The Structure of Witness-Update Tradeoffs in Additive Positive Accumulators

Wei Qi
Bocconi University
Department of Computing Sciences

November 24, 2026

Abstract

An additive positive accumulator represents a growing set using a short public state, while maintaining membership witnesses may require costly updates as new elements are inserted. Previous work has studied tradeoffs involving the maximum and total numbers of witness updates. We establish a general characterization of the tradeoff between accumulator length and the entire pattern of witness updates, obtaining matching lower and upper bounds at the bit level.

We represent witness updates by a forward graph G and let $\alpha(G)$ denote the maximum length of a skipping sequence. We prove that, for any sound accumulator with public states of at most B bits, the probability that $\alpha(G) \geq m$ is negligible whenever $(B+1)/m = O(\log \lambda)$. This lower bound holds even when the update pattern depends arbitrarily on the execution and under a weak soundness notion in which the adversary receives no genuine membership witnesses. Conversely, assuming collision-resistant hash functions with σ-bit outputs, we construct a sound accumulator that realizes any polynomial-time computable update graph G exactly, using at most $\sigma\alpha(G)$ bits.

These results yield a bit-level characterization for arbitrary nonnegative update-cost functions. For a cost function C and a threshold c, define $R_C(c) := \min_{G:C(G)\leq c} \alpha(G)$. In the nontrivial regime $2 \leq R_C(c) \leq n$, achieving cost at most c with overwhelming probability requires $R_C(c) \log \lambda = o(B)$. Conversely, under the same condition, a polynomial-time computable optimal graph and an appropriate collision-resistant hash function suffice to achieve cost at most c with accumulator length at most B.

We apply this characterization to derive exact combinatorial optima for the maximum and total numbers of witness updates, attained simultaneously by a single efficiently constructible graph, and obtain concrete tradeoffs for polynomial and polylogarithmic accumulator lengths. For heterogeneous update costs, we give exact polynomial-time algorithms for insertion-time-dependent costs and nondecreasing time-dependent costs, together with a polynomial-time w-approximation for arbitrary time-dependent costs.

1 Introduction

An additive positive accumulator [2, 5, 10] represents a growing set of elements $S_t = \{x_1, \ldots, x_t\}$ by a short public state acc_t of length at most B, while each element $x_i \in S_t$ has a membership witness $w_{i,t}$ that can be verified against the current state. When a new element x_{t+1} is inserted, the public state changes from acc_t to acc_{t+1}, and previously issued witnesses $w_{i,t}$ may need to be updated to remain valid. Such updates can be expensive: a witness holder may have to obtain new information or perform additional computation whenever its witness changes. This creates a fundamental tradeoff between the accumulator-length bound B and the cost of maintaining membership witnesses.

The tradeoff is particularly relevant when witnesses are maintained by different users. Updating every witness after every insertion is often unnecessary, but the cost of an update may vary

substantially across witnesses or over time. A user who rarely connects to the system may incur a higher communication cost than one who is continuously online; similarly, an update performed at one round may be more expensive than the same update performed at another. Consequently, the maximum number of updates per witness and the total number of updates do not necessarily capture the cost of a particular maintenance policy. To understand such costs, we need to study not only how many updates occur, but also *which witnesses are updated and when.*

Prior work and our question. Previous research has studied the frequency of witness updates through online-merger constructions [8] and lower bounds on maximum and total update counts [3, 10]. We ask a more general question: *what restrictions does cryptographic security impose on the entire pattern of witness updates, and can those restrictions be matched by a construction?*

A structural characterization. Consider the first n insertions and represent their witness updates by a forward graph G on $[n]$: an edge (i, t) indicates that the witness for the element inserted at round i is updated at round $t > i$. Following earlier work [9, 10], an increasing sequence $s_1 < \cdots < s_m$ is a *skipping sequence* if, for every $j < m$, the witness for the element inserted at round s_j is not updated during rounds $s_{j+1}, \ldots, s_m$. Write $\alpha(G)$ for the maximum length of such a sequence. The following theorem states both the security restriction on this parameter and a construction achieving the corresponding graph width.

Theorem 1.1 (Security and exact graph realization)**.** *Let $n = n(\lambda)$ be efficiently computable and polynomially bounded.*

Security. *Let G be the update graph of a sound additive positive accumulator, complete with respect to its update specification, with accumulator-length bound B. Under uniform distinct insertions and the setup and addition randomness, every $2 \le m = m(\lambda) \le n$ with $(B+1)/m = O(\log \lambda)$ satisfies*

$$\Pr[\alpha(G) \ge m] = \mathrm{negl}(\lambda).$$

This holds even if the update specification depends arbitrarily on the execution.

Realization. *For any polynomial-time computable forward graph $G = G_\lambda$ on $[n]$ and collision-resistant hash family with σ-bit outputs, there is a sound accumulator, perfectly complete for the prescribed updates, whose update graph during the first n insertions is exactly G and whose public states satisfy*

$$|\mathsf{acc}_t| \le \sigma\alpha(G) \qquad \text{for every } t \in [n].$$

The first direction holds under a weak notion of soundness in which the adversary observes public accumulator states but not genuine membership witnesses. The second direction incurs no additional witness updates beyond those specified by G. Thus $\alpha(G)$ provides a common combinatorial measure for the lower bound and the matching construction, independently of how update costs are assigned.

For any nonnegative update-cost function C on forward graphs and a feasible cost threshold c, define

$$R_C(c) := \min_{G : C(G) \le c} \alpha(G).$$

The next theorem expresses the resulting tradeoff directly in terms of accumulator length and update cost.

Theorem 1.2 (Tradeoff for arbitrary update costs)**.** *Let $n = n(\lambda)$ be efficiently computable and polynomially bounded, let $B = B(\lambda)$ be a positive, efficiently computable, polynomially bounded integer, and let C be any nonnegative update-cost function with a feasible threshold c. Write $r := R_C(c)$ and suppose $2 \le r \le n$. If a sound additive positive accumulator, complete with respect to its update specification, has accumulator-length bound at most B and achieves $C(G) \le c$ with*

overwhelming probability under uniform distinct insertions and the setup and addition randomness, then

$$r \log \lambda = o(B).$$

Conversely, if $r \log \lambda = o(B)$, a graph attaining $R_C(c)$ is polynomial-time computable, and a collision-resistant hash family with output length $\lfloor B/r \rfloor$ is available, then a sound accumulator, perfectly complete for the prescribed updates, achieves length at most B and cost at most c.

This reduces the cryptographic tradeoff for any such cost function to optimizing $C(G)$ subject to a bound on $\alpha(G)$. We next make this optimization explicit for update counts and give efficient algorithms for heterogeneous costs.

Concrete tradeoffs for update counts. We begin with the maximum and total numbers of witness updates, which have been studied in prior work. For a forward graph G, write $d_G(i) := |\{t > i : (i,t) \in E(G)\}|$ for the number of updates to witness i.

Theorem 1.3 (Exact maximum- and total-update optima)**.** *For all integers $n, w \geq 1$, define*

$$q(n,w) := \max\left\{d \geq 0 : \binom{w+d}{d} \leq n\right\}, \qquad \Phi_w(n) := \sum_{d \geq 1} \left(n + 1 - \binom{w+d}{d}\right)_+.$$

Then

$$\min_{G:\alpha(G)\leq w} \max_{i \in [n]} d_G(i) = q(n,w), \qquad \min_{G:\alpha(G)\leq w} |E(G)| = \Phi_w(n).$$

Moreover, a single polynomial-time constructible forward graph attains both minima.

These formulas give concrete lower and upper bounds under polynomial and polylogarithmic accumulator-length bounds. Some bounds match using ordinary collision-resistant hash functions, while others require an additional assumption on the availability of shorter hash outputs.

Heterogeneous update costs. We also allow different witnesses, or different update times, to incur different nonnegative costs. The following theorem summarizes which of these optimization problems admit efficient exact solutions and what can be achieved without restrictions on time-dependent weights.

Theorem 1.4 (Optimization under heterogeneous costs)**.** *Let $1 \leq w \leq n$, and suppose the nonnegative weights are rational numbers of polynomial bit length.*

If updating witness i costs a_i at every round, then $\min_{\alpha(G)\leq w} \sum_{i=1}^{n} a_i d_G(i)$ and an optimal graph can be computed exactly in polynomial time.

If updating witness i at round t costs $a_{i,t}$, then a polynomial-time algorithm constructs a graph G with $\alpha(G) \leq w$ and

$$\sum_{(i,t)\in E(G)} a_{i,t} \leq w \min_{H:\alpha(H)\leq w} \sum_{(i,t)\in E(H)} a_{i,t}.$$

If, in addition, $a_{i,t}$ is nondecreasing in t for every fixed i, the exact optimum and an optimal graph can be computed in polynomial time.

Our exact optimization results use a correspondence between feasible update graphs and ordered binary trees, while the approximation algorithm for arbitrary time-dependent costs uses linear programming and interval covering.

1.1 Related Work

Accumulators and online-merger constructions. Cryptographic accumulators were introduced by Benaloh and de Mare [2], and dynamic constructions have been developed using different algebraic assumptions [5, 4]. For append-only accumulators, Merkle-based schemes can avoid updating all witnesses after each insertion [8, 3]. Mahmoody and Qi [8] introduced online mergers and established the optimal maximum merge depth for a given width, with accumulator constructions based on the resulting merger schedules. These guarantees concern width and depth, but do not prescribe the individual update edges (i, t). In contrast, our prefix-hash construction realizes every polynomial-time computable forward graph G with exactly its prescribed witness updates and accumulator length at most $\sigma\alpha(G)$, allowing costs to be optimized for individual update pairs.

Lower bounds on update frequency. Mahmoody, Qi, and Rahimi [9] used skipping sequences to derive decryption-update lower bounds for registration-based encryption under restrictions on the update schedule. Bonneau, Chen, Christ, and Karantaidou [3] established lower bounds on the total number of witness updates in succinct accumulators. Qi [10] removed the fixed-update-time restriction for additive positive accumulators, using falling-step sequences and prefix-consistent families of executions to establish a tight tradeoff between accumulator length and the *maximum* number of updates to an individual witness. Although this result allows execution-dependent update times, its theorem assumes a uniform bound on maximum update counts and does not directly bound $\alpha(G)$ for arbitrary realized graphs. It therefore does not immediately imply lower bounds for costs that depend on individual update pairs.

Our proof instead works with a skipping sequence selected from the entire random execution. Rather than extracting prefix-consistent families as in [10], we resample one of the sequence's disjoint intervals. The mutual-information chain rule amortizes the information revealed by the common accumulator state across these intervals, while the KL chain rule accounts for the distributional bias caused by conditioning on the selected sequence. A KL bound on event probabilities transfers verification to fresh replay without requiring statistical closeness; the adversary need only guess an interval and a witness, not find the skipping sequence. The resulting graph-level lower bound, together with exact graph realization, extends the earlier cost-specific tradeoffs to arbitrary nonnegative update-cost functions.

Other aspects of witness maintenance. Abusalah, Anthoine, Avitabile, and Giunta [1] proved lower bounds on the expected number of membership proofs that must be updated after an insertion for short accumulators and updatable vector commitments, with particularly strong bounds when the digest length approaches the concrete security level. This concerns the number of affected proofs per operation rather than the complete update pattern. For vector commitments, Tas and Boneh [12] studied the tradeoff between the size of globally distributed update information and the computation required to update an individual proof. Other work addresses privacy-preserving delegation or outsourcing of witness updates [11, 7]. These works study different aspects of witness maintenance; they do not optimize arbitrary cost functions on update graphs or heterogeneous costs that depend on which witness is updated and when.

2 Technical Overview

We represent witness updates during the first n insertions by a forward graph G on $[n]$, with an edge (i, t) whenever the witness for the value inserted at round i is updated at round t. A sequence $s_1 < \cdots < s_m$ is a *skipping sequence* if, for every $j < m$, the witness for x_{s_j} is not updated during rounds $s_{j+1}, \ldots, s_m$. Let $\alpha(G)$ be the maximum length of such a sequence. Our central observation is that $\alpha(G)$ governs both the accumulator length required by a secure scheme and the

accumulator length sufficient to realize a prescribed update graph. This reduces witness-update tradeoffs for general cost functions to optimization over forward graphs with bounded $\alpha(G)$.

Lower bounds from interval resampling. Suppose the update graph of a random execution contains a skipping sequence $s_1 < \cdots < s_m$, and write $v = s_m$. The sequence partitions the rounds from s_1 to v into disjoint intervals $I_j = [s_j, s_{j+1} - 1]$, where $s_{m+1} = v + 1$. Since the witness for x_{s_j} receives no updates after I_j and before round v, one of the witnesses issued during I_j must remain valid against the common accumulator state acc_v. Thus each interval contains a witness that can be tested against the same public state.

To exploit this property, we select one of the m intervals uniformly and replay it from the original execution prefix, but with independently sampled insertions and addition randomness. We keep the original endpoint state $Y = \mathsf{acc}_v$ unchanged. In the genuine replay, completeness guarantees that at least one candidate witness verifies against Y. The challenge is to show that verification remains sufficiently likely after resampling, even though the skipping sequence and its intervals may depend on the entire execution.

Let $q = \Pr[\alpha(G) \geq m]$, and condition on this event and on a selected skipping sequence. The KL divergence between the genuine and resampled distributions has two contributions: the information about the selected interval revealed by Y, and the change in the interval distribution caused by conditioning on the skipping sequence. Because the intervals are disjoint and consecutive, the mutual-information chain rule bounds the sum of the first contributions by $H(Y) \leq (\ell(\lambda, n)+1) \ln 2$. The KL chain rule bounds the second contributions by the divergence incurred by conditioning on the selected sequence. Averaging over the m intervals gives

$$D_{\mathrm{KL}}(P_B \| Q_B) \leq \frac{(\ell(\lambda, n) + 1) \ln 2 + \ln(1/q)}{m} + \ln n.$$

A KL bound on event probabilities then transfers the verification guarantee from genuine replay to fresh replay. Finally, a polynomial-time soundness adversary can guess the interval, its common endpoint, and one candidate witness, losing only polynomial factors; it need not compute the skipping sequence or receive genuine witnesses. Consequently, for sound and complete accumulators, $\Pr[\alpha(G) \geq m]$ is negligible whenever $(\ell(\lambda, n) + 1)/m = O(\log \lambda)$. This conclusion holds even when the update specification depends arbitrarily on the execution.

Matching construction through prefix hashes. For the converse, fix any forward graph G. At round t, let $r_i(t)$ be the most recent round at which the witness for x_i was issued or updated. Our construction makes this retained witness contain the insertion prefix through round $r_i(t)$, so it can be verified using the hash of any prefix indexed by $s \in [i, r_i(t)]$. It therefore suffices to publish hashes for a set of indices P_t intersecting every interval $[i, r_i(t)]$, for $i \leq t$.

The minimum number of points needed to intersect a family of intervals equals the maximum number of pairwise disjoint intervals. Here, disjoint intervals at round t correspond to a skipping sequence, and a maximum skipping sequence produces disjoint intervals at its last round. Hence the minimum number $\tau_t(G)$ of published prefix hashes satisfies $\max_t \tau_t(G) = \alpha(G)$. Choosing a minimum interval-hitting set at each round yields an accumulator with $|\mathsf{acc}_t| \leq \sigma\alpha(G)$, where σ is the output length of the collision-resistant hash function. The update specification follows the edges of G exactly; completeness follows from the hitting-set property, and soundness reduces to collision resistance. Thus every efficiently computable graph can be realized without introducing additional witness updates.

Together, the two directions characterize the tradeoff in terms of the combinatorial quantity $R_C(c) := \min_{G:C(G)\leq c} \alpha(G)$ for any nonnegative update-cost function C. If a secure accumulator of length at most B achieves cost at most c with overwhelming probability, then $R_C(c) \log \lambda = o(B)$ (for the nontrivial range $2 \leq R_C(c) \leq n$). Conversely, a graph attaining $R_C(c)$ can be realized within length B when a collision-resistant hash function with output length $\lfloor B/R_C(c) \rfloor$ is available,

subject to efficient computability of the graph. The remaining question is therefore how to optimize update costs under a bound on $\alpha(G)$.

Optimizing update graphs. For maximum and total update counts, we relate feasible graphs to ordered binary trees. Given a graph G with $\alpha(G) \le w$, a recursive decomposition produces an ordered tree with fewer than w right edges on each root-to-node path and with at most $d_G(i)$ left edges on the path to vertex i. Conversely, such a tree yields a feasible graph by adding an edge from each vertex to every ancestor whose left subtree contains it. Counting binary words with bounded numbers of left and right moves gives the exact optima

$$\min_{\alpha(G)\le w} \max_i d_G(i) = q(n,w), \qquad \min_{\alpha(G)\le w} |E(G)| = \Phi_w(n),$$

where $q(n,w) = \max\{d \ge 0 : \binom{w+d}{d} \le n\}$ and $\Phi_w(n) = \sum_{d\ge 1}(n+1-\binom{w+d}{d})_+$. A single polynomial-time constructible tree realizes both minima, allowing us to derive concrete accumulator-length tradeoffs for maximum and total updates.

For heterogeneous update costs, the same tree representation gives an exact polynomial-time dynamic program when each update to witness i has a fixed nonnegative cost a_i. When the cost $a_{i,t}$ also depends on the update round, feasibility can be expressed as an integer program requiring every increasing $(w+1)$-tuple to be blocked. Its LP relaxation admits polynomial-time separation, and a row-by-row minimum-cost interval-hitting procedure rounds a fractional solution to a feasible graph at most w times the optimal cost. If the costs for each witness are nondecreasing over time, the tree reduction again preserves optimality and yields an exact polynomial-time dynamic program. These algorithms extend the tradeoff analysis from update counts to general communication costs.

3 Preliminaries

Throughout, $\lambda \in \mathbb{N}$ denotes the security parameter. All asymptotic notation, including negligible probabilities and $o(1)$ terms, is with respect to λ unless stated otherwise. A function $\nu : \mathbb{N} \to \mathbb{R}_{\ge 0}$ is *negligible* if for every constant $c > 0$,

$$\nu(\lambda) < \lambda^{-c}$$

for all sufficiently large λ.

In purely combinatorial statements, n is an arbitrary positive integer. In cryptographic experiments and asymptotic statements, $n = n(\lambda)$ is polynomially bounded unless stated otherwise.

We write

$$[n] := \{1, \dots, n\}, \qquad (z)_+ := \max\{z, 0\}.$$

For each security parameter λ, let $\mathcal{X}_\lambda$ be a finite universe that can be sampled uniformly in probabilistic polynomial time and satisfies

$$|\mathcal{X}_\lambda| \ge 2^{\Omega(\lambda)}.$$

When λ is clear, we write simply $\mathcal{X}$. For $n \ge 0$, let

$$\mathsf{List}_n(\mathcal{X})$$

denote the set of ordered lists of n distinct elements of $\mathcal{X}$, and write

$$\mathsf{List}_{\le q}(\mathcal{X}) := \bigcup_{j=0}^{q} \mathsf{List}_j(\mathcal{X}).$$

All randomized algorithms are viewed as taking an explicit random tape.

3.1 Additive Positive Accumulators

Definition 3.1 (Additive positive accumulator)**.** An *additive positive accumulator* [10] is a tuple of PPT algorithms $\Lambda = (\mathsf{SetUp}, \mathsf{Add}, \mathsf{Ver})$ with the following syntax:

- $\mathsf{SetUp}(1^\lambda) \to \mathsf{crs}$: On input the security parameter 1^λ, the setup algorithm generates a common reference string crs.
- $\mathsf{Add}(\mathsf{crs}, S, \mathsf{acc}, x; r) \to (\mathsf{acc}', W)$: On input the current ordered list $S \in \mathsf{List}_{t-1}(\mathcal{X}_\lambda)$, the current accumulator state acc, and a new element $x \in \mathcal{X}_\lambda \setminus S$, the addition algorithm uses random tape r to produce an updated accumulator state acc' and a list $W = (W[1], \ldots, W[t])$ of fresh membership witnesses for all elements of $S \| x$.
- $\mathsf{Ver}(\mathsf{crs}, \mathsf{acc}, x, w) \to \{0, 1\}$: On input an accumulator state acc, an element x, and a purported membership witness w, the verification algorithm outputs 1 if the witness is accepted and 0 otherwise.

We assume that crs encodes 1^λ, so the security parameter is omitted from the inputs to Add and Ver. Randomness used by Add is written explicitly when describing an execution.

Execution notation. For a fixed setup string crs, insertion sequence $S = (x_1, \ldots, x_n) \in \mathsf{List}_n(\mathcal{X}_\lambda)$, and addition random tapes $R = (r_1, \ldots, r_n)$, define the execution recursively by $\mathsf{acc}_0 = \perp$ and

$$(\mathsf{acc}_t, W_t) := \mathsf{Add}(\mathsf{crs}, S[1 : t-1], \mathsf{acc}_{t-1}, x_t; r_t). \tag{1}$$

Thus, the entire sequence $(\mathsf{acc}_t, W_t)_{t=1}^n$ is determined by (crs, S, R).

Public-state length. The size of the public accumulator state is one of the main resources in our tradeoff analysis. To measure this resource, we introduce a length bound that applies throughout the execution, rather than only to the final accumulator state.

Definition 3.2 (Accumulator-length bound)**.** Let $\Lambda = (\mathsf{SetUp}, \mathsf{Add}, \mathsf{Ver})$ be an additive positive accumulator. An *accumulator-length bound* for Λ is a polynomial-time computable function $\ell : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, where $\ell(\lambda, t)$ bounds the length of the public accumulator state after t insertions at security parameter λ. We require that ℓ is nondecreasing in its second argument and that, for every $\lambda \in \mathbb{N}$, every $n \le |\mathcal{X}_\lambda|$, and every insertion sequence $S \in \mathsf{List}_n(\mathcal{X}_\lambda)$, the resulting execution satisfies

$$|\mathsf{acc}_t| \le \ell(\lambda, t) \qquad \text{for all } t \in [n]$$

with probability one over the setup and addition randomness.

Update times and completeness. As new elements are inserted, previously issued membership witnesses may need to be refreshed to remain valid. We formalize this through two notions: an *update specification* determines the update times of each witness, while *completeness* requires that witnesses refreshed at these prescribed times remain valid. Following [10], we allow the update specification to depend arbitrarily on the realized execution prefix, without requiring it to be efficiently computable.

Definition 3.3 (Update specification)**.** Let $\mathsf{CRS}_\lambda := \mathrm{supp}(\mathsf{SetUp}(1^\lambda))$. An *update specification* is a family of functions

$$\mathsf{NeedUpd}_\lambda : \mathsf{CRS}_\lambda \times \mathbb{N} \times \mathsf{List}_{\le |\mathcal{X}_\lambda|}(\mathcal{X}_\lambda) \times (\{0,1\}^*)^* \to \{0, 1\}.$$

For an execution (crs, S, R) and $1 \le i < t \le |S|$, write

$$\mathsf{NeedUpd}(i, t) := \mathsf{NeedUpd}_\lambda(\mathsf{crs}, i, S[1 : t], R[1 : t]). \tag{2}$$

The value $\mathsf{NeedUpd}(i, t) = 1$ specifies that the witness for x_i is to be updated after round t.

Definition 3.4 (NeedUpd-completeness)**.** Let $\Lambda = (\mathsf{SetUp}, \mathsf{Add}, \mathsf{Ver})$ be an additive positive accumulator with update specification $\mathsf{NeedUpd}$. For an adversary $\mathcal{A}$, define the completeness game $\mathsf{Comp}^{\mathcal{A}}_{\Lambda,\mathsf{NeedUpd}}(1^\lambda)$ as follows:

1. The challenger samples $\mathsf{crs} \leftarrow \mathsf{SetUp}(1^\lambda)$, initializes $S = \varnothing$ and $\mathsf{acc}_0 = \perp$, and gives crs to $\mathcal{A}$.
2. On the t-th insertion request, $\mathcal{A}$ submits a fresh element $x_t \in \mathcal{X}_\lambda \setminus S$. The challenger samples an independent random tape r_t and computes

$$(\mathsf{acc}_t, W_t) := \mathsf{Add}(\mathsf{crs}, S, \mathsf{acc}_{t-1}, x_t; r_t).$$

   It sets $w_{t,t} := W_t[t]$ and, for each $i < t$, updates the retained witness according to

$$w_{i,t} := \begin{cases} W_t[i], & \mathsf{NeedUpd}(i,t) = 1, \\ w_{i,t-1}, & \mathsf{NeedUpd}(i,t) = 0. \end{cases} \tag{3}$$

   The challenger then sets $S \leftarrow S \| x_t$ and returns (acc_t, W_t) to $\mathcal{A}$.
3. After any round t, $\mathcal{A}$ may terminate the interaction and output an index $i \in [t]$. The game outputs 1 if $\mathsf{Ver}(\mathsf{crs}, \mathsf{acc}_t, x_i, w_{i,t}) = 0$, and 0 otherwise.

We say that Λ is $\mathsf{NeedUpd}$-*complete* if, for every possibly computationally unbounded adversary $\mathcal{A}$ making polynomially many insertion requests,

$$\Pr\left[\mathsf{Comp}^{\mathcal{A}}_{\Lambda,\mathsf{NeedUpd}}(1^\lambda) = 1\right] \le \mathsf{negl}(\lambda).$$

Soundness. Soundness requires that no PPT adversary can produce a valid membership witness for an element outside the accumulated set. We consider a weak soundness notion in which the adversary may adaptively choose the inserted elements and observe the resulting public accumulator states, but does not receive the genuine membership witnesses. Proving lower bounds under this weaker notion makes our results applicable to a broader class of accumulators.

Definition 3.5 (Soundness)**.** Let $\Lambda = (\mathsf{SetUp}, \mathsf{Add}, \mathsf{Ver})$ be an additive positive accumulator. For a PPT adversary $\mathcal{A}$, define the soundness game $\mathsf{Sound}^{\mathcal{A}}_{\Lambda}(1^\lambda)$ as follows:

1. The challenger samples $\mathsf{crs} \leftarrow \mathsf{SetUp}(1^\lambda)$, initializes $S = \varnothing$ and $\mathsf{acc}_0 = \perp$, and gives crs to $\mathcal{A}$.
2. On the t-th insertion request, $\mathcal{A}$ submits a fresh element $x_t \in \mathcal{X}_\lambda \setminus S$. The challenger samples an independent random tape r_t and computes

$$(\mathsf{acc}_t, W_t) := \mathsf{Add}(\mathsf{crs}, S, \mathsf{acc}_{t-1}, x_t; r_t).$$

   It sets $S \leftarrow S \| x_t$ and returns only acc_t to $\mathcal{A}$.
3. After polynomially many insertion requests, $\mathcal{A}$ terminates and outputs a pair (x^*, w^*). The game outputs 1 if $x^* \in \mathcal{X}_\lambda \setminus S$ and $\mathsf{Ver}(\mathsf{crs}, \mathsf{acc}_t, x^*, w^*) = 1$, and 0 otherwise.

We say that Λ is *sound* if, for every PPT adversary $\mathcal{A}$,

$$\Pr\left[\mathsf{Sound}^{\mathcal{A}}_{\Lambda}(1^\lambda) = 1\right] \le \mathsf{negl}(\lambda).$$

3.1.1 Update Graphs and Costs

Forward graphs and update graphs. We use forward graphs to represent the temporal structure of witness updates. A forward graph records directed relations between insertion rounds, with edges always pointing from an earlier round to a later one.

Definition 3.6 (Forward graph)**.** A *forward graph* on $[n]$ is a directed graph $G = ([n], E)$ such that every edge $(i,t) \in E$ satisfies $i < t$.

Since each witness is created at its insertion round and can only be updated in subsequent rounds, its update times naturally induce a forward graph.

Definition 3.7 (Update graph)**.** Let $\Lambda = (\mathsf{SetUp}, \mathsf{Add}, \mathsf{Ver})$ be an additive positive accumulator with update specification $\mathsf{NeedUpd}$. Every n-insertion execution (crs, S, R) induces a forward graph $G = G_\lambda(\mathsf{crs}, S, R)$ with

$$E(G) := \{(i,t) : 1 \le i < t \le n,\ \mathsf{NeedUpd}(i,t) = 1\}.$$

We call G the *update graph* of the execution. An edge (i,t) represents an update to the witness for x_i after insertion round t.

Update costs. The update graph captures when each witness is refreshed, but the cost of maintaining witnesses may depend on the application. Previous work has primarily considered two measures: the maximum number of updates to an individual witness and the total number of witness updates. We adopt a more general model in which the maintenance cost is an arbitrary nonnegative function of the update graph.

Definition 3.8 (Update cost)**.** Let $\mathcal{G}_n$ denote the set of forward graphs on $[n]$. An *update-cost function* is a function $C : \mathcal{G}_n \to \mathbb{R}_{\ge 0}$. For an additive positive accumulator Λ with update specification $\mathsf{NeedUpd}$, the update cost of an n-insertion execution (crs, S, R) is defined as

$$C(G_\lambda(\mathsf{crs}, S, R)).$$

3.2 Skipping Sequences and Temporal Blocking

Skipping sequences [9, 10] are central to our analysis, connecting cryptographic security with the combinatorial structure of witness-update graphs. On the cryptographic side, a skipping sequence identifies disjoint execution intervals whose witnesses remain valid at a common endpoint, enabling our information-theoretic lower bound. On the combinatorial side, the absence of long skipping sequences imposes structural constraints on update graphs, which we characterize using ordered binary trees.

Definition 3.9 (Skipping sequence and temporal blocking)**.** Let G be a forward graph on $[n]$, and let $\mathbf{s} = (s_1 < \cdots < s_m)$ be an increasing sequence of vertices. An edge $(s_j, t) \in E(G)$ *blocks* $\mathbf{s}$ at position $j < m$ if

$$s_{j+1} \le t \le s_m.$$

We say that G *blocks* $\mathbf{s}$ if some edge blocks it. Otherwise, $\mathbf{s}$ is called a *skipping sequence* in G.

3.3 Information-Theoretic Tools

We recall the information-theoretic quantities and inequalities used in our cryptographic lower bounds. Throughout the paper, all logarithms in information-theoretic expressions are natural.

Entropy and mutual information. For discrete random variables X, Y, Z, let $H(X)$ denote the Shannon entropy of X, and let $H(X \mid Y)$ denote its conditional entropy. The conditional mutual information is defined as

$$I(X; Y \mid Z) := H(X \mid Z) - H(X \mid Y, Z).$$

In particular, $I(X; Y \mid Z) \ge 0$. If X takes values in the set of bit strings of length at most L, then

$$H(X) < (L+1)\ln 2.$$

KL divergence. For probability distributions P and Q on the same discrete space, their Kullback–Leibler (KL) divergence is

$$D_{\mathrm{KL}}(P\|Q) := \sum_x P(x) \ln \frac{P(x)}{Q(x)}.$$

The divergence is nonnegative and is infinite if P is not absolutely continuous with respect to Q. For jointly distributed random variables X, Y, the conditional mutual information also admits the representation

$$I(X;Y \mid Z) = \mathbb{E}_{Y,Z}\left[D_{\mathrm{KL}}(P_{X|Y,Z}\|P_{X|Z})\right].$$

Chain rules and data processing. For random variables $X_1, \ldots, X_n, Y$, the mutual-information chain rule gives

$$I(X_1, \ldots, X_n; Y) = \sum_{i=1}^{n} I(X_i; Y \mid X_1, \ldots, X_{i-1}).$$

Similarly, for two joint distributions P, Q on $(X_1, \ldots, X_n)$, the KL chain rule states that

$$D_{\mathrm{KL}}(P\|Q) = \sum_{i=1}^{n} \mathbb{E}_P\left[D_{\mathrm{KL}}(P_{X_i|X_{<i}}\|Q_{X_i|X_{<i}})\right],$$

where $X_{<i} := (X_1, \ldots, X_{i-1})$. Both quantities satisfy data processing: applying the same randomized transformation to two distributions cannot increase their KL divergence, and processing a random variable cannot increase its mutual information with another variable.

Event probabilities and KL divergence. The following lemma provides a lower bound on the probability of an event under one distribution, given its probability under another distribution and an upper bound on their KL divergence.

Lemma 3.10 (KL rare-event bound)**.** *Let P, Q be distributions on the same space, and let E be an event with $p := P(E) > 0$ and $q := Q(E)$. If $D_{\mathrm{KL}}(P\|Q) \le \Gamma$, then*

$$q \ge p \exp\left(-1 - \frac{\Gamma}{p}\right).$$

Proof. By data processing,

$$\Gamma \ge p \ln \frac{p}{q} + (1-p) \ln \frac{1-p}{1-q}.$$

Since $(1-p)\ln((1-p)/(1-q)) \ge (1-p)\ln(1-p) \ge -p$, we obtain $p\ln(p/q) \le \Gamma + p$, which gives the claim. □

4 Security Implies Bounds on Skipping-Sequence Length

We show that cryptographic security limits the probability that an accumulator's update graph contains a long skipping sequence. Our main theorem gives a quantitative bound in terms of the accumulator-length bound and the completeness and soundness errors.

Definition 4.1 (Canonical random execution)**.** Let $\Lambda = (\mathsf{SetUp}, \mathsf{Add}, \mathsf{Ver})$ be an additive positive accumulator with update specification $\mathsf{NeedUpd}$. For $\lambda \in \mathbb{N}$ and $n \le |\mathcal{X}_\lambda|$, the experiment $\mathsf{Exec}_{\Lambda,n}(1^\lambda)$ samples $\mathsf{crs} \leftarrow \mathsf{SetUp}(1^\lambda)$, an insertion sequence $S = (x_1, \ldots, x_n)$ uniformly from $\mathsf{List}_n(\mathcal{X}_\lambda)$, and independent random tapes $R = (r_1, \ldots, r_n)$ for the addition algorithm. It outputs (crs, S, R), which determines the execution and its update graph.

Theorem 4.2 (Skipping-sequence length bound)**.** *Let Λ be an additive positive accumulator with update specification* NeedUpd *and accumulator-length bound ℓ. Let $n = n(\lambda)$ be efficiently computable and polynomially bounded. Let $\varepsilon_{\mathrm{comp}}$ and $\varepsilon_{\mathrm{sound}}$ be upper bounds on the completeness and soundness errors, respectively, against adversaries making at most n insertion requests.*

Let $(\mathsf{crs}, S, R) \leftarrow \mathsf{Exec}_{\Lambda,n}(1^\lambda)$, and let $G := G_\lambda(\mathsf{crs}, S, R)$ be the induced update graph. Write $\alpha(G)$ for the maximum length of a skipping sequence in G. Then, for every $2 \le m \le n$,

$$\Pr[\alpha(G) \ge m] \le n(n+1)\varepsilon_{\mathrm{comp}} + \sqrt{2e}\, n^3\, 2^{(\ell(\lambda,n)+1)/m} \sqrt{\varepsilon_{\mathrm{sound}} + \frac{2n^2}{|\mathcal{X}_\lambda| - n}}. \tag{4}$$

Proof roadmap. The proof is organized into four subsections. In Section 4.1, we show that a skipping sequence determines disjoint replay intervals, each containing a witness that is retained at a common endpoint. We use completeness to bound the probability that these witnesses fail verification. In Section 4.2, we resample one uniformly selected interval and use the KL chain rule to bound the resulting loss in verification probability. In Section 4.3, we construct a PPT soundness adversary that guesses the interval and witness without identifying the skipping sequence. Finally, in Section 4.4, we combine these results to prove Theorem 4.2 and derive its consequences.

4.1 Skipping Sequences and Replay Intervals

Fix $2 \le m \le n$. Write $Z_t := (x_t, r_t)$ and $X := (\mathsf{crs}, Z_1, \ldots, Z_n)$, and let R denote the distribution of X induced by $\mathsf{Exec}_{\Lambda,n}(1^\lambda)$. Define

$$B := \{\alpha(G) \ge m\}, \qquad q := \Pr[B].$$

If $q = 0$, the desired probability bound is immediate, so assume $q > 0$.

Conditioned on B, let $K = (s_1 < \cdots < s_m)$ be the lexicographically first skipping sequence of length m. This choice is used only in the analysis and need not be efficiently computable. Write $v := s_m$, set $s_{m+1} := v + 1$, and define

$$I_j := [s_j, s_{j+1} - 1], \qquad j \in [m].$$

These intervals are pairwise disjoint and partition $[s_1, v]$.

Lemma 4.3 (Retained witnesses in replay intervals)**.** *For every execution in B and every $j \in [m]$,*

$$w_{s_j,v} \in \{W_u[s_j] : u \in I_j\}.$$

Proof. For $j < m$, the skipping-sequence property guarantees that $\mathsf{NeedUpd}(s_j, t) = 0$ for all $s_{j+1} \le t \le v$. Consequently, the witness retained at round v was issued at some round in I_j. For $j = m$, the claim follows from $I_m = \{v\}$ and $w_{v,v} = W_v[v]$. □

The lemma shows that each interval contains a witness retained at the common endpoint v, without requiring us to identify its last update time. We exploit this property through interval replay.

Interval replay. Given an execution and an interval $[s, t]$ with $t \le v$, we *replay* the interval by starting from the original execution prefix $(S[1 : s-1], \mathsf{acc}_{s-1})$ and running Add on a sequence of insertions and random tapes for rounds $s, \ldots, t$. This produces a local execution and witness lists $W'_s, \ldots, W'_t$. We collect the witnesses $W'_u[s]$, for $u \in [s, t]$, issued for the element inserted at round s, and test them against the original accumulator state acc_v. If the replay uses the original insertions and randomness, it reproduces the original witnesses; otherwise, it produces witnesses for a fresh continuation from the same prefix.

We now define two replay distributions. For each $j \in [m]$, write

$$P_j := (\mathsf{crs}, Z_1, \ldots, Z_{s_j-1}), \qquad C_j := Z_{s_j:s_{j+1}-1}, \qquad Y := \mathsf{acc}_v.$$

Conditioned on B, choose $A \leftarrow [m]$ uniformly and independently. The following distributions differ only in how the selected interval is generated:

- Under the *genuine distribution* P_B, sample $(\mathsf{crs}, S, R) \leftarrow \mathsf{Exec}_{\Lambda,n}(1^\lambda)$ conditioned on B, determine K, choose $A \leftarrow [m]$ uniformly, and replay I_A using its original insertions and randomness C_A.
- Under the *resampled distribution* Q_B, preserve the joint distribution of (K, A, P_A, Y) from P_B, but replay I_A using an independent sample

$$C'_A \leftarrow \mathsf{R}(C_A \mid P_A). \tag{5}$$

  Thus, the local execution uses a fresh canonical continuation from the original prefix, while the verification state Y remains unchanged.

In either experiment, let E be the event that at least one witness generated during the replay of I_A for the element inserted at round s_A verifies against Y, using independent verifier randomness for each candidate.

Lemma 4.4 (Completeness of genuine replay)**.** *The genuine replay distribution satisfies*

$$P_B(E) \geq 1 - \frac{\binom{n+1}{2}\varepsilon_{\mathrm{comp}}}{q}. \tag{6}$$

Proof. Under P_B, replaying I_A reproduces the original witnesses. By Lemma 4.3, these candidates include $w_{s_A,v}$. Hence, failure of E implies that this retained witness fails verification against acc_v.

For every fixed pair $1 \leq i \leq t \leq n$, the probability that $w_{i,t}$ fails verification in the canonical execution is at most $\varepsilon_{\mathrm{comp}}$, since a completeness adversary can generate this execution through round t and output i. As (s_A, v) is one of these pairs,

$$\Pr[B \wedge E^c] \leq \sum_{1 \leq i \leq t \leq n} \Pr[\mathsf{Ver}(\mathsf{crs}, \mathsf{acc}_t, x_i, w_{i,t}) = 0] \leq \binom{n+1}{2}\varepsilon_{\mathrm{comp}}.$$

Dividing by $q = \Pr[B]$ proves the claim. □

4.2 Bounding KL Divergence Between Genuine and Resampled Distributions

We now bound the KL divergence between the genuine and resampled distributions P_B and Q_B. Recall that these distributions agree on the execution prefix and the endpoint state Y, but differ in how the selected interval is generated. Under P_B, the interval comes from an execution conditioned on B; under Q_B, it is sampled independently from the canonical continuation distribution.

There are two sources of divergence. First, the original interval may be correlated with the endpoint state Y, whereas the resampled interval is independent of Y given the execution prefix. Second, conditioning on B and selecting the skipping sequence K may change the distribution of the interval itself. We control the first contribution using the accumulator-length bound and the second using the KL chain rule. Since the intervals are disjoint, both contributions can be amortized over the m intervals.

Lemma 4.5 (KL bound for interval resampling)**.** *The distributions P_B and Q_B satisfy*

$$D_{\mathrm{KL}}(P_B \| Q_B) \leq \frac{(\ell(\lambda, n) + 1)\ln 2 + \ln(1/q)}{m} + \ln n. \tag{7}$$

Proof. We first condition on the selected skipping sequence. Fix $K = k$ with positive conditional probability, and write $\mathsf{R}_k := \mathsf{R}(\cdot \mid B, K = k)$. Since k is fixed, the intervals $I_1, \ldots, I_m$ and their common endpoint v are deterministic.

Step 1: Decomposing the divergence. For each interval I_j, the genuine distribution of (P_j, C_j, Y) is $\mathsf{R}_k(P_j, C_j, Y)$, whereas the resampled distribution preserves (P_j, Y) but draws C_j from $\mathsf{R}(C_j \mid P_j)$. Write

$$D_j(k) := D_{\mathrm{KL}}\left(\mathsf{R}_k(P_j, C_j, Y) \,\|\, \mathsf{R}_k(P_j, Y)\mathsf{R}(C_j \mid P_j)\right).$$

By the KL chain rule,

$$\begin{aligned} D_j(k) &= \mathbb{E}_{\mathsf{R}_k}\left[D_{\mathrm{KL}}\left(\mathsf{R}_k(C_j \mid P_j, Y) \,\|\, \mathsf{R}(C_j \mid P_j)\right)\right] \\ &= I_{\mathsf{R}_k}(C_j; Y \mid P_j) + \mathbb{E}_{\mathsf{R}_k}\left[D_{\mathrm{KL}}\left(\mathsf{R}_k(C_j \mid P_j) \,\|\, \mathsf{R}(C_j \mid P_j)\right)\right]. \end{aligned} \tag{8}$$

The first term measures the information that the endpoint Y reveals about the interval beyond its prefix. The second measures how conditioning on $(B, K = k)$ changes the interval distribution relative to an unconditional canonical continuation.

Step 2: Amortizing over disjoint intervals. The key observation is that the intervals partition $[s_1, v]$. In particular, $P_{j+1} = (P_j, C_j)$ for every $j < m$. Thus, the mutual-information chain rule gives

$$\begin{aligned} \sum_{j=1}^{m} I_{\mathsf{R}_k}(C_j; Y \mid P_j) &= I_{\mathsf{R}_k}(C_1, \ldots, C_m; Y \mid P_1) \\ &\le H_{\mathsf{R}_k}(Y) \\ &< (\ell(\lambda, n) + 1)\ln 2. \end{aligned} \tag{9}$$

The last inequality follows because $Y = \mathsf{acc}_v$ has length at most $\ell(\lambda, n)$ bits.

For the second term, applying the KL chain rule to the complete execution, partitioned into the prefix before s_1, the intervals $C_1, \ldots, C_m$, and the suffix after v, yields

$$\sum_{j=1}^{m} \mathbb{E}_{\mathsf{R}_k}\left[D_{\mathrm{KL}}\left(\mathsf{R}_k(C_j \mid P_j) \,\|\, \mathsf{R}(C_j \mid P_j)\right)\right] \le D_{\mathrm{KL}}(\mathsf{R}_k \| \mathsf{R}). \tag{10}$$

Indeed, the left-hand side consists of the KL contributions of the selected intervals; the omitted prefix and suffix contributions are nonnegative.

Combining Eqs. (8) to (10), we obtain

$$\sum_{j=1}^{m} D_j(k) \le (\ell(\lambda, n) + 1)\ln 2 + D_{\mathrm{KL}}(\mathsf{R}_k \| \mathsf{R}).$$

Since A is uniformly distributed over $[m]$ under both P_B and Q_B, averaging over A and then over K conditioned on B gives

$$D_{\mathrm{KL}}(P_B \| Q_B) \le \frac{(\ell(\lambda, n) + 1)\ln 2}{m} + \frac{1}{m}\mathbb{E}_{K|B}\left[D_{\mathrm{KL}}(\mathsf{R}_K \| \mathsf{R})\right]. \tag{11}$$

Step 3: Bounding the cost of selecting the skipping sequence. It remains to bound the divergence caused by conditioning on B and K. For each possible sequence k, let $p_k := \Pr[K = k \mid B]$. Since both B and K are determined by the canonical execution, conditioning on $(B, K = k)$ restricts R to an event of probability qp_k. Hence,

$$D_{\mathrm{KL}}(\mathsf{R}_k \| \mathsf{R}) = \ln \frac{1}{qp_k}.$$

Averaging over K yields

$$\begin{aligned}\mathbb{E}_{K|B}\left[D_{\mathrm{KL}}(\mathsf{R}_K\|\mathsf{R})\right] &= \ln\frac{1}{q} + H(K \mid B)\\ &\le \ln\frac{1}{q} + \ln\binom{n}{m}\\ &\le \ln\frac{1}{q} + m\ln n. \end{aligned} \tag{12}$$

Substituting this bound into Eq. (11) proves the lemma. □

We now use this divergence bound to transfer the verification guarantee from genuine replay to resampled replay.

Lemma 4.6 (Fresh replay success)**.** *Suppose* $q > \binom{n+1}{2}\varepsilon_{\mathrm{comp}}$*, and set*

$$p := 1 - \frac{\binom{n+1}{2}\varepsilon_{\mathrm{comp}}}{q}.$$

Then

$$Q_B(E) \ge p\exp\left(-1 - \frac{(\ell(\lambda,n)+1)\ln 2 + \ln(1/q) + m\ln n}{mp}\right). \tag{13}$$

Proof. By Lemma 4.4, the genuine verification event satisfies $P_B(E) \ge p > 0$. Moreover, Lemma 4.5 gives

$$D_{\mathrm{KL}}(P_B\|Q_B) \le \frac{(\ell(\lambda,n)+1)\ln 2 + \ln(1/q)}{m} + \ln n.$$

Applying the KL rare-event bound from the preliminaries to E gives the claimed lower bound on $Q_B(E)$. □

4.3 A PPT Soundness Attack

The resampled distribution Q_B depends on a skipping sequence selected from the genuine execution, which may not be efficiently computable. We construct a PPT soundness adversary that avoids identifying this sequence by guessing the replay interval and its common endpoint. The adversary then selects one of the witnesses generated during the replay. We show that these guesses incur only a polynomial loss in success probability.

Let

$$\mathcal{J} := \{(s,t,v) : 1 \le s \le t \le v \le n\}, \qquad N := |\mathcal{J}| = \binom{n+2}{3}.$$

Definition 4.7 (Guessing adversary)**.** The adversary $\mathcal{A}_n$ operates in the soundness game as follows:

1. Sample (s,t,v) uniformly from $\mathcal{J}$.
2. Generate a canonical random execution through round v by submitting uniformly sampled fresh elements to the challenger. Store the insertion sequence $S = (x_1,\ldots,x_v)$ and the public accumulator states $\mathsf{acc}_0,\ldots,\mathsf{acc}_v$.
3. Starting from the original prefix $(S[1:s-1],\mathsf{acc}_{s-1})$, independently replay rounds $s,\ldots,t$ using fresh insertion elements and addition randomness. Write x'_s for the first replayed element and $W'_u[s]$ for its witness issued at round $u \in [s,t]$.
4. Sample U uniformly from $[s,t]$. If $x'_s \in S$, abort. Otherwise, output $(x'_s, W'_U[s])$ against the genuine accumulator state acc_v.

The adversary uses only the public accumulator states returned by the challenger and witnesses generated by its own executions of Add. It neither evaluates $\mathsf{NeedUpd}$ nor receives genuine witness lists. In particular, $\mathcal{A}_n$ is independent of the skipping-sequence length m.

Lemma 4.8 (Success probability of the guessing adversary)**.** *For every $2 \le m \le n$ with $q > 0$, the adversary $\mathcal{A}_n$ satisfies*

$$\Pr[\mathsf{Sound}_{\Lambda}^{\mathcal{A}_n}(1^\lambda) = 1] \ge \frac{q}{Nn} Q_B(E) - \frac{2n^2}{|\mathcal{X}_\lambda| - n} \tag{14}$$

for all sufficiently large λ.

Proof. We first analyze the adversary under ideal sampling, ignoring collisions between the replayed target and the genuine accumulated set. We then account for these errors.

Step 1: Guessing a successful witness. Fix a realization $\mathsf{e} = (\mathsf{crs}, S, R)$ of the canonical execution and a triple $j = (s, t, v) \in \mathcal{J}$. Let $b_{\mathsf{e}}(j)$ be the probability that at least one candidate witness generated by a fresh replay of $[s, t]$ verifies against acc_v. For each $u \in [s, t]$, let $a_{\mathsf{e}}(j, u)$ be the probability that the witness issued at replay round u verifies against the same state. All probabilities include the fresh replay randomness and independent verifier randomness.

By the union bound,

$$b_{\mathsf{e}}(j) \le \sum_{u=s}^{t} a_{\mathsf{e}}(j, u).$$

Since the adversary selects U uniformly from $[s, t]$, its verification probability for the fixed triple j is at least

$$\frac{1}{t - s + 1} \sum_{u=s}^{t} a_{\mathsf{e}}(j, u) \ge \frac{b_{\mathsf{e}}(j)}{n}. \tag{15}$$

Thus, selecting one candidate witness incurs a loss of at most a factor of n.

Step 2: Guessing a successful interval. Now suppose $\mathsf{e} \in B$. The skipping sequence K selected in the analysis, together with the uniform choice of $A \in [m]$, induces a distribution π_{e} over $\mathcal{J}$: the selected interval $I_A = [s_A, s_{A+1} - 1]$ and its endpoint $v = s_m$ determine the triple $(s_A, s_{A+1} - 1, v)$.

By the definition of the resampled distribution Q_B,

$$Q_B(E) = \mathbb{E}_{\mathsf{e} \leftarrow \mathsf{Exec}_{\Lambda,n}(1^\lambda) | B} \left[\sum_{j \in \mathcal{J}} \pi_{\mathsf{e}}(j) b_{\mathsf{e}}(j) \right]. \tag{16}$$

In contrast, $\mathcal{A}_n$ chooses its triple uniformly from $\mathcal{J}$, independently of the execution. By Eq. (15), for every fixed $\mathsf{e} \in B$, its verification probability is at least

$$\frac{1}{N} \sum_{j \in \mathcal{J}} \frac{b_{\mathsf{e}}(j)}{n} \ge \frac{1}{Nn} \sum_{j \in \mathcal{J}} \pi_{\mathsf{e}}(j) b_{\mathsf{e}}(j), \tag{17}$$

where the inequality follows from $0 \le \pi_{\mathsf{e}}(j) \le 1$.

The genuine execution generated by $\mathcal{A}_n$ follows the unconditional canonical distribution. Although the adversary stops after round v, we may equivalently sample the remaining rounds through n without revealing them to the adversary. Averaging Eq. (17) over these complete executions and retaining only the contribution from B, we obtain

$$\Pr[\text{the replayed witness verifies}] \ge \frac{q}{Nn} Q_B(E).$$

This establishes the desired bound under ideal sampling, before enforcing the requirement that the replayed element be absent from the genuine accumulated set.

Step 3: Bounding collisions and sampling errors. Conditioned on the genuine execution prefix, the replayed target x'_s is uniformly distributed over $\mathcal{X}_\lambda \setminus S[1:s-1]$. Therefore,

$$\Pr[x'_s \in S] \le \frac{v-s+1}{|\mathcal{X}_\lambda| - s + 1} \le \frac{n}{|\mathcal{X}_\lambda| - n}.$$

Whenever this collision does not occur, an accepted replayed witness constitutes a valid soundness forgery.

Uniform sampling without replacement can be implemented by rejection sampling with a fixed polynomial cutoff. Using at most two independent attempts for each fresh element, the probability of a sampling failure during either the genuine execution or the replay is at most

$$\frac{2n^3}{(|\mathcal{X}_\lambda| - n)^2}.$$

Since n is polynomially bounded and $|\mathcal{X}_\lambda| \ge 2^{\Omega(\lambda)}$, the combined collision and sampling error is at most $2n^2/(|\mathcal{X}_\lambda| - n)$ for all sufficiently large λ. Subtracting this error from the ideal verification probability proves the lemma. □

4.4 Proof of the Main Theorem and Consequences

We now combine the bounds from the preceding subsections to prove the main theorem and derive its consequences for sound and complete accumulators.

Proof of Theorem 4.2. Fix $2 \le m \le n$ and let $q := \Pr[\alpha(G) \ge m]$. If $q \le n(n+1)\varepsilon_{\text{comp}}$, the claimed bound holds immediately. Otherwise, define

$$p := 1 - \frac{\binom{n+1}{2}\varepsilon_{\text{comp}}}{q} > \frac{1}{2}.$$

By Lemma 4.6, the resampled replay succeeds with probability at least

$$Q_B(E) \ge p \exp\left(-1 - \frac{(\ell(\lambda,n)+1)\ln 2 + \ln(1/q) + m\ln n}{mp}\right).$$

Combining this with Lemma 4.8 and the soundness bound for $\mathcal{A}_n$, we obtain

$$\varepsilon_{\text{sound}} + \frac{2n^2}{|\mathcal{X}_\lambda| - n} \ge \frac{qp}{Nn} \exp\left(-1 - \frac{(\ell(\lambda,n)+1)\ln 2 + \ln(1/q) + m\ln n}{mp}\right). \tag{18}$$

Using $p > 1/2$, we can remove the dependence on p and simplify the right-hand side:

$$\begin{aligned}\varepsilon_{\text{sound}} + \frac{2n^2}{|\mathcal{X}_\lambda| - n} &\ge \frac{q}{2eNn} \exp\left(-\frac{2((\ell(\lambda,n)+1)\ln 2 + \ln(1/q) + m\ln n)}{m}\right) \\ &= \frac{q^{1+2/m}}{2eNn^3}\, 2^{-2(\ell(\lambda,n)+1)/m}.\end{aligned} \tag{19}$$

Rearranging yields

$$q \le \left(2eNn^3\, 2^{2(\ell(\lambda,n)+1)/m}\left(\varepsilon_{\text{sound}} + \frac{2n^2}{|\mathcal{X}_\lambda| - n}\right)\right)^{m/(m+2)}.$$

Since $q \le 1$ and $m/(m+2) \ge 1/2$, we may weaken this bound by replacing the exponent $m/(m+2)$ with $1/2$. Finally, $N = \binom{n+2}{3} \le n^3$ for $n \ge 2$, so

$$q \le \sqrt{2e}\, n^3\, 2^{(\ell(\lambda,n)+1)/m} \sqrt{\varepsilon_{\text{sound}} + \frac{2n^2}{|\mathcal{X}_\lambda| - n}}.$$

Combining this with the first case proves Eq. (4). Since $\mathcal{A}_n$ is independent of m, the bound holds for every $2 \le m \le n$. □

Our quantitative bound implies that sufficiently long skipping sequences occur with negligible probability in sound and complete accumulators.

Corollary 4.9 (Absence of long skipping sequences)**.** *Let Λ be a sound,* NeedUpd*-complete additive positive accumulator with accumulator-length bound ℓ. Let $n = n(\lambda)$ be efficiently computable and polynomially bounded, and let $2 \le m = m(\lambda) \le n$ satisfy*

$$\frac{\ell(\lambda, n) + 1}{m} = O(\log \lambda).$$

Let $(\mathsf{crs}, S, R) \leftarrow \mathsf{Exec}_{\Lambda,n}(1^\lambda)$ and $G := G_\lambda(\mathsf{crs}, S, R)$. Then

$$\Pr[\alpha(G) \ge m] = \mathsf{negl}(\lambda).$$

Proof. By NeedUpd-completeness and soundness, the errors $\varepsilon_{\text{comp}}$ and $\varepsilon_{\text{sound}}$ in Theorem 4.2 are negligible. Since n is polynomially bounded and $|\mathcal{X}_\lambda| \ge 2^{\Omega(\lambda)}$, the collision term is also negligible. Moreover, the assumption $(\ell(\lambda, n) + 1)/m = O(\log \lambda)$ implies that $2^{(\ell(\lambda,n)+1)/m}$ is polynomially bounded. The conclusion follows from Eq. (4). □

5 Matching Upper Bounds

Section 4 establishes a lower bound on the accumulator length in terms of the skipping-sequence length of the update graphs. We now show that this bound is tight for arbitrary update-cost functions. We first give a construction for any forward graph G, with accumulator length at most $\sigma\alpha(G)$, where σ is the output length of a collision-resistant hash function. We then choose σ to obtain an asymptotically tight accumulator-length bound.

5.1 Exact Realization of Update Graphs

Our construction uses hashes of insertion prefixes. The witness issued for x_i at round u contains the insertion prefix $(x_1, \ldots, x_u)$. It can be verified against the hash of any prefix $(x_1, \ldots, x_s)$ with $i \le s \le u$. We select which prefix hashes to publish at each round so that all retained witnesses remain valid.

Theorem 5.1 (Exact graph realization)**.** *Let $n = n(\lambda)$ be efficiently computable and polynomially bounded, and let $G = G_\lambda$ be a forward graph on $[n]$ whose edge set can be computed in polynomial time from 1^λ. Given a collision-resistant hash family $H_\kappa : \{0,1\}^* \to \{0,1\}^\sigma$, there is a sound, perfectly* NeedUpd*-complete additive positive accumulator whose update graph during the first n insertions is exactly G and whose accumulator states satisfy*

$$|\mathsf{acc}_t| \le \sigma\alpha(G) \qquad \text{for every } t \in [n]. \tag{20}$$

For any nonnegative update-cost function $C : \mathcal{G}_n \to \mathbb{R}_{\ge 0}$ and $1 \le w \le n$, define

$$\mathrm{OPT}_w(C) := \min_{G:\alpha(G)\le w} C(G). \tag{21}$$

For every graph attaining $\mathrm{OPT}_w(C)$, the construction uses at most w prefix hashes and has update cost exactly $\mathrm{OPT}_w(C)$. If such a graph can be computed in polynomial time, this gives a PPT accumulator. Thus the skipping-sequence restriction from Section 4 gives a matching bound on update costs at the combinatorial level.

Proof of Theorem 5.1. We first determine which prefix hashes must be published at each round.

For every $1 \le i \le t \le n$, define

$$r_i(t) := \max\big(\{i\} \cup \{u \le t : (i, u) \in E(G)\}\big).$$

This is the most recent round at which the witness for x_i is prescribed to be updated, including its insertion round. The retained witness for x_i at round t contains the insertion prefix through round $r_i(t)$ and can therefore be verified using any prefix hash indexed by

$$s \in I_i(t) := [i, r_i(t)].$$

At each round t, we need a set $P_t \subseteq [t]$ such that

$$P_t \cap I_i(t) \neq \varnothing \qquad \text{for every } i \leq t. \tag{22}$$

Let $\tau_t(G)$ be the minimum cardinality of such a set. We show that

$$\max_{t \in [n]} \tau_t(G) = \alpha(G). \tag{23}$$

For intervals on a line, the minimum number of points intersecting every interval equals the maximum number of pairwise disjoint intervals. Both quantities are obtained by repeatedly selecting the smallest right endpoint and removing all intervals containing it.

Fix t, and let $I_{s_1}(t), \ldots, I_{s_k}(t)$ be pairwise disjoint intervals with $s_1 < \cdots < s_k$. Then $r_{s_j}(t) < s_{j+1}$ for every $j < k$. Since $s_k \leq t$, no edge (s_j, u) satisfies $s_{j+1} \leq u \leq s_k$. Hence $(s_1, \ldots, s_k)$ is a skipping sequence, giving $\tau_t(G) \leq \alpha(G)$.

Conversely, let $(s_1 < \cdots < s_m)$ be a skipping sequence of maximum length $m = \alpha(G)$, and set $v := s_m$. For every $j < m$, the skipping-sequence property implies $r_{s_j}(v) < s_{j+1}$, while $I_v(v) = \{v\}$. Therefore $I_{s_1}(v), \ldots, I_{s_m}(v)$ are pairwise disjoint, giving $\tau_v(G) \geq m$. This proves Eq. (23).

We now construct the accumulator. For each $t \in [n]$, let P_t be a minimum set satisfying Eq. (22), computed by the greedy algorithm. Let enc be an injective, polynomial-time computable encoding of insertion prefixes, including their lengths. We define the accumulator and its update specification as follows.

- **SetUp.** Sample a hash key κ and output crs containing κ. The initial accumulator state is $\mathsf{acc}_0 = \perp$.
- **Add.** Given the insertion list $(x_1, \ldots, x_{t-1})$ and a fresh element x_t, for $t \leq n$, compute $R_s := H_\kappa(\mathrm{enc}(s, x_1, \ldots, x_s))$ for each $s \in P_t$, and output

$$\mathsf{acc}_t := \Big\|_{s \in P_t} R_s, \tag{24}$$

$$W_t[i] := (i; x_1, \ldots, x_t), \qquad i \in [t]. \tag{25}$$

  The hash values are concatenated in increasing order of s. For $t > n$, output the single hash $H_\kappa(\mathrm{enc}(t, x_1, \ldots, x_t))$ as acc_t, together with the same witnesses $W_t[i]$.
- **Ver.** Given crs, acc, an element x, and a witness $w = (i; y_1, \ldots, y_u)$, reject if $\mathsf{acc} = \perp$, w is malformed, $i \notin [u]$, or $y_i \neq x$. Otherwise, parse acc into σ-bit blocks, rejecting if the parsing fails. Accept if and only if

$$H_\kappa(\mathrm{enc}(s, y_1, \ldots, y_s)) = R \tag{26}$$

  for some $s \in [i, u]$ and some block R of acc.
- **Update specification.** For every $1 \leq i < t \leq n$, set

$$\mathsf{NeedUpd}(i, t) := \mathbf{1}[(i, t) \in E(G)]. \tag{27}$$

  For $t > n$, set $\mathsf{NeedUpd}(i, t) = 1$ for every $i < t$.

We verify the claimed properties.

- *Completeness.* Fix an insertion sequence and a round $t \le n$. For every $i \le t$, the witness retained according to NeedUpd is $W_{r_i(t)}[i]$. By Eq. (22), there exists $s \in P_t$ satisfying $i \le s \le r_i(t)$. The retained witness contains $(x_1, \dots, x_s)$, whose hash appears in acc_t. Hence verification succeeds. For $t > n$, every retained witness contains the complete insertion list, whose hash is published in acc_t. Thus the accumulator is perfectly NeedUpd-complete.

- *Soundness.* Suppose an adversary outputs an accepted pair (x^*, w^*) after round t, where x^* is not accumulated. Write $w^* = (i; y_1, \dots, y_u)$. Acceptance implies $y_i = x^*$ and, for some $s \in [i, u]$ and some published prefix index v,

$$H_\kappa(\mathrm{enc}(s, y_1, \dots, y_s)) = H_\kappa(\mathrm{enc}(v, x_1, \dots, x_v)).$$

  The encoded inputs are distinct. If $s \neq v$, their prefix lengths differ. If $s = v$, their insertion lists differ at position i, since $y_i = x^*$ is not accumulated. Thus every successful forgery yields a collision in H_κ, contradicting collision resistance.

- *Update graph and accumulator length.* By Eq. (27), the update graph during the first n insertions is exactly G. Moreover, by Eq. (23),

$$|\mathsf{acc}_t| = \sigma |P_t| = \sigma \tau_t(G) \le \sigma \alpha(G)$$

  for every $t \in [n]$. Since the verifier checks every candidate prefix against every published hash value, neither prefix indices nor the current round need to be stored in the accumulator.

  Finally, Eq. (23) shows that every choice of sets satisfying Eq. (22) requires at least $\alpha(G)$ prefix hashes at some round. Our construction attains this number.

□

5.2 Asymptotically Tight Accumulator-Length Bounds

The preceding theorem realizes every forward graph G with accumulator length at most $\sigma\alpha(G)$. We now choose σ to match the lower bound of Section 4.

For any nonnegative update-cost function $C = C_\lambda : \mathcal{G}_n \to \mathbb{R}_{\ge 0}$ and feasible cost threshold $c = c(\lambda)$, define

$$R_C(c) := \min_{G : C(G) \le c} \alpha(G). \tag{28}$$

Theorem 5.2 (Tight accumulator-length bound)**.** *Let $n = n(\lambda)$ be efficiently computable and polynomially bounded, and let $B = B(\lambda)$ be a positive, efficiently computable, polynomially bounded integer. For any nonnegative update-cost function $C = C_\lambda : \mathcal{G}_n \to \mathbb{R}_{\ge 0}$ and feasible cost threshold $c = c(\lambda)$, write $r := R_C(c)$ and suppose $2 \le r \le n$.*

Any sound, NeedUpd-complete additive positive accumulator with accumulator-length bound $\ell(\lambda, n) \le B$ that achieves $C(G) \le c$ with overwhelming probability under the canonical random execution must satisfy

$$r \log \lambda = o(B). \tag{29}$$

Conversely, if this condition holds, collision-resistant hash functions with output length $\sigma = \lfloor B/r \rfloor$ are available, and a graph G^* attaining the minimum defining r can be computed in polynomial time, then G^* can be realized by a sound, perfectly NeedUpd-complete accumulator with accumulator length at most B and update cost at most c.

Proof. Suppose $r \log \lambda = o(B)$ does not hold. Then there exist a constant $K > 0$ and infinitely many security parameters for which $B \le Kr \log \lambda$. Every graph with $C(G) \le c$ has $\alpha(G) \ge r$, so the assumed accumulator satisfies

$$\Pr[\alpha(G) \ge r] \ge \Pr[C(G) \le c] = 1 - \mathsf{negl}(\lambda).$$

However, along the same sequence of security parameters,

$$\frac{\ell(\lambda, n) + 1}{r} \leq \frac{B + 1}{r} = O(\log \lambda).$$

Since $2 \leq r \leq n$, the skipping-sequence bound of Section 4 implies that $\Pr[\alpha(G) \geq r]$ is negligible along this sequence, a contradiction.

For the converse, choose a graph G^* with $\alpha(G^*) = r$ and $C(G^*) \leq c$, and let $\sigma = \lfloor B/r \rfloor$. Applying Theorem 5.1 with a collision-resistant hash function of output length σ gives an accumulator satisfying

$$|\mathsf{acc}_t| \leq \sigma\alpha(G^*) = \sigma r \leq B$$

for every $t \in [n]$. For $t > n$, the accumulator stores one σ-bit hash value, so the same bound holds. Its update graph during the first n insertions is exactly G^*, and hence its update cost is $C(G^*) \leq c$. □

For graph families that cannot be computed in polynomial time, the same construction gives nonuniform polynomial-size implementations, assuming collision resistance against nonuniform polynomial-time adversaries.

6 Concrete Witness-Update Tradeoffs

We apply the results of Sections 4 and 5 to study concrete witness-update tradeoffs. We begin with the maximum and total numbers of witness updates, which have been studied in prior work, and derive lower and upper bounds under different accumulator-length bounds. We then consider communication costs that depend on the insertion time of a value or on both its insertion time and the update time. For these heterogeneous costs, we give polynomial-time algorithms for finding optimal update graphs or approximating the optimum, depending on the weights.

6.1 Maximum and Total Updates

We consider two update costs: the maximum number of updates to any witness, $C_{\max}(G) := \max_{i\in[n]} d_G(i)$, and the total number of updates, $C_{\mathrm{tot}}(G) := |E(G)| = \sum_{i=1}^{n} d_G(i)$, where $d_G(i) := |\{t > i : (i, t) \in E(G)\}|$. The following lemma determines the minimum of each cost over graphs with $\alpha(G) \leq w$ and constructs a graph that attains both minima.

Lemma 6.1. *For every integer $w \geq 1$, define*

$$q(n, w) := \max\left\{d \geq 0 : \binom{w+d}{d} \leq n\right\}, \tag{30}$$

$$\Phi_w(n) := \sum_{d\geq 1}\left(n + 1 - \binom{w+d}{d}\right)_+. \tag{31}$$

Then

$$\min_{G:\alpha(G)\leq w} \max_{i\in[n]} d_G(i) = q(n, w), \tag{32}$$

$$\min_{G:\alpha(G)\leq w} |E(G)| = \Phi_w(n). \tag{33}$$

Moreover, there exists a polynomial-time constructible forward graph G^* with $\alpha(G^*) \leq w$ such that

$$\max_{i\in[n]} d_{G^*}(i) = q(n, w) \qquad \text{and} \qquad |E(G^*)| = \Phi_w(n).$$

Proof. We first show that every graph G with $\alpha(G) \le w$ has an ordered binary tree T on $[n]$ such that every root-to-node path has fewer than w right edges and the number $\ell_T(i)$ of left edges on the path to i satisfies $\ell_T(i) \le d_G(i)$. We prove this recursively on intervals of vertices. For the graph induced by the current interval, let $R_G(i)$ be the largest endpoint of an outgoing edge from i, or i if no such edge exists. Choose $v = \min_i R_G(i)$ as the root, and choose i_0 with $R_G(i_0) = v$. Every $i < v$ has an edge ending at or after v, so restricting the graph to the vertices before v decreases each of their out-degrees by at least one. This restriction still has $\alpha \le w$. The restriction to vertices after v has $\alpha \le w - 1$: otherwise, a skipping sequence of length w there could be preceded by i_0, giving a skipping sequence of length $w + 1$ in the original graph. In particular, the right interval is empty when $w = 1$. Recursively construct the left subtree with parameter w and the right subtree with parameter $w - 1$. Attaching them at v adds one left edge to each path in the left subtree and one right edge to each path in the right subtree. The resulting tree therefore satisfies the claimed bounds.

A root-to-node path with fewer than w right edges and fewer than d left edges is represented by a word over $\{L, R\}$ containing fewer than w occurrences of R and fewer than d occurrences of L. There are exactly $\binom{w+d}{d} - 1$ such words. Consequently, for every $d \ge 1$,

$$|\{i : \ell_T(i) \ge d\}| \ge \left(n + 1 - \binom{w+d}{d}\right)_+ .$$

Order all words with fewer than w occurrences of R by their number of occurrences of L, then by word length, breaking ties arbitrarily. The first n words form a prefix-closed binary tree T^* and attain equality in the preceding inequality for every d.

Label the nodes of T^* in inorder and define G^* by placing an edge (i, t) whenever t is an ancestor of i and i lies in the left subtree of t. Thus $d_{G^*}(i) = \ell_{T^*}(i)$. To show that $\alpha(G^*) \le w$, process the vertices in inorder, merging at round t all live components contained in the left subtree of t with the new vertex t. These merges induce exactly the edges of G^*. Let $W(T)$ denote the maximum number of live components when processing a tree T, starting with no components. If T_L and T_R are the left and right subtrees of its root, then

$$W(T) \le \max\{W(T_L), 1 + W(T_R)\}.$$

Since $W(\varnothing) = 0$, induction gives $W(T) \le 1 + \max_i \rho_T(i)$, where $\rho_T(i)$ is the number of right edges on the path to i. Hence the merger schedule for T^* has at most w live components. Any unblocked sequence would have all its vertices in distinct live components after its last insertion, so $\alpha(G^*) \le w$.

For every feasible graph G, the tree constructed in the first paragraph satisfies $\ell_T(i) \le d_G(i)$. The bounds on the numbers of nodes with $\ell_T(i) \ge d$, together with their simultaneous attainment by T^*, therefore show that G^* minimizes both $\max_i d_G(i)$ and $\sum_i d_G(i)$, with values Eqs. (30) and (31). The tree T^* and the graph G^* can be constructed in polynomial time. □

6.1.1 Maximum Number of Updates

For the parameter bounds in this and the next subsection, we take $n = \lambda^k$ for a fixed constant $k > 0$. For most upper bounds, we use an *ordinary CRHF*: a keyed collision-resistant hash family with $O(s)$-bit outputs at security parameter s, instantiated with $s = \lceil \lambda^\varepsilon \rceil$ for any fixed $\varepsilon > 0$. In certain parameter regimes, however, matching the lower bounds requires shorter hash outputs. We therefore also consider a *short-output CRHF* with output length $\sigma = \lceil (\log \lambda)(\log\log \lambda) \rceil$ that remains collision resistant against polynomial-time adversaries in λ.

By Lemma 6.1, the minimum maximum-update cost among graphs with $\alpha(G) \le w$ is $q(n, w)$, attained by G^*. To obtain bounds for a given accumulator length B, we apply Sections 4 and 5 to determine the corresponding values of w, and then evaluate $q(n, w)$. For example, when $n = \lambda^k$ and $B = \lambda^a$, the relevant values of w have polynomial order λ^a up to smaller factors. Since $\binom{w+d}{d} = \Theta(w^d)$ for every fixed d, comparing λ^{ad} with λ^k gives the optimum $\lfloor k/a \rfloor$.

The resulting bounds are summarized in Table 1. Here $a, k > 0$, $b \geq 0$, and $0 < \delta < 1$ are fixed constants. The last column indicates whether the stated upper bound requires the short-output CRHF assumption.

Parameter Regime	**Lower Bound**	**Upper Bound**	**Short CRHF?**
$n = \lambda^k$, $B = \lambda^a$	$\lfloor k/a \rfloor$	$\lfloor k/a \rfloor$	No
$n = \lambda^k$, $B = \lambda^a (\log n)^b$, $b > 0$, $k/a \notin \mathbb{Z}_{>0}$	$\lfloor k/a \rfloor$	$\lfloor k/a \rfloor$	No
$n = \lambda^k$, $B = \lambda^a (\log n)^b$, $k/a \in \mathbb{Z}_{>0}$, $0 < b \leq 1$	k/a	k/a	No
$n = \lambda^k$, $B = \lambda^a (\log n)^b$, $k/a \in \mathbb{Z}_{>0}$, $b > 1$	$k/a - 1$	k/a	No
$n = \lambda^k$, $B = \lambda^a (\log n)^b$, $k/a \in \mathbb{Z}_{>0}$, $b > 1$	$k/a - 1$	$k/a - 1$	Yes
$n = \lambda^k$, $B = \lambda^a n^\delta$	$\left\lfloor \frac{k}{a+k\delta} \right\rfloor$	$\left\lfloor \frac{k}{a+k\delta} \right\rfloor$	No
$n = \lambda^k$, $B = (\log n)^b$, $b > 2$	$\frac{(1-o(1)) \log n}{(b-2) \log \log n}$	$\frac{(1+o(1)) \log n}{(b-2) \log \log n}$	Yes

Table 1: Bounds on the maximum number of updates per witness under different accumulator-length bounds B. The last column indicates whether the stated upper bound requires a short-output CRHF; otherwise, an ordinary CRHF suffices. Lower bounds hold with overwhelming probability, and upper bounds hold for every execution.

The bounds match except when k/a is a positive integer and $B = \lambda^a (\log n)^b$ with $b > 1$ under the ordinary CRHF assumption. In this case, the lower bound is $k/a - 1$, while the ordinary-CRHF construction gives k/a. Short-output CRHFs allow the construction to attain the lower bound. For $B = \lambda^a n^\delta$, the same calculation applies with a replaced by $a + k\delta$, since $n = \lambda^k$. When $B = (\log n)^b$ with $b > 2$, evaluating $q(n, w)$ gives $(1 + o(1)) \log n / ((b-2) \log \log n)$. The bound $B = \lambda^a n^\delta$ is sublinear in n when $a + k\delta < k$.

6.1.2 Total Number of Updates

By Lemma 6.1, the minimum total-update cost among graphs with $\alpha(G) \leq w$ is $\Phi_w(n)$, attained by the same graph G^*. Summing the binomial coefficients gives

$$\Phi_w(n) = q(n, w)(n + 1) - \binom{w + q(n, w) + 1}{q(n, w)} + 1. \tag{34}$$

As in the maximum-update case, we apply Sections 4 and 5 to determine the relevant values of w and evaluate $\Phi_w(n)$. This identity allows us to evaluate the total-update bounds using the same graph parameters as in the maximum-update case.

The resulting bounds are summarized in Table 2. Here $a, k > 0$, $b \geq 0$, and $0 < \delta < 1$ are fixed constants. The last column indicates whether the stated upper bound requires the short-output CRHF assumption.

The bounds match asymptotically except when k/a is a positive integer and $B = \lambda^a (\log n)^b$ with $b > 1$ under the ordinary CRHF assumption. In this case, the lower bound is $(k/a - 1 - o(1))n$, while the ordinary-CRHF construction gives $(k/a + o(1))n$ updates. A short-output CRHF closes this gap. At the boundary $b = 1$, the lower bound $(k/a - o(1))n$ means that, for every fixed $\eta > 0$, the bound $(k/a - \eta)n$ holds with overwhelming probability.

For example, when $n = \lambda^6$ and $B = \lambda^2$, both the lower bound and the ordinary-CRHF construction give $(3 + o(1))n$ total updates. When $B = \lambda^2 (\log n)^2$, the lower bound becomes $(2 - o(1))n$. The short-output CRHF construction achieves $(2 + o(1))n$, whereas the ordinary-CRHF

Parameter Regime	Lower Bound	Upper Bound	Short CRHF?
$n = \lambda^k$, $B = \lambda^a$	$(\lfloor k/a \rfloor - o(1))n$	$(\lfloor k/a \rfloor + o(1))n$	No
$n = \lambda^k$, $B = \lambda^a (\log n)^b$, $b > 0$, $k/a \notin \mathbb{Z}_{>0}$	$(\lfloor k/a \rfloor - o(1))n$	$(\lfloor k/a \rfloor + o(1))n$	No
$n = \lambda^k$, $B = \lambda^a (\log n)^b$, $k/a \in \mathbb{Z}_{>0}$, $0 < b \leq 1$	$(k/a - o(1))n$	$(k/a + o(1))n$	No
$n = \lambda^k$, $B = \lambda^a (\log n)^b$, $k/a \in \mathbb{Z}_{>0}$, $b > 1$	$(k/a - 1 - o(1))n$	$(k/a + o(1))n$	No
$n = \lambda^k$, $B = \lambda^a (\log n)^b$, $k/a \in \mathbb{Z}_{>0}$, $b > 1$	$(k/a - 1 - o(1))n$	$(k/a - 1 + o(1))n$	Yes
$n = \lambda^k$, $B = \lambda^a n^\delta$	$\left(\left\lfloor \frac{k}{a+k\delta} \right\rfloor - o(1)\right) n$	$\left(\left\lfloor \frac{k}{a+k\delta} \right\rfloor + o(1)\right) n$	No
$n = \lambda^k$, $B = (\log n)^b$, $b > 2$	$\frac{(1-o(1))n \log n}{(b-2) \log \log n}$	$\frac{(1+o(1))n \log n}{(b-2) \log \log n}$	Yes

Table 2: Bounds on the total number of witness updates under different accumulator-length bounds B. The last column indicates whether the stated upper bound requires a short-output CRHF; otherwise, an ordinary CRHF suffices. Lower bounds hold with overwhelming probability, and upper bounds hold for every execution.

construction gives $(3 + o(1))n$. If the leading coefficient in a matching polynomial regime is zero, the optimum is exactly zero.

6.2 Heterogeneous Update Costs

We now allow different witness updates to incur different communication costs. We first consider costs depending only on value insertion times, for which we give a polynomial-time algorithm that computes the exact optimum and constructs an optimal update graph for arbitrary nonnegative weights. We then consider costs depending on both insertion and update times, for which we give a polynomial-time approximation algorithm for arbitrary nonnegative weights and an exact polynomial-time algorithm when the costs are nondecreasing over time. Unlike the maximum and total numbers of updates, the resulting optima depend on the specified weights and do not generally admit uniform formulas in terms of n and w alone. We therefore present general algorithms rather than parameter tables for particular weight distributions. We may assume $1 \leq w \leq n$, since the edgeless graph is feasible when $w \geq n$.

6.2.1 Insertion-Time-Dependent Costs

Let $a_i \geq 0$ be the communication cost of updating the witness for the value inserted at round i, and define

$$C_a(G) := \sum_{i=1}^{n} a_i d_G(i). \tag{35}$$

The total-update cost is the special case $a_i = 1$ for every i.

Proposition 6.2. *For nonnegative weights $a_1, \ldots, a_n$, let $V_w(p, q)$ denote the minimum of $\sum_{i=p}^{q} a_i \ell_T(i)$ over ordered binary trees T on $[p, q]$ with fewer than w right edges on every root-to-node path. Then*

$$V_w(p, q) = \min_{v \in [p,q]} \left(V_w(p, v-1) + \sum_{i=p}^{v-1} a_i + V_{w-1}(v+1, q) \right), \tag{36}$$

where $V_w(p,q) = 0$ *for* $p > q$ *and* $V_0(p,q) = +\infty$ *for* $p \le q$*. Moreover,*

$$\min_{G:\alpha(G)\le w} C_a(G) = V_w(1,n).$$

If the weights are rational numbers of polynomial bit length, an optimal update graph can be constructed in polynomial time using $O(wn^3)$ *arithmetic operations.*

Proof. By the tree characterization established in the proof of Lemma 6.1, every graph G with $\alpha(G) \le w$ admits an ordered binary tree T with fewer than w right edges on every root-to-node path and $\ell_T(i) \le d_G(i)$ for every i. Since the weights are nonnegative,

$$C_a(G) \ge \sum_{i=1}^{n} a_i \ell_T(i) \ge V_w(1,n).$$

Conversely, any such tree T determines a graph G_T with $\alpha(G_T) \le w$ and $d_{G_T}(i) = \ell_T(i)$, as shown in the same proof. Hence the minimum graph cost equals $V_w(1,n)$.

To compute this value, choose a root v for the tree on $[p,q]$. Each vertex in the left subtree gains one left edge, contributing $\sum_{i=p}^{v-1} a_i$, while the right subtree gains one right edge and therefore has right-depth bound $w-1$. The two subtrees can otherwise be optimized independently, giving Eq. (36). There are $O(wn^2)$ subproblems, each considering at most n roots. Recording the minimizing roots constructs an optimal tree and hence an optimal update graph. □

Thus arbitrary insertion-time-dependent weights admit an exact polynomial-time optimizer. Sections 4 and 5 turn the resulting values $V_w(1,n)$ into accumulator lower and upper bounds at the corresponding graph widths.

6.2.2 Insertion- and Update-Time-Dependent Costs

We next allow the communication cost to depend on both the insertion time i of a value and the round t at which its witness is updated. Let $a_{i,t} \ge 0$ be this cost, and define

$$C_a(G) := \sum_{1\le i<t\le n} a_{i,t}\mathbf{1}[(i,t) \in E(G)]. \tag{37}$$

This cost depends on the update times as well as on the out-degree vector; Eq. (35) is the special case $a_{i,t} = a_i$.

For arbitrary nonnegative weights, the exact optimum is given by the following integer program: minimize $\sum_{i<t} a_{i,t} z_{i,t}$ over variables $z_{i,t} \in \{0,1\}$, for $i < t$, subject to

$$\sum_{r=1}^{w} \sum_{t=s_{r+1}}^{s_{w+1}} z_{s_r,t} \ge 1 \qquad \text{for every } s_1 < \cdots < s_{w+1}. \tag{38}$$

These constraints require every increasing $(w+1)$-tuple to be blocked, and hence are equivalent to $\alpha(G) \le w$. The integer program therefore gives the exact combinatorial optimum, but does not by itself yield a polynomial-time algorithm. We first show that arbitrary weights nevertheless admit a polynomial-time approximation.

Proposition 6.3. *For arbitrary nonnegative rational weights* $a_{i,t}$ *of polynomial bit length and every integer* $1 \le w \le n$*, there is a polynomial-time algorithm that constructs a graph* G *satisfying*

$$\alpha(G) \le w, \qquad C_a(G) \le w \min_{H:\alpha(H)\le w} C_a(H).$$

Proof. Consider the linear programming relaxation of Eq. (38), replacing $z_{i,t} \in \{0,1\}$ with $0 \le x_{i,t} \le 1$. Let $\mathrm{LP}_w(a)$ denote its optimum.

The LP has a polynomial-time separation oracle. Given a candidate solution x, fix the last vertex v of a sequence and define $b_v(i,j) := \sum_{t=j}^{v} x_{i,t}$ for $i < j \le v$. The left-hand side of the constraint for $s_1 < \cdots < s_{w+1} = v$ is

$$\sum_{r=1}^{w} b_v(s_r, s_{r+1}).$$

Finding the minimum left-hand side is a shortest-path problem with exactly w edges in the acyclic graph on $[v]$, with edge weights $b_v(i,j)$. Dynamic programming solves this problem for each v; checking all terminal vertices gives a polynomial-time separation oracle and hence a polynomial-time LP algorithm [6].

Let x^* be an optimal LP solution. For each i, consider all intervals $[u,v] \subseteq \{i+1,\dots,n\}$ satisfying

$$\sum_{t=u}^{v} x^*_{i,t} \ge \frac{1}{w}.$$

Let $U_i \subseteq \{i+1,\dots,n\}$ be a minimum-cost set intersecting every such interval, where selecting t costs $a_{i,t}$. This minimum-weight interval stabbing problem is solvable in polynomial time: its constraint matrix has the consecutive-ones property and is totally unimodular.

The vector $y_{i,t} := \min\{1, w x^*_{i,t}\}$ is feasible for the interval stabbing LP. By integrality, the minimum-cost integer solution satisfies

$$\sum_{t \in U_i} a_{i,t} \le \sum_{t>i} a_{i,t} y_{i,t} \le w \sum_{t>i} a_{i,t} x^*_{i,t}.$$

Put $(i,t) \in E(G)$ exactly when $t \in U_i$. For every increasing $(w+1)$-tuple $s_1 < \cdots < s_{w+1}$, LP feasibility gives

$$\sum_{r=1}^{w} \sum_{t=s_{r+1}}^{s_{w+1}} x^*_{s_r,t} \ge 1.$$

Hence, for some r, the interval $[s_{r+1}, s_{w+1}]$ has fractional mass at least $1/w$ in row s_r. By construction, U_{s_r} intersects this interval, so the tuple is blocked by an edge of G. Thus $\alpha(G) \le w$.

Summing the costs over all rows gives

$$C_a(G) \le w\,\mathrm{LP}_w(a) \le w \min_{H:\alpha(H)\le w} C_a(H).$$

□

If the cost of updating any fixed witness is nondecreasing over time, the optimum can be computed exactly.

Proposition 6.4. *Suppose the nonnegative weights satisfy*

$$0 \le a_{i,i+1} \le a_{i,i+2} \le \cdots \le a_{i,n} \qquad \text{for every } i \in [n].$$

For an ordered binary tree T, let $L_T(i)$ be the set of ancestors t of i whose left subtree contains i, and define

$$C_a(T) := \sum_{i=1}^{n} \sum_{t \in L_T(i)} a_{i,t}.$$

Let $W_w(p,q)$ be the minimum tree cost over ordered binary trees on $[p,q]$ with fewer than w right edges on every root-to-node path. Then

$$W_w(p,q) = \min_{v \in [p,q]} \left(W_w(p, v-1) + \sum_{i=p}^{v-1} a_{i,v} + W_{w-1}(v+1, q) \right), \tag{39}$$

where $W_w(p,q) = 0$ *for* $p > q$ *and* $W_0(p,q) = +\infty$ *for* $p \le q$. *Moreover,*

$$\min_{G:\alpha(G)\le w} C_a(G) = W_w(1,n).$$

For rational weights of polynomial bit length, an optimal update graph can be constructed using $O(wn^3)$ *arithmetic operations.*

Proof. We first show that every graph G with $\alpha(G) \le w$ admits a tree T satisfying the stated right-depth bound and $C_a(T) \le C_a(G)$.

Consider the recursive tree construction in the proof of Lemma 6.1. For the current interval, it chooses the root $v = \min_i R_G(i)$, where $R_G(i)$ is the largest endpoint of an outgoing edge from i, or i if no such edge exists. For every $i < v$, the construction introduces the tree edge (i, v). Associate it with the edge $(i, R_G(i))$ of G, whose endpoint is at least v. The remaining tree edges are obtained recursively from the graphs induced by the left and right intervals. Their associated edges lie within those intervals, so the associations are distinct. By induction, this gives an injection from the edges of the tree graph G_T into the edges of G that preserves the source of each edge and never moves its endpoint earlier.

Since $a_{i,t}$ is nondecreasing in t, each tree edge has cost at most that of its associated edge in G. Therefore,

$$C_a(T) = C_a(G_T) \le C_a(G).$$

Conversely, every feasible tree T determines a graph G_T with $\alpha(G_T) \le w$ and exactly the edges (i,t) for which $t \in L_T(i)$. Thus the minimum graph cost equals the minimum tree cost.

Finally, fix the root v of a tree on $[p,q]$. Every vertex $i < v$ receives one additional update at round v, contributing $\sum_{i=p}^{v-1} a_{i,v}$. The right subtree has one fewer available right edge, while the remaining costs are determined independently within the two subtrees. This gives Eq. (39). As before, the dynamic program has $O(wn^2)$ subproblems and considers at most n roots per subproblem, yielding the claimed running time and an optimal graph. □

The results of Section 4 give lower bounds through the corresponding combinatorial optima, while Section 5 realizes the graphs produced by these algorithms when the required CRHF is available. The w-approximation guarantee applies to a fixed bound on $\alpha(G)$ and does not necessarily carry over to a fixed accumulator-length bound.

References

[1] Abusalah, H., Anthoine, G., Avitabile, G., Giunta, E.: Lower bounding update frequency in short accumulators and vector commitments. In: Advances in Cryptology – EUROCRYPT 2026. Lecture Notes in Computer Science, vol. 16545, pp. 184–213. Springer (2026)

[2] Benaloh, J., de Mare, M.: One-way accumulators: A decentralized alternative to digital signatures. In: Advances in Cryptology – EUROCRYPT '93. pp. 274–285. Springer (1994)

[3] Bonneau, J., Chen, J., Christ, M., Karantaidou, I.: Merkle mountain ranges are optimal: On the witness update frequency of cryptographic accumulators. In: Advances in Cryptology – CRYPTO 2025, Part II. Lecture Notes in Computer Science, vol. 16001, pp. 170–202. Springer (2025). https://doi.org/10.1007/978-3-032-01878-6_6

[4] Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Public Key Cryptography – PKC 2009. Lecture Notes in Computer Science, vol. 5443, pp. 481–500. Springer (2009). https://doi.org/10.1007/978-3-642-00468-1_27

[5] Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Advances in Cryptology – CRYPTO 2002. pp. 61–76. Springer (2002)

[6] Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, second corrected edn. (1993)

[7] He, H., Wu, Q., Qin, B., Gao, H., Susilo, W.: Privacy-preserving outsourced witness updates for append-only RSA accumulators. In: European Symposium on Research in Computer Security (ESORICS 2026) (2026), cryptology ePrint Archive, Report 2026/1283

[8] Mahmoody, M., Qi, W.: Online mergers and applications to registration-based encryption and accumulators. In: 4th Conference on Information-Theoretic Cryptography (ITC 2023). Leibniz International Proceedings in Informatics (LIPIcs), vol. 267, pp. 15:1–15:23 (2023). https://doi.org/10.4230/LIPIcs.ITC.2023.15, article 15

[9] Mahmoody, M., Qi, W., Rahimi, A.: Lower bounds for the number of decryption updates in registration-based encryption. In: Theory of Cryptography Conference (TCC 2022), Part I. Lecture Notes in Computer Science, vol. 13747, pp. 559–587. Springer (2022)

[10] Qi, W.: Tight lower bound on witness update frequency in additive positive accumulators. IACR Communications in Cryptology **2**(4) (2026). https://doi.org/10.62056/av7t7ta5v

[11] Soóki-Tóth, B., Glasz, B., Kavousi, A., Seres, I.A.: Private delegation of (non-)membership proof updates in cryptographic accumulators. Cryptology ePrint Archive, Report 2026/832 (2026)

[12] Tas, E.N., Boneh, D.: Vector commitments with efficient updates. In: 5th Conference on Advances in Financial Technologies (AFT 2023). Leibniz International Proceedings in Informatics (LIPIcs), vol. 282, pp. 29:1–29:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2023). https://doi.org/10.4230/LIPIcs.AFT.2023.29