efficiently, without any pairing operations.

Using those two building blocks, SP-MAC-EQ and DVSC, we construct our $\mathsf{KVAC}_{\mathsf{MEQ}}$ similarly to the [FHS19] AC template described above. With some details omitted, this means that $\mathsf{KVAC}_{\mathsf{MEQ}}$ uses a set commitment scheme [Ngu05] to commit to attributes $\mathbf{S}$ as $C = (f_{\mathbf{S}}(v)\mathsf{G}_1, \mathsf{G}'_1)$, where $f_{\mathbf{S}}(v) = \prod_{s\in\mathbf{S}}(v-s)$ and $v \in \mathbb{Z}_p$ is hidden from users. Users can compute commitments using values $V_j = (v^j\mathsf{G}_1)_{j=0}^t$ published by the issuer. A credential is an SP-MAC-EQ tag τ on C. To present a credential, disclosing attributes $\mathbf{D} \subseteq \mathbf{S}$, the user randomizes his set commitment C to μC for random $\mu \xleftarrow{\$} \mathbb{Z}_p^*$, which hides its contents. The user then adapts the tag τ to $\mu\tau$ accordingly (to authenticate μC instead of C), and sends the commitment μC, the MAC $\mu\tau$, and the subset witness $\mu W_{\mathbf{D}} = \mu f_{\mathbf{S}\setminus\mathbf{D}}(v)\mathsf{G}_1$ to the verifier. The verifier checks the MAC $\mu\tau$ and the subset witness $\mu W_{\mathbf{D}}$ against the randomized commitment μC. Randomization with μ provides privacy and unlinkability, the unforgeability of SP-MAC-EQ ensures that the user cannot use a different commitment C' to $\mathbf{S}' \neq \mathbf{S}$, and security of the set commitment ensures the (randomized) commitment cannot be opened to any $\mathbf{D}' \not\subseteq \mathbf{S}$.

We solve the challenges arising from losing public verifiability using techniques from prior work, adapted to the new SP-MAC-EQ construction: when it comes to credential issuance, because the user cannot locally verify his credential (MAC), the issuer needs to prove MAC validity with respect to a public commitment to her MAC key (ensuring what is often called *key-parameter consistency* [CMZ14]). This can be achieved using a simple constant-size Schnorr-like proof. When it comes to credential presentation, the user's inability to zero-knowledge prove the validity of his MAC tag is inherently not an issue in our construction: we do not rely on zero-knowledge proofs for presentation.

The resulting construction $\mathsf{KVAC}_{\mathsf{MEQ}}$ is pairing-based (though the number of pairing computations has been minimized to two per credential presentation, independent of the number of attributes) that is significantly more efficient than its parent AC scheme [FHS19] and compares favorably to existing KVAC schemes: presenting a credential only requires the user send four group elements to the verifier. This is in contrast to earlier KVAC constructions that depend on expensive zero-knowledge proofs, whose communication complexity scales with the number of (hidden) attributes. See Table 1 for a detailed comparison. Because of details in our definitions and security proofs, we also significantly simplify the construction compared to its parent AC scheme [FHS19] (e.g., no zero-knowledge proof during presentation at all, and the authenticated message consists of only two group elements rather than the original three). We formally prove our SP-MAC-EQ, DVSC, and KVAC constructions secure in the generic group model (GGM) and random oracle model (ROM).[3] Anonymity guarantees hold under the decisional Diffie-Hellman assumption in the ROM.

Our $\mathsf{KVAC}_{\mathsf{GGM}}$ construction without pairings. Our $\mathsf{KVAC}_{\mathsf{MEQ}}$ construction from SP-MAC-EQ unfortunately requires a bilinear group. Our intuition is that this seems to be an inherent requirement of SP-MAC-EQ. On the one hand, SP-MAC-EQ needs to enable deriving tags on multiples $\mu\mathbf{M}$ of the authenticated message $\mathbf{M}$. On the other hand, SP-MAC-EQ must protect against combining tags on different messages (one must not be able to, say, derive a tag on $\mathbf{M} + \mathbf{M}'$ given tags on $\mathbf{M}$ and $\mathbf{M}'$). The latter requirement means that verification must have some non-linear component. Pairings seem to be the only "natural" means to achieve this in a way compatible with message randomization (the

[3] While a proof in the more realistic algebraic group model (AGM) would be preferable to a GGM proof, the underlying SPS-EQ construction [FHS19] has only recently been proven secure in the AGM [BFR25], which involves a rather complex proof. For simplicity, we provide a proof in the GGM, leaving an AGM-based proof as a direction for future work.

first requirement) and tag randomization. For example, in our SP-MAC-EQ construction, where MAC tags are of the form $(a\left(\sum x_i M_i\right), a^{-1}\mathsf{G}_2)$ for a per-tag random $a \in \mathbb{Z}_p^*$, this verification non-linearity is provided by the pairing operation canceling out the a from the first component with the a^{-1} in the second. This structure ensures that terms $a(\sum x_i M_i)$ and $a'(\sum x_i M_i')$ from different tags cannot be combined meaningfully.

Our second construction builds on the observation that we do not necessarily need the strict no-recombination guarantees of SP-MAC-EQ. In our first construction, the SP-MAC-EQ authenticates a set commitment C. What if, instead of relying on SP-MAC-EQ to prevent recombination of set commitments, we *allow* linear recombination of MACs, while ensuring that the set commitments cannot be meaningfully recombined? Following this idea, we replace SP-MAC-EQ with a *homomorphic* MAC, which explicitly allows homomorphically combining MACs on different messages. Freed from no-recombination requirements, it is exceedingly easy to construct such a homomorphic MAC without pairings: the tag on C is simply xC, where x is the MAC secret key. Of course, this means that MACs xC and xC' can easily be summed up to a valid MAC $x(C + C')$ on $C + C'$, resulting in much weaker unforgeability guarantees than from SP-MAC-EQ. To deal with this, we slightly tweak the set commitment C. We give each individual commitment a different random base $y\mathsf{G}$, i.e. we set $C = f_{\mathbf{S}}(v)y\mathsf{G}$ for $y \xleftarrow{\$} \mathbb{Z}_p^*$. This ensures that set commitments cannot be meaningfully linearly combined. Hence, even if the MAC allows adversarial users to compute MACs on linear combinations of their set commitments C, those combinations are (likely) not valid set commitments, rendering this ability useless and enabling us to prove unforgeability w.r.t. committed attributes. The downside of this idea is that in order to compute subset witnesses $W_{\mathbf{D}} = f_{\mathbf{S}\setminus\mathbf{D}}(v)y\mathsf{G}$, the user needs to know the "powers of v" w.r.t. his specific random base $y\mathsf{G}$. More specifically, a user's credential must also contain $v^j y\mathsf{G}$ for $0 \leq j \leq n$, where n is the number of attributes. This increases the size of the user's credentials compared to our $\mathsf{KVAC}_{\mathsf{MEQ}}$ construction, where in the latter, all commitments are to the base G_1 and the $v^j\mathsf{G}_1$ terms are universal for all credentials.

The resulting $\mathsf{KVAC}_{\mathsf{GGM}}$ construction derived from this idea is very simple: a credential consists of a set commitment $C = f_{\mathbf{S}}(v)y\mathsf{G}$, a tag $\tau = xC$ on C, and the terms $(v^i y\mathsf{G})_{i=0}^n$. To present a credential, disclosing attributes $\mathbf{D} \subseteq \mathbf{S}$, the user simply randomizes his set commitment C to μC for random $\mu \xleftarrow{\$} \mathbb{Z}_p^*$, adapts the tag τ to $\mu\tau$ appropriately, and sends subset witness $\mu W_{\mathbf{D}}$ (computed using $v^i y\mathsf{G}$ as described above) and tag $\mu\tau$ to the verifier. The verifier computes the unique μC for which $\mu W_{\mathbf{D}}$ is a valid witness and checks the tag $\mu\tau$. Randomization provides privacy and the user's ignorance of v, x ensures unforgeability. Presentation involves sending only two group elements (of a non-pairing group) and the verifier's check boils down to a single exponentiation. This makes the construction the most efficient in terms of presentation communication and verification cost *by far*, with the trade-off of larger (but practically reasonable) credentials. See Table 1 for comparison with other constructions and see Section 8 for performance measurements.

We formally prove our $\mathsf{KVAC}_{\mathsf{GGM}}$ construction secure in the GGM and ROM. Notably, anonymity holds statistically/unconditionally, which means that anonymity is preserved even against possible future quantum adversaries.

Summary of our contributions. In this paper, we make the following contributions.

Introducing SP-MAC-EQ. In Section 3, we formally define structure-preserving MACs on equivalence classes (SP-MAC-EQ) and their security properties as a natural adaptation of their signature equivalent [FHS19]. We give a construction based on a well-known SPS-EQ scheme [FHS19], optimizing for the keyed verification case. We formally prove it secure.

Constructing $\mathsf{KVAC}_{\mathsf{MEQ}}$ from SP-MAC-EQ and DVSC. In Section 5, we construct a keyed-verification anonymous credential system (KVAC) with partial disclosure based

on (1) our SP-MAC-EQ construction and (2) a designated verifier set commitment scheme (which is based on [Ngu05], suitably adapted to our keyed-verification scenario, see Section 4). We formally prove that our construction is secure.

Constructing $\mathsf{KVAC}_{\mathsf{GGM}}$. In Section 6, we construct another KVAC, making white-box use of techniques related to homomorphic MAC algorithms and randomized DVSC. We formally prove our construction secure.

Extension sketches. In Section 7, we sketch how to extend our constructions to support blind issuance or non-transferability, which are desirable in some contexts.

Implementation and benchmarking. Section 8 presents the results of our performance evaluation. We have implemented our two KVAC constructions and tested their performance, demonstrating that our constructions are highly practical. Furthermore, we implemented and benchmarked SPS-EQ and SP-MAC-EQ. All implementations are open-source [Ben25].

Applications and discussion. We discuss additional applications that may benefit from our new constructions. Consider, for example, implementing bus tickets with anonymous credentials. One may buy a ticket on an app or website, and verify it by showing a QR code on the bus. We can imagine that the user gets one attribute for every day of validity, and one attribute for every travel zone the ticket is valid for. Partial disclosure allows the user to unlinkably present a ticket showing possession of attributes $\mathbf{D} = \{\mathit{today}, \mathsf{Zone\ A}\}$ (for the appropriate value of *today*), while hiding his other attributes (namely the ticket's validity period and what other zones it is valid for). It is easy to imagine that users shall be able to buy tickets on the bus, hence the secret key is already present in the on-bus ticketing machine. This makes KVAC the ideal choice for this application, using the secret key for ticket verification in addition to ticket issuance.

If ticket presentation happens via a QR code, then presentation size is crucial. While QR codes can theoretically hold up to 2,953 bytes of data, they become noticeably harder to scan the more data they contain. Hence smaller QR codes are vastly preferable in an environment where smooth operation depends on the ability to scan QR codes quickly. Our pairingless construction $\mathsf{KVAC}_{\mathsf{GGM}}$ has a presentation size of only 64 bytes (using secp256k1), which is significantly smaller than the presentation size of prior constructions, even for scenarios that involve only a single attribute (see Table 1). For scenarios like the bus ticketing above (with about 30 attributes for 30-day tickets), the difference is even more pronounced and may easily be the deciding factor between practical feasibility and infeasibilty. Verification performance is extremely lightweight for $\mathsf{KVAC}_{\mathsf{GGM}}$, involving only a single group exponentiation, suitable for embedded hardware.

The downside of $\mathsf{KVAC}_{\mathsf{GGM}}$ is that the credentials are of linear size in the number of attributes. This is not a problem in many applications (such as the one above), where the credential is stored on a phone or similar device with sufficient storage. Still, for scenarios where user storage is limited (e.g., smart cards), one may prefer our pairing-based construction $\mathsf{KVAC}_{\mathsf{MEQ}}$, which has constant-size credentials and constant-size presentations (about 286 bytes using BLS12-381).

In current practice, keyed-verification anonymous credentials are already used in applications such as Signal [CPZ20] and Tor [TG23]. There are lively discussions about the European Digital Identity (EUDI), which aims to implement anonymous-credential-like digital identity in practice by 2026. Due to technical limitations in current consumer hardware and a lack of standards when it comes to pairing groups, KVAC are discussed as the central building block for EUDI [DDKT25]. In this context, server-aided anonymous credentials (SAAC) [CAHLT25] are also discussed to make KVAC publicly verifiable given some semi-trusted helper entity. While our constructions are incompatible with the

current SAAC approach [CAHLT25], perhaps similar techniques can be applied to our constructions upon further research.

1.1 Related Work

Anonymous credential (AC) systems. The "ACS protocol" developed by Meta[4], "Idemix" developed by IBM[5] and "U-Prove" developed by Microsoft[6] are some recent open-sourced AC systems. However, there are many methods for designing an AC system, the most predominant class of them is built on re-randomizable signatures [CL03, CL04, BL13, LMPY16, PS16, CKP+23], and related approaches such as equivalence class signatures [HS14, FHS19, CL19, PM23, HS21, CLPK22, BF20, BSW24] or redactable signatures [CDHK15, San20]. In this approach, the credential is essentially a signature on the list of possessed attributes for each user from a certain issuer. The credentials can be used to prove some facts to any third-party verifier, which highlights the importance of public verifiability in these systems. However, this property usually comes with large communication and computation overhead.

Keyed-Verification Anonymous Credentials (KVAC). KVAC were introduced in [CMZ14] (CCS'14) to achieve a better efficiency when the issuer and verifier are the same entity by replacing digital signatures with symmetric-key primitives. They proposed the notion of algebraic Message Authentication Codes (MACs) based on group operations instead of non-algebraic methods such as block ciphers or hash functions. They proposed two algebraic MACs: $\mathsf{MAC}_{\mathrm{GGM}}$, secure in the Generic Group Model (GGM), and $\mathsf{MAC}_{\mathrm{DDH}}$, based on the DDH assumption. The authors then design an efficient KVAC using these algebraic MACs. Signal later adopted a variation of this KVAC [CPZ20] for private group systems.[7] However, their KVAC had some limitations; the presentation proof grows linearly with the number of unrevealed attributes in group elements. Moreover, their system fails to achieve perfect anonymity during credential blind issuance as ElGamal encryption is used to hide attributes.

As a subsequent work, [BBDT16] (SAC'16) proposed a new KVAC based on a novel algebraic MAC from a pairing-free variant of BBS signatures [BBS04], $\mathsf{MAC}_{\mathrm{BBS}}$ in short, that improved upon [CMZ14]. Their presentation proof remained constant in group elements while being linear in scalar numbers.

Couteau and Reichle [CR19] (PKC'19) proposed a KVAC in the standard model, offering stronger guarantees, however, this comes at the cost of efficiency, as the presentation phase requires $2n + 3$ exponentiations in a 2048-bit group, making it less efficient compared to other schemes. [CDDH19] proposed an efficient KVAC designed specifically for lightweight devices, such as smart cards, by leveraging a novel algebraic MAC based on Boneh-Boyen signatures [BB08], $\mathsf{MAC}_{\mathrm{wBB}}$ in short.

In a 2024 preprint [Orr24], Orrú revisits the notion of KVAC systems by providing a comprehensive security analysis along with some efficient constructions upon the prior works [CMZ14] and [BBDT16]. It improves the underlying MAC constructions with tight security proofs in the Algebraic Group Model (AGM) and proposes two efficient schemes, namely μCMZ and μBBS, to address their prior designs' limitations. μCMZ achieved statistical anonymity and reduced issuance costs from $2n + 1$ group elements to a single group element for n private attributes. μBBS improved [BBDT16] by reducing presentation and MAC costs while aligning with standardization efforts. μCMZ proved more efficient in credential issuance, while μBBS showed better presentation efficiency when $n > 1$. The author also developed lightweight anonymous credentials from these

[4] https://github.com/facebookresearch/acs
[5] https://github.com/IBM/idemix
[6] https://www.microsoft.com/en-us/research/project/u-prove/
[7] https://signal.org/blog/signal-private-group-system/

constructions, trading weaker unforgeability, namely one-more unforgeability instead of standard unforgeability, for better performance. Table 1 compares these KVAC with ours.

Very recently, [CAHLT25] introduced the notion of server-aided anonymous credentials (SAACs), which enable publicly verifiable and multi-use anonymous credentials with the help of auxiliary information generated by a semi-trusted helper. Their work provides the first formal definition of SAACs and presents efficient pairing-free constructions based on KVACs and specialized oblivious issuance protocols. This line of research highlights the importance of pairing-free KVACs.

Structure-preserving signatures on equivalence classes (SPS-EQ). SPS-EQ were proposed by Hanser and Slamanig in [HS14] (AC'14), later extended by Fuchsbauer, Hanser and Slamanig in [FHS19] (JoC'19), FHS19 in short. SPS-EQ enables a controlled form of malleability of both message and signature, and it is possible to validate a signature without depending on complex NIZK proofs. Although the initial work presented an efficient SPS-EQ scheme with signatures composed of only three group elements, subsequent research primarily focused on proposing constructions based on falsifiable assumptions in the standard model as the original work's unforgeability is proved in the GGM. Fuchsbauer and Gay in [FG18] (PKC'18) proposed the first SPS-EQ based on falsifiable assumptions, under the hardness of Matrix-Diffie-Hellman (MDDH) assumptions. However, it achieves a weaker security notion: existential unforgeability against chosen open message attacks (EUF-CoMA). More precisely, the adversary must query the signing oracle with the discrete logarithm of the queried message vector. Also as shown in [KSD19], the adaptation property of this construction relies on the assumption of an honest signer (i.e., credential issuer), which limits its applications. [KSD19] (AC'19) and [CLPK22] (PKC'22) proposed EUF-CMA secure SPS-EQ based on standard assumptions. However, recently [BFR24b] (AC'24) identified a gap in the security proofs of both existing SPS-EQ schemes in the standard model, namely [KSD19] and [CLPK22]. The same authors in [BFR24a] (PKC'24) found an impossibility result, showing that it is not possible to construct SPS-EQ schemes secure under standard assumptions using standard techniques.

Structure-preserving message authentication codes on equivalence classes (SP-MAC-EQ). SP-MAC-EQ was first mentioned by Fuchsbauer and Gay [FG18], where they informally suggested constructing an SP-MAC-EQ based on an affine MAC scheme [BKP14]. However, they observed that under the standard MDDH assumption, their construction fails to achieve the standard notion of unforgeability and instead satisfies a weaker notion, similar to EUF-CoMA, as the challenger without knowing the dlog of the queried message cannot simulate the oracles with MDDH instances. To address this, they proposed an alternative where the tag is defined as a target group element, which resolves the issue. However, this approach violates the structure-preserving property since the tag is no longer in the source group, and it also increases the tag size by a factor of 10, thereby limiting its applications. Due to these limitations, it is not trivial how their proposed SP-MAC-EQ can serve as an efficient building block for more complex systems, such as KVAC. Therefore, to the best of our knowledge, no SP-MAC-EQ with a formal definition with general-purpose applicability currently exists. This primitive could be of independent interest.

2 Preliminaries

Notations. We denote the security parameter by λ and use 1^λ as its unary representation. We call a randomized algorithm $\mathcal{A}$ *probabilistic polynomial time* (PPT) or *efficient* if there exists a polynomial $p(\cdot)$ s.t. for every input x the running time of $\mathcal{A}(x)$ is bounded by $p(|x|)$. A function $\mathsf{negl}(\lambda)$ is called *negligible* if for every positive polynomial $p(\lambda)$, there exists λ_0 s.t. for all $n > n_0 \colon \mathsf{negl}(n) < 1/p(n)$. If clear from the context, we sometimes omit n for improved readability. The set $\{1, \ldots, n\}$ is denoted as $[n]$ for a positive integer

n. We use "==" to denote an equality check: it returns 1 if the two operands are equal, and 0 otherwise. Conversely, "! =" returns 1 if the operands are different, and 0 otherwise. The assign operator is denoted with ":=", whereas randomized assignment is denoted with $a \leftarrow A$, with a randomized algorithm A and where the randomness is not explicit. If the randomness is explicit, we write $a := A(x; r)$ where x is the input and r is the randomness. $[A(x)] = \{A(x; r) \mid r \in \{0,1\}^*\}$ denotes the set of all possible outputs of A. $m \xleftarrow{\$} \mathcal{M}$ shows randomly sampling a value m from a space $\mathcal{M}$. For algorithms $\mathcal{A}$ and $\mathcal{B}$, we write $\mathcal{A}^{\mathcal{B}(\cdot)}(x)$ to denote that $\mathcal{A}$ gets x as an input and has black-box oracle access to $\mathcal{B}$, that is, the response for an oracle query (q, r) is $\mathcal{B}(q; r)$. The expression $\mathsf{view}_{\mathcal{A}}$ for an algorithm $\mathcal{A}$ refers to the list of all inputs $\mathcal{A}$ has received, the randomness used by $\mathcal{A}$, and, if $\mathcal{A}$ has oracle access to some oracle $\mathcal{O}$, then the outputs of oracle queries. Essentially, $\mathsf{view}_{\mathcal{A}}$ contains all the information needed to deterministically retrace the exact computation steps that $\mathcal{A}$ makes (enabling rewinding). For a set $\mathbf{S} \subseteq \mathbb{Z}_p$, we denote by $f_{\mathbf{S}}$ the polynomial $f_{\mathbf{S}}(X) = \prod_{s \in \mathbf{S}}(X - s) \in \mathbb{Z}_p[X]$. We note that given $(v^i \mathsf{G})_{j=0}^n$, one can efficiently compute $f(v)\mathsf{G}$ for any $f \in \mathbb{Z}_p[X]$ of degree at most n.

For any group $\mathbb{G}$, we denote the set of all non-neutral elements by $\mathbb{G}^* = \mathbb{G} \setminus \{0\}$. We generally use additive group notation. We use a type-3 bilinear group [GPS08], $\mathsf{BG} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathrm{T}, p, e, \mathsf{G}_1, \mathsf{G}_2)$, generated by a group parameter generator $\mathcal{BG}(1^\lambda)$ such that $p > 2^\lambda$. We require that the group order $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_\mathrm{T}| = p$ is a prime number, $\mathsf{G}_1 \in \mathbb{G}_1^*, \mathsf{G}_2 \in \mathbb{G}_2^*$ are generators, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\mathrm{T}$ is a non-degenerate bilinear map, i.e. $e(\mathsf{G}_1, \mathsf{G}_2) \neq 1$ and $e(a\mathsf{G}_1, b\mathsf{G}_2) = e(\mathsf{G}_1, \mathsf{G}_2)^{ab}$ for all $a, b \in \mathbb{Z}$. Note that we write $\mathbb{G}_\mathrm{T}$ multiplicatively.

Camenisch and Stadler Notation. We use the standard notation introduced by Camenisch and Stadler [CS97] for NIZK proofs relations:

$$\mathrm{PoK}\left\{(\alpha, \beta) \mid Y = \alpha P \wedge Z = \beta P + \alpha G\right\} \ .$$

This notation represents a non-interactive proof of knowledge for discrete logarithms $\alpha, \beta \in \mathbb{Z}_p$ (the witness), which satisfy the conditions on the right-hand side involving the public group elements Y, P, Z, and G.

2.1 Keyed-Verification Anonymous Credentials

A KVAC system consists of issuers, users, and verifiers. The user receives a credential on their attributes $\mathbf{S}$ and can then generate proofs for the verifier, demonstrating possession of these attributes by revealing a non-empty subset $\mathbf{D}$ to the verifier.

Syntax. Adapting the definition for KVAC [CMZ14] and AC [FHS19], an attribute-based KVAC consists of the following (probabilistic) algorithms:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: Given a security parameter λ, public parameters pp are generated, which are accessible to all parties involved in the protocol.
- $(\mathsf{isk}, \mathsf{ipar}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$: Using pp as an input, the issuer executes the KeyGen algorithm to generate the secret key isk and the public issuer parameters ipar. The ipar implicitly define the attribute universe $\mathcal{S} = \mathcal{S}_{\mathsf{ipar}}$.
- $\mathsf{PreCred} \leftarrow \mathsf{IssueCred}(\mathsf{pp}, \mathbf{S}, \mathsf{isk}, \mathsf{ipar})$: This algorithm is executed by the issuer using its secret key isk to generate a pre-credential, $\mathsf{PreCred}$, for a user with a set of attributes $\mathbf{S} \in \mathcal{S}$.
- $\mathsf{Cred} \leftarrow \mathsf{ObtainCred}(\mathsf{pp}, \mathsf{PreCred}, \mathbf{S}, \mathsf{ipar})$: This is a deterministic algorithm executed by the user, where the user typically verifies the validity of $\mathsf{PreCred}$ using ipar. If the verification fails, the algorithm outputs $\perp$. Otherwise, it computes a credential, Cred.

- $\mathsf{Show} \leftarrow \mathsf{ShowCred}(\mathsf{pp}, \mathsf{Cred}, \mathbf{S}, \mathbf{D})$: By running this algorithm, the user computes a valid credential presentation, Show, attesting that $\mathbf{D} \subseteq \mathbf{S}$ for the user's attributes $\mathbf{S}$.
- $0/1 \leftarrow \mathsf{Verify}(\mathsf{pp}, \mathsf{Show}, \mathbf{D}, \mathsf{isk})$: This deterministic algorithm is executed by the verifier. It outputs 1 if Show is a valid credential presentation; otherwise, it outputs 0.

Security. A KVAC meets the following (security) properties:

Definition 1 (Correctness)**.** A KVAC scheme is correct if for all security parameters λ, all $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{isk}, \mathsf{ipar}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, and all sets of attributes $\mathbf{S} \subseteq \mathcal{S}_{\mathsf{ipar}}$:

$$\Pr\left[\begin{array}{ll} \mathsf{PreCred} \leftarrow \mathsf{IssueCred}(\mathsf{pp}, \mathbf{S}, \mathsf{isk}, \mathsf{ipar}); & \\ \mathsf{Cred} \leftarrow \mathsf{ObtainCred}(\mathsf{pp}, \mathsf{PreCred}, \mathbf{S}, \mathsf{ipar}); & \\ \forall\, \mathbf{D} \subseteq \mathbf{S}, \mathbf{D} \neq \emptyset; & : b = 1 \\ \mathsf{Show} \leftarrow \mathsf{ShowCred}(\mathsf{pp}, \mathsf{Cred}, \mathbf{S}, \mathbf{D}); & \\ b \leftarrow \mathsf{Verify}(\mathsf{pp}, \mathsf{Show}, \mathbf{D}, \mathsf{isk}) & \end{array}\right] = 1\ .$$

Correctness ensures that a user who follows the protocol can successfully convince the verifier that they possess a valid credential for any subset of his attributes.

Definition 2 (Unforgeability)**.** A KVAC scheme is unforgeable if for all PPT adversaries $\mathcal{A}$, we have:

$$\Pr\left[\begin{array}{ll} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); & \\ (\mathsf{isk}, \mathsf{ipar}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}); & b = 1 \wedge \\ (\mathsf{Show}^*, \mathbf{D}^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Cred}}(\cdot), \mathcal{O}_{\mathsf{Verify}}(\cdot)}(\mathsf{pp}, \mathsf{ipar}) & : \{\nexists\, \mathbf{S} \in \mathcal{Q}_{\mathsf{Cred}} \\ b \leftarrow \mathsf{Verify}(\mathsf{pp}, \mathsf{Show}^*, \mathbf{D}^*, \mathsf{isk}) & \mathbf{D}^* \subseteq \mathbf{S}\} \end{array}\right] \leq \mathsf{negl}(\lambda)\ ,$$

where initially, $\mathcal{Q}_{\mathsf{Cred}} = \emptyset$. The oracle $\mathcal{O}_{\mathsf{Cred}}(\mathbf{S})$ generates and returns $\mathsf{PreCred} \leftarrow \mathsf{IssueCred}(\mathsf{pp}, \mathbf{S}, \mathsf{isk}, \mathsf{ipar})$ and adds the attribute set $\mathbf{S}$ to $\mathcal{Q}_{\mathsf{Cred}}$, i.e. $\mathcal{Q}_{\mathsf{Cred}}$ is updated to $\mathcal{Q}_{\mathsf{Cred}} \cup \{\mathbf{S}\}$. The oracle $\mathcal{O}_{\mathsf{Verify}}(\mathsf{Show}, \mathbf{D})$ returns $b = \mathsf{Verify}(\mathsf{pp}, \mathsf{Show}, \mathbf{D}, \mathsf{isk})$.

The unforgeability property guarantees that no entity can generate a valid presentation for attributes for which they have not received a credential.

Definition 3 (Unlinkability)**.** A KVAC scheme is unlinkable if for all PPT adversaries $\mathcal{A}$,

$$\Pr\left[\begin{array}{ll} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); & \\ (\mathbf{S}_0, \mathsf{PreCred}_0, \mathbf{S}_1, \mathsf{PreCred}_1, \mathsf{ipar}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}); & \mathsf{Cred}_0 \neq \perp \wedge \\ \mathsf{Cred}_0 \leftarrow \mathsf{ObtainCred}(\mathsf{pp}, \mathsf{PreCred}_0, \mathbf{S}_0, \mathsf{ipar}); & : \mathsf{Cred}_1 \neq \perp \wedge \\ \mathsf{Cred}_1 \leftarrow \mathsf{ObtainCred}(\mathsf{pp}, \mathsf{PreCred}_1, \mathbf{S}_1, \mathsf{ipar}); & b = b' \\ b \xleftarrow{\$} \{0,1\};\ b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Show}_b}(\cdot)}(\mathsf{pp}, \mathsf{st}) & \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)\ ,$$

where $\mathcal{O}_{\mathsf{Show}_b}(\mathbf{D})$ checks that $\emptyset \neq \mathbf{D} \subseteq \mathbf{S}_0 \cap \mathbf{S}_1$. If the check succeeds, it returns $\mathsf{Show} \leftarrow \mathsf{ShowCred}(\mathsf{pp}, \mathsf{Cred}_b, \mathbf{S}_b, \mathbf{D})$.

The unlinkability property ensures that no entity can learn anything about a user during the credential presentation phase other than the fact that they possess a credential on a set that is a superset of or equal to $\mathbf{D}$. Not only is the presentation phase unlinkable to the obtaining phase, but multiple presentations of the credential are also unlinkable, a property known as multi-show unlinkability.

Note that in this definition, the adversary can internally generate credentials for any set of attributes of their choice, as they provide ipar to the challenger and can thereby have knowledge of the secret keys. In particular, $\mathcal{A}$ can compute $\mathsf{Cred}_0, \mathsf{Cred}_1$ (as $\mathsf{ObtainCred}$ is deterministic). Consequently, this definition accounts for scenarios where the issuer and verifier act as adversaries.

3 Structure-Preserving MAC on Equivalence Classes

In this section, we introduce a new primitive called Structure-Preserving Message Authentication Code on Equivalence-Classes, SP-MAC-EQ in short, and define its security properties. This primitive can be seen as a natural extension of both well-known primitives: standard MAC (cf. Appendix A.2) and SPS-EQ scheme [FHS19] (cf. Appendix A.3). Additionally, we propose an efficient SP-MAC-EQ by turning the initial SPS-EQ into a designated verifier setting.

Throughout the remainder of this paper, we use the same equivalence relation used to partition the message space $(\mathbb{G}^*)^\ell$ as described in [FHS19]:

$$\mathcal{R} := \left\{(\mathbf{M}, \mathbf{M}') \in (\mathbb{G}_1^*)^\ell \times (\mathbb{G}_1^*)^\ell \;\middle|\; \exists \mu \in \mathbb{Z}_p^* :\ \mu\mathbf{M} = \mathbf{M}'\right\} \cdot \tag{1}$$

Therefore, $[\mathbf{M}]_\mathcal{R}$ represents the set of all $\mathbf{M}' = \mu\mathbf{M}$, where $\mu \in \mathbb{Z}_p^*$.

3.1 SP-MAC-EQ: Syntax and Definitions

Similar to the distinction between MACs and digital signatures, the difference between SP-MAC-EQ and SPS-EQ (cf. Definition 19) is that the key generation algorithm in SP-MAC-EQ does not return a public key. As a result, only the party with access to the secret key can run the verification algorithm. The formal definition of SP-MAC-EQ is provided below.

Definition 4 (Structure-Preserving MAC on Equivalence Classes)**.** In an asymmetric bilinear group, an SP-MAC-EQ over message space $\mathcal{M} := (\mathbb{G}_1^*)^\ell$ with $\ell > 1$ consists of the following PPT algorithms:

- $\mathsf{pp} \leftarrow \mathsf{MEQ.Setup}_\mathcal{R}(1^\lambda)$: Take the security parameter λ in its unary representation as input. Output the set of public parameters pp which is given to the following algorithms.
- $\mathsf{sk} \leftarrow \mathsf{MEQ.KeyGen}_\mathcal{R}(\mathsf{pp}, \ell)$: Take an integer $\ell > 1$ as input. Output secret key sk.
- $(\tau, \perp) \leftarrow \mathsf{MEQ.MAC}_\mathcal{R}(\mathsf{pp}, \mathsf{sk}, \mathbf{M}; a)$: Take secret key sk, a representative message $\mathbf{M} \in \mathcal{M}$ for class $[\mathbf{M}]_\mathcal{R}$, and a random scalar $a \in \mathbb{Z}_p^*$ as inputs and output a tag τ.
- $0/1 \leftarrow \mathsf{MEQ.Verify}_\mathcal{R}(\mathsf{pp}, \mathsf{sk}, \tau, \mathbf{M})$: Take a representative message $\mathbf{M} \in \mathcal{M}$, a tag τ and a secret key sk as inputs. Output 0 (reject) or 1 (accept).
- $\tau' \leftarrow \mathsf{MEQ.ChgRep}_\mathcal{R}(\mathsf{pp}, \tau; \mu)$: Take a tag τ on representative message $\mathbf{M} \in \mathcal{M}$, and a scalar $\mu \in \mathbb{Z}_p^*$ as inputs. Return a randomized tag τ' on new representative message $\mathbf{M}' = \mu\mathbf{M}$.

Security Properties. The primary security requirements for an SP-MAC-EQ scheme are *correctness*, *Existential Unforgeability under Chosen Message Attack given a Verification Oracle (UF-CMVA)*, *class-hiding*, and *perfect adaptation of tags*, which we define below.

Definition 5 (Correctness)**.** An SP-MAC-EQ scheme over $\mathcal{M} := (\mathbb{G}_1^*)^\ell$ with $\ell > 1$ is *correct* if for all $\forall\lambda \in \mathbb{N}, \mathbf{M} \in \mathcal{M}$, we have:

$$\Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}_\mathcal{R}(1^\lambda);\ \mathsf{sk} \leftarrow \mathsf{KeyGen}_\mathcal{R}(\mathsf{pp}, \ell); \\ \tau \leftarrow \mathsf{MAC}_\mathcal{R}(\mathsf{pp}, \mathsf{sk}, \mathbf{M}; a);\ \tau' \leftarrow \mathsf{ChgRep}_\mathcal{R}(\mathsf{pp}, \tau; \mu); \\ b = \mathsf{Verify}_\mathcal{R}(\mathsf{pp}, \mathsf{sk}, \mathbf{M}, \tau);\ b' = \mathsf{Verify}_\mathcal{R}(\mathsf{pp}, \mathsf{sk}, \mu\mathbf{M}, \tau') \end{array} : \begin{array}{l} b = 1 \wedge \\ b' = 1 \end{array}\right] = 1\ .$$

Definition 6 (UF-CMVA)**.** An SP-MAC-EQ over $\mathcal{M} \coloneqq (\mathbb{G}_1^*)^\ell$ is UF-CMVA-secure if for all $\ell > 1$ and PPT adversaries $\mathcal{A}$ with access to the MAC oracle, $\mathcal{O}_{\mathsf{MAC}}(\cdot)$, and verification oracle, $\mathcal{O}_{\mathsf{Verify}}(\cdot)$, we have:

$$\Pr\left[\begin{array}{l|l} \mathsf{pp} \leftarrow \mathsf{Setup}_{\mathcal{R}}(1^\lambda);\ \mathsf{sk} \leftarrow \mathsf{KeyGen}_{\mathcal{R}}(\mathsf{pp}, \ell); & \\ (\tau^*, \mathbf{M}^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{MAC}}(\cdot), \mathcal{O}_{\mathsf{Verify}}(\cdot)}(\mathsf{pp}); & \begin{array}{l} b = 1 \wedge \\ \mathbf{M}^* \notin \mathcal{Q}_{\mathsf{MAC}} \end{array} \\ b = \mathsf{Verify}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}, \tau^*, \mathbf{M}^*) & \end{array}\right] \leq \mathsf{negl}(\lambda)\ .$$

The MAC oracle $\mathcal{O}_{\mathsf{MAC}}(\cdot)$ takes a message $\mathbf{M} \in \mathcal{M}$, samples $a \xleftarrow{\$} \mathbb{Z}_p^*$, runs $\mathsf{MAC}(\mathsf{pp}, \mathsf{sk}, \mathbf{M}; a)$ and adds the equivalence class $[\mathbf{M}]_{\mathcal{R}}$ of message to a query set $\mathcal{Q}_{\mathsf{MAC}}$, i.e. $\mathcal{Q}_{\mathsf{MAC}}$ is updated to $\mathcal{Q}_{\mathsf{MAC}} \cup [\mathbf{M}]_{\mathcal{R}}$. The adversary can additionally query the verification oracle $\mathcal{O}_{\mathsf{Verify}}(\cdot)$; it takes a message $\mathbf{M}$ and its tag τ and returns $\mathsf{Verify}(\mathsf{pp}, \mathsf{sk}, \tau, \mathbf{M})$.

Definition 7 (Class-Hiding)**.** A relation $\mathcal{R}$ is called class-hiding if, for all PPT adversaries, $\mathcal{A}$, and $\ell > 1$ we have:

$$\Pr\left[\begin{array}{l} \mathbf{M} \xleftarrow{\$} (\mathbb{G}_1^*)^\ell, \mathbf{M}_0 \xleftarrow{\$} (\mathbb{G}_1^*)^\ell, \mathbf{M}_1 \xleftarrow{\$} [\mathbf{M}]_{\mathcal{R}} \\ b \xleftarrow{\$} \{0,1\}, b' \leftarrow \mathcal{A}(\mathbf{M}, \mathbf{M}_b) \end{array} : b' = b\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)\ .$$

The class-hiding property ensures that it is computationally hard to distinguish elements of the same equivalence class from randomly sampled elements of the same size within the group.

Definition 8 (Perfect Adaptation of Tags)**.** An SP-MAC-EQ scheme over $\mathcal{M} \coloneqq (\mathbb{G}_1^*)^\ell$ perfectly adapts tags if for all tuples $(\mathsf{sk}, \mathbf{M}, \tau, \mu)$, where $\mathsf{sk} \leftarrow \mathsf{KeyGen}_{\mathcal{R}}(\mathsf{pp}, \ell)$, $\mathbf{M} \in (\mathbb{G}_1^*)^\ell$, $\mu \in \mathbb{Z}_p^*$, $a \xleftarrow{\$} \mathbb{Z}_p^*$ and $\mathsf{Verify}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}, \tau, \mathbf{M}) = 1$, the tags output by $\mathsf{MEQ.MAC}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}, \mu\mathbf{M}; a)$ and $\mathsf{MEQ.ChgRep}_{\mathcal{R}}(\mathsf{pp}, \tau; \mu)$ are identically distributed.

Perfect adaptation ensures that tags generated by adapting any valid tag with ChgRep follow the same distribution as a fresh MAC generated via MAC.

3.2 SP-MAC-EQ: An Efficient Instantiation

Our SP-MAC-EQ scheme is constructed by adapting the FHS19's SPS-EQ scheme [FHS19]. While more recent SPS-EQ schemes have been proposed, we opted for this choice due to the efficiency and simple structure offered by this SPS-EQ construction.

As a MAC is a keyed-verification digital signature, we exclude the public verification keys from the design of our SP-MAC-EQ. In this scenario, compared to FHS19's SPS-EQ, the tag (i.e. signature) is one group element shorter, and the verification process is more efficient due to a reduction in the number of pairing operations by a factor of ℓ. More formally, given the formal definition in Definition 4, our SP-MAC-EQ is detailed below.

- $\mathsf{MEQ.Setup}_{\mathcal{R}}(1^\lambda)$: Run $\mathsf{BG} \leftarrow \mathcal{BG}(1^\lambda)$ and return $\mathsf{pp} \coloneqq \mathsf{BG}$ as output.
- $\mathsf{MEQ.KeyGen}_{\mathcal{R}}(\mathsf{pp}, \ell)$: Take pp and vector size $\ell > 1$ as inputs. Output $\mathsf{sk} \coloneqq \{x_i\}_{i\in[1,\ell]} \xleftarrow{\$} (\mathbb{Z}_p^*)^\ell$.
- $\mathsf{MEQ.MAC}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}, \mathbf{M}; a)$: Parse $\mathbf{M} \coloneqq (M_i \in \mathbb{G}_1)_{i\in[1,\ell]}$, random $a \in \mathbb{Z}_p^*$ and $\mathsf{sk} \coloneqq \{x_i\}_{i\in[1,\ell]}$. Return the tag $\tau \coloneqq (R, T) \coloneqq \left(a\left(\sum_{i\in[1,\ell]} x_i M_i\right), a^{-1}\mathsf{G}_2\right) \in \mathbb{G}_1 \times \mathbb{G}_2$ as output.
- $\mathsf{MEQ.Verify}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}, \mathbf{M}, \sigma)$: Parse the secret key $\mathsf{sk} \coloneqq \{x_i\}_{i\in[1,\ell]}$, the message $\mathbf{M} \coloneqq (M_i)_{i\in[1,\ell]}$ and the tag $\tau \coloneqq (R, T)$. Return 1, if $M_i \neq 0_{\mathbb{G}_1}$ for all $i \in [1, \ell]$ and the following equation holds, else output 0.

$$e\left(\sum_{i\in[1,\ell]} x_i M_i, \mathsf{G}_2\right) = e(R, T)\ . \tag{2}$$

- $\mathsf{MEQ.ChgRep}_{\mathcal{R}}(\mathsf{pp}, \tau; \mu)$: Parse $\tau := (R, T)$ along with an integer $\mu \in \mathbb{Z}_p^*$ as input. Sample $\zeta \xleftarrow{\$} \mathbb{Z}_p^*$ and then return $\tau' := (R', T') := (\zeta\mu R, \zeta^{-1} T)$.

Notably, in the original SPS-EQ scheme [FHS19], inclusion of the term $a^{-1}\mathsf{G}_1 \in \mathbb{G}_1$ in signatures is crucial to achieve security. Without it, the SPS-EQ public key $(x_i \mathsf{G}_2)_{i\in[1,\ell]}$ allows deriving the valid signature $(\mathsf{G}_1, \sum x_i \mathsf{G}_2)$ on message $(\mathsf{G}_1, \mathsf{G}_1, \ldots, \mathsf{G}_1)$ (note that this fulfills our $\mathsf{MEQ.Verify}$ equation). In the MAC setting, absent public keys, we are able to omit the $a^{-1}\mathsf{G}_1$ term. This showcases that our MAC construction circumvents the impossibility result that structure-preserving *signatures* (SPS) must be comprised of at least three group elements [AGHO11].

Security. The following theorems state that our proposed SP-MAC-EQ scheme satisfies all the security properties.

Theorem 1. *The proposed SP-MAC-EQ scheme is correct (Definition 5).*

Proof. The output of the $\mathsf{MEQ.MAC}$ algorithm passes the verification check because $e\left(\left(\sum_{i\in[1,\ell]} x_i M_i\right), \mathsf{G}_2\right) = e\left(a\left(\sum_{i\in[1,\ell]} x_i M_i\right), a^{-1}\mathsf{G}_2\right) = e(R, T)$ □

Theorem 2. *The proposed SP-MAC-EQ scheme is UF-CMVA secure (Definition 6) in the Generic Group Model (GGM).*

The proof follows a similar structure to that of [FHS19], but is adapted to our SP-MAC-EQ construction. One notable difference is that SP-MAC-EQ requires a verification oracle. However, in our GGM proof strategy (see Appendix B), this oracle merely checks a polynomial equation that the adversary could verify on its own. A detailed proof is provided in Appendix D.1.

Theorem 3 (Proposition 1 in [FHS19])**.** *Assuming the hardness of the Decisional Diffie-Hellman (DDH) problem (Definition 14), the proposed SP-MAC-EQ scheme's relation is class-hiding (Definition 7).*

The proof is identical to that of Proposition 1 in [FHS19].

Theorem 4. *The proposed SP-MAC-EQ scheme achieves the perfect adaptation of tags property (Definition 8).*

Proof. The tags produced by $\mathsf{MEQ.MAC}$ on some message $\mu\mathbf{M}$ are of the form $(R, T) = \left(a\left(\sum_{i\in[1,\ell]} x_i(\mu M_i)\right),\ a^{-1}\mathsf{G}_2\right)$, for a uniform $a \in \mathbb{Z}_p^*$. The algorithm $\mathsf{MEQ.ChgRep}$, given a valid tag on $\mathbf{M}$ of the form $\left(a'\left(\sum_{i\in[1,\ell]} x_i M_i\right),\ a'^{-1}\mathsf{G}_2\right)$, outputs a tag of the form $(R', T') = \left(\zeta\mu a'\left(\sum_{i\in[1,\ell]} x_i M_i\right),\ \zeta^{-1}a'^{-1}\mathsf{G}_2\right) = \left((\zeta a')\left(\sum_{i\in[1,\ell]} x_i(\mu M_i)\right),\ (\zeta a')^{-1}\mathsf{G}_2\right)$, for uniformly random $\zeta \in \mathbb{Z}_p^*$. Since both a and $\zeta a'$ are uniformly distributed in $\mathbb{Z}_p^*$, the resulting tag pairs (R, T) and (R', T') are identically distributed. □

4 Designated-Verifier Set Commitment

We adapt the set commitment scheme from [FHS19, Ngu05] to fit a designated-verifier scenario. Note that recently Orrú in [Orr24] formally defined the related notion of Designated Verifier Polynomial Commitments; however, in our case, to enable selective disclosure of attributes, it is more natural to talk about designated verifier *set commitments*, which we define as follows.

4.1 DVSC: Syntax and Definitions

Definition 9 (Designated-Verifier Set Commitment schemes)**.** A DVSC consists of the following PPT algorithms:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: The setup algorithm takes the security parameter in its unary representation, and returns public parameters pp as output.
- $(\mathsf{sk}, \mathsf{ipar}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, 1^t)$: The key generation algorithm takes an upper bound t for the size of attribute sets, and returns the secret key sk and parameters ipar as output. ipar is implicit input to the following algorithms except $\mathsf{VerifySubset}(\cdot)$. ipar also implicitly defines the space $\mathcal{S} = \mathcal{S}_{\mathsf{ipar}}$ of committable values.
- $C \leftarrow \mathsf{Commit}(\mathsf{pp}, \mathsf{ipar}, \mathbf{S})$: The commit algorithm as a deterministic algorithm takes a set $\mathbf{S} \in \mathcal{S}$ as inputs, and returns the commitment value C as output.
- $C' \leftarrow \mathsf{Randomize}(\mathsf{pp}, C; \mu)$: The randomize algorithm takes a commitment C and a random element $\mu \xleftarrow{\$} \Gamma$, and returns a randomized commitment C' as output.
- $W \leftarrow \mathsf{OpenSubset}(\mathsf{pp}, \mathsf{ipar}, \mu, \mathbf{S}, \mathbf{D})$: The subset open algorithm takes μ, set $\mathbf{S}$ and a subset $\mathbf{D} \subseteq \mathbf{S}$ as inputs, and returns an opening W for $\mathbf{D}$.
- $0/1 \leftarrow \mathsf{VerifySubset}(\mathsf{pp}, \mathsf{sk}, C', W, \mathbf{D})$: The verify algorithm takes secret key sk, a randomized commitment C', opening W and a subset $\mathbf{D}$ as inputs, and returns either 0 (reject) or 1 (accept).

The inclusion of the extra KeyGen algorithm and the secret key in $\mathsf{VerifySubset}$ arises from transitioning to a designated verifier. For our construction later, it will be useful to have a "canonical" (deterministic) commitment C for every set $\mathbf{S}$. This allows both issuer and user to compute the same commitment. For this reason, unlike the set commitment scheme from [FHS19], our Commit algorithm is deterministic and does not return opening information. To enhance privacy, we introduce an additional algorithm, $\mathsf{Randomize}$, enabling users to randomize their deterministically computed commitments. A (randomized) commitment C' can be opened by simply revealing the set $\mathbf{S}$ and the randomness μ, which allows recomputing $C' = \mathsf{Randomize}(\mathsf{Commit}(\mathbf{S}); \mu)$.

Security Properties. We define security for DVSC, tailored towards use in larger constructions such as our $\mathsf{KVAC}_{\mathsf{MEQ}}$ in Section 5.

Definition 10 (Correctness)**.** A DVSC is correct if for all security parameters λ and limits $t \in \mathbb{N}$, all $\mathsf{pp} \in [\mathsf{Setup}(1^\lambda)]$ and $(\mathsf{sk}, \mathsf{ipar}) \in [\mathsf{KeyGen}(\mathsf{pp}, 1^t)]$, and all $\mathbf{S} \in \mathcal{S}_{\mathsf{ipar}}$ and all non-empty subsets $\mathbf{D} \subseteq \mathbf{S}$:

$$\Pr\left[\begin{array}{ll} C \leftarrow \mathsf{Commit}(\mathsf{pp}, \mathsf{ipar}, \mathbf{S});\ C' \leftarrow \mathsf{Randomize}(\mathsf{pp}, C; \mu); & \\ W \leftarrow \mathsf{OpenSubset}(\mathsf{pp}, \mathsf{ipar}, \mu, \mathbf{S}, \mathbf{D}); & : b = 1 \\ b \leftarrow \mathsf{VerifySubset}(\mathsf{pp}, \mathsf{sk}, C', W, \mathbf{D}) & \end{array}\right] = 1\ .$$

Definition 11 (Subset-Soundness)**.** A DVSC meets the subset soundness property if for all PPT adversaries $\mathcal{A}$, we have:

$$\Pr\left[\begin{array}{ll} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); & \\ (\mathsf{sk}, \mathsf{ipar}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, 1^t); & \{\exists \mu : C' = \mathsf{Randomize}(\mathsf{pp}, \\ (\mathbf{S}, C', \mathbf{D}, W) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{pp}, \mathsf{ipar}); & : \mathsf{Commit}(\mathsf{pp}, \mathsf{ipar}, \mathbf{S}); \mu)\} \wedge \\ b \leftarrow \mathsf{VerifySubset}(\mathsf{pp}, \mathsf{sk}, C', W, \mathbf{D}) & b = 1 \ \wedge \mathbf{D} \not\subseteq \mathbf{S} \end{array}\right] \leq \mathsf{negl}(\lambda)\ .$$

where $\mathcal{O}(C', W, \mathbf{D})$ returns $\mathsf{VerifySubset}(\mathsf{pp}, \mathsf{sk}, C', W, \mathbf{D})$.

Subset soundness states that if C' is a valid commitment to $\mathbf{S}$, then it is hard for $\mathcal{A}$ to output a subset opening proof W for $\mathbf{D} \not\subseteq \mathbf{S}$.

Note that there exists a weaker security property called *binding*, which is insufficient to prove the security of our KVAC construction in Section 5 but we discuss it in Definition 26 of Appendix E.

Definition 12 (Hiding)**.** A DVSC meets the hiding property if for all PPT adversaries $\mathcal{A}$, we have:

$$\Pr\left[\begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda);\ (\mathbf{S}_0, \mathbf{S}_1, \mathsf{st}, \mathsf{ipar}) \leftarrow \mathcal{A}(\mathsf{pp});\\ b \leftarrow \{0,1\};\ b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Randomize}_b}()}(\mathsf{st}, \mathsf{ipar})\end{array} : b' = b\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)\ ,$$

where $\mathcal{O}_{\mathsf{Randomize}_b}()$ on its i-th invocation chooses a random $\mu_i \xleftarrow{\$} \Gamma$ and returns $C' \leftarrow \mathsf{Randomize}(\mathsf{pp}, \mathsf{ipar}, \mathsf{Commit}(\mathsf{pp}, \mathsf{ipar}, \mathbf{S}_b); \mu_i)$.

Definition 13 (Subset Open Simulatability)**.** A DVSC scheme has *subset open simulatability* if for all PPT adversaries $\mathcal{A}$, there exists a PPT simulator $\mathsf{Sim} := (\mathsf{Sim}_0, \mathsf{Sim}_1)$ s.t. we have:

$$\Pr\left[\begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda);\ (\mathsf{st}, \mathsf{ipar}) \leftarrow \mathcal{A}(\mathsf{pp});\\ \mathsf{td} \leftarrow \mathsf{Sim}_0(\mathsf{view}_{\mathcal{A}})\\ b \leftarrow \{0,1\};\ b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{OpenSubset}_b}(\cdot)}(\mathsf{st})\end{array} : b' = b\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)\ ,$$

where $\mathcal{O}_{\mathsf{OpenSubset}_b}(\mu, \mathbf{S}, \mathbf{D})$ checks that $\emptyset \neq \mathbf{D} \subseteq \mathbf{S}$. If so, it returns either $W \leftarrow \mathsf{OpenSubset}(\mathsf{pp}, \mathsf{ipar}, \mu, \mathbf{S}, \mathbf{D})$, in case $b = 0$, or $W \leftarrow \mathsf{Sim}_1(\mathsf{td}, C', \mathbf{D})$, in case $b = 1$, where $C' = \mathsf{Randomize}(\mathsf{pp}, \mathsf{ipar}, \mathsf{Commit}(\mathsf{pp}, \mathsf{ipar}, \mathbf{S}), \mu)$

Subset open simulatability states that using a trapdoor, valid subset opening witnesses W can be simulated given only a randomized commitment C' and a subset $\mathbf{D}$ (but not the set $\mathbf{S}$ or the commitment randomness μ, which are normally needed to run $\mathsf{OpenSubset}$).

4.2 DVSC: An Efficient Instantiation

Next, we propose an efficient DVSC instantiation, heavily inspired by [FHS19, Ngu05]. In this scheme, we define the polynomial $f_{\mathbf{S}}(X)$ in a way that it vanishes on all elements in $\mathbf{S}$, i.e. $f_{\mathbf{S}}(X) = \prod_{s \in \mathbf{S}}(X - s) = \sum_{i=0}^{|\mathbf{S}|} f_i X^i$.

- $\mathsf{DVSC.Setup}(1^\lambda)$: Run $\mathsf{BG} \leftarrow \mathcal{BG}(1^\lambda)$ with prime order $p > 2^\lambda$, choose $\mathsf{G}' \xleftarrow{\$} \mathbb{G}_1$, and return public parameters $\mathsf{pp} := (\mathsf{BG}, \mathsf{G}')$. Since the construction does not require pairings, we only use $\mathbb{G}_1$ in the following and write $\mathsf{G} := \mathsf{G}_1$.
- $\mathsf{DVSC.KeyGen}(\mathsf{pp}, 1^t)$: Sample $v \xleftarrow{\$} \mathbb{Z}_p^*$ and compute $\pi := \mathsf{PoK}\{v \mid V_0 = \mathsf{G} \wedge \bigwedge_{j=0}^{t-1} vV_j = V_{j+1}\}$ (discussed in more detail in Appendix A.5). Define $\mathsf{sk} := v$ and $\mathsf{ipar} := (\{V_j := v^j\mathsf{G}\}_{j=0}^t, \pi)$, and return $(\mathsf{sk}, \mathsf{ipar})$ as output. The set of committable values $\mathcal{S}$ is defined as $\mathcal{S} := \{\mathbf{S} \subseteq \mathbb{Z}_p \setminus \{v\} \mid |\mathbf{S}| \leq t\}$.
- $\mathsf{DVSC.Commit}(\mathsf{pp}, \mathsf{ipar}, \mathbf{S})$: Parse $(\{V_j\}_{j=0}^t, \pi) \leftarrow \mathsf{ipar}$, and check the validity of π. If π is valid and $v \notin \mathbf{S}$, compute $f_{\mathbf{S}}(v)\mathsf{G} = \sum_{i=0}^{|\mathbf{S}|} f_i V_i$ and return $C := (f_{\mathbf{S}}(v)\mathsf{G}, \mathsf{G}')$. Otherwise, return $\perp$.
- $\mathsf{DVSC.Randomize}(\mathsf{pp}, C; \mu)$: Given a random integer $\mu \xleftarrow{\$} \mathbb{Z}_p^*$ and $(C_1, C_2) \leftarrow C$, return $C' := (\mu C_1, \mu C_2)$.
- $\mathsf{DVSC.OpenSubset}(\mathsf{pp}, \mathsf{ipar}, \mu, \mathbf{S}, \mathbf{D})$: Obtain the coefficients of polynomial $f_{\mathbf{S}\setminus\mathbf{D}}(X)$, and compute $f_{\mathbf{S}\setminus\mathbf{D}}(v)\mathsf{G}$. Return $W := \mu f_{\mathbf{S}\setminus\mathbf{D}}(v)\mathsf{G}$ as output.

- $\mathsf{DVSC.VerifySubset}(\mathsf{pp}, \mathsf{sk}, C', W, \mathbf{D})$: Compute $f_{\mathbf{D}}(v)$. Parse $(C'_1, C'_2) \leftarrow C'$, return 1 (accept) if $C'_1 = f_{\mathbf{D}}(v) \cdot W$, and 0 otherwise.

Unlike [FHS19], our commitment $(\mu f_{\mathbf{S}}(v)\mathsf{G}, \mu\mathsf{G}')$ consists of two group elements. This is mostly to make commitments valid messages for SP-MAC-EQ, which authenticates vectors of length at least 2. Additionally, if the commitment were restricted to a single group element $\mu f_{\mathbf{S}}(v)\mathsf{G}$, any commitment could be a randomized version of any other one, which would compromise the subset-soundness property. Specifically, in Definition 11, any arbitrary set $\mathbf{S}$ would automatically satisfy the first condition $(\{\exists \mu : C' = \mathsf{Randomize}(\mathsf{pp}, \mathsf{Commit}(\mathsf{pp}, \mathsf{ipar}, \mathbf{S}); \mu)\})$, making it trivial to break the soundness guarantee. However, with our two-group-element setting, each commitment belongs to a distinct equivalence class (similar to SPS-EQ/SP-MAC-EQ) for randomization. This ensures that randomizations do not trivially collide across different commitments, enabling the subset-soundness property.

For simplicity, the construction above checks the validity of the same proof π in ipar every time the user computes a commitment. In practice, the validity of π in ipar only needs to be checked once as it does not change between different Commit invocations. In particular, for our KVAC system (Section 5), when a user first joins the system, ipar and π are already known to the user and can be checked at that time. There is no need to re-check the same proof π again and again for every credential the user requests. Thus, this verification incurs only a one-time cost.

Security. The following theorems state that our proposed DVSC scheme satisfies all the security properties.

Theorem 5. *The proposed DVSC scheme achieves correctness (Definition 10).*

Note that for $W = \mu f_{\mathbf{S}\setminus\mathbf{D}}(v)\mathsf{G}$, we have $f_{\mathbf{D}}(v) \cdot W = \mu f_{\mathbf{S}}(v)\mathsf{G}$. So $\mathsf{DVSC.VerifySubset}$ accepts W. A detailed proof is provided in Appendix E.1.

Theorem 6. *If the NIZK is zero-knowledge, then the proposed DVSC scheme has the subset-soundness property (Definition 11) in the GGM.*

The winning condition reduces to the adversary producing $(\mathbf{S}, C' = (C'_1, C'_2), W)$, such that $(C'_1, C'_2) = \mu(f_{\mathbf{S}}(v)\mathsf{G}, \mathsf{G}')$ and $C'_1 = f_{\mathbf{D}}(v) \cdot W$. Hence, $\mu f_{\mathbf{S}}(v)\mathsf{G} = f_{\mathbf{D}}(v) \cdot W$. In the GGM, we can treat v as a symbol (cf. Appendix B). Since the adversary is limited to linear operations on the inputs $v^i\mathsf{G}$, the element W must also be of the form $f'(v)\mathsf{G}$ for some polynomial f'. Then, we obtain $\mu f_{\mathbf{S}}(v) = f_{\mathbf{D}}(v) \cdot f'(v)$, i.e., $f_{\mathbf{D}}$ divides $f_{\mathbf{S}}$ as polynomials in v. This implies that $\mathbf{D} \subseteq \mathbf{S}$ whenever the winning condition holds. In other words, the adversary cannot produce a witness for a set $\mathbf{D}$ that is not a subset of $\mathbf{S}$. A detailed proof is provided in Appendix E.3.

Theorem 7. *Assuming the hardness of the DDH problem, the proposed DVSC scheme has the hiding (Definition 12) property.*

Proof. (Informal) An invocation of $\mathcal{O}_{\mathsf{Randomize}_b}$ returns $C' = (\mu f_{\mathbf{S}_b}(v)\mathsf{G}, \mu\mathsf{G}')$ for a random $\mu \in \mathbb{Z}_p^*$. Due to hardness of the DDH problem (analogous to the class-hiding property of the SP-MAC-EQ scheme) and because the discrete logarithm between G and G', as well as the value μ, are unknown, C' is indistinguishable from a uniformly random element in $\mathbb{G}_1 \times \mathbb{G}_1$. □

Theorem 8. *Given a NIZK that is a proof of knowledge (Definition 25), the proposed DVSC scheme has subset open simulatability (Definition 13).*

The simulator extracts the trapdoor v from the PoK. Using this, the subset opening $W = \mu f_{\mathbf{S}\setminus\mathbf{D}}(v)\mathsf{G}$ for the randomized commitment $\mu C = (C'_1, C'_2) = (\mu f_{\mathbf{S}}(v)\mathsf{G},\ \mu\mathsf{G}')$ can be

perfectly simulated by $\mathsf{Sim}_1(v, \mu C, \mathbf{D})$. Specifically, the simulator computes $W = \frac{1}{f_{\mathbf{D}}(v)} C'_0$, which is correct because $W = \mu \frac{f_{\mathbf{S}}(v)}{f_{\mathbf{D}}(v)} \mathsf{G} = \mu f_{\mathbf{S}\setminus\mathbf{D}}(v)\mathsf{G}$. A detailed proof is provided in Appendix E.4.

5 KVAC from SP-MAC-EQ

Our first construction for KVAC is shown in Figure 1. In this construction, we employ our SP-MAC-EQ and DVSC schemes (Sections 4.2 and 3.2) to build a constant-size KVAC system ($\mathsf{KVAC}_{\mathsf{MEQ}}$) meeting all the required security properties discussed in Section 2.1.

Following [FHS19] and adapting their approach to the keyed verification setting, the general idea involves using the DVSC scheme to commit to a set $\mathbf{S}$ of user attributes. The issuer generates a tag $\mathsf{MAC}(\mathsf{Commit}(\mathbf{S}))$ on the commitment using SP-MAC-EQ, which serves as the user's credential. The user can then use $\mathsf{MEQ.ChgRep}$ and $\mathsf{OpenSubset}$ to demonstrate possession of a credential for $\mathbf{S}$ while revealing only a subset $\mathbf{D}$ of attributes. The security guarantees of the DVSC and SP-MAC-EQ schemes ensure the security of the resulting KVAC system. We now turn to the construction of $\mathsf{KVAC}_{\mathsf{MEQ}}$ as shown in Figure 1.

In the Setup phase, using $\mathcal{BG}$, the system generates all necessary public parameters for SP-MAC-EQ. For DVSC, we use $\mathbb{G}_1$, and by sampling a random group element G'_1 from $\mathbb{G}_1$, we obtain all the public parameters required for DVSC. The issuer then generates its secret keys $\mathsf{isk} = (x_1, x_2, v)$ and issuer parameters using the key generation algorithms of SP-MAC-EQ and DVSC. Additionally, the issuer includes a Pedersen commitment X to (x_1, x_2) in ipar.

To issue a credential for a user with attributes $\mathbf{S}$, the issuer first uses the DVSC scheme to compute the commitment $C = (f_{\mathbf{S}}(v)\mathsf{G}_1, \mathsf{G}'_1)$. Then, the issuer generates a tag with randomness a on this commitment using SP-MAC-EQ:

$$\tau = \left(a(x_1 f_{\mathbf{S}}(v)\mathsf{G}_1 + x_2\mathsf{G}'_1), a^{-1}\mathsf{G}_2\right) .$$

A user can employ the algorithms of SP-MAC-EQ and DVSC to present their credential in a privacy-preserving manner, provided the tag is well-formed. However, in the context of unlinkability, a malicious issuer might attempt to violate user privacy in two ways: by using different secret keys for different users or by deviating from the SP-MAC-EQ protocol (e.g., multiplying the first component of the MAC by a and the second by $(a/2)^{-1}$). Since only the issuer or verifier can verify the MAC, the user cannot detect such cheating.

A security concept known as *key-parameter consistency* [CMZ14] aims to prevent such attacks. To ensure key-parameter consistency, the issuer sends a proof π (whose instantiation is discussed in Appendix A.5) along with the tag τ, proving that the MAC is well-formed according to the protocol and that it was generated using the unique secret keys the issuer has committed to in the issuer parameters ipar.

The user first computes the commitment $C = (f_{\mathbf{S}}(v)\mathsf{G}_1, \mathsf{G}'_1)$. If the proof π is valid, the user accepts the tag τ sent by the issuer as their credential Cred. Since the commitment C is not required during the presentation phase, it does not need to be stored. Consequently, the size of Cred is independent of the size of the attribute set $\mathbf{S}$, resulting in a KVAC scheme with a constant-size credential consisting of just two group elements (cf. Table 1).

For the presentation phase, the user employs $\mathsf{MEQ.ChgRep}$ with randomness μ to randomize the tag:

$$\tau' = \left(\zeta a(x_1 \mu f_{\mathbf{S}}(v)\mathsf{G}_1 + x_2 \mu \mathsf{G}'_1), (\zeta a)^{-1}\mathsf{G}_2\right) ,$$

where ζ is additional randomness generated in $\mathsf{MEQ.ChgRep}$. This step ensures that the randomized commitment $C' = (\mu f_{\mathbf{S}}(v)\mathsf{G}_1, \mu\mathsf{G}'_1)$ and tag are unlinkable to the original commitment and tag while still being accepted by the verifier using $\mathsf{MEQ.Verify}$.

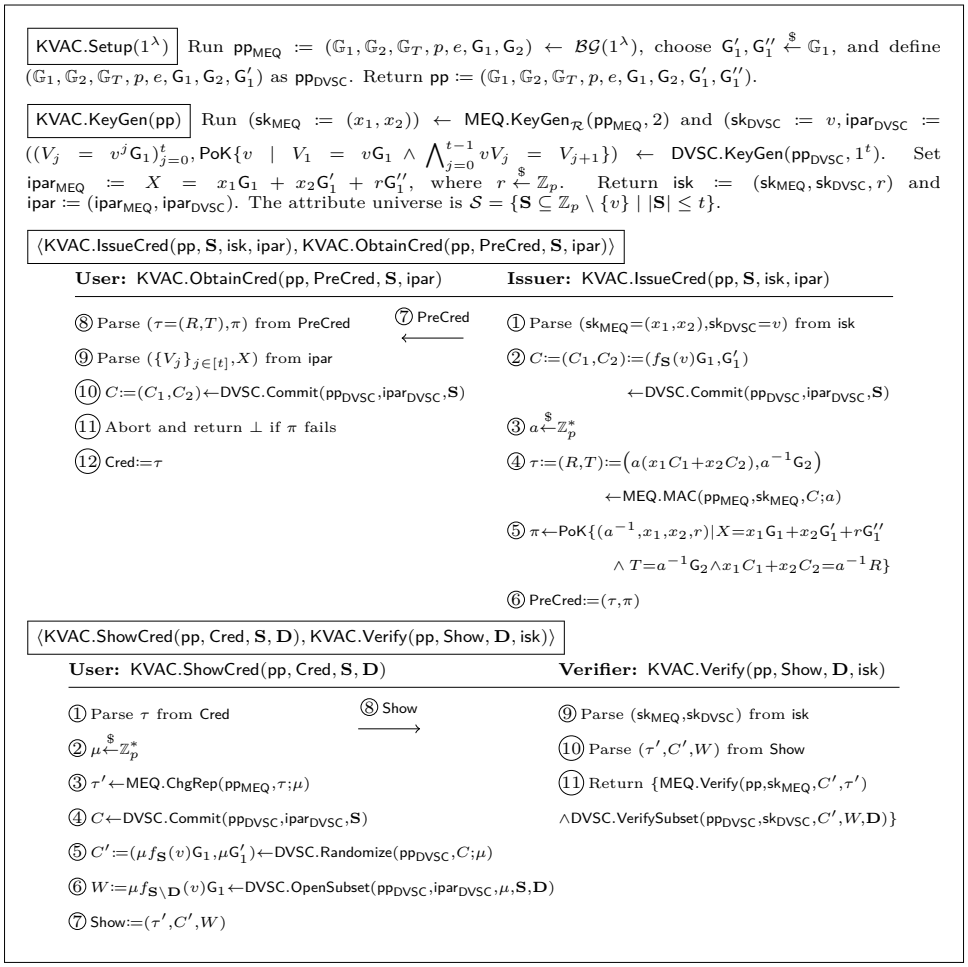

Figure 1: Our pairing-based multi-show unlinkable KVAC system $\mathsf{KVAC}_{\mathsf{MEQ}}$.

Next, the user employs OpenSubset of DVSC with the same randomness μ for a subset of attributes $\mathbf{D}$ to obtain $W = \mu f_{\mathbf{S}\setminus\mathbf{D}}(v)\mathsf{G}_1$. The DVSC guarantees that W is unlinkable to the original commitment and reveals nothing beyond the subset $\mathbf{D}$.

The user then sends the randomized commitment C', the opening subset witness W, and the randomized tag τ' to the verifier. The verifier accepts the presentation if the following conditions hold: (1) τ' is a valid tag for C' using MEQ.Verify, and (2) W is a valid opening for the commitment C' and subset $\mathbf{D}$ using DVSC.VerifySubset.

This process ensures that only the subset $\mathbf{D}$ is revealed, with unforgeability of the KVAC system being guaranteed by the MEQ.Verify and DVSC.VerifySubset algorithms.

Security. The following theorems state that $\mathsf{KVAC}_{\mathsf{MEQ}}$ satisfies all the security properties of a KVAC in Section 2.1.

Theorem 9. *Given the correctness of underlying NIZK, our DVSC, our SP-MAC-EQ schemes,* $\mathsf{KVAC}_{\mathsf{MEQ}}$ *achieves correctness (Definition 1).*

The proof of this theorem is straightforward.

Theorem 10. *Assume the SP-MAC-EQ is UF-CMVA secure (cf. Definition 6), the DVSC is subset sound (cf. Definition 11), and the NIZK is zero-knowledge. Then* $\mathsf{KVAC}_{\mathsf{MEQ}}$ *is unforgeable (Definition 2).*

To forge a valid presentation, an adversary must either (1) forge a MAC tag on a new commitment that was never signed, or (2) reuse an existing MAC tag on a previously signed commitment, but produce a forged subset opening that opens the commitment to a set which is not a subset of (or equal to) any set previously queried to $\mathcal{O}_{\mathsf{Cred}}$. The first case reduces to the UF-CMVA security of the MAC, and the second case reduces to the subset soundness property of the DVSC scheme. A detailed proof of this theorem is provided in Appendix F.1.

Theorem 11. *Assume that the NIZK is a proof of knowledge (Definition 25), the DVSC has subset opening simulatability, and the decisional Diffie-Hellman assumption holds in* $\mathbb{G}_1$*, then* $\mathsf{KVAC}_{\mathsf{MEQ}}$ *meets the unlinkability property defined in Definition 3.*

A detailed proof is provided in Appendix F.2. The general strategy is to incrementally modify the output (τ', C', W) of $\mathcal{O}_{\mathsf{Show}_b}$ to make it independent of b (i.e., independent of both Cred_b and $\mathbf{S}_b$), as follows:

1. **Recompute τ' using the extracted key:** Extract the SP-MAC-EQ key from the PoKs in $\mathsf{PreCred}_0$ and $\mathsf{PreCred}_1$, and use it to compute τ' directly from C', rather than by adapting the tag τ_b. This change is indistinguishable from the original game due to the perfect adaptation property of SP-MAC-EQ.
2. **Simulate the subset opening W:** Replace the actual opening W (which depends on $\mathbf{S}_b$) with a simulated opening computed using only $\mathbf{D}$ and C'. This change is indistinguishable from the previous game due to the subset opening simulatability property of the DVSC scheme.
3. **Replace C' with a random commitment:** Replace C'——normally a randomized version of user b's commitment——with a uniformly random pair of group elements. This modification is indistinguishable from the previous game due to the class-hiding property of SP-MAC-EQ.

After these transformations, the output (τ', C', W) is independent of the bit b, and thus the adversary's advantage is no better than random guessing.

6 $\mathsf{KVAC}_{\mathsf{GGM}}$: our construction without pairings

The construction $\mathsf{KVAC}_{\mathsf{GGM}}$ is defined in Figure 2. As discussed in the introduction, in order to move to a pairingless construction, we replace the SP-MAC-EQ (which requires pairings) with a simple homomorphic MAC. To combat users combining their credentials (which is not prevented by homomorphic MACs in any meaningful way), each credentials' set commitment $C = f_{\mathbf{S}}(v)y\mathsf{G}$ will be based on a random basis $y\mathsf{G}$. In the following, we elaborate on details.

In the Setup phase, as we are not using pairings, only one cyclic group $\mathbb{G}$ with generator G of prime order p is generated as pp.

In the KeyGen algorithm, only two keys (x, v) are generated for use in the MAC and commitment, respectively. Additionally, three group elements are generated for committing to these secret keys as ipar. Unlike $\mathsf{KVAC}_{\mathsf{MEQ}}$, there is no need to publish $(v^j\mathsf{G})_{j=0}^{t}$, as these elements are useless for the users due to the randomness y multiplied in their commitment.

To issue a credential for a user with the attributes $\mathbf{S}$, the issuer computes $C = yf_{\mathbf{S}}(v)\mathsf{G}$ with a random scalar y. The issuer then generates the tag on this commitment as $\tau = xC$. The issuer sends the tag along with:

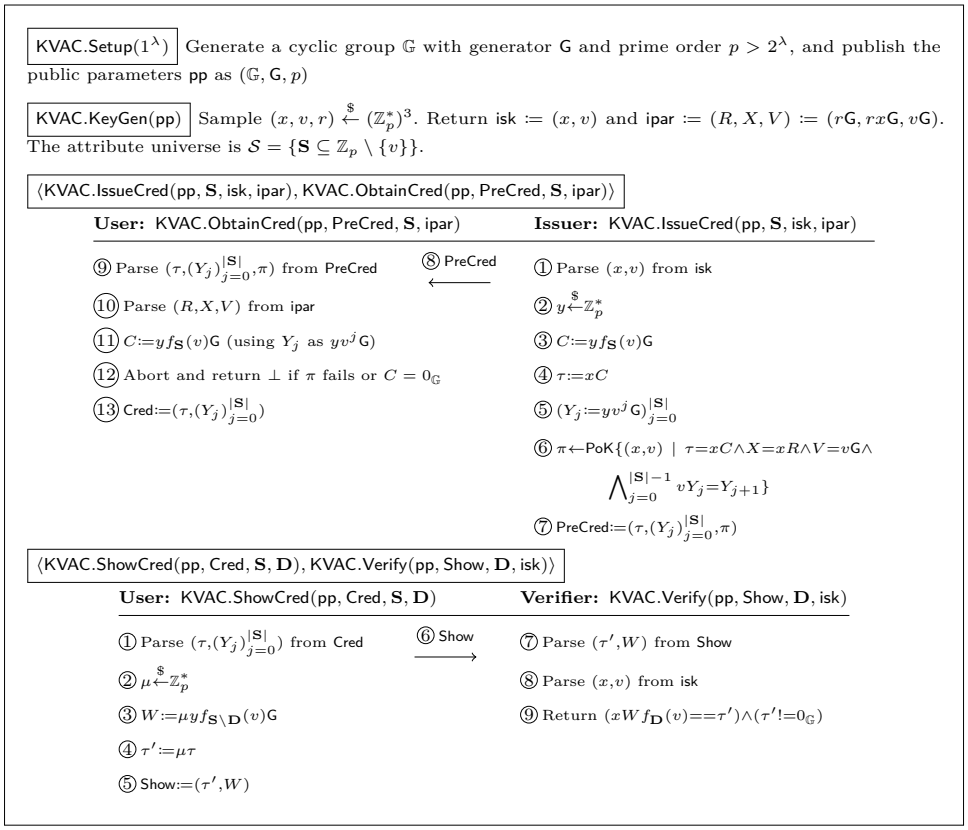

Figure 2: Our pairingless multi-show unlinkable KVAC system $\mathsf{KVAC}_{\mathsf{GGM}}$.

1. $y(v^j\mathsf{G})_{j=0}^{|\mathbf{S}|}$ to enable the user to compute commitments and subset openings;
2. A NIZK proof π (instantiated in Appendix A.5) to ensure key-parameter consistency, preventing the issuer from cheating.

The proof π guarantees that the value x used for computing the MAC and the value v used in $y(v^j\mathsf{G})_{j=0}^{|\mathbf{S}|}$ to create subsequent elements are the same values committed to in ipar.

Next, the user re-computes the commitment C and verifies the validity of the NIZK proof π. If π is valid, the user accepts $\left(\tau, y(v^j\mathsf{G})_{j=0}^{|\mathbf{S}|}\right)$ as the credential.

For the credential presentation with revealing of a non-empty subset of attributes $\mathbf{D} \subseteq \mathbf{S}$, the user computes the open subset element $W = \mu y f_{\mathbf{S}\setminus\mathbf{D}}(v)\mathsf{G}$ using a random scalar μ and the elements $y(v^j\mathsf{G})_{j=0}^{|\mathbf{S}\setminus\mathbf{D}|}$. The randomized tag is $\tau' = \mu\tau$. The verifier only needs to check that τ' is not $0_{\mathbb{G}}$ and that $xWf_{\mathbf{D}}(v)$ is τ'. Therefore, W and τ' are uniformly random elements because of μ but with only one condition $\tau' = xWf_{\mathbf{D}}(v)$, which is true for every valid credential presentation. This provides the unlinkability of $\mathsf{KVAC}_{\mathsf{GGM}}$.

Security. The following theorems state that $\mathsf{KVAC}_{\mathsf{GGM}}$ satisfies all the security properties of a KVAC Section 2.1. Detailed proofs are provided in Appendix G.

Theorem 12. *Given the correctness of underlying NIZK,* $\mathsf{KVAC}_{\mathsf{GGM}}$ *is correct (Definition 1).*

Proof. For an honest execution, the verification equation holds: $xWf_{\mathbf{D}}(v) = x\cdot(\mu y f_{\mathbf{S}\setminus\mathbf{D}}(v)\mathsf{G})\cdot f_{\mathbf{D}}(v) = x\mu y f_{\mathbf{S}}(v)\mathsf{G} = \mu(x\cdot y f_{\mathbf{S}}(v)\mathsf{G}) = \mu(x\cdot C) = \mu\tau = \tau'$. □

Theorem 13. *If the NIZK is zero-knowledge, then* $\mathsf{KVAC}_{\mathsf{GGM}}$ *is unforgeable (Definition 2) in the GGM.*

The detailed proof of the theorem is provided in Appendix G.1. The proof begins by simulating the issuer's NIZK in the issuance phase. We then argue that, in the generic group model, the adversary cannot recombine the terms obtained from issued credentials and public/issuer's parameters into a valid forged presentation. The core of the proof is an algebraic analysis that aligns with the intuition presented in the introduction: different commitments $C = yf_{\mathbf{S}}(v)\mathsf{G}$ on different attribute sets $\mathbf{S}$ cannot be meaningfully recombined because they are based on different scalars y. Ultimately, we show that the only way to produce a valid presentation is to use one of the previously issued credentials and reveal a subset of its attributes—i.e., not to break unforgeability.

Theorem 14. *If the NIZK has the soundness property (Definition 24),* $\mathsf{KVAC}_{\mathsf{GGM}}$ *is unlinkable (Definition 3).*

Both oracles, $\mathcal{O}_{\mathsf{Show}_0}(\mathbf{D})$ and $\mathcal{O}_{\mathsf{Show}_1}(\mathbf{D})$, output a witness W and a tag τ', where $W \in \mathbb{G}^*$ is uniformly random. The soundness of the NIZK proofs (included in $\mathsf{PreCred}_0$ and $\mathsf{PreCred}_1$) guarantees that both oracles produce a valid presentation Show, where the tag τ' is uniquely determined by the random W as $\tau' = xWf_{\mathbf{D}}(v)$. Therefore, as long as NIZK soundness holds, the outputs of both oracles are identically distributed. A detailed proof is provided in Appendix G.2.

7 Extension: blind issuance and non-transferability

Our KVAC definitions (and constructions) do not incorporate two features: *blind issuance* and *non-transferability*. This is not unusual (e.g., [CDDH19]) and serves readability and conceptual simplicity for our main contributions. In this section, we briefly discuss these features and sketch how they can be added to our constructions.

Blind issuance. allows users to obtain credentials without revealing (full information about) their attributes to the issuer [CMZ14, BBDE19]. In many scenarios (such as the building access control scenario in the introduction), this feature is unnecessary. Indeed, since the issuer is supposed to attest to the attributes, she usually wants to be aware of them.

Still, if blind issuance is required, it can be easily added to our constructions $\mathsf{KVAC}_{\mathsf{MEQ}}$ and $\mathsf{KVAC}_{\mathsf{GGM}}$ as follows. In both constructions, the issuer, given attributes $\mathbf{S}$, computes a set commitment to $\mathbf{S}$. To enable blind issuance, roughly speaking, the user can simply compute a randomized version of the set commitment, blinded by a random scalar $d \xleftarrow{\$} \mathbb{Z}_p^*$, namely $C = (df_{\mathbf{S}}(v)\mathsf{G}_1, d\mathsf{G}_1')$ for $\mathsf{KVAC}_{\mathsf{MEQ}}$ and $C = df_{\mathbf{S}}(v)\mathsf{G}$ for $\mathsf{KVAC}_{\mathsf{GGM}}$ and send it to the issuer. The issuer then proceeds with the given commitment in the natural way, i.e. the issuer computes an SP-MAC-EQ tag on C in $\mathsf{KVAC}_{\mathsf{MEQ}}$, or computes the tag as xyC in $\mathsf{KVAC}_{\mathsf{GGM}}$. Of course, the user should also include a non-interactive zero-knowledge proof to prove that the randomized commitment has been honestly generated and that the attributes in $\mathbf{S}$ follow the issuer's rules.

Non-transferability. ensures that users cannot transfer their credentials [CL01], either intentionally or unintentionally.

Unintentional transfer, such as replay attacks [FHS19], must crucially be defended against in anonymous credential systems with public verifiability: verifiers are not trusted

Table 2: Benchmarks of SPS-EQ (SEQ) and SP-MAC-EQ (MEQ). ℓ denotes the message length and the numbers represent mean execution time in milliseconds.

ℓ	2^1	2^3	2^5	2^7	2^9	2^{11}
SEQ.Sign	0.78	1.31	3.44	11.79	45.83	177.34
MEQ.MAC	0.66	1.17	3.21	11.31	43.83	173.90
SEQ.Verify	4.95	10.77	33.91	127.91	499.46	1993.97
MEQ.Verify	2.28	3.16	6.61	20.31	74.81	291.33

by issuers or users and must not be able to capture an honest user's credential or replay his messages to another verifier. In the keyed verification setting, however, verifiers inherently have the ability to create arbitrary credentials themselves. Hence there is no need to cryptographically guard against malicious verifiers capturing credentials or running a replay attack, as malicious verifiers can trivially authenticate anywhere.

In contrast, discouraging *intentional* transfer or sharing of credentials can be a worthwhile goal even for KVAC. For instance, a bus ticket credential may need to be bound to a specific user and that user should not share his ticket with other users. Intentional transfer can be discouraged by requiring users to provide an interactive zero-knowledge proof tied to their PKI secret key usk (which is valuable even outside the system) during the credential presentation phase, making sharing impractical [CL01].[8]

For the following discussion, we use G to refer to G_1 in $\mathsf{KVAC}_{\mathsf{MEQ}}$, allowing us to present the solutions for $\mathsf{KVAC}_{\mathsf{MEQ}}$ and $\mathsf{KVAC}_{\mathsf{GGM}}$ in a uniform way. Each user possesses a pair of secret and public keys $(\mathsf{usk} \xleftarrow{\$} \mathbb{Z}_p^*, \mathsf{upk} := \mathsf{usk}\mathsf{G}')$, where usk is some valuable secret. The idea is that the issuer authenticates upk alongside the set commitment when issuing a credential. For $\mathsf{KVAC}_{\mathsf{MEQ}}$, this means that instead of sending a MAC on (C, G'), the issuer sends a MAC on $(C, \mathsf{G}', \mathsf{upk})$. For $\mathsf{KVAC}_{\mathsf{GGM}}$, this means that instead of sending a homomorphic MAC, xC, on C, the issuer sends a homomorphic MAC, $x_1C + x_2\mathsf{G}' + x_3\mathsf{upk}$, on $(C, \mathsf{G}', \mathsf{upk})$. When presenting the credential, the user reveals $(\mu C, \mu\mathsf{G}', \mu\mathsf{upk})$ for random $\mu \xleftarrow{\$} \mathbb{Z}_p^*$ together with a adapted MAC tag, and proves (in zero-knowledge) that he knows usk such that $\mathsf{usk}(\mu\mathsf{G}') = \mu\mathsf{upk}$. This ensures that anyone *using* the credential must know the credential owner's valuable secret key, disincentivizing sharing of the credential. We leave details for future work.

8 Performance Evaluation

This section presents (1) benchmarks for SP-MAC-EQ and SPS-EQ across various ℓ (message lengths) and (2) benchmarks for the KVAC protocols, $\mathsf{KVAC}_{\mathsf{MEQ}}$ and $\mathsf{KVAC}_{\mathsf{GGM}}$. We implemented SP-MAC-EQ, SPS-EQ, DVSC, $\mathsf{KVAC}_{\mathsf{MEQ}}$, and $\mathsf{KVAC}_{\mathsf{GGM}}$ in pure Rust, with all implementations available open source [Ben25]. For elliptic curve operations, we used Arkworks [Ark22]. All experiments were conducted on a MacBook Pro (2021) with an Apple M1 Max processor (10 cores: 8 performance and 2 efficiency) and 64 GB RAM.

8.1 SP-MAC-EQ vs. SPS-EQ

In Appendix H, we compare all the algorithms in SPS-EQ and SP-MAC-EQ. The MAC operation in SP-MAC-EQ is slightly more efficient than the Sign operation in SPS-EQ due to one fewer exponentiation. However, the Verify algorithm of SP-MAC-EQ ($\mathsf{MEQ.Verify}$) is significantly more efficient than that of SPS-EQ ($\mathsf{SEQ.Verify}$) because $\mathsf{SEQ.Verify}$ requires $\ell + 3$ pairing operations, which scales linearly with ℓ, whereas $\mathsf{MEQ.Verify}$ always requires

[8] Note that we can never prevent a user from relaying communication between a credential-holding user and the verifier, allowing use of another willing user's credential without sharing the secret key. This issue, known as a "terrorist fraud attack", requires distance bounding protocols for mitigation [Vau13].

Table 3: Total execution time of ObtainCred (user)/IssueCred (issuer), and size of credentials Cred.

Input Size	User/Issuer time (ms)		Credential (KiB)	
$\|\mathbf{S}\|$	$\mathsf{KVAC}_{\mathsf{GGM}}$	$\mathsf{KVAC}_{\mathsf{MEQ}}$	$\mathsf{KVAC}_{\mathsf{GGM}}$	$\mathsf{KVAC}_{\mathsf{MEQ}}$
2^4	3.80/2.84	3.21/3.58	0.56	0.14
2^6	14.70/10.27	7.97/8.26	2.06	0.14
2^8	61.18/40.77	27.87/28.46	8.06	0.14
2^{10}	253.89/159.91	121.95/121.88	32.06	0.14
2^{12}	1232.62/636.01	718.68/719.28	128.06	0.14

Table 4: Total execution time of ShowCred (user)/Verify (verifier), and size of presentation tokens Show.

Input Size	User/Verifier time (ms)		Presentation (KiB)	
$(\|\mathbf{S}\|, \|\mathbf{D}\|)$	$\mathsf{KVAC}_{\mathsf{GGM}}$	$\mathsf{KVAC}_{\mathsf{MEQ}}$	$\mathsf{KVAC}_{\mathsf{GGM}}$	$\mathsf{KVAC}_{\mathsf{MEQ}}$
$(2^4, 2^3)$	0.72/0.06	3.26/2.18	0.06	0.28
$(2^6, 2^5)$	2.52/0.06	10.35/2.19	0.06	0.28
$(2^8, 2^7)$	9.89/0.07	40.06/2.19	0.06	0.28
$(2^{10}, 2^9)$	42.99/0.07	176.43/2.20	0.06	0.28
$(2^{12}, 2^{11})$	227.51/0.09	997.21/2.21	0.06	0.28

only 2 pairings regardless of the message length. We implemented both SPS-EQ and SP-MAC-EQ in Rust using the BLS12-381 elliptic curve and benchmarked them for varying values of ℓ. The results, presented in Table 2, confirm the superior efficiency of SP-MAC-EQ. Additionally, our implementations of SPS-EQ and SP-MAC-EQ may be of independent interest.

8.2 $\mathsf{KVAC}_{\mathsf{MEQ}}$ vs. $\mathsf{KVAC}_{\mathsf{GGM}}$

We benchmarked $\mathsf{KVAC}_{\mathsf{MEQ}}$ and $\mathsf{KVAC}_{\mathsf{GGM}}$ for various values of $(|\mathbf{S}|, |\mathbf{D}|)$ using the BLS12-381 and Ed25519 elliptic curves, respectively.[9] The results are presented in Tables 3 and 4. Note that the NIZK verification described in Section 4.2 was excluded from our benchmarks, as it needs to be executed only once by any party.

IssueCred *and* ObtainCred *(Table 3).* This table compares the efficiency of the two systems during the issuing phase. In $\mathsf{KVAC}_{\mathsf{GGM}}$, the issuer must send additional information to the user that enables the user to compute the commitment and corresponding openings for future presentations. The size of this extra information grows linearly with the number of attributes. Moreover, the user in $\mathsf{KVAC}_{\mathsf{GGM}}$ performs more computations than in $\mathsf{KVAC}_{\mathsf{MEQ}}$, as they must verify a proof of well-formedness for the additional information received from the issuer. As a result, $\mathsf{KVAC}_{\mathsf{GGM}}$ incurs higher execution time and produces larger credential sizes. Note that this phase does not involve any pairings (in neither construction).

ShowCred *and* Verify *(Table 4).* This table presents a comparison of the two systems in terms of efficiency during the presentation phase. $\mathsf{KVAC}_{\mathsf{GGM}}$ outperforms $\mathsf{KVAC}_{\mathsf{MEQ}}$ in both execution time and presentation size. This improvement is primarily attributed to the removal of pairing operations in the verification process and the simplification of the verification equation to a single exponentiation. As a result, the verifier's execution time in $\mathsf{KVAC}_{\mathsf{GGM}}$ falls within the range of 0.06 ms to 0.09 ms. Additionally, the user in $\mathsf{KVAC}_{\mathsf{GGM}}$ sends fewer group elements to the verifier, further contributing to the reduced presentation size.

[9] In Appendix I, we compared the implementation of $\mathsf{KVAC}_{\mathsf{GGM}}$ using elliptic curves Ed25519, Secp256k1, and BLS12-381. As $\mathsf{KVAC}_{\mathsf{GGM}}$ does not require pairing, it benefits from curves with faster group operations.

Takeaways. $\mathsf{KVAC}_{\mathsf{GGM}}$ is ideal for applications prioritizing user and verifier efficiency, such as mobile environments or scenarios with frequent credential representations. $\mathsf{KVAC}_{\mathsf{MEQ}}$ is better suited for high-volume credential issuance or storage-constrained systems where compact credentials are essential.

9 Conclusion and Future Work

In this paper, we present two KVAC systems: one pairing-based system leveraging our proposed SP-MAC-EQ and DVSC schemes, which achieves both constant-size credentials and constant-size presentation; and one pairingless system, which only achieves constant-size presentation. A promising avenue for future research is exploring the possibility of a pairingless SP-MAC-EQ, enabling a pairingless construction that achieves both constant-size credentials and constant-size presentation. Alternatively, establishing an impossibility proof to demonstrate that SP-MAC-EQ without pairing is unattainable would also be valuable.

An interesting feature of our constructions is that Show tokens (1) only suffice to prove possession of a specific set $\mathbf{D}$ (not any $\mathbf{D}' \supset \mathbf{D}$), and (2) can be randomized into unlinkable versions. This naturally enables delegation capabilities: The holder of a credential for attributes $\mathbf{S}$ can delegate some of his attributes $\mathbf{D} \subseteq \mathbf{S}$ to someone else by revealing Show, which can then be used to authenticate, unlinkably, with attributes $\mathbf{D}$. This insight leads to two open questions. First, are there applications for this delegation feature, and can it be extended to some notion of *delegatable KVAC* (e.g., allowing the holder of Show token to delegate some further subset $\mathbf{D}' \subset \mathbf{D}$)? Second, given that one can view our KVAC constructions' presentation strategy as delegating a subset of attributes to the verifier, can one construct KVAC from delegatable primitives (such as, say, certain puncturable PRFs or attribute-based encryption)?

Acknowledgments

This work was supported by the Flemish Government through the Cybersecurity Research Program with grant number: VOEWICS02 and through SolidLab Flanders (Flemish Government, EWI).

References

[AC20] Thomas Attema and Ronald Cramer. Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 513–543. Springer, Cham, August 2020. doi:10.1007/978-3-030-56877-1_18.

[AGHO11] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 649–666. Springer, Berlin, Heidelberg, August 2011. doi:10.1007/978-3-642-22792-9_37.

[Ark22] Arkworks contributors. `arkworks` zksnark ecosystem, 2022. URL: https://arkworks.rs.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008. doi:10.1007/s00145-007-9005-7.

[BBDE19] Johannes Blömer, Jan Bobolz, Denis Diemert, and Fabian Eidens. Updatable anonymous credentials and applications to incentive systems. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 1671–1685. ACM Press, November 2019. doi:10.1145/3319535.3354223.

[BBDT16] Amira Barki, Solenn Brunet, Nicolas Desmoulins, and Jacques Traoré. Improved algebraic MACs and practical keyed-verification anonymous credentials. In Roberto Avanzi and Howard M. Heys, editors, *SAC 2016*, volume 10532 of *LNCS*, pages 360–380. Springer, Cham, August 2016. doi:10.1007/978-3-319-69453-5_20.

[BBG05a] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. Cryptology ePrint Archive, Report 2005/015, 2005. URL: https://eprint.iacr.org/2005/015.

[BBG05b] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, pages 440–456, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/11426639_26.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Berlin, Heidelberg, August 2004. doi:10.1007/978-3-540-28628-8_3.

[BEK+20] Jan Bobolz, Fabian Eidens, Stephan Krenn, Daniel Slamanig, and Christoph Striecks. Privacy-preserving incentive systems with highly efficient point-collection. In Hung-Min Sun, Shiuh-Pyng Shieh, Guofei Gu, and Giuseppe Ateniese, editors, *ASIACCS 20*, pages 319–333. ACM Press, October 2020. doi:10.1145/3320269.3384769.

[Ben25] Emad Heydari Beni. KVACs, SPS-EQ and SP-MAC-EQ Implementations. Github repository, 2025. https://github.com/emad7105/sp-mac-eq-kvac. URL: https://github.com/emad7105/sp-mac-eq-kvac.

[BF20] Balthazar Bauer and Georg Fuchsbauer. Efficient signatures on randomizable ciphertexts. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 359–381. Springer, Cham, September 2020. doi:10.1007/978-3-030-57990-6_18.

[BFHK24] Balthazar Bauer, Pooya Farshim, Patrick Harasser, and Markulf Kohlweiss. The uber-knowledge assumption: A bridge to the AGM. *CiC*, 1(3):31, 2024. doi:10.62056/anr-zoja5.

[BFR24a] Balthazar Bauer, Georg Fuchsbauer, and Fabian Regen. On proving equivalence class signatures secure from non-interactive assumptions. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part I*, volume 14601 of *LNCS*, pages 3–36. Springer, Cham, April 2024. doi:10.1007/978-3-031-57718-5_1.

[BFR24b] Balthazar Bauer, Georg Fuchsbauer, and Fabian Regen. On security proofs of existing equivalence class signature schemes. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 3–37. Springer, Singapore, December 2024. doi:10.1007/978-981-96-0888-1_1.

[BFR25] Balthazar Bauer, Georg Fuchsbauer, and Fabian Regen. On security proofs of existing equivalence class signature schemes. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024*, pages 3–37, Singapore, 2025. Springer Nature Singapore. doi:10.1007/978-981-96-0888-1_1.

[BFS16] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, Berlin, Heidelberg, December 2016. doi:10.1007/978-3-662-53890-6_26.

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 408–425. Springer, Berlin, Heidelberg, August 2014. doi:10.1007/978-3-662-44371-2_23.

[BL13] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 1087–1098. ACM Press, November 2013. doi:10.1145/2508859.2516687.

[BSW24] Christian Badertscher, Mahdi Sedaghat, and Hendrik Waldner. Unlinkable policy-compliant signatures for compliant and decentralized anonymous payments. *Proc. Priv. Enhancing Technol.*, 2024(4):226–267, 2024. URL: https://doi.org/10.56553/popets-2024-0115, doi:10.56553/POPETS-2024-0115.

[CAHLT25] Rutchathon Chairattana-Apirom, Franklin Harding, Anna Lysyanskaya, and Stefano Tessaro. Server-aided anonymous credentials. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025*, pages 291–324, Cham, 2025. Springer Nature Switzerland. doi:10.1007/978-3-032-01887-8_10.

[CDDH19] Jan Camenisch, Manu Drijvers, Petr Dzurenda, and Jan Hajny. Fast keyed-verification anonymous credentials on standard smart cards. In Gurpreet Dhillon, Fredrik Karlsson, Karin Hedström, and André Zúquete, editors, *ICT Systems Security and Privacy Protection - 34th IFIP TC 11 International Conference, SEC 2019, Lisbon, Portugal, June 25-27, 2019, Proceedings*, volume 562 of *IFIP Advances in Information and Communication Technology*, pages 286–298. Springer, 2019. doi:10.1007/978-3-030-22312-0_20.

[CDHK15] Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 262–288. Springer, Berlin, Heidelberg, November / December 2015. doi:10.1007/978-3-662-48800-3_11.

[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. doi:10.1007/978-1-4757-0602-4_18.

[CKP+23] Elizabeth C. Crites, Markulf Kohlweiss, Bart Preneel, Mahdi Sedaghat, and Daniel Slamanig. Threshold structure-preserving signatures. In Jian Guo and

Ron Steinfeld, editors, *ASIACRYPT 2023, Part II*, volume 14439 of *LNCS*, pages 348–382. Springer, Singapore, December 2023. doi:10.1007/978-981-99-8724-5_11.

[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. doi:10.1007/3-540-44987-6_7.

[CL03] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, Berlin, Heidelberg, September 2003. doi:10.1007/3-540-36413-7_20.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Berlin, Heidelberg, August 2004. doi:10.1007/978-3-540-28628-8_4.

[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. doi:10.1007/978-3-030-12612-4_27.

[CLPK22] Aisling Connolly, Pascal Lafourcade, and Octavio Perez-Kempner. Improved constructions of anonymous credentials from structure-preserving signatures on equivalence classes. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part I*, volume 13177 of *LNCS*, pages 409–438. Springer, Cham, March 2022. doi:10.1007/978-3-030-97121-2_15.

[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. doi:10.1145/2660267.2660328.

[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. doi:10.1145/3372297.3417887.

[CR19] Geoffroy Couteau and Michael Reichle. Non-interactive keyed-verification anonymous credentials. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part I*, volume 11442 of *LNCS*, pages 66–96. Springer, Cham, April 2019. doi:10.1007/978-3-030-17253-4_3.

[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. doi:10.1007/BFb0052252.

[DDKT25] Nicolas Desmoulins, Antoine Dumanois, Seyni Kane, and Jacques Traoré. Making BBS anonymous credentials eIDAS 2.0 compliant. Cryptology ePrint Archive, Paper 2025/619, 2025. URL: https://eprint.iacr.org/2025/619.

[DKPW12] Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak, and Daniel Wichs. Message authentication, revisited. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 355–374. Springer, Berlin, Heidelberg, April 2012. doi:10.1007/978-3-642-29011-4_22.

[FG18] Georg Fuchsbauer and Romain Gay. Weakly secure equivalence-class signatures from standard assumptions. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 153–183. Springer, Cham, March 2018. doi:10.1007/978-3-319-76581-5_6.

[FHS19] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546, April 2019. doi:10.1007/s00145-018-9281-4.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987. doi:10.1007/3-540-47721-7_12.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008. doi:10.1016/j.dam.2007.12.010.

[HS14] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 491–511. Springer, Berlin, Heidelberg, December 2014. doi:10.1007/978-3-662-45611-8_26.

[HS21] Lucjan Hanzlik and Daniel Slamanig. With a little help from my friends: Constructing practical anonymous credentials. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2004–2023. ACM Press, November 2021. doi:10.1145/3460120.3484582.

[Klo21] Michael Klooß. On expected polynomial runtime in cryptography. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part I*, volume 13042 of *LNCS*, pages 558–590. Springer, Cham, November 2021. doi:10.1007/978-3-030-90459-3_19.

[KSD19] Mojtaba Khalili, Daniel Slamanig, and Mohammad Dakhilalian. Structure-preserving signatures on equivalence classes from standard assumptions. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 63–93. Springer, Cham, December 2019. doi:10.1007/978-3-030-34618-8_3.

[LMPY16] Benoît Libert, Fabrice Mouhartem, Thomas Peters, and Moti Yung. Practical "signatures with efficient protocols" from simple assumptions. In Xiaofeng Chen, XiaoFeng Wang, and Xinyi Huang, editors, *ASIACCS 16*, pages 511–522. ACM Press, May / June 2016. doi:10.1145/2897845.2897898.

[LPS23] Helger Lipmaa, Roberto Parisella, and Janno Siim. Algebraic group model with oblivious sampling. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 363–392. Springer, Cham, November / December 2023. doi:10.1007/978-3-031-48624-1_14.

[Mau05] Ueli Maurer. Abstract models of computation in cryptography. In *Cryptography and Coding: 10th IMA International Conference, Cirencester, UK, December 19-21, 2005. Proceedings 10*, pages 1–12. Springer, 2005. doi:10.1007/11586821_1.

[Mau09] Ueli M. Maurer. Unifying zero-knowledge proofs of knowledge. In Bart Preneel, editor, *AFRICACRYPT 09*, volume 5580 of *LNCS*, pages 272–286. Springer, Berlin, Heidelberg, June 2009. doi:10.1007/978-3-642-02384-2_17.

[Ngu05] Lan Nguyen. Accumulators from bilinear pairings and applications. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 275–292. Springer, 2005. doi:10.1007/978-3-540-30574-3_19.

[Orr24] Michele Orrù. Revisiting keyed-verification anonymous credentials. Cryptology ePrint Archive, Paper 2024/1552, 2024. URL: https://eprint.iacr.org/2024/1552.

[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. doi:10.1007/978-3-031-47818-5_10.

[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. doi:10.1007/978-3-319-29485-8_7.

[San20] Olivier Sanders. Efficient redactable signature and application to anonymous credentials. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 628–656. Springer, Cham, May 2020. doi:10.1007/978-3-030-45388-6_22.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, New York, August 1990. doi:10.1007/0-387-34805-0_22.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_18.

[TG23] Lindsey Tulloch and Ian Goldberg. Lox: Protecting the social graph in bridge distribution. *PoPETs*, 2023(1):494–509, January 2023. doi:10.56553/popets-2023-0029.

[Vau13] Serge Vaudenay. On modeling terrorist frauds. In Willy Susilo and Reza Reyhanitabar, editors, *Provable Security*, pages 1–20, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-41227-1_1.

A Omitted Definitions

A.1 Assumptions

The decisional Diffie-Hellman (DDH) assumption is as follows.

Definition 14 (Decisional Diffie-Hellman (DDH) Assumption)**.** Let $\mathcal{BG}$ be a group parameter generator for prime order groups. The DDH assumption holds if for all PPT adversaries $\mathcal{A}$, there is a negligible function negl such that

$$Adv_{\mathcal{A}}^{DDH} \leq |\Gamma_0 - \Gamma_1| \leq \mathsf{negl}(\lambda) \ ,$$

for $\Gamma_b := \Pr[\mathcal{A}(\mathbb{G}, p, \mathsf{G}, x\mathsf{G}, y\mathsf{G}, (xy + bz)\mathsf{G}) = 1]$, where the probability is over $(\mathbb{G}, p, \mathsf{G}) \leftarrow \mathcal{BG}(1^\lambda)$, $x, y, z \xleftarrow{\$} \mathbb{Z}_p^*$, and the random coins of $\mathcal{A}$.

Note that DDH holds in the generic group model if $p = |\mathbb{G}|$ increases superpoly with λ. We also use the t-co-DL assumption.

Definition 15 (t-co-DL assumption)**.** Let $\mathcal{BG}$ be a prime order bilinear group parameter generator. Let $t \in \mathbb{N}$. The t-co-DL assumption holds if for all PPT adversaries $\mathcal{A}$:

$$\Pr\begin{bmatrix} \mathsf{BG} \leftarrow \mathcal{BG}(1^\lambda);\ a \xleftarrow{\$} \mathbb{Z}_p; \\ a' \leftarrow \mathcal{A}\big(\mathsf{BG}, (a^i\mathsf{G}_1, a^i\mathsf{G}_2)_{i\in[t]}\big) \end{bmatrix} : a' = a \leq \mathsf{negl}(\lambda) \ .$$

Note that t-co-DL holds in the generic group model if $\mathcal{BG}$ outputs groups of superpoly size.

A.2 Message Authentication Code (MAC)

A Message Authentication Code (MAC) is a cryptographic primitive used to ensure both the authenticity and integrity of a message. It enables a party who knows a secret key to generate a tag for a given message, and another (or the same) party who knows the same secret key can verify whether the message has been altered or tampered with.

Definition 16 (Message Authentication Code (MAC) [DKPW12])**.** A MAC scheme consists of the following four algorithms:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: A probabilistic algorithm that, on input of a security parameter λ, outputs the public parameters pp.
- $\mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp})$: A probabilistic algorithm that takes the public parameters pp and generates a secret key sk.
- $\tau \leftarrow \mathsf{MAC}(\mathsf{pp}, \mathsf{sk}, m)$: A probabilistic algorithm that takes as input the public parameters pp, the secret key sk, and a message m, and outputs a tag τ.
- $0/1 \leftarrow \mathsf{Verify}(\mathsf{pp}, \mathsf{sk}, \tau, m)$: A deterministic algorithm that takes as input the public parameters pp, the secret key sk, a tag τ, and a message m, and outputs 1 if the tag τ is valid for m, and 0 otherwise.

Security Properties. A MAC achieves two security properties; correctness and unforgeability against chosen message and verification attack (UF-CMVA).

Definition 17 (Correctness)**.** A MAC guarantees the correctness if for all λ and $m \in \mathcal{M}$, we have:

$$\Pr\begin{bmatrix} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}); \\ \tau \leftarrow \mathsf{MAC}(\mathsf{pp}, \mathsf{sk}, m); \\ b = \mathsf{Verify}(\mathsf{pp}, \mathsf{sk}, \tau, m); \end{bmatrix} : b = 1 = 1 \ .$$

Definition 18 (UF-CMVA)**.** A MAC achieves UF-CMVA, if for all adversaries $\mathcal{A}$ who can make Q_T queries to $\mathcal{O}_{\mathsf{MAC}}(\cdot)$ and Q_V queries to $\mathcal{O}_{\mathsf{Verify}}(\cdot)$ we have:

$$\Pr\left[\begin{array}{ll} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); & b = 1 \wedge \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}); & m^* \notin \mathcal{Q}^{\mathsf{MAC}} \\ (\tau^*, m^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{MAC}}(\cdot),\mathcal{O}_{\mathsf{Verify}}(\cdot,\cdot)}(\mathsf{pp}); & : |Q^{\mathsf{MAC}}| \leq Q_T \wedge \\ b = \mathsf{Verify}(\mathsf{pp}, \mathsf{sk}, \tau^*, m^*) & |Q^{\mathsf{Ver}}| \leq Q_V \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where the oracles are defined as follows:

- $\mathcal{O}_{\mathsf{MAC}}(m)$: Initialize $Q^{\mathsf{MAC}} = \emptyset$. Given m, run $\mathsf{MAC}(\mathsf{pp}, \mathsf{sk}, m)$. Return τ and update $Q^{\mathsf{MAC}} = Q^{\mathsf{MAC}} \cup \{m\}$.
- $\mathcal{O}_{\mathsf{Verify}}(m, \tau)$: Initialize $Q^{\mathsf{Ver}} = \emptyset$ s.t. $Q_V := |Q^{\mathsf{Ver}}|$. Given message m and tag τ run $\mathsf{Verify}(\mathsf{pp}, \mathsf{sk}, \tau, m)$. Return 1 (accept) or 0 (reject) and update $Q^{\mathsf{Ver}} = Q^{\mathsf{Ver}} \cup \{(m, \tau)\}$.

A.3 Structure-Preserving Signatures on Equivalence Classes

Definition 19 (Structure-Preserving Signatures on Equivalence classes [HS14])**.** Given an asymmetric bilinear group and the relation described in Equation (1), a SPS-EQ over message space $\mathcal{M} := (\mathbb{G}_i^*)^\ell$ consists of the following PPT algorithms:

- $\mathsf{pp} \leftarrow \mathsf{Setup}_{\mathcal{R}}(1^\lambda)$: A probabilistic algorithm that takes the security parameter λ in its unary representation as input, and outputs public parameters pp.
- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}_{\mathcal{R}}(\mathsf{pp}, \ell)$: A probabilistic algorithm that takes the public parameters pp and a vector length $\ell > 1$ as inputs, and outputs the key-pair $(\mathsf{sk}, \mathsf{vk})$.
- $\sigma \leftarrow \mathsf{Sign}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}, \mathbf{M})$: A probabilistic algorithm that takes public parameters pp, secret key sk and a representative message $\mathbf{M} \in \mathcal{M}$ for class $[\mathbf{M}]_{\mathcal{R}}$ as inputs. It outputs the signature σ on message $\mathbf{M}$.
- $0/1 \leftarrow \mathsf{Verify}_{\mathcal{R}}(\mathsf{pp}, \mathsf{vk}, \mathbf{M}, \sigma)$: A deterministic algorithm that takes public parameters pp, a verification key vk, representative message $\mathbf{M} \in (\mathbb{G}_i^*)^\ell$, a signature σ as inputs, and outputs 1 if the signature σ is valid for $\mathbf{M}$, and 0 otherwise.
- $(\sigma', \mathbf{M}') \leftarrow \mathsf{ChgRep}_{\mathcal{R}}(\mathsf{pp}, \mathbf{M}, \sigma, \mu, \mathsf{vk})$: The change representation algorithm is a probabilistic algorithm and takes public parameters pp, a representative message $\mathbf{M} \in (\mathbb{G}_i^*)^\ell$, a signature σ, a scalar $\mu \in \mathbb{Z}_p^*$ and the verification key vk as inputs. It outputs a randomized signature σ' on a new representative message $\mathbf{M}' = \mu\mathbf{M}$.

Security Properties. The primary security requirements for a SPS-EQ scheme are *correctness* and *existential unforgeability against chosen message attack*, which are defined as follows:

Definition 20 (Correctness)**.** A SPS-EQ scheme over $\mathcal{M}$ is called *correct*, if for a valid setup pp, any message $\mathbf{M} \in \mathcal{M}$, any (valid) key pair $(\mathsf{sk}, \mathsf{vk})$ in the support of $\mathsf{KeyGen}_{\mathcal{R}}(\mathsf{pp}, \ell)$, and any scalar $\mu \in \mathbb{Z}_p^*$, we have:

$$\Pr\left[\begin{array}{l} \mathsf{Verify}_{\mathcal{R}}\left(\mathsf{pp}, \mathsf{vk}, \mathbf{M}, \mathsf{Sign}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}, \mathbf{M})\right) = 1 \wedge \\ \mathsf{Verify}_{\mathcal{R}}(\mathsf{pp}, \mathsf{vk}, \mu\mathbf{M}, \mathsf{ChgRep}_{\mathcal{R}}(\mathbf{M}, \mathsf{Sign}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}, \mathbf{M}), \mu, \mathsf{vk})) = 1 \end{array}\right] = 1 \ .$$

Definition 21 (Existential Unforgeability)**.** A SPS-EQ over $\mathcal{M}$ is called adaptively $\mathsf{EUF\text{-}CMA}$-secure if for all PPT adversaries $\mathcal{A}$ with access to the signing oracle $\mathcal{O}_{\mathsf{Sign}}(\cdot)$ we have:

$$\Pr\left[\begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Setup}_{\mathcal{R}}(1^\lambda), (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}_{\mathcal{R}}(\mathsf{pp}, \ell), \\ (\mathbf{M}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}(\cdot)}(\mathsf{pp}, \mathsf{vk}) : \\ \mathbf{M}^* \notin \mathcal{Q}^{\mathsf{Sign}} \ \wedge \ \mathsf{Verify}_{\mathcal{R}}\left(\mathsf{pp}, \mathsf{vk}, \mathbf{M}, \sigma^*\right) = 1\end{array}\right] \leq \mathsf{negl}(\lambda) \ ,$$

where the signing oracle $\mathcal{O}_{\mathsf{Sign}}(\cdot)$ takes a message $\mathbf{M} \in \mathcal{M}$ as input, outputs $\mathsf{Sign}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}, \mathbf{M})$ and updates the query set $\mathcal{Q}^{\mathsf{Sign}} = \mathcal{Q}^{\mathsf{Sign}} \cup \{[\mathbf{M}]_{\mathcal{R}}\}$.

Additionally, as discussed in Section 3, similar to SP-MAC-EQ, an SPS-EQ achieves Class-Hiding (cf. Definition 7) and Perfect Adaptation (cf. Definition 8).

A.4 NIZK Definitions

We define non-interactive zero-knowledge proofs of knowledge as follows.

Definition 22 (NIZK)**.** A non-interactive zero-knowledge proof (NIZK) for the relation R_{pp} and random oracle $\mathcal{O}_{\mathsf{RO}}$ is a triple of PPT algorithms $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: Takes as input the unary representation of the security parameter λ and outputs public parameters pp. We require $|\mathsf{pp}| \geq \lambda$.
- $\pi \leftarrow \mathsf{Prove}^{\mathcal{O}_{\mathsf{RO}}}(x, w)$: Takes as input a statement x and a witness w, and outputs a proof π.
- $\{0, 1\} \leftarrow \mathsf{Verify}^{\mathcal{O}_{\mathsf{RO}}}(x, \pi)$: Takes as input a statement x and a proof π, and outputs 0 or 1.

Security Properties. A system NIZK is complete/correct if for any $(x, w) \in R_{\mathsf{pp}}$, proofs π computed by $\mathsf{Prove}^{\mathcal{O}_{\mathsf{RO}}}(x, w)$ are accepted by the verifier, i.e. $\mathsf{Verify}^{\mathcal{O}_{\mathsf{RO}}}(x, \pi) = 1$. This must hold even in the presence of a PPT adversary making calls to $\mathcal{O}_{\mathsf{RO}}$.

The setup procedure, Setup, serves to bring asymptotic security into the NIZK definition. In our cases, Setup will output a (bilinear) group of sufficiently large order as pp. Both the random oracle $\mathcal{O}_{\mathsf{RO}}$ and the relation R_{pp} are parameterized with pp. The random oracle $\mathcal{O}_{\mathsf{RO}}$ in this definition could also be replaced by another oracle to generically model different setups in which the NIZK may function (e.g., an oracle that generates a common reference string and returns it upon request). When using a NIZK in our constructions, we omit pp and the random oracle from the notation.

Definition 23 (Zero-Knowledge)**.** A NIZK for relation R_{pp} and random oracle $\mathcal{O}_{\mathsf{RO}}$ is *zero-knowledge* if there exists a stateful PPT simulator Sim with procedures $\mathsf{Sim}.\mathcal{O}_{\mathsf{RO}}$ and $\mathsf{Sim.Prove}$ such that for all PPT adversaries $\mathcal{A}$,

$$\left|\Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{RO}}, \mathcal{O}_{\mathsf{Prove}}}(\mathsf{pp}) = 1] - \Pr[\mathcal{A}^{\mathsf{Sim.RO}, \mathcal{O}_{\mathsf{Sim}}}(\mathsf{pp}) = 1]\right| \leq \mathsf{negl}(\lambda) \ ,$$

where the randomness is over $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and the randomness of $\mathcal{A}, \mathsf{Sim}, \mathcal{O}_{\mathsf{RO}}, \mathcal{O}_{\mathsf{Prove}}$. Before the experiment begins, Sim is given pp as input. The oracle $\mathcal{O}_{\mathsf{Sim}}(x, w)$ checks that $(x, w) \in R_{\mathsf{pp}}$ and if so, outputs $\pi \leftarrow \mathsf{Sim.Prove}(x)$. The oracle $\mathcal{O}_{\mathsf{Prove}}(x, w)$ simply runs the NIZK's $\mathsf{Prove}(\mathsf{pp}, x, w)$ algorithm.

In the zero-knowledge definition, the simulator gets to take over the random oracle $\mathcal{O}_{\mathsf{RO}}$ and needs to create convincing proofs without being given the witness w.

Definition 24 (Soundness)**.** A NIZK for relation R_{pp} and random oracle $\mathcal{O}_{\mathsf{RO}}$ is *sound* if for all PPT adversaries $\mathcal{A}$,

$$\Pr\begin{bmatrix}\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (x,\pi) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{RO}}}(\mathsf{pp})\end{bmatrix} : \begin{matrix}\mathsf{Verify}^{\mathcal{O}_{\mathsf{RO}}}(x,\pi) = 1 \wedge \\ \nexists\, w :\ (x,w) \in R_{\mathsf{pp}}\end{matrix}\Bigg] \leq \mathsf{negl}(\lambda)\ .$$

The soundness definition guarantees that it is difficult for an adversary to compute a valid proof for a false statement. A stronger property is the proof of knowledge property, which says that additionally, a valid witness can be efficiently *extracted* from a successful prover.

Definition 25 (Proof of Knowledge)**.** A NIZK for relation R_{pp} and random oracle $\mathcal{O}_{\mathsf{RO}}$ is a *proof of knowledge* if for all PPT adversaries $\mathcal{A}, \mathsf{Setup}'$, there exists an expected polynomial-time extractor Ext such that

$$\Pr\begin{bmatrix}\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ aux \leftarrow \mathsf{Setup}'(\mathsf{pp}) \\ (x,\pi) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{RO}}}(\mathsf{pp}, aux)\end{bmatrix} : \begin{matrix}\mathsf{Verify}^{\mathcal{O}_{\mathsf{RO}}}(x,\pi) = 1 \wedge \\ (x, \mathsf{Ext}(\mathsf{pp}, \mathsf{view}_{\mathcal{A}})) \notin R_{\mathsf{pp}}\end{matrix}\Bigg]\ ,$$

is negligible in λ.

Note that $\mathsf{Ext}.\mathsf{Extract}$ can use rewinding techniques: it is defined depending on $\mathcal{A}$ (i.e. it knows its code), and it is given the adversary's view $\mathsf{view}_{\mathcal{A}}$, hence it can replay alternative challenges to an internally run copy of $\mathcal{A}$. The additional adversary Setup' that outputs *aux* is to model additional input to the adversary $\mathcal{A}$, in our case, Setup' will generate parameters input to $\mathcal{A}$ in KVAC or DVSC games. The extractor runs in expected polynomial time instead of strict polynomial time, which introduces some minor technical challenges when using it in security proofs. We disregard these challenges in our proofs for the sake of brevity but refer to [Klo21] for ways of handling them.

A.5 Implementation of our NIZKs

We build upon established methods known as sigma protocols, originally developed by Schnorr [Sch90], and incorporate newer approaches from more recent research by Maurer [Mau09]. To leverage non-interactive versions of these schemes, we apply the Fiat-Shamir technique [FS87], which uses a hash function $\mathcal{O} = \mathcal{H}$ that maps arbitrary binary strings to random numbers into the field $\mathbb{Z}_p$.

Implementation of the Sigma protocol in Figure 1. We implement

$$\begin{aligned}\pi \leftarrow \mathsf{PoK}\{(a^{-1}, x_1, x_2, r)\ | & X = x_1\mathsf{G}_1 + x_2\mathsf{G}'_1 + r\mathsf{G}''_1 \\ & \wedge T = a^{-1}\mathsf{G}_2 \wedge x_1 C_1 + x_2 C_2 = a^{-1}R\}\end{aligned}$$

in Figure 1 as follows. For given $\mathsf{G}_1, \mathsf{G}'_1, \mathsf{G}''_1, X, C_1, C_2, \mathsf{G}_2, R$ from the context of Figure 1, let $\phi(a^{-1}, x_1, x_2, r) \coloneqq (x_1\mathsf{G}_1 + x_2\mathsf{G}'_1 + r\mathsf{G}''_1, a^{-1}\mathsf{G}_2, x_1C_1 + x_2C_2 - a^{-1}R)$ (note that a^{-1} here is just a name for the first input to ϕ, no inversion is involved in computing ϕ for any given inputs). The prover chooses random $r_{\mathrm{a}^{-1}}, r_{\mathrm{x}_1}, r_{\mathrm{x}_2}, r_{\mathrm{r}} \xleftarrow{\$} \mathbb{Z}_p$ and computes Sigma protocol announcement $A = \phi(r_{\mathrm{a}^{-1}}, r_{\mathrm{x}_1}, r_{\mathrm{x}_2}, r_{\mathrm{r}}) \in \mathbb{G}^3$. It then computes Fiat-Shamir challenge $c = \mathcal{H}(\mathbf{S}, X, C_1, C_2, R, T, A) \in \mathbb{Z}_p$. It then computes the response $(s_{\mathrm{a}^{-1}}, s_{\mathrm{x}_1}, s_{\mathrm{x}_2}, s_{\mathrm{r}}) = (r_{\mathrm{a}^{-1}} + ca^{-1}, r_{\mathrm{x}_1} + cx_1, r_{\mathrm{x}_2} + cx_2, r_{\mathrm{r}} + cr) \in \mathbb{Z}_p^4$. The proof is

$$\pi = (c, s_{\mathrm{a}^{-1}}, s_{\mathrm{x}_1}, s_{\mathrm{x}_2}, s_{\mathrm{r}}) \in \mathbb{Z}_p^5\ .$$

The verifier, given $\pi = (c, s_{\mathrm{a}^{-1}}, s_{\mathrm{x}_1}, s_{\mathrm{x}_2}, s_{\mathrm{r}})$ and $X, C_1, C_2, \mathsf{G}_2, R, T$ from the context of Figure 1, computes the unique accepting Sigma protocol announcement $A = \phi(s_{\mathrm{a}^{-1}}, s_{\mathrm{x}_1}, s_{\mathrm{x}_2}, s_{\mathrm{r}}) - c \cdot (X, T, 0) \in \mathbb{G}^3$ and checks that

$$c \stackrel{!}{=} \mathcal{H}(\mathbf{S}, X, C_1, C_2, R, T, A) \; .$$

Implementation of the Sigma protocol in Figure 2. We implement

$$\pi \leftarrow \mathsf{PoK} \left\{ (x, v) \mid \tau = xC \wedge X = xR \wedge V = v\mathsf{G} \wedge \bigwedge_{j=0}^{|\mathbf{S}|-1} vY_j = Y_{j+1} \right\} \; ,$$

in Figure 2 as follows. For given C, R, Y_i from the context of Figure 2, let $\phi(x, v) \coloneqq (xC, xR, (vY_j)_{j=0}^{|\mathbf{S}|-1})$. The prover chooses random $r_{\mathrm{x}}, r_{\mathrm{v}} \xleftarrow{\$} \mathbb{Z}_p$ and computes Sigma protocol announcement $A = \phi(r_{\mathrm{x}}, r_{\mathrm{v}}) \in \mathbb{G}^{2+|\mathbf{S}|}$. It then computes Fiat-Shamir challenge $c = \mathcal{H}(\mathbf{S}, \tau, V, C, X, R, (Y_j)_{j=0}^{|\mathbf{S}|}, A) \in \mathbb{Z}_p$. It then computes the response $(s_{\mathrm{x}}, s_{\mathrm{v}}) = (r_{\mathrm{x}} + cx, r_{\mathrm{v}} + cv) \in \mathbb{Z}_p^2$. The proof is

$$\pi = (c, s_{\mathrm{x}}, s_{\mathrm{v}}) \in \mathbb{Z}_p^3 \; .$$

The verifier, given $\pi = (c, s_{\mathrm{x}}, s_{\mathrm{v}})$ and $\tau, V, C, X, R, (Y_j)_{j=0}^{|\mathbf{S}|}$ from the context of Figure 2, computes the unique accepting Sigma protocol announcement $A = \phi(s_{\mathrm{x}}, s_{\mathrm{v}}) - c \cdot (\tau, X, (Y_{j+1})_{j=0}^{|\mathbf{S}|-1}) \in \mathbb{G}^{2+|\mathbf{S}|}$ and checks that,

$$c \stackrel{!}{=} \mathcal{H}(\mathbf{S}, \tau, V, C, X, R, (Y_j)_{j=0}^{|\mathbf{S}|}, A) \; .$$

Implementation of the Sigma protocol in the DVSC construction. We implement

$$\mathsf{PoK}\{v \mid V_0 = \mathsf{G} \wedge (vV_j = V_{j+1})_{j=0}^{t-1}\} \; ,$$

in Section 4.2 as follows. For given V_j from the context of Section 4.2, let $\phi(v) \coloneqq (vV_j)_{j=0}^{t-1}$. The prover chooses random $r_{\mathrm{v}} \xleftarrow{\$} \mathbb{Z}_p$ and computes Sigma protocol announcement $A = \phi(r_{\mathrm{v}}) \in \mathbb{G}^t$. It then computes Fiat-Shamir challenge $c = \mathcal{H}((V_j)_{j=0}^{t-1}, A) \in \mathbb{Z}_p$. It then computes the response $s_{\mathrm{v}} = r_{\mathrm{v}} + cv \in \mathbb{Z}_p$. The proof is

$$\pi = (c, s_{\mathrm{v}}) \in \mathbb{Z}_p^2 \; .$$

The verifier, given $\pi = (c, s_{\mathrm{v}})$ and $(V_j)_{j=0}^{t}$ from ipar in the context of Section 4.2, computes the unique accepting Sigma protocol announcement $A = \phi(s_{\mathrm{v}}) - c \cdot (vV_j)_{j=0}^{t-1} \in \mathbb{G}^t$ and checks that,

$$c \stackrel{!}{=} \mathcal{H}((V_j)_{j=0}^{t}, A) \; ,$$

and that $V_0 = \mathsf{G}_1$. Note that in practice, the proof only has to be checked once, not for every single invocation of Commit.

B Proofs in the generic group model.

The Generic Group Model (GGM) [Sho97, Mau05] as a well-established proving tool applied to cyclic groups makes it possible to prove many remarkable results that are difficult to achieve in the standard model. Generic algorithms within this model are restricted to outputting only group elements by interacting with an oracle applying the group operations on these elements. The GGM is particularly useful for establishing information-theoretic lower bounds for computational problems. In this section, we provide

an explanation of the generic group model and security proofs in this model. This should help understand the security proofs for our constructions. Readers already familiar with the GGM can safely skip this section.

In the generic group model, the ("generic") adversary interacts with the group in an encoding-agnostic way. Specifically, whenever the adversary would receive a group element G as input, it instead receives an encoding $\sigma(G)$, where $\sigma : \mathbb{G} \to \{0,1\}^{2n}$ is a random injective encoding function. Since group elements are represented as random bit strings, the adversary is effectively prohibited from executing meaningful group operations (it cannot even guess which bit strings correspond to valid group element encodings). To enable the adversary to do group operations, we supply it with an additional oracle $\mathcal{O}$, which takes two encoded elements $\sigma(G), \sigma(G')$ as input and returns the encoded result of the group operation $\sigma(G + G')$. Note that this oracle is sufficient also to compute $\sigma(-G)$ from $\sigma(G)$, by executing a double-and-add algorithm on $\sigma(G)$ to compute $\sigma((p-1)G)$.

As a result, an execution in the ("original") generic group model works as follows:

- The execution chooses a random injective encoding function $\sigma : \mathbb{G} \to \{0,1\}^{2n}$.
- Let G denote the group generator. The adversary receives the random encoding $\sigma(\mathsf{G})$ as input.
- The experiment operates over the group $\mathbb{G}$ as usual. If the adversary outputs some encoded group element $\sigma(G)$ to the experiment, the experiment decodes it using σ^{-1} and operates on the corresponding group element $\sigma^{-1}(\sigma(G)) = G$.
- Whenever the experiment would hand a group element G to the adversary, it instead hands over the encoded group element $\sigma(G)$.
- The adversary gets access to a group operation oracle $\mathcal{O}$ that takes two encoded group elements $\sigma(G), \sigma(G')$ and returns the encoded result of the group operation $\sigma(G + G')$.

The standard structure for generic group model proofs (e.g., [BBG05a], the full version of [BBG05b], Theorem A.2) is to play along the generic group operations with (multivariate) polynomials instead of group elements. We call this alternative execution of the experiment the "polynomial-based execution". It works as follows.

- Let G denote the group generator. Let $v_1, \ldots, v_n$ denote variables for which the security experiment chooses random values $v_i \xleftarrow{\$} \mathbb{Z}_p$ and which the adversary only sees "in the exponent". The point of the polynomial-based execution is to treat these variables as symbols instead of concrete values.
- The execution conceptually chooses a random encoding function $\sigma : \mathbb{Z}_p[\mathsf{G}, v_1, \ldots, v_n] \to \{0,1\}^{2n}$, where $\mathbb{Z}_p[\mathsf{G}, v_1, \ldots, v_n]$ is the ring of polynomials over the finite field $\mathbb{Z}_p$. Note that the encoding function σ is now defined over *polynomials*, not over group elements. (Note that σ cannot actually be chosen randomly at the start because its domain is infinite. Instead, random values for the function would be chosen ad-hoc, and any encoding not yet defined would be treated as not corresponding to any polynomial. We omit details.)
- The adversary receives the random encoding $\sigma(\mathsf{G})$ as input (where $\mathsf{G} \in \mathbb{Z}_p[\mathsf{G}, v_1, \ldots, v_n]$ is now a symbol).
- The experiment operates over the ring of polynomials $\mathbb{Z}_p[\mathsf{G}, v_1, \ldots, v_n]$ now, in the natural way, i.e. it follows the original computation instructions (like "compute $(5 + v_1) \cdot \mathsf{G}$") verbatim but treats all the symbols $\mathsf{G}, v_1, \ldots, v_n$ as polynomial variables instead of concrete group elements.

- If the adversary outputs some encoded group element $\sigma(G)$ to the experiment, the experiment decodes it using σ^{-1} and operates on the corresponding *polynomial* $\sigma^{-1}(\sigma(G)) = G \in \mathbb{Z}_p[\mathsf{G}, v_1, \ldots, v_n]$.
- Whenever the experiment would hand a polynomial G to the adversary, it instead hands over the encoded polynomial $\sigma(G)$.
- The group operation oracle now also operates on polynomials, i.e. $\mathcal{O}(\sigma(G), \sigma(G'))$ returns $\sigma(G + G')$, where $G, G', G + G' \in \mathbb{Z}_p[\mathsf{G}, v_1, \ldots, v_n]$.
- Whenever the original experiment would check an equation over group elements, the polynomial-based execution instead checks the corresponding equation over the polynomials.

Intuitively, because the adversary only sees random encodings, it cannot tell whether those random encodings are backed by group elements or polynomials. More formally, one can argue that as long as the polynomials involved in the polynomial-based execution are low degree, then the polynomial-based execution is indistinguishable from the original execution. This is because we can retrieve the behavior of the original execution by plugging in random values for the polynomial variables $\mathsf{G}, v_1, \ldots, v_n$ into the polynomial-based execution. The polynomial-based execution (with random values in mind) only deviates from the original execution (based on the same random values) if there are two different polynomials $G \neq G'$ during the execution such that $G(\mathsf{G}, v_1, \ldots, v_n) = G'(\mathsf{G}, v_1, \ldots, v_n)$ for the concrete random values of $\mathsf{G}, v_1, \ldots, v_n$, i.e. the polynomials were handled as two different group elements in the polynomial-based execution, but would have been handled as the same group element in the original group-based execution.

Thankfully, the probability for this event can be upper-bounded using the Schwartz-Zippel lemma. So if the polynomials involved are of low degree (polynomial in the security parameter; this is the case for all our proofs), then the probability of a collision is negligible. Hence the adversary's winning probability in the polynomial execution is essentially the same as its winning probability in the original execution.

The above argumentation establishes that the polynomial execution is indistinguishable from the original execution. The rest of the proof then deals with establishing that the adversary *cannot win* in the polynomial execution. Usually, this involves showing that the adversary cannot linearly combine (via $\mathcal{O}$) the polynomials given to it into a polynomial that would break some property.

For example, consider the computational Diffie-Hellman assumption. The experiment for this assumption hands $(\mathsf{G}, a\mathsf{G}, b\mathsf{G})$ for random $a, b \leftarrow \mathbb{Z}_p$ to the adversary $\mathcal{A}$ and checks whether the adversary manages to output $ab\mathsf{G}$. The assumption states that the probability of $\mathcal{A}$ to be successful is negligible. In the generic group model, this translates to $\mathcal{A}^{\mathcal{O}}$ receiving input $(\sigma(\mathsf{G}), \sigma(a\mathsf{G}), \sigma(b\mathsf{G}))$ (where $\mathsf{G} \in \mathbb{G}$ and $a, b \xleftarrow{\$} \mathbb{Z}_p$) and having to output $\sigma(ab\mathsf{G})$.

To prove that the assumption holds, we consider the polynomial execution of the experiment. Here, G, a, b are be polynomial variables and we treat them symbolically, i.e. the adversary actually receives $(\sigma(\mathsf{G}), \sigma(a\mathsf{G}), \sigma(b\mathsf{G}))$ as input, where $\mathsf{G}, a\mathsf{G}, b\mathsf{G} \in \mathbb{Z}_p[\mathsf{G}, a, b]$ are polynomials in the variables G, a, b. The adversary gets to do group operations (on the polynomials) via the oracle $\mathcal{O}$. Eventually, the adversary outputs some encoding $\sigma(H)$ as a potential solution. Now because the adversary can effectively only create valid encodings through the GGM oracle $\mathcal{O}$, we know that $H \in \mathbb{Z}_p[\mathsf{G}, a, b]$ must be a linear combination of the terms that $\mathcal{A}$ got as input, i.e. we can write

$$H = (\alpha a + \beta b + \gamma)\mathsf{G} \qquad \text{(equation over } \mathbb{Z}_p[\mathsf{G}, a, b])$$

for some integers $\alpha, \beta, \gamma \in \mathbb{Z}_p$. Hence the winning condition check $H \stackrel{!}{=} ab\mathsf{G}$ *cannot* be fulfilled in the polynomial execution: for all integers α, β, γ, the polynomial $(\alpha a + \beta b + \gamma)\mathsf{G}$

is not equal to the polynomial $ab\mathsf{G}$. As a result (note that all polynomials involved in this execution are of degree at most 3), the adversary has negligible chance of winning the original generic group model execution.

In our proofs later, we will abuse notation and omit σ, i.e. we may write "we give H to the adversary", which of course means that we give $\mathcal{A}$ the encoding $\sigma(H)$ of H.

We furthermore highlight that the sketch above easily generalizes to the pairing-based setting, where essentially, we have three different encoding functions $\sigma_1, \sigma_2, \sigma_T$ and group operation oracles $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_T$, for groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, respectively. There is also a pairing oracle $\mathcal{O}_\times(\sigma_1(G), \sigma_2(G')) = \sigma_T(e(G, G'))$. For our proofs below, the pairing will play a minor role (it will not help the adversary come up with forgeries since the elements it needs to output live in the source groups $\mathbb{G}_1, \mathbb{G}_2$), so we do not mention it.

Finally, one of our proofs will involve terms of the form a^{-1}. We note that the arguments about the polynomial execution being indistinguishable from the original execution the also extends to Laurent polynomials (i.e. polynomials $f \in \mathbb{Z}_p[a, a^{-1}]$, where some of the exponents may be negative).

C A discussion on oblivious sampling in the GGM

Oblivious sampling is an important feature to rule out spurious assumptions in models like the generic group model or the algebraic group model [LPS23]. To implement oblivious sampling in the generic group model, one would augment the model with an additional oracle $\mathcal{O}_{\text{sample}}()$ that takes no input, chooses a uniformly random group element $H \xleftarrow{\$} \mathbb{G}$, and returns its encoding $\sigma(H)$ to the adversary. Sometimes this model is called *GGM with hashing* [BFHK24, BFS16]. See [LPS23] for oblivious sampling supporting any high-entropy distributions. This oracle models that in practice, there are ways for the adversary to sample group elements without knowing their discrete logarithm representation. For example, in an elliptic curve group, one may simply sample a random x-coordinate and then compute a corresponding y-coordinate to obtain a point on the curve. Done this way, the adversary does not know the discrete logarithm of the sampled point with respect to any other point.

The addition of this oracle is crucial to rule out spurious knowledge assumptions of the form "there is an extractor (with access to the list of GGM queries the adversary makes) such that whenever the adversary outputs a group element H, the extractor can output its discrete logarithm to base G" (called *SpurKE* in [LPS23]). This assumption would be clearly false in practice: the adversary can obliviously sample H, at which point extracting the discrete logarithm is as hard as solving the discrete logarithm problem. However, in the plain GGM the assumption would be true: the only way for the adversary to even produce encodings that correspond to *any* group elements is by operating on the (encoded) input elements (G in this case). Hence the extractor can simply follow the list of GGM queries to compute the discrete logarithm of any (intermediate) group element. The issue can be solved by adding the oblivious sampling oracle $\mathcal{O}_{\text{sample}}$ to the GGM, where the extractor may encounter an $\mathcal{O}_{\text{sample}}$ query, for which the extractor would not be able to compute the discrete logarithm, either.

In our GGM proofs, however, the scenario is different: there is no extractor; we do not prove any knowledge assumptions. More formally, the main differentiator to, say, proof of knowledge NIZKs, is that in our security proofs, the security game deciding the winning condition *does not read the list of GGM queries* made by the adversary. This is different from proof of knowledge definitions, where the game would run the proof of knowledge extractor, which would then read the list of GGM queries to extract the witness, which determines the winning condition.

As a consequence, all our GGM security proofs automatically hold in the GGM with oblivious sampling through the following argument. Let $\mathcal{A}$ be an adversary against Game

in the GGM with oblivious sampling. Construct adversary $\mathcal{A}'$ against Game in the plain GGM as follows: $\mathcal{A}'$ behaves exactly like $\mathcal{A}$, except that whenever $\mathcal{A}$ makes an oblivious sampling query $\mathcal{O}_{\text{sample}}()$, then $\mathcal{A}'$ instead choose a random $r \xleftarrow{\$} \mathbb{Z}_p$ and computes $r\mathsf{G}$ through a series of queries to its GGM oracle. Note that the view of $\mathcal{A}$ has the exact same distribution when run by $\mathcal{A}'$ in the plain GGM and when run by itself in the GGM with oblivious sampling. Because Game does not read the list of GGM queries, $\mathcal{A}'$ wins Game in the plain GGM with the same probability as $\mathcal{A}$ wins Game in the GGM with oblivious sampling. (Note that this would be false for proof of knowledge–style games as explained above.)

For this reason, we do not explicitly consider oblivious sampling in our GGM proofs, but note that our constructions are secure in the GGM with oblivious sampling, too.

D Proof of SP-MAC-EQ security

D.1 Proof of Theorem 2 (UF-CMVA)

Proof. We build on the ideas and notations from the proof from [FHS19], as our scheme is a modification of theirs. Our proof takes a contradiction-based approach. We start by assuming the existence of an adversary in the GGM capable of successfully forging a tag τ^* on a message $\mathbf{M}^*$ that does not belong to $\mathcal{Q}^{\mathsf{MAC}}$. In this model, the adversary can only forge a tag or generate a message for a query by using a linear combination of all the inputs.

Below, we follow the usual GGM proof technique of analyzing the polynomial-based execution of the unforgeability experiment. We explain this technique in Appendix B. In this case, we consider the variables $\mathsf{G}_1, \mathsf{G}_2, x_i, a_j$ as polynomial variables (and the a_j are also used with negative exponents), i.e. we are working in the (Laurent) polynomial ring $\mathbb{Z}_p[\mathsf{G}_1, \mathsf{G}_2, (x_i)_{i=1}^{\ell}, (a_j, a_j^{-1})_{j=1}^{q}]$.

Initially, the adversary gets the generators $\mathsf{G}_1, \mathsf{G}_2$ as input. The adversary is granted access to two oracles (in addition the the GGM oracles), namely $\mathcal{O}_{\mathsf{MAC}}$ and $\mathcal{O}_{\mathsf{Verify}}$. In the polynomial-based execution, $\mathcal{O}_{\mathsf{Verify}}(\mathbf{M}, \tau = (R, T))$ simply outputs the indicator bit checking whether $f' \cdot R = \sum_{i=1}^{\ell} x_i M_i$ (where this equation is over the polynomial ring), where $T = f'\mathsf{G}_2$ for some polynomial $f' \in \mathbb{Z}_p[\mathsf{G}_2, (a_j^{-1})_{j=1}^{q}]$. This perfectly models the pairing check made in the construction.
The oracle $\mathcal{O}_{\mathsf{MAC}}(\mathbf{M}_j)$ for the jth query returns

$$(R_j, T_j) = \left(a_j \left(\sum_{i \in [1,\ell]} x_i M_{j,i} \right), a_j^{-1} \mathsf{G}_2 \right).$$

Let q be the number of queries made by the adversary. Through input to $\mathcal{A}$ and through output of $\mathcal{O}_{\mathsf{MAC}}$, the adversary gets access to the points G_1 and $\{R_i \mid \forall i \in [q]\}$ in $\mathbb{G}_1$, as well as G_2 and $\{T_i \mid \forall i \in [q]\}$ in $\mathbb{G}_2$ before the forgery phase. Therefore, the forged tag and the corresponding message should have the following forms:

$$\begin{aligned} M_i^* &= \gamma_{m_i^*} \mathsf{G}_1 + \sum_{j \in [q]} \beta_{m_i^*, r_j} R_j \ , \\ R^* &= \gamma_{r^*} \mathsf{G}_1 + \sum_{j \in [q]} \beta_{r^*, r_j} R_j \ , \\ T^* &= \gamma_{t^*} \mathsf{G}_2 + \sum_{j \in [q]} \beta_{t^*, a_j} T_j \ . \end{aligned}$$

Factoring out G_1 in the above, we can write $M_{j,i} = m_{j,i}\mathsf{G}_1$ and $R_j = r_j \mathsf{G}_1$ for suitable $m_{j,i}, r_j \in \mathbb{Z}_p[(x_{i'})_{i'=1}^{\ell}, (a_{j'})_{j'=1}^{j-1}]$. Furthermore, we write $T_j = t_j \mathsf{G}_2$, where $t_j =$

$\frac{1}{a_j} \in \mathbb{Z}_p[a_j^{-1}]$. Similarly, let $M_i^* = m_i^* \mathsf{G}_1$, $R^* = r^* \mathsf{G}_1$ and $T^* = t^* \mathsf{G}_2$ for suitable $m_i^*, r^* \in \mathbb{Z}_p[(x_i)_{i=1}^{\ell}, (a_{j'})_{j'=1}^{q}]$ and $t^* \in \mathbb{Z}_p[(a_j^{-1})_{j=1}^{q}]$. From the equations above, we can derive the following forms (which are essentially factoring out $\mathsf{G}_1, \mathsf{G}_2$ for M_i^*, R^*, T^*).

$$m_i^* = \gamma_{m_i^*} + \sum_{j \in [q]} \beta_{m_i^*, r_j} r_j \ , \tag{3}$$

$$r^* = \gamma_{r^*} + \sum_{j \in [q]} \beta_{r^*, r_j} r_j \ , \tag{4}$$

$$t^* = \gamma_{t^*} + \sum_{j \in [q]} \beta_{t^*, a_j} \frac{1}{a_j} \ . \tag{5}$$

According to Equation (2), the adversary succeeds the forgery iff:

$$\sum_{i \in [\ell]} x_i m_i^* = r^* t^* \ . \tag{6}$$

Using Equations (3) to (5), we have:

$$\sum_{i \in [\ell]} x_i \gamma_{m_i^*} + \sum_{i \in [\ell]} \sum_{j \in [q]} x_i \beta_{m_i^*, r_j} r_j = \gamma_{r^*} \gamma_{t^*} + \gamma_{r^*} \sum_{j \in [q]} \beta_{t^*, a_j} \frac{1}{a_j} + \gamma_{t^*} \sum_{j \in [q]} \beta_{r^*, r_j} r_j + \sum_{j \in [q]} \sum_{k \in [q]} \beta_{r^*, r_j} \beta_{t^*, a_k} r_j \frac{1}{a_k} \ . \tag{7}$$

Based on Claim 1 in [FHS19], all the monomials in r_n look like this:

$$\frac{1}{a_v^b} \prod_{k \in [u]} a_{j_k} \prod_{k \in [u]} x_{i_k} \ ,$$

in which $b \in \{0, 1\}$, $u \in [n]$, $\{j_{k_1} \neq j_{k_2} | k_1 \neq k_2\}$ and for all k we have: $j_k \leq n$, $v < j_k$ and $j_u = n$. Moreover, according to the Corollary 1 in [FHS19], each monomial only occurs for one r_n.

Note that although Claim 1 and Corollary 1 are stated for the SPS-EQ scheme, they also hold true in our scheme. This is because the only difference between our scheme and SPS-EQ is the absence of certain terms present in the latter[10].

Although both terms in the RHS of Equation (7) contain x, there is no x in the first two terms of the RHS of Equation (7). So:

$$\gamma_{r^*} \gamma_{t^*} = 0 \ ,$$

$$\gamma_{r^*} \beta_{t^*, a_j} = 0 \ .$$

The third term on the RHS of Equation (7) has the same number of x's and a's. However, both terms on the LHS have one more x. Therefore:

$$\gamma_{t^*} \beta_{r^*, r_j} = 0 \ .$$

And we have:

$$\sum_{i \in [\ell]} x_i \gamma_{m_i^*} + \sum_{i \in [\ell]} \sum_{j \in [q]} x_i \beta_{m_i^*, r_j} r_j = \sum_{j \in [q]} \sum_{k \in [q]} \beta_{r^*, r_j} \beta_{t^*, a_k} r_j \frac{1}{a_k} \ . \tag{8}$$

[10] The only consequence is that the value b is always 0 in our scheme.

Now, we want to show that for every $j \neq k$, the product $\beta_{r^*,r_j}\beta_{t^*,a_k}$ is zero. Firstly, let's consider the case when $k > j$. In this scenario, in the RHS of Equation (8), we have monomials with a's in the denominator with greater indices than a's in the numerator. There is no such term on the LHS. Therefore, for $k > j$, we have $\beta_{r^*,r_j}\beta_{t^*,a_k} = 0$.

Secondly, if $j > k$, we assume that $\beta_{r^*,r_j}\beta_{t^*,a_k}$ is not zero for at least one pair of j and k. Then, as all the monomials are a multiple of $\frac{a_j}{a_k}$, only the second term on the LHS of Equation (8) can have these monomials. However, if r_j has a monomial with a numerator, a_k, on the RHS of Equation (8) a_k will be canceled with the one in the denominator, although we have a_k on the RHS, which is a contradiction. If there is no a_k in any of the monomials for r_j or only on their denominator, then, by multiplying both sides of Equation (8) by a_k or a_K^2, respectively, we have monomials with a_k on the LHS but not any on the RHS. Therefore, for $j > k$ we have $\beta_{r^*,r_j}\beta_{t^*,a_k} = 0$.

Based on what we obtained till now, only $\beta_{r^*,r_j}\beta_{t^*,a_k}$ for $j = k$ can be non-zero. Now, we want to show that, for just one of k's the multiplication of these coefficients is non-zero. Suppose for two different values like j_1 and j_2 we have $\beta_{r^*,r_{j_1}}\beta_{t^*,a_{j_1}\neq 0}$ and $\beta_{r^*,r_{j_2}}\beta_{t^*,a_{j_2}} \neq 0$. Therefore, $\beta_{r^*,r_{j_1}}\beta_{t^*,a_{j_2}\neq 0}$ and $\beta_{r^*,r_{j_2}}\beta_{t^*,a_{j_1}\neq 0}$ as well, which is a contradiction. So for a value $n \in [q]$ we have:

$$\sum_{i\in[\ell]} x_i m_i^* = \beta_{r^*,r_n}\beta_{t^*,a_n} r_n \frac{1}{a_n} \ .$$

r_n and a_n are part of the n-th queried signature. Therefore, they will definitely fulfil the equation 6 for the n-th query. If we show the i-th element of the n-th queried message by $m_{i,n}$, we have:

$$\sum_{i\in[\ell]} x_i m_i^* = \beta_{r^*,r_n}\beta_{t^*,a_n} \sum_{i\in[\ell]} x_i m_{i,n}$$

$$\implies \forall i \in [\ell] : m_i^* = \beta_{r^*,r_n}\beta_{t^*,a_n} m_{i,n} \ .$$

Therefore, the forged signature generated by the adversary is in the equivalence class of a previously queried message. This is not accepted by the challenger, so the proposed scheme is UF-CMVA secure. □

E Proof of DVSC security

E.1 Proof of Theorem 5 (Correctness)

Proof. By running DVSC.Commit with the set of attributes $\mathbf{S}$ and DVSC.Randomize with a random value μ, we obtain $C' = (\mu f_{\mathbf{S}}(v)\mathsf{G}, \mu\mathsf{G}')$. The output of the DVSC.OpenSubset algorithm for $\mathbf{S}$ and a subset $\mathbf{D} \subseteq \mathbf{S}$ is $W = \mu f_{\mathbf{S}\setminus\mathbf{D}}(v)\mathsf{G}$. Since $\mathbf{D}$ is a subset of $\mathbf{S}$, we have $f_{\mathbf{S}}(v) = f_{\mathbf{S}\setminus\mathbf{D}}(v) \cdot f_{\mathbf{D}}(v)$. Therefore, $f_{\mathbf{D}}(v)W = C'_1$, and running DVSC.VerifySubset returns 1 with probability 1. □

E.2 Binding Property of DVSC and Proof

As a warm-up, we consider the following natural definition of the binding property. For our constructions, we will need the stronger subset soundness property, which we prove in Appendix E.3.

Definition 26 (Binding)**.** A DVSC meets the binding property if for all PPT adversaries

$\mathcal{A}$, we have:

$$\Pr\left[\begin{array}{ll} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); & \\ (\mathsf{sk}, \mathsf{ipar}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, 1^t); & \\ (\mathbf{S}_0, \mathbf{S}_1, \mu_0, \mu_1) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{ipar}); & (C_0 = C_1 \vee \\ C_0 \leftarrow \mathsf{Commit}(\mathsf{pp}, \mathsf{ipar}, \mathbf{S}_0); & : \ C'_0 = C'_1) \wedge \\ C_1 \leftarrow \mathsf{Commit}(\mathsf{pp}, \mathsf{ipar}, \mathbf{S}_1) & \mathbf{S}_0 \neq \mathbf{S}_1 \\ C'_0 \leftarrow \mathsf{Randomize}(\mathsf{pp}, C_0; \mu_0) & \\ C'_1 \leftarrow \mathsf{Randomize}(\mathsf{pp}, C_1; \mu_1) & \end{array}\right] \leq \mathsf{negl}(\lambda) \ .$$

The DVSC scheme satisfies the Binding property due to the following lemma:

Lemma 1. *If the t-co-DL assumption defined in Definition 15 holds and the underlying ZK proof has zero-knowledge property, then our DVSC scheme satisfies the Binding property.*

Proof.

We argue that if the NIZK in ipar is replaced with a simulated NIZK, as defined in the binding property in Definition 26, and we call this modified game Hyb, the advantage of any PPT adversary in the binding game and Hyb differs by at most $\mathsf{negl}(\lambda)$, given the zero-knowledge property of the NIZK. Thus, it is sufficient to show that the probability of winning in Hyb is at most $\mathsf{negl}(\lambda)$.

Furthermore, if the adversary finds a collision $(\mathbf{S}_0, \mathbf{S}_1, \mu_0, \mu_1)$ such that $C'_0 = C'_1$, then necessarily, the de-randomized commitments must collide, i.e. $C_0 = C_1$. For this reason, in the following, we restrict our analysis to colliding C_0, C_1.

Assume there exists a PPT adversary $\mathcal{A}$ that can win Hyb with non-negligible probability. Then, we can construct a PPT adversary $\mathcal{B}$ that uses $\mathcal{A}$ internally to break the t-co-DL assumption (cf. Definition 15) with the same probability. The t-co-DL challenger runs $\mathcal{BG}(1^\lambda)$, samples $v \xleftarrow{\$} \mathbb{Z}_p^*, G' \xleftarrow{\$} \mathbb{G}_1$, and then sends $\big(\mathcal{BG}(1^\lambda), \{v^i\mathsf{G}_1, v^i\mathsf{G}_2\}_{i\in[t]}\big)$ to $\mathcal{B}$. Next, $\mathcal{B}$ forwards the public parameters of the first group, $(\mathbb{G}_1, \mathsf{G}_1, q)$, as pp and provides $\{v^i\mathsf{G}_1\}_{i\in[t]}$ as ipar to $\mathcal{A}$ along with a simulated NIZK proof π_{sim}.

Eventually, the adversary $\mathcal{A}$ returns $(\mathbf{S}_0, \mathbf{S}_1, \mu_0, \mu_1)$ such that with non-negligible probability, $\mathsf{DVSC.Commit}(\mathsf{pp}, \mathsf{ipar}, \mathbf{S}_0) = \mathsf{DVSC.Commit}(\mathsf{pp}, \mathsf{ipar}, \mathbf{S}_1)$ and $\mathbf{S}_0 \neq \mathbf{S}_1$. This implies that

$$(f_{\mathbf{S}_0}(v)\mathsf{G}_1, \mathsf{G}'_1) = (f_{\mathbf{S}_1}(v)\mathsf{G}_1, \mathsf{G}'_1) \ ,$$

which results in

$$f_{\mathbf{S}_0}(v)\mathsf{G}_1 - f_{\mathbf{S}_1}(v)\mathsf{G}_1 = 0_{\mathbb{G}_1} \ .$$

This means v is a root of the polynomial $f_{\mathbf{S}_0}(X) - f_{\mathbf{S}_1}(X)$. $\mathcal{B}$ computes the roots of $f_{\mathbf{S}_0}(X) - f_{\mathbf{S}_1}(X)$ and returns the appropriate root v that solves the t-co-DL problem. Overall, $\mathcal{B}$ succeeds with the same non-negligible probability that $\mathcal{A}$ has in breaking Hyb and, consequently, the binding property. This is a contradiction, completing the proof. □

E.3 Proof of Theorem 6 (Subset-Soundness)

Proof. We argue that if the NIZK in ipar is replaced with a simulated NIZK, as defined in the subset soundness property in Definition 11, and we call this modified game Hyb, the advantage of any PPT adversary in the subset soundness game and Hyb differs by at most $\mathsf{negl}(\lambda)$, given the zero-knowledge property of the NIZK. Thus, it is sufficient to show that the probability of winning in Hyb is at most $\mathsf{negl}(\lambda)$.

Let $\mathcal{A}^{\mathcal{O}}$ be any (generic) adversary against Hyb. As sketched in Appendix B, we analyze the game in its polynomial-based execution, i.e. we replace generic group operations with operations over the polynomial ring $\mathbb{Z}_p[\mathsf{G}, \mathsf{G}', v]$. At the start, the adversary $\mathcal{A}$

gets (encodings of) polynomials $(\mathsf{G}', (V_j = v^j\mathsf{G})_{j=0}^t)$ as input. In addition to the GGM oracles, $\mathcal{A}$ has access to the $\mathsf{VerifySubset}$ oracle (cf. Definition 11). When $\mathcal{A}$ queries $\mathcal{O}((C_1', C_2'), W, \mathbf{D})$, for polynomials C_1', C_2', W as, we return 1 iff $C_1' = f_{\mathbf{D}}(v) \cdot W$ over $\mathbb{Z}_p[\mathsf{G}_1, v]$. Due to the Schwartz-Zippel lemma, the difference between the original (GGM) game and the polynomial-based execution is negligible. In the following, we show that the adversary cannot win in the polynomial-based execution, which then concludes the proof.

Assume the adversary outputs $(\mathbf{S}, C', \mathbf{D}, W)$, where (1) C' is the randomized version of the commitment on $\mathbf{S}$, and (2) W is a valid subset opening for $\mathbf{D}$. We show that meeting both the first and second conditions implies $\mathbf{D} \subseteq \mathbf{S}$, which means the adversary cannot succeed.

The first winning condition of Definition 11 enforces C' to be $(C_1', C_2') = (\mu f_{\mathbf{S}}(v)\mathsf{G}, \mu\mathsf{G}')$ for some $\mu \in \mathbb{Z}_p^*$. The second winning condition ("$b = 1$") enforces $C_1' = f_{\mathbf{D}}(v)W$. Together, we get that if the adversary wins, then

$$f_{\mathbf{D}}(v)W = \mu f_{\mathbf{S}}(v)\mathsf{G} \,.$$

The adversary computes W based on the group elements it possesses, namely $(\mathsf{G}', (V_j = v^j\mathsf{G})_{j=0}^t)$ (note that $V_0 = \mathsf{G}$. Also note that we ignore elements from $\mathbb{G}_2, \mathbb{G}_\mathrm{T}$ as they do not play a role in this construction/proof). We write

$$W = \alpha\mathsf{G}' + \sum_{j=0}^{t} \beta_j v^j \mathsf{G}$$

for some coefficients $\alpha, \beta_j \in \mathbb{Z}_p$. Because the discrete logarithm of G' (w.r.t. G) and the random value $v \in \mathbb{Z}_p^*$ are unknown to the adversary, we hence treat them symbolically, following standard GGM proof structures. Plugging the above form of W into the equation from the winning conditions, we get:

$$\left(\alpha\mathsf{G}' + \sum_{j=0}^{t} \beta_j v^j \mathsf{G}\right) f_{\mathbf{D}}(v) = \mu f_{\mathbf{S}}(v)\mathsf{G} \,.$$

Since neither G' nor its discrete logarithm exist on the RHS of the equation, α must be zero. Therefore, we have:

$$\left(\frac{1}{\mu}\sum_{j=0}^{t} \beta_j v^j\right) f_{\mathbf{D}}(v) = f_{\mathbf{S}}(v) \,.$$

As $f_{\mathbf{S}}(v) \neq 0$ (given that we treat v as a polynomial variable and $f_{\mathbf{S}}$ is not the zero polynomial), this implies that $f_{\mathbf{D}}$ divides $f_{\mathbf{S}}$ (as polynomials over the variable v) and hence $\mathbf{D} \subseteq \mathbf{S}$, which means that the adversary's success probability in winning Hyb is negligible. □

E.4 Proof of Theorem 8 (Subset Open Simulatability)

Proof. We define $\mathsf{Sim}_0, \mathsf{Sim}_1$ as follows. $\mathsf{Sim}_0(\mathsf{view}_{\mathcal{A}})$ uses the proof of knowledge property of the NIZK in ipar, specifically the proof that the adversary provides $(\mathsf{PoK}\{v \mid V_0 = \mathsf{G} \wedge \bigwedge_{j=0}^{t-1} vV_j = V_{j+1}\})$, to extract the witness v. It outputs the trapdoor $\mathsf{td} = v$. $\mathsf{Sim}_1(\mathsf{td}, C' = (C_0', C_1'), \mathbf{D})$ outputs simulated opening witness $W = \frac{1}{f_{\mathbf{D}}(v)}C_0'$.

We now analyze the distinguishing advantage of an adversary $\mathcal{A}$ assuming that extraction of the NIZK succeeds (i.e. $V_0 = \mathsf{G} \wedge \bigwedge_{j=0}^{t-1} vV_j = V_{j+1}$ for the extracted value $\mathsf{td} = v$). The oracle $\mathcal{O}_{\mathsf{OpenSubset}_b}(\mu, \mathbf{S}, \mathbf{D})$ first checks whether $\emptyset \neq \mathbf{D} \subseteq \mathbf{S}$ and then proceeds as

follows. If $b = 0$, the oracle returns $W_0 = \mathsf{DVSC.OpenSubset}(\mathsf{pp}, \mathsf{ipar}, \mu, \mathbf{S}, \mathbf{D})$, which, relying on the soundness of the NIZK, is equal to $\mu f_{\mathbf{S}\setminus\mathbf{D}}(v)\mathsf{G}$. If $b = 1$, the oracle returns $W_1 = \mathsf{Sim}_1(\mathsf{td}, \mathsf{Randomize}(\mathsf{pp}, \mathsf{ipar}, \mathsf{Commit}(\mathsf{pp}, \mathsf{ipar}, \mathbf{S}), \mu), \mathbf{D}) = \mathsf{Sim}_1(\mathsf{td}, (\mu f_{\mathbf{S}}(v)\mathsf{G}, \mu\mathsf{G}'), \mathbf{D})$. This results in:

$$W_1 = \mu \frac{f_{\mathbf{S}}(v)}{f_{\mathbf{D}}(v)}\mathsf{G} \,.$$

Since $\mathbf{D} \subseteq \mathbf{S}$, we have:

$$W_1 = \mu f_{\mathbf{S}\setminus\mathbf{D}}(v)\mathsf{G} = W_0 \,.$$

Thus, as long as the NIZK extraction succeeds, the two oracles $\mathcal{O}_{\mathsf{OpenSubset}_b}$ ($b \in \{0,1\}$) behave exactly the same. If the NIZK proof output by $\mathcal{A}$ is valid (otherwise, Commit outputs an error, and indistinguishability is trivial), NIZK extraction fails with only negligible probability. Therefore, the distinguishing advantage of $\mathcal{A}$ is at most $\frac{1}{2} + \mathsf{negl}(\lambda)$, as defined in Definition 25. □

F Proof of $\mathsf{KVAC}_{\mathsf{MEQ}}$ security

F.1 Proof of Theorem 10 (Unforgeability)

Proof.

Let $\mathcal{A}^{\mathcal{O}_{\mathsf{Cred}}, \mathcal{O}_{\mathsf{Verify}}}$ be a PPT adversary against the KVAC's unforgeability. Instead of analyzing the probability for $\mathcal{A}$ to win the unforgeability experiment, we will analyze the probability that $\mathcal{A}$ wins a modified unforgeability experiment where the issuer's zero-knowledge proofs are simulated. For this, let R_{pp} be the relation used for the issuer's NIZK in $\mathsf{KVAC}_{\mathsf{MEQ}}$.IssueCred (cf. Figure 1). Let $\mathcal{O}_{\mathsf{RO}}$ be the NIZK's random oracle ($\mathcal{O}_{\mathsf{RO}} : \{0,1\} \to \mathbb{Z}_p$ in our Schnorr-Fiat-Shamir case). Let Sim be the NIZK simulator (cf. Definition 23).

We denote the event that $\mathcal{A}$ wins the original unforgeability experiment as win. By win′, we denote the event that $\mathcal{A}$ wins the unforgeability experiment modified such that $\mathcal{O}_{\mathsf{Cred}}$ queries compute the proof π using Sim.Prove instead of Prove (and the experiment uses the simulated random oracle Sim.RO instead of the original $\mathcal{O}_{\mathsf{RO}}$).

Claim: $|\Pr[\mathsf{win}] - \Pr[\mathsf{win}']|$ **is negligible.** Let $\mathcal{A}_{\mathsf{zk}}^{\mathcal{O}_{\mathsf{RO}}, \mathcal{O}_{\mathsf{Prove}}}(\mathsf{pp})$ be the adversary that executes $\mathcal{A}$ against the unforgeability experiment, but using pp from its input, its first oracle ($\mathcal{O}_{\mathsf{RO}}$ or Sim.RO) for random oracle queries and its second oracle ($\mathcal{O}_{\mathsf{Prove}}$ or $\mathcal{O}_{\mathsf{Sim}}$) to compute proofs π during $\mathcal{O}_{\mathsf{Cred}}$ queries. $\mathcal{A}_{\mathsf{zk}}$ outputs a bit indicating whether $\mathcal{A}$ wins the unforgeability experiment (note that this can be efficiently decided by $\mathcal{A}_{\mathsf{zk}}$). We get $\Pr[\mathcal{A}_{\mathsf{zk}}^{\mathcal{O}_{\mathsf{RO}}, \mathcal{O}_{\mathsf{Prove}}}(\mathsf{pp}) = 1] = \Pr[\mathsf{win}]$ and $\Pr[\mathcal{A}_{\mathsf{zk}}^{\mathsf{Sim}_{\mathsf{RO}}, \mathcal{O}_{\mathsf{Sim}}}(\mathsf{pp}) = 1] = \Pr[\mathsf{win}']$ and due to the zero knowledge property, we get that $|\Pr[\mathsf{win}] - \Pr[\mathsf{win}']|$ is negligible. In the following, we show that $\Pr[\mathsf{win}']$ is negligible, which then implies that $\Pr[\mathsf{win}]$ must be negligible, too. To simplify notation, we omit the random oracle Sim.RO from the adversary $\mathcal{A}$ below.

Splitting up win′ **into two events.** Let $\mathsf{Show}^* = (\tau^*, C^* = (C_1^*, C_2^*), W^*)$ and $\mathbf{D}^*$ denote the candidate forgery output of the adversary $\mathcal{A}$ (as in Definition 2), and let $\mathcal{Q}_{\mathsf{Cred}}$ be the set of attribute vectors $\mathbf{S}$ queried to the oracle $\mathcal{O}_{\mathsf{Cred}}(\cdot)$. Let $\mathcal{Q}_{\mathsf{MAC}} := \{\mu \cdot \mathsf{Commit}(\mathsf{pp}, \mathsf{ipar}, \mathbf{S}) \mid \mathbf{S} \in \mathcal{Q}_{\mathsf{Cred}}, \mu \in \mathbb{Z}_p^*\}$, i.e. the union of all equivalence classes signed by the issuer due to $\mathcal{O}_{\mathsf{Cred}}$ queries. As defined above, win′ is the event that $\mathcal{A}$ wins the game (with simulated proofs), i.e. $\mathsf{Verify}(\mathsf{pp}, \mathsf{Show}^*, \mathbf{D}^*, \mathsf{isk}) = 1$ and $\nexists \S \in \mathcal{Q}_{\mathsf{Cred}}\ \mathbf{D}^* \subseteq \S$. We divide the winning condition into two events:

Event MacForge**:** We have win′ and $C^* \notin \mathcal{Q}_{\mathsf{MAC}}$ (intuitively: the adversary has forged a MAC tag).

Event DvscForge**:** We have win' and $C^* \in \mathcal{Q}_{\mathsf{MAC}}$ (intuitively: the adversary must have broken subset soundness of the DVSC).

Note that win' if and only if $\mathsf{MacForge} \vee \mathsf{DvscForge}$. In the following, we show that both MacForge and DvscForge happen with negligible probability.

Claim: Pr[MacForge] **is negligible.** We construct $\mathcal{A}_{\mathsf{MEQ}}^{\mathcal{O}_{\mathsf{MAC}},\mathcal{O}_{\mathsf{Verify}}}(\mathsf{pp}_{\mathsf{MEQ}})$ against the UF-CMVA property (Definition 6) for $\ell = 2$ of the SP-MAC-EQ as follows.

- $\mathcal{A}_{\mathsf{MEQ}}$ sets $\mathsf{pp}_{\mathsf{DVSC}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_1')$ and $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_1', \mathsf{G}_1'')$ using $\mathsf{pp}_{\mathsf{MEQ}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, \mathsf{G}_1, \mathsf{G}_2)$ from its input and $\mathsf{G}_1', \mathsf{G}_1'' \xleftarrow{\$} \mathbb{G}_1$. It simulates the commitment to the MAC secret key as $\mathsf{ipar}_{\mathsf{MEQ}} = X \xleftarrow{\$} \mathbb{G}_1$. It computes $v, \mathsf{ipar}_{\mathsf{DVSC}}$ as in $\mathsf{KVAC}_{\mathsf{MEQ}}.\mathsf{KeyGen}$ and sets $\mathsf{ipar} = (\mathsf{ipar}_{\mathsf{MEQ}}, \mathsf{ipar}_{\mathsf{DVSC}})$. Notably, $\mathcal{A}_{\mathsf{MEQ}}$ does not have access to the MAC key $\mathsf{sk}_{\mathsf{MEQ}} = (x_1, x_2)$ or to the commitment randomness r for X.
- $\mathcal{A}_{\mathsf{MEQ}}$ then starts running $\mathcal{A}^{\mathcal{O}_{\mathsf{Cred}},\mathcal{O}_{\mathsf{Verify}}}(\mathsf{pp}, \mathsf{ipar})$.
- Whenever $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Cred}}(\mathbf{S})$, our adversary $\mathcal{A}_{\mathsf{MEQ}}$ computes the commitment C to $\mathbf{S}$ as in $\mathsf{KVAC}_{\mathsf{MEQ}}.\mathsf{IssueCred}$. $\mathcal{A}_{\mathsf{MEQ}}$ then retrieves $\tau \leftarrow \mathcal{O}_{\mathsf{MAC}}(C)$ from its own oracle. It then simulates the proof $\pi \leftarrow \mathsf{Sim}_{\mathsf{Prove}}(\mathsf{stmt})$ using the appropriate statement stmt as in Figure 1 and returns $\mathsf{PreCred} = (\tau, \pi)$.
- Whenever $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Verify}}(\mathsf{Show}, \mathbf{D})$ with $\mathsf{Show} = (\tau', C', W)$, our adversary $\mathcal{A}_{\mathsf{MEQ}}$ queries its own oracle $b' = \mathcal{O}_{\mathsf{Verify}}(C', \tau')$, computes $b'' = \mathsf{DVSC.VerifySubset}(\mathsf{pp}_{\mathsf{DVSC}}, \mathsf{sk}_{\mathsf{DVSC}}, C', W, \mathbf{D})$, and returns $b' \wedge b''$ to $\mathcal{A}$.
- Eventually, $\mathcal{A}$ outputs a candidate presentation forgery $\mathsf{Show}^* = (\tau^*, C^*, W^*)$ and set $\mathbf{D}^*$. $\mathcal{A}_{\mathsf{MEQ}}$ outputs MAC τ^* and message $\mathbf{M}^* := C^*$ as its candidate MAC forgery.

Note that $\mathcal{A}_{\mathsf{MEQ}}$ perfectly simulates the input and the $\mathcal{O}_{\mathsf{Cred}}$ and $\mathcal{O}_{\mathsf{Verify}}$ oracles for $\mathcal{A}$ (for the modified game with simulated proofs). Clearly, whenever MacForge occurs, the output of $\mathcal{A}_{\mathsf{MEQ}}$ is a valid forgery for the SP-MAC-EQ scheme. Because the SP-MAC-EQ scheme is unforgeable, Pr[MacForge] must be negligible.

Claim: Pr[DvscForge] **is negligible.** We construct $\mathcal{A}_{\mathsf{DVSC}}^{\mathcal{O}}(\mathsf{pp}_{\mathsf{DVSC}}, \mathsf{ipar}_{\mathsf{DVSC}})$ against the subset soundness property (Definition 11) of the DVSC scheme as follows.

- $\mathcal{A}_{\mathsf{DVSC}}$ sets $\mathsf{pp}_{\mathsf{MEQ}} = (\mathbb{G}_1, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_1')$ and $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_1', \mathsf{G}_1'')$ using $\mathsf{pp}_{\mathsf{DVSC}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_1')$ from its input and $\mathsf{G}_1'' \xleftarrow{\$} \mathbb{G}_1$. It honestly computes the MAC secret key $\mathsf{sk}_{\mathsf{MEQ}} = (x_1, x_2) \leftarrow \mathsf{MEQ.KeyGen}_{\mathcal{R}}(\mathsf{pp}_{\mathsf{MEQ}}, 2)$ and the commitment $\mathsf{ipar}_{\mathsf{MEQ}} = x_1\mathsf{G}_1 + x_2\mathsf{G}_1' + r\mathsf{G}_1''$. It sets $\mathsf{ipar} = (\mathsf{ipar}_{\mathsf{MEQ}}, \mathsf{ipar}_{\mathsf{DVSC}})$. Notably, $\mathcal{A}_{\mathsf{DVSC}}$ does not have access to the DVSC trapdoor v.
- $\mathcal{A}_{\mathsf{DVSC}}$ then starts running $\mathcal{A}^{\mathcal{O}_{\mathsf{Cred}},\mathcal{O}_{\mathsf{Verify}}}(\mathsf{pp}, \mathsf{ipar})$.
- Whenever $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Cred}}(\mathbf{S})$, $\mathcal{A}_{\mathsf{DVSC}}$ runs the $\mathsf{KVAC}_{\mathsf{MEQ}}.\mathsf{IssueCred}(\mathsf{pp}, \mathbf{S}, \mathsf{isk}, \mathsf{ipar})$ algorithm honestly as in Figure 1 (except with simulated proof $\pi \leftarrow \mathsf{Sim}_{\mathsf{Prove}}(\mathsf{stmt})$ using the appropriate statement stmt). Note that $\mathcal{A}_{\mathsf{DVSC}}$ does not need v for this.
- Whenever $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Verify}}(\mathsf{Show}, \mathbf{D})$ with $\mathsf{Show} = (\tau', C', W)$, our adversary $\mathcal{A}_{\mathsf{DVSC}}$ queries its own oracle $b' = \mathcal{O}_{\mathsf{Verify}}(C', W, \mathbf{D})$, computes $b'' = \mathsf{MEQ.Verify}(\mathsf{pp}_{\mathsf{MEQ}}, \mathsf{sk}_{\mathsf{MEQ}}, C', W, \mathbf{D})$, and returns $b' \wedge b''$ to $\mathcal{A}$.
- Eventually, $\mathcal{A}$ outputs a candidate presentation forgery $\mathsf{Show}^* = (\tau^*, C^*, W^*)$ and set $\mathbf{D}^*$.

- $\mathcal{A}_{\mathsf{DVSC}}$ guesses a random $\mathbf{S} \xleftarrow{\$} \mathcal{Q}_{\mathsf{Cred}}$ from the set of queried attribute vectors. It outputs $(\mathbf{S}, C^*, \mathbf{D}^*, W^*)$ as a candidate subset opening.

Note that $\mathcal{A}_{\mathsf{DVSC}}$ perfectly simulates the input and the $\mathcal{O}_{\mathsf{Cred}}$ and $\mathcal{O}_{\mathsf{Verify}}$ oracles for $\mathcal{A}$ (for the modified game with simulated proofs). Whenever $\mathsf{DvscForge}$ occurs, by definition we have that $C^* \in \mathcal{Q}_{\mathsf{MAC}}$ and $\mathsf{DVSC.VerifySubset}(\mathsf{pp}_{\mathsf{DVSC}}, \mathsf{sk}_{\mathsf{DVSC}}, C^*, W^*, \mathbf{D}^*) = 1$. By definition of $C^* \in \mathcal{Q}_{\mathsf{MAC}}$, we know that $C^* = \mu \cdot \mathsf{Commit}(\mathsf{pp}_{\mathsf{DVSC}}, \mathsf{ipar}_{\mathsf{DVSC}}, \mathbf{S})$ for some $\mu \in \mathbb{Z}_p^*$ and $\mathbf{S} \in \mathcal{Q}_{\mathsf{Cred}}$. With probability at least $1/|\mathcal{Q}_{\mathsf{Cred}}|$, the adversary $\mathcal{A}_{\mathsf{DVSC}}$ outputs such an $\mathbf{S}$. Because win', we have that $\mathbf{D}^* \not\subseteq \mathbf{S}$. Overall, $\mathcal{A}_{\mathsf{DVSC}}$ wins its subset-soundness game with probability $\Pr[\mathsf{DvscForge}]/|\mathcal{Q}_{\mathsf{Cred}}|$. Because the DVSC scheme is subset-sound and $|\mathcal{Q}_{\mathsf{Cred}}|$ is of polynomial size, $\Pr[\mathsf{DvscForge}]$ must be negligible.

In summary,

$$\begin{aligned} & Adv^{\mathsf{Unforge}}_{\mathsf{KVAC}_{\mathsf{MEQ}},\mathcal{A}}(\lambda) \\ \leq\ & Adv^{\mathsf{ZK}}_{\mathcal{A}_{\mathsf{zk}},\mathsf{NIZK}} \\ & + Adv^{\mathsf{EU-CVMA}}_{\mathcal{A}_{\mathsf{MEQ}},\mathsf{MEQ}}(\lambda) \\ & + \frac{1}{p(\lambda)} \cdot Adv^{\mathsf{SubSound}}_{\mathcal{A}_2,\mathsf{DVSC}}(\lambda) \end{aligned}$$

for some polynomial p that upper-bound the number of queries to $\mathcal{O}_{\mathsf{Cred}}$ that $\mathcal{A}$ makes. □

F.2 Proof of Theorem 11 (Unlinkability)

Before proving the unlinkability of the proposed KVAC, we make a slight modification to the class-hiding definition described in Definition 7 to enable the multi-instance case. Intuitively, in this revised definition, the adversary has access to an oracle $\mathcal{O}_b(\cdot)$ that outputs random instances, but is still unable to break the class-hiding property of the given SP-MAC-EQ.

Definition 27 (Multi-Instance Class-Hiding)**.** A relation $\mathcal{R}$ is called multi-instance class-hiding if for all PPT adversaries, $\mathcal{A}$, and $\mathcal{M} \coloneqq (\mathbb{G}_1^*)^\ell$ s.t. $\ell > 1$ we have:

$$\Pr\left[\begin{matrix}\mathbf{M} \xleftarrow{\$} (\mathbb{G}_1^*)^\ell \\ b \xleftarrow{\$} \{0,1\}, b' \leftarrow \mathcal{A}^{\mathcal{O}_b()}(\mathbf{M})\end{matrix} : b' = b\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

where each invocation of $\mathcal{O}_0()$ chooses and returns $\mathbf{M}_0 \xleftarrow{\$} (\mathbb{G}_1^*)^\ell$, and each invocation of $\mathcal{O}_1()$ chooses and returns $\mathbf{M}_1 \xleftarrow{\$} [\mathbf{M}]_{\mathcal{R}}$.

Assuming Decisional Diffie-Hellman, one can show that the relation $\mathcal{R} = \{(\mathbf{M}, \mathbf{M}') \in (\mathbb{G}_1^*)^\ell \times (\mathbb{G}_1^*)^\ell \mid \exists \mu :\ \mu\mathbf{M} = \mathbf{M}'\}$ of the scheme in Section 3.2 is multi-instance class-hiding. We use Definition 27 to simplify our proofs, but essentially, it is the same assumption as Decisional Diffie-Hellman.

Our general strategy will be to gradually change the $\mathcal{O}_{\mathsf{Show}_b}$ oracle until its output is independent of b. The first two game hops (extracting the MAC key from $\mathsf{PreCred}_0, \mathsf{PreCred}_1$ and checking the key-parameter consistency) are in preparation for this strategy.

Proof. First, we form a sequence of hybrids and show two scenarios are computationally indistinguishable from each other described below:

Hyb_0**:** This game is defined as the unlinkability game described in Definition 3, where the adversary $\mathcal{A}$ on input pp returns the tuple $(\mathbf{S}_0, \mathsf{PreCred}_0, \mathbf{S}_1, \mathsf{PreCred}_1, \mathsf{ipar}, \mathsf{st})$ and then gets access to oracle $\mathcal{O}_{\mathsf{Show}_b}$ for $b \xleftarrow{\$} \{0,1\}$.

Hyb_1**:** When the first stage of $\mathcal{A}$ outputs $(\mathbf{S}_0, \mathsf{PreCred}_0, \mathbf{S}_1, \mathsf{PreCred}_1, \mathsf{ipar}, \mathsf{st})$, use the extractor $\mathsf{Ext}(\cdot)$ (from Definition 25) for the proofs $\pi_{\mathsf{PreCred}_\beta}$ for $\beta \in \{0,1\}$. If the extractor fails to output valid witnesses, the game aborts. This happens with probability $\sum_{\beta=0}^{1} Adv^{\mathsf{PoK}}_{\mathsf{NIZK},\mathcal{A}_1^{(\beta)}}(\lambda)$, where $\mathcal{A}_1^{(\beta)}$ is an adversary against the extractability game of the given NIZK, running (the first stage of) $\mathcal{A}$ internally and returning $(\mathsf{PreCred}_\beta, \pi_{\mathsf{PreCred}_\beta})$ for which the extractor fails (cf. Definition 25), i.e. we have:

$$Adv^{\mathsf{Hyb}_1}_{\mathcal{A}}(\lambda) \geq Adv^{\mathsf{Hyb}_0}_{\mathcal{A}}(\lambda) - Adv^{\mathsf{PoK}}_{NIZK,\mathcal{A}_1^{(0)}}(n) - Adv^{\mathsf{PoK}}_{NIZK,\mathcal{A}_1^{(1)}}(n) \cdot$$

Hyb_2**:** The game now aborts if from the extraction in Hyb_1, we get two *different* keys $(x_1^{(0)}, x_2^{(0)}) = \mathsf{sk}_{\mathsf{MEQ}}^{(0)} \neq \mathsf{sk}_{\mathsf{MEQ}}^{(1)} = (x_1^{(1)}, x_2^{(1)})$. Whenever the new abort event happens, the extractor outputs $(x_1^{(0)}, x_2^{(0)}, r^{(0)})$ and $(x_1^{(1)}, x_2^{(1)}, r^{(1)})$ such that both are valid openings to the commitment X (from ipar output by the adversary), i.e. $X = x_1^{(0)}\mathsf{G}_1 + x_2^{(0)}\mathsf{G}_1' + r^{(0)}\mathsf{G}_1'' = x_1^{(1)}\mathsf{G}_1 + x_2^{(1)}\mathsf{G}_1' + r^{(1)}\mathsf{G}_1''$. Hence whenever this abort event happens, we find a Pedersen commitment collision. With the standard reduction $\mathcal{A}_2$ for the Pedersen commitment binding property under the discrete logarithm assumption, one can show that

$$Adv^{\mathsf{Hyb}_2}_{\mathcal{A}}(\lambda) \geq Adv^{\mathsf{Hyb}_1}_{\mathcal{A}}(\lambda) - 2Adv^{\mathsf{dlog}}_{\mathbb{G}_1,\mathcal{A}_2}(n)/2 \cdot$$

(where the factor 2 comes from having to guess whether $x_1^{(0)} \neq x_1^{(1)}$ or $x_2^{(0)} \neq x_2^{(1)}$).

In the following, we simply write $\mathsf{sk}_{\mathsf{MEQ}} := \mathsf{sk}_{\mathsf{MEQ}}^{(0)} = \mathsf{sk}_{\mathsf{MEQ}}^{(1)}$.

Hyb_3**:** During $\mathcal{O}_{\mathsf{Show}_b}$ queries, instead of obtaining the randomized tag τ' by running the $\mathsf{MEQ.ChgRep}(\mathsf{pp}_{\mathsf{MEQ}}, \tau; \mu)$ algorithm in step ③ in Figure 1, run $\mathsf{MEQ.MAC}(\mathsf{pp}_{\mathsf{MEQ}}, \mathsf{sk}_{\mathsf{MEQ}}, C')$ using the extracted secret key $\mathsf{sk}_{\mathsf{MEQ}}$ from above to create the tag τ' on C'. By perfect adaptation (cf. Definition 8), the distribution of τ' is the same in both hybrids. For this, note that τ received in $\mathsf{PreCred}$ is indeed a valid tag on C under $\mathsf{sk}_{\mathsf{MEQ}}$ (as required to apply Definition 8) by guarantees of the proof π_β within $\mathsf{PreCred}_\beta$, otherwise Hyb_2 would have aborted. Thus we have:

$$Adv^{\mathsf{Hyb}_3}_{\mathcal{A}}(\lambda) = Adv^{\mathsf{Hyb}_2}_{\mathcal{A}}(\lambda) \cdot$$

Hyb_4**:** Let $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ be the simulator for subset opening simulatability of the DVSC scheme (cf. Definition 13) for adversary $\mathcal{A}_4$ defined below. Use $\mathsf{Sim}_0(\mathsf{view}_{\mathcal{A}_4})$ (for the appropriate $\mathsf{view}_{\mathcal{A}_4}$) to obtain DVSC trapdoor td after $\mathcal{A}$ outputs ipar (behind the scenes, Sim_0 extracts the NIZK within $\mathsf{ipar}_{\mathsf{DVSC}}$). Then during $\mathcal{O}_{\mathsf{Show}_b}$ queries, use $\mathsf{Sim}_1(\mathsf{td}, C', \mathbf{D})$ to compute W instead of computing W via $\mathsf{DVSC.OpenSubset}(\ldots, \mu, \mathbf{S}, \mathbf{D})$ in step ⑥ in Figure 1.

Let $\mathcal{A}_4(\mathsf{pp}_{\mathsf{DVSC}})$ be the adversary that completes pp as in the original protocol, then runs $\mathcal{A}(\mathsf{pp})$ to obtain $(\mathbf{S}_0, \mathsf{PreCred}_0, \mathbf{S}_1, \mathsf{PreCred}_1, \mathsf{ipar}, \mathsf{st})$. Then $\mathcal{A}_4$ outputs $\mathsf{ipar}_{\mathsf{DVSC}}$ and remembers everything else in st. When invoked again, now with oracle accesss to $\mathcal{O}_{\mathsf{OpenSubset}_{b'}}$, $\mathcal{A}_4$ runs $\mathcal{A}^{\mathcal{O}_{\mathsf{Show}_b}}(\mathsf{pp}, \mathbf{s}^\top)$. Whenever $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Show}_b}$, $\mathcal{A}_4$ behaves like Hyb_3, but queries its own oracle $\mathcal{O}_{\mathsf{OpenSubset}_{b'}}(\mu, \mathbf{S}, \mathbf{D})$ to obtain W in step ⑥ in Figure 1. Using this adversary, we get

$$Adv^{\mathsf{Hyb}_4}_{\mathcal{A}}(\lambda) \geq Adv^{\mathsf{Hyb}_3}_{\mathcal{A}}(\lambda) - Adv^{\mathsf{SubOpSim}}_{\mathsf{DVSC},\mathcal{A}_4}(n) \cdot$$

Hyb_5**:** During queries to $\mathcal{O}_{\mathsf{Show}_b}(\mathbf{D})$ on subsets $\emptyset \neq \mathbf{D} \subseteq \mathbf{S}_0 \cap \mathbf{S}_1$, instead of re–randomizing the set commitment C in step ⑤ in Figure 1 to obtain C', simply choose an independent random commitment C', i.e. $C' \xleftarrow{\$} (\mathbb{G}_1^*)^2$. If the advantage of $\mathcal{A}$ significantly changes,

then it can be shown that an adversary $\mathcal{A}_5$ can internally use $\mathcal{A}$ to break the multi-instance class-hiding property (cf. Definition 27) of the defined equivalence-class relation defined in Equation (1). Thus we have:

$$Adv_{\mathcal{A}}^{\mathsf{Hyb}_5}(\lambda) \geq Adv_{\mathcal{A}}^{\mathsf{Hyb}_4}(\lambda) - Adv_{\mathcal{R},\mathcal{A}_5}^{\mathsf{MI-CH}}(n) \cdot$$

Analysis of Hyb_5: When invoking $\mathcal{O}_{\mathsf{Show}_b}(\mathbf{D})$ in Hyb_5, the reply $\mathsf{Show} = (\tau', C', W)$ is independent of Cred_b and $\mathbf{S}_b$: C' is chosen randomly independent of $\mathsf{Cred}_b, \mathbf{S}_b$ (cf. Hyb_5). τ' is computed from only $\mathsf{sk}_{\mathsf{MEQ}}, C'$ (cf. Hyb_3). W is computed from only $\mathsf{sk}_{\mathsf{DVSC}}, C', \mathbf{D}$ using the DVSC witness simulator (cf. Hyb_4).

This means that $\mathcal{O}_{\mathsf{Show}_0}$ behaves exactly the same as $\mathcal{O}_{\mathsf{Show}_1}$, so the probability for the adversary to correctly guess b in the last hybrid is at most $1/2$.

$$1/2 \geq Adv_{\mathcal{A}}^{\mathsf{Hyb}_5}(\lambda) \cdot$$

This completes the proof, and we have:

$$\begin{aligned} Adv_{\mathsf{KVAC}_{\mathsf{MEQ}},\mathcal{A}}^{\mathsf{Unlink}}(\lambda) \leq 1/2 + \sum_{\beta=0}^{1} Adv_{NIZK,\mathcal{A}_1^{(\beta)}}^{\mathsf{PoK}}(n) \\ + 2Adv_{\mathbb{G}_1,\mathcal{A}_2}^{\mathsf{dlog}}(n) + Adv_{\mathsf{DVSC},\mathcal{A}_4}^{\mathsf{SubOpSim}}(n) + Adv_{\mathcal{R},\mathcal{A}_5}^{\mathsf{MI-CH}}(n) \cdot \end{aligned}$$

By assumption, all summands are negligible, so we have that the proposed KVAC scheme is unlinkable (note that dlog and multi-instance class-hiding are implied by DDH). □

G Proof of $\mathsf{KVAC}_{\mathsf{GGM}}$ security

G.1 Proof of Theorem 13 (Unforgeability)

Proof. The idea of the proof is to take an arbitrary adversary that outputs a valid credential presentation Show^* for a set $\mathbf{D}^*$. We then demonstrate, in the polynomial-based execution of the GGM (cf. Appendix B), that the only way for the adversary to produce such a valid credential presentation is if $\mathbf{D}^*$ is a subset of an attribute set for which it previously queried a credential.

First, we switch to a modified game where the adversary gets simulated NIZK proofs during the credential issuance phase. These simulated proofs don't reveal anything about the secret keys (x, v). Thanks to the zero-knowledge property of the NIZK proof system, the adversary cannot tell the difference between the original game and this simulated one. As a result, the adversary accepts the credentials from the credential oracle and gains no useful information from the NIZK proofs. Crucially, after this switch, x, v are only used by the experiment "in the exponent", which allows us to treat x, v as formal variables in the polynomial-based execution of the GGM.

Consider an execution of the modified game with the adversary $\mathcal{A}^{\mathcal{O}_{\mathsf{Cred}},\mathcal{O}_{\mathsf{Verify}}}(\mathsf{pp}, \mathsf{ipar})$ in the polynomial-based execution in the generic group model (see Appendix B). In this case, we treat the variables G, x, v, r, y_i as polynomial variables, i.e. as symbols. We work over the ring $\mathbb{Z}_p[\mathsf{G}, x, v, r, (y_i)_{i=1}^q]$.

Whenever the adversary queries $\mathcal{O}_{\mathsf{Verify}}(\mathsf{Show} = (\tau', W), \mathbf{D})$, the polynomial-based experiment returns the bit indicating whether $\tau' \neq 0$ and $\tau' = xW \cdot f_{\mathbf{D}}$ (where these equations are over polynomials). Since this oracle does not output any group elements, it does not play any further role below.

When the adversary queries $\mathcal{O}_{\mathsf{Cred}}(\mathbf{S}_i)$ for the ith time, the polynomial-based experiment returns (encodings of) $\mathsf{PreCred} = (\tau = xyf_{\mathbf{S}_i}\mathsf{G}, (Y_{j,i} = y_i v^j \mathsf{G})_{j=0}^{|\mathbf{S}_i|}, \pi)$, where $\pi \in \mathbb{Z}_p^3$ is simulated and $\tau, Y_{j,i} \in \mathbb{Z}_p[\mathsf{G}, x, v, r, (y_i)_{i=1}^q]$.

We now show that the adversary cannot linearly combine the group elements it gets from its input or from oracle queries to produce a valid credential presentation $\mathsf{Show}^* = (\tau^*, W^*)$ for a set $\mathbf{D}^*$ that is not a subset of any attribute set $\mathbf{S}_i$ it has queried. Let us assume, w.l.o.g., that the adversary has queried the credential oracle $\mathcal{O}_{\mathsf{Cred}}(.)$ a total of q times with attribute sets $\{\mathbf{S}_i\}_{i\in[q]}$. In return, it received $\{\tau_i\}_{i\in[q]}$ and $\{Y_{j,i}\}_{j\in\{0,1,\cdots,|\mathbf{S}_i|\},i\in[q]}$ as responses. This means the adversary now has access to the elements $(R, X, V, \{\tau_i\}_{i\in[q]}, \{Y_{j,i}\}_{j\in\{0,1,\cdots,|\mathbf{S}_i|\},i\in[q]})$ provided by the challenger. Using these, it can compute its credential presentation $\mathsf{Show}^* = (\tau^*, W^*)$.

$$\tau^* = \alpha R + \beta X + \gamma V + \sum_{i\in[q]} \eta_i \tau_i + \sum_{i\in[q]} \sum_{j=0}^{|\mathbf{S}_i|} \zeta_{j,i} Y_{j,i} \,,$$

$$W^* = \alpha' R + \beta' X + \gamma' V + \sum_{i\in[q]} \eta'_i \tau_i + \sum_{i\in[q]} \sum_{j=0}^{|\mathbf{S}_i|} \zeta'_{j,i} Y_{j,i} \,,$$

where α, β, γ, $\{\eta_i\}_{i\in[q]}$, and $\{\zeta_{j,i}\}_{j\in\{0,1,\cdots,|\mathbf{S}_i|\},i\in[q]}$ are coefficients that the adversary chooses from $\mathbb{Z}_p$ to compute τ^*. Similarly, the coefficients with primes are chosen from $\mathbb{Z}_p$ to compute W^*.

$\mathsf{Show}^* = (\tau^*, W^*)$ is considered valid if $\tau^* \neq 0$ and:

$$xW^* f_{\mathbf{D}^*} = \tau^*.$$

In what follows, we demonstrate that most of the coefficients must be zero, given that the adversary has no knowledge of the values x, r, v, and the randomness $\{y_i\}_{i\in[q]}$ used for each query:

- In the RHS (right hand side) of the equation, there are no monomials containing x^2. Since W^* is multiplied by a single x on the LHS (left hand side), this implies that the coefficients of the group elements that contain x (namely X and $\{\tau_i\}_{i\in[q]}$) in W^*, specifically β' and $\{\eta'_i\}_{i\in[q]}$, must be zero.
- Since all the monomials on the LHS contain x, the coefficients of the group elements on the RHS that do not contain x (namely R, V, and $\{Y_{j,i}\}_{j\in\{0,1,\cdots,|\mathbf{S}_i|\},i\in[q]}$) in τ^*, specifically α, γ, and $\{\zeta_{j,i}\}_{j\in\{0,1,\cdots,|\mathbf{S}_i|\},i\in[q]}$, must be zero.
- Since $\mathbf{D}^* \neq \emptyset$, if α' were non-zero, the LHS would contain some monomial of the form $rx\{v^i\}_{i\in[|\mathbf{D}^*|]}$, which are absent in the RHS. Therefore, α' must be zero.
- Since there is no monomial containing rx in the LHS, the coefficient of the group element X in τ^*, specifically β, must be zero.
- Since all remaining monomials on the RHS contain some term $\{y_i\}_{i\in[q]}$, the coefficient of V in W^* that does not include any $\{y_i\}_{i\in[q]}$, specifically γ', must be zero.

After removing all zero coefficients, the resulting equation should satisfy:

$$x \left(\sum_{i\in[q]} \sum_{j=0}^{|\mathbf{S}_i|} \zeta'_{j,i} Y_{j,i} \right) f_{\mathbf{D}^*}(v) = \sum_{i\in[q]} \eta_i \tau_i \,.$$

Given

$$Y_{j,i} = y_i v^j \mathsf{G} \,,$$

Table 5: The performance comparison of our proposed SP-MAC-EQ and FHS19's SPS-EQ. t_e, t_p denote the group's scalar exponentiation and pairing costs, respectively. $|\mathbb{G}_i|$ denotes the bit-length of elements in source group $\mathbb{G}_i$ for $i \in \{1,2\}$. ℓ denotes the vector size of message.

Scheme	Signature/ Tag Length	Sign/ MAC Cost	Verification Cost	ChangeRep. Cost
SPS-EQ [FHS19]	$2\|\mathbb{G}_1\| + \|\mathbb{G}_2\|$	$(\ell+2)t_e$	$(\ell+3)t_p$	$3t_e$
Our SP-MAC-EQ (Section 3.2)	$\|\mathbb{G}_1\| + \|\mathbb{G}_2\|$	$(\ell+1)t_e$	$2t_p + \ell t_e$	$2t_e$

$$\tau_i = xy_i f_{\mathbf{S}_i}(v)\mathsf{G}\,,$$

the coefficients of each y_i should be equal, resulting in:

$$\forall i \in [q] : x\left(\sum_{j=0}^{|\mathbf{S}_i|} \zeta'_{j,i} v^j \mathsf{G}\right) f_{\mathbf{D}^*}(v) = x\eta_i f_{\mathbf{S}_i}(v)\mathsf{G}$$

$$\rightarrow \forall i \in [q] : \left(\sum_{j=0}^{|\mathbf{S}_i|} \zeta'_{j,i} v^j\right) f_{\mathbf{D}^*}(v) = \eta_i f_{\mathbf{S}_i}(v)\,.$$

There should be at least one $k \in [q]$ such that $\eta_k \neq 0$ so that the resulting τ^* is not $0_{\mathbb{G}}$. Therefore:

$$\left(\eta_k^{-1}\sum_{j=0}^{|\mathbf{S}_k|} \zeta'_{j,k} v^j\right) f_{\mathbf{D}^*}(v) = f_{\mathbf{S}_k}(v)\,.$$

This implies that f_{D^*} divides $f_{\mathbf{S}_k}$ (as polynomials in v). Because $f_{\mathbf{S}_k} \neq 0$, this means $\mathbf{D}^* \subseteq \mathbf{S}_k$. As a result, the adversary can only produce a valid credential presentation for attributes $\mathbf{D}^*$ from a set $\mathbf{S}_k$ that it has already queried the credential oracle for, ensuring that any forgery attempt is unsuccessful. □

G.2 Proof of Theorem 14 (Unlinkability)

Proof.

Assuming the soundness of the NIZK holds, we have that any credential $\mathsf{Cred} = (\tau, (Y_j)_{j=0}^{|S|})$ is well-formed, i.e. $Y_j = v^j \cdot Y_0$ and $\tau = xyf_{\mathbf{S}}(v)\mathsf{G}$.

For each query on a set $\mathbf{D} \subseteq \mathbf{S}_0 \cap \mathbf{S}_1$, the oracle $\mathcal{O}_{\mathsf{Show}_b}(\mathbf{D})$ returns a uniformly random $W \in \mathbb{G}^*$ (due to randomization with μ and $W \neq 0_{\mathbb{G}}$), and, due to credential well-formedness, $\tau' = xWf_{\mathbf{D}}(v)$. Thus, regardless of the value of the bit b, the oracle returns a uniformly random group element W and a value τ' that is uniquely determined by W and $x, \mathbf{D}$, i.e. does not depend on b.

Consequently, the adversary's advantage in correctly guessing b is at most $\frac{1}{2} + \mathsf{negl}(\lambda)$ (where the negligible term is for the event that NIZK soundness fails).

□

H SP-MAC-EQ and SPS-EQ Efficiency Comparison

We substantiate our claim regarding the efficiency of SP-MAC-EQ compared to SPS-EQ through a detailed comparison of communication and computation costs, as summarized in Table 5.

For communication cost, we compare the number of group elements in the tag of SP-MAC-EQ with the signature of SPS-EQ. For computation cost, we analyze the total time required for three algorithms: MAC/sign, verify, and change representation. In this comparison, we disregard the costs of scalar multiplication, addition of two group points

Table 6: Total execution time (ms) and credential size for IssueCred and ObtainCred using different curves for pairingless KVAC system (Figure 2). The left number represents user execution time, and the right represents the issuer execution time.

Input Size	**User/Issuer time (ms)**			**Credential (KiB)**		
($\|\mathbf{S}\|$, $\|\mathbf{D}\|$)	Ed25519	Secp256k1	BLS12-381	Ed25519	Secp256k1	BLS12-381
(2^4, 2^3)	3.80/2.84	4.13/5.26	4.27/5.34	0.56	0.58	0.84
(2^6, 2^5)	14.70/10.27	14.02/20.06	14.33/20.57	2.06	2.13	3.09
(2^8, 2^7)	61.18/40.77	54.85/82.90	55.75/82.37	8.06	8.31	12.09
(2^{10}, 2^9)	253.89/159.91	218.19/344.21	221.57/345.96	32.06	33.06	48.09
(2^{12}, 2^{11})	1232.62/636.01	868.11/1640.90	884.36/1651.20	128.06	132.06	192.09

in $\mathbb{G}_1$ or $\mathbb{G}_2$, and multiplication of two group elements in G_T. The computation times for scalar exponentiation in a group and a pairing operation are denoted as t_e and t_p, respectively.

The results are as follows:

- The tag in SP-MAC-EQ is one group element smaller than the signature in FHS19's SPS-EQ.
- The MAC algorithm requires one additional scalar exponentiation compared to the signing algorithm in SPS-EQ.
- Verification of a tag requires $\ell + 1$ fewer pairing operations, which are computationally expensive in bilinear groups. This efficiency comes at the cost of ℓ extra scalar exponentiations during tag generation. Given that pairing operations are significantly more costly than scalar exponentiations, this results in a net gain in verification efficiency.
- The change representation algorithm in SP-MAC-EQ involves one fewer scalar exponentiation compared to SPS-EQ.

In summary, SP-MAC-EQ offers superior efficiency in both communication and computation compared to SPS-EQ.

I Detailed Performance Benchmarks

Performance of the $\mathsf{KVAC}_{\mathsf{GGM}}$ system highly depends on the performance of the underlying Elliptic Curves implementing the protocol. Therefore, we provide three implementations. The results of our benchmarks for Ed25519, Secp256k1 and BLS12-381 are listed in Tables 6 and 7.

Table 7: Total execution time (ms) and presentation size for ShowCred and Verify using different curves for pairingless KVAC system (Figure 2). The left number represents user execution time, and the right represents the verifier execution time.

Input Size	**User/Verifier time (ms)**			**Presentation (KiB)**		
($\|\mathbf{S}\|$, $\|\mathbf{D}\|$)	Ed25519	Secp256k1	BLS12-381	Ed25519	Secp256k1	BLS12-381
(2^4, 2^3)	0.72/0.06	0.99/0.08	0.97/0.09	0.06	0.06	0.09
(2^6, 2^5)	2.52/0.06	3.49/0.09	3.35/0.10	0.06	0.06	0.09
(2^8, 2^7)	9.89/0.07	13.70/0.08	13.05/0.09	0.06	0.06	0.09
(2^{10}, 2^9)	42.99/0.07	58.34/0.10	55.98/0.10	0.06	0.06	0.09
(2^{12}, 2^{11})	227.51/0.09	303.24/0.12	282.68/0.12	0.06	0.06	0.09