This ePrint is a preliminary draft dated 13:37 UTC, October 3, 2024. It has not been externally reviewed.

Revisiting Keyed-Verification Anonymous Credentials

Michele Orrù, CNRS
m@orru.net

Abstract

Keyed-verification anonymous credentials are widely recognized as among the most efficient tools for anonymous authentication. In this work, we revisit two prominent credential systems: the scheme by Chase et al. (CCS 2014), commonly referred to as CMZ or PS MAC, and the scheme by Barki et al. (SAC 2016), known as BBDT or BBS MAC. We show how to make CMZ statistically anonymous and BBDT compatible with the BBS RFC draft. We provide a comprehensive security analysis for strong(er) properties of unforgeability and anonymity. These properties allow them to be composed with extensions that users can pick and choose. We show that simpler variants satisfying one-more unforgeability can still be anonymous tokens (Kreuter et al., CRYPTO 2020).

To enable faster proofs for complex presentations, we present a compiler that uses an interactive oracle proof and a designated-verifier polynomial commitment to construct a designated-verifier non-interactive argument. For keyed-verification anonymous credentials, designated-verifier proofs suffice since the verifier is known in advance. We explore extensions that could benefit from this approach.

1 Introduction

Anonymous credential systems, introduced by David Chaum [Cha85], allow a user to obtain certified credentials from an organization, and later prove their possession, disclosing as little information as possible about the user's identity. Chaum envisioned a system where each organization would know a user by a different pseudonym. Pseudonyms are unlinkable, and colluding organizations cannot track a user.

After Chaum, a practical anonymous credential system was proposed in Stefan Brands's PhD thesis [Bra95, Bra00], leading to what is known today as the U-Prove technology from Microsoft Research.[1] Under the hood, Brands credentials tweak blind Schnorr signatures [Sch01] to support complex statements about the user.

Shortly after, Camenisch and Lysyanskaya [CL01, CL03] approached the problem of anonymous credentials by constructing a signature scheme over which it is possible to produce efficient zero-knowledge proofs (of knowledge). Their idea is to have the issuer sign a commitment to the user's attributes, and have the user later prove (in zk) knowledge of a signature on them. The proof might partially reveal some properties of the predicates, e.g. an "age" attribute being within a specific range. If the proof is valid, then via the knowledge extractor the reduction can recover a signature and a message, reducing the security of the credential to the one of the signature or the commitment scheme. If the attribute space of the signature scheme is "compatible" (*algebraic*) with the commitment space, proofs can be very efficient. This line of work led to the Idemix technology from IBM Research [CV02].

Chase, Meiklejohn, and Zaverucha [CMZ14] remarked that, if the issuer and the redeemer are the same entity, it is sufficient to rely on message authentication codes with similar algebraic properties (*algebraic MACs*), as opposed to signatures. They show that such schemes can be instantiated on prime-order groups without a bilinear pairing map. Such credentials are called *keyed-verification anonymous credentials*. Today, algebraic MACs are one of the most efficient and widespread approaches at anonymous credentials.

The simplest feature of anonymous credential systems is *selective disclosure*, where the user wants to authenticate to the server while keeping private some parts of their identity (we will call them *attributes*). However, this feature is woefully insufficient for real-world deployments, for which more involved access policies are needed. Numerous extensions of (publicly verifiable) anonymous credentials have been proposed in the literature to overcome this limitation: signatures with revocation [Sta96], rate-limiting credentials [TFS04], traceable anonymous credentials [BMW03], delegatable credentials [BCC+09], updatable credentials [BBDE19], threshold issuance of credentials [SAB+19, RP22, DKL+23], redactable credentials [San20], and more.

[1] https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/U-Prove20Cryptographic20Specification20V1.1.pdf

Limitations. Ideally, one would like to support similar extensions in the keyed-verification setting. For instance, on the web, often times a server needs to authenticate users and rate-limit their requests, or identify them under some context-based identity – generally cookies. Yet, the security notions of keyed-verification credentials (differently from, say, attribute-based credential systems) do not account for man-in-the middle attackers that can observe other user's presentation messages, do not state anything about the ability to recover an attribute from a credential, and do not support blind issuance with arbitrary predicates. For many applications (e.g. rate-limiting via a PRF key, age verification via passports), it is desirable to have this stronger security model.

Additionally, the concrete security and efficiency tradeoffs of the most widely known algebraic MACs ([CMZ14] and [BBDT17]) are unclear. To the best of our knowledge, the security of CMZ relies on the generic group model (or DDH), while the security of BBDT relies on q-SDH in the standard model. BBDT is generally thought to outperform CMZ in communication and computation (for issuance and redemption). The overall anonymity of the system is split into three security notions: key consistency, blind issuance, and anonymity, and it's hard to understand what are the long-term privacy guarantees of keyed-verification anonymous credentials.

1.1 Our contribution

As our first main contribution, we revisit the definition of keyed-verification credentials, and provide stronger notions of anonymity and unforgeability. We provide tight security proofs of the MAC of Chase, Meiklejohn, and Zaverucha [CMZ14], and Barki et al. [BBDT17] in the algebraic group model. We notice that both schemes can be improved in terms of efficiency and security, more specifically: CMZ is be made statistically anonymous, and issuance cost 1 group element instead of $2n + 1$ (where n is the number of private attributes), and BBDT can be made compatible with the ongoing standardization effort, with presentation and MAC cost accounting for 1 less group element. Our variants are denoted μCMZ and μBBS. We include a rigorous analysis of more efficient anonymous tokens derived from the above with weaker unforgeability guarantees, and compare the resulting schemes on the same grounds. Some of our techniques may also be of interest in the public verifiability case. The result, perhaps surprising, is that for the same bits of security, μCMZ is actually more efficient than μBBS at issuance, and presentation is less efficient for $n > 1$ attributes.

Then, we note that keyed-verification credentials can rely on malicious designated-verifier zero-knowledge proofs. We build a compiler that uses an interactive oracle proof (IOP) and a designated-verifier polynomial commitment to produce a designated-verifier zero-knowledge non-interactive argument. We build a designated-verifier polynomial commitment that does not require pairings, inspired from KZG. As an example application, we instantiate an efficient constant-sized range proof that does not require pairings. While it is common to think of designated-verifier proofs as "more expensive than normal proofs" (1 NIZK + 1 OR proof), we show that this is not always the case.

We provide efficient schemes for credential expiry, public metadata, pseudonyms, and rate-limiting. The schemes are agnostic on the credential system and the proof system used. We prove that sometimes one can rely solely on one-more unforgeability (instead of unforgeability) and thus use anonymous tokens where issuance even simpler.

1.2 Related work

The literature that builds modular credential systems is vast. Camenisch, Krenn, Lehmann et al. [CKL+16] develop a modular framework for constructing anonymous credentials called *privacy-enhancing attribute-based credential systems*. Using a commitment scheme, a signature scheme (more precisely, a privacy-enhancing attribute-based signature scheme), a revocation scheme and a pseudonym scheme, they show how to compose them into a full-fledged credential system. Their framework can be instantiated with Camenisch–Lysyanskaya [CL01, CL03] and Brands signatures [Bra95, Bra00], and has been foundational for extensions on credential systems, such as *issuer-hiding attribute-based credentials* [BEK+21], *encrypted attribute-based credentials* [KLSS17], and more. Chase and Lysyanskaya [CL06] introduce *signatures of knowledge*, signatures that can be issued on behalf of any NP statement. Belenkiy et al. [BCKL07] introduce the notion of *P-signatures*, a cryptographic primitive that consists of a signature scheme, a commitment scheme, a non-interactive protocol for obtaining signatures on committed values, and a non-interactive proof system for proving that a pair of commitments are commitments to the same value. Abe, Fuchsbauer, Groth et al. [AFG+16] introduced the notion of *structure-preserving signatures*, signatures that lie in a bilinear group for which the verification equation consists of pairing products. *Mercurial signatures* [CL19] are signatures that can be re-randomized into signatures valid under some (randomized) verification key for the same message, and some of them can be used for constructing anonymous credentials. Compared to all above works, instead of showing how to build full-fledged credentials from smaller cryptographic primitives, we define a (slightly different) "vanilla" credential system (that supports issuance predicates), and then put on together optional extensions that can be added at discretion of the user.

Campanelli, Fiore, and Querol introduce LegoSNARK [CFQ19], a framework to build modular SNARKs that can be composed to enable for different efficiency trade-offs. Their work is inscribed within the commit-and-prove paradigm and, more specifically, investigates SNARKs about commitments made ahead of time, without knowing the statement they are going to be proven on.

Our work is complementary, and is meant to be used in tandem with commit-and-prove systems. Malicious designated-verifier zero-knowledge was introduced by Quach, Rothblum, and Wichs [QRW19]. The log-sized arguments of Bootle et al. [BCC+16] have been proven to be simulation extractable [GOP+22, DG23].

Among real-world cryptographic deployments, we identify the following groups of credentials:

(a) *Based on Camenisch–Lysyanskaya* [CL01, CL03]. Bifold[2] and Aries RFC[3], both sponsored by the Open Wallet Foundation.

(b) *Based on Chase–Meiklejohn–Zaverucha* [CMZ14, PS16]. Used in Signal [CPZ20] for private group systems[4], by NYM Technologies[5], Tor [TG23] for distribution of bridges.

(c) *Based on BBS signatures* [BBS04, TZ23]. The W3C hosts multiple efforts in this direction, more specifically on decentralized identity[6] and verifiable credentials.[7] The IETF currently has an ongoing proposal for BBS credentials.[8] This is also what is adopted by Idemix[9] for their DLOG credentials, which are based on the work of Mu et al. [ASM06]. (Idemix is also implemented by IRMA[10].) BBS are also at the core of Dock[11], a blockchain network that provides reusable digital identification and verifiable credentials.

(d) *Tokens Based on Blind Signatures.* A number of show-once credentials have been created in the past years. The Privacy Pass IETF working group[12] internally relies either on the VOPRF of Jareki et al. [JKK14] or the Blind RSA signature scheme of Chaum [Cha82]. Google's BoringSSL implements anonymous tokens [KLOR20]. Blind RSA signatures are also used by Apple Cloud Relay[13] and Google One's VPN service.[14]

(e) *Based on SNARKs.* This broad class of credentials generally relies on SNARKs and recursive SNARKs [Chi10], and generally boils down to creating Merkle trees of secret keys in possession of users and then proving statements of membership and non-membership in order to authenticate. This class generally lacks a public provable-security formalisation. The Semaphore library[15] and Anon Aadhaar protocol[16] from Privacy Scaling Explorations (PSE) are a collection of tools used for building applications that leverage anonymous signaling on the Ethereum blockchain, relying on general-purpose zero-knowledge succinct arguments (zk-SNARKs). Zupass[17] is an authentication system based on the proof-carrying data paradigm.

Section 5 is relevant for (b); section 6 is relevant for (c); the anonymous token variants $\mu\mathsf{CMZ}_{AT}, \mu\mathsf{BBS}_{AT}$ may be relevant for approaches based on blind signatures (item (d)); section 7 may be relevant for approaches conceding generic SNARKS (item (e)).

2 Technical overview

An anonymous credential is a signature or a MAC over some *attributes*. Instead of just revealing them, the user proves (in zk) that the attributes satisfy some properties. We call this to *present* or *show* a credential. The entity issuing a credential is called *issuer*; the entity to which a credential is issued *user*. The entity that verifies the credential is called *redeemer*.

Public- and keyed-verification. It is possible to distinguish two lines of literature in the credential space: (i) keyed-verification credentials, where the issuer and the redeemer are the same person, and both hold the same signing key sk; (ii) public-verification credentials, where the issuer and the redeemer are different entities. In this case, the issuer holds a signing key sk and the redeemer holds the respective verification key vk. The verification key vk is insufficient to produce new credentials. It is a common practice to build public-verification credentials from keyed-verification credentials (and vice-versa) with the help of a pairing map in a bilinear group, but this is not true for all credential systems.[18] Two popular choices of credentials are Chase–Meiklejohn–Zaverucha [CMZ14] MACs, with their publicly-verifiable variant Pointcheval–Sanders [PS16]; and Boneh–Boyen–Shacham [BBS04], with their keyed-verification variant Barki et al. [BBDT17].

[2] https://github.com/openwallet-foundation/bifold-wallet
[3] https://hyperledger.github.io/aries-rfcs/latest/
[4] https://signal.org/blog/signal-private-group-system/
[5] https://nymtech.net/docs/coconut.html
[6] https://decentralized-id.com/web-standards/w3c/verifiable-credentials/data-integrity-bbs+/
[7] https://www.w3.org/TR/vc-data-model-2.0/
[8] https://datatracker.ietf.org/doc/draft-irtf-cfrg-bbs-signatures/
[9] https://github.com/hyperledger/fabric/blob/main/docs/source/idemix.rst
[10] https://github.com/privacybydesign/irmago
[11] https://github.com/docknetwork/crypto
[12] https://datatracker.ietf.org/wg/privacypass/documents/
[13] https://www.apple.com/icloud/docs/iCloud_Private_Relay_Overview_Dec2021.pdf
[14] https://one.google.com/about/vpn/howitworks
[15] https://semaphore.pse.dev/
[16] https://pse.dev/en/projects/anon-aadhaar
[17] https://github.com/proofcarryingdata/zupass
[18] For instance, it seems unlikely to produce a "keyed-verification variant" of Groth's structure preserving signatures [Gro15] without a pairing map.

Table 1: Concrete communication and space costs for some keyed-verification credential systems with n hidden attributes (public attributes at issuance/presentation are for free). The bit size of $\mathbb{G}$ is denoted g, the scalar field bit size s, and the security parameter λ. In [square brackets] we indicate the size for empty predicates or optional elements, using Schnorr proofs. ATR is checked if $n = \mathsf{poly}(\lambda)$ attributes are supported, PMB denotes private metadata bit feature, NYM refers to section 8.3, RTL to section 8.2, THRLD to thresholding, and IBR to efficient issuer-hiding presentation [BEK+21]. Empty cell represent open problems, not impossibility.

Scheme	Key material		Issuance		Presentation	Supported features						Security	
	$\lvert pp\rvert$	$\lvert\sigma\rvert$	$\lvert$I.Usr$\rvert$	$\lvert$I.Srv$\rvert$	$\lvert$P.Usr$\rvert$	ATR	PMB	NYM	RTLT	THRLD	IHP	Unforgeability	Anonymity
µCMZ, figure 5	$(n{+}1)g$	$g + [g]$	$g + [(n+2)s]$	$2g + [3s]$	$(n{+}2)g + [(2n{+}2)s]$	✓	✓	✓	✓	✓		AGM + 3-DL	statistical
µBBS, figure 6	$1g$	$1g + [2\lambda]$	$g + [(n+2)s]$	$g + 2\lambda + [2s]$	$2g + [(n+4)s]$	✓		✓	✓	✓		AGM + q-DL	statistical*
[BBDT17]	$1g$	$1g + [2\lambda]$	$g + [(n+2)s]$	$g + 2\lambda + s + [2s]$	$3g + [(n{+}7)s]$	✓		✓	✓	✓		q-SDH	statistical
[CMZ14]	$(n{+}1)g$	$g + [g]$	$(2n+1)g + [(2n+2)s]$	$3g + [(2n{+}4)s]$	$(n{+}2)g + [(2n{+}2)s]$	✓	✓	✓	✓	✓		GGM	DDH
[CPZ20]	$2g$	$2g + s$	$(3n+2)g + [4s]$	$3g + s + [(n{+}6)s]$	$(n+3)g + [4s]$	✓		✓	✓			GGM	DDH
[DGS+18]	$1g$	$2\lambda + [g]$	$1g$	$1g + [2s]$	4λ	✗	✗	✗	naïve	✓	✓	RO + gap-OMCDH	statistical

*A technicality in the simulator makes it difficult to simulate issuance when the user is invoked on a message $\vec{m}$ such that $\sum_i m_i G_i = -G_0$. This can be circumvented setting the message space to exclude this bad case or relying on the DL assumption during credential issuance.

Table 2: Concrete sizes over 256-bit curves. Relevant examples are secp256k1 [Qu99], Ristretto [Ber06, Ham15], BN254 curves [BN06], Pallas, Vesta, and JubJub curves. Note that the comparison does not take into account that, to have comparable levels of security, µBBS and [BBDT17] should be instantiated over larger (≈300 bit) curves.

Scheme	Attributes $n = 1$					Attributes $n = 5$				
	Key material		Issuance		Presentation	Key material		Issuance		Presentation
	$\lvert pp\rvert$	$\lvert\sigma\rvert$	$\lvert$I.Usr$\rvert$	$\lvert$I.Srv$\rvert$	$\lvert$P.Usr$\rvert$	$\lvert pp\rvert$	$\lvert\sigma\rvert$	$\lvert$I.Usr$\rvert$	$\lvert$I.Srv$\rvert$	$\lvert$P.Usr$\rvert$
µCMZ, figure 5	64B	64B	128B	160B	224B	192B	64B	256B	160B	608B
µBBS, figure 6	32B	64B	128B	128B	224B	32B	64B	256B	128B	352B
[BBDT17]	32B	64B	128B	160B	352B	32B	64B	256B	160B	480B
[CMZ14]	64B	64B	224B	288B	224B	192B	64B	736B	544B	608B
[CPZ20]	64B	96B	288B	352B	256B	64B	96B	672B	480B	384B
[DGS+18]	32B	64B	32B	96B	64B	32B	64B	32B	96B	64B

Table 3: Concrete sizes over ≈ 400-bit curves. Relevant examples are BLS12-381 and BLS12-377 [BLS04].

Scheme	Attributes $n = 1$					Attributes $n = 5$				
	Key material		Issuance		Presentation	Key material		Issuance		Presentation
	$\lvert pp\rvert$	$\lvert\sigma\rvert$	$\lvert$I.Usr$\rvert$	$\lvert$I.Srv$\rvert$	$\lvert$P.Usr$\rvert$	$\lvert pp\rvert$	$\lvert\sigma\rvert$	$\lvert$I.Usr$\rvert$	$\lvert$I.Srv$\rvert$	$\lvert$P.Usr$\rvert$
µCMZ, figure 5	96B	96B	144B	192B	272B	288B	96B	272B	192B	720B
µBBS, figure 6	48B	80B	144B	144B	256B	48B	80B	272B	144B	384B
[BBDT17]	48B	80B	144B	176B	400B	48B	80B	272B	176B	528B
[CMZ14]	96B	96B	272B	336B	272B	288B	96B	912B	592B	720B
[CPZ20]	96B	128B	368B	400B	320B	96B	128B	944B	528B	512B
[DGS+18]	48B	80B	48B	112B	64B	48B	80B	48B	112B	64B

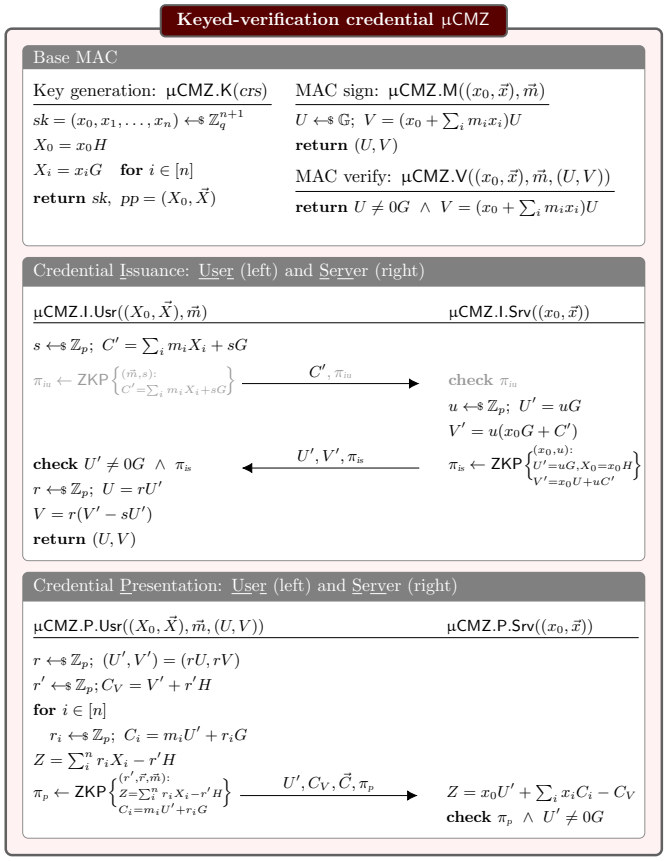
Security. Anonymous credentials enjoy two properties: unforgeability (it is not possible to present a credential that was not previously authorized) and anonymity (no information about the user is disclosed beyond what the user gave their consent to). Generally, it is desirable for anonymity to hold even against computationally unbounded adversaries. We call this property *statistical anonymity*. If, on the other hand, anonymity holds against computationally unbounded eavesdroppers, we talk about *everlasting forward anonymity* (Similarly to public-key encryption, where no information remains after a transaction is complete which could later identify the participants if one side or the other is compromised [DFM01].)

2.1 Chase–Meiklejohn–Zaverucha MACs

Given a secret key $(x_0, x_1, \dots, x_n)$, a $(n+1)$-tuple of integers uniformly distributed modulo a large prime p, a CMZ credential for attributes $\vec{m} \in \mathbb{Z}_p^n$ is a pair

$$(U \leftarrow\!\$\, \mathbb{G},\quad V = (x_0 + \textstyle\sum_i x_i m_i)\, U) \tag{1}$$

where U is sampled uniformly at random from $\mathbb{G}$, an additive group of order p where discrete logarithm is hard. Credentials can be trivially checked given the message $\vec{m}$ and the secret key $(x_0, \vec{x})$: the redeemer just has to re-compute the MAC and check for equality, being careful about some edge cases (when $U = 0G$). Unforgeability of CMZ was historically shown in the generic group model [CMZ14]. In theorems 7 and 11 we revisit the analysis for a more natural variant (illustrated below and denoted μCMZ) in the algebraic group model. We prove that it is unforgeable in the algebraic group model assuming it is hard to find x given $(G, xG, x^2G, x^3G) \in \mathbb{G}$. (Differently from e.g. BBS credentials, where a polynomial number of powers is required.) In a pairing group, when given a "verification key" in the "second group", i.e., given $vk = (\bar{X}_0 = x_0\bar{G}, \bar{X}_1 = x_1\bar{G}, \dots) \in \bar{\mathbb{G}}^{n+1}$, the verification equation can be checked in the "target group" checking $\mathsf{e}(V, \bar{G}) = \mathsf{e}(U, \bar{G}) + \sum_i \mathsf{e}(m_i U, \bar{X}_i)$, where $\mathsf{e} : \mathbb{G} \times \bar{\mathbb{G}} \mapsto \mathbb{G}_T$ denotes the bilinear pairing map. The resulting scheme is known as the Pointcheval–Sanders signature scheme [PS16, PS18].



Perfect blind issuance. Sometimes, we need to issue a MAC over some hidden ("blind") attributes, signed by the server, without fully knowing them. A classic use-case is when the credential is signing also a user's secret key. In these cases, the approach of Chase et al. [CMZ14] is to use the linear-homomorphic properties of ElGamal: given ciphertexts $(E_{1,i} = rG, E_{2,i} = rY + mG)$ of each attribute m_i (encrypted under the user's public key Y), one can blindly issue a credential by sampling $u \leftarrow\!\$\, \mathbb{Z}_p$, and computing

$$\begin{bmatrix} D_1 \\ D_2 \end{bmatrix} = \begin{bmatrix} 0 \\ ux_0G \end{bmatrix} + u \sum_i x_i \begin{bmatrix} E_{i,1} \\ E_{i,2} \end{bmatrix}$$

and sending $(U = uG, D_1, D_2)$ along with a zero-knowledge proof of correct computation. The above approach relies on semantic security of ElGamal encryption and therefore the user's attributes are at best computationally hidden. We change this to a Pedersen commitment to the attributes

$$C' = \sum_i m_i X_i + sG$$

where $X_1, \dots, X_n$ are part of the public parameters. Then, the signer computes $(U, V) = (uG, x_0U + uC')$ where $u \leftarrow\!\$\, \mathbb{Z}_p$. The user can unblind the credential subtracting sU' from C'. This variant of the issuance protocol is more efficient (as it is independent of the number of attributes) and perfectly hides the attributes.

One last remaining obstacle is in the way of statistical anonymity: in the original CMZ scheme, X_0 is computed as $X_0 = x_rG + x_0H$. While this means that x_0 is perfectly hidden, it also means that there exists many secret keys associated to the same public key, and a powerful adversary may be able to de-anonymize users by finding $\log_G H$ and "equivocating" X_0 using a different secret key to authenticate the credential. We fix this issue removing x_r from the public parameters and setting $X_0 = x_0H$. In theorem 12 we show that this does not affect unforgeability (in fact, satisfies an even stronger property), and it achieves statistical anonymity when instantiated with a statistically knowledge-sound zero-knowledge proof.

One-more unforgeability. The zero-knowledge proof π_{iu} sent by the user at issuance time in μCMZ may be removed at the price of trading off extractability with *one-more unforgeability*, which is the security notion of anonymous tokens [KLOR20, CDV23], lightweight spend-once credentials. The MAC procedure can in fact be seen as computing an authentication code $(U = uG, x_0U + uC)$ for a group element $C \in \mathbb{G}$, where no assumption about the algebraic representation of C is made. We prove this in theorem 14 by relying the algebraic group model. Surprisingly, this notion is sufficient also for complex systems such as pseudonyms and rate-limiting tokens.

Presentation. Thanks to the algebraic structure of the MAC, it is possible to efficiently prove knowledge of a MAC without revealing it. This is called to *present* a credential. A common approach here is to use Σ-protocols over prime-order groups for linear relations, using the Fiat–Shamir heuristic. For instance, given a CMZ credential $\sigma = (U, V)$, for attributes $(m_1, \ldots, m_n)$, the user can perfectly blind the MAC via

$$(U', C_V) = (rU, rV + r'H) \text{ where } r, r' \leftarrow\$ \mathbb{Z}_p$$

and commit to each attribute as a Pedersen commitment $C_i = m_iU' + r_iG$, where $i \in [n]$ and $r_i \leftarrow\$ \mathbb{Z}_p$. To anonymously present a valid MAC the user proves the following representation equality:

$$\textstyle\sum_i r_iX_i - r'H = x_0U' + \sum_i x_iC_i - C_V$$

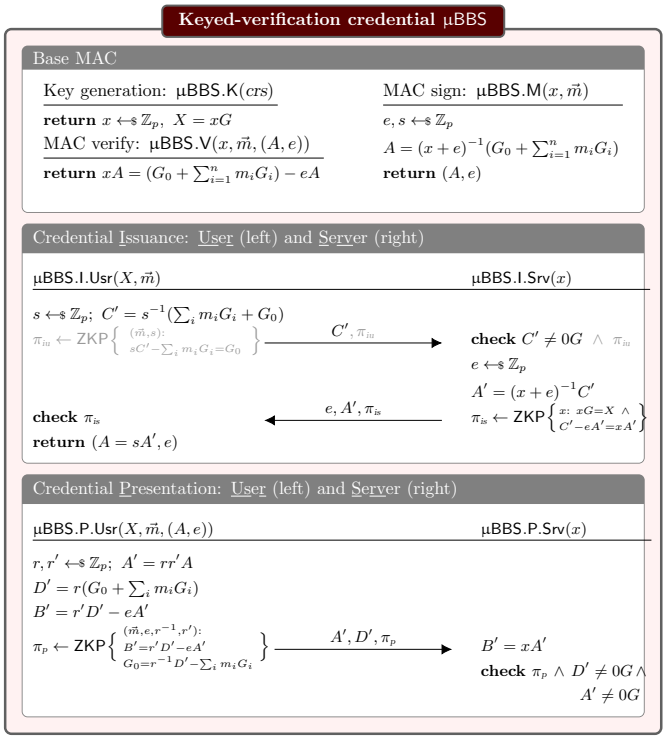
where the left-hand side can be computed by the user and the right-hand side can be computed by the issuer. The Pedersen commitments C_i's can be used to prove arbitrary predicates over the credentials.

2.2 Boneh–Boyen–Shacham MACs

A BBS+ anonymous credential [BBS04, ASM06] for attributes $\vec{m} \in \mathbb{Z}_p^n$ is a triple

$$\left(A = \frac{1}{x+e}(G_0 + \textstyle\sum_i m_iG_i + sG), \quad e \leftarrow\$ \mathbb{Z}_p, \quad s \leftarrow\$ \mathbb{Z}_p\right) \quad (2)$$

where x is the signing key, e, s are random in $\mathbb{Z}_p$, and $(G_0, \ldots, G_n)$ are random independent generators of $\mathbb{G}$. The verifier may check in a pairing group that $\mathsf{e}(A, \bar{X} + e\bar{G}) = \mathsf{e}(G_0 + \sum_i m_iG_i, \bar{G})$ where $\bar{X} = x\bar{G}$ is the verification key. Barki et al. [BBDT17] propose a keyed-verification variant where verification is performed using x, without relying on a pairing map. We propose a variant of BBDT's MAC scheme fixing $s = 0$ and thus considers the shorter MAC (A, e). We denote this variant μBBS and prove it is unforgeable in the algebraic group model under the q-DL assumption in lemma 19 and theorem 20. This analysis is similar to Tessaro and Zhu [TZ23], except that here we lack a pairing map for checking MAC validity, which requires a higher degree for the discrete logarithm challenge in the q-DL assumption. Additionally, to prove the security of the keyed-verification scheme, the reduction needs to re-compute B' without knowledge of the secret key x. We solve this problem proving a slightly stronger unforgeability game where the adversary is also given a DDH oracle.



Dropping the nonce on blind issuance. To issue a BBS MAC without knowing the message being authenticated, BBDT relies on the "s" term: the user computes the commitment $C = sG + \sum_i m_iG_i$, and sends it to the server, together with a zero-knowledge proof of representation in $G_1, \ldots, G_n, G$. The server computes the MAC over the commitment as $(A = (x + e)^{-1}(C + G_0), e)$ for a random $e \leftarrow\$ \mathbb{Z}_p$.[19] A similar approach is taken in the current RFC draft[20], where "s" term component it called *secret blind*. The main challenge here is to "unblind" the s term once signed, without carrying it along as part of the final MAC. We show how to do so considering multiplicative (instead of additive) blinding: $C = s^{-1}(\sum_i m_iG_i + G_0)$

[19] In [BBDT17], the server actually also re-randomises the commitment, but this shouldn't be required in light of [TZ23].
[20] https://www.ietf.org/archive/id/draft-kalos-bbs-blind-signatures-01.html

for a uniformly random $s \neq 0$. A representation proof can still be given, proving knowledge of $\vec{m}, s$ such that $sC - \sum_i m_i G_i = G_0$. (This is needed to reduce blind issuance of a credential to unforgeability of the underlying MAC scheme.) The commitment C is still uniformly distributed and perfectly hiding $\vec{m}$, unless $\sum_i m_i G_i = -G_0$, in which case a solution for DL has been found. The server computes the MAC as before, that is $(A = (x+e)^{-1}(C+G_0), e)$ for a random $e \leftarrow\!\$\ \mathbb{Z}_p$. The server's computation must be shown to have been done using $x = \log_G X$, which is generally done with the help of a zero-knowledge proof. The user can unblind the MAC by multiplying A by s. The resulting scheme still satisfies statistical anonymity for unbounded adversaries, except for the edge case described above where, at issuance time, the user's commitment is zero. Here, the simulator won't be able to reproduce the user distribution. However, it is possible to build a reduction to DL. To circumvent the problem, one may decide to exclude this bad case from the attribute space (that is, the user and server do not consider such message to be valid). We study this in more detail in theorem 21.

One-more unforgeability. Just like the previous scheme, also in this MAC it is possible to remove the user's issuance proof π_{iu}, maintaining one-more unforgeability. However, without any guarantee of the actual form of the user's message, a concrete attack (section 6.2.3) relying on Cheon [Che06] can be shown (*nota bene*, the reduction still relies on q-DL). The only way in which we know to break μBBS (with π_{iu}) and BBS signatures is the reduction by Jao and Yoshida [JY09], which requires all signatures to be on the same message, an important restriction of real-world cases. In the one-more unforgeability setting, this restriction is lifted (cf. lemma 24). When concretely evaluating the bits of security, the best attack known for the q-DL assumption is $O(\sqrt{d} + \sqrt{(p \pm 1)/d})$ where $d \mid p \pm 1$ is the number of issuance sessions required. Realistically, d can be upper-bounded at 2^{40} and around 20 bits of security are lost.

2.3 Extensions beyond selective disclosure

At credential issuance and presentation time, the zero-knowledge proofs π_{iu} and π_p prove knowledge of some attributes $\vec{m}$ (in the case of credential presentation, also knowledge of a valid MAC). If the proof system is powerful enough, it is possible additional "predicates" about the authenticated attributes. We denote such predicates as $\phi(\vec{m})$, and in the formal descriptions of μCMZ and μBBS (figures 5 and 6) they will be embedded in the credential issuance and presentation protocols.

Camenisch et al. [CHK+06] showed that, if the proof system allows proving PRF evaluations, various interesting credential extensions are possible. One such example are pseudonyms (section 8.3). In this scenario, the user possesses (as an attribute) a PRF key k generated (and MAC'd) at issuance time during a credential issuance protocol. The PRF key is used to generate context-based identities. Upon presenting the credential e.g. for accessing a website url (a *scope*) identified by a string *scp*, the user can produce a random and unlinkable pseudonym via $nym = \mathsf{PRF}(k, scp)$. Upon logging in, the user presents a valid credential for the PRF key (without revealing it) and proves that *nym* has been evaluated correctly. On the right, we provide an illustration of how presentation can be done for CMZ credentials highlighting the additional presentation material. The intuition behind the security of this system is that, since the PRF is a function, one cannot produce more than one identity for the same scope; since the PRF is pseudorandom, the ephemeral identities look random to any adversary that does not possess the key k. Some simple PRF choices, such as Naor–Pinkas–Reingold $\mathsf{NPR}(k, scp) = k\mathsf{H}_g(scp)$ are particularly efficient, and is the one used on the right.

Extension example: pseudonyms

Setup: μCMZ credential with $n = 1$ attributes.
Server has: (x_0, x_1) associated to public parameters $(X_0 = x_0 H, X_1 = x_1 G)$.
User has: (U, V) valid for a hidden attribute $k \in \mathbb{Z}_p$.
User wants to access some "scope" $scp \in \{0,1\}^*$ with a unique identity.
H_g is a random oracle with output in $\mathbb{G}$.

$\mathsf{P.Usr}((X_0, X_1), k, (U, V))$		$\mathsf{P.Srv}((x_0, x_1))$
$nym = k \cdot \mathsf{H}_g(scp)$		
$r \leftarrow\!\$\ \mathbb{Z}_p;\ (U', V') = (rU, rV)$		
$r' \leftarrow\!\$\ \mathbb{Z}_p; C_V = V' + r'H$		
$r_1 \leftarrow\!\$\ \mathbb{Z}_p;\ C_1 = kU' + r_1 G$		
$Z = r_1 X_1 - r'H$		
$\pi_p \leftarrow \mathsf{ZKP}\{(r', \vec{r}, \vec{m}) : nym = k\mathsf{H}_g(scp) \wedge Z = r_1 X_1 - r'H \wedge C_1 = kU' + r_1 G\}$	$\xrightarrow{nym,\ U', C_V, C_1, \pi_p}$	$Z = x_0 U' + x_1 C_1 - C_V$ **check** $\pi_p \wedge U' \neq 0G$

Extension example: rate limiting

Setup: μCMZ credential with $n = 1$ attributes.
Access at most ℓ times a server, each time spending a token "*tkn*".
Server has: (x_0, x_1) associated to public parameters $(X_0 = x_0 H, X_1 = x_1 G)$.
User has: (U, V) valid for a hidden attribute $k \in \mathbb{Z}_p$.
User wants to access some "scope" $scp \in \{0,1\}^*$ for the i-th time $(i \leq \ell)$.
H_g is a random oracle with output in $\mathbb{G}$.

$\mathsf{P.Usr}((X_0, X_1), k, (U, V))$		$\mathsf{P.Srv}((x_0, x_1))$
$tkn = (k+i)^{-1} \cdot \mathsf{H}_g(scp)$		
$r \leftarrow\!\$\ \mathbb{Z}_p;\ (U', V') = (rU, rV)$		
$r' \leftarrow\!\$\ \mathbb{Z}_p; C_V = V' + r'H$		
$r_1 \leftarrow\!\$\ \mathbb{Z}_p;\ C_1 = kU' + r_1 G$		
$Z = r_1 X_1 - r'H$		
$\pi_p \leftarrow \mathsf{ZKP}\{(r', \vec{r}, \vec{m}, i) : Z = r_1 X_1 - r'H \wedge \mathsf{H}_g(scp) = k \cdot tkn + i \cdot tkn \wedge i \leq \ell \wedge C_1 = kU' + r_1 G\}$	$\xrightarrow{tkn,\ U', C_V, C_1, \pi_p}$	$Z = x_0 U' + x_1 C_1 - C_V$ **check** $\pi_p \wedge U' \neq 0G$

Another example, relevant in practice, is rate-limiting anonymous tokens (section 8.2). In this scenario, a server wants to budget the number of requests users can make, without tracking them. In other words, we are looking for an efficient credential that is keyed-verification and allows users to present many times. The protocol must be correct (all tokens issued to honest users verify), unforgeable (users cannot present more tokens than allowed by the system), and unlinkable (all tokens are indistinguishable, even if coming from the same user). A common approach here is to issue many blind signatures (or VOPRFs) upon authentication: the server holds a list of "spent tokens", and issues batches of credentials to

users, each for a different hidden attribute (a *token*), that is revealed upon presentation.[21] However, this approach is very expensive per-se, and different techniques have been introduced to scale this solution in large services doing rate-limiting across different *scopes* (e.g., a URL) [SS22, AYY23]. An alternative approach, initially described in the context of k-times anonymous credentials [TFS04, CHK+06], is to use a PRF evaluation over the pair (scp, i), where $0 \le i < \ell$ is a counter kept by the user for each access in scp.

A variant of Dodis–Yampolskiy $\mathsf{HashDY}(k, (i, scp)) = (k+i)^{-1}\mathsf{H}_g(scp)$ allows for even easier proofs in cases where parts of the scope (e.g. "i ", a rate-limit counter) are meant to be hidden. The use of Dodis–Yampolskiy in the way we describe it has been introduced in [ASM06], but we give a formal proof of its security in theorem 35 with more precise security bounds.

2.4 Instantiating the zero-knowledge proofs

Straight-line extraction for Σ-protocols. For all above proofs, Σ-protocols are the go-to choice given the simplicity of the relations. However, formally the security of the resulting system is tedious to argue. Σ-protocols have been historically studied using rewinding, whereas credentials often times need to rely on straight-line extraction techniques. Theoretically, this problem has been circumvented with the Fischlin transform [Fis05] but, due to the lack of concrete attacks over the strong Fiat–Shamir heuristic [FS87, BPW12] its adoption has been underwhelming. To fill the gap between theory and practice, we show straight-line extractability in the algebraic group model for linear relations:

$$\mathsf{R}_{\mathbf{F}} = \left\{ (\vec{x}, \vec{X}) \in \mathbb{Z}_p^n \times \mathbb{G}^m : \sum_k^n x_k F_{1,k} = X_1 \ \wedge \ \cdots \ \wedge \ \sum_k^n x_k F_{m,k} = X_m \right\} ,$$

where:

- each linear constraint $\vec{F}_j$ is hard: it is computationally hard to find a non-zero vector $\vec{w} \in \mathbb{Z}_p^n$ such that $\sum_k w_k F_{j,k} = 0G$;
- all non-trivial group elements appearing in the matrix are independent. In other words, for any p.p.t. adversary A, it is computationally hard to find a non-zero vector $\vec{w}$ such that $\sum_k w_k G_k = 0$, where $(G_k)_k$ are the distinct non-zero elements appearing in $\mathbf{F}$.

Informally, the above conditions are necessary to avoid the adversary "malleating" a simulated proof, for instance $(\vec{R}, \vec{s})$ with $\vec{w} \in \ker(\mathbf{F})$, and returning $(\vec{R}, \vec{s} + \vec{w})$. While not all our instantiations rely on statements of this form, we restrict the possible attack vectors, and consider it to be of independent interest.

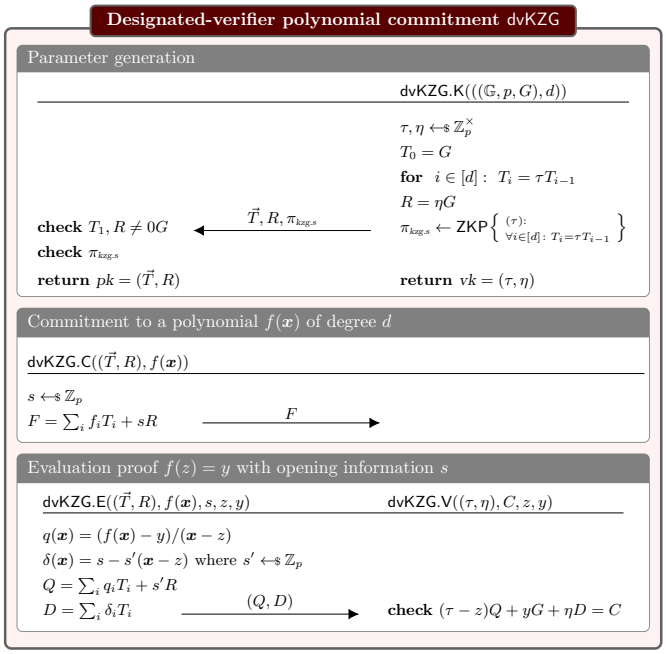
Compatibility with modern SNARKs. For keyed-verification credentials, since the verifier is known in advance, designated-verifier zero-knowledge proofs are sufficient for issuance and presentation. We present a slightly-modified version of the Kate–Zaverucha–Goldberg [KZG10] that is designated verifier and does not rely on pairing-friendly groups. Consider the KZG commitment scheme, where the commitment key is:

$$pk = (G, \tau G, \tau^2 G, \ldots, \tau^{n-1} G) \tag{3}$$

and the trapdoor τ is set to be the verification key. A KZG commitment to a polynomial $f \in \mathbb{Z}_p[x]$ is a group element $C = f(\tau)G$ computed via add-and-multiply operations over the commitment key. An opening proof for $z \in \mathbb{Z}_p$, that is, a proof that $f(z) = y$ is

$$Q = q(\tau)G$$

where $q(x)$ is the quotient of the Euclidian division of $f(x) - y$ by $(x - z)$. The proof can be trivially checked in the group using the trapdoor, and soundness follows the same argument given in [KZG10]. Kohrita and Towa [KT23] provide an efficient way for hiding KZG commitments. The user now "blinds" f committing to it as $f(\tau)G + rH$ for $r \leftarrow\!\$\, \mathbb{Z}_p$, and proceeds similarly for the quotient polynomial. An evaluation proof now has an additional term to "balance out" the blinding factors of the verification equation.



In the designated-verifier case, the (malicious) verifier can build the trapdoor information itself, preserving extractability of the commitment. But zero-knowledge doesn't hold! In fact, we have no guarantees that the proving key doesn't come from

[21] https://datatracker.ietf.org/doc/draft-ietf-privacypass-rate-limit-tokens/

a bad distribution that does not preserve hiding (a trivial example is when $H = 0G$). We show that it is possible to prove zero-knowledge (see theorem 27) for any proving key that is "well-formed", independently of its distribution.

One can prove that the commitment key is well-formed (and argue security against malicious verifiers) by equipping pk with a proof that the powers of τ have been correctly encoded. Concretely, this amounts to 1 DLEQ proof (2 group elements). Upon receiving pk, a user would check that the proof is valid, and that no trivial (identity) elements are present in pk.

The most common ingredients for building non-interactive SNARKs today are interactive oracle proofs (IOPs) [BCS16] and commitment schemes. In an IOP, the prover may send field elements as messages, as well as special "oracle messages", that the verifier can query. Roughly speaking, the "IOP oracles" are replaced by commitments to instantiate the protocol and build an argument. Among the most common combinations of IOP+commitment we have polynomial IOPs (PIOPs), where the prover may send polynomial as oracle messages and the verifier may ask for polynomial evaluations to those oracles, and polynomial commitment scheme (PCS). We show that, in the designated-verifier case, one can build a designated-verifier argument via a variant of the IOP compiler given in [CHM+20], that uses a polynomial IOP and a designated-verifier commitment scheme. See theorem 29.

As an example application, we provide a constant-sized range proof protocol that is both asymptotically and concretely more efficient than generic algebraic approaches for keyed-verification credentials. The LegoSNARK framework [CFQ19] is a helpful tool to bridge presentation proofs (which are generally done with Σ-protocols over generic linear relations) with the arguments resulting from our compiler, via the commit-then-prove paradigm.

3 Preliminaries

Notation. The security parameter is denoted λ. We consider the group generation procedure $\mathsf{GrGen}(1^\lambda)$ which takes as input the security parameter in unary form 1^λ and outputs a group description $\Gamma = (\mathbb{G}, p, G)$ consisting of a group $\mathbb{G}$ of odd prime order p generated by $G \in \mathbb{G}$ where discrete logarithm (DL) is hard. Throughout this work we will use two random oracles: H_p, with image over the field $\mathbb{Z}_p$, and H_g, with image over the group $\mathbb{G}$. The domain of both oracles is the set of strings of arbitrary length.

The writing $[n]$ denotes the range $\{1, \ldots, n\}$ and $[n, m]$ denotes the range $\{n, n+1, \ldots, m\}$ if $n \leq m$, otherwise it denotes the empty set. Assignment is denoted ":=", which we distinguish from assignment of the output of a randomized algorithm, where we use the symbol "$\leftarrow$" to underscore the sampling of random coins before assignment. The range of a p.p.t. algorithm A (that is, the set of possible outputs that happen with non-zero probability) is denoted $[\mathsf{A}(x)]$; the number of random coins as $\mathsf{A.rl}(\lambda)$. The notation $z \leftarrow (\mathsf{A}(x) \rightleftharpoons \mathsf{B}(y))$ denotes the interaction of the interactive Turing machine A on the input x with the (interactive) Turing machine B on input y. The output of A (the machine on the left-hand side) is assigned to z. When both machines return some value, respectively z_0 for A and z_1 for B, we separate them with the semicolon, i.e. $(z_0; z_1) \leftarrow (\mathsf{A}(x) \rightleftharpoons \mathsf{B}(y))$.

3.1 Cryptographic assumptions

The discrete logarithm (DL) assumption is hard for GrGen if it is computationally hard, given $X \in \mathbb{G}$ uniformly distributed, to compute $x \in \mathbb{Z}_p$ such that $xG = X$, where $(\mathbb{G}, p, G)$ is output by GrGen. The decisional Diffie–Hellman (DDH) assumption holds for GrGen if it is hard to distinguish the tuple (P, aP, bP, abP) from (P, aP, bP, cP) with $a, b, c \leftarrow\!\!\$\ \mathbb{Z}_p$ and $P \in \mathbb{G}$. Their advantage w.r.t. an adversary A will be denoted respectively as $\mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$, $\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$. The gap discrete logarithm assumption is hard if DL is hard even in the presence of a DDH oracle for the DL challenge.

q-type assumptions. The q-strong discrete logarithm (q-DL) assumption holds for the group generator GrGen if it is hard for any p.p.t. adversary to recover x uniformly distributed over $\mathbb{Z}_p$ given as input $(G, xG, x^2G, \ldots, x^qG)$. Its advantage w.r.t. an adversary A is denoted $\mathsf{Adv}^{\text{q-dl}}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$. The q-DDHI assumption holds for the group generator GrGen if it is hard for any p.p.t. adversary to distinguish $(xG, x^2G, \ldots, x^qG, x^{-1}G)$ from $(xG, x^2G, \ldots, x^qG, Z)$ for $x \leftarrow\!\!\$\ \mathbb{Z}_p$ and $Z \leftarrow\!\!\$\ \mathbb{G}$. Its advantage w.r.t. an adversary A is denoted $\mathsf{Adv}^{\text{q-ddhi}}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$.

Algebraic Group Model. In the algebraic group model [FKL18], adversaries are assumed to know a representation of any group element they return. This means that, after having received group elements $Z_1, \ldots, Z_n$, whenever the adversary returns a group element X, it must also return coefficients $\zeta_1, \ldots, \zeta_n$ so that $X = \sum_i \zeta_i Z_i$. We call such adversaries *algebraic*.

Generic Group Model. In the generic group model [Sho97] the adversary is only given access to randomly-chosen encodings of group elements, and an oracle that executes the group operation: the oracle takes two encodings and returns the encoding of the sum. If the oracle supports a pairing operation, an oracle for the pairing map is also given.

Game $\text{UF-CMVA}_{\mathsf{MAC},\mathsf{A}}(\lambda, n)$	Oracle $\textsc{Sign}(\vec{m})$
$Qrs \coloneqq \emptyset$; $crs \leftarrow \mathsf{MAC.S}(\lambda, n)$	$Qrs \coloneqq Qrs \cup \{\vec{m}\}$
$(sk, pp) \leftarrow \mathsf{MAC.K}(crs)$	**return** $\mathsf{MAC.M}(sk, \vec{m})$
$(\vec{m}^*, \sigma^*) \leftarrow \mathsf{A}^{\textsc{Sign},\textsc{Verify}}(pp)$	Oracle $\textsc{Verify}(\vec{m}, \sigma)$
return $\vec{m}^* \notin Qrs \ \wedge \ \mathsf{MAC.V}(pp, \vec{m}^*, \sigma^*) = 1$	**return** $\mathsf{MAC.V}(sk, \vec{m}, \sigma)$

Figure 1: Unforgeability of an algebraic message authentication code MAC.

3.2 Algebraic message authentication codes

Definition 1. An **algebraic message authentication code** (MAC) for n attributes over message family $\mathcal{M} = \{\mathbb{M}_\lambda\}_\lambda$ is a tuple $\mathsf{MAC} = (\mathsf{S}, \mathsf{K}, \mathsf{M}, \mathsf{V})$:

- $crs \leftarrow \mathsf{MAC.S}(1^\lambda, n)$ The setup algorithm, which takes as input the security parameter in unary form and the number of attributes, and outputs a common-reference string crs.

  The setup algorithm implicitly defines an attribute space $\mathbb{M}_{crs} = \mathbb{M}$ (we drop the "index" crs to ease notation) describing the family of messages supported by the MAC scheme.

- $(sk, pp) \leftarrow \mathsf{MAC.K}(crs)$ The key generation algorithm, which takes as input the crs and outputs a secret key sk and some public parameters pp.

  In the remainder of this paper, we will omit crs from inputs of the algorithms below for brevity, and assume that it can be implicitly derived from either sk or pp.

- $\sigma \leftarrow \mathsf{MAC.M}(sk, \vec{m})$ The MAC algorithm, which takes as input the signing key sk and attributes $\vec{m} \in \mathcal{M}^n$ and outputs a message authentication code σ.

- $0/1 \coloneqq \mathsf{MAC.V}(sk, \vec{m}, \sigma)$ The (deterministic) verification algorithm, which takes as input the signing key sk, a message $\vec{m}$, and a MAC σ, and outputs 1 if the MAC is valid, and 0 otherwise.

MAC must satisfy *correctness* (every honestly-generated MAC verifies) and *unforgeability* (it is hard to forge MACs). We denote the unforgeability advantage of A with respect to the algebraic message authentication code MAC on n attributes as $\mathsf{Adv}^{\text{ufcmva}}_{\mathsf{MAC},\mathsf{A}}(\lambda, n) \coloneqq |\Pr[\text{UF-CMVA}_{\mathsf{MAC},\mathsf{A}}(\lambda, n)]|$, where $\text{UF-CMVA}_{\mathsf{MAC},\mathsf{A}}(\lambda, n)$ is illustrated in figure 1.

Remark 1. Differently from MACs, algebraic MACs are randomized algorithms. It is possible to construct a "de-randomized" algebraic mac in the random oracle model using the message to seed the randomness, i.e. letting $\mathsf{H}(\vec{m})$ be the random coins for the MAC algorithm.

3.3 Zero-knowledge proofs of knowledge

A non-interactive proof system ZKP for a relation family $\mathcal{R} = \{\mathsf{R}_\lambda\}_\lambda$ consists of the following three algorithms:

- $crs \leftarrow \mathsf{ZKP.S}(1^\lambda)$, the setup algorithm, which takes as input the security parameter in unary form and outputs a common reference string crs. It implicitly selects a relation R_{crs}.

- $\pi \leftarrow \mathsf{ZKP.P}(crs, x, w)$, a prover which takes as input $(x, w) \in \mathsf{R}$ and outputs a proof π.

- $0/1 \leftarrow \mathsf{ZKP.V}(crs, x, \pi)$, a verifier that, given as input an instance x together with a proof π outputs 0 if the proof is rejected and 1 otherwise.

For clarity, we will talk about a relation R assuming it is indexed by the range of crs, and omit the common reference string from the prover and verifier algorithms. A proof system is complete if every correctly-generated proof for an element of R verifies.

A proof system is **a knowledge-sound argument** [BLCL91] if there exists an extractor Ext that takes as input the random coins and the code of the p.p.t. adversary A (optionally, a trapdoor for the crs, bot not required in our instantiations) such that, whenever $\mathsf{A}(crs)$ outputs (x, π), then Ext outputs w. The adversary wins if $\mathsf{ZKP.V}(crs, x, \pi) = 1 \ \wedge \ (x, w) \notin \mathsf{R}$. The advantage is denoted as $\mathsf{Adv}^{\text{ksnd}}_{\mathsf{ZKP},\mathsf{Ext},\mathsf{A}}(\lambda)$. A proof system is **zero-knowledge** [GMR89] if there exists a simulator Sim such that, for any adversary A,

$$\mathsf{Adv}^{\text{zk}}_{\mathsf{ZKP},\mathsf{Sim},\mathsf{A}}(\lambda) \coloneqq \left| \Pr\left[b' = 1 : \begin{array}{l} crs \leftarrow \mathsf{ZKP.S}(1^\lambda) \\ b' \leftarrow \mathsf{A}^{\textsc{Prove}_0}(crs) \end{array} \right] - \Pr\left[b' = 1 : \begin{array}{l} crs \leftarrow \mathsf{ZKP.S}(1^\lambda) \\ b' \leftarrow \mathsf{A}^{\textsc{Prove}_1}(crs) \end{array} \right] \right| \leq \mathsf{negl}(\lambda) \ .$$

Game $\mathrm{OMUF}_{\mathsf{AT},\mathsf{A}}(\lambda, n)$	Oracle $\textsc{Sign}(\mu)$
$q \coloneqq 0$	$q = q + 1$
$crs \leftarrow \mathsf{AT.S}(1^\lambda, n)$	**return** $\mathsf{AT.I.Srv}(sk, \mu)$
$(sk, pp) \leftarrow \mathsf{AT.K}(crs)$	
$(\vec{m}_i, \sigma_i)_{i=1}^{q+1} \leftarrow \mathsf{A}^{\textsc{Sign},\textsc{Verify}}(pp)$	Oracle $\textsc{Verify}(\vec{m}, \sigma)$
return $\forall i \neq j \colon \vec{m}_i \neq \vec{m}_j \wedge \forall i \in [q+1] \colon \mathsf{AT.V}(sk, \vec{m}_i, \sigma_i) = 1$	**return** $\mathsf{AT.V}(sk, \vec{m}, \sigma)$

Figure 2: One-more unforgeability game for an anonymous token scheme AT with non-interactive issuance. The variable "μ" denotes the issuance request message from the user.

Game $\mathrm{UNF}_{\mathsf{KVC},\mathsf{A}}(\lambda, n)$	Oracle $\textsc{Sign}(\vec{m})$
$Qrs \coloneqq \emptyset$	$Qrs \coloneqq Qrs \cup \{\vec{m}\}$
$crs \leftarrow \mathsf{KVC.S}(1^\lambda, n)$	**return** $\mathsf{KVC.M}(sk, \vec{m})$
$(sk, pp) \leftarrow \mathsf{KVC.K}(crs)$	
$(\phi^*, \rho^*) \leftarrow \mathsf{A}^{\textsc{Sign},\textsc{Present}}(pp)$	Oracle $\textsc{Present}(\phi, \rho)$
$b \leftarrow \mathsf{KVC.P.Srv}(sk, \phi^*, \rho^*)$	**return** $\mathsf{KVC.P.Srv}(sk, \phi, \rho)$
return $(b = 1) \ \wedge \ (\forall \vec{m} \in Qrs \colon \phi^*(\vec{m}) = 0)$	

Figure 3: Canonical unforgeability game for a keyed-verification credential system KVC [CMZ14]. The variables "ϕ, ϕ^*" denotes the algebraic predicate to be shown on $\vec{m}$ (cf. definition 2) while "ρ" denotes the presentation message.

where $\textsc{Prove}_b(x, w)$ checks if $(x, w) \in \mathsf{R}$ and outputs $\mathsf{ZKP.P}(crs, x, w)$ if $b = 0$ and $\mathsf{Sim}(crs, x)$ if $b = 1$. We assume that both adversary and simulator have access to a random oracle, and that the simulator can explicitly re-program the random oracle. A proof system is (strongly) **simulation-extractable** [DHLW10] if it is knowledge-sound even when the adversary has access to simulated proofs, and the output pair (x, π) was not previously returned by the zero-knowledge simulator. For the formal definitions, see Dao and Grubbs [DG23, Fig. 2, 3, and 4].

3.4 Anonymous Tokens

An anonymous token [KLOR20] is a keyed-verification blind signature (with an optional private metadata bit). More formally, an anonymous token scheme AT for $n > 0$ attributes over attribute family $\mathcal{M}$ is a tuple $\mathsf{AT} = (\mathsf{S}, \mathsf{K}, \mathsf{I}, \mathsf{V})$ where $\mathsf{AT.S}(1^\lambda, n)$ outputs a crs; $\mathsf{AT.K}(crs)$ outputs a signing key sk and some public parameters pp; the issuance protocol involves a user $\mathsf{AT.I.Usr}(pp, \vec{m})$ and a server $\mathsf{AT.I.Srv}(sk)$ and produces a token σ for the user; the verification algorithm $\mathsf{AT.V}(sk, \vec{m}, \sigma)$ returns $0/1$ if the credential is accepted for the message $\vec{m}$. We demand anonymous tokens to be correct, one-more unforgeable, and unlinkable.

Correctness means that all tokens generated via $\mathsf{AT.I}.\{\mathsf{Usr}, \mathsf{Srv}\}$ for messages in the family successfully verify. One-more unforgeability asks that, after blindly issuing at most q credentials, no adversary can successfully present $q + 1$ valid message/-credential pairs. More formally, given an anonymous token scheme AT for $n > 0$ attributes over attribute family $\mathcal{M}$, an adversary A has one-more unforgeability advantage

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{AT},\mathsf{A}}(\lambda) \coloneqq \Pr\left[\mathrm{OMUF}_{\mathsf{AT},\mathsf{A}}(\lambda, n) = 1\right]$$

where $\mathrm{OMUF}_{\mathsf{AT},\mathsf{A}}(\lambda, n)$ is illustrated in figure 2.

Unlinkability (similarly to blindness for blind signatures) demands that, any malicious issuer, after blindly issuing q credentials to honest users and then observing a permutation of the issued credentials, cannot to better than guessing the link between a credential and its issuance. Looking ahead, keyed-verification credential systems (definition 3) are also anonymous tokens: the syntax is the same (except for predicates) but the security guarantees are weaker: instead of asking for extractability (definition 6) and anonymity (definition 5), we demand one-more unforgeability and unlinkability. We will exploit the weaker security requirements of unforgeability to produce more efficient schemes, but will maintain the stronger anonymity properties and therefore omit the definition of unlinkability in this work, and direct the curious reader to Kreuter et al. [KLOR20, Fig. 4] and Chase, Durak, Vaudenay [DVC22, Fig. 3].

4 Keyed-verification credential systems

We first generalize the issuance algorithm of credential systems (procedures BlindIssue and BlindObtain in Chase et al. [CMZ14] and here $\mathsf{KVC.I}.\{\mathsf{Usr}, \mathsf{Srv}\}$) so that arbitrary predicates about the attributes over which a credential is about to be issued can be proven. Then, we re-define security and given a unique notion unforgeability and anonymity.

4.1 Syntax

We enrich the classical definition of keyed-verification anonymous credentials (KVAC) with issuance with predicates beyond partial disclosure of attributes. In this setting, at issuance time the user can partially disclose some attributes satisfying a predicate $\phi \in \Phi$ over them. For the sake of simplicity, we use the same predicate family employed in the presentation protocol, and make it explicit when they are not. This is a generalization of the traditional definition, where the predicate ϕ is concerned with disclosure of only some attributes, and where the server has no memory of previously-seen credentials.

Definition 2. An **algebraic predicate** is an efficiently-computable function $\phi(\vec{m})$ mapping some elements $\vec{m} \in \mathbb{M}^n$ ($\mathbb{M}$ referred to as *attribute space*) to a boolean value (1 if the predicate is satisfied, 0 otherwise). With $\phi_\emptyset$ we denote the trivial predicate that always returns 1. A **predicate family** is a non-empty set of algebraic predicates $\Phi = \{\phi \colon \mathbb{M}^n \to \{0,1\}\}$ containing the trivial predicate and closed under conjunction of statements (i.e., if $\phi_1, \phi_2 \in \Phi$ then $\phi_1 \wedge \phi_2 \in \Phi$).

Whenever the predicates is parametrized by some constants $\vec{a}$ we will denote it as $\phi^{\vec{a}}(\vec{m})$.

Definition 3. A **keyed-verification credential system** $\mathsf{KVC} = (\mathsf{S}, \mathsf{K}, \mathsf{I}, \mathsf{P})$ for predicate family Φ over a message family $\mathcal{M} = \{\mathbb{M}_\lambda\}_\lambda$ for $n \in \mathbb{N}$ attributes is a tuple of algorithms:

- $crs \leftarrow \mathsf{KVC.S}(1^\lambda, n)$ The setup algorithm, which takes as input the security parameter in unary form and the max number of attributes $n > 0$, and outputs a common-reference string crs.

  The setup algorithm implicitly defines an attribute space $\mathbb{M}$ and a predicate family Φ.

- $(sk, pp) \leftarrow \mathsf{KVC.K}(crs)$ The key generation algorithm, which given as input the crs produces a signing key sk and some public issuer parameters pp.

  In the remainder of this work, we will omit crs from inputs to the algorithms below for brevity, and assume that it can be implicitly derived from either sk or pp. We will also assume that it is possible to efficiently test whether some given keypair (sk, pp) is correctly generated, i.e.,$(sk, pp) \in [\mathsf{KVC.K}(crs)]$. This is without loss of generality: it is always possible to consider the secret key sk as the random coins used in the key generation algorithm as a syntactical change.

- $\sigma \leftarrow (\mathsf{KVC.I.Usr}(pp, \vec{m}, \phi) \rightleftharpoons \mathsf{KVC.I.Srv}(sk, \phi))$ The issuance algorithm allows a user to obtain a credential σ for a set of attributes $\vec{m}$, kept hidden from the issuer, but satisfying a predicate $\phi \in \Phi$.

  All future issuance algorithm will be non-interactive, i.e. the server will receive a message from the user and respond with a (blinded) credential, that will be further processed by the user. Therefore, to simplify the description, we will split the protocol into 3 non-interactive algorithms:

  - $(st_u, \mu) \leftarrow \mathsf{KVC.I.Usr}_1(pp, \vec{m}, \phi)$, producing the first message μ and the user state st_u.
  - $\sigma' \leftarrow \mathsf{KVC.I.Srv}(sk, \phi, \mu)$, producing the server's blinded credential σ'.
    If the server does not accept the user's message, the server will return $\sigma' = \perp$.
  - $\sigma \leftarrow \mathsf{I.Usr}_2(st_u, \sigma')$, producing the credential σ given as input the state st_u and the server message σ'.

  The canonical definition of keyed-verification anonymous credentials, instead of arbitrary predicates, admit a set J of hidden attributes. This can be seen as a special instance of this definition: for any set $J \in 2^{[n]}$ the corresponding predicate of partial disclosure is $\phi^{\vec{a}}_J(\vec{m}) \coloneqq (\forall j \in [n] \setminus J : a_j = m_j)$.

- $0/1 \leftarrow (\mathsf{KVC.P.Srv}(sk, \phi) \rightleftharpoons \mathsf{KVC.P.Usr}(pp, \vec{m}, \sigma, \phi))$ The present (or *show*) algorithm allows a user to prove possession of a credential σ over some (partially disclosed) attributes satisfying a predicate $\phi \in \Phi$, without revealing any information other than what can be explicitly inferred from ϕ. Optimizations related to the above two special cases are straightforward to obtain for the schemes we will present. We will mention how to make proofs more efficient for the case of partial disclosure, so that the presentation proof will be independent of the total number of attributes.

  All presentation algorithms in this document are non-interactive, i.e. the user will send a single message to the server and the server will output a single bit. Consequently, we will split the protocol into two non-interactive algorithms:

  - $\rho \leftarrow \mathsf{KVC.P.Usr}(pp, \vec{m}, \sigma, \phi)$, producing the presentation message ρ.
  - $0/1 \leftarrow \mathsf{KVC.P.Srv}(sk, \phi, \rho)$, verifying the presentation message ρ for ϕ with sk.

A keyed-verification credential system KVC satisfies *correctness*, *anonymity*, and *unforgeability*.

Similarly to previous works [CMZ14, CR19], we define two extra algorithms to simplify formalism:

- $\sigma \leftarrow \mathsf{KVC.M}(sk, \vec{m})$ generates a credential σ for the attributes $(m_1, \ldots, m_n)$. This is syntactic sugar for running $(\mu, st) \leftarrow \mathsf{KVC.I.Usr}_1(pp, \vec{m}, \phi_\emptyset)$, $\sigma' \leftarrow \mathsf{KVC.I.Srv}(sk, \phi_\emptyset, \vec{m}, \mu)$, and returning $\sigma \leftarrow \mathsf{KVC.I.Usr}_2(st, \sigma')$.
- $0/1 \leftarrow \mathsf{KVC.V}(sk, \vec{m}, \sigma)$ verifies that the credential σ for the attributes $\vec{m}$ using sk. This is syntactic sugar for running $\rho \leftarrow \mathsf{KVC.P.Usr}(pp, \vec{m}, \sigma, \phi_\emptyset)$ and returning $\mathsf{KVC.P.Srv}(sk, \phi_\emptyset, \rho)$.

The notation clash with the MAC and verification algorithms of a MAC scheme is on purpose, as we will focus on credentials derived from algebraic MACs.

4.2 Correctness

Correctness informally states that messages satisfying the issuance predicate should lead to correct credential issuance, and credential presentation should succeed for any valid statement.

Definition 4 (Correctness)**.** A keyed-verification credential system KVC for a family of $n \leq \mathsf{poly}(\lambda)$ attributes and nontrivial predicates $\Phi \supseteq \{\phi_J^{\vec{a}} \colon J \in 2^{[n]}, \vec{a} \in \mathbb{M}^n\}$ and a non-empty message family $\mathcal{M}$ is **correct** if for any p.p.t. adversary A, for any $crs \in [\mathsf{KVC.S}(1^\lambda, n)]$ and $(sk, pp) \in [\mathsf{KVC.K}(crs)]$ and for any $\phi, \phi' \in \Phi$, $\vec{m} \in \mathbb{M}_{crs}^n$ such that $\phi(\vec{m}) = \phi'(\vec{m}) = 1$:

$$\Pr\left[b = 1 : \begin{array}{l} \sigma \leftarrow (\mathsf{KVC.I.Usr}(pp, \vec{m}, \phi) \rightleftharpoons \mathsf{KVC.I.Srv}(sk, \phi)) \\ b \leftarrow (\mathsf{KVC.P.Srv}(sk, \phi') \rightleftharpoons \mathsf{KVC.P.Usr}(pp, \vec{m}, \sigma, \phi')) \end{array}\right]$$

is overwhelming in λ.

4.3 Anonymity

Anonymity is captured with an indistinguishability game where the adversary, given as input crs, is asked to distinguish between the interaction with an honest user and a simulator that does not know any of the attributes (for issuance and presentation). Users for which the issuance/presentation predicates hold are then indistinguishable. More formally, the simulator is a pair of procedures $\mathsf{Sim} = (\mathsf{Sim.I}, \mathsf{Sim.P})$ where $\mathsf{Sim.I}(pp, \phi)$ is an interactive procedure whose distribution is meant to be indistinguishable from $\mathsf{KVC.I.Usr}$, returning at the end some simulator state st_{Sim}, and $\mathsf{Sim.P}(st_{\mathsf{Sim}}, \phi)$ returns a transcript that can be used to distinguish the "real" from the "simulated" interaction. During its execution it has access to a presentation oracle, either consisting of the user procedure or of $\mathsf{Sim.P}$.

Definition 5. A keyed-verification credential system KVC for a family of $n \leq \mathsf{poly}(\lambda)$ attributes, predicate family Φ, and a non-empty message family $\mathbb{M}$ is **anonymous** if there exists a simulator $\mathsf{Sim} = (\mathsf{Sim.I}, \mathsf{Sim.P})$ such that, for all $crs \in [\mathsf{KVC.S}(1^\lambda, n)]$ and $(sk, pp) \in [\mathsf{KVC.K}(crs)]$, $\vec{m} \in \mathbb{M}_{crs}^n$, $\phi \in \Phi$, adversaries A, D, the advantage $\mathsf{Adv}^{\mathrm{anon}}_{\mathsf{KVC},\mathsf{A},\mathsf{D}}(\lambda, n)$ defined as:

$$\left|\Pr\left[b' = 1 : \begin{array}{l} (\sigma; st_{\mathsf{A}}) \leftarrow (\mathsf{KVC.I.Usr}(pp, \vec{m}, \phi) \rightleftharpoons \mathsf{A}(sk, pp, \phi, \vec{m})) \\ b' \leftarrow \mathsf{D}^{\textsc{Present}_0}(st_{\mathsf{A}}) \end{array}\right] - \Pr\left[b' = 1 : \begin{array}{l} (st_{\mathsf{Sim}}; st_{\mathsf{A}}) \leftarrow (\mathsf{Sim.I}(pp, \phi) \rightleftharpoons \mathsf{A}(sk, pp, \phi, \vec{m})) \\ b' \leftarrow \mathsf{D}^{\textsc{Present}_1}(st_{\mathsf{A}}) \end{array}\right]\right|$$

is negligible in λ, where the $\textsc{Present}_b(\phi)$ oracle checks if $\phi(\vec{m})$ holds for $\vec{m}$, and if so returns $\mathsf{KVC.P.Usr}(pp, \vec{m}, \sigma, \phi)$ otputus (if $b = 0$) or $\mathsf{Sim.P}(st_{\mathsf{Sim}}, \phi)$ outputs (if $b = 1$)

A KVC has **statistical anonymity** if $\mathsf{Adv}^{\mathrm{anon}}_{\mathsf{KVC},\mathsf{A},\mathsf{D}}(\lambda, n)$ is negligible for unbounded adversaries A, D. A KVC has **everlasting forward anonymity** if $\mathsf{Adv}^{\mathrm{anon}}_{\mathsf{KVC},\mathsf{A},\mathsf{D}}(\lambda)$ is negligible when D is unbounded.

Comparison with original KVACs [CMZ14]. The canonical definition of anonymity for keyed-verification credentials is concerned solely with anonymity of different presentation sessions. Formally, it asks that for any p.p.t. adversary A, for any $crs \in [\mathsf{KVC.S}(1^\lambda, n)]$ and $(sk, pp) \in [\mathsf{KVC.K}(crs)]$ and for any $\phi \in \Phi$, $\vec{m} \in \mathbb{M}_{crs}^n$ such that $\phi(\vec{m}) = 1$,

$$\left|\Pr\left[\mathsf{A}(sk, pp, \phi, \rho) = 1 : \begin{array}{l} \sigma \leftarrow \mathsf{KVC.M}(sk, \vec{m}) \\ \rho \leftarrow \mathsf{KVC.P.Usr}(pp, \vec{m}, \sigma, \phi) \end{array}\right] - \Pr\left[\mathsf{A}(sk, pp, \phi, \rho) = 1 : \rho \leftarrow \mathsf{Sim.P}(sk, \phi)\right]\right|$$

is negligible in λ. In some credentials such as Chase–Meiklejohn–Zaverucha, where anonymity is only computational, this difference is tangible. (In [CMZ14], other notions such as key-parameter consistency and blind issuance are concerned with user anonymity.)

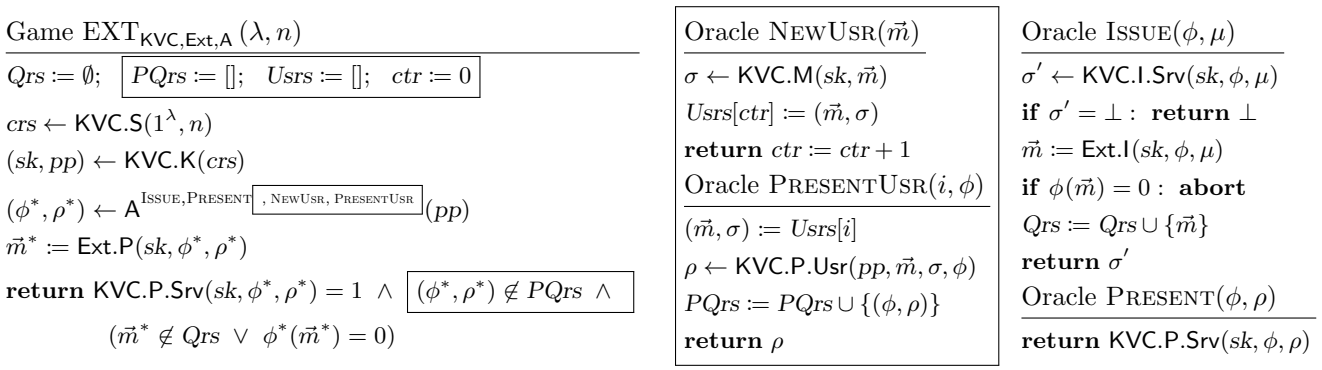


Figure 4: Extraction game for a keyed-verification credential system KVC with extractors Ext.I and Ext.P, and adversary A. The variables "ϕ" and "ϕ^*" denote the predicate to be shown on $\vec{m}$ (cf. definition 2) whereas "μ" denotes the issuance message, and "ρ" the presentation message. Boxed, the experiment in the multi-user setting with man-in-the-middle attacks.

4.4 Extractability

In keyed-verification credential systems, unforgeability states that an adversary should not be able to present a credential for a predicate ϕ that does not hold over any of the previously-issued credentials. The notion we propose is stronger: we require an extractor than can recover the attributes from issuance/presentation messages, and consider man-in-the-middle adversaries. The adversary wins if at the end of the game it presents a valid credential from which the extracted message was not previously issued to the adversary or an honest user.

Definition 6. A keyed-verification credential system KVC for a family of $n \leq \mathsf{poly}(\lambda)$ attributes, predicate family Φ, and a non-empty message family $\mathbb{M}$ is **extractable** if there exists $\mathsf{Ext} = (\mathsf{Ext.I}, \mathsf{Ext.P})$ such that, for any p.p.t. adversary A:

$$\mathsf{Adv}^{\mathrm{ext}}_{\mathsf{KVC},\mathsf{Ext},\mathsf{A}}(\lambda) \coloneqq \Pr\left[\mathrm{EXT}_{\mathsf{KVC},\mathsf{Ext},\mathsf{A}}(\lambda, n) = 1\right]$$

is negligible in λ, where $\mathrm{EXT}_{\mathsf{KVC},\mathsf{Ext},\mathsf{A}}(\lambda, n)$ is defined in figure 4.

In the game, the adversary receives as input some public parameters (from which it is possible also to infer the common reference string *crs*) and can interact with issuance and presentation oracles. The oracles NewUsr and PresentUsr deal with issuance and presentation of credentials of *other* users, and are meant to model man-in-the-middle attackers that might re-use presentation messages for other presentation predicates. The oracles Issue and Present instead deal with issuance and presentation of a credential: the Issue oracle returns a server message σ' representing the blinded credential that the adversary may unblind. The Present oracle returns the result of the presentation protocol. The adversary is then asked to output a pair $(\vec{m}^*, \phi^*)$ that was not queried to the oracles, and that is valid for the predicate ϕ^*.

Comparison with original KVACs [CMZ14]. The original definition of unforgeability for keyed-verification anonymous credentials (illustrated in figure 3) is implied by extractability. Any adversary A for the game $\mathrm{UNF}_{\mathsf{KVC},\mathsf{A}}(\lambda, n)$ can in fact be used to construct an adversary B for the game $\mathrm{EXT}_{\mathsf{KVC},\mathsf{Ext},\mathsf{B}}(\lambda, n)$. B will receive as input pp and internally run $\mathsf{A}(pp)$. For every query of the form $\textsc{Sign}(\vec{m})$ made by A, then B will run $\mu \leftarrow \mathsf{KVC.I.Usr}_1(pp, \vec{m}, \phi_\emptyset)$ and query the issuance oracle. Since the game does not abort during issuance (except with negligible probability), the extractor Ext.P will recover $\vec{m}$ with non-negligible probability (it is the only element of $\mathbb{M}^n$ satisfying $\phi^{\vec{m}}_\emptyset$). For every presentation oracle query from A, the adversary B forwards it to the respective oracle Present of the extractability game. B returns the same output as A, a pair (ϕ^*, ρ^*). By the winning condition of the unforgeability game we have that $\phi^*(\vec{m}) = 0$ for every $\vec{m}$ previously queried, and yet (by winning condition), $\phi^*(\vec{m}^*) = 1$. This means that $\vec{m}^*$ is different from all previously-queried messages, and therefore $\mathsf{Adv}^{\mathrm{unf}}_{\mathsf{KVC}}(\lambda) \leq \mathsf{Adv}^{\mathrm{ext}}_{\mathsf{KVC}}(\lambda)$.

Comparison with PABSs [CKL+16]. Extraction is similar to the unforgeability experiment for *privacy-enhancing attribute-based signatures* (PABSs), with some syntactical differences, mostly due to the modularity of PABSs, where each attribute is committed with an *opening extractable* commitment (a commitment equipped with a zero-knowledge proof of knowledge for the opening relation). Their notion of unforgeability schemes demands that there exist extractors $(\mathsf{E}^c, \mathsf{E}^s)$ recovering the attributes from respectively issuance and presentation. For any adversary that outputs a valid credential presentation message (composed of a signature presentation token *spt* satisfying SignTokenVf for public attributes $\vec{a}$, committed attributes $\vec{c}$, and a message M), one of the following is true: (i) the extractor E^s failed to recover a valid opening for one of the commitments returned from the adversary (ComOpenVf); (ii) the extractor E^s failed to recover a valid signature from the adversary's output (SigVf); (iii) the

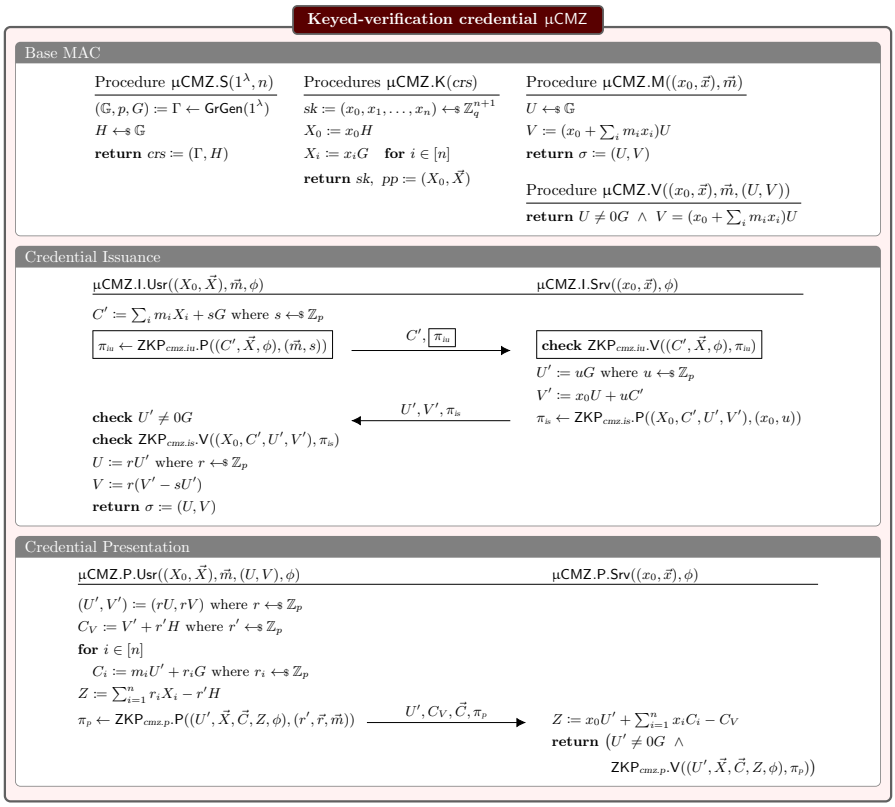


Figure 5: The keyed-verification credential system μCMZ. Differs from [CMZ14] in key generation and issuance algorithms to achieve statistical anonymity. Boxed, the part that may be removed for anonymous tokens (one-more unforgeability). The variable ϕ denotes the arbitrary predicates that must be enforced on the attributes during issuance or presentation. The relations to be proven are defined in equations (4) to (6).

message and credentials (as extracted from E^c) were not previously queried. Throughout the game, the issuance oracle checks that the commitment verification proof is satisfied, which by opening extraction, implies that the issuance message is correctly extracted similarly to the abort condition in Issue. Items (i) and (ii) ask that a valid message is extracted every time that the credential verification equation is successful (similarly to our request on Ext.P); item (iii) is similar to our request of the forgery not being present in *Qrs*, the set of credentials extracted from Ext.I.

Strategy employed in our analysis. When proving our keyed-verification credential systems, we do not consider the multi-user setting. All credentials examined internally rely on a proof system ZKP that is zero-knowledge and knowledge-sound, and it is sufficient to replace knowledge-soundness with strong simulation extractability to prove the definition of extractability as presented in figure 4. In fact, the output of PresentUsr is always a prover message ρ composed of commitments and proofs (the commitments, included in ρ, are part of the statement of the proofs). Therefore, the additional winning condition guarantees extraction in the strong simulation extractability game.

5 Chase–Meiklejohn–Zaverucha credentials

Figure 5 formally illustrates our credential μCMZ. It can be seen as a variant of $\mathrm{MAC}_{\mathrm{GGM}}$, secure in the generic group model, as introduced by Chase, Meiklejohn, and Zaverucha [CMZ14].

5.1 Protocol description

Message authentication code. The key generation procedure samples random $(x_0, \vec{x}) \leftarrow\$ \mathbb{Z}_p^{n+1}$, and sets the public parameters $(X_0 = x_0 H, X_1 = x_1 G, \ldots, X_n = x_n G)$. The signing algorithm computes $V = (x_0 + \sum_i m_i x_i)U$. The special property of this MAC is that they are structure-preserving, i.e. given a MAC (U, V) for $\vec{m}$, then $(U', V') := (rU, rV)$ for $r \neq 0$ still satisfies

the verification equation $V' = (x_0 + \sum_i m_i x_i)U$. The verification algorithm $\mu\mathsf{CMZ.V}((x_0, \vec{x}), \vec{m}, (U, V))$ checks $U \neq 0G$ and $V = (x_0 + \sum_i x_i m_i)U$. We prove the following theorem:

Theorem 7. *In the algebraic group model, $\mu\mathsf{CMZ}$ is an algebraic MAC for $n = \mathsf{poly}(\lambda)$ attributes over $\mathbb{Z}_p$ with advantage:*

$$\mathsf{Adv}^{\mathrm{ufcmva}}_{\mu\mathsf{CMZ}}(\lambda, n) \leq \mathsf{Adv}^{\text{3-dl}}_{\mathsf{GrGen}}(\lambda) + \mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda) + \frac{3}{p} \ ,$$

Previous analyses in the generic group model, for schemes slightly different to this one, can be found in Chase et al. [CMZ14, Thm. 2] and the analysis of Pointcheval and Sanders [PS16, Assumption 2].

Blind issuance. The user commits to the attributes as $C = \sum_i m_i X_i + sH$, where s is sampled uniformly at random in $\mathbb{Z}_p$. Then, it proves that $\vec{m}$ satisfies the issuance predicate ϕ, i.e., that $((C, \vec{X}, \phi), (\vec{m}, s))$ is in the relation:

$$\mathsf{R}_{cmz.iu} \coloneqq \left\{ ((C', \vec{X}, \phi), (\vec{m}, s)) : \ C' = \textstyle\sum_i m_i X_i + sG \ \wedge \ \phi(\vec{m}) = 1 \right\} \tag{4}$$

and sends a proof of knowledge π_{iu} for $(\vec{m}, s)$. Public messages don't need to be committed (nor proven in zk) and the server can add them itself using the homomorphic properties of C'. The server computes a MAC for the message $\vec{m}$ and proves its correctness via a proof π_{is} for the relation:

$$\mathsf{R}_{cmz.is} \coloneqq \left\{ ((X_0, C', U', V'), (x_0, u)) : \begin{array}{l} U' = uG \ \wedge \ X_0 = x_0 H \ \wedge \\ V' = uC' + x_0 U' \end{array} \right\} \tag{5}$$

Credential presentation. To present a credential, the user first re-randomizes the MAC $\sigma = (U, V)$ into $\sigma' = (U' = rU, V' = rV)$ with $r \leftarrow\!\!\$\ \mathbb{Z}_p$, produces commitments C_i to each message m_i, and finally prove that the committed messages satisfy a predicate ϕ.[22] The user will prove the relation:

$$\mathsf{R}_{cmz.p} \coloneqq \left\{ \begin{array}{c} ((U', X_1, \ldots, X_n, C_1, \ldots, C_n, Z, \phi), (r', \vec{r}, \vec{m})) : \\ (\forall i \in [n] \ \ C_i = m_i U' + r_i G) \ \wedge \ Z = \sum_{i=1}^{n} r_i X_i - r' H \ \wedge \ \phi(\vec{m}) = 1 \end{array} \right\} . \tag{6}$$

The case of partial disclosure of attributes, where a subset of attributes $J \subseteq [n]$ is meant to be hidden, can be optimized in a straightforward way: the user will commit only to the attributes in J and the server will compute $Z = x_0 U' + \sum_{i \in J} x_i C_i + \sum_{i \in [n] \setminus J} m_i U' - C_V$, using the hidden attributes as input to the proof. The case of full disclosure of messages is also straightforward: it consists in sending $(U', V' + r'G)$ and proving the relation fixing $r_i = 0$ for all $i \in [n]$.

If the predicate ϕ only concerns a subset $J \in [n]$ of attributes then the presentation proof message can be optimized via standard batching techniques. The user will still send to the server commitments for all hidden attributes, but instead of proving knowledge of an opening for each C_i, the user computes $\hat{C} = \sum_{i \in [n] \setminus J} \mu^i C_i$ where $\mu \coloneqq \mathsf{H}(\vec{C})$ and proves knowledge instead of $\hat{m} = \sum_{i \in [n] \setminus J} \mu^i m_i$ and $\hat{r} = \sum_{i \in [n] \setminus J} \mu^i r_i$.

Let $\mathsf{R}_{cmz} \coloneqq \mathsf{R}_{cmz.iu} \cup \mathsf{R}_{cmz.is} \cup \mathsf{R}_{cmz.p}$. We prove the following:

Theorem 8. *If ZKP is a proof system for the relation $\mathsf{R} \supseteq \mathsf{R}_{cmz}$, then $\mu\mathsf{CMZ}$ is a keyed-verification extractable credential for $n = \mathsf{poly}(\lambda)$ attributes with anonymity advantage:*

$$\mathsf{Adv}^{\mathrm{anon}}_{\mu\mathsf{CMZ}}(\lambda, n) \leq \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{cmz.iu}}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{cmz.p}}(\lambda) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{cmz.is}}(\lambda) \ ,$$

and extractability advantage:

$$\mathsf{Adv}^{\mathrm{ext}}_{\mu\mathsf{CMZ}}(\lambda, n) \leq \mathsf{Adv}^{\mathrm{ufcmva}}_{\mu\mathsf{CMZ}}(\lambda, n) + \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{cmz.is}}(\lambda) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{cmz.iu}}(\lambda) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{cmz.p}}(\lambda) \ .$$

In particular, $\mu\mathsf{CMZ}[\mathsf{ZKP} = \Sigma]$ has statistical anonymity if we conjecture that Σ is statistically knowledge sound.

Let $\mu\mathsf{CMZ}_{AT}$ denote the variant of $\mu\mathsf{CMZ}$ in figure 5 where the issuance algorithm does not return the proof π_{iu} (that is, the boxed areas are removed). From the above, and theorem 14, we also have:

Theorem 9. *If ZKP is a proof system for the relation $\mathsf{R} \supseteq \mathsf{R}_{cmz.p} \cup \mathsf{R}_{cmz.is}$, then $\mu\mathsf{CMZ}_{AT}$ is an anonymous token for $n = \mathsf{poly}(\lambda)$ attributes with anonymity advantage:*

$$\mathsf{Adv}^{\mathrm{anon}}_{\mu\mathsf{CMZ}_{AT}}(\lambda, n) \leq \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{cmz.iu}}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{cmz.p}}(\lambda) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{cmz.is}}(\lambda) \ ,$$

and (in the algebraic group model) one-more unforgeability advantage:

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mu\mathsf{CMZ}_{AT}}(\lambda, n) \leq \mathsf{Adv}^{\mathrm{2dl}}_{\mathsf{GrGen}}(\lambda) + q\mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda) + \frac{q+5}{p} \ ,$$

where q is the number of queries to the issuance oracle.

[22] Despite not explicit in the original work, the check $U' \neq 0G$ is required as otherwise a user having a credential for $\vec{m} = 0$ will verify for $\vec{m} = \vec{0}, r' = 0, r_i = 0$.

5.2 Analysis

5.2.1 Algebraic MAC

We prove that μCMZ is an algebraic MAC first for the case $n = 1$ (lemma 10) and then reduce the case for many attributes to the case $n = 1$ (theorem 11). Looking ahead, the keyed-verification scheme will reduce to unforgeability of the base MAC, but will need access to the secret key in order to check that $Z + C_V = x_0 U' + \sum_i x_i C_i$ where $U', C_V, \vec{C}$ are adversarially chosen and $(x_0, \vec{x})$ are the MAC secrets. To solve this technicality, we study a stronger experiment where the adversary is also given an oracle $\textsc{Help}(A_0, A_1, Z)$ that returns 1 if $Z = x_0 A_0 + x_1 A_1$ and 0 otherwise.

Lemma 10. *In the algebraic group model,* μCMZ *is an algebraic MAC for $n = 1$ attributes in $\mathbb{Z}_p$ with advantage:*

$$\mathsf{Adv}^{\text{ufcmva}}_{\mu\mathsf{CMZ}}(\lambda, 1) \leq \mathsf{Adv}^{\text{3-dl}}_{\mathsf{GrGen}}(\lambda) + \frac{1}{p} \ .$$

Proof. Statistical correctness follows by inspection; we prove unforgeability. Let A be an adversary that receives as input some public parameters $pp = (X_0, X_1)$ has access to a sign oracle $\textsc{Sign}$, a verification oracle $\textsc{Verify}$, and a helper oracle $\textsc{Help}$. It outputs a forgery $(m^*, (U^*, V^*))$. The j-th oracle query to $\textsc{Sign}(m_j)$ takes as input $m_j \in \mathbb{Z}_p$ and outputs a MAC $\sigma_j = (U_j, V_j)$. Let q be the number of signing queries made during the execution of the adversary. The output of an algebraic adversary is accompanied by an algebraic representation $\vec{\alpha}, \vec{\beta}$ of the form

$$U^* = \alpha_g G + \alpha_h H + \alpha_{x_0} X_0 + \alpha_{x_1} X_1 + \sum_j^q \alpha_{j,u} U_j + \alpha_{j,v} V_j \ ,$$
$$V^* = \beta_g G + \beta_h H + \beta_{x_0} X_0 + \beta_{x_1} X_1 + \sum_j^q \beta_{j,u} U_j + \beta_{j,v} V_j \ .$$

The verification equation requires that U^* is non-zero and that the following polynomial equation in $\mathbb{Z}_p[\boldsymbol{\eta}, \boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{u_1}, \ldots, \boldsymbol{u_q}]$

$$\begin{aligned}\left(\alpha_g + \alpha_h \boldsymbol{\eta} + \alpha_{x_0} \boldsymbol{x_0}\boldsymbol{\eta} + \alpha_{x_1} \boldsymbol{x_1} + \textstyle\sum_j^q \alpha_{j,u} \boldsymbol{u_j} + \alpha_{j,v} \boldsymbol{u_j}(\boldsymbol{x_0} + m_j \boldsymbol{x_1})\right)(\boldsymbol{x_0} + m^* \boldsymbol{x_1}) = \\ \beta_g + \beta_h \boldsymbol{\eta} + \beta_{x_0} \boldsymbol{x_0}\boldsymbol{\eta} + \beta_{x_1} \boldsymbol{x_1} + \textstyle\sum_j^q \beta_{j,u} \boldsymbol{u_j} + \beta_{i,v} \boldsymbol{u_j}(\boldsymbol{x_0} + m_j \boldsymbol{x_1})\end{aligned} \tag{7}$$

holds when evaluated in $\boldsymbol{x_0} = \log_H X_0$, $\boldsymbol{x_1} = \log_G X_1$, $\boldsymbol{\eta} = \log_G H$ and $\boldsymbol{u_j} = \log_G U_j$. We first study the case that equation (7) holds over the polynomial ring, and then argue equality over the field. For the equation to hold, we must have that all monomials on the left-hand side must appear also on the right-hand side of the equation. Thus, it can be simplified as:

$$(\alpha_g + \sum_i \alpha_{i,u} \boldsymbol{u_j})\boldsymbol{x_0} + \alpha_h \boldsymbol{\eta}\boldsymbol{x_0} + m^*(\alpha_g + \sum_j \alpha_{j,u} \boldsymbol{u_j})\boldsymbol{x_1} = (\sum_j \beta_{j,v} \boldsymbol{u_j})\boldsymbol{x_0} + \beta_{x_0} \boldsymbol{\eta}\boldsymbol{x_0} + (\beta_{x_1} + \sum_j \beta_{j,v} m_j \boldsymbol{u_j})\boldsymbol{x_1}$$

and therefore: $\alpha_{j,u} = \beta_{j,v}$ for all $j \in [q]$, $\alpha_g = 0$, $\alpha_h = \beta_{x_0}$, $\beta_{x_1} = 0$ (from $\alpha_g = 0$), and $\beta_{j,v} m_j = \alpha_{j,u} m^*$ for all $j \in [q]$ which in turn means that $m^* = m_j$ for some j. This is a contradiction to the fact that m^* was not queried to the signing oracle. It must therefore be the case that, for a winning adversary, equation (7) does not hold over $\mathbb{Z}_p[\boldsymbol{\eta}, \boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{u_1}, \ldots, \boldsymbol{u_q}]$ but does hold when evaluated in the relative discrete logarithms.

We build a reduction B to 3-DL. The adversary B takes as input some group description Γ and $(X, X', X'') \in \mathbb{G}^3$. It samples random $a_h, b_h, a_{x_0}, b_{x_0}, a_{x_1}, b_{x_1} \leftarrow\!\!\$\ \mathbb{Z}_p$ and invokes A with public parameters

$$pp \coloneqq (\Gamma,\ H = a_h G + b_h X,\ X_0 = a_h a_{x_0} G + (b_h a_{x_0} + a_h b_{x_0}) X + b_h b_{x_0} X',\ X_1 = a_{x_1} G + b_{x_1} X) \ , \tag{8}$$

which is identically distributed to the one of an honest signer. During its execution, the adversary can make queries to the oracles $\textsc{Sign}, \textsc{Verify}, \textsc{Help}$. For the j-th signing query on a message m_j, the reduction B samples $a_{u,j}, b_{u,j} \leftarrow\!\!\$\ \mathbb{Z}_p$ and returns

$$\begin{aligned}&(U_j = a_{u,j} G + b_{u,j} X \ , \\ &V_j = a_{u,i}(a_{x_0} + m_i a_{x_1}) G + (a_{u,j} b_{x_0} + m_j a_{u,j} b_{x_1} + b_{u,j} a_{x_0} + m b_{u,j} a_{x_1}) X + b_{u,j}(b_{x_0} + m b_{x_1}) X') \ .\end{aligned} \tag{9}$$

A query to the verification oracle consists of a message m and a MAC (U, V), accompanied by the algebraic representation $\vec{\alpha}, \vec{\beta}$ such that:

$$U = \alpha_g G + \alpha_h H + \alpha_{x_0} X_0 + \alpha_{x_1} X_1 + \sum_j^{n_s} \alpha_{j,u} U_j + \alpha_{j,v} V_j,$$
$$V = \beta_g G + \beta_h H + \beta_{x_0} X_0 + \beta_{x_1} X_1 + \sum_j^{n_s} \beta_{j,u} U_j + \beta_{j,v} V_j$$

where $n_s < q$ is the current number of queries made to SIGN. The oracle tests the verification equation plugging equations (8) and (9) into the verification equation and checking it using (X, X', X'') as the maximum degree of the resulting polynomial is 3. If the equation is not satisfied, the oracle returns 0. Otherwise, it returns 1.

Any HELP query of the form $(A_0, A_1, Z) \in \mathbb{G}^3$ has algebraic representation:

$$\begin{aligned} A_0 &= \gamma_g^{(0)} G + \gamma_h^{(0)} H + \gamma_{x_0}^{(0)} X_0 + \gamma_{x_1}^{(0)} X_1 + \sum_j \gamma_{j,u}^{(0)} U + \gamma_{j,v}^{(0)} V \ , \\ A_1 &= \gamma_g^{(1)} G + \gamma_h^{(1)} H + \gamma_{x_0}^{(1)} X_0 + \gamma_{x_1}^{(1)} X_1 + \sum_j \gamma_{j,u}^{(1)} U + \gamma_{j,v}^{(1)} V \ , \\ Z &= \delta_g G + \delta_h H + \delta_{x_0} X_0 + \delta_{x_1} X_1 + \sum_j \delta_{j,u} U + \delta_{j,v} V \ , \end{aligned} \tag{10}$$

and, similarly to the verification oracle, B can respond to the query evaluating the associated polynomial in X via (G, X, X', X''), as the maximum degree of the resulting expression is 3. At the end, the adversary A returns a forgery $(m^*, (U^*, V^*))$ satisfying the verification equation and such that $m^* \neq m_i$ for all $i \in [q]$. From equation (7) we know that (since $m^* \neq m_i$ for all i's) the polynomial of equation (7)

$$\begin{aligned} \varphi(\boldsymbol{\eta}, \boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{u_1}, \ldots, \boldsymbol{u_q}) \coloneqq (\alpha_g + \alpha_h \boldsymbol{\eta} + \alpha_{x_0} \boldsymbol{x_0 \eta} + \alpha_{x_1} \boldsymbol{x_1} + \textstyle\sum_j^q \alpha_{j,u} \boldsymbol{u_j} + \alpha_{j,v} \boldsymbol{u_j}(\boldsymbol{x_0} + m_j \boldsymbol{x_1}))(\boldsymbol{x_0} + m^* \boldsymbol{x_1}) - \\ \Big(\beta_g + \beta_h \boldsymbol{\eta} + \beta_{x_0} \boldsymbol{x_0 \eta} + \beta_{x_1} \boldsymbol{x_1} + \textstyle\sum_j^q \beta_{j,u} \boldsymbol{u_j} + \beta_{i,v} \boldsymbol{u_j}(\boldsymbol{x_0} + m_j \boldsymbol{x_1})\Big) \end{aligned}$$

is non-zero. Since the $b_h, b_{x_0}, b_{x_1}, b_{u,j}$'s are all uniformly random and perfectly hidden by the respective $a_h, a_{x_0}, a_{x_1}, a_{u,j}$'s, the partial evaluation

$$\psi(\boldsymbol{\chi}) = \varphi(a_h + \boldsymbol{\chi} b_h, a_{x_0} + \boldsymbol{\chi} b_{x_0}, a_{x_1} + \boldsymbol{\chi} b_{x_1}, a_{u,1} + \boldsymbol{\chi} b_{u,1}, \ldots, a_{u,q} + \boldsymbol{\chi} b_{u,q})$$

is still a non-zero polynomial in $\mathbb{Z}_p[\boldsymbol{\chi}]$, of degree at most 3, with overwhelming probability. Yet, since the verification equation is satisfied, $\psi(\log_G X) = 0$. Therefore, one of the roots of ψ is in the field and is the discrete logarithm of X with respect to G. The reduction finds the root of ψ and returns the one that is the discrete logarithm of X. □

Theorem 11. *μCMZ is an algebraic MAC for $n = \mathsf{poly}(\lambda)$ attributes in $\mathbb{Z}_p$ with advantage:*

$$\mathsf{Adv}^{\mathrm{ufcmva}}_{\mu\mathsf{CMZ}}(\lambda, n) \leq \mathsf{Adv}^{\mathrm{ufcmva}}_{\mu\mathsf{CMZ}}(\lambda, 1) + \mathsf{Adv}^{\mathrm{gapdl}}_{\mathsf{GrGen}}(\lambda) + \frac{2}{p} \ .$$

Proof. Let A be an adversary against MAC unforgeability, taking as input the public parameters $(\Gamma, H, X_0, \ldots, X_n)$ and outputting some forgery $(\vec{m}^*, (U^*, V^*))$. During its execution, A has access to the oracles SIGN, VERIFY, and HELP. SIGN outputs a MAC (U_j, V_j) for a queried message $\vec{m}_j$. VERIFY outputs the verification output of μCMZ. We distinguish two events:

(i) the output $\vec{m}^*$ is a valid forgery and $\sum_{i=1}^n m_i^* X_i = \sum_{i=1}^n m_{j,i} X_i$ for some previously-queried message $\vec{m}_j$ $(j \in [q])$;

(ii) the output $\vec{m}^*$ is a valid forgery and $\sum_i m_i^* X_i \neq \sum_i m_{j,i} X_i$ for all $j \in [q]$,

where q is the number of queries done to SIGN.

We claim that (i) happens at most with negligible probability. To do so, we build an adversary B that solves gap DL for GrGen every time that the (bad) event of A returning a forgery $\vec{m}^*$ such that $\sum_i m_i^* X_i = \sum_i m_{j,i} X_i$ for some $j \in [q]$ occurs. The adversary B has access, during its execution, to an oracle $\mathrm{DDH}(A, Z)$ that returns 1 if Z is the DH of X and A. B takes as input (Γ, X) with $X \in \mathbb{G}$ and samples $a_i, b_i \leftarrow\!\!\$\ \mathbb{Z}_p$ and computes $X_i = a_i G + b_i X$ for $i \in [n]$ and $X_0 = zH$ for some $z \leftarrow\!\!\$\ \mathbb{Z}_p$. Invokes A with $pp = (\Gamma, X_0, X_1, \ldots, X_n)$. For every signing query on message $\vec{m}_j$ it returns

$$(U \coloneqq uG, \quad V \coloneqq (z + \textstyle\sum_i a_i m_{j,i})\, U + u\,(\sum_i b_i m_i)\, X)$$

for $u \leftarrow\!\!\$\ \mathbb{Z}_p$. For every $\mathrm{VERIFY}(\vec{m}, (U, V))$ query, it checks if $\sum_i b_i m_i = 0$. If $\sum_i b_i m_i = 0$, it returns 1 if $zU + \sum_i a_i m_i U = V$ and 0 otherwise. If $\sum_i b_i m_i \neq 0$, it returns the output bit of $\mathrm{DDH}(U, (\sum_i b_i m_i)^{-1} V - zU)$. For every $\mathrm{HELP}(A_0, \vec{A}, Z)$ query, it returns the output bit of $\mathrm{DDH}(\sum_i b_i A_i, Z - zA_0 - \sum_i a_i A_i)$. At the end, if the adversary A returns $\vec{m}^*$ and a forged MAC (U^*, V^*) satisfying

$$\sum_i m_{j,i}(a_i G + b_i X) = \sum_i m_i^*(a_i G + b_i X)$$

and $\vec{m}^* \neq \vec{m}_j$, for some $j \in [q]$. Thus, $(\sum_i (m_{j,i} - m_i^*) a_i) G = (\sum_i (m_{j,i} - m_i^*) b_i) X$. The equation is non-trivial with overwhelming probability $1/p$, since the b_i's are uniformly random and perfectly hidden in X_i's by a_i. Therefore, except with negligible

probability $1/p$ the reduction B solves the above equation and returns $x = (\sum_i a_i(m_{j,i} - m^*))(\sum_i b_i(m_{j,i} - m^*))^{-1}$, the discrete logarithm of X.

For item (ii), we reduce to the security of $\mu\mathsf{CMZ}$ for $n = 1$. The reduction B gets as input the public parameters (Γ, H, X_0, X_1). For each $i \in [n]$, sample $z_i \leftarrow\!\$\ \mathbb{Z}_p$ and compute $X_i = z_i X_1$ for $i \in [2, n]$. Assuming $X_1 \neq 0G$ (which happens with probability $1/p$), all the X_i's are uniformly distributed. The reduction then internally runs the adversary A for the unforgeability game with public parameters $(X_0, X_1, \ldots, X_n)$. For every MAC query of the form $\vec{m}$, forward the query to the oracle Sign at disposal to B for the message $m_1 + \sum_i^n z_i m_i$, obtaining (U, V). Return (U, V) to A. For every verification query for a message $\vec{m}$ and a MAC (U, V), forward the query to the oracle Verify with $(m_1 + \sum_i z_i m_i, (U, V))$. Similarly, proceed for Help queries. Given an output forgery (U^*, V^*) for a message $\vec{m}^*$, we have that $\sum_i m_i^* X_i \neq \sum_i m_{j,i} X_i$ for all $j \in [q]$, B outputs the forged message $m_1 + \sum_{i=2}^n z_i m_i$ along with the MAC (U^*, V^*). Therefore, $m_1^* + \sum_{i=2}^n m_i^* z_i \neq m_{j,1} + \sum_{i=2} m_{j,i} z_i$ for all $j \in [q]$ and thus the output message was never queried to the Sign oracle; by construction of the X_i's we have that

$$(U^*, V^*) = (U, x_0 U + \sum_{i=1}^{n} m_i^* x_i U) = (U, x_0 U + (m_1^* + \sum_{i=2}^{n} z_i m_i^*) x_1 U)$$

where $x_0 = \log_H X_0$, $x_1 = \log_G X_1$, and $z_i x_1 = \log_G X_i$ for $i \in [2, n]$. Hence, the forgery is valid and B produces a forgery every time A produces a forgery.

Thus, the overall advantage is $2/p + \mathsf{Adv}^{\mathrm{ufcmva}}_{\mu\mathsf{CMZ}}(\lambda, 1) + \mathsf{Adv}^{\mathrm{gapdl}}_{\mathsf{GrGen}}(\lambda)$. □

5.2.2 Keyed-verification credential

Theorem 12. *If* ZKP *a proof system for the relation* $\mathsf{R} \supseteq \mathsf{R}_{cmz}$*, then* $\mu\mathsf{CMZ}$ *is anonymous for* $n = \mathsf{poly}(\lambda)$ *attributes with advantage:*

$$\mathsf{Adv}^{\mathrm{anon}}_{\mu\mathsf{CMZ},\mathsf{A},\mathsf{D}}(\lambda, n) \leq \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{cmz.iu},\mathsf{A}'}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{cmz.p},\mathsf{D}'}(\lambda) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{cmz.is},\mathsf{A}''}(\lambda) \ ,$$

where A, D *are adversaries in the anonymity game and* $\mathsf{A}', \mathsf{A}'', \mathsf{D}'$ *are described in the proof.*

Overall, the technical challenges here are making sure that extracting x_0 is sufficient for simulating proofs (it is the only witness of π_{is}), and keeping track of the different adversaries in the game.

Proof. We define the simulator Sim as:

- The issuance simulator $\mathsf{Sim.I}((X_0, \vec{X}), \phi)$ samples a random group element $C' \leftarrow\!\$\ \mathbb{G}$ and simulates a representation proof π_{iu} for the instance $(C, \vec{X}, \phi))$. Upon receiving the response from the server (U', V', π_{is}), the simulator checks if the proof is valid and $U' \neq 0$. If so, it invokes the extractor for the instance (X_0, U', V') and proof π_{is} recovering (x_0, u) satisfying $x_0 = \log_H X_0$ and $u = \log_G U$. Note that x_0 is the unique value associated to the public parameter's X_0. Finally, stores the state as $st_{\mathsf{Sim}} = (x_0, \vec{X})$.
- The presentation simulator $\mathsf{Sim.P}((x_0, X_1), \vec{m})$, which does not know the message. It samples random group elements $U', C_V \leftarrow\!\$\ \mathbb{G}^2$, and $(C_1 = \gamma_1 G, \ldots, C_n := \gamma_n G)$ where $\gamma_i \leftarrow\!\$\ \mathbb{Z}_p$ for $i \in [n]$, and computes π_p via $\mathsf{ZKP}_{cmz.p}.\mathsf{Sim}((U', X_0, \vec{X}, \vec{C}, Z))$ where $Z = x_0 U' + \sum_i \gamma_i X_i - C_V$. The simulator outputs $(U', C_V, \vec{C}, \pi_p)$.

Let A be a p.p.t. machine taking as input a keypair (sk, pp), a predicate $\phi \in \Phi$, and a message $\vec{m}$ such that $\phi(\vec{m}) = 1$.

Indistinguishability is shown via a hybrid argument.

H_0 This is the real interaction.

H_1 Instead of generating π_{iu} using the witness, do so via the zero-knowledge simulator Sim with input $(C', \vec{X}, \phi)$. If A's output is noticeably different after this hybrid change, then it is possible to build an adversary A' that is able to distinguish real from simulated proofs from $\mathsf{ZKP}_{cmz.iu}$. Therefore:

$$\mathsf{Adv}^{\mathsf{H}_1}_{\mathsf{A},\mathsf{D}}(\lambda, n) \geq \mathsf{Adv}^{\mathsf{H}_0}_{\mathsf{A},\mathsf{D}}(\lambda, n) - \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{cmz.iu},\mathsf{A}'}(\lambda) \ .$$

H_2 Instead of generating π_p using the attributes $\vec{m}$ and MAC σ, use the zero-knowledge simulator Sim with input $(U', X_0, \vec{X})$. Similarly to the previous hybrid, there exists an adversary D' that wins every time D's output is noticeably different. Therefore:

$$\mathsf{Adv}^{\mathsf{H}_2}_{\mathsf{A},\mathsf{D}}(\lambda, n) \geq \mathsf{Adv}^{\mathsf{H}_1}_{\mathsf{A},\mathsf{D}}(\lambda, n) - \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{cmz.iu},\mathsf{D}'}(\lambda) \ .$$

H_3 Use the extractor of π_{is} for the statement (X_0, U', V') and check its validity.

The extractor will produce a witness (x_0, u) satisfying:

$$\begin{bmatrix} H & 0 \\ 0 & G \\ U' & C' \end{bmatrix} \begin{bmatrix} x_0 \\ u \end{bmatrix} = \begin{bmatrix} X_0 \\ U' \\ V' \end{bmatrix}$$

except with probability $\mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{cmz.is},\mathsf{A}''}(\lambda)$, where A'' is the adversary in the knowledge soundness game that runs A internally and returns the proof π_{is} for which extraction fails. In other words,

$$\mathsf{Adv}^{\mathsf{H}_3}_{\mathsf{A},\mathsf{D}}(\lambda, n) \geq \mathsf{Adv}^{\mathsf{H}_2}_{\mathsf{A},\mathsf{D}}(\lambda, n) - \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{cmz.iu},\mathsf{A}''}(\lambda) \ .$$

H_4 At issuance time, instead of computing C' using the message as $C' = \sum_i m_i X_i + sG$ for some $s \leftarrow\!\$\ \mathbb{Z}_p$, send a random element $C' \leftarrow\!\$\ \mathbb{G}$ and extract from the response π_{is} the witness (u, x_0). Then, using the extracted witness, compute a fresh MAC $(U = ruG, x_0 rU + \sum_i rum_i X_i)$ for $r \leftarrow\!\$\ \mathbb{Z}_p$ and return it. The distribution in this hybrid is identical to the previous one: U is distributed identically, and V is the (unique) group element satisfying $V = (\log_H X_0)U + \sum_i m_i(\log_G U)X_i$. Hence:

$$\mathsf{Adv}^{\mathsf{H}_4}_{\mathsf{A},\mathsf{D}}(\lambda, n) = \mathsf{Adv}^{\mathsf{H}_3}_{\mathsf{A},\mathsf{D}}(\lambda, n) \ .$$

H_5 At presentation time, sample $U', C_V \leftarrow\!\$\ \mathbb{G}^2$ and $\vec{C}$ such that $C_i = \gamma_i G$ with $\gamma_i \leftarrow\!\$\ \mathbb{Z}_p$. The (simulated) proof π_p is made for the statement $(U', C_V, \vec{C}, Z)$ where Z is computed as:

$$Z = x_0 U' + \sum_i \gamma_i X_i - C_V \ .$$

The elements $\vec{C}, C_V$ are perfectly indistinguishable from the real ones by the perfect hiding property of Pedersen commitments. The element U' is also perfectly indistinguishable from the real one, since r is uniformly distributed and $U \neq 0G$. Finally, the value Z, despite being computed differently, is exactly the same in both distributions since $\gamma_i X_i = x_i C_i$ where $x_i = \log_G X_i$. Hence:

$$\mathsf{Adv}^{\mathsf{H}_5}_{\mathsf{A},\mathsf{D}}(\lambda, n) = \mathsf{Adv}^{\mathsf{H}_4}_{\mathsf{A},\mathsf{D}}(\lambda, n) \ .$$

The last hybrid is the simulated interaction; the result follows. □

Remark 2 (Knowledge soundness vs simulation extractability)**.** To extract from π_{iu} after seeing (simulated) issuance proofs π_{is}, one would formally need to make sure that the proofs π_{is} cannot be turned into proofs for $\mathsf{R}_{cmz.iu}$. This is generally achieved by proofs that are simulation extractable, but in many cases the actual requirement can be weaker. For instance, in Schnorr proofs, the vector sizes mismatch and knowledge soundness suffices.

Theorem 13. *If* ZKP *is a proof system for the relation* $\mathsf{R} \supseteq \mathsf{R}_{cmz}$*, then* $\mu\mathsf{CMZ}$ *is extractable for* $n = \mathsf{poly}(\lambda)$ *attributes with advantage:*

$$\mathsf{Adv}^{\mathrm{ext}}_{\mu\mathsf{CMZ}}(\lambda, n) \leq \mathsf{Adv}^{\mathrm{ufcmva}}_{\mu\mathsf{CMZ}}(\lambda, n) + \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{cmz.is}}(\lambda) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{cmz.iu}}(\lambda) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{cmz.p}}(\lambda) \ .$$

Roughly speaking, the extractor matches the zero-knowledge extractor and a reduction is made to unforgeability of the underlying MAC. However, the term "Z" which is computed using the MAC key $(x_0, \vec{x})$, which is not available during the reduction and can't be computed by the unforgeability adversary. To circumvent this, we are going to assume that Z is also sent at presentation time and that the oracle HELP (introduced in section 5.2.1) is able to check that Z is indeed correctly computed. In practice, this requirement can often be dropped, as for many proof systems (including Schnorr) it is possible to recover the statement from the actual proof by looking at the trace of random oracle queries.

Proof. We use the extractors of ZKP to define the extractors for the credential system.

– $\mathsf{Ext.I}(\phi, (C', \pi_{iu}))$ is the zero-knowledge extractor of $\mathsf{ZKP}_{cmz.iu}$ which takes as input the proof π_{iu} and an instance (C', X, ϕ) and $(\vec{m}, s)$ such that $\phi(\vec{m}) = 1$ and

$$\begin{bmatrix} X_1 & X_2 & \dots & X_n & G \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \\ s \end{bmatrix} = C' \ .$$

If no such message is found, the extractor returns $\perp$ and the game aborts.

– $\mathsf{Ext.P}(\phi^*, (U', C_V, \vec{C}, \pi_p, Z))$ runs the extractor of $\mathsf{ZKP}_{cmz.p}$, which takes as input the proof π_p and an instance $(U', \vec{X}, \vec{C}, Z, \phi^*)$ and returns $(\vec{m}, \vec{r}, r')$ such that $\phi(\vec{m}) = 1$ and

$$\begin{bmatrix} U' & & & G & & & & \\ & U' & & & G & & & \\ & & \ddots & & & \ddots & & \\ & & & X_1 & X_2 & \dots & X_n & -H \end{bmatrix} \begin{bmatrix} m_1 \\ \vdots \\ m_n \\ r_1 \\ \vdots \\ r_n \\ r' \end{bmatrix} = \begin{bmatrix} C_1 \\ \vdots \\ C_n \\ Z \end{bmatrix} .$$

If the extracted witness does not satisfy the above, the extractor returns $\perp$ and the game aborts. Internally, the extractor checks that Z was correctly computed via the secret key $(x_0, \vec{x})$.

Let A be an adversary for extraction that gets as input $((\Gamma, H), X_0, X_1, \dots, X_n)$ and has access to oracles $\textsc{Issue}, \textsc{Present}$. At the end the adversary wins if one of the following holds: (i) there exists a query to $\textsc{Issue}$ of the form $(\phi, (C', \pi_{iu}))$ such that $\mathsf{ZKP}_{cmz.iu}.\mathsf{V}((C', \vec{X}, \phi), \pi_{iu})$, but extraction failed (in particular this happens if $\phi(\vec{m}) = 0$); (ii) the adversary returned a pair $(\phi^*, \rho^* = (U', C_V, C_1, \dots, C_n))$ such that $\mathsf{ZKP}_{cmz.p}.\mathsf{V}((U', \vec{X}, \vec{C}, Z), \pi_p) = 1$, but extraction failed (in particular this happens if $\phi^*(\vec{m}) = 0$); (iii) the adversary returned a pair $(\phi^*, \rho^* = (U', C_V, C_1, \dots, C_n))$ and extraction recovered a valid witness $(\vec{m}^*, \vec{r}, r')$ such that $\phi^*(\vec{m}^*) = 1$ and $\vec{m}^*$ has not been previously queried.

From the first two cases it is immediate to build an adversary for knowledge soundness of the zero-knowledge proof system ZKP. Item (iii) can be reduced to unforgeability of the underlying MAC as follows. Let B be an adversary that gets as input $pp = ((\Gamma, H), X_0, \dots, X_n)$ and has access to the oracles $\textsc{Sign}, \textsc{Verify}$ of the MAC unforgeability game. B internally runs $\mathsf{A}(pp)$. For every $\textsc{Issue}(C', \pi_{iu})$ query, it recovers a valid witness $\vec{m}, s$. It queries $\textsc{Sign}(\vec{m})$ and obtains (U, V). Returns $(U, V + sU)$ and simulates the proof π_{is}. For every query $\textsc{Present}(\phi, (U', C_V, \vec{C}, \pi_p, Z))$ from the adversary, B uses the zero-knowledge extractor to recover $(\vec{m}, \vec{r}, r')$ and returns $\textsc{Verify}(\vec{m}, (U', C_V - r'H))$. At the end A returns a value $\phi^*, \rho^* = (U', C_V, \vec{C}, \pi_p)$. B runs the extractor to obtain m^* and returns $(U', C_V - r'H)$.

The signing queries are answered in the same way. For presentation messages, we have:

$$C_V - r'G = x_0 U' + \sum_i x_i C_i - Z - r'H = (x_0 + \sum_i x_i m_i) U'$$

where the first equality follows from the answer of $\textsc{Help}$ and the second follows from the representation proof π_p. Thus, the output $(U', C_V - r'G)$ is a valid MAC only if C_V is a valid presentation message that the server accepted. Validity of the returned forgery from B follows from the winning condition $\vec{m}^* \notin Qrs$. □

5.2.3 One-more unforgeability

Let $\mu\mathsf{CMZ}_{AT}$ be the $\mu\mathsf{CMZ}$ scheme of figure 5 where the boxed parts are removed.

Theorem 14. *In the algebraic group model, $\mu\mathsf{CMZ}_{AT}$ is an anonymous token for $n = \mathsf{poly}(\lambda)$ attributes in $\mathbb{Z}_p$ with advantage:*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mu\mathsf{CMZ}_{AT}}(\lambda, n) \leq 2\mathsf{Adv}^{\mathrm{2dl}}_{\mathsf{GrGen}}(\lambda) + q\mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda) + \frac{q+5}{p} ,$$

where q is the number of signing queries allowed to the adversary.

This is a generalization of the proof from Chase, Durak, and Vaudenay [CDV22]. Anonymity follows from theorem 12. We are left with one-more unforgeability. We prove first the case $n = 1$, as the case for $n = \mathsf{poly}(\lambda)$ can be shown later using techniques similar to the ones of the previous section.

Proof. Let A be an algebraic adversary that receives as input $(\gamma, H, (X_0, X_1))$. During its execution, it makes at most q queries to the signing oracle, each indexed in $j \in [q]$ and accompanied by an algebraic representation $\vec{\gamma}^{(j)}$ satisfying:

$$C_j = \gamma_g^{(j)} G + \gamma_h^{(j)} H + \gamma_{x_0}^{(j)} X_0 + \gamma_{x_1}^{(j)} X_1 + \sum_{k<j} \gamma_{k,u}^{(j)} U_k + \gamma_{v,k}^{(j)} V_k \tag{11}$$

Consider the associated polynomial in $\mathbb{Z}_p[\boldsymbol{\eta}, \boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{u_1}, \dots, \boldsymbol{u_q}]$:

$$\mathsf{c}_j(\boldsymbol{\eta}, \boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{u_1}, \dots, \boldsymbol{u_{j-1}}) = \gamma_g^{(j)} + \gamma_h^{(j)} \boldsymbol{\eta} + \gamma_{x_0}^{(j)} \boldsymbol{x_0 \eta} + \gamma_{x_1}^{(j)} \boldsymbol{x_1} + \sum_{k<j} \gamma_{k,u}^{(j)} \boldsymbol{u_k} + \gamma_{v,k}^{(j)} (\boldsymbol{x_0 u_k} + \boldsymbol{u_k} \mathsf{c}_k) \tag{12}$$

where $\boldsymbol{\eta} = \log_G H, \boldsymbol{x_0} = \log_H X_0, \boldsymbol{x_1} = \log_G X_1$ For instance, we have:

$$\begin{aligned}
\mathsf{c}_1(\boldsymbol{\eta}, \boldsymbol{x_0}, \boldsymbol{x_1}) &= \gamma_g^{(1)} + \gamma_h^{(1)}\boldsymbol{\eta} + \gamma_{x_0}^{(1)}\boldsymbol{x_0\eta} + \gamma_{x_1}^{(1)}\boldsymbol{x_1} \\
\mathsf{c}_2(\boldsymbol{\eta}, \boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{u_1}) &= \gamma_g^{(2)} + \gamma_h^{(2)}\boldsymbol{\eta} + \gamma_{x_0}^{(2)}\boldsymbol{x_0\eta} + \gamma_{x_1}^{(2)}\boldsymbol{x_1} + \gamma_{1,u}^{(2)}\boldsymbol{u_1} + \gamma_{1,v}^{(2)}(\boldsymbol{x_0u_1} + \boldsymbol{u_1}\mathsf{c}_1) \\
\mathsf{c}_3(\boldsymbol{\eta}, \boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{u_1}, \boldsymbol{u_2}) &= \gamma_g^{(3)} + \gamma_h^{(3)}\boldsymbol{\eta} + \gamma_{x_0}^{(3)}\boldsymbol{x_0\eta} + \gamma_{x_1}^{(3)}\boldsymbol{x_1} + \gamma_{1,u}^{(3)}\boldsymbol{u_1} + \gamma_{1,v}^{(3)}(\boldsymbol{x_0u_1} + \boldsymbol{u_1}\mathsf{c}_1) + \gamma_{2,u}^{(3)}\boldsymbol{u_2} + \gamma_{2,v}^{(3)}(\boldsymbol{x_0u_2} + \boldsymbol{u_2}\mathsf{c}_2)
\end{aligned}$$

and so on. Equation (12) lives in $\mathbb{Z}_p[\boldsymbol{\eta}, \boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{u_1}, \ldots, \boldsymbol{u_{j-1}}]$, no (individual) quadratic terms are present, and the total degree of at most i. At the end of the execution, A returns the MACs $(m_i^*, (U_i^*, V_i^*))_{i=1}^{q+1}$ followed by algebraic representations $\vec{\alpha}^{(i)}, \vec{\beta}^{(i)}$ such that

$$\begin{aligned}
U_i^* &= \alpha_g^{(i)}G + \alpha_h^{(i)}H + \alpha_{x_0}^{(i)}X_0 + \alpha_{x_1}^{(i)}X_1 + \sum_j^q \alpha_{j,u}^{(i)}U_j + \alpha_{j,v}^{(i)}V_j \ , \\
V_i^* &= \beta_g^{(i)}G + \beta_h^{(i)}H + \beta_{x_0}^{(i)}X_0 + \beta_{x_1}^{(i)}X_1 + \sum_j^q \beta_{j,u}^{(i)}U_j + \beta_{j,v}^{(i)}V_j \ .
\end{aligned} \tag{13}$$

From the verification equation $U^* \neq 0$ and the polynomial equality

$$\begin{aligned}
\left(\alpha_g^{(i)} + \alpha_h^{(i)}\boldsymbol{\eta} + \alpha_{x_0}^{(i)}\boldsymbol{x_0\eta} + \alpha_{x_1}^{(i)}\boldsymbol{x_1} + \textstyle\sum_{j=1}^q \alpha_{j,u}^{(i)}\boldsymbol{u_j} + \alpha_{j,v}^{(i)}(\boldsymbol{x_0u_j} + \mathsf{c}_\mathsf{j}\boldsymbol{u_j})\right)(\boldsymbol{x_0} + m_i^*\boldsymbol{x_1}) = \\
\beta_g^{(i)} + \beta_h^{(i)}\boldsymbol{\eta} + \beta_{x_0}^{(i)}\boldsymbol{x_0\eta} + \beta_{x_1}^{(i)}\boldsymbol{x_1} + \textstyle\sum_{j=1}^q \beta_{j,u}^{(i)}\boldsymbol{u_j} + \beta_{j,v}^{(i)}(\boldsymbol{x_0u_j} + \mathsf{c}_\mathsf{j}\boldsymbol{u_j})
\end{aligned} \tag{14}$$

holds for $i \in [q+1]$ when evaluated in $\boldsymbol{\eta} = \log_G H, \boldsymbol{x_0} = \log_H X_0, \boldsymbol{x_1} = \log_G X_1$ and $\boldsymbol{u_j} = \log_G U_j$. We study the equation in the polynomial ring $\mathbb{Z}_p[\boldsymbol{\eta}, \boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{u_1}, \ldots, \boldsymbol{u_q}]$ first. The individual degree is 1 and the total degree is q.

For the verification equation to hold, $\beta_g^{(i)}, \beta_h^{(i)} = 0$ (since the left-hand side does not contain constant terms or terms in $\boldsymbol{\eta}$), $\alpha_g^{(i)}, \beta_h^{(i)} = 0$ (since the right-hand side has no monomial in $\boldsymbol{x_0}$ nor $\boldsymbol{\eta x_1}$), $\alpha_{x_0}^{(i)}, \alpha_{x_1}^{(i)} = 0$ (since the equation right-hand side does not contain any quadratic term), $\alpha_{j,v}^{(i)} = 0$ (since there are no quadratic terms on the right-hand side), and $\beta_{x_0}^{(i)}, \beta_{x_1}^{(i)} = 0$ (since all monomials on the left-hand sides now have a factor $\boldsymbol{u_j}$). We therefore have:

$$\left(\sum_{j=1}^q \alpha_{j,u}^{(i)}\boldsymbol{u_j}\right)(\boldsymbol{x_0} + m_i^*\boldsymbol{x_1}) = \sum_{j=1}^q \beta_{j,u}^{(i)}\boldsymbol{u_j} + \beta_{j,v}^{(i)}(\boldsymbol{x_0u_j} + \mathsf{c}_\mathsf{j}\boldsymbol{u_j}) \tag{15}$$

We can now remark that $\mathsf{c}_j = \gamma_g^{(j)} + \gamma_{x_1}^{(j)}\boldsymbol{x_1}$, since $\boldsymbol{\eta}, \boldsymbol{\eta x_0}$, and factors in $\boldsymbol{u_j u_k}$ (for some $k < j$) are not present. Therefore:

$$\sum_{j=1}^q \alpha_{j,u}^{(i)}\boldsymbol{u_j x_0} + \alpha_{j,u}^{(i)}m_i^*\boldsymbol{u_j x_1} = \sum_{j=1}^q (\beta_{j,u}^{(i)} + \beta_{j,v}^{(i)}\gamma_g^{(i)})\boldsymbol{u_j} + \beta_{j,v}^{(i)}\boldsymbol{u_j x_0} + \beta_{j,v}^{(i)}\gamma_{x_1}^{(j)}\boldsymbol{u_j x_1}$$

which implies that $\alpha_{u,j}^{(i)} = \beta_{u,j}^{(i)}$ (when looking at the terms in $\boldsymbol{u_j x_0}$) and there exists at least one nonzero j-indexed term since $U_i^* \neq 0G$, and $m_i^*\alpha_{u,j} = \beta_{v,j}\gamma_{x_1}^{(j)}$ (looking at the terms in $\boldsymbol{u_j x_1}$) implies $m_i^* = \gamma_{x_1}^{(j)}$. However, since $i \in [q+1]$ and $j \in [q]$, two forgeries must be on the same message, which contradicts the winning condition that requires all forgeries to be on distinct messages.

We conclude that the adversary's forgeries are accompanied by algebraic representations $\vec{\alpha}^{(i)}, \vec{\beta}^{(i)}$ such that the polynomial

$$\begin{aligned}
\varphi_i(\boldsymbol{\eta}, \boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{u_1}, \ldots, \boldsymbol{u_q}) = \left(\alpha_g^{(i)} + \alpha_h^{(i)}\boldsymbol{\eta} + \alpha_{x_0}^{(i)}\boldsymbol{x_0\eta} + \alpha_{x_1}^{(i)}\boldsymbol{x_1} + \textstyle\sum_{j=1}^q \alpha_{j,u}^{(i)}\boldsymbol{u_j} + \alpha_{j,v}^{(i)}(\boldsymbol{x_0u_j} + \mathsf{c}_\mathsf{j}\boldsymbol{u_j})\right)(\boldsymbol{x_0} + m_i^*\boldsymbol{x_1}) - \\
\left(\beta_g^{(i)} + \beta_h^{(i)}\boldsymbol{\eta} + \beta_{x_0}^{(i)}\boldsymbol{x_0\eta} + \beta_{x_1}^{(i)}\boldsymbol{x_1} + \textstyle\sum_{j=1}^q \beta_{j,u}^{(i)}\boldsymbol{u_j} + \beta_{j,v}^{(i)}(\boldsymbol{x_0u_j} + \mathsf{c}_\mathsf{j}\boldsymbol{u_j})\right) \neq 0
\end{aligned} \tag{16}$$

And yet its evaluation in $\log_G H, \log_H X_0, \log_G X_1, \log_G U_1, \ldots, \log_G U_q$ is zero. Note that φ is a multivariate polynomial of maximum total degree $q+1$ and individual degree 1.

We first prove that, if φ possesses a monomial in $\boldsymbol{u_j}$ for $j \in [q]$, then a solution for DL can be found. Let B be an adversary for DL that receives as input a group description Γ and a DL challenge $X \in \mathbb{G}$. The adversary samples a random $a, b \leftarrow\!\!\$\ \mathbb{Z}_p$ and an index $\iota \in [q+3]$. The index will determine the coefficient where we embed the DL challenge.

If $0 < \iota \leq q$, B internally runs A with public parameters generated as in $\mu\mathsf{CMZ}$: sample $\eta, x_0, x_1 \leftarrow\!\!\$\ \mathbb{Z}_p$ and invoke $\mathsf{A}(\Gamma, H \coloneqq \eta G, X_0 = x_0 H, X_1 = x_1 G)$. Queries to $\textsc{Sign}(C_j, \vec{\gamma}^{(j)})$ ($\vec{\gamma}^{(i)}$ being the algebraic representation of C_j) are responded as follows:

– For all $j < \iota$, each query is responded following the issuance procedure: sample $u_j \leftarrow\!\$\ \mathbb{Z}_p$ and return $(U_j = u_j G, V_j = x_0 U_j + u_j C_j)$. Note that each signing or verification query up to this point contains an algebraic representation in $H, X_0, X_1, U_1, \ldots, U_{\iota-1}$ and that therefore the discrete logarithm of every group element sent by the adversary is known. We denote $\hat{\gamma}^{(j)} = \log_G C_j$ (for $j < \iota$) the discrete logarithm of the j-th signed element queried. $\hat{\gamma}^{(j)}$ is computed with the inner product $\langle \vec{\gamma}, (1, \eta, x_0, x_1, u_1, \ldots, u_{j-1}) \rangle$.

– For the ι-th query, return $(U_\iota = aG + bX, V_\iota = x_0 aG + x_0 bX + aC + b\hat{\gamma}^{(\iota)} X)$

– All queries $\iota < j \leq q$ are responded as in the issuance procedure. Sample $u_j \leftarrow\!\$\ \mathbb{Z}_p$ and return $(U_j = u_j G, V_j = x_0 U_j + u_j C_j)$.

Each $\textsc{Verify}(m_i, (U_i, V_i))$ query is responded with 1 if $U_i \neq 0$ and $V_i = (x_0 + m_i x_1) U_i$, 0 otherwise. At the end of the execution, A returns forgeries $(m_i^*, (U_i^*, V_i^*))_{i=1}^{q+1}$ and algebraic representations $\vec{\alpha}^{(i)}, \vec{\beta}^{(i)}$. If $\exists i \in [q+1]$ such that

$$\phi_i^{(u,\iota)}(\boldsymbol{x}) = \varphi_i(\eta, x_0, x_1, u_1, \ldots, a + b\boldsymbol{x}, u_{\iota+1}, \ldots, u_q) \neq 0 \ ,$$

then it is possible to solve the linear equation and find the discrete logarithm of X. Note that, if φ of equation (16) has a monomial in $\boldsymbol{u_\iota}$ with a nonzero coefficient, then the polynomial $\phi(\boldsymbol{x})$ is nonzero with high probability, since b is uniformly distributed and has a monomial in $\boldsymbol{u_j}$ for $j \in [q]$.

We now prove that if φ has a monomial in $\boldsymbol{\eta}$, then a solution for DL can be found. Similarly to before, let $X \in \mathbb{G}$ be a DL challenge. The adversary B samples $a, b \leftarrow\!\$\ \mathbb{Z}_p$ and invokes $\mathsf{A}(\Gamma, H \coloneqq aG + bX, X_0 = x_0 aG + x_0 bX, X_1 = x_1 G)$. Signing and verification queries are responded as prescribed in the protocol description. At the end, if $\exists i \in [q+1]$ such that

$$\phi_i^{(h)}(\boldsymbol{x}) = \varphi_i(a + b\boldsymbol{x}, x_0, x_1, u_1, \ldots, \ldots, u_q) \neq 0 \ ,$$

it is possible to solve the linear equation and find the discrete logarithm of X with high probability.

We prove that if φ has a monomial in $\boldsymbol{x_0}$, then a solution for 2-DL can be found. (The case of $\boldsymbol{x_1}$ is analogous.) Let $\mathsf{B}(\Gamma, X, X')$ be an adversary for 2-DL. The adversary samples $\eta, a, b, x_1 \leftarrow\!\$\ \mathbb{Z}_p$ and internally runs the one-more unforgeability adversary as $\mathsf{A}(\Gamma, H \coloneqq \eta G, X_0 \coloneqq aH + b\eta X, X_1 \coloneqq x_1 G)$. During its execution, the adversary may query the signing oracle with query $\textsc{Sign}(C_j)$ which is responded as $(U_j \coloneqq u_j G, aU_j + bu_j X + u_j C_j)$. The adversary may also query the verification oracle with query $\textsc{Verify}(m_i, (U_i, V_i))$ which is responded with the help of the algebraic representation and X, X': if $U_i \neq 0$ then its algebraic representation has degree one in $\boldsymbol{x_0}$ at most, and therefore the verification equation can be tested with the help of X' for the quadratic term. At the end of the execution, the adversary returns forgeries $(m_i^*, (U_i^*, V_i^*))_{i=1}^{q+1}$ and, if $\phi_i^{(x_0)}(\boldsymbol{x})$ similar to the above is nonzero, then a solution for DL can be found with high probability. □

Theorem 15. *In the algebraic group model, if DL holds for* GrGen*, then* μCMZ *is one-more unforgeable for* $n = \mathsf{poly}(\lambda)$ *attributes.*

The strategy is to distinguish the two winning events:

(i) The adversary returns forgeries $(m_k^*, (U_k^*, V_k^*))_{k=1}^{q+1}$ such that $\forall k_1 \neq k_2 : \sum_i m_{i,k_1}^* X_i \neq \sum_i m_{i,k_2}^* X_i$.

(ii) The adversary returns forgeries $(m_k^*, (U_k^*, V_k^*))_{k=1}^{q+1}$ such that $\exists k_1 \neq k_2 : \sum_i m_{i,k_1}^* X_i = \sum_i m_{i,k_2}^* X_i$.

The first case reduces to theorem 14, while the second case can be reduced to DL using an argument similar to theorem 11.

6 Boneh–Boyen–Shacham credentials

Anonymous credentials from Boneh–Boyen–Shacham signatures were first introduced as BBS+ withing the framework of k-times anonymous authentication primitive [ASM06], and as algebraic MACs by Barki, Brunet, et al. [BBDT17]. We propose a variant in figure 6 and study it in the algebraic group model.

6.1 Protocol description

Message authentication code. The algebraic MAC for BBS is identical to standard μBBS signatures. A MAC (A, e) for a message $\vec{m}$ is

$$(A = (x + e)^{-1}(G_0 + \textstyle\sum_{i=1}^n m_i G_i), e \leftarrow\!\$\ \mathbb{Z}_p) \ .$$

Verification consists in checking $(x + e)A = G_0 s + \sum_{i=1}^n m_i G_i$. This is identical to the MAC presented by Barki, Brunet, et al. [BBDT17] setting $s = 0$. For the purpose of the proof, it is actually not needed that e is randomly chosen and the sole requirement for is that they are all different. This property can be formalized with a stateful procedure EGen that takes as input some internal state, the input message $\vec{m}$, and outputs an e-value for $\vec{m}$. Some examples are:

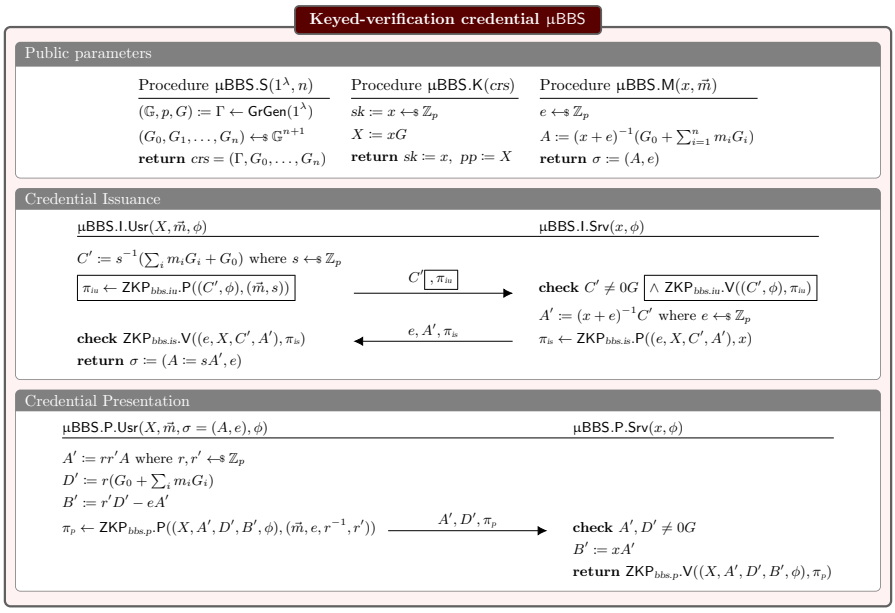


Figure 6: The keyed-verification credential system μBBS. Differs from [BBDT17] in issuance and presentation. The variable ϕ denotes the arbitrary predicates that must be enforced on the attributes during issuance or presentation. Boxed, the part that may be removed for anonymous tokens (one-more unforgeability). The zero-knowledge relations associated to the proof system (for user issuance, server issuance, and presentation respectively) are defined in equations (17) to (19).

- The server picks e uniformly at random from $\mathbb{Z}_p$ for each signature (as displayed in figure 6). In this case, EGen does not require a state nor a message. In the case where $p > 2^{2\lambda}$, EGen may just sample a uniformly random string of length 2λ and read it as an integer $\pmod p$.
- The server can hash the message into $e = \mathsf{H}_s(\vec{m})$. In this case, EGen does not require a state.
- The server can be stateful and set e to be a counter. In this case, EGen does not require a message, but it is stateful; the counter concretely may be much smaller than 2λ bits – e.g., 64 bits would suffice, and the probability that any two e-values are equal is zero.

We denote the collision advantage for this procedure as $\delta(\lambda, q)$, that is, the probability that an adversary A has, making q queries to an oracle that for the i-th query $\vec{m}_i$ returns $(e_i, st_{i+1}) \leftarrow \mathsf{EGen}(st_i, \vec{m}_i)$ with $st_0 := (crs, 0)$. In [TZ23], this quantity is indicated as $\delta_{\mathsf{eS},\mathsf{eG}}(q, \lambda)$.

A special property of BBS signatures is that it is possible to extend the credential system for more attributes without requiring a new key generation or re-issuance of a credential, and that the public parameters pp are a single element. We prove the following theorem:

Theorem 16. *In the algebraic group model,* μBBS *is an algebraic MAC for $n = \mathsf{poly}(\lambda)$ attributes in $\mathbb{Z}_p$ with advantage:*

$$\mathsf{Adv}^{\mathrm{ufcmva}}_{\mu\mathsf{BBS}}(\lambda, n) = \mathsf{Adv}^{(\mathrm{q+2})\text{-dl}}_{\mathsf{GrGen}}(\lambda) + \mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda) + \frac{3}{p} + \delta(\lambda, q) \ ,$$

where q is the number of signing queries made by the adversary.

In particular, when e is sampled as a uniformly-random 2λ-bit string, we have $\delta(\lambda, q) \leq q^2/2^{2\lambda}$. The proof follows along the lines of [TZ23] with the major difference being that we need to expose two additional oracles: one for (keyed-)verification, and a helper oracle that will be indispensable for proving the credential system.

Blind issuance. The user commits to the message as $C' = s^{-1}(\sum_i m_i G_i + G_0)$ where $s \leftarrow\$ \mathbb{Z}_p$, and proves knowledge of the discrete logarithm of C' with respect to $(G_0, G_1, \ldots, G_n)$ via $\mathsf{ZKP}_{bbs.iu}$, a simulation-extractable proof system for the relation

$$\mathsf{R}_{bbs.iu} := \{((C', \phi), (\vec{m}, s)) : G_0 = sC' - \textstyle\sum_i m_i G_i \ \wedge \ \phi(\vec{m}) = 1\} \tag{17}$$

The server then computes $A' = (x+e)^{-1}C'$ and proves knowledge of the discrete logarithm of (e, X, C', A') via $\mathsf{ZKP}_{bbs.is}$, a simulation-extractable proof system for the relation:

$$\mathsf{R}_{bbs.is} := \{((e, X, A', C'), x) : X = xG \ \wedge \ C' - eA' = xA'\} \ . \tag{18}$$

Remark 3. This is different from the original approach of Barki et al. [BBDT17] where the user commits to the message as $C' = \sum_i m_i G_i + sG$ and proves knowledge of the discrete logarithm of C' with respect to $(G_0, G_1, \ldots, G_n)$.

Presentation. The core idea for presentation is to note that:

$$xA - G_0 = \sum_{i=1}^{n} m_i G_i - eA$$

where the right-hand side can be seen as a Pedersen commitment over the attributes. So, all the user has to do is provide a proof for the above equality, after "blinding' A in order to provide anonymity across presentations. To present a credential, the user will send a commitment to A (denoted A') and then prove that xA' is a commitment to the different m_i's, and they satisfy the given predicate. More formally π_p is a zero-knowledge proof of knowledge for:

$$\mathsf{R}_{bbs.p} \coloneqq \left\{ ((X, A', D', B', \phi), (\vec{m}, e, r'', r')) : \begin{array}{l} B' = r'D' - eA' \ \wedge \ G_0 = r''D' - \sum_i m_i G_i \ \wedge \\ \phi(\vec{m}) = 1 \end{array} \right\} \tag{19}$$

as shown by Tessaro and Zhu [TZ23].

Remark 4 (Difference with [BBDT17])**.** In Barki et al., the user algorithm takes as input the private attributes $\vec{m}$ along with the tuple (A, e, s, C), where $(A, e) \in \mathbb{G} \times \mathbb{Z}_p$ is the standard BBS signature (denoted "(A, r)"), $s \in \mathbb{Z}_p$ is the nonce attribute (denoted "s_u"), and $C \in \mathbb{G}$ is the commitment to the message (denoted "$\tilde{C}_m$"). To present the credential, the user sends $(B_0 \coloneqq lA,\ C_0 \coloneqq lC - eB_0,\ E \coloneqq l^{-1}C_0 + tF)$ for $l, t \leftarrow\!\$\ \mathbb{Z}_p$. The guarantees are the same, the proof is only slightly less efficient.

Remark 5 (Differences with [BBS04])**.** The original BBS paper introduces a protocol for proving knowledge of a signature in the scope of group signatures. The main difference is in that the honest-verifier zero-knowledge of their protocol is only computational while here it is perfect.

Let $\mathsf{R}_{bbs} \coloneqq \mathsf{R}_{bbs.iu} \cup \mathsf{R}_{bbs.is} \cup \mathsf{R}_{bbs.p}$. We prove the following theorems.

Theorem 17. *If* ZKP *is a proof system for* $\mathsf{R} \supseteq \mathsf{R}_{bbs}$*, then* $\mu\mathsf{BBS}$ *is a keyed-verification extractable credential for* $n = \mathsf{poly}(\lambda)$ *attributes with anonymity advantage:*

$$\mathsf{Adv}^{\mathrm{anon}}_{\mu\mathsf{BBS}}(\lambda, n) \le \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{bbs.iu}}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{bbs.p}}(\lambda) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{bbs.is}}(\lambda) + \mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda) + \frac{1}{p} \ ,$$

and extractability advantage:

$$\mathsf{Adv}^{\mathrm{ext}}_{\mu\mathsf{BBS}}(\lambda, n) \le \mathsf{Adv}^{\mathrm{ufcmva}}_{\mu\mathsf{BBS}}(\lambda, n) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{bbs.iu}}(\lambda) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{bbs.p}}(\lambda) \ .$$

In particular, $\mu\mathsf{BBS}[\mathsf{ZKP} = \Sigma]$ has everlasting forward anonymity.[23]

Let $\mu\mathsf{BBS}_{AT}$ denote the variant of $\mu\mathsf{BBS}$ in figure 6 where the issuance algorithm does not return the proof π_{iu} (that is, the boxed areas are removed). From the above and theorem 23 we also have

Theorem 18. *If* ZKP *is a proof system for the relation* $\mathsf{R} \supseteq \mathsf{R}_{bbs.is} \cup \mathsf{R}_{bbs.p}$*, them* $\mu\mathsf{BBS}_{AT}$ *is an anonymous token system for* $n = \mathsf{poly}(\lambda)$ *attributes with anonymity advantage:*

$$\mathsf{Adv}^{\mathrm{anon}}_{\mu\mathsf{BBS}_{AT}}(\lambda, n) \le \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{bbs.iu}}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{bbs.p}}(\lambda) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{bbs.is}}(\lambda) + \mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda) \ ,$$

and (in the algebraic group model) one-more unforgeability advantage:

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mu\mathsf{BBS}_{AT}}(\lambda, n) \le \mathsf{Adv}^{(\mathrm{q}+2)\text{-dl}}_{\mathsf{GrGen}}(\lambda, 1) + \mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda) + \frac{1}{p} \ .$$

6.2 Analysis

6.2.1 Algebraic MAC

We prove that $\mu\mathsf{BBS}$ is a MAC for $n = 1$ attributes and then reduce the case of $n = \mathsf{poly}(\lambda)$ attributes to it. The proof will actually be about a stronger game where the adversary may also query an oracle $\textsc{Ddh}(A, B)$ to check if $B = xA$.

[23] Recall that forward anonymity is concerned with unbounded presentation distinguishers.

Lemma 19. *In the algebraic group model,* μBBS *is an algebraic MAC for $n = 1$ attributes in $\mathbb{Z}_p$ with advantage:*

$$\mathsf{Adv}^{\text{ufcmva}}_{\mu\mathsf{BBS}}(\lambda, 1) \leq \mathsf{Adv}^{\text{(q+2)-dl}}_{\mathsf{GrGen}}(\lambda) + \frac{1}{p} + \delta(\lambda, q) \ ,$$

where q is the number of signing queries made by the adversary.

Proof. An algebraic adversary A for the unforgeability game receives as input the public parameters $((\Gamma, G_0, G_1, \ldots, G_n), X)$ and during its execution can query an oracle $\textsc{Sign}(\vec{m})$ that returns a MAC (A, e) valid on $\vec{m}$ and a verification oracle $\textsc{Verify}(\vec{m}, (A, e))$ that returns 1 if the MAC is valid and 0 otherwise. During its execution, it makes at most q queries to the oracle $\textsc{Sign}$. The output forgery is a message m^* not previously queried, a MAC (A^*, e^*) accompanied by an algebraic representation $\vec{\alpha}$ such that

$$A^* = \alpha_{g,0} G_0 + \alpha_{g,1} G_1 + \alpha_g G + \alpha_x X + \sum_{j=1}^{q} \alpha_{a,j} A_j \ , \tag{20}$$

where q is the number of MACs produced so far. Form the algebraic group model, we have that the equality

$$A^* = \frac{\boldsymbol{g_0} + m^* \boldsymbol{g_1}}{\boldsymbol{x} + e^*} G \tag{21}$$

holds when $\boldsymbol{g_0}, \boldsymbol{g_1}, \boldsymbol{x}$ are the discrete logarithm of the respective group elements G_0, G_1, X base G. Putting together equations (20) and (21), we obtain the rational function in $\boldsymbol{g_0}, \boldsymbol{g_1}, \boldsymbol{x}$ over $\mathbb{Z}_p$

$$\frac{\boldsymbol{g_0} + m^* \boldsymbol{g_1}}{\boldsymbol{x} + e^*} = \alpha_g + \alpha_{g,0} \boldsymbol{g_0} + \alpha_{g,1} \boldsymbol{g_1} + \alpha_x \boldsymbol{x} + \sum_{j=1}^{q} \alpha_{a,j} \frac{\boldsymbol{g_0} + m_j \boldsymbol{g_1}}{\boldsymbol{x} + e_j}$$

we have two possibilities: $e^* \in \{e_1, \ldots, e_q\}$ or $e^* \notin \{e_1, \ldots, e_q\}$.

In the first case, let $f(\boldsymbol{x}) = \prod_j (\boldsymbol{x} + e_j)$ and $f_k(\boldsymbol{x}) = \prod_{j \neq k} (\boldsymbol{x} + e_j) = f(\boldsymbol{x})/(\boldsymbol{x} + e_k)$. Let the e_i's be all different (this will always happen except with probability $\delta(\lambda, q)$). Without loss of generality, let $e^* = e_1$. We have:

$$((1 - \alpha_{a,1}) \boldsymbol{g_0} + (m^* - \alpha_{a,1} m_1) \boldsymbol{g_1}) \, f_1(\boldsymbol{x}) = (\alpha_g + \alpha_{g,0} \boldsymbol{g_0} + \alpha_{g,1} \boldsymbol{g_1} + \alpha_x \boldsymbol{x}) f(\boldsymbol{x}) + \sum_{j=2}^{q} \alpha_{a,j} (\boldsymbol{g_0} + m_j \boldsymbol{g_1}) f_j(\boldsymbol{x})$$

and since $(\boldsymbol{x} + e_1)$ divides the right-hand side but not the left-hand side, then $\alpha_1 = 1$ and $m_1 = m^*$ which leads to a contradiction.

If $e^* \notin \{e_1, \ldots, e_q\}$, then let for simplicity $e_{q+1} = e^*$ and let $f^* = \prod_j^{q+1} (\boldsymbol{x} + e_j)$ and, similarly to before, $f^*_k = f(\boldsymbol{x})/(\boldsymbol{x} + e_k)$. Then we have

$$(\boldsymbol{g_0} + m^* \boldsymbol{g_1}) f^*_{q+1}(\boldsymbol{x}) = (\alpha_g + \alpha_{g,0} \boldsymbol{g_0} + \alpha_{g,1} \boldsymbol{g_1} + \alpha_x \boldsymbol{x}) f^*(\boldsymbol{x}) + \sum_{j=1}^{q} \alpha_{a,j} (\boldsymbol{g_0} + m_j \boldsymbol{g_1}) f^*_j(\boldsymbol{x})$$

Similarly to the previous case, we have that the right-hand side is divisible by $(\boldsymbol{x} + e^*)$, but the left-hand side is not, leading to a contradiction. This implies that, in the above polynomial equation, at least two monomials are different.

We build a reduction to $(q+2)$-DL. The reduction B receives as input $(\Gamma = (\mathbb{G}, p, G), X^{(1)}, X^{(2)}, \ldots, X^{(q+1)}, x^{(q+2)})$. Let $X^{(0)} \coloneqq G$. B samples $e_1, \ldots, e_q$ from EGen. (In the figure, $e_i \leftarrow\!\!\$\ \mathbb{Z}_p$.) In the fortunate case where $\log_G X \in \{0, 1, -e_1, \ldots, -e_q\}$, B stops and returns the discrete logarithm found. Let $f = \prod_{j=1}^{q} (\boldsymbol{x} + e_j)$ and $f_k = f(\boldsymbol{x})/(\boldsymbol{x} + e_k)$ for $k \in [q]$. Note that, f and f_k can be evaluated the group $\mathbb{G}$ using the "powers of X" (the $X^{(j)}$'s) given as input. Sample $a_0, b_0, a_1, b_1 \leftarrow\!\!\$\ \mathbb{Z}_p$ and let $G' \coloneqq f(X)$, $X \coloneqq X^{(1)}$, $G_0 \coloneqq ((a_0 + \boldsymbol{x} b_0) f)(X)$ and $G_1 \coloneqq ((a_1 + \boldsymbol{x} b_1) f)(X)$.(By $f(X)$ we mean $\sum_i f_i X^{(i)}$.) The degree of both polynomials is at most $q + 1$. Invoke A with public parameters $((\Gamma, G_0, G_1), X)$. To respond the j-th query to the signing oracle, denoted $\textsc{Sign}(m_j)$, compute

$$A_j = (a_0 + m_j a_1) \sum_{i=0}^{q} f_{j,i} X^{(i)} + (b_0 + m_j b_1) \sum_{i=0}^{q} f_{j,i} X^{(i+1)}$$

and return (A_j, e_j). For any verification query $\textsc{Verify}(m, (A, e, \vec{\alpha}))$, where $\vec{\alpha}$ denotes the algebraic representation of A (similar to equation (20)) check the verification equation over the algebraic representation. To do so, consider the following multivariate polynomial in $\mathbb{Z}_p[\boldsymbol{g_0}, \boldsymbol{g_1}, \boldsymbol{x}]$

$$\varphi_{m,e}(\boldsymbol{g_0}, \boldsymbol{g_1}, \boldsymbol{x}) \coloneqq (\boldsymbol{g_0} + m \boldsymbol{g_1}) f(\boldsymbol{x}) - (\boldsymbol{x} + e) \left(\alpha_g f(\boldsymbol{x}) + \alpha_{g,0} \boldsymbol{g_0} + \alpha_{g,1} \boldsymbol{g_1} - \alpha_x \boldsymbol{x} + \textstyle\sum_j \alpha_{a,j} (\boldsymbol{g_0} + \boldsymbol{g_1} m_j) f_j(\boldsymbol{x}) \right)$$

The polynomial $\varphi_{m,e}(a_0 + \boldsymbol{x}b_0, a_1 + \boldsymbol{x}b_1, \boldsymbol{x})$ can be tested for zero when evaluated in $\log_G X$ using the inputs $G, X^{(1)}, \ldots, X^{(q+2)}$ since the resulting polynomial is of degree at most $q+2$. For every DDH oracle (A, B), consider the algebraic representation of A, which is similar to equation (20). Replacing each group element with its representation in the DL challenge $(G, X^{(1)}, X^{(2)}, \ldots, X^{(q+1)})$ we obtain a vector $\vec{\alpha}'$ such that $A = \alpha'_0 G + \sum_{j=1}^{q+1} \alpha' X^j)$. Return 1 if $B = \sum_{j=1}^{q+2} \alpha'_j X^{(j)}$. At the end the adversary returns a forgery $(m^*, (A^*, e^*))$.

From the argument at the beginning of the proof, we have $\varphi_{m^*,e^*}(\boldsymbol{g_0}, \boldsymbol{g_1}, \boldsymbol{x}) \neq 0$. Consider then the partial evaluation $\phi(\boldsymbol{x}) \coloneqq \varphi((a_0 + b_0\boldsymbol{x}), (a_1 + b_1\boldsymbol{x}), \boldsymbol{x})$ and note that the resulting polynomial is of degree at most $q+2$ and is with overwhelming probability non-trivial too since b_0, b_1 are uniformly random and perfectly hidden in G_0, G_1 by a_0, a_1. One of the root of this polynomial must be the discrete logarithm of X. The reduction B uses Berlekamp's algorithm to factor ϕ and recovers the discrete logarithm of X. □

Theorem 20. $\mu\mathsf{BBS}$ *is an algebraic MAC for* $n = \mathsf{poly}(\lambda)$ *attributes in* $\mathbb{Z}_p$ *with advantage:*

$$\mathsf{Adv}^{\mathrm{ufcmva}}_{\mu\mathsf{BBS}}(\lambda, n) \leq \mathsf{Adv}^{\mathrm{ufcmva}}_{\mu\mathsf{BBS}}(\lambda, 1) + \mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda) + \frac{2}{p} \ .$$

Proof. Let A be a p.p.t. adversary for unforgeability. It receives as input the public parameters $pp = (\Gamma, (G_0, \ldots, G_n), X)$, during its executions it queries SIGN with messages $\vec{m}_j$ for $j \in [q]$ (q denoting the number of queries of A); finally, it returns a forgery satisfying one of the following:

(i) the output $(\vec{m}^*, (A^*, e^*))$ is a valid forgery and $\sum_i m^*_i G_i = \sum_i m_{j,i} G_i$ for some $j \in [q]$
(ii) the output $(\vec{m}^*, (A^*, e^*))$ is a valid and $\sum_i m^*_i G_i \neq \sum_i m_{j,i} G_i$ for all $j \in [q]$

We claim that (i) happens at most with negligible probability. To do so, we build an adversary B that solves DL every time A wins the game with a forgery satisfying $\sum_i (m^*_i - m_{j,i}) G_i = 0G$. Let $\mathsf{B}(\Gamma, Y)$ be an adversary for $\mathrm{DL}_{\mathsf{GrGen}}(\lambda)$. B samples at random $a_0, a_1, \ldots, a_n, b_1, \ldots, b_n, x \leftarrow\!\!\$\ \mathbb{Z}_p$ and sets $G_0 \coloneqq a_0 G$, $G_i = a_i G + b_i Y$ for all $i \in [n]$ and finally sets $X \coloneqq xG$ for a random $x \leftarrow\!\!\$\ \mathbb{Z}_p$. It invokes $\mathsf{A}((\Gamma, G_0, \vec{G}), X)$. Queries to the oracles $\textsc{Sign}(\vec{m}_j)$ are answered with (A_j, e_j), where $e_j \leftarrow\!\!\$\ \mathbb{Z}_p$ and $A_j = (x + e_j)^{-1}(G_0 + \sum_i m_i G_i)$; queries $\textsc{Verify}(\vec{m}, (A, e))$ are answered testing the equality $(x + e)A = G_0 + \sum_i m_i G_i$. To any $\textsc{Ddh}(A, B)$ query, it responds 1 if $xA = B$. Given as output a forgery satisfying item (i), we have that there exists a $j \in [q]$ such that:

$$\sum_{i=1}^{n} (m^*_i - m_{j,i}) a_i G = \sum_{i=1}^{n} (m^*_i - m_{j,i}) b_i Y$$

and the above equation is non-trivial with overwhelming probability since $\vec{m}^* \neq \vec{m}_j$ (and thus $\exists i. m^*_i \neq m_{j,i}$) and the b_i's are uniformly distributed and perfectly hidden. Therefore, with overwhelming probability B can compute $\log_G Y = (\sum_i (m^*_i - m_{j,i}) b_i)^{-1} (\sum_i (m^*_i - m_{j,i}) a_i)$.

The claim item (ii) reduces to unforgeability of $\mu\mathsf{BBS}$ for $n = 1$. Let B be an unforgeability adversary receiving as input Γ, G_0, G_1, X. Let $G_i = a_i G_1$ for $i \in [2, n]$ with $a_i \leftarrow\!\!\$\ \mathbb{Z}_p$. The two are identically distributed if $G_1 \neq 0G$. Invoke $\mathsf{A}((\Gamma, G_0, G_1, \ldots, G_n), X)$. For every signing query for a message $\vec{m}_j$, respond with $\textsc{Sign}(\sum_i a_i m_{j,i})$. For every verification query for a message $\vec{m}$, respond with $\textsc{Verify}(\sum_i a_i m_i)$. All DDH queries are sent to the challenger as-is. At the end, we obtain an output forgery satisfying item (ii) and therefore $(\sum_i a_i m^*_i, (A^*, e^*))$ is a valid forgery also for $\mu\mathsf{BBS}$ for $n = 1$. □

6.2.2 Keyed-verification credential

Theorem 21. *Let* ZKP *be a proof system for the relation* $\mathsf{R} \supseteq \mathsf{R}_{bbs}$*, then* $\mu\mathsf{BBS}$ *is anonymous for* $n = \mathsf{poly}(\lambda)$ *attributes over* $\mathbb{Z}_p$ *with advantage:*

$$\mathsf{Adv}^{\mathrm{anon}}_{\mu\mathsf{BBS},\mathsf{A},\mathsf{D}}(\lambda, n) \leq \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{bbs.iu},\mathsf{A}'}(\lambda) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{bbs.is},\mathsf{A}''}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{bbs.p},\mathsf{D}'}(\lambda) + \mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen},\mathsf{A}'''}(\lambda) + \frac{1}{p} \ ,$$

where A, D *are adversaries in the anonymity game and* $\mathsf{A}', \mathsf{A}'', \mathsf{A}''', \mathsf{D}$ *are described in the proof.*

Proof. We define the simulator Sim as follows.

- Let $\mathsf{Sim.I}$ sample a uniformly-random group element $C' \leftarrow\!\!\$\ \mathbb{G}$ and invoke $\mathsf{ZKP}_{bbs.iu}.\mathsf{Sim}$ on the statement (C', ϕ) obtaining a simulated proof π_{iu}. Upon receiving the server response (e, A', π_{is}), verify the zero-knowledge proof via $\mathsf{ZKP.V}((e, X, A', C'), \pi_{is})$ and (if the output is 1) proceed extracting a witness x from π_{is} via the extractor for $\mathsf{ZKP}_{bbs.is}$ for the statement (e, X, A', C'). Store the simulator state $st_{\mathsf{Sim}} = (x)$.
- The presentation simulator $\mathsf{Sim.P}(x, \phi)$ samples random A', D' and simulates the zero-knowledge proof π_p for the statement (X, A', D', B', ϕ) where $B' = xA'$. It returns (A', D', π_p).

We show that the real and simulated distributions are indistinguishable via a hybrid argument.

H_0 This first hybrid is the honest interaction, where issuance and presentation are done honestly using the input message $\vec{m}$.

H_1 Instead of generating π_{iu} using the witness, we do so via the zero-knowledge simulator of $\mathsf{ZKP}_{bbs.iu}$ with input (C', ϕ). If A's output is noticeably different after this hybrid change, then it is possible to build an adversary A' that is able to distinguish real from simulated proofs for $\mathsf{ZKP}_{bbs.iu}$. Therefore:

$$\mathsf{Adv}^{\mathsf{H}_1}_{\mathsf{A},\mathsf{D}}(\lambda, n) \geq \mathsf{Adv}^{\mathsf{H}_0}_{\mathsf{A},\mathsf{D}}(\lambda, n) - \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{bbs.iu},\mathsf{A}'}(\lambda) \ .$$

H_2 Instead of generating π_p using the attributes $\vec{m}$ and MAC σ, use the zero-knowledge simulator of $\mathsf{ZKP}_{bbs.p}$ with input (X, A', D', B', ϕ). Similarly to the previous hybrid, there exists an adversary D' that wins every time D's output is noticeably different. Therefore:

$$\mathsf{Adv}^{\mathsf{H}_2}_{\mathsf{A},\mathsf{D}}(\lambda, n) \geq \mathsf{Adv}^{\mathsf{H}_1}_{\mathsf{A},\mathsf{D}}(\lambda, n) - \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{ZKP}_{bbs.p},\mathsf{A}'}(\lambda) \ .$$

H_3 Extract from π_{is} a witness for the statement (e, X, A', C') and check its validity.
The extractor will produce a witness x satisfying:

$$x \begin{bmatrix} G \\ A' \end{bmatrix} = \begin{bmatrix} X \\ C' - eA' \end{bmatrix} \ ,$$

except with probability $\mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{bbs.is},\mathsf{A}''}(\lambda)$ where A'' is the adversary in the knowledge soundness game that runs A internally and returns the proof π_{is} for which extraction fails. In other words:

$$\mathsf{Adv}^{\mathsf{H}_1}_{\mathsf{A},\mathsf{D}}(\lambda, n) \geq \mathsf{Adv}^{\mathsf{H}_3}_{\mathsf{A},\mathsf{D}}(\lambda, n) - \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{bbs.is},\mathsf{A}''}(\lambda) \ .$$

H_4 If $G_0 + \sum_i m_i G_i = 0G$, abort.
In the real distribution, if $G_0 + \sum_i m_i G_i = 0G$ the user will always send $C' = 0G$, whereas in the simulated case the simulator aborts. The above bad event happens only with advantage $\mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda)$. Let (Γ, Y) be a DL challenge and A be an adversary for the anonymity game. The reduction A''' invokes A with inputs

$$pp \coloneqq (\Gamma, X \coloneqq xG, G_0 = a_0 G + b_0 Y, \ldots, G_n \coloneqq a_n G + b_n Y)$$

where $x, a_0, \ldots, a_n, b_0, \ldots, b_n \leftarrow\!\$\ \mathbb{Z}_p$. At issuance time the adversary makes queries a message $\vec{m}$ and A''' checks if $G_0 + \sum_i m_i G_i = 0G$. If so, then we have $(a_0 + \sum_i m_i a_i)G = (b_0 + \sum_i m_i b_i)Y$ and the equation is non-trivial with overwhelming probability $1/p$. The reduction A''' can thus recover the discrete logarithm of Y. Therefore:

$$\mathsf{Adv}^{\mathsf{H}_3}_{\mathsf{A},\mathsf{D}}(\lambda, n) \geq \mathsf{Adv}^{\mathsf{H}_4}_{\mathsf{A},\mathsf{D}}(\lambda, n) - \mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen},\mathsf{A}'''}(\lambda) \ .$$

H_5 Replace C' with a uniformly-distributed group element $C' \leftarrow\!\$\ \mathbb{G}$. The two distributions are perfectly indistinguishable, that is:

$$\mathsf{Adv}^{\mathsf{H}_4}_{\mathsf{A},\mathsf{D}}(\lambda, n) = \mathsf{Adv}^{\mathsf{H}_5}_{\mathsf{A},\mathsf{D}}(\lambda, n) \ .$$

H_6 Finally, we replace the responses of the presentation oracle. Sample $A', D' \leftarrow\!\$\ \mathbb{G}$ and compute B' differently: instead of setting $B' \coloneqq rr'((G_0 + \sum_i m_i G_i) - eA)$ with $r, r' \leftarrow\!\$\ \mathbb{Z}_p$ and (A, e) the MAC obtained from the server, we set $B' \coloneqq xA'$. The distribution of the two is exactly the same. In fact, fixed A', D', for any message $\vec{m} \in \mathbb{Z}_p^n$ and $e \in \mathbb{Z}_p$ there exist $(\rho', \rho'') \in \mathbb{Z}_p^2$ such that:

$$B' = xA' = \rho' D' - eA' \ ,$$
$$\rho'' D' = \sum_i m_i G_i + G_0 \ ,$$

if $D' \neq 0G$. Thus:

$$\mathsf{Adv}^{\mathsf{H}_5}_{\mathsf{A},\mathsf{D}}(\lambda, n) = \mathsf{Adv}^{\mathsf{H}_6}_{\mathsf{A},\mathsf{D}}(\lambda, n) + \frac{1}{p} \ .$$

The last hybrid is running the simulator code. By the difference lemma, the result follows. □

Theorem 22. *If* ZKP *is a proof system for the relation* $\mathsf{R} \supseteq \mathsf{R}_{bbs}$*, then* $\mu\mathsf{BBS}$ *for* $n = \mathsf{poly}(\lambda)$ *is extractable with advantage:*

$$\mathsf{Adv}^{\mathrm{ext}}_{\mu\mathsf{BBS}}(\lambda, n) \leq \mathsf{Adv}^{\mathrm{ufcmva}}_{\mu\mathsf{BBS}}(\lambda, n) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{bbs.iu}}(\lambda) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{ZKP}_{bbs.p}}(\lambda) \ .$$

Proof. We define the extractors for the credential system as follows.

– $\mathsf{Ext.I}$ receives as input (x, ϕ, C', π_{iu}) and recovers a witness $(\vec{m}, s)$ for the instance (C', ϕ) such that $\phi(\vec{m}) = 1$ and

$$\begin{bmatrix} C' & -G_1 & \cdots & -G_n \end{bmatrix} \begin{bmatrix} s \\ m_1 \\ \vdots \\ m_n \end{bmatrix} = G_0 \ . \tag{22}$$

If extraction fails the extractor aborts and the adversary wins.

– $\mathsf{Ext.P}$ receives as input (x, ϕ) and a prover message (A', D', π_p, B'). It checks if B' is well-formed via $\textsc{Ddh}$ and, if that is the case, it internally runs $\mathsf{ZKP}_{bbs.p}.\mathsf{Ext}$ on input (X, A', D', B', ϕ) where $X = xG$ and obtain a witness $(\vec{m}, e, r'', r')$ such that $\phi(\vec{m}) = 1$ and

$$\begin{bmatrix} -G_1 & -G_2 & \cdots & -G_n & & D' & \\ & & & & D' & & -A' \end{bmatrix} \begin{bmatrix} m_1 \\ \vdots \\ m_n \\ e \\ r'' \\ r' \end{bmatrix} = \begin{bmatrix} G_0 \\ B' \end{bmatrix} \ . \tag{23}$$

If the recovered witness does not satisfy the above equation, return $\perp$. A wins if one of the following happens:

(i) Takahashihe adversary returns $(\phi, \mu = (C', \pi_{iu}))$ such that $\mathsf{ZKP}_{bbs.iu}$ verifies π_{iu} for the statement (C', ϕ) but extraction failed.
(ii) The adversary returns $(\phi^*, \rho^* = (A', D', \pi_p, B'))$ such that extraction fails for the instance (A', D', B', ϕ^*).
(iii) The presentation message returned from the adversary $\rho^* = (A', D', \pi_p, B')$ satisfies $\mu\mathsf{BBS.P.Srv}(x, \phi^*, \rho^*)$ and the message $\vec{m}^*$ extracted from π_p is such that $\vec{m} \notin Qrs$.

The first two cases can be reduced to knowledge soundness of the proof system. We are left with the last case, which reduces to unforgeability of $\mu\mathsf{BBS}$. We do so by building an adversary B for unforgeability that internally uses the adversary A.

The reduction B receives as input the public parameters $pp = (\Gamma, G_0, G_1, \ldots, G_n, X)$ and internally invokes the adversary A. For every $\textsc{Issue}(\phi, (C', \pi_{iu}))$ query, it runs the extractor and obtains $(\vec{m}, s)$ satisfying equation (22). It queries $\textsc{Sign}(\vec{m})$ obtaining a MAC (A, e). Returns $(A' \coloneqq s^{-1}A, e)$ (or $\perp$ if $s = 0$). For every $\textsc{Present}(\phi^*, (A', D', \pi_p, B'))$ query, B queries $\textsc{Ddh}(A', B')$ to check that $B' = xA'$ and then runs the extractor to obtain $(\vec{m}, e, r'', r')$ satisfying equation (23). It queries the verification oracle on the message $\vec{m}$ and MAC $(A = r''/r'A', e)$. It proceeds similarly for the returned message, outputting a forgery $\vec{m}^*$ (the message extracted from ρ^*) and $(A = r''/r'A', e)$.

The output of $\textsc{Sign}$ follow exactly the same distribution as in the original game, since from equation (22) (and commutativity in $\mathbb{Z}_p$)

$$\frac{1}{x+e}C' = s^{-1}\left(\frac{1}{x+e}(G_0 + \sum_{i=1}^{n} m_i G_i)\right)$$

The unfortunate case where $s = 0$ implies (via equation (22)) that $G_0 + \sum_i m_i G_i = 0$ and therefore $C' = 0$, so we return $\perp$ in both cases. The output of $\textsc{Verify}$ is also well-formed since, from π_p:

$$B' = xA' = r'D' - eA' = \frac{r'}{r''}(\sum_i m_i G_i + G_0) - eA' \implies A' = \frac{r'}{r''} \cdot \frac{1}{x+e} \cdot (\sum_i m_i G_i + G_0)$$

The result follows. □

6.2.3 One-more unforgeability

Theorem 23. *In the algebraic group model,* $\mu\mathsf{BBS}_{AT}$ *is one-more unforgeable with advantage:*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mu\mathsf{BBS}_{AT}}(\lambda, n) \leq \mathsf{Adv}^{(q+2)\text{-dl}}_{\mathsf{GrGen}}(\lambda) + \mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda) + \frac{1}{p} \ .$$

The proof is the same as in [TZ23]. We complement their analysis with a more direct reduction to the q-DL assumption.

Lemma 24. *There exists a reduction for the game* $\mathrm{OMUF}_{\mu\mathsf{BBS}_{AT},\mathsf{A}}(\lambda, n)$ *to* $q\text{-}\mathrm{DL}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ *with: q signing queries,* $O(q^3)$ *group operations,* $O(q)$ *space.*

We construct an adversary A that, given as input $X \in \mathbb{G}$ and access to a signing oracle $\textsc{Sign}(\cdot)$ performing a MAC on the input group element, outputs a q-DL challenge $(P, xP, \ldots, x^q P)$ for some generator $P \in \mathbb{G}$.

Let $A_0 \coloneqq G$. For each $i \in [q]$, the attacker A queries $\textsc{Sign}(A_{i-1})$ obtaining the MACs (A_i, e_i). We have the invariant

$$\boldsymbol{x} A_i = A_{i-1} - e_i A_i \ , \tag{24}$$

for each $i \in [q]$, with $\boldsymbol{x}$ denoting the unknown. In particular, we have that:

$$A_{q-k} = \prod_{i=k}^{q} (\boldsymbol{x} + e_i) A_q = \sum_i \nu_i^{(k)} \boldsymbol{x}^i A_q$$

where the polynomial coefficients $\nu_i^{(k)}$ can be computed in time $O(q^2)$ naïvely for each $k \in [q]$. In particular, the k-th polynomial is monic, and the leading term can always be isolated by recursively applying equation (24) to obtain an expression independent of $\boldsymbol{x}$, that is:

$$\begin{aligned} \boldsymbol{x} A_q &= A_{q-1} - e_q A_q =: B_1 \\ \boldsymbol{x}^2 A_q &= A_{q-2} - e_{q-1} A_{q-1} - e_q B_1 =: B_2 \\ &\vdots \end{aligned}$$

(Since $i < q$, then $\boldsymbol{x}^i A_q$ will stop when replacing the term A_{q-i}). A returns $(A_q, B_1, B_2, \ldots, B_{q-1})$.

Corollary 25. *There exists a p.p.t. adversary* A *for one-more unforgeability of* $\mu\mathsf{BBS}_{AT}$ *that can recover the signing key in time* $O(\sqrt{p/d} + \sqrt{d})$ *and space* $\max\left\{\sqrt{p/d}, \sqrt{d}\right\}$ *where* $d \mid p \pm 1$ *is the number of signing queries allowed.*

7 Designated-verifier fully-succinct SNARKs without pairings

A polynomial commitment scheme (PCS) is designated-verifier (dv) if only the holder of the *secret* verification key vk can verify the proof, as opposed to standard PCS definitions where vk can be efficiently computed from the proving key pk. The definition below, is an adaptation of [CHM+20].

Definition 26. A **designated-verifier polynomial commitment scheme** (PCS) over the field family $\mathcal{F} = \{\mathbb{F}_\lambda\}_{\lambda \in \mathbb{N}}$ is a tuple of algorithms $\mathsf{PCS} = (\mathsf{S}, \mathsf{K}, \mathsf{C}, \mathsf{E}, \mathsf{V})$ where

- $crs \leftarrow \mathsf{PCS.S}(1^\lambda, d)$ given as input the security parameter in unary form and a maximum degree $d > 0$, outputs a common reference string crs and defines the field $\mathbb{F} \in \mathcal{F}$.
- $(pk, vk) \leftarrow \mathsf{PCS.K}(crs, d)$ outputs a key pair (pk, vk) called respectively proving key and verification key. The proving key allows committing to polynomials over $\mathbb{F}$ up to degree d.
- $(C, r) \leftarrow \mathsf{PCS.C}(pk, f)$ outputs a commitment C to a polynomial $f(\boldsymbol{x}) \in \mathbb{F}[\boldsymbol{x}]$ represented via its coefficients f_i's, together with some opening information r.

  When clear from the context, to facilitate exposition, we will write $C \leftarrow \mathsf{PCS.C}(f; r)$, omit pk and using the random coins r.
- $\pi \leftarrow \mathsf{PCS.E}(pk, (f, r), z, y)$ outputs an *evaluation proof* π, showing that $y = f(z)$.
- $0/1 \leftarrow \mathsf{PCS.V}(vk, C, z, y, \pi)$ outputs 1 if the evaluation proof is correct and 0 otherwise.

A PCS satisfies completeness, extractability, and hiding.

Completeness requires that all honestly-generated proofs verify. More precisely, for any $f \in \mathbb{Z}_p[\boldsymbol{x}]$ of degree $d > 0$, $z \in \mathbb{Z}_p$ and $y = f(z)$:

$$\Pr\left[\mathsf{PCS.V}(vk, C, z, y, \pi) = 1 : \begin{array}{l} crs \leftarrow \mathsf{PCS.S}(1^\lambda, d) \\ (pk, vk) \leftarrow \mathsf{PCS.K}(crs) \\ (C, r) \leftarrow \mathsf{PCS.C}(pk, f) \\ \pi \leftarrow \mathsf{PCS.E}(pk, (f, r), z, y) \end{array}\right] = 1 \ .$$

Extractability asks that there exists an extractor Ext such that, for any p.p.t. adversary A, B, and query sampler Q[24] sharing the same random coins:

$$\Pr\left[\begin{array}{l} \mathsf{PCS.V}(vk, C, z, y, \pi) = 1 \\ \Downarrow \\ \deg(f) \le d \ \wedge \ y = f(z) \end{array} : \begin{array}{l} crs \leftarrow \mathsf{PCS.S}(1^\lambda, d) \\ (pk, vk) \leftarrow \mathsf{PCS.K}(crs) \\ \rho \leftarrow\$ \{0,1\}^{\mathsf{A.rl}(\lambda)} \\ C \coloneqq \mathsf{A}(pk; \rho) \\ f \coloneqq \mathsf{Ext}(pk; \rho) \\ z \coloneqq \mathsf{Q}(pk; \rho) \\ (\pi, y) \coloneqq \mathsf{B}(pk; \rho) \end{array}\right] \le \mathsf{negl}(\lambda) \ .$$

Hiding asks that for any $d > 0$, adversaries A, D, and any $crs \in [\mathsf{PCS.S}(1^\lambda, d)]$, $(pk, vk) \in [\mathsf{PCS.K}(crs)]$ following experiments are indistinguishable:

$$\left|\Pr\left[b' = 1 : \begin{array}{l} f \leftarrow \mathsf{A}(pk, vk) \\ (C, r) \leftarrow \mathsf{PCS.C}(pk, f) \\ b' \leftarrow \mathsf{D}^{\textsc{Eval}}(pk, C) \end{array}\right] - \Pr\left[b' = 1 : \begin{array}{l} f \leftarrow \mathsf{A}(pk, vk) \\ C \leftarrow \mathsf{Sim.C}(vk, \deg(f)) \\ b' \leftarrow \mathsf{D}^{\textsc{Eval}}(pk, C) \end{array}\right]\right| \le \mathsf{negl}(\lambda) \ ,$$

where $\textsc{Eval}(z)$ sets $y \coloneqq f(z)$ and, if the condition holds, in the real game returns $\mathsf{PCS.E}(pk, (f, r), z, y)$ and in the simulated game $\mathsf{Sim.E}(vk, z, y)$.

For the remainder of this work, we will consider homomorphic polynomial commitments, that is, $\mathsf{PCS.C}(f; r) + \mathsf{PCS.C}(g; s) = \mathsf{PCS.C}(f + g; r + s)$, where r, s are the randomness of the commit algorithms In particular the opening algorithm trivially recomputes the commitment with the randomness provided and checks that it matches the commitment given as input.

Malicious designated-verifier zero-knowledge [QRW19]. In practical applications, we would like the verifier to publish the proving key pk and privately store the secret key sk without relying on a trusted third party. In the context of keyed-verification credential systems, in fact, the verifier is also the entity publishing the public parameters. While the definition of hiding above holds for every keypair (pk, vk), one must make sure that the keypair is in the range of the setup algorithm to guarantee that hiding holds against malicious servers. (The proving key of the next section, for instance, holds a specific structure and proving pk is a correct output of the key generation is not immediate.) Our approach here will be to prove security for a proving key that is within the range of $\mathsf{PCS.S}$, and then expect the verifier publishing the proving key to prove that it is indeed in the range of $\mathsf{PCS.S}$.

Remark 6. The compiler of Chiesa et al. [CHM+20] introduces another procedure Trim which, given as input a prover key pk for a PCS of size D, and an integer $d < D$, it returns a new prover key pk' for a PCS of size d. This is trivial to implement for our schemes and is a technicality of the compiler, which requires the prover key to be generated before seeing the actual size of the index. The opening algorithm $\mathsf{PCS.O}$ and verification algorithms $\mathsf{PCS.V}$ (called Open and Check in the literature) can also be generalized as in [CHM+20, BDFG20] to allow for batch opening and evaluation checking. Given polynomials $f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}) \in \mathbb{F}[\boldsymbol{x}]$ claimed to evaluate in $y_1, \ldots, y_m$ in points $z_1, \ldots, z_m$, and given a random challenge $r \in \mathbb{F}$ one can consider euclidean division of the *batch polynomial* $f(\boldsymbol{x}) \coloneqq \sum_i r^i f_i(\boldsymbol{x})$ by $Z(\boldsymbol{x}) = \prod_i (\boldsymbol{x} - z_i)$:

$$f(\boldsymbol{x}) = Z(\boldsymbol{x}) q(\boldsymbol{x}) + r(\boldsymbol{x}) \ ,$$

where $q(\boldsymbol{x})$ is the polynomial that will be sent as proof to the verifier, and the polynomial $r(\boldsymbol{x})$ of degree $m-1$ is the polynomial defined by interpolation as $r(z_i) = y_i$.

7.1 Designated-verifier Kate–Zaverucha–Goldberg commitments

We describe a variant of KZG meant for designated-verifier proof systems. We assume that the polynomial is given in coefficient form as a vector of field elements.

Procedure $\mathsf{dvKZG.S}(1^\lambda, d)$

$\Gamma \coloneqq \mathsf{GrGen}(1^\lambda)$

return $crs \coloneqq (\Gamma, d)$

Procedure $\mathsf{dvKZG.K}(crs)$

$\tau \leftarrow\$ \mathbb{Z}_p^\times$

$T_i \coloneqq \tau^i G$ **for** $i \in [0, d+1]$

$\vec{T} \coloneqq (T_0, \ldots, T_d)$

$R \coloneqq \eta G$ where $\eta \leftarrow\$ \mathbb{Z}_p^\times$

return $pk \coloneqq (T_0, \vec{T}, R), vk \coloneqq (pk, \tau, \eta)$

Procedure $\mathsf{dvKZG.C}(pk, f(\boldsymbol{x}))$

$s \leftarrow\$ \mathbb{Z}_p$

return $C \coloneqq \sum_{i=0}^{d} f_i T_i + sR$

[24] A query samples for an interactive oracle proof is an efficient algorithm that, given as input the random coins of the verifier, returns the point at which the oracle messages of the prover is queried. See [CHM+20] for a more formal definition.

The idea for evaluation proofs stems from the observation that for a polynomial $f \in \mathbb{Z}_p[\boldsymbol{x}]$, the Euclidian division by $(\boldsymbol{x} - z)$ has reminder $y = f(z)$:

$$f(\boldsymbol{x}) = (\boldsymbol{x} - z)q(x) + y$$

The proof consists into exhibiting a quotient polynomial $q(x)$ satisfying the above (and accounting for the additional term $r(z)$ introduced in the commitment).

Procedure $\mathsf{dvKZG.E}(pk, f(\boldsymbol{x}), s, z, y)$	Procedure $\mathsf{dvKZG.V}(vk, C, z, y, (Q, D))$
$q(\boldsymbol{x}) \coloneqq (f(\boldsymbol{x}) - y)/(\boldsymbol{x} - z)$	**return** $(\tau - z)Q + yG + \eta D = C$
$Q \coloneqq \sum_i q_i T_i + s'R$ where $s' \leftarrow\!\!\$\ \mathbb{Z}_p$	
$D \coloneqq \sum_i \delta_i T_i$ where $\delta(\boldsymbol{x}) \coloneqq s - s'(\boldsymbol{x} - z)$	
return $\pi \coloneqq (Q, D)$	

Knowledge soundness holds in the algebraic group model under the $(q+1)$-DL assumption, and the proof is the same as [KT23, B.2.2]. We now focus on hiding.

Theorem 27. dvKZG *is a perfectly hiding polynomial commitment scheme.*

Proof. The simulator Sim obtains as input $vk = (pk = (T_0, \ldots, T_d, R), (\tau, \eta))$ such that:

$$\begin{aligned} \eta, \tau \in \mathbb{Z}_p^\times \iff T_1, R \neq 0G \ \wedge \\ \forall \tau \in [d] \colon \tau T_{i-1} = T_i \end{aligned} \tag{25}$$

Given vk along with a commitment C and an evaluation proof y, samples uniformly at random $Q \leftarrow\!\!\$\ \mathbb{G}$ and computes $D \coloneqq \eta^{-1}(C - (\tau - z)Q - yG)$.

We have thus a distinguishing game where an adversary A returns a polynomial f and $pk = (T_0, \vec{T}, R), vk = (\tau, \eta)$ are well-formed and satisfying equation (25). A has to distinguish between:

- a real interaction, where:

  $$C = \sum_i f_i T_i + sR = (f(\tau) + r\eta)G$$

  with $s \leftarrow\!\!\$\ \mathbb{Z}_p$, $\tau \coloneqq \log_G T_1$ and $\eta \coloneqq \log_G R$ with $\tau, \eta \neq 0$. A can query on any point z the evaluation oracle for $\textsc{Eval}(z)$ which in turn computes $\mathsf{dvKZG.E}(pk, (f, s), z, f(z))$ returning (Q, D) such that:

  $$\begin{aligned} Q = \sum_i q_i T_i + s'H = (q(\tau) + s'\eta)G \\ D = (r + s'z)T_0 - s'T_1 = (s + s'z - s'\tau)G \end{aligned} \tag{26}$$

  where the right-hand side is given by equation (25). From correctness, we have:

  $$(\tau - z)(q(\tau) + s'\eta) + y + \eta((s + s'z) - s'\tau) = f(\tau) + \eta s$$

- a simulated interaction, where the simulator Sim returns a commitment $C = \alpha G$ with $\alpha \leftarrow\!\!\$\ \mathbb{Z}_p$ as a commitment, and, for every evaluation query on $z \in \mathbb{Z}_p$ and y, the proof returned is:

  $$\begin{aligned} Q = \beta G \\ D = \eta^{-1}(C - (\tau - z)Q - y) = \eta^{-1}(\alpha - (\tau - z)\beta - y)G \end{aligned}$$

  which satisfies the verification equation since:

  $$(\tau - z)\beta + y + (\alpha - (\tau - z)\beta - y) = \alpha$$

  (note that, since $R \neq 0G$, η is invertible).

The two distributions are perfectly indistinguishable: both α and $f(\tau) + s\eta$ are uniformly distributed (since $\eta \neq 0$ and s is uniformly distributed), and the same can be argued about β and $q(\tau) + s'\eta$. Finally, D is uniquely determined by τ, η, C, Q, z, y and the verification equation is satisfied in both cases. □

Degree-check for polynomials whose degree may be less than d can be enforced using standard techniques as described in Kohrita and Towa [KT23, 5.2], roughly speaking by checking instead the equation:

$$(f - z)\boldsymbol{x}^{d - \deg(f)} = q\boldsymbol{x}^{d - \deg(f)}(\boldsymbol{x} - y) \ ,$$

by having the prover commit to the polynomial $f(\boldsymbol{x})\boldsymbol{x}^{d - \deg(f)}$.

7.2 IOP compiler for designated-verifier polynomial commitments

An indexed relation is a relation $\mathsf{R} = \{((i,x),w)\}$ consisting of an index i, an instance x, and a witness w. An interactive proof for an indexed relation generally has the prover's first message depend only on the index i and not depending on the prover's randomness. The first-message function is called indexer, and is relevant in the context of preprocessing SNARKs [CHM+20].

Definition 28 (PIOP)**.** A **polynomial interactive oracle proof** for an indexed relation R over a field family $\mathcal{F}$ is a tuple $\mathsf{IOP} = (k, o, d, \mathsf{I}, \mathsf{P}, \mathsf{V})$ where k, o, d are maps $\{0,1\}^* \to \mathbb{N}$ and $\mathsf{I}, \mathsf{P}, \mathsf{V}$ are algorithms called respectively indexer, prover, and verifier. The map k specifies the number of interaction rounds, o the number of polynomials in each round, and d the degree bound of these polynomials.

In the 0-th round (offline phase), the indexer $\mathsf{IOP.I}$ receives as input a field $\mathbb{F} \in \mathcal{F}$ and an index i for R, and outputs $o(0)$ polynomials $p_1^{(0)}(x), \ldots, p_{o(0)}^{(0)} \in \mathbb{F}[x]$ of degree at most $d(|i|,0,1), \ldots, d(|i|,0,o(n))$ respectively.[25]

In any other round (online phase), given an instance x and a witness w such that $(i,x,w) \in \mathsf{R}$, the prover $\mathsf{IOP.P}$ receives as input $(\mathbb{F}, i, x, w)$ and $\mathsf{IOP.V}$ receives as input oracle access to the polynomials output by $\mathsf{IOP.I}(\mathbb{F}, i)$. The prover $\mathsf{IOP.P}$ and the verifier $\mathsf{IOP.V}$ interact over $k(|i|)$ rounds. For $j \in [k(i)]$, in the j-th round of interaction the verifier $\mathsf{IOP.V}$ sends messages $m_j \in \mathbb{F}$ to the verifier $\mathsf{IOP.P}$; the prover then replies with $o(j)$ oracle polynomials $p_1^{(j)}, \ldots, p_{o(j)-1}^{(j)} \in \mathbb{F}[x]$. The verifier may query any of the polynomial it has received any number of times. A query consists of a point $z \in \mathbb{F}$ and the prover replies with the evaluation of the polynomial at that point. After the interaction the verifier accepts or rejects. The function d determines which prover to consider for the completeness and soundness properties of the proof system. In more detail, a prover P is admissible if for every $j \in [k(|i|)]$ and $i \in [o(j)]$ the degree of $p_i^{(j)}$ is at most $d(|i|, j, i)$.

Construction. The compiler $\mathsf{ARK}[\mathsf{IOP}, \mathsf{PCS}]$ that transforms the polynomial IOP IOP into an interactive argument of knowledge using the polynomial commitment scheme PCS.is almost identical to the one of Chiesa et al. [CHM+20] with minimal syntactical variations.

- The trusted party invokes $crs \leftarrow \mathsf{PCS.S}(1^\lambda)$. Then computes $(pk, vk) \leftarrow \mathsf{PCS.K}(crs, d)$ where the integer d is computed as the maximum degree bound of the polynomial degrees of IOP of indices of size at most N:

  $$d := \max\{d(N, j, i) : j \in [k(|i|)], i \in [o(j)]\}$$

  It publishes the proving key pk and stores the (secret) verification key vk.[26]

- The indexer, given as input the index in a relation R computes the commitment to any polynomials in the 0-th round using no randomness as $C_{0,i} := \mathsf{PCS.C}(pk, p_i^{(0)}; 0)$ for $i \in [o(0)]$. The indexer returns $\rho_0 = (i, C_{0,i}, \ldots, C_{0,o(0)})$.

- The prover $\mathsf{P}(pk, \rho_0, x, w)$ receives as input the proving key pk and the indexer's output ρ_0, the instance x and witness w. The verifier $\mathsf{V}(vk, \rho_0, x)$ receives as input the verification key vk, the indexer's output and the instance.

  At the i-th round, the verifier internally runs the verifier to receive the challenge c_i and sends it to the prover. The prover internally runs the next-message function of the prover $\mathsf{IOP.P}$ obtaining the polynomials $p_i^{(j)}$ for $i \in [o(n)]$. It commits to each of them running $C_j^{(i)} \leftarrow \mathsf{PCS.C}(pk, p_i^{(j)})$. The verifier notifies $\mathsf{IOP.V}$ that the round has finished.

  At the end of the interactive phase, the verifier sends the randomness for the query phase $c_{k(|i|)+1}$. The verifier computes the query set $Q = Q_{\mathsf{IOP.V}}(x; c_1, \ldots, c_{n+1})$ and sends those to the prover. The prover responds with claimed evaluations $z_i^{(j)}$ for each polynomial $p_i^{(j)}$ queries at location $y_i^{(j)}$ from the query set. The verifier sends a batch opening challenge ζ and the prover replies with the batch open procedure using pk, the list of polynomials sent throughout the protocol $(p_i^{(j)})_{j \in [k(|i|)], i \in [o(j)]}$, their respective degree d, the query set Q and the batch opening challenge ζ. The verifier accepts if the verification equation is satisfied and the polynomial verification is correct with respect to $\mathsf{PCS.V}$.

The proof of the following theorems is omitted as mostly identical to the one of Chiesa et al. [CHM+20, Thm. 8.3,8.4].

Theorem 29. *Let* IOP *be a knowledge-sound, polynomial interactive oracle proof over field family* $\mathcal{F}$ *for* R*, and* PCS *be a designated-verifier extractable polynomial commitment scheme over* $\mathcal{F}$*. Then* $\mathsf{ARK}[\mathsf{IOP}, \mathsf{PCS}]$ *is a designated-verifier argument of knowledge for the relation* R*.*

Theorem 30. *Let* IOP *be a* q*-query bounded zero-knowledge polynomial interactive oracle proof for* R *over field family* $\mathcal{F}$*, and* PCS *be a designated-verifier hiding polynomial commitment scheme over* $\mathcal{F}$*. Then* $\mathsf{ARK}[\mathsf{IOP}, \mathsf{PCS}]$ *is a designated-verifier argument of knowledge for the relation* R*. Let* PCS *be a designated-verifier zero-knowledge proof system.*

[25] The 0-th round does not depend on any particular instance or witness, and merely considers the task of encoding a given index i.

[26] In practice, we seek $\mathsf{PCS.K}$ to be generated by the (potentially malicious) verifier. The verifier will then prove that the key is well-formed.

Let $\mathsf{ZKP}_{pcs.k}$ be a designated-verifier zero-knowledge proof system for the relation $\mathsf{R}_{pcs.k} = \{(vk, pk) : (vk, pk) \in [\mathsf{PCS.K}(crs)]\}$. Let $\mathsf{ARK}[\mathsf{IOP}, \mathsf{PCS}, \mathsf{ZKP}_{pcs.k}]$ denote the compiler above where $\mathsf{PCS.K}$ is performed by the verifier, and pk is accompanied by a proof $\pi_{pcs.k}$ that pk lives in the range of the key generation algorithm. Then, we have the following corollary.

Corollary 31. *Let* IOP *be a q-query bounded zero-knowledge polynomial interactive oracle proof for* R *over field family* $\mathcal{F}$*, and* PCS *be a designated-verifier hiding polynomial commitment scheme over* $\mathcal{F}$*, and* $\mathsf{ZKP}_{pcs.k}$ *as above. Then* $\mathsf{ARK}[\mathsf{IOP}, \mathsf{PCS}, \mathsf{ZKP}_{pcs.k}]$ *is a designated-verifier argument of knowledge for the relation* R*.*

Example 1 (A designated-verifier range proof RAP)**.** We consider the protocol sketched by Boneh, Fisch, Gabizon, Williamson[27] to prove that a committed value v is in a range $[0, 2^d]$. Let $H \coloneqq \langle \omega \rangle$ be a subgroup of $\mathbb{F}^\times$ of size d and $Z(x)$ be the polynomial where H vanishes. A common choice here is to select the group of the d-th roots of unity for which the vanishing polynomial can be computed in $\log d$ field operations.

The proof RAP is for a witness $v \in [0, 2^d - 1]$ opening a previously-committed polynomial such that $f(\omega^d) = v$. (The polynomial f is committed using the designated-verifier polynomial commitment scheme; proving that the commitment opens to a signed value boils down to a DLEQ proof.) A common choice is $d = 64$ to prove 64-bit integers.

At a high level, the verifier generates the proving key vk, pk via $\mathsf{dvKZG.S}(1^\lambda, d + 2)$ and then proves in $\pi_{kzg.s}$ knowledge of τ such that:

$$\tau \cdot \left(\sum_{i=0}^{n} \mu_i G_i \right) = \sum_{i=0}^{n-1} \mu^i G_{i+1}$$

where $\mu \coloneqq \mathsf{H}_s(pk)$. Upon downloading the proving key, the prover checks that none of the elements in pk is zero and $\pi_{kzg.s}$ is verified. The prover $\mathsf{RAP.P}$ considers the binary decomposition $(b_0, \ldots, b_{d-1})$ of v and define the polynomial $g(x)$ such that $g(\omega^i) = \sum_{j<i} 2^j b_j$ for $i \in [0, d]$. The polynomial satisfies:

$$\begin{aligned} g(\omega^0) &\in \{0, 1\} \\ g(\omega^d) &= f(\omega^d) \\ g(\omega^{i+1}) - 2g(\omega^i) &\in \{0, 1\} \end{aligned} \tag{27}$$

To provide zero-knowledge for the query bound of this interactive oracle proof, interpolate g on two additional random points $\delta_1, \delta_2 \notin H$ such that $g(\delta_1) = \alpha_1 \leftarrow\!\!\$\, \mathbb{Z}_p$ and $g(\delta_2) = \alpha_2 \leftarrow\!\!\$\, \mathbb{Z}_p$. The prover commits to $g(\boldsymbol{x})$ via $\mathsf{dvKZG.C}$.

The verifier samples $\rho \leftarrow\!\!\$\, \mathbb{Z}_p$ and sends it to the prover.

The prover computes:

$$\begin{aligned} w_1(\boldsymbol{x}) &\coloneqq \frac{Z(\boldsymbol{x})}{\boldsymbol{x}\omega^0} \cdot g(\boldsymbol{x}) \cdot (1 - g(\boldsymbol{x})) \\ w_2(\boldsymbol{x}) &\coloneqq \frac{Z(\boldsymbol{x})}{\boldsymbol{x}\omega^d} \cdot (g(\boldsymbol{x}) - f(\boldsymbol{x})) \\ w_3(\boldsymbol{x}) &\coloneqq \frac{Z(\boldsymbol{x})}{\prod_{i \neq d-1} \omega^i \boldsymbol{x}} (g(\omega\boldsymbol{x}) - 2g(\boldsymbol{x}))(1 - g(\omega\boldsymbol{x}) + 2g(\boldsymbol{x})) \end{aligned}$$

and "batches" them into $w(\boldsymbol{x}) = w_1 + \rho w_2 + \rho^2 w_3$. To show that $Z \mid w$ (which implies equation (27)) it is sufficient to exhibit $q(\boldsymbol{x})$ such that:

$$q(\boldsymbol{x})Z(\boldsymbol{x}) = w(\boldsymbol{x}) \tag{28}$$

$$= \frac{Z(\boldsymbol{x})}{\boldsymbol{x}\omega^0} \cdot g(\boldsymbol{x}) \cdot (1 - g(\boldsymbol{x})) + \frac{Z(\boldsymbol{x})}{\boldsymbol{x}\omega^d} \cdot (g(\boldsymbol{x}) - f(\boldsymbol{x})) + \frac{Z(\boldsymbol{x})}{\prod_{i \neq d-1} \boldsymbol{x} - \omega^i} (g(\omega\boldsymbol{x}) - 2g(\boldsymbol{x}))(1 - g(\omega\boldsymbol{x}) + 2g(\boldsymbol{x})) \tag{29}$$

The prover sends a commitment to $q(\boldsymbol{x})$ to the verifier.

The verifier samples $r \leftarrow\!\!\$\, \mathbb{Z}_p$ and the proves sends openings of $g(\omega r), g(r), \hat{q}(r)$ where:

$$\hat{q}(\boldsymbol{x}) \coloneqq \frac{Z(r)}{r - \omega^d} f(\boldsymbol{x}) + q(\boldsymbol{x})Z(r) \tag{30}$$

The verifier checks the polynomial openings well-formedness of equation (28) Performances of the above range proof and comparisons are displayed in table 4.

[27] Descriptions of the protocols can be found in https://hackmd.io/@dabo/B1U4kx8XI, https://decentralizedthoughts.github.io/2020-03-03-range-proofs-from-polynomial-commitments-reexplained/ and the SoK of Christ et al. [CBC+24, sec. 3.2].

Table 4: Table of range proof sizes for 64-bit integers fit for credentials applications. The column $|\pi|$ indicates the size of the range proof over a 64-bit integer, and in parentheses is the size of the range proof over secp256k1. In Sharp, which involves check of small norm for scalar elements, only concrete size is given as it can't be expressed simply in terms of group elements. Note that batch range proofs will be more performing than the naïve approach. We omit the proving key from the table, which in our case amount to at least $64g$. Timing estimates are based on `zka.lc` [ECK+23].

Scheme	$\|\pi\|$ (64-bits)	P time	V time	Notes
Bulletproofs++ [Eag22]	$10g + 3s$ ($416B$)	4ms	1.8ms	
Sharp [CGKR22]	n/a ($389B$)	1.17ms	0.75ms	Requires fine-tuning on the chosen elliptic curve
HashWires [CCL+21]	n/a (200B)	0.65ms	0.61ms	Relies on a trusted party generating the commitment
This example (RAP)	6g (192B)	1.16ms	0.752ms	Designated-verifier NIZK

8 Building on keyed-verification credential systems

In this section we gather some simple extensions and techniques that could be of interest for future applications.

8.1 Time-based policies

To limit the lifetime of a credential and guarantee safe expiry without key rotation is a desirable feature in many applications. As a warm-up example, we study the case of credentials subject to time-based policies such as expiry.

Let KVC be a keyed-verification credential over message family the integers modulo p satisfying $64 < \lceil \log p \rceil$, and assume time is represented in unix time as a 64-bit signed integer. During issuance, server and user agree on a timestamp t and run the issuance protocol with attributes $(\vec{m}, t)$. At presentation time, the user proves that the credential is not yet expired (that is, $t < e$ where e denotes the expiry time) via:

$$\phi^{e}_{\text{exp}}(\vec{m}) \coloneqq \big((e - t) \mod p \in \{0, \ldots, 2^{64} - 1\}\big)$$

where t denotes the timestamp attribute and e denotes the expiry timestamp, after which credentials are to be considered stale. Christ et al. [CBC+24] present and detail state-of-art range proofs and the trade-offs between them. In example 1 we present an example range proof.

No adversary in possession of an expired credential (that is, with attribute $t > e$) can have a successful presentation for the above statement as long as $\log p > 64$. Any two users with valid attributes for which ϕ_{exp} holds are indistinguishable to the server, by anonymity (which in turn implies indistinguishability of two credentials satisfying the same predicate)

Therefore, any keyed-verification credential system supporting $n > 1$, attribute space $\mathbb{M} = \mathbb{Z}_p$ with $\lceil \log p \rceil > 64$ and a predicate space Φ sufficiently expressive, can be used to implement the above feature.

Remark 7. Other ways are possible to enforce expiry of a credential. One of them is to partition expiration into time frames. When issuing a credential, the user is associated to an expiration time frame and their credential embedded with it. This approach, while simpler, entails a more complex trade-off with anonymity as it may fraction the anonymity set into users per time-frame.

8.2 Rate-limiting

In keyed-verification credentials, authorizations can be presented an indefinite number of times while preserving privacy. A valid specialization of the authorization case is to control the limited use of some finite resource, while preserving anonymity of the users. This problem is treated in *k-times anonymous authentication* [CHK+06, TFS04, NS04], a credential system where users, a group manager (in charge of distributing credentials), and some application providers (in charge of moderating accesses) enforce rate-limiting access to users.

Here we work in a simple scenario composed only of a server and a user, without revocation to target a simple and efficient rate-limiting system similar to Privacy Pass [DGS+18]. In this model, we don't consider revocation, nor credential hijacking. At a high level, the approach here is to have the user get a credential for a PRF key k and evaluate the PRF over a counter to generate pseudorandom "tokens" that are unlinkable. The server maintains a list of "spent tokens" and enforces double-spending by ensuring no token is ever re-used for the same scope. The user, to present a valid token, proves the PRF has been correctly evaluated and that its input is "small".

8.2.1 Syntax

A rate-limiting anonymous token $\mathsf{RTL} = (\mathsf{S}, \mathsf{K}, \mathsf{I}, \mathsf{P})$ is composed of:

Game $\mathrm{UNF}_{\mathsf{RTL},\mathsf{A}}(\lambda,\ell)$

$q \coloneqq 0;\ ctr \coloneqq 0;\ Qrs \coloneqq [\,]$
$crs \leftarrow \mathsf{RTL.S}(1^\lambda)$
$(sk, pp) \leftarrow \mathsf{RTL.K}(crs)$
$(scp^*, (t_i^*, \rho_i^*)_{i=1}^{\ell q+1}) \leftarrow \mathsf{A}^{\textsc{Issue},\textsc{Present},\textsc{NewUsr},\textsc{PresentUsr}}(pp)$
return $\forall i\colon \mathsf{RTL.P.Srv}(sk, \ell, scp^*, t_i^*, \rho_i^*) = 1$
$\forall i \neq j\colon t_i \neq t_j \ \wedge$
$\forall i\colon (t_i^*, \rho_i^*) \notin Qrs$

Game $\mathrm{UNLINK}^b_{\mathsf{RTL},\mathsf{A},\mathsf{D}}(\lambda,\ell)$

$crs \leftarrow \mathsf{RTL.S}(crs)$
$Ctr \coloneqq [\,]$
$(pp, st_{\mathsf{A}}) \leftarrow \mathsf{A}(crs)$
$(\sigma_0; st'_{\mathsf{A}}) \leftarrow (\mathsf{RTL.I.Usr}(pp) \rightleftharpoons \mathsf{A}(st_{\mathsf{A}}))$
$(\sigma_1; st_{\mathsf{D}}) \leftarrow (\mathsf{RTL.I.Usr}(pp) \rightleftharpoons \mathsf{A}(st'_{\mathsf{A}}))$
$b' \leftarrow \mathsf{D}^{\textsc{Get},\textsc{Chal}}(st_{\mathsf{D}})$
return $b' = 1$

Oracle $\textsc{NewUsr}()$

$\hat{\sigma} \coloneqq \mathsf{RTL.M}(sk)$
$Usrs[ctr] \coloneqq \hat{\sigma}$
$ctr \coloneqq ctr + 1$
return ctr

Oracle $\textsc{PresentUsr}(i, scp)$

$(t, \rho) \leftarrow \mathsf{RTL.P.Usr}(pp, Usrs[i], \ell, scp, i)$
$Qrs \coloneqq Qrs \cup \{(t, \rho)\}$

Oracle $\textsc{Get}(\beta, scp)$

$Ctr[scp][\beta] \coloneqq Ctr[scp][\beta] + 1$ **or** 1
if $Ctr[scp][0] > \ell \ \vee\ Ctr[scp][1] > \ell$ **: return** $\perp$
return $\mathsf{RTL.P.Usr}(sk, \hat{\sigma}_\beta, (\ell, scp, Ctr[scp][\beta]))$

Oracle $\textsc{Chal}(scp)$

return $(\textsc{Get}(b, scp), \textsc{Get}(1-b, scp))$

Oracle $\textsc{Issue}(\mu)$

$q \coloneqq q + 1$
return $\mathsf{RTL.I.Srv}(sk, \mu)$

Oracle $\textsc{Present}(scp, t, \rho)$

return $\mathsf{RTL.P.Srv}(sk, \ell, scp, t, \rho)$

Figure 7: Unforgeability and unlinkability game for a rate-limiting anonymous token RTL. The adversary of unlinkability is assumed to be stateful and to preserve state across calls.

- $crs \leftarrow \mathsf{RTL.S}(1^\lambda)$ the setup algorithm, which generates a common reference string.
- $(sk, pp) \leftarrow \mathsf{RTL.K}(crs)$ the key generation algorithm, which produces a secret key sk and some public parameters pp.
- $\hat{\sigma} \leftarrow (\mathsf{RTL.I.Usr}(pp) \rightleftharpoons \mathsf{RTL.I.Srv}(sk))$ the issuance algorithm, which produces a credential $\hat{\sigma}$. We consider the procedure to be non-interactive, that is, we consider the following 3 algorithms:
  - $(\mu, st) \leftarrow \mathsf{RTL.I.Usr}_1(pp)$ the first part of the issuance algorithm, returning μ and some user state st.
  - $\sigma' \leftarrow \mathsf{RTL.I.Srv}(sk, \mu)$, the server signing algorithms, which returns a blinded credential σ'. If the input was malformed, $\sigma' = \perp$.
  - $\hat{\sigma} \leftarrow \mathsf{RTL.I.Usr}_2(st, \sigma')$ the second part of the user issuance algorithm, which returns the credential $\hat{\sigma}$.
- $0/1 \leftarrow (\mathsf{RTL.P.Usr}(pp, \hat{\sigma}, (\ell, scp, i)) \rightleftharpoons \mathsf{RTL.P.Srv}(sk, \ell, scp))$ the presentation algorithm, which takes as input a scope $scp \in \{0,1\}^*$ identifying the resource and (for the user) a counter i for generating the i-th token. It returns a bit (and updates the server state). We assume presentation to be non-interactive and thus consider the following two algorithms:
  - $(t, \rho) \leftarrow \mathsf{RTL.P.Usr}(pp, \sigma, (\ell, scp, i))$ the user presentation algorithm, which returns a pair (t, ρ) where t identifies the token.
  - $0/1 \leftarrow \mathsf{RTL.P.Srv}(sk, (\ell, scp, t), \rho)$ the server presentation algorithm, which returns 1 if the token (t, ρ) is considered valid and for at most ℓ accesses and 0 otherwise.

Similarly to the case of keyed-verification credentials, we will consider the procedure $\mathsf{RTL.M}(sk)$ that generates a fresh credential using sk interacting with a mock user. This is syntactic sugar of a server emulating the user and returning its result.

A rate-limiting scheme must be *correct*, *unforgeable*, and *unlinkable*. Correctness means that for every $crs \in [\mathsf{RTL.S}(1^\lambda)]$, $(sk, pp) \leftarrow \mathsf{RTL.K}(crs)$, $\ell > 0$ and $0 \le i < \ell$, $scp \in \{0,1\}^*$

$$\Pr\left[b = 1\colon \begin{array}{l} \hat{\sigma} \leftarrow (\mathsf{RTL.I.Usr}(pp) \rightleftharpoons \mathsf{RTL.I.Srv}(sk)) \\ b \leftarrow (\mathsf{RTL.P.Usr}(pp, \hat{\sigma}, (\ell, scp, i)) \rightleftharpoons \mathsf{RTL.P.Srv}(sk, (\ell, scp))) \end{array}\right] = 1\ .$$

A rate-limiting anonymous token is one-more unforgeable if for every $\ell \ge 0$, and any adversary A, the advantage $\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{RTL},\mathsf{A}}(\lambda) \coloneqq \Pr[\mathrm{OMUF}_{\mathsf{RTL},\mathsf{A}}(\lambda,\ell) = 1]$ (defined in figure 7) is negligible in λ. A rate-limiting anonymous token is unlinkable if for any $\ell > 0$ and (stateful) adversary A, the advantage $\mathsf{Adv}^{\mathrm{unlink}}_{\mathsf{RTL},\mathsf{A}}(\lambda,\ell)$ defined in figure 7 is negligible in λ.

8.2.2 Our compiler

Let $\phi^{\ell}_{\text{range}}$ denote the predicate that checks if the input is in the range $[0, \ell - 1]$. Let $\mathsf{PRF}(k, (x, y))$ denote a pseudo-random function (PRF) evaluation on key k and message (x, y). The message is presented as a pair so to make it easy to prove statements about parts of the rate-limiting input. A PRF is **compatible** with the keyed-verification credential KVC if the attribute space of the credential system is equal to the key space of the PRF, and $\phi^{m,e}_{\text{prf}}(k) \coloneqq (\mathsf{PRF}(k, m) = e)$ is part of the predicate family. The more general case is easy to derive from this one.

We describe a compiler that, given a keyed-verification credential KVC and a pseudorandom function PRF compatible with KVC, produces a rate-limiting scheme. We denote such system as $\mathsf{cRTL}[\mathsf{KVC}, \mathsf{PRF}]$ and define it as follows:

- $\mathsf{cRTL.S}(1^{\lambda})$, the setup algorithm runs $crs \leftarrow \mathsf{KVC.S}(1^{\lambda}, 1)$ and returns crs.
- $\mathsf{cRTL.K}(crs)$, the key generation algorithm returns $(sk, pp) \leftarrow \mathsf{KVC.K}(crs)$.
- $(\mathsf{cRTL.I.Usr}(pp) \rightleftharpoons \mathsf{cRTL.I.Srv}(sk))$, the credential issuance algorithm has the server run $\mathsf{KVC.I.Srv}(sk, st, \phi_{[1]})$, while the user algorithm samples a random $k \leftarrow\!\!\$\ \mathbb{M}$ and runs $\mathsf{KVC.I.Usr}(pp, k, \phi_{[1]})$. At the end of the interaction, the user obtains a credential σ and returns the pair $\hat{\sigma} \coloneqq (k, \sigma)$.
- $(\mathsf{cRTL.P.Usr}(pp, \hat{\sigma} = (k, \sigma), (\ell, scp, i)) \rightleftharpoons \mathsf{cRTL.P.Srv}(sk, (\ell, scp)))$, the credential presentation algorithm has the server run $\mathsf{KVC.P.Srv}(sk, \phi^{scp,t}_{\text{prf}} \ \wedge\ \phi^{\ell}_{\text{range}})$, while the user runs $\mathsf{KVC.P.Usr}(pp, k, \sigma, \phi^{scp,t}_{\text{prf}} \ \wedge\ \phi^{\ell}_{\text{range}})$.

The next section is devoted to proving the following:

Theorem 32. *Let $\ell > 0$, and $\Phi \supset \{\phi_{\text{prf}}, \phi_{\text{range}}\}$. If KVC be a keyed-verification credential for $n = 1$ attributes and predicate family Φ, and PRF is a pseudorandom function compatible with KVC, then $\mathsf{cRTL}[\mathsf{KVC}, \mathsf{PRF}]$ is a rate-limiting anonymous token with one-more unforgeability advantage:*

$$\mathsf{Adv}^{\text{omuf}}_{\mathsf{cRTL}[\mathsf{KVC},\mathsf{PRF}]}(\lambda, \ell) \leq (q\ell + 1)\mathsf{Adv}^{\text{ext}}_{\mathsf{KVC}}(\lambda, 1) + q\mathsf{Adv}^{\text{anon}}_{\mathsf{KVC}}(\lambda, 1) + q\mathsf{Adv}^{\text{prnd}}_{\mathsf{PRF}}(\lambda) \ ,$$

(where q is the number of queries to the issuance oracles) and unlinkability advantage:

$$\mathsf{Adv}^{\text{unlink}}_{\mathsf{cRTL}[\mathsf{KVC},\mathsf{PRF}]}(\lambda, \ell) \leq 2(\mathsf{Adv}^{\text{anon}}_{\mathsf{KVC}}(\lambda, 1) + \mathsf{Adv}^{\text{prnd}}_{\mathsf{PRF}}(\lambda)) \ .$$

Remark 8. The restriction on the PRF being compatible with KVC may be lifted, provided that the user proves at issuance time well-formedness of the PRF key.

Remark 9. In the proof, extraction is required solely to carry out a counting argument after making sure that all tokes are evaluated from keys issued to the user. In fact, for the credentials $\mu\mathsf{CMZ}$ and $\mu\mathsf{BBS}$ it is still possible to extract from the zero-knowledge proofs sent at presentation time and carry out a reduction to one-more unforgeability.

Corollary 33. *Let $\ell > 0$, and $\Phi \supset \{\phi_{\text{prf}}, \phi_{\text{range}}\}$. Consider a one-more unforgeable keyed-verification credential $\mathsf{KVC}_{AT} \in \{\mu\mathsf{CMZ}_{AT}, \mu\mathsf{BBS}_{AT}\}$, for attribute damily Φ and let PRF be a pseudorandom function compatible with KVC_{AT}, then cRTL is a rate-limiting anonymous token with one-more unforgeability advantage:*

$$\mathsf{Adv}^{\text{omuf}}_{\mathsf{cRTL}[\mathsf{KVC}_{AT}[\mathsf{ZKP}],\mathsf{PRF}]}(\lambda, \ell) \leq \mathsf{Adv}^{\text{omuf}}_{\mathsf{KVC}_{AT}}(\lambda, 1) + (q\ell + 1)\mathsf{Adv}^{\text{ksnd}}_{\mathsf{ZKP}}(\lambda) + q\mathsf{Adv}^{\text{anon}}_{\mathsf{KVC}_{AT}}(\lambda, 1) + q\mathsf{Adv}^{\text{prnd}}_{\mathsf{PRF}}(\lambda) \ ,$$

and unlinkability advantage:

$$\mathsf{Adv}^{\text{unlink}}_{\mathsf{cRTL}[\mathsf{KVC}_{AT},\mathsf{PRF}]}(\lambda, \ell) \leq 2(\mathsf{Adv}^{\text{anon}}_{\mathsf{KVC}_{AT}}(\lambda, 1) + \mathsf{Adv}^{\text{prnd}}_{\mathsf{PRF}}(\lambda)) \ ,$$

where q is the number of oracle queries made by the adversary.

8.2.3 Analysis

Correctness follows by inspection; unlinkability follows directly from pseudorandomness and anonymity. We focus on unforgeability.

Lemma 34. *Let $\ell > 0$, and $\Phi \supset \{\phi_{\text{prf}}, \phi_{\text{range}}\}$. If KVC be a keyed-verification credential for $n = 1$ attributes and predicate family Φ, and PRF is a PRF, then $\mathsf{cRTL}[\mathsf{KVC}, \mathsf{PRF}]$ has one-more unforgeability advantage:*

$$\mathsf{Adv}^{\text{omuf}}_{\mathsf{cRTL}[\mathsf{KVC},\mathsf{PRF}]}(\lambda, \ell) \leq (q\ell + 1)\mathsf{Adv}^{\text{ext}}_{\mathsf{KVC}}(\lambda, 1) + q_u\mathsf{Adv}^{\text{anon}}_{\mathsf{KVC}}(\lambda, 1) + q_u\mathsf{Adv}^{\text{prnd}}_{\mathsf{PRF}}(\lambda) \ ,$$

where q is the number of queries to Issue *and q_u is the number of queries to* NewUsr.

Proof. Consider an adversary A for $\text{OMUF}_{\mathsf{cRTL}[\mathsf{KVC},\mathsf{PRF}]}(\lambda)$. A receives as input pp and during the execution has access to oracles:

– Issue, to issue a credential to the adversary;

– PRESENT, to present a credential to the adversary;
– NEWUSR, and PRESENTUSR, to respectively create new honest users, and have them produce a new presentation token.

We build a reduction B for extractability of KVC. B receives as input pp and initializes a list $Keys = []$. It runs internally $\mathsf{A}(pp)$, responding to each oracle query as follows:

- each issuance query with input μ is responded with $\textsc{Issue}(\mu, \phi_{[1]})$ ($\textsc{Issue}$ being the issuance oracle of the extractability game);
- each present query with input (scp, t, ρ) is responded with $\textsc{Present}(\phi^{scp,t}_{\text{prf}} \wedge \phi^{\ell}_{\text{range}}, \rho)$;
- each new user query is responded by sampling $k \leftarrow\!\$\, \mathbb{M}$, appending it to $Keys$, and querying $\textsc{NewUsr}(k)$;
- each user presentation query with input (i, scp) has the adversary B compute $t \coloneqq \mathsf{PRF}(Keys[i], scp)$ and respond with the answer of $\textsc{PresentUsr}(i, \phi^{scp,t}_{\text{prf}} \wedge \phi^{\ell}_{\text{range}})$.

At the end of the execution, A outputs valid (for scp^*) forgeries $(t^*_i, \rho^*_i)_i^{\ell q+1}$ such that $t^*_i \neq t^*_j$ for all $i \neq j$ and (t^*_i, ρ^*_i) was not previously queried. We distinguish the following events:

- $\exists i \in [\ell q + 1]\colon t^*_i \neq \mathsf{PRF}(k^*_i, scp^*, j^*_i)$ (for some $j^*_i \in [0, \ell - 1]$), in which case extraction failed, and the predicate does not hold. This can be shown with a reduction to the extractability game, where the reduction at the end guesses the presentation message of the adversary output that constitutes a valid forgery.
- $\exists i \in [\ell q + 1]\colon k^*_i \in Keys$, which means that a forgery was made from an honest user. However, it can be shown that since the PRF evaluations are pseudorandom (and independent of the keys), satisfying this event is equivalent to guessing the PRF key of a user. This can be shown with a hybrid argument, first replacing the honest user issuance and presentation sessions with simulated ones, and then replacing the PRF evaluations with random ones. At the end of both changes, the $\textsc{NewUsr}$ and $\textsc{PresentUsr}$ oracles do not create any PRF key, nor credentials. The resulting proofs are not part of the (winning) adversary output (as they are not part of $PQrs$).

By the pigeonhole principle on $\mathsf{PRF}(\cdot, scp, \cdot)$, where the first argument is Qrs (the set of issued credentials in the extractability game, of size q) and the last argument is in $[0, \ell - 1]$ (of size ℓ), it must be that $\exists i, j \in [\ell q + 1]\colon k^*_i = k^*_j$ (which contradicts one of the winning conditions) and so the adversary can never win the game in this case. □

8.2.4 Instantiation

We describe a variation of the Dodis–Yampolskiy's PRF [DY05] that makes it easy to reason and prove statements on the counter. The PRF is defined as:

$$\mathsf{HashDY}(k, (scp, ctr)) \coloneqq (k + ctr)^{-1} \cdot \mathsf{H}_g(scp)$$

for $ctr \in \mathbb{Z}_p$ and $ctr \leq \ell = \mathsf{poly}(\lambda)$ and $\mathsf{H}_g : \{0,1\}^* \to \mathbb{G}$ a hash that maps elements in the group. In this case, a "token" is $T \in \mathbb{G}$ (indicated here in uppercase to match the group description). Note that the predicate:

$$\phi^{scp,T,\ell}_{\text{prf}}(\vec{m}, k, i) \coloneqq ((k + i)T = \mathsf{H}_g(scp) \ \wedge \ 0 \leq i < \ell)$$

can be proven easily within Schnorr proofs, with the help of a range proof. We prove that HashDY is a PRF.

Theorem 35. *In the random oracle model, if q-DDHI holds over* GrGen*, then* HashDY *is pseudorandom with advantage:*

$$\mathsf{Adv}^{\text{prnd}}_{\mathsf{HashDY},\mathsf{A}}(\lambda) \leq q\mathsf{Adv}^{\text{q-ddhi}}_{\mathsf{GrGen}}(\lambda) + \frac{q^2}{p} \ ,$$

where q is the number of evaluation queries.

Proof. Let A be a p.p.t. adversary for pseudorandomness. A has access to an oracle $\textsc{Eval}$ for arbitrary $scp \in \{0,1\}^*$ and counters $0 \leq i < \ell$. Consider a hybrid argument where, in the i-th step, we replace the $\textsc{Eval}$ responses up to the i-th query with uniformly random group elements. Consider an adversary A_i able to distinguish the i-th hybrid. We construct an adversary B_i for the i-th hybrid able to distinguish every time that the adversary A_i's output is different during the i-th hybrid change.

B_i receives as input a group description Γ and a q-DDHI challenge $\vec{G} = (G, \tau G, \tau^2 G, \ldots, \tau^q G)$ and $P \in \mathbb{G}$ (either $P = 1/\tau \cdot G$ in $q\text{-}\mathrm{DDHI}^0_{\mathsf{A},\mathsf{GrGen}}(\lambda)$, or a uniformly distributed random element $P \leftarrow\!\$\, \mathbb{G}$ in $q\text{-}\mathrm{DDHI}^1_{\mathsf{A},\mathsf{GrGen}}(\lambda)$), with $q \leq \ell$. Let $\alpha \coloneqq \tau - i$ and define $\vec{A} \coloneqq (G, \alpha G, \alpha^2 G, \ldots, \alpha^q G)$. These can be computed using the binomial theorem using $\vec{G}$ only. Once computed, B internally runs the adversary A using the crs received as input. During its execution, the adversary A query to H_g and $\textsc{Eval}$:

Game $\mathrm{ANON}^b_{\mathsf{NYM},\mathsf{A},\mathsf{D}}(\lambda)$	Oracle $\textsc{Get}(scp, m)$	Oracle $\textsc{Chal}(scp, m)$
$crs \leftarrow \mathsf{NYM.S}(\lambda)$	$Qrs \coloneqq Qrs \cup \{(scp, m)\}$	**if** $(scp, m) \in Qrs$: **return** $\perp$
$(pp, st_{\mathsf{A}}) \leftarrow \mathsf{A}(crs)$	**for** $\beta \in \{0, 1\}$	$(\alpha_0, \alpha_1) \leftarrow \textsc{Get}(scp, m)$
$(\sigma_0; st'_{\mathsf{A}}) \leftarrow (\mathsf{NYM.I.Usr}(pp) \rightleftharpoons \mathsf{A}(st_{\mathsf{A}}))$	$nym_\beta \coloneqq \mathsf{NYM.E}(pp, k_\beta, scp)$	**return** (α_b, α_{1-b})
$(\sigma_1; st_{\mathsf{D}}) \leftarrow (\mathsf{NYM.I.Usr}(pp) \rightleftharpoons \mathsf{A}(st_{\mathsf{A}}))$	$\rho_\beta \leftarrow \mathsf{NYM.P.Usr}(pp, (k_\beta, \sigma_\beta), (scp, nym_\beta, m))$	
$b' \leftarrow \mathsf{D}^{\textsc{Get},\textsc{Chal}}(st_{\mathsf{D}})$	**return** $((nym_0, \rho_0), (nym_1, \rho_1))$	
return b'		

Figure 8: Anonymity game for a keyed-verification pseudonym system NYM.

- Upon receiving the j-th query to the random oracle H_g of the form $\mathsf{H}_g(scp_j)$ such that $scp_j \in \{0,1\}^*$, B checks if the element was previously queried. If that is not the case, B computes the polynomial $f(x) \coloneqq \prod_{\substack{\iota=1 \\ \iota \neq i}}^{\ell}(x + \iota) = \sum_k f_k x^k$ and samples $\theta_j \leftarrow\!\!\$\ \mathbb{Z}_p$. Finally, it computes $T_j = \theta_j f(\alpha) G = \theta_j \cdot \sum_k f_k A_k$ and stores in a table (scp_j, θ_j, T_j) and returns to the user T_j. If such a query was already made, B fetches T_j from its records and returns it the same value as before.
- Upon receiving the j-th $\textsc{Eval}$ query of the form (scp_j, ι_j) from the adversary A, the reduction B checks if scp_j was already queried in H_g and, if not, proceeds doing so. Then, it checks if ι_j is equal to i. If it is not equal, it retrieves the record (scp_j, θ_j, T_j) and returns $(\tau + \iota_j)^{-1} \cdot T_j$, computed via $h(x) \coloneqq \prod_{\iota \neq i, \iota_j}(x + \iota)$ and returning $\theta_j \cdot \sum_k h_k G_k$. If $\iota_j = i$, then by euclidean remainder we can write $f(x) = h(x)(x + i) + r$ where $r = f(i)$. Therefore, $(\alpha + i)^{-1} \cdot \theta_j \cdot f(\alpha) \cdot G = \theta_j \cdot (h(\alpha) + r/(\alpha + i))\, G = \theta_j(\sum_k h_k A_k + f(i)P)$.

The output distribution of H_g is uniformly random except with statistically negligible probability (when α is a root of f). n the case $q\text{-}\mathrm{DDHI}^0_{\mathsf{A},\mathsf{GrGen}}(\lambda)$ the output distribution of $\textsc{Eval}$ is identically distributed, while it is uniformly distributed in $q\text{-}\mathrm{DDHI}^1_{\mathsf{A},\mathsf{GrGen}}(\lambda)$. (Recall that $P = 1/\tau G = 1/(\alpha + i)G$ in $q\text{-}\mathrm{DDHI}^0_{\mathsf{A},\mathsf{GrGen}}(\lambda)$ and $A_i = \alpha^i G$.) Indistinguishability then is straightforward by the distance lemma over each of the hybrid changes. □

8.2.5 Other methods for rate-limiting

We briefly mention below other approaches that may be used to enforce rate limiting.

Grinding. Simpler methods for rate-limiting are folklore, such as providing a proof of work associated with the request. This technique can be embedded in the rate-limiting token part t: roughly speaking, upon presentation the token t must also satisfy that t is "small". This is sometimes referred to as *grinding* in the literature and may be implemented appending a random nonce to the scope, having then the user attempt to find a nonce such that the PRF output is small.

Batch issuance of spend-once credentials. Another common approach is to issue multiple spend-once credentials: the user sets (at issuance time) a blind attribute to be a nonce, and fully discloses it to the server upon presentation. The server, on the other hand, keeps a list of previously-spend nonces and enforces rate-limiting by adjusting the number of spend-once credentials. This approach naïvely incurs in a n-times communication and computation overhead in the issuance phase for limiting n-accesses per-user, but classical batching techniques may be available to partially reduce this cost.

8.3 Pseudonyms

In some cases, the server might require the user to adopt an identity for a specific resource denoted $scp \in \{0,1\}^*$, and have that identity be otherwise unlinkable. Such pseudonym can be used for blocking and logging in users in a specific service seamlessly (that is, without making requests to third-party services or requiring another registration process.). The scope $scp \in \{0,1\}^*$ identifies the scope to be accessed, but one may also make other valid choices. For instance, a timestamp can restrict access over time, a unique random nonce for spend-once credentials, the identifier of a group chat to select a group-dependent pseudonym, etc.

8.3.1 Syntax

A keyed-verification pseudonym system NYM consists of the following efficient procedures:

- $crs \leftarrow \mathsf{NYM.S}(1^\lambda)$, the setup algorithm, getting as input the security parameter in unary form and returning a common reference string crs;

Game $\mathrm{UNF}_{\mathsf{NYM},\mathsf{A}}(\lambda)$

$PQrs := []$; $Creds := []$; $Keys := []$; $q := 0$

$crs \leftarrow \mathsf{NYM.S}(1^\lambda)$

$(sk, pp) \leftarrow \mathsf{NYM.K}(crs)$

$(scp^*, (nym_i^*, m_i^*, \rho_i^*)_{i=1}^n) \leftarrow \mathsf{A}^{\textsc{Issue},\textsc{Present},\textsc{NewUsr},\textsc{PresentUsr}}(pp)$

return $\forall i \in [n]\colon \mathsf{NYM.P.Srv}(sk, (scp_i^*, nym_i^*, m_i^*), \rho_i) = 1 \ \wedge$
$(scp^*, nym_i^*, m_i^*) \notin PQrs \ \wedge$
$\Big((\forall i \neq j\colon nym_i^* \neq nym_j^* \ \wedge \ n > q) \ \vee$
$\exists i, j\colon nym_i^* = \mathsf{NYM.E}(pp, Keys[j], scp^*)\Big)$

Oracle $\textsc{Issue}(\mu)$

$q := q + 1$

return $\mathsf{NYM.I.Srv}(sk, \mu)$

Oracle $\textsc{Present}((scp, nym, m), \rho)$

return $\mathsf{NYM.P.Srv}(sk, (scp, nym, m), \rho)$

Oracle $\textsc{Eval}(i, scp)$

return $\mathsf{NYM.E}(pp, Keys[i], scp)$

Oracle $\textsc{NewUsr}()$

$k \leftarrow\!\!\$\ \mathbb{M}$

$\sigma \leftarrow \mathsf{NYM.M}(sk, k)$

$(Keys[q], Creds[q]) := (k, \sigma)$

return 1

Oracle $\textsc{PresentUsr}(i, scp, m)$

$nym := \mathsf{NYM.E}(Keys[i], scp)$

$PQrs := PQrs \cup \{(i, scp, nym, m)\}$

$\rho \leftarrow \mathsf{NYM.P.Usr}(pp, (scp, nym, m))$

return (nym, ρ)

Figure 9: Unforgeability for a keyed-verification pseudonym system NYM.

- $(sk, pp) \leftarrow \mathsf{NYM.K}(crs)$, the key-generation algorithm produces a signing key sk together with some public parameters pp;
- $(k, \sigma) \leftarrow (\mathsf{NYM.I.Usr}(pp) \rightleftharpoons \mathsf{NYM.I.Srv}(sk))$, the issuance algorithm, producing an identity key k and a credential σ over it. We consider the issuance to be non-interactive, that is, we consider the following 3 algorithms:
  - $(\mu, st) \leftarrow \mathsf{NYM.I.Usr}_1(pp)$ the first part of the issuance algorithm, returning μ and some user state st.
  - $\sigma' \leftarrow \mathsf{NYM.I.Srv}(sk, \mu)$, the server signing algorithms, which returns a blinded credential σ'. If the input was malformed, $\sigma' = \bot$.
  - $(k, \sigma) \leftarrow \mathsf{NYM.I.Usr}_2(st, \sigma')$ the second part of the user issuance algorithm, which returns the identity key k and credential σ.
- $nym := \mathsf{NYM.E}(pp, k, scp)$ the evaluation algorithm generates a pseudonym nym for the scope $scp \in \{0,1\}^*$.
- $0/1 \leftarrow (\mathsf{NYM.P.Srv}(sk, (scp, nym, m)) \rightleftharpoons \mathsf{NYM.P.Usr}(pp, (k, \sigma), (scp, nym, m)))$ the presentation algorithm, which authenticates as message m for an authorized identity nym within the scope scp, using the identity key k and credential σ. Since we consider non-interactive presentations, we can simplify the above as:
  - $\mu \leftarrow \mathsf{NYM.P.Usr}(pp, (k, \sigma), (scp, nym, m))$,
  - $0/1 \leftarrow \mathsf{NYM}(sk, (scp, nym, m), \mu)$.

The core security requirements of a pseudonym system are correctness (all honestly-generated nyms verify), unforgeability (that is, a user cannot generate multiple identities for the same scope or act of behalf of another user), and anonymity (that is, two users are indistinguishable across different scopes). More formally, correctness requires that, for all $scp, m \in \{0,1\}^*$:

$$\Pr\left[b = 1\colon \begin{array}{l} crs \leftarrow \mathsf{NYM.S}(1^\lambda) \\ (sk, pp) \leftarrow \mathsf{NYM.K}(crs) \\ (k, \sigma) \leftarrow (\mathsf{NYM.I.Usr}(pp) \rightleftharpoons \mathsf{NYM.I.Srv}(sk)) \\ nym := \mathsf{NYM.E}(pp, k, scp) \\ b \leftarrow (\mathsf{NYM.P.Srv}(sk, (scp, nym, m)) \rightleftharpoons \mathsf{NYM.P.Usr}(pp, (k, \sigma), (scp, nym, m))) \end{array}\right]$$

is overwhelming in λ. Unforgeability asks that $\mathsf{Adv}^{\mathrm{unf}}_{\mathsf{NYM},\mathsf{A}}(\lambda) := \Pr\left[\mathrm{UNF}_{\mathsf{NYM},\mathsf{A}}(\lambda) = 1\right]$ is negligible in λ, where $\mathrm{UNF}_{\mathsf{NYM},\mathsf{A}}(\lambda)$ is defined in figure 9. Anonymity requires that $\mathsf{Adv}^{\mathrm{anon}}_{\mathsf{NYM},\mathsf{A}}(\lambda)$ is negligible in λ, where the distinguishing game $\mathrm{ANON}^b_{\mathsf{NYM},\mathsf{A}}(\lambda)$ is illustrated in figure 8.

Remark 10 (Differences with VRFs.)**.** It is possible to re-state those properties in terms of verifiable random functions, but the syntax here is slightly different. Verifiable random functions have a verification algorithm that relies on a "public key" (which here is not used) and require *uniqueness*, that is, it is not possible to prove two different VRF outputs are valid for the same input and the same secret key. Pseudonym evaluation is a deterministic algorithm, and therefore uniqueness perfectly holds.

8.3.2 Our compiler

We say that a keyed-verification credential KVC supports **labelled presentation** if the presentation message of the user can be labeled. This is denoted as a "dummy predicate" ϕ^m_{lbl}. For the credentials we present, this boils down to demading ZKP to supporting labelling, e.g. with a message embedded in the Fiat–Shamir transformation.

As in the previous section, we say that a pseudorandom function PRF is **compatible** with a keyed-verification credential KVC if the attribute space of the credential system is equal to the key space of the PRF and $\phi^{m,e}_{\text{prf}}$ is in the predicate family of KVC.

We say that a keyed-verification credential KVC supports i**-rerandomizable issuance** if there exists an issuance protocol $\mathsf{KVC.I}_{rnd}$ which, as long as at least one of the two participants (user or server) is honest, the credential at the end of the protocol is such that the i-th attribute is uniformly distributed over $\mathbb{M}$.

Let KVC be a keyed-verification credential for $n = 1$ attributes supporting re-randomizable issuance and labels. Let PRF be a pseudorandom function compatible with KVC. We describe a compiler $\mathsf{cNYM}[\mathsf{KVC}, \mathsf{PRF}]$ and define it as follows:

- $crs \leftarrow \mathsf{cNYM.S}(1^\lambda)$, the setup algorithm internally runs $\mathsf{KVC.S}(1^\lambda, 1)$ and return its common reference string;
- $(sk, pp) \leftarrow \mathsf{cNYM.K}(crs)$ returns the output of the keyed-verification credential key generation;
- $(k, \sigma) \leftarrow (\mathsf{cNYM.I}(pp) \rightleftharpoons \mathsf{cNYM.I.Srv}(sk))$, the issuance algorithm has the user run $\mathsf{KVC.I}_{rnd}.\mathsf{Usr}(pp, [], \phi_\emptyset)$ and the server $\mathsf{KVC.I}_{rnd}.\mathsf{Srv}(sk, \phi_\emptyset)$. At the end of the protocol execution, the user gets a uniformly-distributed attribute k and a credential σ on it.
- $nym \coloneqq \mathsf{cNYM.E}(pp, k, scp)$, the pseudonym evaluation function returns $nym \coloneqq \mathsf{PRF}(k, scp)$.
- $0/1 \leftarrow (\mathsf{cNYM.P.Usr}(pp, (k, \sigma), (scp, nym, m) \rightleftharpoons \mathsf{cNYM.P.Srv}(sk, (scp, nym, m)))$, the presentation algorithm has the user receive as input a pseudonym $nym \coloneqq \mathsf{PRF}(k, scp)$ and a message m to be authenticated. It runs $\mathsf{KVC.P.Usr}(pp, k, \sigma, \phi^{scp,nym}_{\text{prf}} \wedge \phi^m_{\text{lbl}})$, while the server responds with whatever the underlying protocol $\mathsf{KVC.P.Srv}(sk, (scp, nym, m))$ returns.

Theorem 36. *Let* KVC *be a keyed-verification credential for* $n = 1$ *attributes, with re-randomizable issuance and labelled presentation, for predicate family* $\Phi \supset \{\phi_{\text{prf}}\}$*. Let* PRF *be a pseudorandom function compatible with* KVC*. Then,* $\mathsf{cNYM}[\mathsf{KVC}, \mathsf{PRF}]$ *is a keyed-verification pseudonym system with unforgeability advantage:*

$$\mathsf{Adv}^{\text{unf}}_{\mathsf{cNYM}[\mathsf{KVC},\mathsf{PRF}]}(\lambda) \leq q\mathsf{Adv}^{\text{ext}}_{\mathsf{KVC}}(\lambda, 1) + q\mathsf{Adv}^{\text{anon}}_{\mathsf{KVC}}(\lambda) + q\mathsf{Adv}^{\text{prnd}}_{\mathsf{PRF}}(\lambda) + \frac{q^2}{|\mathbb{M}|} \ ,$$

(where q *is the number of oracle queries) and unlinkability advantage:*

$$\mathsf{Adv}^{\text{unlink}}_{\mathsf{cNYM}[\mathsf{KVC},\mathsf{PRF}]}(\lambda) \leq 2(\mathsf{Adv}^{\text{anon}}_{\mathsf{KVC}}(\lambda, 1) + \mathsf{Adv}^{\text{prnd}}_{\mathsf{PRF}}(\lambda)) \ .$$

Remark 11. The requirement of PRF compatibility and re-randomization of KVC can be relaxed with an alternative construction, that we sketch below.

At issuance time, the user generates a key $k \leftarrow \mathsf{NYM.K}(crs)$ and an evaluation $nym_0 \leftarrow \mathsf{NYM.E}(k, \varepsilon)$ (where ε denotes the empty string). At issuance time, the user sets the key k to be one of the (hidden) credential attributes, and adds to the user message the registration identity nym_0. Along with the predicate associated with the other attributes, the user proves knowledge of a valid PRF key $k \in \mathbb{K}$ (a hidden attribute) and that nym_0 is well-formed (via the predicate ϕ_{nym}). The server, on the receiving side, checks that nym_0 has not already been used in the past and that it is indeed correctly generated by verifying the predicate (via ϕ_{nym}). If both checks pass, the server proceeds with the issuing of a credential. Upon accessing any other resource $scp \neq \varepsilon$ the user can provide an identity associated to that resource computing $nym \leftarrow \mathsf{NYM.E}(k, scp)$ and proving via the predicate ϕ_{nym} that the computation was done correctly.

Remark 12. In the proof, extraction is used to carry out a counting argument after making sure that nym^*_i are all correct evaluations of the function $\mathsf{PRF}(\cdot, scp^*)$. Therefore, one may wonder if one-more unforgeability is sufficient for proving security. While we don't have at disposal the final messages, the credentials $\mu\mathsf{CMZ}$ and $\mu\mathsf{BBS}$ both rely on a presentation message that is sound, and therefore the argument still applies. In fact, by relying on knowledge soundness of the proofs π_p in both protocols, one can show that $\mu\mathsf{CMZ}_{AT}$ and $\mu\mathsf{BBS}_{AT}$ (the schemes without the issuance user proofs) are sufficient for achieving the desired security properties. Note that the "q" factor required to "guess" the forgery is now moved to the knowledge soundness experiment.

Corollary 37. *Consider a one-more unforgeable keyed-verification credential* $\mathsf{KVC}_{AT} \in \{\mu\mathsf{CMZ}_{AT}, \mu\mathsf{BBS}_{AT}\}$*, for predicate family* $\Phi \supset \{\phi_{\text{prf}}\}$*. Let* PRF *be a pseudorandom function compatible with* KVC_{AT}*. Then,* $\mathsf{cNYM}[\mathsf{KVC}_{AT}, \mathsf{PRF}]$ *is a keyed-verification pseudonym system with unforgeability advantage:*

$$\mathsf{Adv}^{\text{unf}}_{\mathsf{cNYM}[\mathsf{KVC}_{AT},\mathsf{PRF}]}(\lambda) \leq \mathsf{Adv}^{\text{omuf}}_{\mathsf{KVC}_{AT}}(\lambda, 1) + q\mathsf{Adv}^{\text{ksnd}}_{\mathsf{ZKP}}(\lambda) + q\mathsf{Adv}^{\text{anon}}_{\mathsf{KVC}_{AT}}(\lambda) + \mathsf{Adv}^{\text{prnd}}_{\mathsf{PRF}}(\lambda) + \frac{q^2}{|\mathbb{M}|} \ ,$$

(where q *is the number of oracle queries) and unlinkability advantage:*

$$\mathsf{Adv}^{\text{unlink}}_{\mathsf{cNYM}[\mathsf{KVC},\mathsf{PRF}]}(\lambda) \leq 2(\mathsf{Adv}^{\text{anon}}_{\mathsf{KVC}_{AT}}(\lambda, 1) + \mathsf{Adv}^{\text{prnd}}_{\mathsf{PRF}}(\lambda)) \ .$$

8.3.3 Analysis

Correctness follows by inspection; unlinkability follows directly from pseudorandomness and anonymity. We focus on unforgeability.

Lemma 38. *Let* KVC *be a keyed-verification credential for* $n = 1$ *attributes, with re-randomizable issuance and labelled presentation, for predicate family* $\Phi \supset \{\phi_{\text{prf}}\}$*. Let* PRF *be a pseudorandom function compatible with* KVC*. Then,* $\mathsf{cNYM}[\mathsf{KVC}, \mathsf{PRF}]$ *is a keyed-verification pseudonym system with unforgeability advantage:*

$$\mathsf{Adv}^{\text{unf}}_{\mathsf{cNYM}[\mathsf{KVC},\mathsf{PRF}]}(\lambda) \leq q\mathsf{Adv}^{\text{ext}}_{\mathsf{KVC}}(\lambda, 1) + q\mathsf{Adv}^{\text{anon}}_{\mathsf{KVC}}(\lambda) + q\mathsf{Adv}^{\text{prnd}}_{\mathsf{PRF}}(\lambda) + \frac{q^2}{|\mathbb{M}|} \ ,$$

where q *is the number of queries to the issuance oracles.*

Proof. We consider, without loss of generality, two types of adversaries:

- a "one-more" adversary $\mathsf{A}_1(pp)$ that returns $(scp, (nym_i^*, m_i^*, \rho_i^*)_{i=1}^{q+1})$, such that $(nym_i)_i$ are all different and q is the number of queries to the ISSUE oracle. This attacker can be reduced to extractability of the underlying keyed-verification credential system. To do so, we build a reduction B for the extraction game $\text{EXT}_{\mathsf{KVC},\mathsf{Ext},\mathsf{A}}(\lambda, 1)$.

  B receives as input pp and internally runs A_1, mapping each oracle query to the respective one in the extractability game:
  - each issuance query with adversarial user message μ is forwarded as $\text{ISSUE}(\phi_{\text{prf}}^{scp,nym}, \mu)$
  - each presentation query of the form $(scp, nym, m), \rho$ is forwarded as $\text{PRESENT}(\phi_{\text{prf}}^{scp,nym} \ \wedge \ \phi_{\text{lbl}}^{m}, \rho)$
  - each new user query has B sample $k \leftarrow\!\!\$\ \mathbb{M}$, runs $\text{NEWUSR}(k)$ and upon receiving a counter ctr for the given user, it stored $Keys[ctr] \coloneqq k$. Finally, returns ctr.
  - each user presentation query of the form (i, scp, m) is forwarded to the extractability oracle as $\text{PRESENTUSR}(i, \phi_{\text{prf}}^{scp,nym} \wedge \phi_{\text{lbl}}^{m})$ where $nym \coloneqq \mathsf{NYM.E}(Keys[i], scp)$. (If the index is invalid, we assume the oracle returns $\perp$).
  - finally, for each $\text{EVAL}(i, scp)$, returns $\mathsf{PRF}(Keys[i], scp)$.

  The oracles provide the exact same output distribution as in the original game. At the end, the adversary B guesses an index $j \in [q+1]$ and returns $(scp^*, nym_j^*, m_j^*, \rho_j^*)$.

  Since there are at most q different issuance queries, at most q keys have been issued. However, the adversary produced $q+1$ different evaluations of the function $\mathsf{PRF}(\cdot, scp)$. This implies that either one of the keys is not present as a part of the issued credentials (and the winning condition "$\vec{m} \notin Qrs$" in the extractability game is satisfied) or an extracted credential does not satisfy the pseudonym predicate (so the condition "$\phi(\vec{m}) = 0$" in the extractability game is satisfied).

- an adversary stealing pseudonyms from honest users $\mathsf{A}_2(pp)$ that returns a valid forgery $(scp^*, nym^*, m^*, \rho^*)$ not previously queried, and such that there exists an index $j \in [q_u]$ (q_u being the number of queries to NEWUSR) such that $nym^* = \mathsf{NYM.E}(Keys[j], scp^*)$. This can be proven via a hybrid argument:

  $\mathsf{H_0}$ the first hybrid runs the extractor for the credential presentation and checks if the extracted values satisfy the predicate. Similarly to the previous argument, if the extracted values are not correct PRF evaluation than it is possible to build a reduction to extractability of the underlying credential system.

  $\mathsf{H_1}, \ldots, \mathsf{H_{q_u}}$ replace the i-th query to NEWUSR() and queries of the form $\text{PRESENTUSR}(i, \cdot)$ to simulated responses. Upon receiving a NEWUSR() query, sample $k_i \leftarrow\!\!\$\ \mathbb{M}$ and run $st_i \leftarrow \mathsf{Sim.I}(pp, \phi_{[1]})$. Store k_i in $Keys$ and the simulator state st_i in $Creds$. Upon receiving a $\text{PRESENTUSR}(i, scp, m)$, run the pseudonym evaluation using k_i and return the simulated proof via $\mathsf{Sim.P}(pp, \phi_{\text{prf}}^{scp,nym} \ \wedge \ \phi_{\text{lbl}}^{m})$. Evaluation queries are dealt as before using the relative PRF key.
  If the adversary's output is distinguishable with non-negligible probability then it is possible to build a distinguisher for anonymity of KVC.

  $\mathsf{H_{q_u+1}}, \ldots, \mathsf{H_{2q_u}}$ replace all pseduonyms with uniformly-random PRF images. The challenger now holds a table $Nyms$ of evaluations and each $\text{EVAL}(i, scp)$ query is responded as follows: if $Nyms[i, scp]$ has associated a pseudonym already, then return it; if no such entry exists, then sample a fresh pseudonym $nym \leftarrow\!\!\$\ \mathbb{I}$, where $\mathbb{I}$ is the image of the PRF.

  At this point we have that the returned pseudonyms collides with a random string only if there is a collision within PRF. The probability that this bad event occurs is upper bounded by $q^2/|\mathbb{I}|$.

□

8.3.4 Instantiation

A suitable PRF that composes well with the studied keyed-verification systems is the PRF from Naor, Pinkas, and Reingold [NPR99]:

$$\mathsf{NPR}(k, scp) \coloneqq k \cdot \mathsf{H}_g(scp) \tag{31}$$

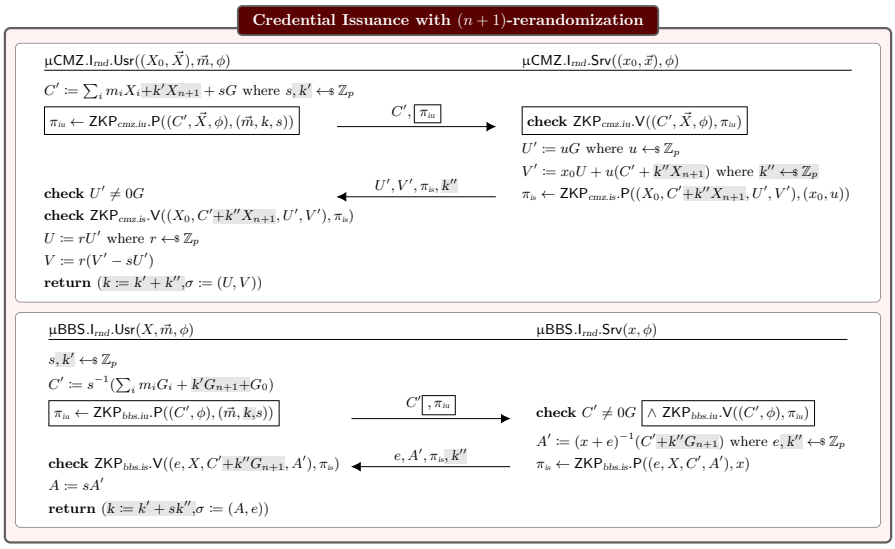


Figure 10: Randomized issuance of μCMZ and μBBS credentials. The relation π_{is} is defined in equations (5) and (18) for μCMZ and μBBS respectively. Highlighted, the differences with the vanilla protocol; boxed, the parts that may be removed for one-more unforgeability.

The PRF is secure in the random oracle model under the DDH assumption: distinguishing a tuple $(\mathsf{H}_g(scp_0), k\mathsf{H}_g(scp_0), \mathsf{H}_g(scp_1), k\mathsf{H}_g(scp_1))$ from a uniformly random element of $\mathbb{G}^4$ is equivalent to solving the DDH problem: given a DDH challenge (with generator) (P,U,Q,V), set $\mathsf{H}_g(scp_0) = P$, $\mathsf{H}_g(scp_1) = Q$, and $k = \log_P U$.

8.3.5 One-time linking

It is also possible to upgrade to statistical anonymity in a different model. Instead of providing a *nym*, the user can commit to it and prove it has the same value of a commitment sent in a previous interaction. Given two pseudonyms $C_0 = k\mathsf{H}_g(scp) + rH$ and $C_1 = k\mathsf{H}_g(scp) + r'H$ the user can prove knowledge of the discrete logarithm of $C_0 - C_1$ base H and that C_0, C_1 are correct commitments to nym evaluations. The advantage of this last approach is that the user has perfect hiding.

9 Straight-line extraction from Σ-protocols

The notion of soundness required for credentials that we use is strong, as it demands that the simulator is able to extract a witness from an oracle query in order to properly answer the challenge. In the literature, it has sometimes been called online [Fis05] or straight-line [GMY06, Sah99] extraction, and is believed not to be satisfied by traditional non-interactive Schnorr proofs relying on rewinding techniques.

The Fiat–Shamir transform of a Σ-protocol is however the most common tool for instantiating the NIZK in credential systems. To complement this heuristic and resolve the theoretical gap, we prove that representation proofs are straight-line extractable in the algebraic group model.

We denote with Σ such a protocol for generic linear relations (m of them) over vectors of size n:

$$\mathsf{R}_{\mathbf{F}} \coloneqq \left\{ (\vec{X},\vec{x}) \in \mathbb{Z}_p^n \times \mathbb{G}^m : \sum_k^n x_k F_{1,k} = X_1 \ \wedge\ \cdots\ \wedge\ \sum_k^n x_k F_{m,k} = X_m \right\}$$

We will employ the more compact matrix-vector notation $\mathbf{F}\vec{x} = \vec{X}$ where $\mathbf{F} \in \mathbb{G}^{m \times n}$ denotes the matrix whose j-th row vector is $\vec{F_j}$. Denote with $\mathsf{Uniq}(\mathbf{F})$ the list $G_1, \ldots, G_\ell$ with $\ell \leq |\mathbf{F}|$ the non-trivial group elements appearing in $\mathbf{F}$, repeated only once.

Definition 39. A relation family $R_{\mathbf{F}}$ is *admissible* if: for any p.p.t. adversary A, it is computationally hard to find a non-zero vector in $\ker(\vec{F_j})$ (for $j \in [m]$) or in $\ker(\mathsf{Uniq}(\mathbf{F}))$, where Uniq denotes the set of non-trivial group elements appearing in $\mathbf{F}$.

We recall here the protocol Σ for the relation $\mathsf{R}_{\mathbf{F}}$:

Procedure $\Sigma.\mathsf{P}(\vec{X}, \vec{x})$	Procedure $\Sigma.\mathsf{V}(\vec{X}, (\vec{R}, \vec{s}))$
$\vec{R} \coloneqq \mathbf{F}\vec{r}$ where $r \leftarrow\!\!\$\ \mathbb{Z}_p^n$	$c \coloneqq \mathsf{H}_p(\vec{X}, \vec{R})$
$c \coloneqq \mathsf{H}_p(\vec{X}, \vec{R})$	**return** $\mathbf{F}\vec{s} = R + c\vec{X}$
$\vec{s} \coloneqq \vec{r} + c\vec{x}$	
return $(\vec{R}, \vec{s})$	

in the above, the notation $\mathbf{F}\vec{r}$ indicates standard matrix-vector product, i.e., $\vec{R}$ is the vector of m elements indexed in $j \in [m]$ such that $R_j \coloneqq \sum_k F_{j,k} r_k$. Similarly one proceeds for $\mathbf{F}\vec{s}$. This protocol is a standard Fiat–Shamir transform for a Σ-protocol proving knowledge of a group morphism [Mau09]. The transform as presented here is sometimes also known as "strong Fiat–Shamir transform" [BPW12]. It is a well-known result in the literature that such protocol is zero-knowledge in the random oracle model: the simulator samples $\vec{s}, c$ at random from $\mathbb{Z}_p^n, \mathbb{Z}_p$ respectively and sets $\vec{R} \coloneqq \mathbf{F}\vec{s} - c\vec{X}$. Finally, it programs the random oracle to respond with c for the query $(\vec{X}, \vec{R})$ finally returns $(\vec{R}, \vec{s})$. The proof distribution is identical to the one generated by the prover: $\vec{R}$ and $\vec{s}$ are uniformly distributed satisfying the verification equation.

Theorem 40. *The protocol Σ for an admissible relation $\mathsf{R}_{\mathbf{F}}$ is strongly simulation-extractable in the algebraic group model and the random oracle model.*

Proof. Let A be the algebraic adversary in the game for simulation extractability, and q the number of queries made to the random oracle during its execution. For a statement $(\mathbf{F}, \vec{X})$, let $\vec{F}_j$ denote the j-th row of the matrix $\mathbf{F}$ and X_j the j-th element of X. During its execution, the adversary A queries the simulation oracle for instance $\vec{X} \in \mathbb{G}^m$ and witness $\vec{x} \in \mathbb{Z}_p^n$. Being algebraic, A also provides an algebraic representation of $\vec{X}$. Upon receiving the i-th simulation query, the simulator picks $c_i, \vec{s}_i \leftarrow\!\!\$\ \mathbb{Z}_p$, sets

$$\vec{R}_i \coloneqq \mathbf{F}\vec{s}_i - c_i\vec{X}_i \tag{32}$$

and programs the random oracle to respond to $\mathsf{H}_p(\vec{X}_i, \vec{R}_i)$ with c_i. If such query exists, the simulator aborts and the adversary wins. Since R_i is uniformly distributed over $\mathbb{G}$, no such query has been previously made except with probability at most q/p which is negligible. Finally, the simulator returns $(\vec{R}_i, \vec{s}_i)$. The algebraic representation (χ_i, δ_i) of the i-th queried statement $\vec{X}_i$ is such that:

$$X_{i,j} = \sum_k^{\ell} \chi_{i,j,k} G_k + \sum_{\substack{\iota<i \\ \kappa\in[m]}} \delta_{i,\iota,\kappa} R_{\iota,\kappa}$$

where the generators $G_1, \ldots, G_\ell = \mathsf{Uniq}(\mathbf{F})$ are the group elements appearing in $\mathbf{F}$. The elements $R_{k,j}$ can be simplified thanks to equation (32). In fact, for the first query, we have that for every element $j \in [m]$:

$$X_{1,j} = \sum_k^{\ell} \alpha_{1,j,k} G_k \text{ where } \quad \alpha_{1,j,k} = \chi_{1,j,k}$$

since no previous query has been made. For the second query we can consider:

$$X_{2,j} = \sum_k^{\ell} \alpha_{2,j,k} G_k \text{ where } \quad \alpha_{2,j,k} = \chi_{2,j,k} + \delta_{2,1,k}(s_1 - c_1\alpha_{1,j,k})$$

and so on. We can partition the indices $k \in [\ell]$ into I_j – the set of indices of $\mathsf{Uniq}(\mathbf{F})$ appearing in $\vec{F}_j$ and $\bar{I}_j$ the set if indices that do not. We can thus see the i-th simulation query ($i \in [q]$) as:

$$X_{i,j} = \sum_{k\in I_j} \alpha_{i,j,k} G_k + \sum_{k\in\bar{I}_j} \alpha_{i,j,k} G_k$$

for all $j \in [m]$, with $\alpha_{i,j,k}$ in $\mathbb{Z}_p$. At the end of the execution the adversary returns a final statement $\vec{X}^*$ and a proof $\pi^* = (\vec{K}^*, \vec{s}^*)$ such that $(\vec{X}^*, \vec{R}^*, \vec{s}^*) \neq (\vec{X}_i, \vec{R}_i, \vec{s}_i)$ for all $i \in [q]$ satisfying $\mathbf{F}\vec{s}^* = \vec{K}^* + c^*\vec{X}^*$ and $c^* = \mathsf{H}_p(\vec{X}^*, \vec{R}^*)$.

We claim that $(\vec{X}^*, \vec{R}^*) \neq (\vec{X}_i, \vec{R}_i)$ for all $i \in [q]$. In fact, by contradiction, if $\exists i \in [q]$ such that $\vec{s}_i \neq \vec{s}^*$ and $\vec{X}_i = \vec{X}^*$, $\vec{R}_i = \vec{R}^*$ then $\mathbf{F}(\vec{s}_i - \vec{s}^*) = \vec{0}$ and non-trivial element of the kernel was found. From the algebraic representation we have that,

for each $j \in [m]$

$$\begin{aligned} \vec{R}_j^* &= \sum_{k \in I_j} \beta_{j,k}^* G_k + \sum_{k \in \bar{I}_j} \bar{\beta}_{j,k}^* G_k \\ \vec{X}_j^* &= \sum_{k \in I_j} \alpha_{j,k}^* G_k + \sum_{k \in \bar{I}_j} \bar{\alpha}_{j,k}^* G_k \end{aligned} \tag{33}$$

where the algebraic representation already reduced the terms R_i as linear combinations of the G_k's in $I_j \cup \bar{I}_j = \mathsf{Uniq}(\mathbf{F})$ using the verification equation as shown above. We distinguish the following possibilities:

1. If $\exists j^* \in [m], k^* \in \bar{I}_{j^*}$ s.t. $\bar{\alpha}_{j^*,k^*}^* \neq 0$, then plugging equation (33) into the verification equation we have that:

$$\sum_{k \in I_{j^*}} \left(\beta_{j^*,k}^* + c^* \cdot \alpha_{j^*,k}^* - s_k^*\right) G_k + \sum_{k \in \bar{I}_{j^*} \setminus \{k^*\}} \left(\bar{\beta}_{j^*,k}^* + c^* \bar{\alpha}_{j^*,k}^*\right) G_k + \left(\bar{\beta}_{j^*,k^*}^* + c^* \bar{\alpha}_{j^*,k^*}^*\right) G_{k^*} = 0$$

   and the coefficient of G_{k^*} is uniformly distributed and nonzero with overwhelming probability since c^* is uniformly distributed and $\bar{\alpha}_{j^*,k^*}^* \neq 0, \bar{\beta}_{j^*,k^*}^*$ are chosen before seeing the challenge. Therefore, we have a non-trivial relation of $\mathsf{Uniq}(\mathbf{F})$.

2. If $\forall i, j, k\colon \bar{\alpha}_{j,k}^* = 0$ modulo p then we can rearrange the indices of $\alpha_{j,k}^*$ and can consider vectors $\vec{\alpha}_j$ $(j \in [m])$ such that $X_1 = \langle F_1, \vec{\alpha}_1 \rangle, \ldots, X_m = \langle F_m, \vec{\alpha}_m \rangle$. We claim that $\vec{\alpha}_1, \ldots, \vec{\alpha}_m$ can be used to reconstruct a unique witness vector.

   (a) If $\exists j \neq j'$ such that for some $k \in I_j \cap I_{j'} : \alpha_{j,k} \neq \alpha_{j',k}$ then

$$\vec{F}_j \cdot (\vec{\beta}_j + c^* \vec{\alpha}_j - \vec{s}^*) = 0 \text{ and } \vec{F}_{j'} \cdot (\vec{\beta}_{j'} + c^* \vec{\alpha}_{j'} - \vec{s}^*) = 0$$

   and the adversary found another nontrivial relation of the kernel of $\vec{F}_j$ or $\vec{F}_{j'}$: if $(\vec{\beta}_j + c^* \vec{\alpha}_j - \vec{s}^*) \neq 0$ we are done, and if it is instead equal to zero then the term $(\vec{\beta}_{j'} + c^* \vec{\alpha}_{j'} - \vec{s}^*)$ will be also zero only with negligible probability $1/p$ since $(\vec{\beta}_{j'} - \vec{\beta}_j) + c(\vec{\alpha}_{j'} - \vec{\alpha}_j) = 0$ only if the adversary guessed c correctly (note that all the terms are sent to the random oracle before seeing the challenge).

   (b) Otherwise, the extractor defines $\vec{x}^* \in \mathbb{Z}_p^n$ as the vector whose k-th entry is the (unique) element $\alpha_{j,k}$ for all $j \in [m]$ such that $k \in I_j$. The witness is correct since, by construction, $\vec{F}_j \vec{x}^* = \vec{F}_j \vec{\alpha}_j = X_j$ for all $j \in [m]$.

The witness output from item 2b is correct, and all other cases happen with negligible probability, thus the protocol is strongly simulation-extractable. □

10 Acknowledgements

This work would not have been possible without the ideas of Trevor Perrin and Dennis Jackson (Firefox); the numerous insightful discussions with Geoffroy Couteau (CNRS), Greg Zaverucha (Microsoft Research), Georg Fuchsbauer (TU Wien), Balthazar Bauer (UVSQ), Alexander Koch (IRIF).

References

[AFG+16] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. *Journal of Cryptology*, 29(2):363–421, April 2016. 2

[ASM06] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 111–125, September 2006. 3, 6, 8, 23

[AYY23] Ghous Amjad, Kevin Yeo, and Moti Yung. RSA blind signatures with public metadata. Cryptology ePrint Archive, Report 2023/1199, 2023. 8

[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. 1

[BBDT17] Amira Barki, Solenn Brunet, Nicolas Desmoulins, and Jacques Traoré. Improved algebraic macs and practical keyed-verification anonymous credentials. In *Selected Areas in Cryptography–SAC 2016: 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers 23*, pages 360–380. Springer, 2017. 2, 3, 4, 6, 23, 24, 25

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55, August 2004. 3, 6, 25

[BCC+09] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125, August 2009. 1

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357, May 2016. 3

[BCKL07] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. Non-interactive anonymous credentials. Cryptology ePrint Archive, Report 2007/384, 2007. 2

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60, October / November 2016. 9

[BDFG20] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Efficient polynomial commitment schemes for multiple points and polynomials. Cryptology ePrint Archive, Paper 2020/081, 2020. https://eprint.iacr.org/2020/081. 31

[BEK+21] Jan Bobolz, Fabian Eidens, Stephan Krenn, Sebastian Ramacher, and Kai Samelin. Issuer-hiding attribute-based credentials. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *CANS 21*, volume 13099 of *LNCS*, pages 158–178, December 2021. 2, 4

[Ber06] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 207–228, April 2006. 4

[BLCL91] Gilles Brassard, Sophie Laplante, Claude Crépeau, and Christian Léger. Computationally convincing proofs of knowledge. In Christian Choffrut and Matthias Jantzen, editors, *STACS 91*, pages 251–262, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg. 10

[BLS04] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. On the selection of pairing-friendly groups. In Mitsuru Matsui and Robert J. Zuccherato, editors, *SAC 2003*, volume 3006 of *LNCS*, pages 17–25, August 2004. 4

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629, May 2003. 1

[BN06] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *SAC 2005*, volume 3897 of *LNCS*, pages 319–331, August 2006. 4

[BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 626–643, December 2012. 8, 44

[Bra95] Stefan Brands. Off-line electronic cash based on secret-key certificates. In Ricardo A. Baeza-Yates, Eric Goles Ch., and Patricio V. Poblete, editors, *LATIN 1995*, volume 911 of *LNCS*, pages 131–166, April 1995. 1, 2

[Bra00] Stefan Brands. *Rethinking public key infrastructures and digital certificates: building in privacy*. Mit Press, 2000. 1, 2

[CBC+24] Miranda Christ, Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Deepak Maram, Arnab Roy, and Joy Wang. SoK: Zero-knowledge range proofs. Cryptology ePrint Archive, Paper 2024/430, 2024. https://eprint.iacr.org/2024/430. 34, 35

[CCL+21] Konstantinos Chalkias, Shir Cohen, Kevin Lewi, Fredric Moezinia, and Yolan Romailler. HashWires: Hyperefficient credential-based range proofs. *PoPETs*, 2021(4):76–95, October 2021. 35

[CDV22] Melissa Chase, F. Betül Durak, and Serge Vaudenay. Anonymous tokens with hidden metadata bit from algebraic MACs. Cryptology ePrint Archive, Paper 2022/1622, 2022. 21

[CDV23] Melissa Chase, F. Betül Durak, and Serge Vaudenay. Anonymous tokens with stronger metadata bit hiding from algebraic MACs. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 418–449, August 2023. 6

[CFQ19] Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2075–2092. ACM Press, November 2019. 2, 9

[CGKR22] Geoffroy Couteau, Dahmun Goudarzi, Michael Klooß, and Michael Reichle. Sharp: Short relaxed range proofs. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 609–622. ACM Press, November 2022. 35

[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. 3

[Cha85] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985. 1

[Che06] Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 1–11, May / June 2006. 7

[Chi10] Alessandro Chiesa. *Proof-carrying data*. PhD thesis, Massachusetts Institute of Technology, 2010. 3

[CHK+06] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 201–210. ACM Press, October / November 2006. 7, 8, 35

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768, May 2020. 9, 30, 31, 33

[CKL+16] Jan Camenisch, Stephan Krenn, Anja Lehmann, Gert Læssøe Mikkelsen, Gregory Neven, and Michael Østergaard Pedersen. Formal treatment of privacy-enhancing credential systems. In Orr Dunkelman and Liam Keliher, editors, *SAC 2015*, volume 9566 of *LNCS*, pages 3–24, August 2016. 2, 14

[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, May 2001. 1, 2, 3

[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, September 2003. 1, 2, 3

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96, August 2006. 2

[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, March 2019. 2

[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. 1, 2, 3, 4, 5, 11, 12, 13, 14, 15, 16

[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. 3, 4

[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, April 2019. 13

[CV02] Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In Vijayalakshmi Atluri, editor, *ACM CCS 2002*, pages 21–30. ACM Press, November 2002. 1

[DFM01] Roger Dingledine, Michael J. Freedman, and David Molnar. The free haven project: distributed anonymous storage service. In *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, page 67–95, Berlin, Heidelberg, 2001. Springer-Verlag. 5

[DG23] Quang Dao and Paul Grubbs. Spartan and bulletproofs are simulation-extractable (for free!). In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 531–562, April 2023. 3, 11

[DGS+18] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *PoPETs*, 2018(3):164–180, July 2018. 4, 35

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 613–631, December 2010. 11

[DKL+23] J. Doerner, Y. Kondi, E. Lee, A. Shelat, and L. Tyner. Threshold bbs+ signatures for distributed anonymous credential issuance. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 773–789, Los Alamitos, CA, USA, may 2023. IEEE Computer Society. 1

[DVC22] F. Betül Durak, Serge Vaudenay, and Melissa Chase. Anonymous tokens with hidden metadata bit from algebraic MACs. Cryptology ePrint Archive, Report 2022/1622, 2022. 11

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431, January 2005. 38

[Eag22] Liam Eagen. Bulletproofs++. Cryptology ePrint Archive, Report 2022/510, 2022. 35

[ECK+23] Jens Ernstberger, Stefanos Chaliasos, George Kadianakis, Sebastian Steinhorst, Philipp Jovanovic, Arthur Gervais, Benjamin Livshits, and Michele Orrù. zk-bench: A toolset for comparative evaluation and performance benchmarking of snarks. Cryptology ePrint Archive, Paper 2023/1503, 2023. https://eprint.iacr.org/2023/1503. 35

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168, August 2005. 8, 43

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62, August 2018. 9

[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, August 1987. 8

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. 10

[GMY06] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. *Journal of Cryptology*, 19(2):169–209, April 2006. 43

[GOP+22] Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 397–426, May / June 2022. 3

[Gro15] Jens Groth. Efficient fully structure-preserving signatures for large messages. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 239–259, November / December 2015. 3

[Ham15] Mike Hamburg. Decaf: Eliminating cofactors through point compression. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 705–723, August 2015. 4

[JKK14] Stanislaw Jarecki, Aggelos Kiayias, and Hugo Krawczyk. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 233–253, December 2014. 3

[JY09] David Jao and Kayo Yoshida. Boneh-boyen signatures and the strong diffie-hellman problem. In Hovav Shacham and Brent Waters, editors, *Pairing-Based Cryptography – Pairing 2009*, pages 1–16, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. 7

[KLOR20] Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. Anonymous tokens with private metadata bit. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 308–336, August 2020. 3, 6, 11

[KLSS17] Stephan Krenn, Thomas Lorünser, Anja Salzer, and Christoph Striecks. Towards attribute-based credentials in the cloud. In Srdjan Capkun and Sherman S. M. Chow, editors, *CANS 17*, volume 11261 of *LNCS*, pages 179–202, November / December 2017. 2

[KT23] Tohru Kohrita and Patrick Towa. Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments. Cryptology ePrint Archive, Paper 2023/917, 2023. https://eprint.iacr.org/2023/917. 8, 32

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194, December 2010. 8

[Mau09] Ueli M. Maurer. Abstraction in cryptography (invited talk). In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, page 465, August 2009. 44

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and KDCs. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 327–346, May 1999. 42

[NS04] Lan Nguyen and Reihaneh Safavi-Naini. Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings. In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 372–386, December 2004. 35

[PS16] David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126, February / March 2016. 3, 5, 16

[PS18] David Pointcheval and Olivier Sanders. Reassessing security of randomizable signatures. In Nigel P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 319–338, April 2018. 5

[QRW19] Willy Quach, Ron D. Rothblum, and Daniel Wichs. Reusable designated-verifier NIZKs for all NP from CDH. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 593–621, May 2019. 3, 31

[Qu99] Minghua Qu. Sec 2: Recommended elliptic curve domain parameters, 1999. 4

[RP22] Alfredo Rial and Ania M. Piotrowska. Security analysis of coconut, an attribute-based credential scheme with threshold issuance. Cryptology ePrint Archive, Report 2022/011, 2022. 1

[SAB+19] Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. In *NDSS 2019*. The Internet Society, February 2019. 1

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999. 43

[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, May 2020. 1

[Sch01] Claus-Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *ICICS 01*, volume 2229 of *LNCS*, pages 1–12, November 2001. 1

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266, May 1997. 9

[SS22] Tjerand Silde and Martin Strand. Anonymous tokens with public metadata and applications to private contact tracing. In Ittay Eyal and Juan A. Garay, editors, *FC 2022*, volume 13411 of *LNCS*, pages 179–199, May 2022. 8

[Sta96] Markus Stadler. *Cryptographic protocols for revocable privacy*. PhD thesis, ETH Zurich, 1996. 1

[TFS04] Isamu Teranishi, Jun Furukawa, and Kazue Sako. k-Times anonymous authentication (extended abstract). In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 308–322, December 2004. 1, 8, 35

[TG23] Lindsey Tulloch and Ian Goldberg. Lox: Protecting the social graph in bridge distribution. *PoPETs*, 2023(1):494–509, January 2023. 3

[TZ23] Stefano Tessaro and Chenzhi Zhu. Revisiting BBS signatures. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 691–721, April 2023. 3, 6, 24, 25, 29