Bulletproofs are Optimal: Lower Bounds for Vector Commitments from Fiat-Shamir in Pairing-Free Groups

Marian Dietz, Emanuele Giunta

Department of Computer Science
ETH Zurich
Zurich, Switzerland
{marian.dietz,emanuele.giunta}@inf.ethz.ch

Abstract. Vector Commitments allow parties to commit to an (ordered) vector and to later succinctly open it at any desired position. In plain (i.e., known order and pairing-free) prime-order groups, all state-of-the-art constructions rely on the combination of a generalized Pedersen commitment and an inner product argument. This includes the celebrated constructions of Bootle et al. (EUROCRYPT 2016) and Bünz et al. (S&P 2018), and subsequent improvements. All of these achieve proofs consisting of $\Theta(\log n)$ group elements, and further provide useful properties including malleability, subvector opening and transparent setup. However, breaking the $O(\log n)$ barrier in the plain discrete logarithm setting has proven to be a hard problem. This is in stark contrast with other settings, e.g. from pairing-friendly groups or groups of unknown order, where constructions with constant-size commitment and openings have been known for over a decade.

In this work we investigate whether this limitation is inherent to constructions based on prime-order groups. Specifically we prove new lower bounds for any accumulator (a weaker object than vector commitments, thus making our result more general), with interactive public-coin membership proof in Maurer's Generic Group Model. An implication of our result is that in such setting at least one of the following must occur: (i) the commitment has super-constant size; (ii) the opening proof contains $\Omega(\log n)$ group elements; (iii) the opening proof has length $n^{1-o(1)}$.

Our bound further extends to primitives that directly imply accumulators/VCs including polynomial/functional commitments and inner product arguments.

Table of Contents

1 Introduction **3**
- 1.1 Our Results . . . 4
- 1.2 Technical Overview . . . 5
- 1.3 Related Works . . . 9

2 Preliminaries **10**
- 2.1 Notation . . . 10
- 2.2 Entropy . . . 11
- 2.3 Compression Arguments . . . 12
- 2.4 Generic and Type-Safe Algorithms . . . 12
- 2.5 Interactive Type-Safe Accumulators . . . 16

3 Internal Procedures **18**
- 3.1 Information Theoretical Game . . . 18
- 3.2 Extraction Subroutine . . . 19
- 3.3 Forge Subroutine . . . 24

4 Witness Length Bound **26**
- 4.1 Basic Decoding Strategy . . . 27
- 4.2 Full Compression Argument . . . 29
- 4.3 Lower Bound . . . 32

A Postponed Proofs **39**
- A.1 Compression Arguments . . . 39
- A.2 Type-Safe Algorithms . . . 40
- A.3 Basic Decoder . . . 42
- A.4 Full Compression Argument . . . 43

1 Introduction

Vector Commitments (VC) [CF13] are a cryptographic primitive allowing a sender to commit to an ordered list of n messages, and only later opening to a specific entry of this vector in a publicly verifiable way. Along with *correctness*, the key properties VCs are required to satisfy are *position binding* and *succinctness*. The first mandates that opening a commitment in a given position to two different values is infeasible, while the latter requires commitment and openings to be short, i.e., to depend only poly-logarithmically on n.

Over the years, Vector Commitments have found applications in many concrete settings and in the realizations of more structured primitives. Among them, zero-knowledge sets [MRK03, LY10, CF13], proof of retrievability [JK07], space [DFKP15] and sequential work [MMV13], stateless blockchain [CPZ18, BBF19, GRWZ20] and succinct arguments, e.g. [Kil92].

The first realization of VC dates back to Merkle trees [Mer88], where however opening proofs have size $\Theta(\log n)$. VCs with constant (in n) commitment and opening size were later realized from a variety of public-key settings including pairing groups [LY10, KZG10, CF13] and groups of unknown order [CF13]. These also provide additional functionalities, including homomorphism/malleability, subset opening, and functional opening [LRY16] (with the most prominent example being that of polynomial commitments).

VC from pairing-free prime-order groups. Despite the existence of constant size group-theoretic constructions, realizing VC from plain prime-order groups has remained an elusive problem. To this day, two classes of constructions achieve $\Theta(\log n)$ opening proof size, while keeping commitment size $O(1)$ in n:

On the one hand, Merkle trees based on Pedersen hash provides $\Theta(\log n)$ opening length and verification time. However, due to the non-algebraic usage of the group, these constructions offer no homomorphic properties or efficient subvector opening.

On the other hand, in the Random Oracle Model, a generalized version of Pedersen commitments can be combined with an inner product argument (IPA) of knowledge to yield a VC. Specifically, given a commitment $C = m_1G_1 + \ldots + m_nG_n$ to a vector $(m_1, \ldots, m_n)$, opening to m_i can be done by proving knowledge of scalars $m_1, \ldots m_{i-1}, m_{i+1}, \ldots, m_n$ such that $C - m_iG_i = \sum_{j \neq i} m_jG_j$. This approach further allows opening to any linear function of the committed vector, and may offer succinct subset openings depending on the underlying IPA.

Inner Products Arguments. The opening size of VC in this second family is directly linked to the efficiency of the underling IPA. The first construction of an IPA with logarithmic proof length has been proposed in [BCC$^+$16], where $4\log n$ group elements and $2\log n$ field elements, along with a constant number of extra terms, are needed per proof. [BBB$^+$18] later reduced the proof length

to $2\log n$ group elements and $O(1)$ field elements. Subsequently, [AC20] revisited their construction within the compressed sigma-protocol framework, while [CHJ+22, EKRN24] provided concrete constant improvements, especially in the applications of IPA to range proofs. Given the current state of the art, we therefore identify the following open question:

Do there exist Vector Commitments from pairing-free prime-order groups based on Fiat-Shamir that (i) use the group algebraically (up to the usage of Fiat-Shamir) and (ii) have $O(1)$ commitment and $o(\log n)$ opening length?

We find this problem to be of significant interest both in practice, given the scale at which vector commitments and IPA (for range proofs an more) are deployed, and in theory. Indeed, as the $\log n$ limitation in currently known constructions appears to stem from the reliance on recursive folding argument, any positive answer would likely require new techniques.

1.1 Our Results

In this we answer the above question *negatively* in many cases of interest. We do so by providing lower bounds for set accumulators [BdM94] with public-coin interactive membership proofs making black-box usage of a prime-order group. These are weaker objects than (public-coin, interactive) VCs in that they allow committing to an *unordered* set S, and thus only support membership proofs, as opposed to positional opening.

Specifically, when accumulating a set S of size n, if the accumulator has ρ rounds, digest length $|\mathsf{cm}|$, membership proof length $|w|$ and each round i requires μ_i many group elements to be sent from the prover, then asymptotically our bound implies that

$$|\mathsf{cm}| \cdot |w| \cdot 2^{\left(\sum_{i=1}^{\rho-1} \mu_i\right)} = \Omega(n).$$

In particular, this shows that one of the following conditions must be met:

- the digest does not have constant size in n;
- the total number of group elements[1] per proof is $\sum_{i=1}^{\rho-1} \mu_i = \Omega(\log n)$; or
- the proof bit-length is $|w| = \Omega(n^{\epsilon})$ for any constant $\epsilon > 0$.

Furthermore, our results naturally extend to any primitive which directly yields an accumulator, such as vector commitments and inner product arguments.

We formally prove our result assuming the accumulator to be in Maurer's Generic Group Model (GGM) [Mau05], and that security holds in the GGM against computationally unbounded adversaries. These assumptions are necessary to avoid schemes that ignore the given prime-order group entirely, or whose security would otherwise reduce to assumptions that are not group-theoretic. In line with similar group-based impossibility results (see e.g. [DHH+21, CFGG22, DKT26]), our barrier can be bypassed in the following ways:

[1] even excluding those sent in the last round

– making non-black-box use of the group, which in our case includes explicitly relying on the group element's bit representations;
– assuming additional structure about the group, such as pairings or unknown order; or
– reducing security to hardness assumptions that do not hold statistically in the GGM.

1.2 Technical Overview

Impossibility of Signatures in the GGM. Our starting point are the negative results for signatures and vector commitments in the GGM (without interaction) from [DHH+21, CFGG22]. Both results, in line with other lower bounds in the area [DKT26], essentially rely on the following facts that hold in Maurer's GGM.

1. Equality queries made by a procedure $\mathcal{M}$ in the GGM are equivalent to testing whether a given vector of scalars $\mathbf{x} \in \mathbb{F}_q^n$ is orthogonal to the vector of group elements $\mathfrak{a} \in \mathbb{G}^n$ passed to $\mathcal{M}$ as input.
2. For any procedure $\mathcal{M}$ in the GGM on input $(\mathfrak{a}, x) \in \mathbb{G}^n \times \{0,1\}^m$ we can extract the space of linear relations on $\mathfrak{a}$ that $\mathcal{M}$ found in its execution by looking at its query history. A linear relation is a vector $\mathbf{x} \in \mathbb{F}_q^n$ for which $\mathbf{x}^\top \cdot \mathfrak{a} = \mathfrak{o}$ holds.
3. For any procedure $\mathcal{M}$ in the GGM, we can *simulate* its execution by replacing group elements $\mathbf{a} \in \mathbb{G}^n$ with a space of linear relations $L \subseteq \mathbb{F}_q^n$ that $\mathbf{a}$ satisfies. Specifically, when $\mathcal{M}$ asks whether $\mathbf{x}^\top \mathbf{a} = \mathfrak{o}$ is true, a simulator replies positively iff $\mathbf{x} \in L$. The simulation itself does not cost any group operations.
4. For any procedure $\mathcal{M}$ and input $(\mathfrak{a}, x) \in \mathbb{G}^n \times \{0,1\}^m$ with $\mathfrak{a}$ satisfying all relations in L, either $\mathcal{M}$ finds a new relation on $\mathfrak{a}$ (i.e., it finds some $\mathbf{x} \notin L$ with $\mathbf{x}^\top \cdot \mathfrak{a} = \mathfrak{o}$), or it returns the same output as when simulated with L and input x.

Given the above observations, assume there exists a signature scheme with statistical security in the GGM, i.e., against an unbounded adversary that makes at most a polynomial number of group operations. Then one considers the following EUF-CMA adversary[2] $\mathcal{A}$, which for each message m either is able to produce a forgery for m, or it will find a new linear relation on the verification key's group elements after querying a signature for m. Since there are at most polynomially many linearly independent relations $\mathcal{A}$ can find, the adversary is guaranteed to find a forgery after sufficiently many iterations.

In order to make this argument about impossibility of signatures in the GGM more precise, let us fix some notation. W.l.o.g., the verification key is of the

[2] Technically, the adversaries in [DHH+21, CFGG22] follow somewhat different strategies, but we opt for the following formulation to better present our techniques.

form $\mathsf{vk} = (\mathfrak{c}, t_0) \in \mathbb{G}^{\mu_0} \times \{0,1\}^{\eta_0}$, and signatures are of the form $\sigma = (\mathfrak{a}, t_1) \in \mathbb{G}^{\mu_1} \times \{0,1\}^{\eta_1}$. Given vk, $\mathcal{A}$ maintains a set of all linear relations $L_0 \subseteq \mathbb{F}_q^{\mu_0}$ on $\mathfrak{c}$ found so far. For each message m then $\mathcal{A}$ proceeds as follows:

- First, for all strings t_1 and all spaces of linear relations $L_1 \subseteq \mathbb{F}_q^{\mu_0} \times \mathbb{F}_q^{\mu_1}$ consistent with L_0,[3] $\mathcal{A}$ simulates the verifier as described above with L_1 and input $t_0||t_1$. Note there may be an exponential number of such L_1's and t_1's, but since simulation requires no actual group operation, this does not affect the efficiency of $\mathcal{A}$ in the GGM.
  If no pair (L_1, t_1) such that the simulated verifier accepts is found, $\mathcal{A}$ gives up and requests a signature $\sigma = (\mathfrak{a}, t_1)$ for m. In this case the space of all relations the verifier finds when validating σ must contain a new relation on $\mathfrak{c}$ (or else an accepting pair (L_1, t_1) would have been found).
- Conversely, if an accepting (L_1, t_1) is found, $\mathcal{A}$ samples a uniformly random $\mathfrak{a} \in \mathbb{G}^{\mu_1}$, conditioned on $\mathfrak{c}||\mathfrak{a}$ satisfying all relations in L_1. This can be done with polynomially many group operations given only $\mathfrak{c}$ and L_1. Next, $\mathcal{A}$ checks whether $\sigma = (\mathfrak{a}, t_1)$ is a valid signature. If so, then $\mathcal{A}$ successfully found a forgery σ for m. Otherwise, the verifier must have found a new relation $\mathbf{x}$ on $\mathfrak{c}||\mathfrak{a}$ not in L_1. Finally, due to $\mathbf{x}$ and $\mathfrak{a}$ being independent, with high probability $\mathbf{x}$ is of the form $\mathbf{x}_0||0^{\mu_0}$, i.e., we found a new relation $\mathbf{x}_0 \notin L_0$ on $\mathfrak{c}$.

For showing that non-interactive VC is impossible in the GGM, the argument in [CFGG22] is closely related. The idea there is to compile a given VC into an artificial signature with large enough message space, where the verification key corresponds to the VC public parameters and commitment to a vector $\mathbf{m} \in \{0,1\}^n$, while signatures for a message $i \in \{1, \ldots, n\}$ correspond to the opening in position i.

Issues in the interactive settings. Although the above strategy works well against non-interactive signatures and VCs, the interactive setting is significantly more challenging. Indeed, since public-coin "interactive signatures" based on the GGM exist, we cannot hope to rely on a generic VC-to-signature compiler as in [CFGG22].

Concretely, the issue is that when reducing to the signature's security, our adversary is only allowed to query signatures on a given message, with no control over the random challenges.

Our approach through a compression argument. To circumvent the issue above, we refine the strategy in [CFGG22] through a compression argument. Introduced implicitly in [GT00], a compression argument is an information-theoretical technique where, assuming towards contradiction the existence of

[3] By consistent here we mean that the relations in L_1 involving only verification key elements are the same as those in L_0. Formally, $L_0 \times \{0^{\mu_1}\} = L_1 \cap \mathbb{F}_q^{\mu_0} \times \{0^{\mu_1}\}$.

a particularly efficient instantiation of a given primitive, one can define a computationally unbounded encoder-decoder pair $(\mathcal{E}, \mathcal{D})$ capable of transmitting a message $\mathbf{m}$ with less that $\mathrm{H}_\infty(\mathbf{m})$ bits. This applies more generally to $(\mathcal{E}, \mathcal{D})$ which can share the same random tape (of arbitrary length), and are allowed to interact during their execution.

Applying this to our setting, assuming existence of an efficient set accumulator[4], we will construct an interactive encoder-decoder pair $(\mathcal{E}, \mathcal{D})$ such that: $\mathcal{E}$ encodes a message $\mathbf{m}$ by first converting it into a set $S = \{(i, m_i) : i \in [n]\}$, computing a commitment cm to S, and sending cm to $\mathcal{D}$. Then, for all $i \in [n]$, $\mathcal{D}$ will attempt to either (1) extract m_i through a Forge procedure, without talking to $\mathcal{E}$, or (2) query $\mathcal{E}$ for m_i and afterwards extract a linear relation on the group elements in pp and in cm through an $\mathsf{Extract}$ procedure (which requests some costly information from $\mathcal{E}$).

Importantly, the amount of communication inside $\mathsf{Extract}$ must be small enough so that $\mathbf{m}$ is actually being *compressed*. We discuss these two procedure separately.

Extract procedure. The key idea for our extract procedure is to mirror a special-soundness extractor strategy by collecting a tree of transcripts. Specifically, to fix notation, say that $\mathfrak{p}||\mathfrak{c} \in \mathbb{G}^{\mu_0}$ are the group elements in the public parameters pp and digest cm respectively, while an opening transcript (for proving $(i, m_i) \in S$) consists of $w_1, c_1, \ldots, w_\rho, c_\rho$ with $w_i = (\mathfrak{a}_i, t_i) \in \mathbb{G}^{\mu_i} \times \{0,1\}^{\eta_i}$ being the i-th message sent from the prover to the verifier, and $c_i \leftarrow^{\$} \mathcal{C}_i$ being i-th challenge chosen by the verifier after receiving w_i.

Given an accepting transcript with group elements $\mathfrak{p}||\mathfrak{c}, \mathfrak{a}_1, \ldots, \mathfrak{a}_{\rho-1}, \mathfrak{a}_\rho$, $\mathsf{Extract}$ maintains a sequence of consistent[5] spaces $L_0, \ldots, L_{\rho-1}$ where L_i contains all relations found so far on $\mathfrak{p}||\mathfrak{c}||\mathfrak{a}_1|| \ldots ||\mathfrak{a}_i$. Then, it performs essentially the same attack as in [DHH+21], treating $(\mathsf{pp}, \mathsf{cm}, w_1, \ldots, w_{\rho-1})$ as the verification key, the challenge space $\mathcal{C}_{\rho-1}$ as the message space, and w_ρ as the signature. If successful, $\mathsf{Extract}$ finds a new relation among group elements $\mathfrak{p}||\mathfrak{c}||\mathfrak{a}_1|| \ldots ||\mathfrak{a}_j$ for some $j < \rho$. Thus, $\mathsf{Extract}$ updates its space L_j with the new relation, and proceeds with requesting a new transcript with freshly sampled challenges $c'_{j+1}, \ldots, c'_\rho$.

Informally, we show that if all of its intermediate attempts succeed, then $\mathsf{Extract}$ finds a new relation among $\mathfrak{p}||\mathfrak{c}$ (i.e., a relation solely among those group elements from pp and cm) using at most $\prod_{i=1}^{\rho-1}(\mu_i + 1)$ transcripts, where μ_i is the number of group elements in $w_i = (\mathfrak{a}_i, t_i)$.

To extract or not to extract? In light of our compression argument, $\mathsf{Extract}$ as described above is very expensive. Not only does $\mathcal{D}$ have to request m_i from

[4] Recall we will prove our bound for set accumulator, as opposed to Vector Commitments, for the sake of generality.

[5] As before, we say $L_i \subseteq \mathbb{F}_q^{n_i}$ is consistent with $L_{i+1} \subseteq \mathbb{F}_q^{n_{i+1}}$ if $L_i \times \{0^{n_{i+1}-n_i}\} = L_{i+1} \cap (\mathbb{F}_q^{n_i} \times \{0^{n_{i+1}-n_i}\})$ for $n_i \leq n_{i+1}$.

$\mathcal{E}$, but it must also rely on $\mathcal{E}$ transmitting $\prod_{i=1}^{\rho-1}(\mu_i+1)$ transcripts. Therefore it is crucial to devise a criteria to determine when extractions are performed in such a way that

1. when $\mathsf{Extract}$ is executed, the probability for it to find a new linear relation must be very high;
2. when Forge instead of $\mathsf{Extract}$ is executed, there is a very high probability for correctly predicting m_i (or, alternatively, to find a new linear relation on $\mathfrak{p}||\mathfrak{c}$ through other means) without communicating with $\mathcal{E}$.

Identifying such criteria is among our main technical contributions. Specifically, we do so via an information theoretical game: Fix $L_0 \subseteq \mathbb{F}_q^{\mu_0}$ (where μ_0 is the number of group elements in $\mathfrak{p}||\mathfrak{c}$), the bit-string t_0 that comprises both the bit-string contained in pp and that contained in cm, and $s = (i, m_i)$. In each round $i \in [\rho]$ of this game, the first player sends (L_i, t_i) and the second one replies with a challenge c_i. The game ends after ρ rounds and we say that the first player wins if all the spaces $L_0, \dots L_\rho$ are consistent and if the accumulator's verifier would accept when simulated with L_ρ.

Having defined this game, we let $\mathcal{D}$ perform an extraction on i only when the winning probability p of this game is very low for both $s = (i, 0)$ and $s = (i, 1)$. In the main body, we prove that the first of the two criteria is now satisfied.

Note that while estimating the winning probability of the above game requires exponential time, it does not involve any group operation or communication with the encoder, and therefore does not affect our information-theoretical argument.

Forge procedure. To conclude, we must equip $\mathcal{D}$ with a procedure Forge to extract m_i when the winning probability for $(L_0, t_0), (i, b)$ for some $b \in \{0, 1\}$ is greater than p. Note that in this case there exists an optimal strategy $(\bar{L}_i, \bar{t}_i)$ for the first player, where $\bar{L}_i, \bar{t}_i$ are functions of the previous challenges $c_1, \dots, c_{i-1}$.

Given this strategy, Forge can use it to simulate an attack against the accumulator's security as follows. For each step j in the interactive protocol, given $c_1, \dots, c_{j-1}$ it computes the next optimal $\bar{L}_j, \bar{t}_j$ and samples $\mathfrak{a}_j$ uniformly conditioned on $\mathfrak{p}||\mathfrak{c}||\mathfrak{a}_1|| \dots ||\mathfrak{a}_j$ satisfying all relations in $\bar{L}_j$ and stores the proof message $w_i = (\mathfrak{a}_i, t_i)$. With probability p, it will eventually end with a transcript containing group elements $\mathfrak{p}, \mathfrak{c}, \mathfrak{a}_1, \dots, \mathfrak{a}_\rho$ satisfying all relations in $\bar{L}_\rho$, and the accumulator's verifier accepting when simulated on this transcript, set element (i, b) and relations L_ρ. At this point one of two cases may occur.

- The verifier accepts the transcript $w_1, c_1, \dots w_\rho, c_\rho$, which confirms that $m_i = b$ is indeed true. (Otherwise, note that we have constructed a successful attack against the accumulator!)
- The verifier rejects the transcript. Then we show that the verifier must have found a new relation on $\mathfrak{p}||\mathfrak{c}$ with high probability.

Ignoring Public Parameters. Finally, we note that the number of $\mathsf{Extract}$'s executed by our decoder is upper bounded by the number of linear relations it

finds on $\mathfrak{p}||\mathfrak{c}$. While the digest cm (and hence also $\mathfrak{c}$) is expected to be short if the accumulator is succinct, the same may not be true for the public parameters $\mathfrak{p}$. To ensure we only run $\mathsf{Extract}$ at most as many times as group elements in cm, we add an initial exploration step to the decoder, where it simulates its interaction with the encoder on a randomly sampled message $\mathbf{m}'$, each time storing relations the execution found only pertaining $\mathfrak{p}$.

Calling ϑ_0 the number of group elements in $\mathfrak{p} \in \mathbb{G}^{\vartheta_0}$, then a new relation on $\mathfrak{p}$ can be found at most ϑ_0 times. Hence, the idea is to run $i \cdot \vartheta_0$ simulated executions in the exploration step, where $i \leftarrow^{\$} [C \cdot \vartheta_0]$. Doing so ensures that there is a $\leq 1/C$ probability of finding a new relation on $\mathfrak{p}$ during the real execution between $\mathcal{E}$ and $\mathcal{D}$, where C can be an arbitrarily large polynomial.

1.3 Related Works

Lower Bounds in the Generic Group Model. The study of the limitations of cryptographic primitives in generic models of computation is a rich and active area of research. Negative results have been provided in the pairing-free known order setting for Identity Based Encryption in both Maurer's GGM [PRV12, SGS21] and Shoup's GGM [Zha22] model; order revealing encryption [ZZ18]; Verifiable Delay Functions [RSS20]; digital signatures [DHH+21]; non-interactive zero-knowledge proofs [Giu23]; round-optimal blind-signatures in the GGM+ROM model [DKT26], and for Registration Based Encryption in Shoup's GGM [HMQS23]. Generic model of computations have also been used in black-box separations for other primitives, e.g. in [GM24] for separating sequential function from one-way functions.

[SGS20] provides a lower bound on the time-complexity of batch verification in generic groups possibly of unknown order. Their bound for batch of size 1 implies $\Omega(\lambda/\log\lambda)$ operations are needed in general, with no direct implication for membership proof length.

Most directly related to our work are the results from [CFGG22], where it is shown that for any (non-interactive) vector commitment in Maurer's GGM, when committing to a vector of n elements, then $|\mathsf{cm}| \cdot |w| = \Omega(n)$ (with $|\mathsf{cm}|$ denoting the commitment's bit length and $|w|$ the opening's bit length). We remark that our main result, applied to the special case of a 1-round accumulator/VC, directly implies $|\mathsf{cm}| \cdot |w| = \Omega(n)$, and therefore our result subsumes that of [CFGG22].

We further remark that our approach of establishing lower bounds of a primitive, *restricted to the case where it is derived using a Fiat-Shamir transform*, has already previously appeared in [BGZ23], where lower bounds for identification protocols and signatures from generic *group actions* are provided.

Lower Bounds for Vector Commitments. (Vector) commitments and accumulators, even beyond the scope of algebraic construction based on the GGM, have been extensively studied in terms of inherent efficiency limitations. [AHO12] has shown how structure preserving commitments do not shrink, which does not apply in our case as we explicitly restrict the universe to bit strings only. A recently

very active line of research studied lower bounds for the update frequency in dynamic accumulators/vector commitments in various regimes [CH10, BCD+17, CB23, BCCK25, AAAG25, Qi25]. The focus in all these works is on the efficiency of update operations, which is an orthogonal to our problem, namely bounding the membership proof length. Moreover, we stress that our bound is proven assuming only static accumulators for the sake of generality.

2 Preliminaries

2.1 Notation

Given a set X, we write X^* to denote $\cup_{n\in\mathbb{N}} X^n$. Given sets $X_1, \ldots, X_n$, an element $\mathbf{x} \in \prod_{j=1}^n X_j$ is written in bold font, and we use $x_1, \ldots, x_n$ to denote its elements. Given $\mathbf{x} \in \prod_{j=1}^n X_j$ and $\mathbf{x}' \in \prod_{j=1}^n X'_j$, we write $\mathbf{x}||\mathbf{x}' = (x_1, \ldots, x_n, x'_1, \ldots, x'_n)$ to denote their concatenation. We use $\pi_i : \prod_{j=1}^n X_j \to \prod_{j=1}^i X_j$ to denote the projection over the first i components, i.e., $\pi_i(\mathbf{x}) = (x_1, \ldots, x_i)$. Similarly, π_{-i} denotes the projection over the last i components, i.e., $\pi_{-i}(\mathbf{x}) = (x_{n-i+1}, \ldots, x_n)$.

We use $(x == y)$ to denote the indicator function of "$x = y$", i.e., it is 1 if $x = y$ and 0 otherwise. Similarly, we use $(x \in S)$ to denote the indicator function of "$x \in S$", i.e., it is 1 if $x \in S$ and 0 otherwise.

In pseudocode, we use "$\leftarrow$" for deterministic assignment or for assignment of a value computed by a (probabilistic) machine, and we use "$\leftarrow^{\$}$" for sampling a uniformly random element from a given set. Running a machine $\mathcal{M}$ on input x with randomness r is denoted by $\mathcal{M}(x; r)$. Writing only $\mathcal{M}(x)$ means that the random tape r is chosen uniformly at random.

Groups. In this work we only consider prime-order groups. We always write the group operation *additively*, and its elements are always written in mathfrak font (e.g. $\mathfrak{g}$). When a group $\mathbb{G}$ of order q is clear from context, we denote its neutral element by $\mathfrak{o}$. Given $r \in \mathbb{N}$, we define $r \cdot \mathfrak{g}$ as the r-fold sum of $\mathfrak{g}$ with itself. Note that $\mathbb{G}^n$ forms an $\mathbb{F}_q$-vector space for any $n \in \mathbb{N}$.

Vector spaces. Given any vector space V, we write $W \leq V$ to denote that W is a subspace of V. $\mathsf{Sub}(V) := \{W : W \leq V\}$ denotes the lattice of subspaces of V. Given vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$, we write $\mathsf{span}(\mathbf{v}_1, \ldots, \mathbf{v}_n) \leq V$ to denote the linear span of those vectors. If $V = \mathbb{F}_q^n$ is clear from context, we use $\mathbf{e}_1, \ldots, \mathbf{e}_n$ to denote the canonical basis vectors.

We define $\pi^*_m : \mathsf{Sub}(\mathbb{F}_q^n) \to \mathsf{Sub}(\mathbb{F}_q^m)$ as the slice map $\pi^*_m(V) = \pi_m(V \cap \mathbb{F}_q^m \times \{0^{n-m}\})$, i.e., $\pi^*_m(V)$ is the vector space consisting of all vectors in V for which the last $n - m$ coordinates are zero, with their last $n - m$ coordinates removed.

Given vector space $V \leq \mathbb{F}_q^n$ and a group $\mathbb{G}$ of prime order q, we define the annihilator $\mathrm{Ann}_{\mathbb{G}}(V) = \{\mathfrak{a} \in \mathbb{G}^n \mid \forall \mathbf{v} \in V, \mathbf{v} \cdot \mathfrak{a} = \mathfrak{o}\}$, where $\cdot$ denotes the inner product. Note that $\mathrm{Ann}_{\mathbb{G}}(V)$ is a subspace of $\mathbb{G}^n$.

2.2 Entropy

Definition 1. *Given X, Y random variables with support in $\mathcal{X}$ and $\mathcal{Y}$, we define the min-entropy of X and the average min-entropy of X given Y respectively as*

$$\mathrm{H}_\infty(X) \coloneqq -\log\left(\max_{x\in\mathcal{X}} \Pr[X=x]\right)$$

$$\mathrm{H}_\infty(X|Y) \coloneqq -\log\left(\sum_{y\in\mathcal{Y}} \Pr[Y=y]\cdot\max_{x\in\mathcal{X}} \Pr[X=x\,|\,Y=y]\right)$$

Moreover, given an event E we define the min-entropy of X conditioned on E and the average min-entropy of X given Y conditioned on E respectively as

$$\mathrm{H}_\infty(X|E) \coloneqq -\log\left(\max_{x\in X} \Pr[X=x\,|\,E]\right)$$

$$\mathrm{H}_\infty(X|Y;E) \coloneqq -\log\left(\max_{y\in Y} \Pr[Y=y\,|\,E]\cdot\max_{x\in X} \Pr[X=x\,|\,Y=y,\ E]\right)$$

In what follows we further recall the definition of max-entropy or Hartley-entropy. For the conditional case, we provide the following formulation due to their role in Lemma 1.

Definition 2. *Given random variable X, Y distributed over $\mathcal{X}, \mathcal{Y}$ respectively and an event E, we define the max entropy of X (resp. conditioned to Y, to E and to Y, E) as*

$$\mathrm{H}_0(X) = \log|\{x\in\mathcal{X} \,:\, \Pr[X=x]>0\}|$$

$$\mathrm{H}_0(X|Y) = \log\left(\max_{y\in\mathcal{Y}}|\{x\in\mathcal{X} \,:\, \Pr[X=x\,|\,Y=y]>0\}|\right)$$

$$\mathrm{H}_0(X|E) = \log|\{x\in\mathcal{X} \,:\, \Pr[X=x\,|\,E]>0\}|$$

$$\mathrm{H}_0(X|Y;E) = \log\left(\max_{y\in\mathcal{Y}}|\{x\in\mathcal{X} \,:\, \Pr[X=x\,|\,Y=y,\,E]>0\}|\right)$$

Lemma 1 ([DORS08, §2.4], [ABG+25, §2.8]). *Given X, Y, Z random variables with finite support in $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ respectively, and E an event. Then*

1. $\mathrm{H}_\infty(X|E) \le \log(|\mathcal{X}_E|)$ *where* $\mathcal{X}_E = \{x\in\mathcal{X} \,:\, \Pr[X=x|E]>0\}$;
2. $\mathrm{H}_\infty(X|Y;E) \ge \mathrm{H}_\infty(X|Y,Z;E)$;
3. $\mathrm{H}_\infty(X,Y|Z;E) \ge \mathrm{H}_\infty(X|Z;E)$;
4. $\mathrm{H}_\infty(X|Y,Z;E) \ge \mathrm{H}_\infty(X,Y|Z;E) - \log\max_{z\in\mathcal{Z}}|\mathcal{Y}_{z,E}|$ *where*
$$\mathcal{Y}_{z,E} = \{y\in\mathcal{Y} \,:\, \Pr[Y=y|Z=z,\ E]>0\}.$$
5. $\mathrm{H}_\infty(X|Y;E) = \mathrm{H}_\infty(X|E)$ *if X, Y are mutually independent given E*
6. $\mathrm{H}_\infty(X|Y;E) \ge \mathrm{H}_\infty(X|Y) + \log\Pr[E]$

Lemma 2 (Guessing Lemma). *Given X, Y, E as in Lemma 1, then for any function $\phi : \mathcal{Y}\to\mathcal{X}$*

$$\Pr[\phi(Y)=X\,|\,E] \le 2^{-\mathrm{H}_\infty(X|Y;E)}.$$

2.3 Compression Arguments

Compression arguments, introduced implicitly in [GT00], is a proof strategy particularly effective to show efficiency lower bounds for a targeted primitive. Assuming toward contradiction that a very efficient instantiations exists, one uses it then to realize a *computationally unbounded* encoder-decoder pair: the encoder receives a random messages and attempt to compress it. The decoder instead attempts to recover the given message. Contradiction is reached if the input message can be compressed beyond its entropy. We will eventually phrase our result in a more general setting, where $\mathcal{E}, \mathcal{D}$ are further allowed to communicate and share common random coins (independent from the input message). This relaxation does not technically yield a stronger proof technique, but greatly simplifies the exposition. Below we provide a formal definition for an encoder-decoder pair, and the core information-theoretically lemma we will base our main result on.

Definition 3. *Let $(\mathcal{E}, \mathcal{D})$ be a pair of deterministic machines $\mathcal{E}, \mathcal{D}$ with input space $Z_{\mathcal{E}} \times Q \subseteq \{0,1\}^* \times \{0,1\}^*$ and $Z_{\mathcal{D}} \subseteq \{0,1\}^*$, respectively, s.t. $\mathcal{D}$ additionally has access to some oracle with input set Q. Then, if for any $(z_1, z_2) \in Z_{\mathcal{E}} \times Z_{\mathcal{D}}$ the execution $\mathcal{D}^{\mathcal{E}(z_1,\cdot)}(z_2)$ terminates, $(\mathcal{E}, \mathcal{D})$ is called an encoder-decoder pair.*

Furthermore, we denote by $\chi_{\mathcal{E},\mathcal{D}}(z_1, z_2)$ the total number of queries made by $\mathcal{D}$ when running $\mathcal{D}^{\mathcal{E}(z_1,\cdot)}(z_2)$. If y_i is the i-th query made by $\mathcal{D}^{\mathcal{E}(z_1,\cdot)}(z_2)$ and $x_i = \mathcal{E}(z_1, y_i)$ the corresponding response, then we define $\mathsf{len}_{\mathcal{E},\mathcal{D}}(z_1, z_2) = \sum_{i=1}^{\chi} |x_i|$, i.e., the total length of all responses provided by the encoder $\mathcal{E}$.

To capture the shared randomness setting, it is meaningful to consider $Z_{\mathcal{E}} = M \times R$, with M message space and R random coins set, and $Z_{\mathcal{D}} = R$. If for a random (m, r), the decoder guesses m only given r and access to $\mathcal{E}((m, r), \cdot)$, then the incoming communication form $\mathcal{E}$ should match the min-entropy of m. For completeness, a proof the following Lemma appears in Appendix 2.2.

Lemma 3. *Let $(\mathcal{E}, \mathcal{D})$ be an encoder-decoder pair with input spaces $M \times R$ and R, respectively. For (m, r) uniformly sampled and E an event[6] such that E implies $\mathcal{D}^{\mathcal{E}((m,r),\cdot)}(r) = m$, then*

$$\log |M| \;\leq\; 1 + \max_{(m,r) \in E} \mathsf{len}_{\mathcal{E},\mathcal{D}}((m, r), r) - \log \Pr[E]$$

2.4 Generic and Type-Safe Algorithms

In order to model black-box usage of the underlying group, we rely on Maurer's GGM [Mau05]. Here, group elements are treated as syntactically different objects than bit strings, and the only operations allowed are additions and equality tests. In [Mau05] this is captured by equipping generic algorithms with two stateful

[6] Formally, given that the probability space is defined over $M \times R$ with the normalized counting measure, E is a subset of $M \times R$ and $(m_0, r_0) \in E$ iff $\Pr[E \mid (m, r) = (m_0, r_0)] = 1$.

oracles performing those operations, without providing any representation for the group elements involved. [Zha22, §3.3] instead proposed the notion of *Type-Safe* circuits, where an explicit representation for group elements is provided, but computation on them only happens via addition or equality gates.

Compromising between these two approaches, we will adopt the same approach as [DKT26]: we distinguish between a *generic* algorithm (that has oracle access to group operations) and a *type-safe* algorithm, which is essentially a generic algorithm instantiated with a specific group (hence, the type-safe algorithm does not require a group operation oracle anymore).

Definition 4. *A generic (or GGM) algorithm is a probabilistic oracle machine $\mathcal{M}$ with access to two oracles* add *and* eq*, input in $\mathbb{N} \times \mathbb{N} \times \{0,1\}^*$ and output in $\mathbb{N}^* \times \{0,1\}^*$. We further say $\mathcal{M}$ is:*

- *χ-queries if $\mathcal{M}^{\mathsf{add},\mathsf{eq}}(q,n,x)$ performs at most $\chi(q,n,x)$ queries;*
- *query-efficient if it is χ-queries with $\chi(q,n,x)$ being a polynomial in $\log q$, n, and $|x|$;*
- *bits-output if its output is in $\{0,1\}^*$.*

Definition 5. *Let $\mathcal{M}$ be a GGM algorithm, and let $\mathbb{G}$ of prime order q. Given $(\mathfrak{a}, x) \in \mathbb{G}^* \times \{0,1\}^*$ and random coins $r \in \{0,1\}^*$, the* execution *of $\mathcal{M}$ is defined as $\mathsf{Exec}_{\mathcal{M}}(q, \mathfrak{a}, x; r)$, as detailed in Figure 1.*

$\mathsf{Exec}_{\mathcal{M}}(q, \mathfrak{a}, x; r)$: // $\mathfrak{a} = (\mathfrak{a}_1, \ldots, \mathfrak{a}_n) \in \mathbb{G}^n$

Run $\mathcal{M}^{\mathsf{add},\mathsf{eq}}(q, n, x; r)$
When $\mathcal{M}$ queries $\mathsf{add}(i, j)$:
  If $i, j \in \{1, \ldots, n\}$:
    $\mathfrak{a}_{n+1} \leftarrow \mathfrak{a}_i + \mathfrak{a}_j$, $n \leftarrow n + 1$
When $\mathcal{M}$ queries $\mathsf{eq}(i, j)$:
  If $i, j \in \{1, \ldots, n\}$:
    Reply $(\mathfrak{a}_i == \mathfrak{a}_j)$
  Else: **Reply** $\perp$
When $\mathcal{M}$ returns $(i_1, \ldots, i_m, y)$:
  If $i_j \in \{1, \ldots, n\}$ $\forall j \in \{1, \ldots, m\}$:
    Return $(\mathfrak{a}_{i_1}, \ldots, \mathfrak{a}_{i_m}, y)$
  Else: **Return** $\perp$

Fig. 1. Execution of a GGM algorithm $\mathcal{M}$ using a group $\mathbb{G}$ of prime order q.

Definition 6. *A probabilistic machine $\mathcal{T}$ is type-safe w.r.t. some group $\mathbb{G}$ of prime order q if there exists a GGM-algorithm $\langle\mathcal{T}\rangle$ such that for every input $(\mathfrak{a}, x) \in \mathbb{G}^* \times \{0,1\}^*$ and random coins $r \in \{0,1\}^*$, it holds that:*

$$\mathcal{T}(q, \mathfrak{a}, x; r) \;=\; \mathsf{Exec}_{\langle\mathcal{T}\rangle}(q, \mathfrak{a}, x; r).$$

We further say $\mathcal{T}$ is respectively χ-query, query-efficient, or bits-output, whenever there exists a choice of $\langle\mathcal{T}\rangle$ for which $\langle\mathcal{T}\rangle$ is χ-query, query-efficient, or bits-output.[7]

For proving lower bounds, it will be important to execute a GGM algorithm $\mathcal{M}$ using different "modes" that allow either simulating it (for free) or learning the relations that its execution implies about the input $\mathfrak{a}$:

Definition 7. *Let $\mathcal{M}$ be a GGM algorithm, and let $\mathbb{G}$ be a group of prime order q. Given input $(\mathfrak{a}, x) \in \mathbb{G}^n \times \{0,1\}^*$ and random coins $r \in \{0,1\}^*$, the* implied vector space of relations $\mathsf{Rel}_{\mathcal{M}}(q, \mathfrak{a}, x; r)$ *(which is a subset of $\mathbb{F}_q^n$) is defined on the right-hand side of Figure 2.*

If $\mathcal{M}$ further is bits-output, then given a vector space $R \leq \mathbb{F}_q^n$, bitstring $x \in \{0,1\}^*$ and random coins $r \in \{0,1\}^*$, the simulation of $\mathcal{M}$ according to R *is defined as $\mathsf{Sim}_{\mathcal{M}}(q, R, x; r)$, as detailed on the left-hand side of Figure 2.*

$\mathsf{Sim}_{\mathcal{M}}(q, R, x; r)$: // $R \leq \mathbb{F}_q^n$

1 : $\mathbf{v}_i \leftarrow \mathbf{e}_i \in \mathbb{F}_q^n$ for $i \in \{1, \ldots, n\}$
2 :
3 : Run $\mathcal{M}^{\mathsf{add},\mathsf{eq}}(q, n, x; r)$
4 : **When** $\mathcal{M}$ queries $\mathsf{add}(i, j)$:
5 : **If** $i, j \in \{1, \ldots, n\}$:
6 : $\mathbf{v}_{n+1} \leftarrow \mathbf{v}_i + \mathbf{v}_j$, $n \leftarrow n + 1$
7 : **When** $\mathcal{M}$ queries $\mathsf{eq}(i, j)$:
8 : **If** $i, j \in \{1, \ldots, n\}$:
9 : **Reply** $(\mathbf{v}_i - \mathbf{v}_j \in R)$
10 :
11 :
12 : **Else**: **Reply** $\perp$
13 : **When** $\mathcal{M}$ returns $y \in \{0,1\}^*$:
14 : **Return** y

$\mathsf{Rel}_{\mathcal{M}}(q, \mathfrak{a}, x; r)$: // $\mathfrak{a} \in \mathbb{G}^n$

1 : $\mathbf{v}_i \leftarrow \mathbf{e}_i \in \mathbb{F}_q^n$ for $i \in \{1, \ldots n\}$
2 : $R \leftarrow \{0^n\}$
3 : Run $\mathcal{M}^{\mathsf{add},\mathsf{eq}}(q, n, x; r)$
4 : **When** $\mathcal{M}$ queries $\mathsf{add}(i, j)$:
5 : **If** $i, j \in \{1, \ldots, n\}$:
6 : $\mathbf{v}_{n+1} \leftarrow \mathbf{v}_i + \mathbf{v}_j$, $n \leftarrow n + 1$
7 : **When** $\mathcal{M}$ queries $\mathsf{eq}(i, j)$:
8 : **If** $i, j \in \{1, \ldots, n\}$:
9 : **Reply** $(\mathbf{v}_i^T \cdot \mathfrak{a} == \mathbf{v}_j^T \cdot \mathfrak{a})$
10 : **If** $\mathbf{v}_i^T \cdot \mathfrak{a} = \mathbf{v}_j^T \cdot \mathfrak{a}$:
11 : $R \leftarrow R + \mathsf{span}(\mathbf{v}_i - \mathbf{v}_j)$
12 : **Else**: **Reply** $\perp$
13 : **When** $\mathcal{M}$ halts:
14 : **Return** R

Fig. 2. Simulation of a GGM algorithm $\mathcal{M}$ according to a vector space $R \leq \mathbb{F}_q^n$ (left) and the vector space of relations implied by running a GGM algorithm $\mathcal{M}$ (right).

[7] Note that the bits-output property is independent of the choice of $\langle\mathcal{T}\rangle$, while the other two properties are not.

We now state a variety of properties referring to the notions defined above. Many of them have been used in implicit form in previous works [DHH+21, CFGG22, DKT26]. We give full proofs for all of these lemmas in Appendix A.2.

Lemma 4. *Consider any GGM algorithm $\mathcal{M}$, a group $\mathbb{G}$ of prime order q, and any input $(\mathfrak{a}, x) \in \mathbb{G}^* \times \{0,1\}^*$. Then, denoting*

$$R \leftarrow \mathsf{Rel}_{\mathcal{M}}(q, \mathfrak{a}, x; r) \ ,$$

we have that $\mathfrak{a} \in \mathrm{Ann}_{\mathbb{G}}(R)$.

Proof. By construction, the output R of Rel is $R = \mathsf{span}(\mathbf{w}_1, \ldots, \mathbf{w}_k)$ for some vectors $\mathbf{w}_i$ with $\mathbf{w}^T \cdot \mathfrak{a} = \mathfrak{o}$. Hence, by linearity, $\mathfrak{a} \in \mathrm{Ann}_{\mathbb{G}}(R)$. □

Lemma 5. *Consider any bits-output GGM algorithm $\mathcal{M}$, a group $\mathbb{G}$ of prime order q, any input $(\mathfrak{a}, x) \in \mathbb{G}^n \times \{0,1\}^*$, random tape $r \in \{0,1\}^*$ and vector space $R \leq \mathbb{F}_q^n$, and denote*

$$y = \mathsf{Exec}_{\mathcal{M}}(q, \mathfrak{a}, x; r), \quad R' \leftarrow \mathsf{Rel}_{\mathcal{M}}(q, \mathfrak{a}, x; r), \quad y' \leftarrow \mathsf{Sim}_{\mathcal{M}}(q, R, x; r).$$

Whenever both $R' \subseteq R$ and $\mathfrak{a} \in \mathrm{Ann}_{\mathbb{G}}(R)$ hold, then $y = y'$. In particular, $R' = R$ implies $y = y'$.

Lemma 6. *Fix any χ-queries bits-output GGM algorithm $\mathcal{M}$, a group $\mathbb{G}$ of prime order q, a vector space $R \leq \mathbb{F}_q^{n+m}$ (with $n, m \in \mathbb{N}$), group elements $\mathfrak{a} \in \mathrm{Ann}_{\mathbb{G}}(\pi_n^*(R))$, and strings $x, r \in \{0,1\}^*$ let $\mathfrak{b}$ be uniformly distributed over $\{\mathfrak{b} \in \mathbb{G}^m \mid (\mathfrak{a}||\mathfrak{b}) \in \mathrm{Ann}_{\mathbb{G}}(R)\}$. Then, calling $y = \mathsf{Exec}_{\mathcal{M}}(q, \mathfrak{a}||\mathfrak{b}, x; r)$, $R' = \mathsf{Rel}_{\mathcal{M}}(q, \mathfrak{a}||\mathfrak{b}, x; r)$ and $y' = \mathsf{Sim}_{\mathcal{M}}(q, R, x; r)$, we get*

$$\Pr\left[y = y' \ \vee \ \pi_n^*(R + R') \nsubseteq \pi_n^*(R)\right] \ \geq \ 1 - \frac{\chi(q, n+m, x)}{q}.$$

Lemma 7. *Given any group $\mathbb{G}$ of prime order q, vector spaces $R_i \leq \mathbb{F}_q^{n+\sum_{j=1}^{i} m_j}$ for $i \in \{0, \ldots, \ell\}$ (with $n, m_1, \ldots, m_\ell \in \mathbb{N}$) such that $\pi^*_{n+\sum_{j=1}^{i-1} m_j}(R_i) = R_{i-1}$ for all $i \in \{1, \ldots, \ell\}$, as well as group elements $\mathfrak{b}_0 \in \mathrm{Ann}_{\mathbb{G}}(R_0)$, the following two distributions are identical:*

- *For i from 1 to ℓ, sample $\mathfrak{b}_i$ uniformly conditioned on $(\mathfrak{b}_0||\ldots||\mathfrak{b}_{i-1}||\mathfrak{b}_i) \in \mathrm{Ann}_{\mathbb{G}}(R_i)$. Then output $(\mathfrak{b}_0||\ldots||\mathfrak{b}_\ell)$.*
- *Sample $(\mathfrak{b}_1||\ldots||\mathfrak{b}_\ell)$ uniformly conditioned on $(\mathfrak{b}_0||\ldots||\mathfrak{b}_\ell) \in \mathrm{Ann}_{\mathbb{G}}(R_\ell)$, and output $(\mathfrak{b}_0||\ldots||\mathfrak{b}_\ell)$.*

Lemma 8. *For any group $\mathbb{G}$ of prime order q with generator $\mathfrak{g}$, there exists a $O(n \cdot m \cdot \log q)$-queries type-safe probabilistic machine $\mathcal{T}$, s.t. given input $(\mathfrak{g}||\mathfrak{a}, (n, m, R)) \in \mathbb{G}^{n+1} \times \{0,1\}^*$[8] (with a vector space $R \leq \mathbb{F}_q^{n+m}$ s.t. $\mathfrak{a} \in \mathrm{Ann}_{\mathbb{G}}(\pi_n^*(R))$), the output distribution is identical to the uniform distribution over $\{\mathfrak{b} \in \mathbb{G}^m \mid (\mathfrak{a}||\mathfrak{b}) \in \mathrm{Ann}_{\mathbb{G}}(R)\}$.*

[8] We abuse notation to denote with (n, m, R) an arbitrarily chosen encoding in $\{0,1\}^*$ of the triple $(n, m, R) \in \mathbb{N} \times \mathbb{N} \times \mathsf{Sub}(\mathbb{F}_q^{n+m})$.

2.5 Interactive Type-Safe Accumulators

In the following we adapt the definition of cryptographic accumulators from [FN02, BCY20]. Specifically, as we are interested in black-box construction based entirely on the hardness of a prime-order group, we restrict all primitives to be type-safe, and require security only to hold against query-efficient adversaries, but otherwise unbounded.[9] Moreover, since our goal is to capture only constructions based on Fiat-Shamir [FS87], we allow membership proofs to be interactive but also require them to be public-coin.

Definition 8. *A type-safe $\rho(\lambda, n)$-rounds public-coin interactive accumulator* A *consists of a group $\mathbb{G} = \mathbb{G}(\lambda)$ whose prime order $q = q(\lambda)$ fulfills $q = 2^{\Theta(\lambda)}$, a generator $\mathfrak{g} \in \mathbb{G}$, polynomially bounded functions $\rho, \vartheta_0, \vartheta_1, \zeta_0, \zeta_1, (\mu_i, \eta_i, \ell_i)_{i \in \mathbb{N} \setminus \{0\}}$ in λ, n, and a tuple of query-efficient type-safe algorithms* $(\mathsf{A.Gen}, \mathsf{A.Acc}, \mathsf{A.Wit}, \mathsf{A.Ver})$ *as follows:*

- $\mathsf{A.Gen}(\mathfrak{g}, 1^\lambda, 1^n) \xrightarrow{\$} \mathsf{pp}$ *where* $\mathsf{pp} = (\mathfrak{p}, \tau_0) \in \mathbb{G}^{\vartheta_0} \times \{0,1\}^{\zeta_0}$ *returns the accumulator's public parameters;*[10]
- $\mathsf{A.Acc}(\mathsf{pp}, S) \xrightarrow{\$} (\mathsf{cm}, \mathsf{aux})$ *where* $\mathsf{cm} = (\mathfrak{c}, \tau_1) \in \mathbb{G}^{\vartheta_1} \times \{0,1\}^{\zeta_1}$ *is the digest of a set $S \subseteq \{0,1\}^\lambda$;*
- $\mathsf{A.Wit}(\mathsf{pp}, \mathsf{aux}, s, c_1, \ldots, c_{i-1}) \xrightarrow{\$} w_i$*, for $i \in [\rho]$ with challenges $c_j \in \{0,1\}^{\ell_j}$ (for $j \in \{1, \ldots, i-1\}$) outputs the i-th message $w_i = (\mathfrak{a}_i, t_i) \in \mathbb{G}^{\mu_i} \times \{0,1\}^{\eta_i}$ in the interactive proof of membership of s in* cm*;*
- $\mathsf{A.Ver}(\mathsf{pp}, \mathsf{cm}, s, \mathsf{tr}) \to 1/0$ *deterministically checks whether the transcript $\mathsf{tr} = (w_1, c_1, \ldots, w_\rho, c_\rho)$ is a valid proof for the membership of s in the set accumulated by* cm*.*

We use $\chi = \chi(\lambda, n)$ to denote a bound on the query-efficiency of $\langle \mathsf{A.Ver} \rangle$.[11]

Remark 1. Note that, despite the formal definition above requiring $\mathsf{A.Ver}$ to be deterministic, this model still captures randomized verification procedures: observe that the ρ-th challenge c_ρ remains *unanswered* by the prover, and hence we may identify c_ρ with the random tape that a randomized verification procedure would have. Our lower bound does not depend at all on ℓ_ρ, and hence the restriction to let $\mathsf{A.Ver}$ be deterministic is without loss of generality.

Remark 2. Strictly speaking, the (somewhat structured) input/output spaces of the accumulator's algorithms do not match the requirements of Definition 6, which only supports spaces of the form $\mathbb{G}^* \times \{0,1\}^*$. However, since all length functions $\rho, \vartheta_0, \vartheta_1, (\mu_i, \nu_i, \ell_i)_{i \in \mathbb{N}}$ are fixed, we will abuse notation when writing

$$\mathsf{A.Ver}(\mathfrak{p} || \mathfrak{c} || \mathfrak{a}_1 || \ldots || \mathfrak{a}_\rho, (\tau_0, \tau_1, t_1, \ldots, t_\rho, c_1, \ldots, c_\rho))$$

[9] In fact, our lower bound would even hold if only $\mathsf{A.Ver}$ is type-safe, but for the sake of simplicity we put the same restriction on all algorithms.

[10] Note that we need to pass the generator $\mathfrak{g}$ to $\mathsf{A.Gen}$ since a type-safe algorithm cannot create group elements on its own if it doesn't have any as input.

[11] Note that χ is a polynomial in λ, n due to $\mathsf{A.Ver}$'s input being polynomial in λ, n.

to mean $\mathsf{A.Ver}((\mathfrak{p}, \tau_0), (\mathfrak{c}, \tau_1), \mathsf{tr})$, where $\mathsf{tr} = ((\mathfrak{a}_1, t_1), c_1, \ldots, (\mathfrak{a}_\rho, t_\rho), c_\rho)$. Doing so will allow us to apply Exec, Sim, and Rel to the modified $\mathsf{A.Ver}$ interface.

Definition 9 (Correctness). *An accumulator* A *as per Definition 8 is* $\varepsilon = \varepsilon(\lambda, n)$*-correct if for any* $\lambda, n \in \mathbb{N}$*, any set* $S \subseteq \{0,1\}^\lambda$ *of size* $|S| \leq n$ *and any element* $s \in S$*,*

$$\Pr\left[\mathsf{A.Ver}(\mathsf{pp}, \mathsf{cm}, s, \mathsf{tr}) = 0\right] \;\leq\; \varepsilon(\lambda, n)\ ,$$

taken over $\mathsf{pp} \leftarrow^{\$} \mathsf{A.Gen}(\mathfrak{g}, 1^\lambda, 1^n)$*, digest* $(\mathsf{cm}, \mathsf{aux}) \leftarrow^{\$} \mathsf{A.Acc}(\mathsf{pp}, S)$*, challenges* $c_i \leftarrow^{\$} \{0,1\}^{\ell_i}$ *and witnesses* $w_i \leftarrow \mathsf{A.Wit}(\mathsf{pp}, \mathsf{aux}, s, c_1, \ldots, c_{i-1}; r_{\mathsf{wit}})$ *for some common random tape* $r_{\mathsf{wit}} \leftarrow^{\$} \{0,1\}^*$*, and where we define* $\mathsf{tr} = (w_1, c_1, \ldots, w_\rho, c_\rho)$*.*

Definition 10 (Collision Freeness). *An accumulator* A *as per Definition 8 is collision-free if for any query-efficient type-safe adversary* $\mathcal{A}$ *and polynomially bounded function* $n = n(\lambda)$*, there exists a negligible* ε *such that for all* $\lambda \in \mathbb{N}$*:*

$$\mathsf{Adv}^{\mathsf{coll\text{-}free}}_{\mathsf{A},\mathcal{A}}(\lambda, n) \;\coloneqq\; \Pr\left[\mathsf{Exp}^{\mathsf{coll\text{-}free}}_{\mathsf{A},\mathcal{A}}(\lambda, n) = 1\right] \;\leq\; \varepsilon(\lambda)\ ,$$

where the experiment $\mathsf{Exp}^{\mathsf{coll\text{-}free}}_{\mathsf{A},\mathcal{A}}(\lambda, n)$ *is defined in Figure 3.*

$\mathsf{Exp}^{\mathsf{coll\text{-}free}}_{\mathsf{A},\mathcal{A}}(\lambda, n)$:

1 : $\mathsf{pp} \leftarrow^{\$} \mathsf{A.Gen}(\mathfrak{g}, 1^\lambda, 1^n)$
2 : $S \leftarrow^{\$} \mathcal{A}(1^\lambda, 1^n, \mathsf{pp})$
3 : $(\mathsf{cm}, \mathsf{aux}) \leftarrow^{\$} \mathsf{A.Acc}(\mathsf{pp}, S)$
4 : $s \leftarrow^{\$} \mathcal{A}(1^\lambda, 1^n, \mathsf{pp}, \mathsf{cm}, \mathsf{aux})$
5 : For $i \in [\rho]$:
6 : **Wait** for $\mathcal{A}$ to send $w_i \in \mathbb{G}^{\mu_i} \times \{0,1\}^{\eta_i}$
7 : Sample $c_i \leftarrow^{\$} \{0,1\}^{\ell_i}$ and **reply** to $\mathcal{A}$ with c_i
8 : $\mathsf{tr} \leftarrow (w_1, c_1, \ldots, w_\rho, c_\rho)$
9 : $b \leftarrow \mathsf{A.Ver}(\mathsf{pp}, \mathsf{cm}, s, \mathsf{tr})$
10 : **Return** $b \wedge (s \notin S)$

Fig. 3. Collision-freeness experiment (see Definition 10).

Remark 3. In the definition of collision-freeness above, we are again slightly abusing notation by having an *interactive* adversary $\mathcal{A}$, something that is not immediately captured by our definition of type-safe algorithms. However, we can easily transform any such interactive adversary into a non-interactive one by letting it halt each time it sends a message, and then re-executing it with the entire history of messages (as well as the same random coins) once a response is sent to the adversary. "Query-efficient" now means that each execution of $\mathcal{A}$ is query-efficient (in terms of the length of all messages received so far combined).

Remark 4. For any A accumulator as per Definition 8, whenever objects $\mathsf{pp} \in \mathrm{Supp}\big(\mathsf{A.Gen}(\mathfrak{g}, 1^\lambda, 1^n)\big)$, and $(\mathsf{cm}, \mathsf{aux}) \in \mathrm{Supp}(\mathsf{A.Acc}(\mathsf{pp}, \cdot))$ are fixed, we will write $\mathfrak{p}$, $\mathfrak{c}$, τ_1, and τ_2 to implicitly mean the values $(\mathfrak{p}, \tau_0) = \mathsf{pp}$ and $(\mathfrak{c}, \tau_1) = \mathsf{cm}$. We further implicitly define $t_0 = \tau_0 || \tau_1$.

3 Internal Procedures

In this section we define subroutines Forge and $\mathsf{Extract}$ as required by our main result. To this end, we first describe an information-theoretical game in Section 3.1 and then use this to construct $\mathsf{Extract}$ in Section 3.2 and Forge in Section 3.3.

3.1 Information Theoretical Game

We now define an information-theoretical game that is at the heart of our compression argument. Informally, given a fixed element $s \in \{0,1\}^\lambda$ (as per Definition 8), the first player impersonates the prover $\mathsf{A.Wit}$, sending at each round the next witness message, while the second player acts as the honest verifier, replying with uniformly sampled challenges. Intuitively, the first player should win whenever it successfully convinces the second player of the membership $s \in S$.

However, we need the winning probability to be measurable in a *query-efficient* way. Hence, we replace the group elements $\mathfrak{a}_i$ in witness w_i with an "approximation", i.e., a vector space of linear relations L_i satisfied by $\mathfrak{p}||\mathfrak{c}||\mathfrak{a}_1||\ldots||\mathfrak{a}_i$. In order to win the game, (1) *simulated* verification $\mathsf{Sim}_{\mathsf{A.Ver}}$ must succeed, and (2) each L_i must be consistent with the previous L_{i-1} (since, later on, we will need to be able to actually run the "real" accumulator security game where the witnesses w_i must be producible according to L_i if it is consistent to L_{i-1}).

Definition 11 (Terminal Game History). *Let $\lambda, n \in \mathbb{N}$ and A be an accumulator as per Definition 8. We define $\mathcal{Z}^{(\mathsf{A},\lambda,n)}$ as the set of all "terminal game history" tuples $((L_0, t_0), s, (L_1, t_1), c_1, \ldots, (L_\rho, t_\rho), c_\rho)$ which, denoting $\mu_0 = \vartheta_0 + \vartheta_1$, $\eta_0 = \zeta_0 + \zeta_1$ and $\bar{\mu}_i \coloneqq \mu_0 + \ldots + \mu_i$, fulfill*

$$t_i \in \{0,1\}^{\eta_i}, \quad s \in \{0,1\}^\lambda, \quad c_i \in \{0,1\}^{\ell_i}, \quad L_i \leq \mathbb{F}_q^{\bar{\mu}_i}$$

Given some terminal game history $h \in \mathcal{Z}^{(\mathsf{A},\lambda,n)}$, we define its output as

$$\mathsf{out}(h) \coloneqq \mathsf{Sim}_{\langle \mathsf{A.Ver} \rangle}(q, L_\rho, (t_0, s, t_1, c_1, \ldots, t_\rho, c_\rho)).$$

Definition 12 (Game History). *Let $\lambda, n \in \mathbb{N}$ and A be an accumulator as per Definition 8. We define $\mathcal{H}^{(\mathsf{A},\lambda,n)} = \{\pi_i(h) | h \in \mathcal{Z}^{(\mathsf{A},\lambda,n)},\ i \in \mathbb{N}\}$ as the set of game histories associated to A. Moreover, we define*

- $\mathcal{H}_i^{(\mathsf{A},\lambda,n)} = \{\pi_i(h) | h \in \mathcal{Z}^{(\mathsf{A},\lambda,n)}\}$ *the set of game histories of i rounds*
- $\mathcal{H}_{\geq i}^{(\mathsf{A},\lambda,n)} = \cup_{j \geq i} \mathcal{H}_j^{(\mathsf{A},\lambda,n)}$ *the set of games histories of i or more rounds.*

Remark 5. When clear from context we may omit (A, λ, n) and simply denote $\mathcal{Z} = \mathcal{Z}^{(\mathsf{A},\lambda,n)}$ and $\mathcal{H} = \mathcal{H}^{(\mathsf{A},\lambda,n)}$.

Definition 13 (Winning State and Winning Probability). *Given* A *as in Definition 8,* $\lambda, n \in \mathbb{N}$ *we define the winning condition on terminal game histories as* $\omega : \mathcal{Z} \to \{0,1\}$ *such that* $\omega(h) = 1$ *iff*

$$\mathsf{out}(h) = 1 \quad \wedge \quad \bigwedge\nolimits_{i=0}^{\rho-1} (L_i = \pi^*_{\bar{\mu}_i}(L_{i+1}))$$

We also extend ω *to arbitrary game histories as* $\omega : \mathcal{H}_{\geq 2} \to [0,1]$ *so that for* $i \in \{1, \ldots, \rho\}$ *when:*

- $h \in \mathcal{H}_{2i}$*, then* $\omega(h) = \max\{\omega(h||(L_i, t_i)) \,:\, h||(L_i, t_i) \in \mathcal{H}_{2i+1}\}$*;*
- $h \in \mathcal{H}_{2i+1}$*, then* $\omega(h) = \frac{1}{2^{\ell_i}} \cdot \sum_{c_i \in \{0,1\}^{\ell_i}} \omega(h||c_i)$*.*

The following lemma, stating that for any L_0, t_0, s there exists a way for the adversary to win with probability $\omega((L_0, t_0), s)$, follows immediately from how we defined the winning probability ω on game histories:

Lemma 9 (Optimal Strategy). *Given* A *as per Definition 8,* $\lambda, n \in \mathbb{N}$*, for any* $L_0 \leq \mathbb{F}_q^{\mu_0}$*,* $t_0 \in \{0,1\}^n$*,* $s \in \{0,1\}^\lambda$ *there exist functions* $(\bar{L}_i, \bar{t}_i)_{i=1}^{\rho}$*,*

$$\bar{L}_i : \prod\nolimits_{j=1}^{i-1} \{0,1\}^{\ell_j} \to \mathsf{Sub}(\mathbb{F}_q^{\bar{\mu}_i}), \qquad \bar{t}_i : \prod\nolimits_{j=1}^{i-1} \{0,1\}^{\ell_j} \to \{0,1\}^{\eta_i},$$

s.t. sampling $\mathbf{c} \leftarrow^{\$} \prod_{j=1}^{\rho-1} \{0,1\}^{\ell_j}$*, setting* $L_i = \bar{L}_i(c_1, \ldots, c_{i-1})$*,* $t_i = \bar{t}_i(c_1, \ldots, c_{i-1})$*, and* $h = ((L_0, t_0), s, (L_1, t_1), c_1, \ldots, (L_\rho, t_\rho), c_\rho)$ *then*

$$\Pr[\omega(h) = 1] \;=\; \omega(((L_0, t_0), s)).$$

The following lemma states that it is impossible to win with probability larger than $\omega((L_0, t_0), s)$. It follows immediately from basic probability theory:

Lemma 10. *Fix* A *as per Definition 8,* $\lambda, n \in \mathbb{N}$*, and* $L_0 \leq \mathbb{F}_q^{\mu_0}$*,* $t_0 \in \{0,1\}^n$*,* $s \in \{0,1\}^\lambda$*. For every stateful unbounded probabilistic machine* $\mathcal{A}$ *that, for each* $i \in [\rho]$*, outputs* (L_i, t_i) *and receives a uniformly random* $c_i \leftarrow^{\$} \{0,1\}^{\ell_i}$*, we have*

$$\Pr[\omega(h) = 1] \leq \omega(((L_0, t_0), s)) \;,$$

where $h = ((L_0, t_0), s, (L_1, t_1), c_1, \ldots, (L_\rho, t_\rho), c_\rho)$*.*

3.2 Extraction Subroutine

We present the full $\mathsf{Extract}$ subroutine in Figure 4. It is intended to be called as $\mathsf{Extract}^{\mathcal{W},\mathcal{C}}(1, \mathsf{pp}, \mathsf{cm}, s, L_0)$ for some $s \in \{0,1\}^\lambda$ and $L_0 \leq \mathbb{F}_q^{\mu_0}$ such that $\mathfrak{p}||\mathbf{c} \in \mathrm{Ann}_{\mathbb{G}}(L_0)$. As we will show in Lemma 12, as long as the winning probability $\omega((L_0, t_0), s)$ is indeed sufficiently low, then there is a high chance that $\mathsf{Extract}$ returns a strictly larger vector space $L_0' \supsetneq L_0$ of linear constraints satisfied by $\mathfrak{p}||\mathbf{c}$.

$\mathsf{Extract}^{\mathcal{W},\mathcal{C}}(i, \mathsf{pp}, \mathsf{cm}, s, L_{i-1}, (w_1, c_1, j_1), \ldots, (w_{i-1}, c_{i-1}, j_{i-1}))$:

1 : // $i \in \{1, \ldots, \rho\}$, $L_{i-1} \in \mathsf{Sub}(\mathbb{F}_q^{\bar{\mu}_{i-1}})$
2 : **If** $i = \rho + 1$:
3 : Parse $\mathsf{pp} = (\mathfrak{p}, \tau_0)$ and $\mathsf{cm} = (\mathfrak{c}, \tau_1)$, set $t_0 = \tau_0 || \tau_1$
4 : Parse $w_i = (\mathfrak{a}_i, t_i)$ for all $i \in \{1, \ldots, \rho\}$
5 : **Return** $\mathsf{Rel}_{\langle \mathsf{A.Ver} \rangle}(q, \mathfrak{p} || \mathfrak{c} || \mathfrak{a}_1 || \ldots || \mathfrak{a}_\rho, (t_0, s, t_1, c_1, \ldots, t_\rho, c_\rho))$
6 : **Else**:
7 : $w_i \leftarrow \mathcal{W}(s, c_1, \ldots, c_{i-1})$, parse $w_i = (\mathfrak{a}_i, t_i)$
8 : $L_i \leftarrow L_{i-1} \times \{0^{\mu_i}\}$ // $\in \mathsf{Sub}(\mathbb{F}_q^{\bar{\mu}_i})$
9 : **For** each $j_i \in \{0, \ldots, \mu_i\}$:
10 : $c_i \leftarrow \mathcal{C}(i, j_1, \ldots, j_i, L_i, t_i)$
11 : $L_i \leftarrow L_i + \mathsf{Extract}^{\mathcal{W},\mathcal{C}}(i+1, \mathsf{pp}, \mathsf{cm}, s, L_i, (w_1, c_1, j_1), \ldots, (w_i, c_i, j_i))$
12 : **Return** $\pi^*_{\bar{\mu}_{i-1}}(L_i)$

$\mathcal{W}^{[\mathsf{pp},\mathsf{aux},r_{\mathsf{wit}}]}_{\mathsf{real}}(s, c_1, \ldots, c_i)$:

1 : **Return** $\mathsf{A.Wit}(\mathsf{pp}, \mathsf{aux}, s, c_1, \ldots, c_i; r_{\mathsf{wit}})$

$\mathcal{C}^{\mathsf{Extract}}_{\mathsf{rand}}(i, j_1, \ldots, j_i, L_i, t_i)$

1 : **Return** $c_i \leftarrow^{\$} \{0,1\}^{\ell_i}$

Fig. 4. Extractor subroutine, for any fixed accumulator A and $\lambda, n \in \mathbb{N}$. It expects oracles $\mathcal{W}$ for generating the proofs w_i, and $\mathcal{C}$ for sampling challenges c_i. We normally consider $\mathcal{W} = \mathcal{W}^{[\mathsf{pp},\mathsf{aux},r_{\mathsf{wit}}]}_{\mathsf{real}}$, but later (Section 4.1) we will use $\mathcal{W}$ to inject proofs generated in a different way. Similarly, we normally consider $\mathcal{C} = \mathcal{C}^{\mathsf{Extract}}_{\mathsf{rand}}$, but in the proof of Lemma 12 we will replace it with challenges generated from a different source (which is why $\mathcal{C}$ takes as input the values $j_1, \ldots, j_i, L_i, t_i$ ignored by $\mathcal{C}^{\mathsf{Extract}}_{\mathsf{rand}}$).

Lemma 11. *Fix any* A *as per Definition 8,* $\lambda, n \in \mathbb{N}$, $S \subseteq \{0,1\}^\lambda$, *as well as* $\mathsf{pp} \in \mathrm{Supp}\big(\mathsf{A.Gen}(\mathfrak{g}, 1^\lambda, 1^n)\big)$, $(\mathsf{cm}, \mathsf{aux}) \in \mathrm{Supp}(\mathsf{A.Acc}(\mathsf{pp}, S))$, $L_0 \leq \mathbb{F}_q^{\mu_0}$ *such that* $\mathfrak{p} || \mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L_0)$, *and (potentially randomized) oracles* $\mathcal{W}$ *and* $\mathcal{C}$.

Then, for any $L'_0 \in \mathrm{Supp}\Big(\mathsf{Extract}^{\mathcal{W},\mathcal{C}}(1, \mathsf{pp}, \mathsf{cm}, s, L_0)\Big)$, *we have* $L_0 \subseteq L'_0$ *and* $\mathfrak{p} || \mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L'_0)$.

Proof. $L_0 \subseteq L'_0$ is obviously fulfilled by construction.

We now show for every $i \in \{0, \ldots, \rho\}$, all $(w_1, c_1, j_1), \ldots, (w_i, c_i, j_i)$ with $(\mathfrak{a}_j, t_j) := w_j$ for $j \in \{1, \ldots, i\}$, and for every $L_i \leq \mathbb{F}_q^{\bar{\mu}_i}$ with $\mathfrak{u}_i := \mathfrak{p} || \mathfrak{c} || \mathfrak{a}_1 || \ldots || \mathfrak{a}_i \in \mathrm{Ann}_{\mathbb{G}}(L_i)$: we also have $\mathfrak{u}_i \in \mathrm{Ann}_{\mathbb{G}}(L'_i)$, for any potential return value $L'_i \in \mathrm{Supp}\Big(\mathsf{Extract}^{\mathcal{W},\mathcal{C}}(i+1, \mathsf{pp}, \mathsf{cm}, s, L_i, (w_1, c_1, j_1), \ldots, (w_i, c_i, j_i))\Big)$.

We prove this by induction over i, starting with $i = \rho$ and decreasing to $i = 0$. Clearly, using Lemma 4 (i.e., the fact that Rel only returns valid relations), the statement is true for $i = \rho$.

Suppose that the statement holds for $i+1$, and we want to show it for i. Let $w_{i+1} = \mathcal{W}(s, c_1, \ldots, c_i)$ returned in line 7, and define $(\mathfrak{a}_{i+1}, t_{i+1}) := w_{i+1}$ as well as $\mathfrak{u}_{i+1} := \mathfrak{u}_i || \mathfrak{a}_{i+1}$. Due to $\mathfrak{u}_{i+1} \in \mathrm{Ann}_{\mathbb{G}}(L_i \times \{0^{\mu_i}\})$ and by induction

hypothesis, the statement now follows from the fact that for any two vector spaces $R, R' \leq \mathbb{F}_q^{\bar{\mu}_{i+1}}$ with $\mathbf{u}_{i+1} \in \mathrm{Ann}_{\mathbb{G}}(R)$ and $\mathbf{u}_{i+1} \in \mathrm{Ann}_{\mathbb{G}}(R')$, we also have $\mathbf{u}_{i+1} \in \mathrm{Ann}_{\mathbb{G}}(R + R')$. Because of this, the loop in line 9 ensures that at all times, $\mathbf{u}_{i+1} \in \mathrm{Ann}_{\mathbb{G}}(L_{i+1})$, where L_{i+1} denotes the variable initialized in line 8; and hence also $\mathbf{u}_i \in \mathrm{Ann}_{\mathbb{G}}(\pi^*_{\bar{\mu}_i}(L_{i+1})) = \mathrm{Ann}_{\mathbb{G}}(L'_i)$. □

Lemma 12. *Fix any ε-correct* A *as per Definition 8, $\lambda \in \mathbb{N}$, $n = n(\lambda)$ polynomially bounded, $S \subseteq \{0,1\}^\lambda$, and some $p \in [0,1]$.*

Then, denoting $\mathcal{C} := \mathcal{C}^{\mathsf{Extract}}_{\mathsf{rand}}$, we have

$$\Pr\left[\begin{matrix}\omega((L_0,t_0),s) \leq p \wedge \\ \mathsf{Extract}^{\mathcal{W},\mathcal{C}}(1,\mathsf{pp},\mathsf{cm},s,L_0) = L_0\end{matrix}\right] \leq \left(\prod\nolimits_{i=1}^{\rho}(\mu_i+1)\right)\cdot(n\cdot\varepsilon+p)\ , \quad (1)$$

where this probability is taken over $\mathsf{pp} \leftarrow^{\$} \mathsf{A.Gen}(\mathfrak{g}, 1^\lambda, 1^n)$*, the execution of* $(\mathsf{cm},\mathsf{aux}) \leftarrow^{\$} \mathsf{A.Acc}(\mathsf{pp}, S)$*, an arbitrarily distributed random variable $L_0 \leq \mathbb{F}_q^{\mu_0}$ that fulfills $\mathfrak{p}||\mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L_0)$, an arbitrarily distributed random variable $s \in S$, and the choice of the random tape r_{wit} for which we define $\mathcal{W} := \mathcal{W}^{[\mathsf{pp},\mathsf{aux},r_{\mathsf{wit}}]}_{\mathsf{real}}$.*

Proof. We label points in the execution of $\mathsf{Extract}^{\mathcal{W},\mathcal{C}}(1,\mathsf{pp},\mathsf{cm},s,L_0)$ using tuples $(j_1,\ldots,j_k) \in \mathbb{Z}_{\mu_1+1}\times\ldots\times\mathbb{Z}_{\mu_{k-1}+1}\times\mathbb{Z}_{\mu_k+2}$, where $1 \leq k \leq \rho$ is the current depth of the recursion. Such a tuple $\mathbf{j} = (j_1,\ldots,j_k)$ identifies the following state:

- For $i < k$, $\mathsf{Extract}^{\mathcal{W},\mathcal{C}}(i,\mathsf{pp},\mathsf{cm},s,L_{i-1},\ldots)$ is currently inside iteration j_i of its for-loop in line 11 *in the process of calling* $\mathsf{Extract}^{\mathcal{W},\mathcal{C}}(i{+}1,\mathsf{pp},\mathsf{cm},s,L_i,\ldots)$.
- For $i = k$, $\mathsf{Extract}^{\mathcal{W},\mathcal{C}}(k,\mathsf{pp},\mathsf{cm},s,L_{k-1},\ldots)$ is currently inside iteration j_k of its for-loop in line 11 *about to call* $\mathsf{Extract}^{\mathcal{W},\mathcal{C}}(i+1,\mathsf{pp},\mathsf{cm},s,L_i,\ldots)$, or if $j_k = \mu_k + 1$ then it is in line 12 about to return to its caller.

Note that the call stack corresponding to a tuple $\mathbf{j} = (j_1,\ldots,j_k)$ defines vector spaces $L_0,\ldots,L_k$ (while L_k does not currently belong to the input of any $\mathsf{Extract}$, it is still is defined as a variable in the scope of $\mathsf{Extract}(k,\ldots)$), which we denote by $L_1^{(\mathbf{j})},\ldots,L_k^{(\mathbf{j})}$. Furthermore, whenever $j_k \leq \mu_k$, the call stack also defines witnesses $w_1,\ldots,w_k$ and challenges $c_1,\ldots,c_k$, which we denote by $w_1^{(\mathbf{j})},\ldots,w_k^{(\mathbf{j})}$ and $c_1^{(\mathbf{j})},\ldots,c_k^{(\mathbf{j})}$ with $w_i^{(\mathbf{j})} = (\mathfrak{a}_i^{(\mathbf{j})}, t_i^{(\mathbf{j})})$.

If $k = \rho$, then we additionally define write $\mathbf{u}^{(\mathbf{j})} = \mathfrak{p}||\mathfrak{c}||\mathfrak{a}_1^{(\mathbf{j})}||\ldots||\mathfrak{a}_\rho^{(\mathbf{j})}$ to mean the concatenation of all group elements in the transcript. Note that by Lemma 4 we always have $\mathbf{u}^{(\mathbf{j})} \in \mathrm{Ann}_{\mathbb{G}}(L_\rho^{(\mathbf{j})})$.

We call a tuple $\mathbf{j}$ that is in $\mathbb{Z}_{\mu_1+1}\times\ldots\times\mathbb{Z}_{\mu_\rho+1}$ *terminating*, as it corresponds to an execution path that defines an entire transcript consisting of $c_i^{(\mathbf{j})}$ and $w_i^{(\mathbf{j})}$ defined for all $i \in \{1,\ldots,\rho\}$. For a terminating tuple $\mathbf{j}$, we define the following two bad events:

- $\mathsf{bad}_1(\mathbf{j})$: $\mathsf{Exec}_{\langle\mathsf{A.Ver}\rangle}(q,\mathbf{u}^{(\mathbf{j})},(t_0,s,t_1,c_1,\ldots,t_\rho,c_\rho)) = 0$.
- $\mathsf{bad}_2(\mathbf{j})$: $\omega(((L_0,t_0),s)) \leq p$ and $\mathsf{Sim}_{\langle\mathsf{A.Ver}\rangle}(q,L_\rho^{(\mathbf{j})},(t_0,s,t_1,c_1,\ldots,t_\rho,c_\rho)) = 1$ and $L_{i-1}^{(\mathbf{j})} = \pi^*_{\bar{\mu}_{i-1}}(L_i^{(\mathbf{j})})$ for $1 \leq i \leq \rho$.

We will start by showing, for all terminating tuples $\mathbf{j}$: $\Pr[\mathsf{bad}_1(\mathbf{j})] \leq n \cdot \varepsilon(\lambda, n)$ and $\Pr[\mathsf{bad}_2(\mathbf{j})] \leq p$.

First, by ε-correctness of A, for any fixed set element $s_0 \in S$, we have $\Pr[\mathsf{bad}_1(\mathbf{j}) \wedge s = s_0] \leq \varepsilon(\lambda, n)$, because $\mathsf{bad}_1(\mathbf{j})$ implies that the real execution of $\mathsf{A.Ver}$ (for fixed S and $s = s_0$, using uniformly random challenges $c_i^{(\mathbf{j})}$ for $c \in [\rho]$) resulted in the verifier returning 0. The additional clause "$s = s_0$" does not increase this probability. Therefore, we obtain

$$\Pr[\mathsf{bad}_1(\mathbf{j})] = \sum_{s_0 \in S} \Pr[\mathsf{bad}_1(\mathbf{j}) \wedge s = s_0] \leq \sum_{s_0 \in S} \varepsilon(\lambda, n) = n \cdot \varepsilon(\lambda, n) \ .$$

Second, using the following oracle $\mathcal{C}^{(\mathbf{j})}$, the algorithm $\mathsf{Extract}^{\mathcal{W}_{\mathsf{real}}^{[\mathsf{pp},\mathsf{aux},r_{\mathsf{wit}}]},\mathcal{C}^{(\mathbf{j})}}(1, \mathsf{pp}, \mathsf{cm}, s, L_0)$, is a player against the information-theoretical game from Section 3.1 (as per Lemma 10).

$\mathcal{C}^{(\mathbf{j})}(i, j'_1, \ldots, j'_i, L_i, t_i)$:

1 : **If** $(j'_1, \ldots, j'_i) = (j_1, \ldots, j_i)$:
2 : **Send** (L_i, t_i) to the challenger and **receive** c_i
3 : **Return** c_i
4 : **Else**:
5 : **Return** $c_i \leftarrow^{\$} \{0,1\}^{\ell_i}$

Let us briefly assume that all the random variables (such as $\mathsf{pp}, \mathsf{cm}, \mathsf{aux}, L_0, s, r_{\mathsf{wit}}$, and the random strings generated within $\mathcal{C}^{(\mathbf{j})}$) are fixed. Then, there are two cases: Either $\omega(((L_0, t_0), s)) > p$ or $\omega(((L_0, t_0), s)) \leq p$. In the former case, we always have $\mathsf{bad}_2(\mathbf{j})$ false by definition, and therefore $\Pr[\mathsf{bad}_2(\mathbf{j})] = 0 \leq p$. In the latter case, we have

$$\mathsf{bad}_2(\mathbf{j}) \iff \omega((L_0, t_0), s, (L_1^{(\mathbf{j})}, t_1^{(\mathbf{j})}), c_1^{(\mathbf{j})}, \ldots, (L_\rho^{(\mathbf{j})}, t_\rho^{(\mathbf{j})}), c_\rho^{(\mathbf{j})}) = 1 \ .$$

Hence, the probability $\Pr[\mathsf{bad}_2(\mathbf{j})]$ (taken over the information-theoretical game challenger) is exactly the probability that $\mathsf{Extract}^{\mathcal{W},\mathcal{C}^{(\mathbf{j})}}(1, \mathsf{pp}, \mathsf{cm}, s, L_0)$ wins the information-theoretical game from Lemma 10 on starting point $((L_0, t_0), s)$, which means that $\Pr[\mathsf{bad}_2(\mathbf{j})] \leq \omega(((L_0, t_0), s)) \leq p$. Combining both cases, even when considering the full random process described in the lemma statement, we have $\Pr[\mathsf{bad}_2(\mathbf{j})] \leq p$ for any fixed terminating tuple $\mathbf{j}$.

For any tuple $\mathbf{j} = (j_1, \ldots, j_k)$, we define $d_i^{(\mathbf{j})} := \dim(L_i^{(\mathbf{j})}) - \dim(L_{i-1}^{(\mathbf{j})})$ for $i \in \{1, \ldots, k\}$ as the number of additional relations that have been learned at level i compared to level $i-1$.

We now prove the following invariant by induction over all tuples $\mathbf{j} = (j_1, \ldots, j_k)$ in lexicographic order, under the assumption that none of the bad events occur for any terminating tuple:

(1) either $d_i^{(\mathbf{j})} \geq j_i$ for every $1 \leq i \leq k$; or
(2) there exists an index $i^* \in \{1, \ldots, k\}$ such that $d_i^{(\mathbf{j})} \geq j_i$ for every $1 \leq i < i^*$ and $L_{i^*-1}^{(\mathbf{j})} \subsetneq \pi^*_{\bar{\mu}_{i^*-1}}(L_{i^*}^{(\mathbf{j})})$.

We distinguish the following cases:

$\mathbf{j} = (0)$ **(base case).** Invariant (1) clearly holds for $\mathbf{j} = (0)$ due to $j_1 = 0$.

$\mathbf{j} = (j_1, \ldots, j_k)$ **for** $1 \leq k < \rho$**.** The new state will be $\mathbf{j}' = (j_1, \ldots, j_k, 0)$. Clearly any invariant that holds for $(j_1, \ldots, j_k)$ transfers over to $(j_1, \ldots, j_k, 0)$.

$\mathbf{j} = (j_1, \ldots, j_{k-1}, \mu_k + 1)$ **for** $2 \leq k \leq \rho$**.** Then, $\mathbf{j}' := (j_1, \ldots, j_{k-1}+1)$ will be the new state. First note that invariant (1) on $\mathbf{j}$ would also imply invariant (2) on $\mathbf{j}$: due to $d_k^{(\mathbf{j})} \geq j_k = \mu_k + 1$ we have $L_{k-1}^{(\mathbf{j})} \subsetneq \pi^*_{\bar{\mu}_{k-1}}(L_k^{(\mathbf{j})})$ (otherwise, since there exist at least $\mu_k + 1$ linearly independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_{\mu_k+1} \notin L_{k-1}^{(\mathbf{j})}$ with $L_{k-1}^{(\mathbf{j})} + \mathsf{span}(\mathbf{v}_1, \ldots, \mathbf{v}_{\mu_k+1}) \subseteq L_k^{(\mathbf{j})}$, we would have $\mathsf{span}(\mathbf{v}_1, \ldots, \mathbf{v}_{\mu_k+1}) \cap \mathbb{F}_q^{\bar{\mu}_{k-1}} = \{0\}$, contradicting the fact that all vectors have dimension $\bar{\mu}_k = \bar{\mu}_{k-1} + \mu_k$).
Hence, we may assume that invariant (2) holds on $\mathbf{j}$, and we show that the same invariant also holds in $\mathbf{j}'$. If invariant (2) holds on $\mathbf{j}$ for some index $i^* < k$, then this immediately also applies to $\mathbf{j}'$. Suppose invariant (2) holds in $\mathbf{j}$ for $i^* = k$. Then, we will have

$$L_{k-1}^{(\mathbf{j}')} = \pi^*_{\bar{\mu}_{k-1}}(L_k^{(\mathbf{j})}) + L_{k-1}^{(\mathbf{j})} \not\subseteq L_{k-1}^{(\mathbf{j})} ,$$

i.e., we get the strict inequality $d_{k-1}^{(\mathbf{j}')} > d_{k-1}^{(\mathbf{j})}$, which together with $d_i^{(\mathbf{j}')} = d_i^{(\mathbf{j})} \geq j_i$ (for all $i < k-1$) implies that invariant (1) holds in the new state.

$\mathbf{j} = (j_1, \ldots, j_\rho)$ **for** $j_\rho \leq \mu_\rho$**,** i.e., $\mathbf{j}$ is terminal. Then, $\mathbf{j}' := (j_1, \ldots, j_{\rho-1}, j_\rho + 1)$ will be the new state. By definition of $\overline{\mathsf{bad}_1(\mathbf{j})}$ and $\overline{\mathsf{bad}_2(\mathbf{j})}$, there are two cases: either we have $L_{i-1}^{(\mathbf{j})} \neq \pi^*_{\bar{\mu}_{i-1}}(L_i^{(\mathbf{j})})$ for some $1 \leq i \leq \rho$, i.e., invariant (2) holds for $\mathbf{j}'$; or both of the following two conditions hold:

$$\mathsf{Exec}_{\langle \mathsf{A.Ver} \rangle}(q, \mathbf{u}^{(j)}, (t_0, s, t_1, c_1, \ldots, t_\rho, c_\rho)) = 1$$
$$\mathsf{Sim}_{\langle \mathsf{A.Ver} \rangle}(q, L_\rho^{(\mathbf{j})}, (t_0, s, t_1, c_1, \ldots, t_\rho, c_\rho)) = 0$$

In this case, because the real and the simulated outputs differ, Lemma 5 combined with the fact that $\mathbf{u}^{(\mathbf{j})} \in \mathrm{Ann}_{\mathbb{G}}(L_\rho^{(\mathbf{j})})$ implies $R \not\subseteq L_\rho^{(\mathbf{j})}$, where

$$R := \mathsf{Rel}_{\langle \mathsf{A.Ver} \rangle}(q, \mathbf{u}^{(j)}, (t_0, s, t_1, c_1, \ldots, t_\rho, c_\rho))$$

is the set of relations learned during the real execution. Then, because R is exactly what is being computed by $\mathsf{Extract}(\rho + 1, \ldots)$, we have $L_\rho^{(\mathbf{j}')} = L_\rho^{(\mathbf{j})} + R \supsetneq L_\rho^{(\mathbf{j})}$, i.e., $d_\rho^{(\mathbf{j}')} > d_\rho^{(\mathbf{j})}$. Hence, if invariant (1) holds for $\mathbf{j}$, it also holds for $\mathbf{j}'$ (and invariant (2) on $\mathbf{j}$ would immediately transfer over to $\mathbf{j}'$ regardless).

To summarize, we know that (unless a bad event occurs), the invariant holds for the largest tuple $\mathbf{j} = (\mu_1 + 1)$. As before, invariant (1) would imply invariant (2) in this case due to $d_1^{(\mathbf{j})} \geq \mu_1 + 1$ implying $L_0 \subsetneq \pi^*_{\bar{\mu}_0}(L_1^{(\mathbf{j})})$. With invariant (2)

on the final state $\mathbf{j} = (\mu_1 + 1)$, the event in (1) indeed holds because the final output of $\mathsf{Extract}$ $(\pi^*_{\vec{\mu}_0}(L_1^{(\mathbf{j})}))$ is not a subset of L_0.

Therefore, taking a union bound of $\overline{\mathsf{bad}_1(\mathbf{j})}$ and $\overline{\mathsf{bad}_2(\mathbf{j})}$ over all possible terminating tuples (of which there are exactly $\prod_{i=1}^{\rho}(\mu_i + 1)$ many) concludes the proof. □

3.3 Forge Subroutine

We present the full Forge subroutine in Figure 5. It is intended to be called as $\mathsf{Forge}^{\mathcal{C}}(L_0, \mathsf{pp}, \mathsf{cm}, s, \kappa)$ for some $s \in \{0,1\}^\lambda$ and $L_0 \leq \mathbb{F}_q^{\mu_0}$ such that $\mathfrak{p}||\mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L_0)$. It runs κ attempts of forging a proof of membership of s. In Lemma 13 we show that as long as the winning probability $\omega((L_0, t_0), s)$ is sufficiently high, then forge will indeed succeed with high probability (or, alternatively, it will find a new relation on L_0). This is due to the collision-freeness of A.

Lemma 13. *Let* A *be as per Definition 8,* $\lambda \in \mathbb{N}$, $n \in \mathbb{N}$, $S \subseteq \{0,1\}^\lambda$, $s \in \{0,1\}^\lambda$ *(not necessarily* $s \in S$*),* $\kappa \in \mathbb{N}$, $\mathsf{pp} \in \mathrm{Supp}\big(\mathsf{A.Gen}(\mathfrak{g}, 1^\lambda, 1^n)\big)$, $(\mathsf{cm}, \mathsf{aux}) \in \mathrm{Supp}(\mathsf{A.Acc}(\mathsf{pp}, S))$*, and* $L_0 \leq \mathbb{F}_q^{\mu_0}$ *such that* $\mathfrak{p}||\mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L_0)$.

Then, denoting $\mathcal{C} := \mathcal{C}^{\mathsf{Forge}}_{\mathsf{rand}}$*, and letting* χ *denote a bound on the query-efficiency of* $\langle \mathsf{A.Ver} \rangle$*, we have*

$$\Pr\Big[\mathsf{Forge}^{\mathcal{C}}(L_0, \mathsf{pp}, \mathsf{cm}, s, \kappa) = \bot\Big] \;\leq\; \left(1 - \omega((L_0, t_0), s) + \frac{\chi}{q}\right)^{\kappa}.$$

Furthermore, $\mathrm{Supp}\Big(\mathsf{Forge}^{\mathcal{C}}(L_0, \mathsf{pp}, \mathsf{cm}, s, \kappa)\Big) \subseteq \{\bot, \mathtt{accept}\} \cup \mathsf{Sub}(\mathbb{F}_q^{\mu_0})$*, and any potential vector space output* $L_0' \in \mathrm{Supp}\Big(\mathsf{Forge}^{\mathcal{C}}(L_0, \mathsf{pp}, \mathsf{cm}, s, \kappa)\Big)$ *fulfills* $L_0 \subsetneq L_0'$ *and* $\mathfrak{p}||\mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L_0')$.

Proof. First note that no randomness is shared between iterations of the outer for-loop. The only sources of randomness are line 9 (which is clearly independent across iterations) and the challenges selected by $\mathcal{C}$ in line 11 (again, by definition of $\mathcal{C}$ and since k is input into each call of $\mathcal{C}$, this is independent across iterations).

For every iteration $k \in [\kappa]$, we consider the game history

$$h^{(k)} = ((L_0, t_0), s, (L_1^{(k)}, t_1^{(k)}), c_1^{(k)}, \ldots, (L_\rho^{(k)}, t_\rho^{(k)}), c_\rho^{(k)})$$

and we define two bad events:

- $\mathsf{bad}_1(k)$: $\omega(h^{(k)}) = 0$;
- $\mathsf{bad}_2(k)$: $\omega(h^{(k)}) = 1$ and $b^{(k)} = 0$ and $\pi^*_{\mu_0}(L_\rho^{(k)} + R^{(k)}) \subseteq L_0$.

Note that if $\overline{\mathsf{bad}_1(k)} \wedge \overline{\mathsf{bad}_2(k)}$ occurs for some k, then Forge will *not* output $\bot$: by $\overline{\mathsf{bad}_1(k)}$ we have $\omega(h^{(k)}) = 1$, and hence by $\overline{\mathsf{bad}_2(k)}$ we have either $b^{(k)} = 1$ (in which case the subroutine returns $\mathtt{accept}$) or $\pi^*_{\mu_0}(L_\rho^{(k)} + R^{(k)}) \not\subseteq L_0$ (in which case the subroutine returns a vector space).

$\mathsf{Forge}^{\mathcal{C}}(L_0, \mathsf{pp}, \mathsf{cm}, s, \kappa)$: // $L_0 \leq \mathbb{F}_q^{\mu_0}$, $s \in \{0,1\}^\lambda$, $\kappa \in \mathbb{N}$

1 : Parse $\mathsf{pp} = (q, \mathfrak{p}, \tau_0)$ and $\mathsf{cm} = (\mathfrak{c}, \tau_1)$
2 : Set $\mathbf{u}_0 = \mathfrak{p}||\mathfrak{c} \in \mathbb{G}^{\mu_0}$ and $t_0 = \tau_0||\tau_1$
3 : Compute $\bar{t}_i, \bar{L}_i$ optimal strategies for $((L_0, t_0), s)$ // See Lemma 9
4 : **For** $k \in \{1, \ldots, \kappa\}$:
5 : $\quad \mathbf{u}_0^{(k)} \leftarrow \mathbf{u}_0$
6 : $\quad$ **For** $i \in \{1, \ldots, \rho\}$:
7 : $\quad\quad t_i^{(k)} \leftarrow \bar{t}_i(c_1, \ldots, c_{i-1})$
8 : $\quad\quad L_i^{(k)} \leftarrow \bar{L}_i(c_1, \ldots, c_{i-1})$
9 : $\quad\quad \mathfrak{a}_i^{(k)} \leftarrow^{\$} \mathbb{G}^{\mu_i}$ such that $\mathbf{u}_{i-1}^{(k)}||\mathfrak{a}_i^{(k)} \in \mathrm{Ann}_{\mathbb{G}}(L_i^{(k)})$ // use Lemma 8
10 : $\quad\quad \mathbf{u}_i^{(k)} \leftarrow \mathbf{u}_{i-1}^{(k)}||\mathfrak{a}_i^{(k)}$
11 : $\quad\quad$ Set $w_i^{(k)} = (\mathfrak{a}_i^{(k)}, t_i^{(k)})$ and query $c_i^{(k)} \leftarrow \mathcal{C}(s, k, i, w_i^{(k)})$
12 : $\quad b^{(k)} \leftarrow \mathsf{A.Ver}(\mathsf{pp}, \mathsf{cm}, s, (w_1^{(k)}, c_1^{(k)}, \ldots, w_\rho^{(k)}, c_\rho^{(k)}))$
13 : $\quad R^{(k)} \leftarrow \mathsf{Rel}_{\langle \mathsf{A.Ver} \rangle}(q, \mathbf{u}^{(k)}, (t_0, s, t_1^{(k)}, c_1^{(k)}, \ldots, t_\rho^{(k)}, c_\rho^{(k)}))$
14 : // After all iterations, check for any successful proof or new relation
15 : **For** $k \in \{1, \ldots, \kappa\}$:
16 : $\quad$ **If** $b^{(k)} = 1$: **Return** `accept` // w.h.p. s is in the accumulated set
17 : $\quad$ **If** $\pi^*_{\mu_0}(L_\rho^{(k)} + R^{(k)}) \not\subseteq L_0$: **Return** $L_0 + \pi^*_{\mu_0}(L_\rho^{(k)} + R^{(k)})$
18 : **Return** $\perp$

$\mathcal{C}^{\mathsf{Forge}}_{\mathsf{rand}}(s, k, i, w_i^{(k)})$

1 : **Return** $c_i \leftarrow^{\$} \{0,1\}^{\ell_i}$

Fig. 5. The Forge subroutine, for any fixed accumulator A and $\lambda, n \in \mathbb{N}$. L_0 is a set of initially known relations among the group elements of pp, cm, and $\kappa \in \mathbb{N}$ denotes the number of forging attempts before giving up. Note that Forge takes an oracle $\mathcal{C}$ for generating the challenges c_i. We are normally interested in the case $\mathcal{C} = \mathcal{C}^{\mathsf{Forge}}_{\mathsf{rand}}$ that always returns random challenges, but in our final reduction to collision-freeness (see proof of Theorem 1) we will need to inject challenges from a different source (this is the reason why $\mathcal{C}$ takes as input the values s, k, w_i ignored by $\mathcal{C}^{\mathsf{Forge}}_{\mathsf{rand}}$).

Therefore, by union bound, it only remains to prove for each k: $\Pr[\mathsf{bad}_1(k)] \leq 1 - \omega((L_0, t_0), s)$ and $\Pr[\mathsf{bad}_2(k)] \leq \frac{\chi}{q}$. The first of these is immediate by Lemma 9 (in fact, even the *equality* $\Pr[\mathsf{bad}_1(k)] = 1 - \omega((L_0, t_0), s)$ holds) due to $\mathcal{C}$ returning uniformly random challenges and Forge is following the optimal strategy $(\bar{L}_i, \bar{t}_i)_{i=1}^{\rho}$.

Now, we bound $\Pr[\mathsf{bad}_2(k)]$. Note that we could restructure the k-th iteration of Forge so that it first plays the game from Section 3.1 to compute the game history $h^{(k)}$, and only afterwards samples $\mathfrak{a}_i^{(k)}$ for all $i \in \{1, \ldots, \rho\}$ (which was previously done in line 9) and computes $\mathbf{u}^{(k)} = \mathbf{u}_0||\mathfrak{a}_1^{(k)}|| \ldots \mathfrak{a}_\rho^{(k)}$. By Lemma 7,

this method of sampling $\mathbf{u}^{(k)}$ is identical to immediately sampling the entire $\mathbf{a}_1^{(k)}||\ldots||\mathbf{a}_\rho^{(k)} \in \mathbb{G}^{\mu_1+\ldots\mu_\rho}$ uniformly random such that $\mathbf{u}_0||\mathbf{a}_1^{(k)}||\ldots||\mathbf{a}_\rho^{(k)} \in \mathrm{Ann}_{\mathbb{G}}(L_\rho^{(k)})$. With this change we can apply Lemma 6 to conclude that

$$\Pr\left[\begin{array}{l} b^{(k)} \neq \mathsf{Sim}_{\langle \mathsf{A.Ver}\rangle}(q, L_\rho^{(k)}, (t_0, s, t_1^{(k)}, c_1^{(k)}, \ldots, t_\rho^{(k)}, c_\rho^{(k)})) \wedge \\ \pi^*_{\mu_0}(L_\rho^{(k)} + R^{(k)}) \subseteq \pi^*_{\mu_0}(L_\rho^{(k)}) \end{array}\right] \leq \frac{\chi}{q},$$

so we just need to prove that this event is implied by $\mathsf{bad}_2(k)$. Indeed, by definition of $\mathsf{bad}_2(k)$ we have $b^{(k)} = 0$ and $\omega(h^{(k)}) = 1$, where the latter implies $\mathsf{Sim}_{\langle \mathsf{A.Ver}\rangle}(q, L_\rho^{(k)}, (t_0, s, t_1^{(k)}, c_1^{(k)}, \ldots, t_\rho^{(k)}, c_\rho^{(k)})) = 1$, i.e., the first condition of the event above is satisfied. Furthermore, the second condition follows due to $L_0 = \pi^*_{\mu_0}(L_\rho^{(k)})$, which again is a consequence of $\omega(h^{(k)}) = 1$.

In order to prove the final part of the lemma, note that whenever Forge outputs neither $\perp$ nor $\mathtt{accept}$, it must have output some $L_0' := L_0 + \pi^*_{\mu_0}(L_\rho^{(k)} + R^{(k)})$ with $\pi^*_{\mu_0}(L_\rho^{(k)} + R^{(k)}) \not\subseteq L_0$, hence $L_0 \subsetneq L_0'$. In addition, $\mathbf{u}_\rho^{(k)} \in \mathrm{Ann}_{\mathbb{G}}(L_\rho^{(k)})$ by construction and $\mathbf{u}_\rho^{(k)} \in \mathrm{Ann}_{\mathbb{G}}(R^{(k)})$ by Lemma 4.

Therefore, $\mathbf{u}_0 \in \mathrm{Ann}_{\mathbb{G}}(\pi^*_{\mu_0}(L_\rho^{(k)} + R^{(k)}))$, which implies $\mathbf{u}_0 \in L_0'$. □

4 Witness Length Bound

Before proving our main result, we first fix some notation and parameters that we will throughout the section. Let A be an accumulator as per Definition 8, and let $\lambda, n \in \mathbb{N}$ be fixed s.t. $\lambda \geq 1 + \log_2 n$.

- Given a message vector $\mathbf{m} \in \{0,1\}^n$, we define $S_{\mathbf{m}} := \{(i, m_i) : i \in [n]\} \subseteq [n] \times \{0,1\}$. Furthermore, due to $\lambda \geq 1 + \log_2 n$, we can treat $S_{\mathbf{m}}$ as a size-n subset of $\{0,1\}^\lambda$ (i.e., $S_{\mathbf{m}}$ may be used as an input to $\mathsf{A.Acc}$).
- Let γ_{pp}, γ_{cm}, and γ_{wit} be upper bounds on the number of random coins used by $\mathsf{A.Gen}$, $\mathsf{A.Acc}$, and $\mathsf{A.Wit}$ respectively.
- For random tape $r_{\mathsf{pp}} \in \{0,1\}^{\gamma_{\mathsf{pp}}}$, we implicitly define $\mathsf{pp} \leftarrow \mathsf{A.Gen}(\mathfrak{g}, 1^\lambda, 1^n; r_{\mathsf{pp}})$, and given random tape $r_{\mathsf{cm}} \in \{0,1\}^{\gamma_{\mathsf{cm}}}$ as well as message $\mathbf{m}$, we implicitly define $(\mathsf{cm}, \mathsf{aux}) \leftarrow \mathsf{A.Acc}(\mathsf{pp}, S_{\mathbf{m}}; r_{\mathsf{cm}})$. Then, $\mathfrak{p}$, $\mathfrak{c}$, and t_0 are also defined by Remark 4.
- We denote the bitlength of a commitment cm and a proof (consisting of witnesses $w_1, \ldots, w_\rho$) as

$$\xi_{\mathsf{cm}} := \eta_0 + \log q\vartheta_1, \qquad \xi_{\mathsf{wit}} := \sum_{i=1}^{\rho} \eta_i + \log q \sum_{i=1}^{\rho} \mu_i,$$

  respectively.
- We define p and κ (as needed by the extract and forge subroutines, respectively) as follows:

$$p := \frac{1}{12n\prod_{i=1}^{\rho}(\mu_i + 1)} - n\varepsilon, \qquad \kappa := \left\lceil \frac{2\ln(12n)}{p} \right\rceil.$$

4.1 Basic Decoding Strategy

In Figure 6, we define the "basic" encoder-decoder pair $(\mathcal{E}_0, \mathcal{D}_0)$. Note that, by themselves, they are not used for a compression argument yet. Instead, we will use them as subroutines inside $(\mathcal{E}, \mathcal{D})$ (defined in Section 4.2), where $\mathcal{D}_0$ will first be repeated a large number of times (using a self-simulated encoder $\mathcal{E}_0$) in order to find relations about the group elements $\mathfrak{p}$ from pp, before then running $\mathcal{D}_0$ one final time to decode the message $\mathbf{m}$ without finding new relations about $\mathfrak{p}$.

$\mathcal{D}_0^{\mathcal{C},\mathcal{E}^*}(L_0, \mathsf{pp})$: // Basic decoder taking hint $L_0 \leq \mathbb{F}_q^{\mu_0}$

Parse $\mathsf{pp} = (\mathfrak{p}, \tau_0)$ and set $L_0^{(1)} = L_0$
Send `init` and wait for $\mathsf{cm} = (\mathfrak{c}, \tau_1)$; Set $t_0 = \tau_0 || \tau_1$
For $i \in [n]$:
  $B^{(i)} \leftarrow \{b \in \{0,1\} \,:\, \omega((L_0^{(i)}, t_0), (i,b)) \geq p\}$
  If $B^{(i)} \neq \varnothing$: // Given high winning probability, guess m_i
    Choose $b^{(i)} \leftarrow \min(B^{(i)})$
    $\mathsf{res}^{(i)} \leftarrow \mathsf{Forge}^{\mathcal{C}}(L_0^{(i-1)}, \mathsf{pp}, \mathsf{cm}, (i, b^{(i)}), \kappa)$
    If $\mathsf{res}^{(i)} = \perp$: **Return** $\perp$
    Elif $\mathsf{res}^{(i)} = \texttt{accept}$: Store $\bar{m}_i \leftarrow b^{(i)}$; Set $L_0^{(i+1)} \leftarrow L_0^{(i)}$
    Else: // Forge did not validate $b^{(i)}$ but found a new relation
      $L_0^{(i+1)} \leftarrow \mathsf{res}^{(i)}$
      $\bar{m}_i \leftarrow \mathcal{E}^*(\texttt{bit}, i)$
  Else: // Given low winning probability, extract new relations
    $\bar{m}_i \leftarrow \mathcal{E}^*(\texttt{bit}, i)$; Set $s \leftarrow (i, \bar{m}_i)$
    $L_0^{(i+1)} \leftarrow \mathsf{Extract}^{\mathcal{W},\mathcal{C}_{\mathsf{rand}}^{\mathsf{Extract}}}(1, \mathsf{pp}, \mathsf{cm}, s, L_0^{(i)})$ // oracle $\mathcal{W}$ defined as follows:
    When it queries $\mathcal{W}(s, c_1, \ldots, c_k)$: **Reply** $w \leftarrow \mathcal{E}^*(\texttt{wit}, s, c_1, \ldots, c_k)$
    If $L_0^{(i+1)} \subseteq L_0^{(i)}$: **Return** $\perp$
Return $((\bar{m}_1, \ldots, \bar{m}_n), L_0^{(n+1)})$

$\mathcal{E}_0(\mathbf{m}, \mathsf{pp}, \mathsf{cm}, \mathsf{aux}, r_{\mathsf{wit}}, x)$: // Basic encoder

If $x = \texttt{init}$: **Return** cm
If $x = (\texttt{bit}, i)$: **Return** m_i
If $x = (\texttt{wit}, s, c_1, \ldots, c_k)$: **Return** $\mathsf{A.Wit}(\mathsf{pp}, \mathsf{aux}, s, (c_1, \ldots, c_k); r_{\mathsf{wit}})$

Fig. 6. Basic decoder, for any fixed accumulator A and $\lambda, n \in \mathbb{N}$ with $\lambda \geq 1 + \log_2 n$. Parameters p and κ are defined at the beginning of Section 4. The decoder makes queries to the it's encoder oracle $\mathcal{E}^*$, which by default (i.e., using $\mathcal{E}_0$) just replies using some predefined information $\mathbf{m}$, pp, cm, aux, and r_{wit}.

First, the following lemma captures some basic facts about $(\mathcal{D}_0, \mathcal{E}_0)$, which we prove in Appendix A.3.

Lemma 14. *Given* A *as per Definition 8,* $\lambda, n \in \mathbb{N}$ *with* $\lambda \geq 1 + \log_2 n$, $\mathbf{m} \in \{0,1\}^n$, $\mathsf{pp} \in \mathrm{Supp}\big(\mathsf{A.Gen}(\mathfrak{g}, 1^\lambda, 1^n)\big)$, $(\mathsf{cm}, \mathsf{aux}) \in \mathrm{Supp}(\mathsf{A.Acc}(\mathsf{pp}, S_{\mathbf{m}}))$, $r_{\mathsf{wit}} \in \{0,1\}^{\gamma_{\mathsf{wit}}}$, *and an oracle* $\mathcal{C}$, *the following holds:*

1. *When running* $\mathcal{D}_0^{\mathcal{C},\mathcal{E}_0(\mathbf{m},\mathsf{pp},\mathsf{cm},\mathsf{aux},r_{\mathsf{wit}},\cdot)}(L_0,\mathsf{pp})$ *for some vector space* $L_0 \leq \mathbb{F}_q^{\mu_0}$ *with* $\mathfrak{p}||\mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L_0)$, *then* $L_0^{(i)} \subseteq L_0^{(i+1)}$ *and* $\mathfrak{p}||\mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L_0^{(i+1)})$ *for all* $i \in \{1,\ldots,n'\}$, *where* n' *is the number of executed iterations of the For loop in Line 3 that did not abort by returning* $\perp$.
2. *If* $\mathcal{C}$ *is deterministic,* r_0 *is some fixed randomness, and the output of the basic decoder* $\mathcal{D}_0^{\mathcal{C},\mathcal{E}_0(\mathbf{m},\mathsf{pp},\mathsf{cm},\mathsf{aux},r_{\mathsf{wit}},\cdot)}(L_0,\mathsf{pp};r_0)$ *is* $(\bar{\mathbf{m}}, L_0') \neq \perp$, *then*

$$\begin{aligned}&\mathsf{len}_{\mathcal{E}_0,\mathcal{D}_0^{\mathcal{C}}}((\mathbf{m},\mathsf{pp},\mathsf{cm},\mathsf{aux},r_{\mathsf{wit}}),(L_0,\mathsf{pp};r_0))\\ &\leq \xi_{\mathsf{cm}} + (\dim(L_0') - \dim(L_0)) \cdot \left(1 + \xi_{\mathsf{wit}} \cdot \prod_{i=1}^{\rho-1}(\mu_i+1)\right).\end{aligned}$$

Next, the following lemma (showing that $\mathcal{D}_0$ is unlikely to abort) essentially follows from our results on Extract and Forge.

Lemma 15. *Let* A *be as per definition 8,* $\lambda, n \in \mathbb{N}$ *with* $\lambda \geq 1 + \log_2 n$, *and* $\mathbf{m} \leftarrow^{\$} \{0,1\}^n$. *If* $\chi/q \leq p/2$ *then*

$$\Pr\left[\mathcal{D}_0^{\mathcal{C}_{\mathsf{rand}}^{\mathsf{Forge}},\mathcal{E}_0(\mathbf{m},\mathsf{pp},\mathsf{cm},\mathsf{aux},r_{\mathsf{wit}},\cdot)}(L_0,\mathsf{pp}) = \perp\right] \;\leq\; \frac{1}{6}$$

where the probability is taken over $\mathsf{pp} \leftarrow^{\$} \mathsf{A.Gen}(\mathfrak{g}, 1^\lambda, 1^n)$, $(\mathsf{cm},\mathsf{aux}) \leftarrow^{\$} \mathsf{A.Acc}(\mathsf{pp}, S_{\mathbf{m}})$, $r_{\mathsf{wit}} \leftarrow^{\$} \{0,1\}^{\gamma_{\mathsf{wit}}}$, *an arbitrarily distributed random variable* $L_0 \in \mathsf{Sub}(\mathbb{F}_q^{\mu_0})$ *for which* $\mathfrak{p}||\mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L_0)$ *is fulfilled, and the randomness used both inside* $\mathcal{D}_0$ *and inside* $\mathcal{C}_{\mathsf{rand}}^{\mathsf{Forge}}$.

Proof. We prove the lemma via a union bound. By construction, $\mathcal{D}_0$ aborts if and only if for some $i \in [n]$ either the condition in Line 8 or the condition in Line 17 is satisfied. Note the first only occurs when $B^{(i)} \neq \varnothing$ while the second one only when $B^{(i)} = \varnothing$. Therefore, considering the following events

1. $\mathsf{bad}_1(i) \;:\; B^{(i)} \neq \varnothing \wedge \mathsf{res}^{(i)} = \perp$
2. $\mathsf{bad}_2(i) \;:\; B^{(i)} = \varnothing \wedge L_0^{(i+1)} \subseteq L_0^{(i)}$

we have that $\mathcal{D}_0^{\mathcal{C}_{\mathsf{rand}}^{\mathsf{Forge}},\mathcal{E}_0(\mathbf{m},\mathsf{pp},\mathsf{cm},\mathsf{aux},r_{\mathsf{wit}},\cdot)}(L_0,\mathsf{pp}) = \perp$ iff $\bigvee_{i=1}^n(\mathsf{bad}_1(i) \vee \mathsf{bad}_2(i))$. Thus we proceed bounding the probability of each event and conclude with a union bound.

– Regarding $\mathsf{bad}_1(i)$ observe that by construction $b^{(i)} \in B^{(i)}$. Therefore it implies the event

$$\omega((L_0^{(i)}, t_0), (i, b^{(i)})) \geq p \;\wedge\; \mathsf{res}^{(i)} = \perp$$

By Lemma 14, we know $\mathfrak{p}||\mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L_0^{(i)})$. Thus we can apply Lemma 13 (which bounds the success probability of the Forge subroutine) for any fixed choice of pp, cm, $S_{\mathbf{m}}$ and $s = (i, b^{(i)})$. Hence,

$$\begin{aligned}
&\Pr\left[\mathsf{bad}_1(i)\right] \\
&\leq \Pr\left[\left(\omega((L_0^{(i)}, t_0), (i, b^{(i)})) \geq p\right) \wedge \left(\mathsf{Forge}^{\mathcal{C}^{\mathsf{Forge}}_{\mathsf{rand}}}(1, L_0^{(i)}, \mathsf{pp}, \mathsf{cm}, (i, b^{(i)}), \kappa) = \bot\right)\right] \\
&\leq \left(1 - p + \frac{\chi}{q}\right)^{\kappa} \;\leq\; (1 - p/2)^{\frac{2}{p}\cdot\ln(12n)} \;\leq\; e^{-\ln(12n)} \;=\; \frac{1}{12n}\,,
\end{aligned}$$

where the third inequality follows as we assume $\chi/q \leq p/2$, and by definition $\kappa \geq (2/p)\ln(12n)$.

– Regarding $\mathsf{bad}_2(i)$ observe that it implies

$$\left(\omega((L_0^{(i)}, t_0), (i, m_i)) < p\right) \;\wedge\; \left(\mathsf{Extract}(1, \mathsf{pp}, \mathsf{cm}, (i, m_i), L_0^{(i)}) \subseteq L_0^{(i)}\right).$$

Using Lemma 14 we again observe $\mathfrak{p}||\mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L_0^{(i)})$ for any choice of $r, \mathbf{m}$. Therefore $L_0^{(i)}$, as a random variable, satisfies the hypothesis of Lemma 12. Thus

$$\begin{aligned}
&\Pr\left[\mathsf{bad}_2(i)\right] \\
&\leq \Pr\left[\left(\omega((L_0^{(i)}, t_0), (i, m_i)) < p\right) \;\wedge\; \left(\mathsf{Extract}(1, \mathsf{pp}, \mathsf{cm}, (i, m_i), L_0^{(i)}) \subseteq L_0^{(i)}\right)\right] \\
&\leq (p + n\varepsilon)\prod\nolimits_{i=1}^{\rho}(\mu_i + 1) \;=\; \frac{1}{12n}
\end{aligned}$$

where we used the definition of $p = (12n\prod_{i=1}^{\rho}(\mu_i + 1))^{-1} - n\varepsilon$.

A union bound thus yields

$$\begin{aligned}
&\Pr\left[\mathcal{D}_0^{\mathcal{C}^{\mathsf{Forge}}_{\mathsf{rand}}, \mathcal{E}_0(\mathbf{m}, \mathsf{pp}, \mathsf{cm}, \mathsf{aux}, r_{\mathsf{wit}}, \cdot)}(L_0, \mathsf{pp}) = \bot\right] \\
&\leq \sum\nolimits_{i=1}^{n} \Pr\left[\mathsf{bad}_1(i)\right] + \Pr\left[\mathsf{bad}_2(i)\right] \;=\; n\frac{1}{12n} + n\frac{1}{12n} \;=\; \frac{1}{6}\,. \qquad \square
\end{aligned}$$

4.2 Full Compression Argument

We define the full encoder-decoder pair $(\mathcal{E}, \mathcal{D})$ in Figure 7.

The decoder $\mathcal{D}$ uses another helper decoder $\overline{\mathcal{D}}$ (also defined in Figure 7), which requires an oracle $\mathcal{C}$ for supplying challenges to the Forge subroutine inside the final invocation of $\mathcal{D}_0$ (see Line 11 of $\overline{\mathcal{D}}$). The purpose of $\overline{\mathcal{D}}$ is to later allow a reduction to collision-freeness, in which only $\overline{\mathcal{D}}$ but not $\mathcal{D}$ will be required.

In the following, and in Figure 7, we use the following additional notation:

– $\gamma_{\mathcal{C}}$ denotes an upper bound on the number of random coins used by an oracle $\mathcal{C}^{\mathsf{Forge}}_{\mathsf{rand}}$ when it is passed as the oracle $\mathcal{C}$ supplied to an execution of $\mathcal{D}_0$ (i.e., $\gamma_{\mathcal{C}}$ must be large enough to provide challenges to n invocations of Forge).

$\overline{\mathcal{D}}^{\mathcal{C},\mathcal{E}^*}(\mathsf{pp})$: // Full decoder helper, taking an additional Forge challenge oracle $\mathcal{C}$

$P^{(1)} \leftarrow \{0^{\vartheta_0}\}$
// 1. Exploration Phase
Sample $i^* \leftarrow^{\$} [6\vartheta_0]$
For $i \in \{1, \ldots, i^*-1\}$:
  Sample $\mathbf{m}^{(i)} \leftarrow^{\$} \{0,1\}^n$ and $r_{\mathsf{wit}}^{(i)} \leftarrow^{\$} \{0,1\}^{\gamma_{\mathsf{wit}}}$
  $(\mathsf{cm}^{(i)}, \mathsf{aux}^{(i)}) \leftarrow \mathsf{A.Acc}(\mathsf{pp}, S_{\mathbf{m}^{(i)}})$
  $\mathsf{res} \leftarrow \mathcal{D}_0^{\mathcal{C}_{\mathsf{rand}}^{\mathsf{Forge}}, \mathcal{E}_0(\mathbf{m}^{(i)}, \mathsf{pp}, \mathsf{cm}^{(i)}, \mathsf{aux}^{(i)}, r_{\mathsf{wit}}^{(i)}, \cdot)}(P^{(i)} \times \{0^{\vartheta_1}\}, \mathsf{pp})$
  If $\mathsf{res} = \bot$: Set $P^{(i+1)} = P^{(i)}$
  Elif $\mathsf{res} = (\bar{\mathbf{m}}^{(i)}, L^{(i)})$: Set $P^{(i+1)} \leftarrow \pi^*_{\vartheta_0}(L^{(i)})$
// 2. Interactive Decoding Phase
$\mathsf{res} \leftarrow \mathcal{D}_0^{\mathcal{C},\mathcal{E}^*}(P^{(i^*)} \times \{0^{\vartheta_1}\}, \mathsf{pp})$ // i.e. with the real encoder $\mathcal{E}^*$
If $\mathsf{res} = \bot$: **Return** $\bot$
Elif $\mathsf{res} = (\bar{\mathbf{m}}, L^{(i^*)})$:
  If $\pi^*_{\vartheta_0}(L^{(i^*)}) \notin P^{(i^*)}$: **Return** $\bot$
  Else: **Return** $\bar{\mathbf{m}}$

$\mathcal{D}^{\mathcal{E}^*}(r)$ // Full decoder

Parse $r = r_{\mathsf{pp}}||r_{\mathsf{cm}}||r_{\mathsf{wit}}||r_{\mathcal{C}}||\bar{r}$
Set $\mathsf{pp} \leftarrow \mathsf{A.Gen}(\mathfrak{g}, 1^\lambda, 1^n; r_{\mathsf{pp}})$
Return $\overline{\mathcal{D}}^{\mathcal{C}_{\mathsf{rand}}^{\mathsf{Forge}}(\cdot;r_{\mathcal{C}}),\mathcal{E}^*}(\mathsf{pp};\bar{r})$

$\mathcal{E}(\mathbf{m}, r, x)$: // Full encoder

Parse $r = r_{\mathsf{pp}}||r_{\mathsf{cm}}||r_{\mathsf{wit}}||r_{\mathcal{C}}||\bar{r}$
Set $\mathsf{pp} \leftarrow \mathsf{A.Gen}(\mathfrak{g}, 1^\lambda, 1^n; r_{\mathsf{pp}})$
Set $(\mathsf{cm}, \mathsf{aux}) \leftarrow \mathsf{A.Acc}(\mathsf{pp}, S_{\mathbf{m}}; r_{\mathsf{cm}})$
Return $\mathcal{E}_0(\mathbf{m}, \mathsf{pp}, \mathsf{cm}, \mathsf{aux}, r_{\mathsf{wit}}, x)$

Fig. 7. Full encoder-decoder pair $(\mathcal{E}, \mathcal{D})$, for any fixed accumulator A and $\lambda, n \in \mathbb{N}$ with $\lambda \geq 1 + \log_2 n$. The random tape $r \in \{0,1\}^\gamma$ given to $\mathcal{E}$ and $\mathcal{D}$ is split into $r_{\mathsf{pp}} \in \{0,1\}^{\gamma_{\mathsf{pp}}}$, $r_{\mathsf{cm}} \in \{0,1\}^{\gamma_{\mathsf{cm}}}$, $r_{\mathsf{wit}} \in \{0,1\}^{\gamma_{\mathsf{wit}}}$, $r_{\mathcal{C}} \in \{0,1\}^{\gamma_{\mathcal{C}}}$, and $\bar{r} \in \{0,1\}^{\gamma_{\overline{\mathcal{D}}}}$.

- $\gamma_{\overline{\mathcal{D}}}$ denotes an upper bound on the number of random coins used by $\overline{\mathcal{D}}$ (which includes up to $6\vartheta_0$ random messages $\mathbf{m}^{(i)}$, random tapes $r_{\mathsf{wit}}^{(i)}$, and random coins used within up to $6\vartheta_0$ invocations of $\mathcal{D}_0$ as well as their oracles $\mathcal{C}_{\mathsf{rand}}^{\mathsf{Forge}}$)
- $\gamma := \gamma_{\mathsf{pp}} + r_{\mathsf{cm}} + r_{\mathsf{wit}} + \gamma_{\mathcal{C}} + \gamma_{\overline{\mathcal{D}}}$ denotes an upper bound on the entire randomness used by $\mathcal{D}$ and $\mathcal{E}$

Inside $\mathcal{E}$ and $\mathcal{D}$, we make the randomness r explicit in the definition of $\mathcal{E}$ and $\mathcal{D}$ so that they become deterministic machines, allowing us to then apply a compression argument. Hence, the randomness r is composed of $r = r_{\mathsf{pp}}||r_{\mathsf{cm}}||r_{\mathsf{wit}}||r_{\mathcal{C}}||\bar{r}$, with $r_{\mathsf{pp}} \in \{0,1\}^{\gamma_{\mathsf{pp}}}$, $r_{\mathsf{cm}} \in \{0,1\}^{\gamma_{\mathsf{cm}}}$, $r_{\mathsf{wit}} \in \{0,1\}^{\gamma_{\mathsf{wit}}}$, $r_{\mathcal{C}} \in \{0,1\}^{\gamma_{\mathcal{C}}}$, $\bar{r} \in \{0,1\}^{\gamma_{\overline{\mathcal{D}}}}$.

We again start with some basic facts about $\mathcal{D}$ and $\mathcal{E}$, which we prove in Appendix A.4.

Lemma 16. *Given* A *as per Definition 8,* $\lambda, n \in \mathbb{N}$ *with* $\lambda \geq 1 + \log_2 n$, $\mathbf{m} \in \{0,1\}^n$, *and randomness* $r \in \{0,1\}^\gamma$, *the following holds:*

- *When running* $\mathcal{D}^{\mathcal{E}(\mathbf{m},r,\cdot)}(r)$, *then* $P^{(i)} \subseteq P^{(i+1)}$ *(for* $i \in [i^* - 1]$*) and* $\mathfrak{p} \in \mathrm{Ann}_{\mathbb{G}}(P^{(i)})$ *(for* $i \in [i^*]$*), where* i^* *and* $P^{(i)}$ *refer to the variables inside* $\overline{\mathcal{D}}$.
- *If* $\mathcal{D}^{\mathcal{E}(\mathbf{m},r,\cdot)}(r) = \mathbf{m}$, *then*

$$\mathsf{len}_{\mathcal{E},\mathcal{D}}((\mathbf{m},r),r) \;\leq\; \xi_{\mathsf{cm}} + \vartheta_0 \cdot \left(1 + \xi_{\mathsf{wit}} \cdot \prod\nolimits_{i=1}^{\rho-1}(\mu_i + 1)\right)$$

Next, we show that $\mathcal{D}$ is unlikely to abort, which follows (1) from Lemma 15, and (2) from the fact that, with high probability, no new relations about $\mathfrak{p}$ are found in the final $\mathcal{D}_0$-call due to the preceding exploration phase that is executed for a random number of iterations.

Lemma 17. *Given* A *as per Definition 8 and* $\lambda, n \in \mathbb{N}$ *with* $\lambda \geq 1 + \log_2 n$. *If* $\chi/q \leq p/2$ *then*

$$\Pr\left[\mathcal{D}^{\mathcal{E}(\mathbf{m},r,\cdot)}(r) = \perp\right] \leq \frac{1}{3}\,,$$

with probability taken over message $\mathbf{m} \leftarrow^{\$} \{0,1\}^n$ *and randomness* $r \leftarrow^{\$} \{0,1\}^\gamma$.

Proof. For $\mathcal{D}$ to return $\perp$ we distinguish two events: (i) it aborts in Line 12; (ii) it aborts in Line 14:

1. Abort_1 : $\overline{\mathcal{D}}$ returns $\perp$ in Line 12 because $\mathcal{D}_0^{\mathcal{C}_{\mathsf{rand}}^{\mathsf{Forge}}(\cdot;r_{\mathcal{C}}),\mathcal{E}^*}(P^{(i^*)} \times \{0^{\vartheta_1}\}, \mathsf{pp})$ returned $\perp$;
2. Abort_2 : $\pi^*_{\vartheta_0}(L^{(i^*)}) \not\subseteq P^{(i^*)}$;

We prove each to occur with probability asymptotically $\leq 1/6$, which by union bound implies the thesis.

Basic decoder aborts. Regarding Abort_1 we begin by observing that, by construction, $\mathcal{E}(\mathbf{m}, r, \cdot) = \mathcal{E}_0(\mathbf{m}, \mathsf{pp}, \mathsf{cm}, \mathsf{aux}, r_{\mathsf{wit}}, \cdot)$. By Lemma 16, for all i the space $P^{(i)}$ is such that $\mathfrak{p} \in \mathrm{Ann}_{\mathbb{G}}(P^{(i)})$ and in particular $\mathfrak{p}||\mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(P^{(i^*)} \times \{0^{\vartheta_0}\})$. We can then apply Lemma 15, which shows

$$\Pr\left[\mathsf{Abort}_1\right] \;=\; \Pr\left[\mathcal{D}_0^{\mathcal{E}_0(\mathbf{m},\mathsf{pp},\mathsf{cm},\mathsf{aux},r_{\mathsf{wit}},\cdot)}(P^{(i^*)} \times \{0^{\vartheta_1}\}, \mathsf{pp}) = \perp\right] \;\leq\; \frac{1}{6}.$$

New relations are found. Let us fix arbitrary randomness $r_{\mathsf{pp}}||r_{\mathsf{cm}}||r_{\mathsf{wit}}$. We will prove $\Pr[\mathsf{Abort}_2] \leq 1/6$ over the random choice of message $\mathbf{m}$ and the remainder $r_{\mathcal{C}}||\bar{r}$ of r.

To do so, let us first define a new random process $\mathsf{Update}(P)$ as follows, where $P \leq \mathbb{F}_q^{\vartheta_0}$ is a vector space:

1. Sample $\mathbf{m} \leftarrow^{\$} \{0,1\}^n$ and generate $(\mathsf{cm}, \mathsf{aux}) \leftarrow \mathsf{A.Acc}(\mathsf{pp}, S_{\mathbf{m}})$.
2. Sample $r_{\mathsf{wit}} \leftarrow^{\$} \{0,1\}^{\gamma_{\mathsf{wit}}}$ and run $L \leftarrow \mathcal{D}_0^{\mathcal{C}_{\mathsf{rand}}^{\mathsf{Forge}},\mathcal{E}_0(\mathbf{m},\mathsf{pp},\mathsf{cm},\mathsf{aux},r_{\mathsf{wit}},\cdot)}(P \times \{0^{\vartheta_1}\}, \mathsf{pp})$ (note that this uses randomness within $\mathcal{C}_{\mathsf{rand}}^{\mathsf{Forge}}$ and inside $\mathcal{D}_0$).

3. If $L = \bot$, return P. Otherwise, return $\pi^*_{\vartheta_0}(L)$.

First of all note that, by Lemma 14, we always have $P \subseteq P'$ (for any potential output of $\mathsf{Update}(P)$), and hence we will treat $P' \not\subseteq P$ and $P \subsetneq P'$ as equivalent.

Now observe that the event $\mathcal{D}^{\mathcal{E}(\mathbf{m},r,\cdot)}(r) = \bot$ (for random $\mathbf{m}$ and $r_{\mathcal{C}}||\overline{r}$) is equivalent to the event $P^{(i^*+1)} \not\subseteq P^{(i^*)}$ in the following process: sample $i^* \leftarrow^{\$} [6\vartheta_0]$, set $P^{(1)} \leftarrow \{0^{\vartheta_0}\}$ and for $i \in [i^*]$ set $P^{(i+1)} \leftarrow \mathsf{Update}(P^{(i)})$.

More precisely:

$$\Pr\left[\mathsf{Abort}_2\right] = \Pr\left[\begin{array}{l} i^* \leftarrow^{\$} [6\vartheta_0], \\ P^{(1)} \leftarrow \{0^{\vartheta_0}\}, \\ P^{(i+1)} \leftarrow \mathsf{Update}(P^{(i)})\ \forall i \in [i^*-1] \end{array} : P^{(i^*+1)} \not\subseteq P^{(i^*)}\right],$$

where the left probability is taken over random $\mathbf{m}$ and $r_{\mathcal{C}}||\overline{r}$ and running $\mathcal{D}^{\mathcal{E}(\mathbf{m},r,\cdot)}(r)$, while the right one is taken over the randomness of i^* and the randomness inside Update.

The above is then equal to (where we define $P^{(1)} := \widehat{P}^{(1)} := \{0^{\vartheta_0}\}$):

$$\begin{aligned}
&= \Pr\left[\begin{array}{l} i^* \leftarrow^{\$} [6\vartheta_0], \\ P^{(1)} \leftarrow \{0^{\vartheta_0}\}, \\ P^{(i+1)} \leftarrow \mathsf{Update}(P^{(i)})\ \forall i \in [6\vartheta_0] \end{array} : P^{(i^*+1)} \not\subseteq P^{(i^*)}\right] \\
&= \sum_{\widehat{P}^{(2)} \leq \cdots \leq \widehat{P}^{(6\vartheta_0+1)} \leq \mathbb{F}_q^{\vartheta_0}} \left(\Pr\left[i^* \leftarrow^{\$} [6\vartheta_0] : \widehat{P}^{(i^*+1)} \not\subseteq \widehat{P}^{(i^*)}\right] \right. \\
&\qquad \left. \cdot \prod_{i=1}^{6\vartheta_0} \Pr\left[P^{(i+1)} \leftarrow \mathsf{Update}(\widehat{P}^{(i)}) : P^{(i+1)} = \widehat{P}^{(i+1)}\right] \right) \\
&\leq \sum_{\widehat{P}^{(2)} \leq \cdots \leq \widehat{P}^{(6\vartheta_0+1)} \leq \mathbb{F}_q^{\vartheta_0}} \frac{1}{6} \sum_{i=1}^{6\vartheta_0} \Pr\left[P^{(i+1)} \leftarrow \mathsf{Update}(\widehat{P}^{(i)}) : P^{(i+1)} = \widehat{P}^{(i+1)}\right] \\
&= \frac{1}{6}
\end{aligned}$$

The very first equality holds since we just added dummy updates after i^*, which do not change the event $P^{(i^*+1)} \not\subseteq P^{(i^*)}$. The inequality holds as $\Pr[i^* \leftarrow^{\$} [6\vartheta_0] : \widehat{P}^{(i^*+1)} \not\subseteq \widehat{P}^{(i^*)}] \leq \frac{1}{6}$, which is due to the fact that $\widehat{P}^{(i)} \subseteq \widehat{P}^{(i+1)}$ for all $i \in [\vartheta_0]$ and all of these are subspaces of $\mathbb{F}_q^{\vartheta_0}$ (i.e., there can be at most ϑ_0 indices $i \in [\vartheta_0]$ for which $\widehat{P}^{(i)} \subsetneq \widehat{P}^{(i+1)}$). □

4.3 Lower Bound

Now we are ready to present our main result.

Theorem 1. *Let* A *be an accumulator as per Definition 8, and* $n = n(\lambda) \geq 1$ *a polynomially bounded function. Then, asymptotically in* $\lambda \in \mathbb{N}$,[12] *it holds that*

$$2 + \xi_{\mathsf{cm}} + \vartheta_1 + \vartheta_1 \cdot \xi_{\mathsf{wit}} \cdot \prod_{i=1}^{\rho-1} (\mu_i + 1) \ \geq \ n$$

[12] Formally, there exists some λ_0 s.t. this inequality holds for all $\lambda \geq \lambda_0$.

Proof. Towards contradiction assume the asymptotic bound above does not hold, i.e. that it is false for infinitely many $\lambda \in \Lambda_0 \subseteq \mathbb{N}$. Observe this implies that $\prod_{i=1}^{\rho-1}(\mu_i+1) < n$ for all $\lambda \in \Lambda_0$ (because of $\vartheta_1 \geq 0$; and if we had $\xi_{\mathsf{wit}} = 0$ then $\mu_i = 0$ for every i, i.e., the product would be equal to $1 \leq n$).

In particular, given our definition of p, we have $p \geq (12n^2(\mu_\rho+1))^{-1} - n\varepsilon$ for all $\lambda \in \Lambda_0$ as well as $\kappa = \lceil 2\ln(12n)/p \rceil \leq 1 + 2/n(\mu_\rho+1)$. Since μ_ρ is bounded by polynomials in λ (otherwise the Theorem would be trivial), this means that both $1/p$ and κ are bounded by polynomials in λ for all $\lambda \in \Lambda_0$.

We further refine λ as follows. Since both $\varepsilon(\lambda, n(\lambda))$ and χ/q are negligible in λ (due to correctness of A and due to $\mathsf{A.Ver}$ being query-efficient, respectively), and because q is exponential in λ, we can find a λ_1 such that $\lambda \geq \lambda_1$ implies

$$\frac{1}{24n^2(\mu_\rho+1)} \geq \frac{n\varepsilon}{2} + \frac{\chi}{q} \quad \text{and} \quad \lambda \geq 1 + \log_2 n\,. \tag{2}$$

Let us denote $\Lambda_1 = \Lambda_0 \cap \{\lambda \in \mathbb{N} : \lambda \geq \lambda_1\}$, which is still infinite. Then, combining this with the bound on p from above, we have $p/2 \geq \chi/q$ for all $\lambda \in \Lambda_1$.

Next, consider the interactive compression argument $(\mathcal{E}, \mathcal{D})$ from Figure 7 with message space $\{0,1\}^\lambda$, and common random coins $\{0,1\}^\gamma$. With $\mathbf{m}, r$ uniformly sampled, let E be the event $\mathcal{D}^{\mathcal{E}(\mathbf{m},r,\cdot)}(r) = \mathbf{m}$. Then, Lemma 16 implies that for all $\lambda \in \mathbb{N}$ and each choice of $\mathbf{m}, r$ for which E holds, we have

$$\mathsf{len}_{\mathcal{E},\mathcal{D}}((\mathbf{m},r),r) \;\leq\; \xi_{\mathsf{cm}} + \vartheta_0 \left(1 + \xi_{\mathsf{wit}} \cdot \prod\nolimits_{i=1}^{\rho-1}(\mu_i+1)\right).$$

Next we need to bound $\Pr[E]$. For $\lambda \in \Lambda_1$ we can apply Lemma 17, meaning that $\Pr\left[\mathcal{D}^{\mathcal{E}(\mathbf{m},r,\cdot)}(r) = \bot\right] \leq 1/3$. To further bound the probability that $\mathcal{D}$ outputs an incorrect $\bar{\mathbf{m}} \neq \mathbf{m}$ without aborting, we will reduce to the collision-freeness of A against query-bounded TS-machines. A full description of the reduction is given in Figure 8.

Observe that in both the collision-freeness experiment $\mathsf{Exp}^{\mathsf{coll\text{-}free}}_{\mathsf{A},\mathcal{A},q}(\lambda, n)$ and in the execution of $\mathcal{D}^{\mathcal{E}(\mathbf{m},r,\cdot)}(r)$ (for a uniform choice of $\mathbf{m}, r$), exactly one call to $\overline{\mathcal{D}}$ is made. Let $\bar{\mathbf{m}}$ denote $\overline{\mathcal{D}}$'s return value, and let Abort be the event that $\bar{\mathbf{m}} = \bot$.

The two executions of $\overline{\mathcal{D}}$ have identical behavior in their respective experiments: (1) $\overline{\mathcal{D}}$'s input pp is generated honestly by $\mathsf{A.Gen}$ in both cases, (2) the encoder oracles $\mathcal{E}_0(\mathbf{m}, \mathsf{pp}, \mathsf{cm}, \mathsf{aux}, r_{\mathsf{wit}}, \cdot)$ and $\mathcal{E}(\mathbf{m}, r, \cdot)$ have identical behavior (both utilize a uniformly random message $\mathbf{m}$ and r_{wit}, and both utilize $(\mathsf{cm}, \mathsf{aux})$ generated by $\mathsf{A.Acc}(\mathsf{pp}, S_{\mathbf{m}})$), and (3) the oracles $\mathcal{C}$ and $\mathcal{C}^{\mathsf{Forge}}_{\mathsf{rand}}$ have identical behavior (both always reply with uniformly random challenges; note that in case of $\mathcal{C}$ as defined in Line 7 of $\mathcal{A}$ this is due to the definition of the collision-freeness experiment).

Denoting by $\mathsf{tr}_{j',k'} = (t_1^{(k')}, c_1^{(k)}, \ldots, t_\rho^{(k')}, c_\rho^{(k')})$ the transcript computed in Line 12 of Forge's k'-th execution when it is queried on $s = (j', b_{j'})$, we now have (where, by the discussion above it doesn't matter whether the probability

$\mathcal{A}(1^\lambda, 1^n, \mathsf{pp})$:

1 : **If** $\prod_{i=1}^{\rho-1}(\mu_i+1) > n$: **Return** $\perp$ // Do nothing when $\lambda \notin \Lambda_0$
2 : Sample $j_0 \leftarrow^{\$} [n], k_0 \leftarrow^{\$} [\kappa]$ // Guess s.t. $\bar{m}_{j_0} \neq m_{j_0}$ and Forge wins the k_0-th attempt.
3 : Sample $\mathbf{m} \leftarrow^{\$} \{0,1\}$, $r_{\mathsf{wit}} \leftarrow^{\$} \{0,1\}^{\gamma_{\mathsf{wit}}}$
4 : **Send** $S_{\mathbf{m}}$ to the challenger and wait for $\mathsf{cm}, \mathsf{aux}$
5 : **Send** $(j_0, 1-m_{j_0})$ // element not in S we prove membership of
6 : $\overline{\mathcal{D}}^{\mathcal{C},\mathcal{E}_0(\mathbf{m},\mathsf{pp},\mathsf{cm},\mathsf{aux},r_{\mathsf{wit}},\cdot)}(\mathsf{pp})$ // oracle $\mathcal{C}$ defined as follows:
7 : **When** it queries $\mathcal{C}(s,k,i,w_i^{(k)})$:
8 : Parse $s = (j,b)$
9 : **If** $(j,k) \neq (j_0,k_0)$: **Reply** $c_i \leftarrow^{\$} \{0,1\}^{\ell_i}$
10 : **Else**: **Send** $w_i^{(k)}$ to the challenger, wait for c_i; **Reply** c_i

Fig. 8. Adversary against the collision-freeness of A against query-efficient type-safe attacks, for any fixed accumulator A and $\lambda, n \in \mathbb{N}$ with $\lambda \geq 1 + \log_2 n$. The parameter κ is the defined at the beginning of Section 4.

space is the collision-freeness experiment or the execution of $\mathcal{D}^{\mathcal{E}(\mathbf{m},r)}(r)$):

$$
\begin{aligned}
&\Pr[\neg\mathsf{Abort} \wedge \bar{\mathbf{m}} \neq \mathbf{m}] \\
&\leq \sum_{j'=1}^{n} \Pr[\neg\mathsf{Abort} \wedge \bar{m}_{j'} \neq m_{j'}] \\
&\leq \sum_{j'=1}^{n} \Pr\left[\begin{array}{l} B_{j'} \neq \varnothing \ \wedge\ b_{j'} \neq m_{j'} \wedge \\ (\vee_{k'=1}^{\kappa} \mathsf{A.Ver}(\mathsf{pp},\mathsf{cm},(j',b_{j'}),\mathsf{tr}_{j',k'}) = 1)\end{array}\right] \qquad (3)\\
&\leq \sum_{j'=1}^{n}\sum_{k'=1}^{\kappa} \Pr\left[\begin{array}{l} B_{j'} \neq \varnothing \wedge b_{j'} \neq m_{j'} \wedge \\ \mathsf{A.Ver}(\mathsf{pp},\mathsf{cm},(j',b_{j'}),\mathsf{tr}_{j',k'}) = 1\end{array}\right] \\
&\leq \sum_{j'=1}^{n}\sum_{k'=1}^{\kappa} \Pr[\mathsf{A.Ver}(\mathsf{pp},\mathsf{cm},(j',1-m_{j'}),\mathsf{tr}_{j',k'}) = 1] \qquad (4)\\
&= \sum_{j'=1}^{n}\sum_{k'=1}^{\kappa} \Pr\left[\begin{array}{l} \mathsf{A.Ver}(\mathsf{pp},\mathsf{cm},(j',1-m_{j'}),\mathsf{tr}_{j',k'}) = 1 \wedge \\ j_0 = j' \ \wedge\ k_0 = k'\end{array}\right] \cdot n\kappa \qquad (5)\\
&= \sum_{j'=1}^{n}\sum_{k'=1}^{\kappa} \Pr\left[\mathsf{Exp}^{\mathsf{coll\text{-}free}}_{\mathsf{A},\mathcal{A},q}(\lambda,n) = 1 \wedge j_0 = j' \wedge k_0 = k'\right] \cdot n\kappa \qquad (6)\\
&= \mathsf{Adv}^{\mathsf{coll\text{-}free}}_{\mathsf{A},\mathcal{A}}(\lambda,n) \cdot n\kappa.
\end{aligned}
$$

(3) holds because whenever $\neg\mathsf{Abort}$ then $\bar{m}_{j'} \neq m_{j'}$ implies that $B_{j'} \neq \varnothing$, since otherwise $\mathcal{D}_0$ sets $\bar{m}_{j'} \leftarrow (\mathtt{bit}, j')$ in Line 14, which is always equal to $m_{j'}$. For the same reason it implies Forge must have returned `accept` on the set element $(j, \min(B_{j'})) = (j, b_{j'})$ and therefore $b_{j'} = \bar{m}_{j'} \neq m_{j'}$. (4) holds as $B_{j'} \neq \varnothing$ implies $b_{j'} \in \{0,1\}$ and $b_{j'} \neq m_{j'} \in \{0,1\}$ implies $b_{j'} = 1 - m_{j'}$. (5) holds since j_0, k_0, as sampled by $\mathcal{A}$, are statistically independent from any randomness visible to $\overline{\mathcal{D}}$ (in fact, we could modify $\mathsf{Exp}^{\mathsf{coll\text{-}free}}_{\mathsf{A},\mathcal{A},q}(\lambda,n)$ so that j_0, k_0 are only sampled at the very end of the experiment). Finally, (6) holds since the transcript tr_{j_0,k_0} is exactly the transcript that is checked by the collision-freeness challenger for validity.

To conclude we remark that $\mathcal{A}$ is query-efficient (i.e., the number of group operations is polynomial): since n and κ are both polynomial in λ as shown before, the only computationally inefficient step is Line 3 of Forge where the optimal strategy is computed. However, doing so requires no group operations. By collision-freeness the advantage of $\mathcal{A}$ is smaller than a negligible function $\varepsilon'(\lambda)$. Thus for all $\lambda \in \Lambda_1$, $\Pr[\bar{\mathbf{m}} \neq \mathbf{m} \wedge \neg\mathsf{Abort}] \leq \kappa n \varepsilon'(\lambda)$, and there exists a λ_2 s.t. this term is $\leq 1/6$ for all $\lambda \in \Lambda_2$ where $\Lambda_2 := \Lambda_1 \cap \{\lambda \in \mathbb{N} : \lambda \geq \lambda_2\}$.

We therefore get a bound on $\Pr[E]$ for $\lambda \in \Lambda_2$ as follows:

$$\Pr[E] \leq \Pr\left[\mathcal{D}^{\mathcal{E}(\mathbf{m},r,\cdot)} = \perp\right] + \Pr\left[\mathcal{D}^{\mathcal{E}(\mathbf{m},r,\cdot)} \neq \perp \wedge \bar{\mathbf{m}} \neq \mathbf{m}\right] \;\leq\; \frac{1}{3} + \frac{1}{6} \;=\; \frac{1}{2}.$$

In conclusion, by Lemma 3 we have shown that for infinitely many $\lambda \in \Lambda_2$

$$\begin{aligned} n \;&\leq\; 1 + \left(\xi_{\mathsf{cm}} + \vartheta_1 \cdot \left(1 + \xi_{\mathsf{wit}} \prod_{i=1}^{\rho-1} (\mu_i + 1)\right)\right) - \log \Pr[E] \\ &\leq\; 2 + \xi_{\mathsf{cm}} + \vartheta_1 \cdot \left(1 + \xi_{\mathsf{wit}} \prod_{i=1}^{\rho-1} (\mu_i + 1)\right) \end{aligned}$$

which contradicts our initial hypothesis. □

Corollary 1. *Let* A *be an accumulator as per Definition 8, and* $n = n(\lambda) \geq 1$ *a polynomially bounded function. Let* $\mu = \sum_{i=1}^{\rho-1} \mu_i$ *be the number of group elements per proof (except those sent in the last round). Then, asymptotically in* $\lambda \in \mathbb{N}$*, it holds that*

$$2 + \xi_{\mathsf{cm}} + \vartheta_1 + \vartheta_1 \cdot \xi_{\mathsf{wit}} \cdot 2^{\mu} \geq n \;,$$

and in particular also

$$\xi_{\mathsf{cm}} \cdot \xi_{\mathsf{wit}} \cdot 2^{\mu} = \Omega(n) \;.$$

Furthermore,

$$\xi_{\mathsf{cm}} \cdot \xi_{\mathsf{wit}} \cdot \mu^{\rho-1} = \Omega(n) \;.$$

Proof. The first bound follows from Theorem 1 combined with

$$\prod_{i=1}^{\rho-1} (\mu_i + 1) = \prod_{\substack{i \in [\rho-1] \\ \mu_i > 0}} (\mu_i + 1) \leq \prod_{\substack{i \in [\rho-1] \\ \mu_i > 0}} 2^{\mu_i} = 2^{\sum_{i \in [\rho-1] : \mu_i > 0} \mu_i} = 2^{\mu}$$

The second bound follows from the first one due to $\vartheta_1 \leq \xi_{\mathsf{cm}}$.

The third bound follows from Theorem 1 combined with $\prod_{i=1}^{\rho-1}(\mu_i+1) \leq \mu^{\rho-1}$ whenever $\mu_i \geq 1$ for at least two i's (or if $\rho = 1$). If $\mu_i \geq 1$ for only one i and simultaneously $\rho \geq 2$, then we have $\prod_{i=1}^{\rho-1}(\mu_i+1) = \mu+1 \leq O(\mu^{\rho-1})$ instead. □

References

AAAG25. Hamza Abusalah, Gaspard Anthoine, Gennaro Avitabile, and Emanuele Giunta. Lower bounding update frequency in short accumulators and vector commitments. Cryptology ePrint Archive, Report 2025/1558, 2025.

ABG+25. Gennaro Avitabile, Vincenzo Botta, Emanuele Giunta, Marcin Mielniczuk, and Francesco Migliaro. The malice of ELFs: Practical anamorphic-resistant encryption without random oracles. Cryptology ePrint Archive, Report 2025/305, 2025.

AC20. Thomas Attema and Ronald Cramer. Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 513–543. Springer, Cham, August 2020.

AHO12. Masayuki Abe, Kristiyan Haralambiev, and Miyako Ohkubo. Group to group commitments do not shrink. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 301–317. Springer, Berlin, Heidelberg, April 2012.

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

BBF19. Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 561–586. Springer, Cham, August 2019.

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. Springer, Berlin, Heidelberg, May 2016.

BCCK25. Joseph Bonneau, Jessica Chen, Miranda Christ, and Ioanna Karantaidou. Merkle mountain ranges are optimal: On witness update frequency for cryptographic accumulators. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part II*, volume 16001 of *LNCS*, pages 170–202. Springer, Cham, August 2025.

BCD+17. Foteini Baldimtsi, Jan Camenisch, Maria Dubovitskaya, Anna Lysyanskaya, Leonid Reyzin, Kai Samelin, and Sophia Yakoubov. Accumulators with applications to anonymity-preserving revocation. In *2017 IEEE European Symposium on Security and Privacy*, pages 301–315. IEEE Computer Society Press, April 2017.

BCY20. Foteini Baldimtsi, Ran Canetti, and Sophia Yakoubov. Universally composable accumulators. In Stanislaw Jarecki, editor, *CT-RSA 2020*, volume 12006 of *LNCS*, pages 638–666. Springer, Cham, February 2020.

BdM94. Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 274–285. Springer, Berlin, Heidelberg, May 1994.

BGZ23. Dan Boneh, Jiaxin Guan, and Mark Zhandry. A lower bound on the length of signatures based on group actions and generic isogenies. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 507–531. Springer, Cham, April 2023.

CB23. Miranda Christ and Joseph Bonneau. Limits on revocable proof systems, with implications for stateless blockchains. In Foteini Baldimtsi and Christian Cachin, editors, *FC 2023, Part II*, volume 13951 of *LNCS*, pages 54–71. Springer, Cham, May 2023.

CF13. Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 55–72. Springer, Berlin, Heidelberg, February / March 2013.

CFGG22. Dario Catalano, Dario Fiore, Rosario Gennaro, and Emanuele Giunta. On the impossibility of algebraic vector commitments in pairing-free groups. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part II*, volume 13748 of *LNCS*, pages 274–299. Springer, Cham, November 2022.

CH10. Philippe Camacho and Alejandro Hevia. On the impossibility of batch update for cryptographic accumulators. In Michel Abdalla and Paulo S. L. M. Barreto, editors, *LATINCRYPT 2010*, volume 6212 of *LNCS*, pages 178–188. Springer, Berlin, Heidelberg, August 2010.

CHJ+22. Heewon Chung, Kyoohyung Han, Chanyang Ju, Myungsun Kim, and Jae Hong Seo. Bulletproofs+: Shorter proofs for a privacy-enhanced distributed ledger. *Ieee Access*, 10:42081–42096, 2022.

CPZ18. Alexander Chepurnoy, Charalampos Papamanthou, and Yupeng Zhang. Edrax: A cryptocurrency with stateless transaction validation. Cryptology ePrint Archive, Report 2018/968, 2018.

DFKP15. Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 585–605. Springer, Berlin, Heidelberg, August 2015.

DHH+21. Nico Döttling, Dominik Hartmann, Dennis Hofheinz, Eike Kiltz, Sven Schäge, and Bogdan Ursu. On the impossibility of purely algebraic signatures. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 317–349. Springer, Cham, November 2021.

DKT26. Marian Dietz, Julia Kastner, and Stefano Tessaro. On the impossibility of round-optimal pairing-free blind signatures in the ROM. Cryptology ePrint Archive, Paper 2026/090, 2026.

DORS08. Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM journal on computing*, 38(1):97–139, 2008.

EKRN24. Liam Eagen, Sanket Kanjalkar, Tim Ruffing, and Jonas Nick. Bulletproofs++: Next generation confidential transactions via reciprocal set membership arguments. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part V*, volume 14655 of *LNCS*, pages 249–279. Springer, Cham, May 2024.

FN02. Nelly Fazio and Antonio Nicolosi. Cryptographic accumulators: Definitions, constructions and applications. *Paper written for course at New York University: www. cs. nyu. edu/nicolosi/papers/accumulators. pdf*, 24, 2002.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

Giu23. Emanuele Giunta. On the impossibility of algebraic NIZK in pairing-free groups. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 702–730. Springer, Cham, August 2023.

GM24. Jiaxin Guan and Hart Montgomery. On sequential functions and fine-grained cryptography. In Leonid Reyzin and Douglas Stebila, editors,

CRYPTO 2024, Part V, volume 14924 of *LNCS*, pages 393–428. Springer, Cham, August 2024.

GRWZ20. Sergey Gorbunov, Leonid Reyzin, Hoeteck Wee, and Zhenfei Zhang. Pointproofs: Aggregating proofs for multiple vector commitments. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 2007–2023. ACM Press, November 2020.

GT00. Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st FOCS*, pages 305–313. IEEE Computer Society Press, November 2000.

HMQS23. Mohammad Hajiabadi, Mohammad Mahmoody, Wei Qi, and Sara Sarfaraz. Lower bounds on assumptions behind registration-based encryption. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part II*, volume 14370 of *LNCS*, pages 306–334. Springer, Cham, November / December 2023.

JK07. Ari Juels and Burton S. Kaliski, Jr. Pors: proofs of retrievability for large files. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 2007*, pages 584–597. ACM Press, October 2007.

Kil92. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

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. Springer, Berlin, Heidelberg, December 2010.

LRY16. Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016*, volume 55 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl, July 2016.

LY10. Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 499–517. Springer, Berlin, Heidelberg, February 2010.

Mau05. Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Berlin, Heidelberg, December 2005.

Mer88. Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 369–378. Springer, Berlin, Heidelberg, August 1988.

MMV13. Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. Publicly verifiable proofs of sequential work. In Robert D. Kleinberg, editor, *ITCS 2013*, pages 373–388. ACM, January 2013.

MRK03. Silvio Micali, Michael O. Rabin, and Joe Kilian. Zero-knowledge sets. In *44th FOCS*, pages 80–91. IEEE Computer Society Press, October 2003.

PRV12. Periklis A. Papakonstantinou, Charles W. Rackoff, and Yevgeniy Vahlis. How powerful are the DDH hard groups? Cryptology ePrint Archive, Report 2012/653, 2012.

Qi25. Wei Qi. Tight lower bound on witness update frequency in additive positive accumulators. Cryptology ePrint Archive, Report 2025/1378, 2025.

RSS20. Lior Rotem, Gil Segev, and Ido Shahaf. Generic-group delay functions require hidden-order groups. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 155–180. Springer, Cham, May 2020.

SGS20. Gili Schul-Ganz and Gil Segev. Accumulators in (and beyond) generic groups: Non-trivial batch verification requires interaction. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 77–107. Springer, Cham, November 2020.

SGS21. Gili Schul-Ganz and Gil Segev. Generic-group identity-based encryption: A tight impossibility result. In Stefano Tessaro, editor, *ITC 2021*, volume 199 of *LIPIcs*, pages 26:1–26:23. Schloss Dagstuhl, July 2021.

Zha22. Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 66–96. Springer, Cham, August 2022.

ZZ18. Mark Zhandry and Cong Zhang. Impossibility of order-revealing encryption in idealized models. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 129–158. Springer, Cham, November 2018.

A Postponed Proofs

A.1 Compression Arguments

Proof of Lemma 3. Given m, r, define y_i as the i-th query of $\mathcal{D}$ and $x_i = \mathcal{E}((m,r), x_i)$ as the corresponding response. Then, note that the output of $\mathcal{D}^{\mathcal{E}((m,r),\cdot)}(r)$ is a function of $r, x_1, \ldots, x_\chi$. Let us denote by $\mathcal{D}^*(r, x_1, \ldots, x_\chi)$ such function. Then, by Lemma 1, Lemma 2 and the fact that E implies that $\mathcal{D}$ correctly returns m

$$\begin{aligned}
0 = -\log 1 &= -\log \Pr\left[\mathcal{D}^{\mathcal{E}((m,r),\cdot)}(r) = m \,\middle|\, E\right] \\
&= -\log \Pr\left[\mathcal{D}^*(r, \mathbf{x}) = m \mid E\right] \\
&\geq \mathrm{H}_\infty(m | r, \mathbf{x}; E) \\
&\geq \mathrm{H}_\infty(m | r; E) - \mathrm{H}_0(\mathbf{x} | r; E) \\
&\geq \mathrm{H}_\infty(m | r) + \log \Pr[E] - \mathrm{H}_0(\mathbf{x} | r; E) \\
&= \mathrm{H}_\infty(m) + \log \Pr[E] - \mathrm{H}_0(\mathbf{x} | r; E)\,.
\end{aligned}$$

To conclude we need an upper bound on the max-entropy. For any $(m,r) \in E$, the bit length of $\mathbf{x}$ is $\mathsf{len}_{\mathcal{E},\mathcal{D}}((m,r), r) \leq L$ with $L = \max_{(m,r)\in E} \mathsf{len}_{\mathcal{E},\mathcal{D}}((m,r), r)$. Define the following sets:

$$\begin{aligned}
X_{r_0} &= \{\mathbf{x}_0 \in \{0,1\}^* \,:\, \Pr[\mathbf{x} = \mathbf{x}_0 \,|\, r = r_0,\ E] > 0\} \\
X_{m_0,r_0} &= \{\mathbf{x}_0 \in \{0,1\}^* \,:\, \Pr[\mathbf{x} = \mathbf{x}_0,\ m = m_0 \,|\, r = r_0,\ E] > 0\}.
\end{aligned}$$

By definition $\mathrm{H}_0(\mathbf{x}|r;E) = \log(\max_{r_0 \in R} |X_{r_0}|)$. By total probability $X_{r_0} = \cup_{m_0 \in M} X_{m_0,r_0}$ and since $\Pr[m = m_0 | r = r_0,\ E] = 0$ for $(m_0, r_0) \notin E$ then $X_{r_0} = \cup_{m_0 \in M:(m_0,r_0)\in E} X_{r_0,m_0}$. For $(r_0, m_0) \in E$ instead by definition of L

we have that $X_{r_0,m_0} \subseteq \cup_{j=1}^{L}\{0,1\}^j$. In particular $X_{r_0} \subseteq \cup_{j=1}^{L}\{0,1\}^j$ for any r_0 and therefore

$$\mathrm{H}_0(\mathbf{x}|r;E) \;=\; \log\left(\max_{r_0\in R}|X_{r_0}|\right) \;\leq\; \log\left(\max_{r_0\in R}\left|\cup_{j=1}^{L}\{0,1\}^j\right|\right) \;\leq\; L+1.$$

□

A.2 Type-Safe Algorithms

Proof of Lemma 5. First note that $\mathsf{Rel}_{\mathcal{M}}(q,\mathfrak{a},x;r)$ has the same behavior as $\mathsf{Exec}_{\mathcal{M}}(q,\mathfrak{a},x;r)$ on each query that $\mathcal{M}$ makes to its oracles add and eq. This fact is obvious for add queries, but it also holds for eq queries because at any point in time, each group element $\mathfrak{a}_i$ in Exec (for $i \in \{1,\ldots,n\}$) is identical to $\mathbf{v}_i^T \cdot \mathfrak{a}$ in Rel (where $\mathfrak{a}$ denotes the unmodified input to Rel).

Hence, it only remains to prove that $\mathsf{Sim}_{\mathcal{M}}(q,R,x;r)$ has the exact same behavior as $\mathsf{Rel}_{\mathcal{M}}(q,\mathfrak{a},x;r)$ on each query that $\mathcal{M}$ makes to its oracle eq. Let us consider any such query $\mathsf{eq}(i,j)$ made by $\mathcal{M}$ for some $i,j \in \{1,\ldots,n\}$, assuming that all previous queries have been answered identically by both Sim and Rel. The current state of n and all $\mathbf{v}_i$'s (for $i \in \{1,\ldots,n\}$) are identical in both executions.

- Suppose that Rel replies with 1, i.e., $\mathbf{v}_i^T \cdot \mathfrak{a} = \mathbf{v}_j^T \cdot \mathfrak{a}$. Then, by definition of Rel, we have that $\mathbf{v}_i - \mathbf{v}_j \in R' \subseteq R$, hence Sim also replies with 1.
- Suppose that Sim replies with 1. Then $\mathfrak{a} \in \mathrm{Ann}_{\mathbb{G}}(R)$ implies $(\mathbf{v}_i - \mathbf{v}_j)^\top \mathfrak{a} = \mathfrak{o}$. Thus Rel also returns 1.

Finally, to show the last implication it suffices to observe that both requirements are fulfilled: $R' \subseteq R$ due to $R' = R$, and $\mathfrak{a} \in \mathrm{Ann}_{\mathbb{G}}(R')$ due to Lemma 4, which implies that $\mathbf{v}^T\mathfrak{a} = \mathfrak{o}$ is fulfilled for *all* vectors $\mathbf{v} \in R'$. □

Proof of Lemma 6. Let $(i^{(k)}, j^{(k)})$ be the k-th query to eq made by $\mathcal{M}$ in $\mathsf{Sim}_{\mathcal{M}}(q,R,x;r)$, for $k \in \{1,\ldots,\chi\}$, and let $\mathbf{v}^{(k)} := \mathbf{v}_{i^{(k)}} - \mathbf{v}_{j^{(k)}}$. Further, let $\mathrm{res}_k := (\mathbf{v}^{(k)} \in R)$ be the simulator's reply to the k-th query, and let $\mathrm{res}'_k := ((\mathbf{v}^{(k)})^T \cdot (\mathfrak{a}||\mathfrak{b}) = \mathfrak{o})$ be the reply that Exec would give to the same query (under the assumption that the first $k-1$ queries to eq have been answered identically by Sim and Exec).

We will prove, for all $k \in \{1,\ldots,\chi\}$:

$$\Pr_{\mathfrak{b}}\left[\mathrm{res}_k < \mathrm{res}'_k \;\wedge\; \pi_n^*(R + \mathsf{span}(\mathbf{v}^{(k)})) \subseteq \pi_n^*(R)\right] \leq \frac{1}{q}\,, \tag{7}$$

i.e., it is unlikely that Sim answers 0 to the k-th query while Exec answers 1, unless a new relation about the first n coordinates is learned.

From this, the lemma follows by a union bound over all k, because if the event in (7) is false for all k, then we have:

- Either $\mathrm{res}_k \geq \mathrm{res}'_k$ for all k. Since $\mathrm{res}_k > \mathrm{res}'_k$ is impossible (due to $(\mathbf{v}^{(k)})^T \cdot (\mathfrak{a}||\mathfrak{b}) = \mathfrak{o}$ by definition of $\mathfrak{b}$ and res_k). Thus, $\mathrm{res}_k = \mathrm{res}'_k$ for all k, i.e., the simulated execution and the "real" execution are identical: $\mathsf{Sim}_{\mathcal{M}}(q,R,x;r) = \mathsf{Exec}_{\mathcal{M}}(q,(\mathfrak{a}||\mathfrak{b}),x;r)$.

– Or there exists some k for which $\mathrm{res}_k < \mathrm{res}'_k$ and $\pi_n^*(R+\mathsf{span}(\mathbf{v}^{(k)})) \not\subseteq \pi_n^*(R)$ while $\mathrm{res}_{k'} = \mathrm{res}'_{k'}$ for all $k' < k$. The latter (i.e., the first $k-1$ queries being answered identically by Sim and Rel) together with $\mathrm{res}'_k = 1$ means that Rel adds the query's relation $\mathbf{v}^{(k)}$ to its output, i.e., $\mathbf{v}^{(k)} \in \mathsf{Rel}_{\mathcal{M}}(q, (\mathfrak{a}||\mathfrak{b}), x; r)$. Hence, $\pi_n^*(R + \mathsf{Rel}_{\mathcal{M}}(q, (\mathfrak{a}||\mathfrak{b}), x; r)) \not\subseteq \pi_n^*(R)$.

Now we prove (7). First note that the probability is trivially 0 whenever $\pi_n^*(R + \mathsf{span}(\mathbf{v}^{(k)})) \not\subseteq \pi_n^*(R)$ (which does not depend on the random choice of $\mathfrak{b}$). Hence, what we will actually show is that for any vector space $R \leq \mathbb{F}_q^{n+m}$, for any vector $\mathbf{v} \notin R$ with $\pi_n^*(R+\mathsf{span}(\mathbf{v})) \subseteq \pi_n^*(R)$, and for any group elements $\mathfrak{a} \in \mathrm{Ann}_{\mathbb{G}}(\pi_n^*(R))$, we have the following, with $\mathfrak{b}$ uniformly distributed over $\{\mathfrak{b} \in \mathbb{G}^m \mid (\mathfrak{a}||\mathfrak{b}) \in \mathrm{Ann}_{\mathbb{G}}(R)\}$:

$$\Pr_{\mathfrak{b}}[\mathbf{v}^T \cdot (\mathfrak{a}||\mathfrak{b}) = \mathfrak{o}] = \frac{1}{q}$$

We define the two vector spaces

$$H := \pi_{-m}(R) \quad \text{and}$$
$$H' := \pi_{-m}(R + \mathsf{span}(\mathbf{v}))$$

as the projections on the last m components of R and $R+\mathsf{span}(\mathbf{v})$, respectively. Then, the two sets

$$B := \{\mathfrak{b} \in \mathbb{G}^m \mid (\mathfrak{a}||\mathfrak{b}) \in \mathrm{Ann}_{\mathbb{G}}(R)\} \quad \text{and}$$
$$B' := \{\mathfrak{b} \in \mathbb{G}^m \mid (\mathfrak{a}||\mathfrak{b}) \in \mathrm{Ann}_{\mathbb{G}}(R + \mathsf{span}(\mathbf{v}))\}$$

(i.e., the set of choices that $\mathfrak{b}$ is sampled from, and the subset containing only those choices for which the event in the probability above is fulfilled) are cosets of $\mathrm{Ann}_{\mathbb{G}}(H)$ and $\mathrm{Ann}_{\mathbb{G}}(H')$, respectively. (This can be shown by observing that for any $\mathfrak{b}_1, \mathfrak{b}_2 \in B$, we have $\mathfrak{b}_1 - \mathfrak{b}_2 \in \mathrm{Ann}_{\mathbb{G}}(H)$, and that for any $\mathfrak{b} \in B$ and any $\mathfrak{b}' \in \mathrm{Ann}_{\mathbb{G}}(H)$, we have $\mathfrak{b} + \mathfrak{b}' \in B$. The same holds for B' and $\mathrm{Ann}_{\mathbb{G}}(H')$.)

We will now show $H \subsetneq H'$, which implies

$$|B| = |\mathrm{Ann}_{\mathbb{G}}(H')| = \frac{|\mathrm{Ann}_{\mathbb{G}}(H)|}{q} = \frac{|B'|}{q} \ ,$$

hence concluding the proof. Clearly, $H \subseteq H'$. Hence, it suffices to prove $H \subsetneq H'$. Suppose $H = H'$. Then, using $\mathbf{v}_r \in \mathbb{F}_q^m$ to denote the last m components of $\mathbf{v}$, we have $\mathbf{v}_r \in \pi_{-m}(R)$ due to $\mathbf{v} \in R + \mathsf{span}(\mathbf{v})$. This implies existence of some $\mathbf{w} := (\mathbf{w}_l||\mathbf{v}_r) \in R$. Due to the last m components in $\mathbf{v} - \mathbf{w}$ being 0, we have

$$\mathbf{v} - \mathbf{w} \in \pi_n^*(R + \mathsf{span}(\mathbf{v} - \mathbf{w})) = \pi_n^*(R + \mathsf{span}(\mathbf{v})) = \pi_n^*(R) \subseteq R \ ,$$

where, the first equality holds due to $\mathbf{w} \in R$. Again due to $\mathbf{w}$, the inclusion above contradicts $\mathbf{v} \notin R$, i.e., the assumption $H = H'$ is false. □

Proof of Lemma 7. Every potential output of the first distribution is clearly also a potential output of the second distribution. The converse holds as well, because

$(\mathfrak{b}_0||\ldots||\mathfrak{b}_\ell) \in \mathrm{Ann}_{\mathbb{G}}(R_\ell)$ implies $(\mathfrak{b}_0||\ldots||\mathfrak{b}_{i-1}||\mathfrak{b}_i) \in \mathrm{Ann}_{\mathbb{G}}(R_i)$ for any $i < \ell$ due to $\pi^*_{n+\sum_{j=1}^{i-1} m_j}(R_\ell) = R_{i-1}$. In the second distribution, each potential output has identical probability due to uniform sampling. In the first distribution, the same is true: in each iteration i, the set of valid choices for $\mathfrak{b}_i$ is a coset of $\mathrm{Ann}_{\mathbb{G}}(\pi_{-m_i}(R_i))$ (as in the proof of Lemma 6), whose size is independent of the choices made in previous iterations. □

Proof of Lemma 8. We use the following procedure for sampling $\mathfrak{b}$:

1. Choose arbitrary matrices $A \in \mathbb{F}_q^{\dim R \times n}$, $B \in \mathbb{F}_q^{\dim R \times m}$ s.t. $R = \mathsf{span}\begin{pmatrix} A^T \\ B^T \end{pmatrix}$.
2. Sample a uniformly random vector $\mathbf{z} \leftarrow^{\$} \mathrm{Ann}_{\mathbb{F}_q^m}(\pi_{-m}(R))$ (as defined in the proof of Lemma 6).
3. Return $\mathfrak{b} := -B^+A \cdot \mathfrak{a} + \mathbf{z}^T \cdot \mathfrak{g}$, where B^+ denotes the Moore-Penrose pseudoinverse of B (which always exists and is efficiently computable).

Since the set $B := \{\mathfrak{b} \in \mathbb{G}^m \mid (\mathfrak{a}||\mathfrak{b}) \in \mathrm{Ann}_{\mathbb{G}}(R)\}$ of valid choices for $\mathfrak{b}$ is a coset of $\mathrm{Ann}_{\mathbb{G}}(\pi_{-m}(R))$ (as in the proof of Lemma 6), it only remains to prove that $-B^+A \cdot \mathfrak{a} \in B$, i.e., only valid choices are generated by this procedure. For every $\mathbf{u} \in R$, we want to prove $\mathbf{u}^T \cdot (\mathfrak{a}|| - B^+A \cdot \mathfrak{a}) = \mathfrak{o}$, i.e., when writing $\mathbf{u}$ as $\mathbf{u} = (\mathbf{u}_l||\mathbf{u}_r)$, we need $(\mathbf{u}_l^T - \mathbf{u}_r^T B^+A) \cdot \mathfrak{a} = \mathfrak{o}$. To do so, it suffices to show $(\mathbf{u}_l - A^T(B^+)^T\mathbf{u}_r||0^m) \in R$, because $\mathfrak{a} \in \mathrm{Ann}_{\mathbb{G}}(\pi_n^*(R))$ by assumption. Due to $(\mathbf{u}_l||\mathbf{u}_r) = \mathbf{u} \in R$, this will be implied by $(A^T(B^+)^T\mathbf{u}_r||\mathbf{u}_r) \in R$.

By definition of the pseudo-inverse, we have $BB^+B = B$. Since $\mathbf{u} \in R$, there exists some $\mathbf{w} \in \mathbb{F}_q^{\dim R}$ s.t. $\mathbf{u} = \begin{pmatrix} A^T \\ B^T \end{pmatrix} \cdot \mathbf{w}$, i.e., $\mathbf{u}_r = B^T \cdot \mathbf{w}$. Multiplying $\mathbf{w}$ to the previous equation (transposed) hence yields $B^T(B^+)^T\mathbf{u}_r = \mathbf{u}_r$. Therefore,

$$\begin{aligned}(A^T(B^+)^T\mathbf{u}_r||\mathbf{u}_r) &= (A^T(B^+)^T\mathbf{u}_r||B^T(B^+)^T\mathbf{u}_r) \\ &= \begin{pmatrix} A^T \\ B^T \end{pmatrix} \cdot (B^+)^T\mathbf{u}_r \in R\,,\end{aligned}$$

where the inclusion in R follows by definition of matrices A and B. □

A.3 Basic Decoder

Proof of Lemma 14. We prove each property separately.

First Statement. We proceed by induction on i. By construction, $L_0^{(1)} = L_0$ which satisfies $\mathfrak{p}||\mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L_0)$. Fix $i \in [n']$ and assume $\mathfrak{p}||\mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L_0^{(i)})$. The next space is defined either in Line 9, in Line 11, or in Line 15.

- If Line 9 is executed, then $L_0^{(i+1)} = L_0^{(i)}$ and the statement follows immediately.
- If Line 11 is executed, then $\mathfrak{p}||\mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L_0^{(i+1)})$ follows from Lemma 13,
- If Line 15 is executed, then $\mathfrak{p}||\mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L_0^{(i+1)})$ follows from Lemma 11.

Second statement. First, note that `init` is always queried once, costing ξ_{cm} bits.

`bit` calls occurs in the For loop either in Line 12 or 14. In the first case, since Forge returns $L_0^{(i+1)}$ then by Lemma 13 we know $L_0^{(i)} \subsetneq L_0^{(i+1)}$ and therefore $\dim L_0^{(i+1)} > \dim L_0^{(i)}$. In the second case, $\mathsf{Extract}$ returned some $L_0^{(i+1)} \supseteq L_0^{(i)}$ by Lemma 11. If equality $L_0^{(i+1)} = L_0^{(i)}$ holds, then the decoder aborts the current iteration by returning $\perp$. Thus, under the assumption this iteration does not abort, we again have $\dim L_0^{(i+1)} > \dim L_0^{(i)}$. By induction, at most $\dim L_0' - \dim L_0$ times a query `bit` is made, each costing one bit.

`wit` calls occur only in Line 16. The same argument as above shows that $\mathsf{Extract}$ is invoked at most $\dim L_0' - \dim L_0$ times (unless the decoder aborts). Observe that by construction, $\mathsf{Extract}$ performs exactly $\prod_{i=1}^{j-1}(\mu_i+1) \leq \prod_{i=1}^{\rho-1}(\mu_i+1)$ calls of the form $\mathcal{W}(s, c_1, \ldots, c_{j-1})$ for each $j \in [\rho]$. Each corresponds to one `wit` message, which involves a $|w_i| \leq \xi_{\mathsf{wit}}$ bits long reply. Thus each $\mathsf{Extract}$ execution entails $\xi_{\mathsf{wit}} \cdot \prod_{i=1}^{\rho-1}(\mu_i + 1)$ bits.

Summing the three terms up concludes the proof. □

A.4 Full Compression Argument

Proof of Lemma 16. The first part follows by induction over i: clearly, $\mathfrak{p} \in \mathrm{Ann}_{\mathbb{G}}(\{0^{\vartheta_0}\}) = \mathrm{Ann}_{\mathbb{G}}(P^{(1)})$. Given i for which $\mathfrak{p} \in \mathrm{Ann}_{\mathbb{G}}(P^{(i)})$, $P^{(i+1)}$ is computed either in Line 8 or in Line 9. In the first case, $P^{(i+1)} = P^{(i)}$ and the thesis follows immediately. In the second case, $P^{(i+1)} = \pi^*_{\vartheta_0}(L^{(i)})$ for some $L^{(i)}$ that fulfills $\mathfrak{p}||\mathfrak{c} \in \mathrm{Ann}_{\mathbb{G}}(L^{(i)})$ (by Lemma 14), and thus $\mathfrak{p} \in \mathrm{Ann}_{\mathbb{G}}(P^{(i+1)})$. We also have $P^{(i)} \subseteq P^{(i+1)}$ as either $P^{(i+1)} = P^{(i)}$ or $P^{(i+1)} = \pi^*_{\vartheta_0}(L^{(i)})$ with $L^{(i)} \supseteq P^{(i)} \times \{0^{\vartheta_1}\}$ (again by Lemma 14).

For the second part, observe that all the queries that $\mathcal{D}$ makes to $\mathcal{E}(\mathbf{m}, r, \cdot)$ occur within the invocation of $\mathcal{D}_0$ in Line 11 (in the exploration phase in Lines 2-9, only $\mathcal{E}_0(\mathbf{m}^{(i)}, \mathsf{pp}, \mathsf{aux}^{(i)}, r_{\mathsf{wit}}, \cdot)$ is queried which is simulated by $\overline{\mathcal{D}}$ itself). As we assume that $\mathcal{D}$ does not abort, $\mathcal{D}_0^{\mathcal{C},\mathcal{E}_0(\mathbf{m},\mathsf{pp},\mathsf{cm},\mathsf{aux},r_{\mathsf{wit}},\cdot)}(L_0^{(i^*)}, \mathsf{pp}; r_0^{(i^*)})$ returns some $(\bar{\mathbf{m}}, L^{(i^*)}) \neq \perp$ (i.e. it does not abort either). By Lemma 14 we can thus derive that, denoting $\mathcal{C} := \mathcal{C}_{\mathsf{rand}}^{\mathsf{Forge}}(\cdot; r_{\mathcal{C}})$, we have

$$\begin{aligned}
&\mathsf{len}_{\mathcal{E},\mathcal{D}}((\mathbf{m}, r), r) \;=\; \mathsf{len}_{\mathcal{E}_0,\mathcal{D}_0^{\mathcal{C}}}((\mathbf{m}, \mathsf{pp}, \mathsf{cm}, \mathsf{aux}, r_{\mathsf{wit}}), (L_0^{(i^*)}, \mathsf{pp}; r_0^{(i^*)})) \\
&\leq\; \xi_{\mathsf{cm}} + (\dim(L^{(i^*)}) - \dim(P^{(i^*)} \times \{0^{\vartheta_1}\})) \cdot \left(1 + \xi_{\mathsf{wit}} \cdot \prod\nolimits_{i=1}^{\rho-1}(\mu_i + 1)\right).
\end{aligned}$$

Since $\mathcal{D}$ does not abort, the check in line 14 is not satisfied, i.e., $\pi^*_{\vartheta_0}(L^{(i^*)}) \subseteq P^{(i^*)}$ and in particular $\vartheta_1 \geq \dim(L^{(i^*)}) - \dim(P^{(i^*)}) = \dim(L^{(i^*)}) - \dim(P^{(i^*)} \times \{0^{\vartheta_1}\})$. The thesis follows. □