Non-Delegatable Commitments

Georg Fuchsbauer[1] Pranav Garimidi[2] Guru-Vamsi Policharla[3] Max Resnick[4]
Ertem Nusret Tas[2]

[1]TU Wien
[2]a16z Crypto Research
[3]UC Berkeley
[4]Anza

Abstract

Cryptographic commitments allow a party to commit to a value such that it is infeasible to later open that commitment to a different value. Standard commitment schemes allow the committer to outsource both the generation of the commitment and the openings to a third party. This is benign for most use cases; however, when commitments serve as cryptographic attestations of work such as storing data or relaying blocks, participants can outsource the task and still claim credit, undermining the intended economic properties of the protocol. This work initiates the study of **non-delegatable commitments**, a new primitive where forming a commitment requires possession of a private key, and delegating the commitment process necessarily leaks that key. We formally define the primitive and provide a generic construction from a polynomial commitment scheme in the random oracle model. Additionally, we show how this primitive can be applied to solve a variety of mechanism design problems.

1 Introduction

Commitments are among the most fundamental primitives in cryptography: they allow a party to fix a hidden value now and reveal it later. An important use case for commitments is Proof of Data Possession, which allows a verifier to check that an untrusted server continues to store some file, without downloading the whole file [ABC+07, APMT08, CKBA08, JK07, SW13]. In this work, we are interested in a related application, where a verifier would like to succinctly check if a designated client called the *receiver* retrieved some data from a server called the *sender*.

Concretely, suppose a bandwidth-limited *weak* verifier wishes to ascertain whether a receiver $\mathbf{R}$ received some large file from a sender $\mathbf{S}$ in a timely manner, where the verifier has a succinct commitment to the file. Importantly, the verifier cannot ask the receiver to relay the entire file and check it against the commitment, because it is bandwidth-limited. Consider the following natural approach for achieving a low communication overhead for the verifier:

1. The sender $\mathbf{S}$ sends a subset of the file to the receiver $\mathbf{R}$.
2. The receiver produces a succinct (vector) commitment com to the received subset of the file and submits it to the verifier.
3. The verifier challenges $\mathbf{R}$ to open a few randomly chosen positions of the received file with respect to its commitment.
4. The verifier returns true if all of the openings provided by the receiver verify against the commitment.

Unfortunately this approach has a fatal flaw. If the sender and receiver collude, they can avoid sending the file altogether, and instead $\mathbf{S}$ can simply provide $\mathbf{R}$ with the commitment and the openings corresponding to the verifier's queries.

As the issue above implies, standard commitment schemes are *delegatable*: a designated committer can outsource the generation of the commitment and openings to a third party and simply relay these commitments and openings to the verifier. Although this is acceptable for most applications, delegatable commitments cannot be used to attest that some messages were received, or some work was performed by a designated party. To address this problem, we formally define and construct *non-delegatable* (vector) commitments, which are cryptographic vector commitments with the additional guarantee that a designated party cannot delegate the task of creating commitments (and openings) to another party without also leaking its secret key. Given such a primitive, we can attach some value to the secret key of the receiver (*e.g.*, the control of some funds locked in a smart contract), thereby providing an economic guarantee for data dissemination, as delegating the commitment to a third party would give that party control over the funds attached to the secret key.

More formally, let the data $\mathbf{m}$ be a finite ordered collection of blocks $\mathbf{m} = (m_1, \ldots, m_n)$, where each block $m_i \in \mathcal{M}$. $\mathbf{R}$ sends a non-delegatable commitment $\mathsf{com}_{\mathbf{R}}$ of the data it has received to a verifier that holds a succinct commitment com to $\mathbf{m}$ and wishes to check that $\mathbf{S}$ indeed disseminated $\mathbf{m}$ to $\mathbf{R}$ in a *timely* manner. For this purpose, the verifier challenges $\mathbf{R}$ to open k randomly chosen blocks of the data and concludes that $\mathbf{R}$ downloaded the data if the proofs of opening verify against $(\mathsf{com}, \mathsf{com}_{\mathbf{R}})$.

In Section 1.1, we highlight use-cases of non-delegatable commitments, where clients must perform some work or download some data without outsourcing. We also demonstrate how to combine non-delegatable commitments with a reward scheme to disincentivize outsourcing. In Section 2, we provide a technical overview of the primitive and the constructions, which are formalized and proven secure in the random oracle model in Sections 5 and 6 respectively. Section 7 gives the benchmarks for our implementation of non-delegatable commitments.

1.1 Applications

In this section, we sketch two examples where non-delegatable commitments help provide economic security guarantees for tasks in distributed protocols that previously only had indirect economic incentives. We emphasize that these applications are not meant to be exhaustive, but rather illustrative of the types of problems that non-delegatable commitments can help solve.

Block Propagation in Blockchains. In high-performance blockchains, networking costs are one of the most expensive parts of running a validator. For example, the current Solana validator requirement is at least 1 Gb/s Bandwidth with 10 Gb/s preferred [Sol24]. A large part of this network cost comes from block propagation, as the nodes must forward the blocks they hear to the rest of the network. Due to the large block size and number of nodes, the creator of the block may not always have the bandwidth itself to directly broadcast the blocks it produced to the *entire* network. Instead, the creator erasure-codes its block and sends shares to a randomly chosen set of nodes designated as relays, who are in turn responsible for broadcasting their shares (much smaller than a full block) to the entire network [Anz25]. This allows amortizing bandwidth resources across the network, so that every node receives enough shares to reconstruct the block as quickly as possible.

For this scheme to work, the block producer must broadcast all its shares to the relays *and* relays must broadcast their received shares to the rest of the network in a timely manner. However, there are currently no incentives to do so. While block creators are incentivized to broadcast their shares so that they can collect transaction fees from the confirmed block, relays have no such incentive. Thus, the relays may wish to save on bandwidth costs by not dispersing the received shares.

Reward Scheme for Relays. To resolve the issue above, we outline a reward scheme for relays that crucially relies on non-delegatable commitments. Suppose a relay $\mathbf{S}$ incurs a cost c per share m_i sent to some client $\mathbf{R}$. Here, $\mathbf{R}$ uses the non-delegatable commitment scheme to attest to the shares it has received from $\mathbf{S}$. Then, a challenge is issued by the trusted verifier (*e.g.*, a smart contract on the blockchain) that is in charge of dispensing rewards. If $\mathbf{R}$ successfully provides all k valid openings to the verifier's queries, then

both $\mathbf{S}$ and $\mathbf{R}$ are paid αr, where r is a fixed reward, and α is the ratio between how many shares $\mathbf{R}$ claimed to have received, and n, the total number of shares to be sent.

Note that $\mathbf{R}$ does not have the incentive to under-report what it received, since this only decreases its reward. If it claims to have heard more messages than it actually did, it risks being challenged on a missing opening and losing the entire reward for only a marginal gain otherwise. This is because $\mathbf{R}$ cannot delegate the task of creating openings on the missing shares to $\mathbf{S}$ without also leaking its secret key and thus potentially losing its funds p secured by that key. Therefore, $\mathbf{R}$ is incentivized to report exactly what it received, and $\mathbf{S}$ is in turn incentivized to send all n messages to maximize its reward as long as $r > \min(p, cn)$, where p is $\mathbf{R}$'s stake and c was $\mathbf{S}$'s cost per message disseminated.

Non-Delegatable Computation Our non-delegatable commitment scheme can be extended to achieve the functionality of "non-delegatable computation" which provides proof that a node carried out a computational task itself rather than delegating that task to another node. At a high level, a node runs a computational task and then makes a non-delegatable commitment to the trace of the computation, and engages with a verifier to prove its possession of the intermediate values of the computation. The verifier also checks if the revealed values are correct (*e.g.*, if the computation involves a circuit, it challenges k circuit gates and asks for the inputs and outputs to the k challenged gates, running each gate to see if the revealed values are correct). This solution does assume it is more expensive for nodes to download all (or most) of the intermediate values from a central node than to compute all the values themselves. This may be a reasonable assumption in e.g. LLM inference where communicating the entire trace may take terabytes for frontier open source models.

This functionality is especially useful for protocols involving *optimistic* verification. In optimistic verification, a set of nodes check that another node correctly executed a computational task by rerunning the computation themselves and checking whether the output matches. In the case that one of the nodes disagrees about the validity of the computation, a verifier resolves the dispute via a bisection game between the two parties' committed traces. However, often these schemes only reward nodes upon detecting a fault, as without a fault, there is no evidence that any of the verifying nodes did any work.

This leads to a challenge, as if the computation is always performed correctly, there is never any fault for nodes to detect, and hence nodes never get rewarded for the work they do. This causes such schemes to rely either on incentives external to the protocol for nodes to perform verification or on "attention challenges" where faulty work is intentionally done. Instead, we would like to reward nodes directly for the work they do in rerunning computations, even when the outcome is correct. Non-delegatable computation provides exactly this functionality. Optimistic verification schemes can require nodes to post non-delegatable commitments to the work they do in re-executing the computation. Then a verifier can be a smart contract issuing challenges to nodes to provide openings to their computational traces and reward them accordingly. This strengthens the economic security of optimistic verification protocols; so that all the verifying nodes can be expected to faithfully work without external incentives.

2 Technical Overview

We provide an overview of how we define non-delegatable commitments (Section 5) and our constructions (Section 6) in the random oracle model.

A non-delegatable commitment scheme is a cryptographic primitive, where creating a commitment requires knowledge of the message being committed and a *secret key*. Verification of openings is done using the corresponding public key. Non-delegatable commitments satisfy standard notions of binding and hiding, but in addition they are *non-delegatable* in that it is computationally infeasible to *delegate* the creation of the commitment without also leaking the secret key. In practice, by appropriately attaching a *monetary value* to the secret key (say, by staking funds in a smart contract), we can provide an *economic* guarantee that any party creating a commitment was indeed in possession of the underlying message.

Impossibility. Clearly, non-delegatable commitments are impossible in the standard model as any functionality of the secret key can be delegated (even to an untrusted party), with full privacy, using fully homomorphic encryption [Gen09]. But no such lower bound applies to building *non-delegatable* primitives in idealized models such as the Random Oracle Model or the Generic/Algebraic Group Model [Sho97, Mau05, FKL18]. Of course, to actually deploy such a protocol, the idealized primitives would need to be *heuristically* instantiated, in which delegation is possible again, seemingly making the whole exercise pointless!

However, there is an interesting statement to be made here about the *efficiency* of such a delegation. Assuming that we prove an appropriate notion of "non-delegatability" in an idealized model (say the random oracle model), any successful delegation must inherently make *non-blackbox* use of the cryptographic hash function that is used to heuristically instantiate the random oracle. In the case of non-delegatable commitments, the delegator and delegatees must engage in a secure two-party computation protocol to evaluate a function involving (potentially many) cryptographic hash function calls. For a non-delegatable commitment to be meaningful in applications such as those outlined in Section 1.1, it is sufficient to make the economic cost of delegation higher than the cost of actually carrying out the work of data dissemination. Moreover, one can even design hash functions that are inherently MPC-, FHE- and TEE-*unfriendly* and thus hard to delegate, making it substantially more expensive than to "just do the work yourself".

Thus, a non-delegatable commitment scheme in an idealized model with *minimal overhead* for honest parties is valuable, so long as the best possible delegation protocol (for the heuristic instantiation) is more expensive than carrying out the commitment honestly.

In this work, we focus on vector commitment schemes as they are most relevant for the applications outlined in Section 1.1, but other variants such as non-delegatablepolynomial commitment schemes are an interesting direction for future work.

Definition. A non-delegatable commitment scheme is a commitment scheme, satisfying the standard notions of binding (and optionally hiding).[1] We demand two additional properties:

- **Non-Delegatability:** it must be computationally infeasible for an adversary to "delegate" creation of a commitment without leaking its secret key.
- **Key-Hiding:** an honestly generated commitment and opening should not leak any information about the secret key. Note that this does not imply any privacy guarantees for the message.

The key-hiding property is modeled in a natural manner via a simulator that can produce a computationally indistinguishable transcript of the commitment and few openings at any given point of the vector without knowledge of the secret key.

Non-delegatability is modeled via a $\mu(k)$-soundness game (Definition 8) between a challenger and an adversary $\mathcal{A} = (\mathbf{S}_1, \ldots, \mathbf{S}_\ell, \mathbf{R}_1, \mathbf{R}_2)$, consisting of a set of senders and a (two-stage) receiver. To start, $\mathbf{R}_1$ outputs a public key pk. The adversary's goal is to have $\mathbf{R}_1$ produce a valid commitment com, but by delegating it to $\mathbf{S}_1, \ldots, \mathbf{S}_\ell$. $\mathbf{R}_1$ is allowed to interact with $\mathbf{S}_j$'s, but it must do so without leaking its secret key sk. At the end of this interaction, $\mathbf{S}_j$'s output the message $m \in \mathcal{M}^n$ that they claim was committed, and $\mathbf{R}_1$ outputs com and a state st. After the commitment is created, $\mathbf{R}_2$ is tasked with producing valid opening proofs on k randomly chosen points. $\mathbf{R}_2$ can interact with $\mathbf{S}_j$'s and receives st as input, and could thus know the entire message m.

We say that a commitment scheme is non-delegatable if for every $\mathcal{A}$, whenever it succeeds in the above game, one of the following extractors must succeed (except with negligible probability):

- $\mathsf{Ext}^{\mathbf{R}_1}$: given access to $\mathbf{R}_1$'s random oracle queries, the transcript of the communication between $\mathbf{S}_1, \ldots, \mathbf{S}_\ell$ and $\mathbf{R}_1$, as well as the commitment com, it outputs at least $(1 - \mu(k))$ fraction of the message m.
- $\mathsf{Ext}^{\mathbf{S}_j}$: given access to $\mathbf{S}_j$'s random oracle queries, its communication with $\mathbf{R}_1$ and $\mathbf{R}_2$, the commitment com, and the final opening proofs, it outputs $\mathbf{R}$'s secret key sk.

[1] Commitment schemes with succinct commitments and openings are interesting even without the hiding property.

This thus captures the fact that either $\mathbf{R}_1$ knew (most of) the message, or one of the $\mathbf{S}_j$ learned the secret key. We formally define non-delegatable (vector) commitments in Section 5. Before describing our construction, we provide justification for various moving parts in the above security game:

- **Timeliness.** Observe that $\mathsf{Ext}^{\mathbf{R}_1}$ is intentionally NOT given access to the random oracles queries made by $\mathbf{R}_2$. In order to succeed it must extract the message before the adversary even responds to the challenger's opening queries.[2]

  This simple omission critically captures *timeliness* guarantees: If a party successfully passes the challenger's queries, then it must have known the underlying data at the time it created the commitment. Allowing access to $\mathbf{R}_2$'s random oracle queries degrades this guarantee to "eventually" obtaining the data.

- **Key Compromise.** Note that the extractor $\mathsf{Ext}^{\mathbf{S}_j}$ is allowed access to the random oracle queries made by $\mathbf{S}_j$ over the entire game as the threat of *eventually* leaking the secret key is sufficient to deter delegation. Also observe that because $\mathsf{Ext}^{\mathbf{S}_j}$ is NOT given access to the random oracle queries made by $\{\mathbf{S}_i\}_{i\neq j}$ the secret key is compromised and leaked to a single party $\mathbf{S}_j$ – not secret shared amongst a committee.

Construction ideas. We build a non-delegatable commitment in the random oracle model, given a polynomial commitment scheme PC, and a vector commitment scheme VC.[3]

Our construction will crucially leverage the fact that queries to the random oracle must be made explicitly by a *single* party. Consider the following (obviously insecure) scheme. Given a message $m \in \mathcal{M}^n$, commit to it using a *hiding* Merkle tree, where the leaf nodes are $\{(m_i, \mathsf{sk})\}_{i\in[n]}$ and the hash function is modeled as a random oracle. To open it at a position $i \in [n]$, provide the leaf node and the root to the leaf path.

In the $\mu(k)$-soundness game above, for any leaf node that can be successfully opened, the query (m_i, sk) must have been made by either some $\mathbf{S}_j$, or $\mathbf{R}_1$. If they were not made by either, then that position cannot be opened, and if there are too many of these, $\mathbf{R}_2$ will not be able to pass the challenge phase. Thus, a large fraction must be queried. Clearly, if some $\mathbf{S}_j$ makes even one query, then the secret key is leaked. Therefore, $\mathbf{R}_1$ must be the one to make all of them and hence "knows" a large fraction of the message. While this seems to achieve non-delegatability, it is obviously not key-hiding.

So instead of placing the secret key sk directly in all of the leaf nodes, we secret-share it. Concretely, let $f(X)$ be a random degree-k polynomial conditioned on $f(0) = \mathsf{sk}$ and set the leaf nodes to $\{(m_i, f(i))\}_{i\in[n]}$. Then, we can reveal k points in an honestly generated commitment (as done in the challenge phase) without leaking any information about the secret key. However, an adversary is not forced to use a valid secret sharing, and can insert arbitrary values. We therefore additionally commit to $f(X)$ using a hiding polynomial commitment scheme and add a zero-knowledge proof that $f(0) = \mathsf{sk}$. The verifier then demands openings of the Merkle commitment to (m_i, y_i), as well as an opening of the polynomial commitment to y_i at position i.

Extraction Issues. It seems like we are essentially done since, if any $\mathbf{S}_j$ makes any leaf node query different from the k openings of the Merkle tree leaves generated in response to a challenge, it learns the secret key by simply interpolating $f(X)$. Hence, $\mathbf{R}_1$ must still make a large fraction of the queries, and we should be able to recover the message from them. While this is technically true, it's not clear that there's an *efficient* way to recover the message.

In fact, here is an explicit strategy between some $\mathbf{S}_j$ and $\mathbf{R}_1$ that makes extraction difficult:

- $\mathbf{R}_1$ knows the message m, it secret-shares sk as described above and makes random oracle queries of the form $\{(m_i, f(i))\}_{i\in[n]}$ and $\{(\hat{m}_i, f(i))\}_{i\in[n]}$ in randomized order, for some arbitrary $\hat{m} \in \mathcal{M}^n$.
- $\mathbf{R}_1$ sends the answers to these queries to $\mathbf{S}_j$, which also knows the randomized order.

[2] Note that this immediately disallows rewinding-based strategies to recover the message by asking for openings at different locations.

[3] Although a polynomial commitment scheme implies a vector commitment scheme, we separate the two as vector commitments can be instantiated very efficiently using Merkle trees, and it will be committing to a substantially larger input than the polynomial commitment scheme.

- $\mathbf{S}_j$ then completes the commitment to m by computing all of the internal nodes of the Merkle tree and sends over the root to $\mathbf{R}_1$, who outputs it.
- $\mathbf{S}_j$ outputs m.
- $\mathbf{R}_2$ can interact with $\mathbf{S}_j$ to produce opening proofs and always pass the game.

Clearly $\mathsf{Ext}^{\mathbf{S}_j}$ cannot succeed. Even though $\mathsf{Ext}^{\mathbf{R}_1}$ can output a set of $2n$ tuples $\{(m_i, f(i))\}_{i\in[n]}$ and $\{(\hat{m}_i, f(i))\}_{i\in[n]}$ within which lies the message m, it's not clear how $\mathsf{Ext}^{\mathbf{R}_1}$ can successfully *identify* it. A brute-force search would require checking 2^n possible combinations of leaf nodes.

Disambiguation. We are almost at the finish line: we (extractor) have the message in our hands (among other things), and we just need a way to disambiguate between the real and dummy leaf nodes. We do so by creating a separate vector commitment $\mathsf{com}_{\mathsf{VC}}$ to the leaf nodes $\{(i, m_i)\}_{i\in[n]}$ (no secret key used) and embedding the opening proofs inside the Merkle tree such that the i-th leaf node is $(i, m_i, \pi_{\mathsf{VC},i}, f(i))$. By checking the opening proof $\pi_{\mathsf{VC},i}$ against the vector commitment, we are able to identify which nodes belong to m. Including $\mathsf{com}_{\mathsf{VC}}$ in the non-delegatable commitment also lets one use it as a standard commitment: defining an alternative opening algorithm that only reveals $\pi_{\mathsf{VC},i}$ allows for an unbounded number of openings without revealing the secret key.

Our constructions. In Section 6, we present a generic construction that can be instantiated with any vector commitment scheme and hiding polynomial commitment scheme. We then instantiate it using a hiding variant of KZG commitments [KZG10, KT24], which also lets us prove $f(0) = \mathsf{sk}$ very efficiently when key pairs are of the form $\mathsf{pk} = g^{\mathsf{sk}}$. Our instantiation is highly efficient for both the receiver (prover) and the verifier: an honest receiver only performs $O(k) \ll n$ elliptic-curve operations and commits to the message using a Merkle commitment, where k is the number of opened leaves and n is the size of the data. The verifier can check the proof using $O(k)$ elliptic-curve operations and $O(1)$ pairings. We provide an implementation of this scheme in Section 7, and our benchmarks corroborate the efficiency claim: it takes less than 2 seconds to commit to over $100,000$ messages and less than 5 ms to create an opening for 50 messages.

As a side note, it is possible to construct non-delegatable commitments that are plausibly post-quantum secure by instantiating the generic construction with vector and hiding polynomial commitment schemes that are post-quantum secure (*e.g.*, [NS24, WWW25]).

3 Related Work

There is a rich literature on *rational cryptography* [DHR00, HT04, ADGH06, KN08, FKN10, AL10, PH10, ACH11, GK12, GKM+13, BYK21, GKS22] that analyzes security in the presence of profit-maximizing ("rational") participants. A major challenge for such protocols is extending the notions of game-theoretic equilibria (such as subgame-perfect equilibrium used for repeated games) to the computational setting [GK06, Hal11, GLR13]. In this context, rather than defining security as a property of the game among rational parties, the *rational protocol design* framework [GKM+13, GKTZ15, BGM+18] focuses on a "meta-game" between a protocol designer, who specifies the protocol rules for the honest parties, and an attacker.

An intermediate step towards security with rational protocol participants is to ensure that if security is ever violated, then the responsible parties can be financially penalized. This notion enables decoupling the challenge of designing cryptographic protocols for strengthening the guarantees provided by mechanism design, and that of defining game-theoretic equilibria and achieving it in a computational setting (as in [DHR00, SCW23]). Some examples of this intermediate notion include accountable threshold signatures (*e.g.*, [BK22, BPR24b]), double-authentication-preventing signatures, and accountable assertions [PS14, BKN17, BPS17, CFS22, RKS15], which enable identifying a signer, or extracting its secret key if it creates two different signatures in the same *context*. Similarly, in the context of distributed computing protocols, accountability [BG17, SWN+21, BLR24] guarantees that if security is ever violated, then the responsible adversarial parties can be provably identified, and their escrowed stake can be burned. Although

accountability has been extended to contexts beyond signatures and consensus (*cf.* [CFN94, GSS21, Zha25, BPR24a, BPR24c, AADG25, BPR25]), these protocols often do not support public verification of the proof identifying the malicious parties, instead relying on a trusted entity with private state for this purpose.

Our protocol fits in the same "staked" adversary paradigm, with a notable difference: in the works cited above, even if multiple parties collude to violate security, once the violation happens, a large number of them can be identified and punished up to large coalition sizes. In contrast, we restrict our attention to a setting with two classes of parties and a verifier for detecting security violations, where if both types of parties fully collude, then providing security becomes *provably* impossible. Therefore, we model our adversary as two distrustful, computationally bounded entities, which can collude to breach security, yet, would like to hide their secret keys from each other. This definition and the achieving protocol comes with its own challenges that were summarized in Section 2.

Another technique to prevent collusion among "staked" parties is whistleblowing [KGP+25], where a subset of colluders create a cryptographic proof that reveals the adversarial behavior of their counterparts. However, these techniques come with limitations, as blockchain smart contracts can help colluders retaliate against whistleblowers, and such retaliation can only be prevented under a certain amount of minimum stake.

Related Constructions. Two closely related notions to non-delegatable commitments have been studied in prior work – proofs of "individual knowledge" [DFL23] and "complete knowledge" [KBD+24], which aim to guarantee that there is a single party that knows a secret key, even in the presence of a Trusted Execution Environment or Secure Multi-Party Computation committees, which allow controlled delegation of the secret key. The key insight in these works is that cryptographic hash functions are expensive to compute in MPC (in comparison to local evaluation) or in a TEE (in comparison to an ASIC). By appropriately designing the proof of knowledge of the secret key, they are able to make the cost of delegating the proof of knowledge to an MPC/TEE infeasible if there is an upper-bound on the number of rounds of communication allowed or the time in which the proof must be generated. We view these works as complementary to ours – while they focus on preventing delegation of the secret key, we design commitments that cannot be delegated without leaking the secret key. While one could treat the messages in our case as the secret key in [DFL23, KBD+24], as these schemes were designed for short secrets, their communication and computation costs scale with large messages, limiting their practicality.

Another related concept to non-delegatable commitments is leakage-deterring public-key cryptosystems [KT13], where some secret data is embedded in a public key so that leaking any cryptographic implementation making partial use of the corresponding secret key enables extracting the secret data. In a follow-up work [KT15], authors extend their model to the multi-party setting by designing traitor-deterring schemes, where any pirate decryption box constructed by colluding parties allows recovery of at least one party's collateral on a blockchain. Our constructions can be viewed as a concrete instantiation of leakage deterrence for vector commitments that enable extracting the full secret key without embedding any secret data in the public key.

Concurrent Work. In concurrent work [DFK+25], the authors independently introduce and construct *beholder signatures*, where parties sign commitments to long messages, with the guarantee that signatures can be produced only if the signer knows the entire committed message. More formally, they show that if a "distributed" adversary [DFL23] produces a valid signature on a commitment, then there exists some sub-adversary from which one can extract the secret key, and some fraction of the committed message. Non-delegatable commitments demand a slightly weaker property (which is nevertheless sufficient for applications), where the non-delegatability game is played by $(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$ and $\mathbf{R}$, where $\mathbf{R}$ has a public key pk and is instructed to create a commitment to some data m. If a commitment is produced and opened successfully, then one can extract a large fraction of the data from $\mathbf{R}$, or the secret key sk from one of $(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$. In particular, we do not demand that the secret key can be extracted from $\mathbf{R}$ (although our construction does satisfy this property).

In terms of concrete constructions, the two approaches are quite different and interesting in their own right. Fundamentally, they both use the fact that evaluating random oracles in a *distributed* setting is too expensive. [DFK+25] uses a variant of the Fischlin transform in order to extract the message and secret key, whereas we use a Merkle tree commitment combined with verifiable secret sharing in order to prevent delegation. In terms of asymptotic efficiency when committing to data of size n, [DFK+25] has total communication $O(\mathsf{poly}(\lambda))$ and uses $\tilde{O}(n)$ public key operations, whereas our work uses $O(\mathsf{poly}(\lambda, \log n))$ communication but only requires $\tilde{O}(\lambda)$ public-key operations. We provide a comparison of the benchmarks for both schemes in Section 7. Due to the difference in the number of public key operations, the time to create an opening proof for non-delegatable commitments is over $100\times$ smaller than the time to generate beholder signatures for the same security guarantee.

4 Preliminaries

Let $\mathsf{e}\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ denote a type-III bilinear map, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$ are multiplicative groups of prime order p with generators g and h for $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We let $[n]$ denote the set $\{1, 2, \ldots, n\}$ for positive integers n. We denote a sequence $(x_{a_1}, \ldots, x_{a_\ell})$, where $a_1 < \ldots < a_\ell$, by $(x_{a_i})_{i\in[\ell]}$, and assume that knowledge of the indices is transferred along with the tuple (this matters when $(x_{a_1}, \ldots, x_{a_\ell}) \subset [n]$ for some $n > \ell$). Given a security parameter $\lambda \in \mathbb{Z}^+$, an event is said to happen with probability negligible in λ, denoted by $\mathsf{negl}(\lambda)$, if its probability, as a function of λ, is $\lambda^{-\omega(1)}$. A value is said to be succinct in n if it scales as $O(\mathrm{polylog}(n))$ as a function of n. By $y \leftarrow M(x_1, \ldots; r)$ we denote the operation of running algorithm M on inputs $x_1, \ldots$ and coins r and letting y denote the output. By $y \leftarrow M(x_1, \ldots)$, we denote letting $y \leftarrow M(x_1, \ldots; r)$ for random r. We use $\langle M, (M_1, \ldots, M_\ell)\rangle$ to denote the transcripts of communication between the Turing machine M and $M_1, \ldots, M_\ell$ respectively. In the random oracle model (ROM), Ext^M implies that an algorithm Ext has access to the random oracle queries of a Turing machine M.

Definition 1 (NIZK-AoK)**.** *A non-interactive zero-knowledge argument of knowledge, denoted by* NIZK*, for a relation $\mathcal{R}$ consists of three algorithms:*

- $\mathsf{NIZK.Setup}(1^\lambda, \mathcal{R}) \to \mathsf{crs}$: *The (randomized) setup algorithm takes a security parameter and a relation $\mathcal{R}$ as input, and outputs a common reference string* crs*.*
- $\mathsf{NIZK.Prove}(\mathsf{crs}, x, w) \to \pi$: *The (randomized) proving algorithm takes a* crs*, a problem instance x and a witness w as input, and outputs a zero-knowledge (ZK) proof π.*
- $\mathsf{NIZK.Verify}(\mathsf{crs}, x, \pi) \to 0/1$: *The (deterministic) verification algorithm takes a* crs*, a problem instance x and a proof π as input, and outputs 1 if the proof is accepted.*

A secure NIZK *satisfies the following properties:*

Completeness: *For all $\lambda \in \mathbb{N}$, all instances/witness pairs $(x, w) \in \mathcal{R}$,*

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w) \end{array}\right] = 1.$$

Knowledge Soundness: *For all PPT adversaries $\mathcal{A}$, there exists an expected-polynomial-time extractor* Ext *such that*

$$\Pr\left[\begin{array}{r} \mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 \\ \wedge\ (x, w) \notin \mathcal{R} \end{array} \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}) \\ (x, \pi) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs}) \\ w \leftarrow \mathsf{Ext}^{\mathcal{A}}(\mathsf{crs}, x, \pi) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Statistical (Computational) Zero-knowledge *(non-adaptive): There exists a PPT simulator $\mathcal{S}$ such that for all $\lambda \in \mathbb{N}$, for all $(x, w) \in \mathcal{R}$ and all unbounded (PPT) adversaries $\mathcal{A}$;*

$$\Pr\left[\begin{array}{l|l} \mathcal{A}(\mathsf{crs}_b, \pi_b) = b & \begin{array}{l} b \leftarrow \{0,1\} \\ \mathsf{crs}_0 \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}) \\ \pi_0 \leftarrow \mathsf{Prove}(\mathsf{crs}_0, x, w), \\ (\mathsf{crs}_1, \pi_1) \leftarrow \mathcal{S}(1^\lambda, x) \end{array} \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Definition 2 (Vector Commitments)**.** *A vector commitment (VC) scheme for message space $\mathcal{M}$ is defined by the following algorithms:*

- $\mathsf{VC.Setup}(1^\lambda, 1^n) \to \mathsf{crs}$: *The setup algorithm takes a security parameter λ and a vector length n as input and outputs a CRS, which is an implicit input to all subsequent algorithms.*
- $\mathsf{VC.Commit}((m_i)_{i\in[n]} \in \mathcal{M}^n) \to (\mathsf{com}_{\mathsf{VC}}, \mathsf{aux}_{\mathsf{VC}})$: *The (randomized) commitment algorithm takes a sequence of n messages from some set $\mathcal{M}$ as input and outputs a commitment $\mathsf{com}_{\mathsf{VC}}$ and auxiliary information $\mathsf{aux}_{\mathsf{VC}}$.*
- $\mathsf{VC.Open}(\mathsf{com}_{\mathsf{VC}}, \mathsf{aux}_{\mathsf{VC}}, i \in [n]) \to (m_i, \pi_{\mathsf{VC}})$: *The (randomized) opening algorithm takes $\mathsf{com}_{\mathsf{VC}}$, $\mathsf{aux}_{\mathsf{VC}}$ and an index i as input and outputs the message at index i along with a corresponding opening proof π_{VC}.*
- $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i, m_i, \pi_{\mathsf{VC}}) \to 0/1$: *The (deterministic) verification algorithm takes a commitment $\mathsf{com}_{\mathsf{VC}}$, an index $i \in [n]$, a claimed value m_i at index i, and an opening proof π_{VC} as input and outputs 1 if the proof verifies.*

A secure VC satisfies correctness, binding, position-binding and hiding.

VC Correctness: *For all $\lambda \in \mathbb{N}$, $n = \mathrm{poly}(\lambda)$, messages $(m_i)_{i\in[n]} \in \mathcal{M}^n$, and indices $i \in [n]$, it holds that*

$$\Pr\left[\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i, m_i, \pi_{\mathsf{VC}}) = 1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{VC.Setup}(1^\lambda, 1^n) \\ (\mathsf{com}_{\mathsf{VC}}, \mathsf{aux}_{\mathsf{VC}}) \leftarrow \mathsf{VC.Commit}((m_i)_{i\in[n]}) \\ (m_i, \pi_{\mathsf{VC}}) \leftarrow \mathsf{VC.Open}(\mathsf{com}_{\mathsf{VC}}, \mathsf{aux}_{\mathsf{VC}}, i) \end{array}\right] > 1 - \mathsf{negl}(\lambda)$$

VC Binding: *For all $\lambda \in \mathbb{N}$, $n = \mathrm{poly}(\lambda)$, and PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{l} \mathsf{VC.Commit}((m_i)_{i\in[n]}; r) \\ = \mathsf{VC.Commit}((m'_i)_{i\in[n]}; r') \\ \quad \wedge (m_i)_{i\in[n]} \neq (m'_i)_{i\in[n]} \end{array} \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{VC.Setup}(1^\lambda, 1^n) \\ \left((m_i)_{i\in[n]}, (m'_i)_{i\in[n]}, r, r'\right) \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

VC Position-Binding: *For all $\lambda \in \mathbb{N}$, $n = \mathrm{poly}(\lambda)$, and PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{l} \mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i, m', \pi'_{\mathsf{VC}}) = 1 \\ \wedge\, \mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i, m, \pi_{\mathsf{VC}}) = 1 \\ \qquad\qquad \wedge\, m \neq m' \end{array} \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{VC.Setup}(1^\lambda, 1^n) \\ (\mathsf{com}_{\mathsf{VC}}, i, m, m', \pi_{\mathsf{VC}}, \pi'_{\mathsf{VC}}) \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

VC Statistical (Computational) Hiding: *For all $\lambda \in \mathbb{N}$, $n = \mathrm{poly}(\lambda)$, proper subsets $S \subset [n]$ (of size $< n$), messages $(m^0_i)_{i\in[n]}, (m^1_i)_{i\in[n]} \in (\mathcal{M})^n$ such that $m^0_i = m^1_i$ for all $i \in S$, and unbounded (PPT) adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{l|l} \mathcal{A}(\mathsf{crs}, \mathsf{com}_b, (m^b_x, \pi^b_x)_{x\in S}) = b & \begin{array}{l} b \leftarrow \{0,1\} \\ \mathsf{crs} \leftarrow \mathsf{VC.Setup}(1^\lambda, 1^n) \\ (\mathsf{com}_0, \mathsf{aux}_0) \leftarrow \mathsf{VC.Commit}(\mathsf{crs}, (m^0_i)_{i\in[n]}) \\ \{(m^0_x, \pi^0_x) \leftarrow \mathsf{VC.Open}(\mathsf{com}_0, \mathsf{aux}_0, x)\}_{x\in S} \\ (\mathsf{com}_1, \mathsf{aux}_1) \leftarrow \mathsf{VC.Commit}(\mathsf{crs}, (m^1_i)_{i\in[n]}) \\ \{(m^1_x, \pi^1_x) \leftarrow \mathsf{VC.Open}(\mathsf{com}_1, \mathsf{aux}_1, x)\}_{x\in S} \end{array} \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Remark 1. *In our construction (Section 6), we use a Merkle tree, where each input message m_i is concatenated with a fresh, uniformly random value $r_i \leftarrow \{0,1\}^\lambda$. This construction satisfies binding, position-binding, and computational hiding, when the hash function is modeled as a random oracle. For this VC, we use the notation 'MT' in place of VC in the algorithms. In this case, there is no CRS, $\mathsf{aux}_{\mathsf{MT}}$ corresponds to the Merkle tree, the commitment is the Merkle root, and the opening proofs consist of the vector length n along with a subset of Merkle tree's inner nodes. The randomly sampled values r_i attached to the messages m_i are provided as part of the opening proofs π_i to enable verification.*

Definition 3 (Polynomial Commitments)**.** *A univariate polynomial commitment scheme* PC *is characterized by the following algorithms:*

- $\mathsf{PC.Setup}(1^\lambda, 1^d) \to \mathsf{crs}$: *The setup algorithm takes a security parameter and a degree bound d as input and outputs a CRS, which is an implicit input to all subsequent algorithms.* crs *specifies a finite field $\mathbb{Z}_p$ of prime order p.*
- $\mathsf{PC.Commit}(f(X) \in \mathbb{Z}_p^{\leq d}[X]) \to (\mathsf{com}_{\mathsf{PC}}, \mathsf{aux}_{\mathsf{PC}})$: *The (randomized) commitment algorithm takes a degree $\leq d$ polynomial $f(X)$ as input and outputs a commitment $\mathsf{com}_{\mathsf{PC}}$ and auxiliary information $\mathsf{aux}_{\mathsf{PC}}$.*
- $\mathsf{PC.Open}(\mathsf{com}_{\mathsf{PC}}, \mathsf{aux}_{\mathsf{PC}}, x \in \mathbb{Z}_p) \to (y \in \mathbb{Z}_p, \pi_{\mathsf{PC}})$: *The (randomized) opening algorithm takes $\mathsf{com}_{\mathsf{PC}}$, $\mathsf{aux}_{\mathsf{PC}}$ and an opening point $x \in \mathbb{Z}_p$ as input and outputs the opening value $y \in \mathbb{Z}_p$ and a proof π_{PC}.*
- $\mathsf{PC.Verify}(\mathsf{com}_{\mathsf{PC}}, x, y, \pi_{\mathsf{PC}}) \to 0/1$: *The (deterministic) verification algorithm takes a commitment $\mathsf{com}_{\mathsf{PC}}$, an opening point x, a claimed value y, and an opening proof π_{PC}; it outputs 1 if π_{PC} verifies with respect to $\mathsf{com}_{\mathsf{PC}}$, x, and y.*

A secure PC *satisfies correctness, binding, evaluation-binding and hiding. We additionally require the* PC *to be extractable as defined below.*

PC Correctness: *For all $\lambda \in \mathbb{N}$, $d = \mathrm{poly}(\lambda)$, polynomials $f(X) \in \mathbb{Z}_p^{\leq d}[X]$, and points $x \in \mathbb{Z}_p$, it holds that*

$$\Pr\left[\mathsf{PC.Verify}(\mathsf{com}_{\mathsf{PC}}, x, y, \pi_{\mathsf{PC}}) = 1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{PC.Setup}(1^\lambda, 1^d) \\ (\mathsf{com}_{\mathsf{PC}}, \mathsf{aux}_{\mathsf{PC}}) \leftarrow \mathsf{PC.Commit}(f(X)) \\ (y, \pi_{\mathsf{PC}}) \leftarrow \mathsf{PC.Open}(\mathsf{com}_{\mathsf{PC}}, \mathsf{aux}_{\mathsf{PC}}, x) \end{array}\right] > 1 - \mathsf{negl}(\lambda)$$

PC Binding: *For all $\lambda \in \mathbb{N}$, $d = \mathrm{poly}(\lambda)$, and PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{r} \mathsf{PC.Commit}(f(X); r) = \mathsf{PC.Commit}(f'(X); r') \\ \wedge\, f(X) \neq f'(X) \end{array} \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{PC.Setup}(1^\lambda, 1^d) \\ (f(X), f'(X), r, r') \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

PC Evaluation-Binding: *For all $\lambda \in \mathbb{N}$, $d = \mathrm{poly}(\lambda)$, and PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{r} \mathsf{PC.Verify}(\mathsf{com}_{\mathsf{PC}}, x, y', \pi'_{\mathsf{PC}}) = 1 \\ \wedge\, \mathsf{PC.Verify}(\mathsf{com}_{\mathsf{PC}}, x, y, \pi_{\mathsf{PC}}) = 1 \\ \wedge\, y \neq y' \end{array} \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{PC.Setup}(1^\lambda, 1^d) \\ \left(\mathsf{com}_{\mathsf{PC}}, x, y, y', \pi_{\mathsf{PC}}, \pi'_{\mathsf{PC}}\right) \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

PC Statistical (Computational) Hiding: *For all $\lambda \in \mathbb{N}$, $d = \mathrm{poly}(\lambda)$, subsets $S \subset \mathbb{Z}_p$ of size $\leq d$, polynomials $f_0(X), f_1(X) \in \mathbb{Z}_p^{\leq d}[X]$ such that $f_0(x) = f_1(x)$ for all $x \in S$, and unbounded (PPT) adversaries $\mathcal{A}$,*

$$\Pr\left[\mathcal{A}(\mathsf{crs}, \mathsf{com}_b, (f_b(x), \pi_x^b)_{x \in S}) = b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ \mathsf{crs} \leftarrow \mathsf{PC.Setup}(1^\lambda, 1^d) \\ (\mathsf{com}_0, \mathsf{aux}_0) \leftarrow \mathsf{PC.Commit}(f_0(X)) \\ \{(f_0(x), \pi_x^0) \leftarrow \mathsf{PC.Open}(\mathsf{com}_0, \mathsf{aux}_0, x)\}_{x \in S} \\ (\mathsf{com}_1, \mathsf{aux}_1) \leftarrow \mathsf{PC.Commit}(f_1(X)) \\ \{(f_1(x), \pi_x^1) \leftarrow \mathsf{PC.Open}(\mathsf{com}_1, \mathsf{aux}_1, x)\}_{x \in S} \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

PC Extractability: *The opening and verification algorithms of the PCS scheme constitute a non-interactive proof system for the language*

$$\left\{(\mathsf{com}_{\mathsf{PC}}, x, y) \;\middle|\; \exists\,(f(X), r) \in \mathbb{Z}_p^{\leq d} \times \{0,1\}^* : \mathsf{PC.Commit}(f(X); r) = \mathsf{com}_{\mathsf{PC}} \wedge f(x) = y\right\}$$

that satisfies knowledge soundness. Note that a PCS scheme satisfies evaluation-binding if it is binding and extractable.

4.1 Hiding and Extractable KZG Polynomial Commitments

We briefly recall a hiding and extractable variant of the KZG polynomial commitment scheme [KZG10, ZGK+17, KT24]. We also state algorithms that allow for opening (and verifying) many positions simultaneously.

- $\mathsf{KZG.Setup}(1^\lambda, 1^d) \to \mathsf{crs}$: The setup algorithm takes a security parameter λ and a degree bound d as input, generates a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \mathsf{e}, g, h)$, samples $\tau, \eta \leftarrow \mathbb{Z}_p$, and outputs the following as public parameters:
$$\left(g^\tau, g^{\tau^2}, \ldots, g^{\tau^d}, g^\eta, h^\tau, h^\eta\right) \in (\mathbb{G}_1)^{d+1} \times (\mathbb{G}_2)^2$$
- $\mathsf{KZG.Commit}(f(X)) \to (\mathsf{com}_{\mathsf{KZG}}, \mathsf{aux}_{\mathsf{KZG}})$: The commitment algorithm takes a polynomial $f(X) = \sum_{j=0}^{\deg(f)} f_j X^j$, with $\deg(f) \leq d$ as input, samples a random integer $r \leftarrow \mathbb{Z}_p$, and outputs $\mathsf{com}_{\mathsf{KZG}} = g^{f(\tau)+\eta r}$ and $\mathsf{aux}_{\mathsf{KZG}}$ computed as follows:
$$\mathsf{com}_{\mathsf{KZG}} = (g^\eta)^r \cdot \prod_{j=0}^{\deg(f)} (g^{\tau^j})^{f_j} \qquad \mathsf{aux}_{\mathsf{KZG}} = (f(X), r)$$
- $\mathsf{KZG.Open}(\mathsf{com}_{\mathsf{KZG}}, \mathsf{aux}_{\mathsf{KZG}}, x) \to (f(x), \pi_{\mathsf{KZG}})$: The opening algorithm takes $\mathsf{com}_{\mathsf{KZG}}$, and $\mathsf{aux}_{\mathsf{KZG}} = (f(X), r)$ as well as an opening point $x \in \mathbb{Z}_p$ as input, samples a random integer $s \leftarrow \mathbb{Z}_p$, and outputs the opening proof
$$\pi_{\mathsf{KZG}} := \left((g^\eta)^s \cdot g^{f_x(\tau)},\ g^{(r+sx)}/(g^\tau)^s\right),$$
where $f_x(X) := \frac{f(X)-f(x)}{X-x}$ is the quotient polynomial, and $g^{f_x(\tau)}$ is computed as shown above using the polynomial $f_x(X)$.
- $\mathsf{KZG.Verify}(\mathsf{com}_{\mathsf{KZG}}, x, y, \pi_{\mathsf{KZG}}) \to 0/1$: The verification algorithm takes a commitment $\mathsf{com}_{\mathsf{KZG}}$, an opening point $x \in \mathbb{Z}_p$, a claimed value $y \in \mathbb{Z}_p$, and an opening proof $\pi_{\mathsf{KZG}} := (\pi^1_{\mathsf{KZG}}, \pi^2_{\mathsf{KZG}}) \in (\mathbb{G}_1)^2$. It outputs 1 iff
$$\mathsf{e}(\mathsf{com}_{\mathsf{KZG}}/g^y, h) = \mathsf{e}(\pi^1_{\mathsf{KZG}}, h^\tau/h^x) \cdot \mathsf{e}(\pi^2_{\mathsf{KZG}}, h^\eta)$$
- $\mathsf{KZG.BatchOpen}(\mathsf{com}_{\mathsf{KZG}}, \mathsf{aux}_{\mathsf{KZG}}, (x_i)_{i\in[k]}) \to \left((f(x_i))_{i\in[k]}, \pi_{\mathsf{KZG}}\right)$: The batch opening algorithm takes $\mathsf{com}_{\mathsf{KZG}}$, $\mathsf{aux}_{\mathsf{KZG}} = (f(X), r)$ and a sequence of opening points $(x_i)_{i\in[k]} \in (\mathbb{Z}_p)^k$ as input, samples a random integer $s \leftarrow \mathbb{Z}_p$, and finds the degree-$(k-1)$ polynomial $r(X)$ such that $r(x_i) = f(x_i)$ for $i \in [k]$, and the degree-k polynomial $z(X) = \prod_{i\in[k]}(X - x_i)$. It then outputs the opening proof
$$\pi_{\mathsf{KZG}} := \left((g^\eta)^s \cdot g^{f_{(x_i)_{i\in[k]}}(\tau)}, g^r/(g^{z(\tau)})^s\right),$$
where $f_{(x_i)_{i\in[k]}}(X) := \frac{f(X)-r(X)}{\prod_{i\in[k]}(X-x_i)}$ is the quotient polynomial.
- $\mathsf{KZG.BatchVerify}(\mathsf{com}_{\mathsf{KZG}}, (x_i)_{i\in[k]}, (y_i)_{i\in[k]}, \pi_{\mathsf{KZG}}) \to 0/1$: The verification algorithm takes a commitment $\mathsf{com}_{\mathsf{KZG}}$, a sequence of opening points $(x_i)_{i\in[k]} \in (\mathbb{Z}_p)^k$, claimed values $(y_i)_{i\in[k]} \in (\mathbb{Z}_p)^k$, and an opening proof $\pi_{\mathsf{KZG}} := (\pi^1_{\mathsf{KZG}}, \pi^2_{\mathsf{KZG}})$. It finds the degree-$(k-1)$ polynomial $r(X)$ such that $r(x_i) = y_i$ for $i \in [k]$, and the degree-k polynomial $z(X) = \prod_{i\in[k]}(X - x_i)$. It outputs 1 iff
$$\mathsf{e}(\mathsf{com}_{\mathsf{KZG}}/g^{r(\tau)}, h) = \mathsf{e}(\pi^1_{\mathsf{KZG}}, h^{z(\tau)}) \cdot \mathsf{e}(\pi^2_{\mathsf{KZG}}, h^\eta)$$

Before we state the security guarantees of the scheme, we recap the q-Discrete-Logarithm assumption.

Definition 4. *The q-Discrete-Logarithm assumption, denoted by q-DLOG and parameterized by $q \in \mathbb{N}$, states that for any PPT adversary $\mathcal{A}$,*

$$\Pr\left[\tau' = \tau \;\middle|\; \tau \leftarrow \mathbb{Z}_p,\ \tau' \leftarrow \mathcal{A}\left(g, g^{\tau}, \ldots, g^{\tau^q}, h, h^{\tau}, \ldots, h^{\tau^q}\right)\right] \leq \mathsf{negl}(\lambda)$$

Extractability of the PCS scheme is shown in the Algebraic Group Model [FKL18], which assumes that any adversary that outputs a group element also outputs its representation as a linear combination of the group elements previously received. The following was proven in [KT24, Sections 3.5.3, B.2.2].

Theorem 1. *The PCS scheme above satisfies completeness, statistical hiding, binding under the d-DLOG assumption, and extractability under the $(d+1)$-DLOG assumption in the Algebraic Group Model.*

5 Definition

In this section, we formally define Non-Delegatable Commitments.

Definition 5 (Non-Delegatable Commitments)**.** *A non-delegatable commitment scheme is a tuple of five algorithms* $(\mathsf{KeyGen}, \mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$*, characterized by a key-pair relation $\mathcal{R}$, and message space $\mathcal{M}$:*

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: *The key generation algorithm takes as input a security parameter and outputs a key pair* $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{R}$*.*
- $\mathsf{Setup}(1^\lambda, 1^n, 1^k) \to \mathsf{crs}$: *The setup algorithm takes as input a security parameter, the message length n and a soundness parameter k, and outputs a common reference string, which is an implicit input to all remaining algorithms.*
- $\mathsf{Commit}(\mathsf{sk}, m) \to (\mathsf{com}, \mathsf{aux})$: *The commitment algorithm is a randomized algorithm that takes as input the secret key and a message $m \in \mathcal{M}^n$, and outputs a commitment* com*, along with auxiliary information* aux *that will be used to open the commitment.*
- $\mathsf{Open}(\mathsf{sk}, \mathsf{com}, \mathsf{aux}, i) \to (m_i, \pi)$: *The opening algorithm is a randomized algorithm that takes as input the secret key, a commitment, auxiliary information* aux *and an index $i \in [n]$, and outputs the i-th entry $m_i \in \mathcal{M}$ and a proof π.*
- $\mathsf{Verify}(\mathsf{pk}, \mathsf{com}, i, m_i, \pi) \to 0/1$: *The verification algorithm is a deterministic algorithm that takes as input a public key* pk*, commitment* com*, index $i \in [n]$, message m_i, and proof π, and outputs 1 if π is a valid opening proof for the message at the i-th position with respect to* com*.*

A non-delegatable commitment scheme satisfies correctness if for all $\lambda \in \mathbb{N}$, $n = \mathrm{poly}(\lambda)$, $k \leq n$ for all messages $m \in \mathcal{M}^n$ and all indices $i \in [n]$, we have:

$$\Pr\left[\mathsf{Verify}(\mathsf{pk}, \mathsf{com}, i, m_i, \pi) = 1 \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^k) \\ (\mathsf{com}, \mathsf{aux}) \leftarrow \mathsf{Commit}(\mathsf{sk}, m) \\ (m_i, \pi) \leftarrow \mathsf{Open}(\mathsf{sk}, \mathsf{com}, \mathsf{aux}, i) \end{array}\right] = 1$$

A non-delegatable commitment scheme is said to be secure if in addition to the standard notions of Binding and Position-Binding (Definition 6), it satisfies Non-Delegatability (Definition 8) and Key-Hiding (Definition 9). One can optionally demand the standard notion of hiding in commitments but it is not required in our applications.

Definition 6 (Binding and Position-Binding)**.** *A non-delegatable commitment scheme is said to be binding and position binding if for all PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{r|l} \mathsf{Commit}(\mathsf{sk}, m; r) = \mathsf{Commit}(\mathsf{sk}', m'; r') & \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^k) \\ \wedge\, m \neq m' & (\mathsf{sk}, \mathsf{sk}', m, m' \in \mathcal{M}^n, r, r') \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

$$\Pr\left[\begin{array}{r|l} \mathsf{Verify}(\mathsf{pk}, \mathsf{com}, i, m_i, \pi) = 1 & \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^k) \\ \wedge\, \mathsf{Verify}(\mathsf{pk}, \mathsf{com}, i, m_i', \pi') = 1 & (\mathsf{pk}, \mathsf{com}, i, m_i, \pi, m_i', \pi') \leftarrow \mathcal{A}(\mathsf{crs}) \\ \wedge\, m_i \neq m_i' & \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Definition 7 (Succinctness)**.** *Both* $|\mathsf{com}|$ *and* $|\pi|$ *are* $O(\mathsf{polylog}(n))$*, and the* Verify *algorithm runs in time* $O(\mathsf{polylog}(n))$*.*

Non-delegatability. This notion demands that it is computationally infeasible for an adversary to "delegate" the creation of a commitment without leaking its secret key. We formally model it as a game (Fig. 1) between a challenger and adversary $\mathcal{A} = (\mathbf{S}_1, \ldots, \mathbf{S}_\ell, \mathbf{R}_1, \mathbf{R}_2)$. To start, $(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$ collectively output a message $m \in \mathcal{M}^n$ and $\mathbf{R}_1$ outputs a public key pk. The goal is to have $\mathbf{R}_1$ produce a commitment com to m, without actually "knowing" (a large part of) the underlying message. $\mathbf{R}_1$ is allowed to coordinate with $(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$ but it must do so without leaking its secret key sk. After the commitment is created, $\mathbf{R}_1$ can pass arbitrary state to $\mathbf{R}_2$, which must successfully produce valid opening proofs for k randomly chosen points in order to pass the game. $\mathbf{R}_2$ is allowed to interact arbitrarily with $(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$ but again, must not leak the secret key sk.

To model these knowledge requirements more formally, we demand that one of the two must be true:

- **Data is Known** (at time of committing)**.** There exists an efficient extractor that is able to recover a sufficiently large fraction of the message, given only the transcript of the communication $\langle \mathbf{R}_1, (\mathbf{S}_1, \ldots, \mathbf{S}_\ell) \rangle$, along with any random oracle queries made by $\mathbf{R}_1$. This captures the case that the data is "known" to the receiver when it produces the commitment.
- **Key is Leaked.** There exists $j \in [\ell]$, such that an efficient extractor can recover the secret key corresponding to pk, given only the opening proofs and the transcripts of communication $\langle \mathbf{S}_j, \mathbf{R}_1 \rangle$ and $\langle \mathbf{S}_j, \mathbf{R}_2 \rangle$, along with any random oracle queries made by $\mathbf{S}_j$.

We remark that if the latter case is true, then even if $(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$ offered a secure MPC service to $(\mathbf{R}_1, \mathbf{R}_2)$, it must be that there was at least one party $\mathbf{S}_j$ that learned the secret key. If $(\mathbf{R}_1, \mathbf{R}_2)$ are rational and have put up some collateral against their public key, then it is not in their best interest to engage in such a protocol. Therefore, the former case must be true, and the data must have been "known" to $\mathbf{R}_1$ at the time of committing.

Definition 8 (Non-Delegatability)**.** *A non-delegatable commitment scheme is $\mu(k)$-sound, if for all sufficiently large n and all PPT $\mathcal{A} = (\mathbf{S}_1, \ldots, \mathbf{S}_\ell, \mathbf{R}_1, \mathbf{R}_2)$, there exists a polynomial-time extractor* Ext *such that $\mathcal{A}$ can win the $\mu(k)$-soundness game defined in Fig. 1 at most with negligible probability.*
(Recall that $\langle M, (M_1, \ldots, M_\ell) \rangle$ denotes the transcript between M and $(M_1, \ldots, M_\ell)$ and that Ext^M implies that Ext *has black-box access to the Turing machine M and to M's random oracle queries.)*

Key-hiding. This notion differs from the standard notion of *hiding* in that it only guarantees privacy for the committer's secret key but not for the committed message itself. In particular, it guarantees that, k opening proofs will (computationally) not reveal anything about the secret key.

Definition 9 (Key-Hiding)**.** *A non-delegatable commitment scheme is k-key-hiding if there exists a simulator $\mathcal{S}$ such that for all messages $m \in \mathcal{M}^n$, and all key pairs sampled as $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, the following distributions are computationally indistinguishable:*

$$\{\mathsf{pk}, \mathsf{crs}, \mathsf{com}, \mathsf{chal}, m, \{\pi_i\}_{i \in \mathsf{chal}}\} \approx_c \mathcal{S}(\mathsf{pk}, m)$$

where

$\mu(k)$-**Soundness Game** (in the ROM)

1. Run $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^k)$ and send crs to $\mathcal{A} = (\mathbf{S}_1, \ldots, \mathbf{S}_\ell, \mathbf{R}_1, \mathbf{R}_2)$.
2. $(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$ and $\mathbf{R}_1$ engage in an interactive protocol (during which they can query the random oracle) at the end of which
   - $(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$ output a message $m \in (\mathcal{M})^n$
   - $\mathbf{R}_1$ outputs a public key pk, a commitment com and state st
3. Sample a uniformly random k-sized subset $\mathsf{chal} \subseteq [n]$
4. $\mathbf{R}_2$, on input $(\mathsf{st}, \mathsf{pk}, \mathsf{com}, \mathsf{chal})$, engages in an interactive protocol with $(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$ (during which they can query the random oracle) at the end of which
   - $\mathbf{R}_2$ outputs $\{(m'_i, \pi_i)\}_{i \in \mathsf{chal}}$
5. Run the extractors
   - $m'' \leftarrow \mathsf{Ext}^{\mathbf{R}_1}(\mathsf{pk}, \mathsf{com}, \langle \mathbf{R}_1, (\mathbf{S}_1, \ldots, \mathbf{S}_\ell) \rangle)$
   - $\mathsf{sk}_j \leftarrow \mathsf{Ext}^{\mathbf{S}_j}(\mathsf{pk}, \mathsf{com}, \langle \mathbf{S}_j, \mathbf{R}_1 \rangle, \langle \mathbf{S}_j, \mathbf{R}_2 \rangle, \{(m'_i, \pi_i)\}_{i \in \mathsf{chal}})$ for all $j \in [\ell]$
6. Output 1 if all of the following hold:
   - $m'_i = m_i$ for all $i \in \mathsf{chal}$
   - $\mathsf{Verify}(\mathsf{pk}, \mathsf{com}, i, m_i, \pi_i) = 1$ for all $i \in \mathsf{chal}$
   - $\left|\left\{i \in [n] \mid m_i = m''_i\right\}\right| \leq \mu(k)n$
   - $(\mathsf{pk}, \mathsf{sk}_j) \notin \mathcal{R}$ for all $j \in [\ell]$

Figure 1: The soundness game for a non-delegatable commitment with adversary $\mathcal{A} = (\mathbf{S}_1, \ldots, \mathbf{S}_\ell, \mathbf{R}_1, \mathbf{R}_2)$.

- $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^k)$
- $(\mathsf{com}, \mathsf{aux}) \leftarrow \mathsf{Commit}(\mathsf{sk}, m)$
- chal *is a* k*-sized subset of* $[n]$ *sampled uniformly at random*
- $\{(m_i, \pi_i) \leftarrow \mathsf{Open}(\mathsf{sk}, \mathsf{com}, \mathsf{aux}, i)\}_{i \in \mathsf{chal}}$

Remark 2. *One could define a fully non-interactive scheme where the commitment always comes with a proof, consisting of openings at positions determined via the Fiat-Shamir transform. Indeed, by demanding that the challenge is uniform, we can always recover the non-interactive version. We opt for an interactive definition, as it allows mechanism designers to* dynamically *tune the security guarantee needed, by increasing/decreasing the number of queries made. For instance, one may choose the number of queries a party must reveal based on the amount of collateral they have put up – a larger loss may be a large deterrent for cheating and therefore can be coupled with a lower probability of getting caught. Explicitly allowing this flexibility can result in more efficient protocols.*

Our construction in Section 6 is a public-coin protocol, and can thus be made non-interactive using the Fiat-Shamir transformation.

6 Construction

In this section, we give two constructions of non-delegatable commitments in the random oracle model.

6.1 Generic Construction

Our first construction is generic over any vector commitment scheme VC and any hiding polynomial commitment scheme PC (not necessarily extractable) as defined in Section 4, and any efficiently samplable key-pair relation $\mathcal{R}$. For ease of presentation, we assume the witness space of $\mathcal{R}$ is contained in $\mathbb{Z}_p$ (the corresponding field of PC) and define the following relation:

$$\mathcal{R}_{\mathsf{NIZK}} = \left\{ \Big((\mathsf{pk}, \mathsf{com}_{\mathsf{PC}}), (\mathsf{sk}, f(X), r)\Big) \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \in \mathcal{R} \\ f(X) \in \mathbb{Z}_p^{\leq k}[X] \\ \mathsf{com}_{\mathsf{PC}} = \mathsf{PC.Commit}(f(X); r) \\ f(0) = \mathsf{sk} \end{array} \right\}$$

- $\mathsf{KeyGen}(1^\lambda)$:
  - sample and output $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{R}$.
- $\mathsf{Setup}(1^\lambda, 1^n, 1^k)$:
  - sample $\mathsf{crs}_{\mathsf{PC}} \leftarrow \mathsf{PC.Setup}(1^\lambda, 1^k)$
  - sample $\mathsf{crs}_{\mathsf{VC}} \leftarrow \mathsf{VC.Setup}(1^\lambda, 1^n)$
  - sample $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda, \mathcal{R}_{\mathsf{NIZK}})$
  - output $\mathsf{crs} := (\mathsf{crs}_{\mathsf{PC}}, \mathsf{crs}_{\mathsf{VC}}, \mathsf{crs}_{\mathsf{NIZK}})$.
- $\mathsf{Commit}(\mathsf{sk}, m)$:
  - commit to $m \in \mathcal{M}^n$ with VC and compute opening proofs for all points:

$$\begin{aligned} (\mathsf{com}_{\mathsf{VC}}, \mathsf{aux}_{\mathsf{VC}}) &\leftarrow \mathsf{VC.Commit}(m) \\ \big\{(m_i, \pi_{\mathsf{VC},i}) \leftarrow \mathsf{VC.Open}(\mathsf{com}_{\mathsf{VC}}, \mathsf{aux}_{\mathsf{VC}}, i)\big\}_{i\in[n]} & \end{aligned}$$

  - sample a degree-k polynomial $f(X) \in \mathbb{Z}_p^{\leq k}[X]$ with uniformly random coefficients, and set the constant term $f(0) = \mathsf{sk}$
  - commit to $f(X)$ with PC using randomness r:

$$(\mathsf{com}_{\mathsf{PC}}, \mathsf{aux}_{\mathsf{PC}}) \leftarrow \mathsf{PC.Commit}(f(X); r)$$

  - commit to the vector of tuples $L = \{(i, m_i, \pi_{\mathsf{VC},i}, f(i))\}_{i\in[n]}$ using a *hiding* Merkle commitment[4]:

$$(\mathsf{com}_{\mathsf{MT}}, \mathsf{aux}_{\mathsf{MT}}) \leftarrow \mathsf{MT.Commit}(L)$$

  - output $\mathsf{com} := (\mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{PC}}, \mathsf{com}_{\mathsf{VC}})$, and $\mathsf{aux} := (\mathsf{aux}_{\mathsf{PC}}, \mathsf{aux}_{\mathsf{MT}}, \mathsf{aux}_{\mathsf{VC}})$
- $\mathsf{Open}(\mathsf{sk}, \mathsf{com}, \mathsf{aux}, i) \to (m_i, \pi)$:
  - parse $\mathsf{com} = (\mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{PC}}, \mathsf{com}_{\mathsf{VC}})$ and $\mathsf{aux} = (\mathsf{aux}_{\mathsf{PC}}, \mathsf{aux}_{\mathsf{MT}}, \mathsf{aux}_{\mathsf{VC}})$.
  - compute $(f(i), \pi_{\mathsf{PC},i}) \leftarrow \mathsf{PC.Open}(\mathsf{com}_{\mathsf{PC}}, \mathsf{aux}_{\mathsf{PC}}, i)$
  - compute $\big((i, m_i, \pi_{\mathsf{VC},i}, f(i)), \pi_{\mathsf{MT},i}\big) \leftarrow \mathsf{MT.Open}(\mathsf{com}_{\mathsf{MT}}, \mathsf{aux}_{\mathsf{MT}}, i)$
  - generate a proof π_{NIZK} for the statement-witness pair $\big((\mathsf{pk}, \mathsf{com}_{\mathsf{PC}}), (\mathsf{sk}, f(X), r)\big)$.
  - output m_i and $\pi = (f(i), \pi_{\mathsf{PC},i}, \pi_{\mathsf{MT},i}, \pi_{\mathsf{VC},i}, \pi_{\mathsf{NIZK}})$
- $\mathsf{Verify}(\mathsf{pk}, \mathsf{com}, i, m_i, \pi)$:

[4]The hash function will be modeled as a random oracle

- parse $\mathsf{com} = (\mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{PC}}, \mathsf{com}_{\mathsf{VC}})$ and $\pi = (y_i, \pi_{\mathsf{PC},i}, \pi_{\mathsf{MT},i}, \pi_{\mathsf{VC},i}, \pi_{\mathsf{NIZK}})$
- check that $\mathsf{PC.Verify}(\mathsf{com}_{\mathsf{PC}}, i, y_i, \pi_{\mathsf{PC},i}) = 1$
- check that $\mathsf{MT.Verify}(\mathsf{com}_{\mathsf{MT}}, i, (i, m_i, \pi_{\mathsf{VC},i}, y_i), \pi_{\mathsf{MT},i}) = 1$
- check that $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i, m_i, \pi_{\mathsf{VC},i}) = 1$
- check that $\mathsf{NIZK.Verify}((\mathsf{pk}, \mathsf{com}_{\mathsf{PC}}), \pi_{\mathsf{NIZK}}) = 1$
- output 1 if all checks pass, else output 0

Remark 3. *Note that, following the definition of non-delegatable commitments (Definition 5),* Open *is part of the protocol to show that the commitment was computed by the "receiver", i.e., the holder of* pk*. The receiver will not reveal the output of* Open *for more than k positions, lest it leaks its secret key. Commitments from the above construction (and the concrete one in Section 6.2) can however also be used as "standard" commitments by simply opening the contained vector commitment* $\mathsf{com}_{\mathsf{VC}}$*, and there is no limit on the number of times this can be done.*

6.2 Concrete Instantiation

We next instantiate the construction with a variant of the KZG polynomial commitments. KZG commitments are extractable in the algebraic group model [FKL18] and can be made statistically hiding, as described in Section 4.1, so that opening proofs do not reveal anything else [KT24]. We denote this variant by KZG. Moreover, they have the property that opening proofs for a value y at position x can be verified given only g^y. Assuming the standard key-pair relation $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{R} \Leftrightarrow \mathsf{pk} = g^{\mathsf{sk}}$, we can thus replace the NIZK proof by a KZG opening proof at $x = 0$ and check that it verifies with respect to pk. Given that our construction is black-box in the vector commitment scheme, it can be instantiated with a Merkle tree (say), without affecting efficiency.

- $\mathsf{KeyGen}(1^\lambda)$:
  - sample $\mathsf{sk} \leftarrow \mathbb{Z}_p$.
  - set $\mathsf{pk}^1 = g^{\mathsf{sk}}$, $\mathsf{pk}^2 = h^{\mathsf{sk}}$, and $\mathsf{pk} = (\mathsf{pk}^1, \mathsf{pk}^2)$.
  - output $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Setup}(1^\lambda, 1^n, 1^k)$:
  - sample $\mathsf{crs}_{\mathsf{KZG}} \leftarrow \mathsf{KZG.Setup}(1^\lambda, 1^k)$
  - sample $\mathsf{crs}_{\mathsf{VC}} \leftarrow \mathsf{VC.Setup}(1^\lambda, 1^n)$
  - output $\mathsf{crs} := (\mathsf{crs}_{\mathsf{KZG}}, \mathsf{crs}_{\mathsf{VC}})$.
- $\mathsf{Commit}(\mathsf{sk}, m)$:
  - commit to $m \in \mathcal{M}^n$ with VC and compute opening proofs for all points:

$$(\mathsf{com}_{\mathsf{VC}}, \mathsf{aux}_{\mathsf{VC}}) \leftarrow \mathsf{VC.Commit}(m)$$
$$\left\{(m_i, \pi_{\mathsf{VC},i}) \leftarrow \mathsf{VC.Open}(\mathsf{com}_{\mathsf{VC}}, \mathsf{aux}_{\mathsf{VC}}, i)\right\}_{i\in[n]}$$

  - sample a degree-k polynomial $f(X) \in \mathbb{Z}_p^{\leq k}[X]$ with uniformly random coefficients, and set the constant term to $f(0) = \mathsf{sk}$
  - commit to $f(X)$ with KZG: $(\mathsf{com}_{\mathsf{KZG}}, \mathsf{aux}_{\mathsf{KZG}}) \leftarrow \mathsf{KZG.Commit}(f(X))$
  - commit to the vector of tuples $L = \{(i, m_i, \pi_{\mathsf{VC},i}, f(i))\}_{i\in[n]}$ using a *hiding* Merkle commitment:

$$(\mathsf{com}_{\mathsf{MT}}, \mathsf{aux}_{\mathsf{MT}}) \leftarrow \mathsf{MT.Commit}(L)$$

  - output $\mathsf{com} := (\mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{KZG}}, \mathsf{com}_{\mathsf{VC}})$ and $\mathsf{aux} := (\mathsf{aux}_{\mathsf{KZG}}, \mathsf{aux}_{\mathsf{MT}}, \mathsf{aux}_{\mathsf{VC}})$

- $\mathsf{Open}(\mathsf{sk}, \mathsf{com}, \mathsf{aux}, i) \to (m_i, \pi)$:
  - parse $\mathsf{com} = (\mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{KZG}}, \mathsf{com}_{\mathsf{VC}})$ and $\mathsf{aux} = (\mathsf{aux}_{\mathsf{KZG}}, \mathsf{aux}_{\mathsf{MT}}, \mathsf{aux}_{\mathsf{VC}})$
  - compute $(\mathsf{sk}, \pi_{\mathsf{KZG},0}) \leftarrow \mathsf{KZG.Open}(\mathsf{com}_{\mathsf{KZG}}, \mathsf{aux}_{\mathsf{KZG}}, 0)$
  - compute $(f(i), \pi_{\mathsf{KZG},i}) \leftarrow \mathsf{KZG.Open}(\mathsf{com}_{\mathsf{KZG}}, \mathsf{aux}_{\mathsf{KZG}}, i)$
  - compute $((i, m_i, \pi_{\mathsf{VC},i}, f(i)), \pi_{\mathsf{MT},i}) \leftarrow \mathsf{MT.Open}(\mathsf{com}_{\mathsf{MT}}, \mathsf{aux}_{\mathsf{MT}}, i)$
  - output m_i and $\pi = (f(i), \pi_{\mathsf{KZG},0}, \pi_{\mathsf{KZG},i}, \pi_{\mathsf{MT},i}, \pi_{\mathsf{VC},i})$
- $\mathsf{Verify}(\mathsf{pk}, \mathsf{com}, i, m_i, \pi)$:
  - check $\mathsf{e}(\mathsf{pk}^1, h) = \mathsf{e}(g, \mathsf{pk}^2)$.
  - parse $\mathsf{com} = (\mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{KZG}}, \mathsf{com}_{\mathsf{VC}})$, $\pi = (f(i), \pi_{\mathsf{KZG},0}, \pi_{\mathsf{KZG},i}, \pi_{\mathsf{MT},i}, \pi_{\mathsf{VC},i})$
  - parse $\pi_{\mathsf{KZG},0} := (\pi^1_{\mathsf{KZG},0}, \pi^2_{\mathsf{KZG},0})$, retrieve h^τ and h^η from $\mathsf{crs}_{\mathsf{KZG}}$, and check:

$$\mathsf{e}(\mathsf{com}_{\mathsf{KZG}}/\mathsf{pk}^1, h) = \mathsf{e}(\pi^1_{\mathsf{KZG},0}, h^\tau) \cdot \mathsf{e}(\pi^2_{\mathsf{KZG},0}, h^\eta)$$

  - check that $\mathsf{KZG.Verify}(\mathsf{com}_{\mathsf{KZG}}, i, f(i), \pi_{\mathsf{KZG},i}) = 1$
  - check that $\mathsf{MT.Verify}(\mathsf{com}_{\mathsf{MT}}, i, (i, m_i, \pi_{\mathsf{VC},i}, f(i)), \pi_{\mathsf{MT},i}) = 1$
  - check that $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i, m_i, \pi_{\mathsf{VC},i}) = 1$
  - output 1 if all checks pass, else output 0

Remark 4. *In practice, when the opening algorithm is invoked k times for the challenge indices, instead of returning a separate opening proof π_{KZG,x_i} for each challenge index $x_i \in \mathsf{chal}$, we can return a single batch opening proof for all of the k indices using the batch opening algorithm of the statistically hiding KZG scheme from Section 4.1: $\pi_{\mathsf{KZG}} \leftarrow \mathsf{KZG.BatchOpen}(\mathsf{com}_{\mathsf{KZG}}, \mathsf{aux}_{\mathsf{KZG}}, \mathsf{chal})$. This reduces the number of pairing operations for the verifier from $O(k)$ to $O(1)$.*

6.3 Security of our Constructions

Theorem 2. *If VC is a computationally position-binding vector commitment scheme, PC is a hiding and computationally evaluation-binding polynomial commitment scheme, and the NIZK scheme satisfies knowledge soundness and computational zero-knowledge, the construction in Section 6.1 is a secure non-delegatable commitment scheme with $\mu(k) = (2^{-\lambda/k} - k/n)$-soundness in the random oracle model.*

Theorem 3. *If VC is a computationally position-binding vector commitment scheme, and the $(k+1)$-DLOG assumption holds, then the construction in Section 6.2 is a secure non-delegatable commitment scheme with $\mu(k) = (2^{-\lambda/k} - k/n)$-soundness in the random oracle model against algebraic adversaries.*

To illustrate the concrete soundness guarantee, we show that our schemes require the receiver to obtain $\mu(k) = 2^{-\lambda/k} - k/n$ fraction of the messages from a sender to answer the queries (without revealing its secret key). For $\lambda = 128$, $k = \lambda$, and $n \gg k$, $\mu(k) \approx 1/2$; whereas for $k = 1000$ and $n = 1,000,000$ messages, $\mu(k) > 93\%$ with only $k/n = 0.1\%$ of the indices being challenged.

Remark 5. *Note that in Theorem 2, we set the number of openings to be the same as the security parameter λ of the polynomial to ensure $\mu(k)$-soundness with overwhelming probability. More generally, given a target 'rate of extraction' μ and $\ell \leq k$ openings selected uniformly at random, the non-delegatable commitment scheme achieves μ-soundness except with probability $\epsilon + \mathsf{negl}(\lambda)$, where $\epsilon = (1 - \mu - k/n)^\ell$. For $\mu = 1/2$, $\epsilon \approx 3.6 \times 10^{-5}$, $n = 2^{20}$, and $k = 64$, this implies $\ell = 15$ openings.*

Correctness. Correctness of both schemes follows from the correctness of the vector and polynomial commitment schemes and Merkle tree commitments, as well as completeness of the NIZK proof. The formal proof is presented in Appendix A.1.

Binding and Position-Binding. Binding and Position-Binding follow from the respective properties of the Merkle and polynomial commitments (*i.e.*, Binding and Evaluation-Binding for polynomial commitments).

Succinctness. If the vector and polynomial commitment schemes VC and PC have succinct commitments, opening proofs and verification time, then the generic construction in Section 6.1 satisfies succinctness. If the vector commitment scheme VC has succinct commitments, opening proofs and verification time, then the construction in Section 6.2 satisfies succinctness, as the KZG scheme has succinct commitments, evaluation proofs and verification time.

Key-Hiding. We first prove the key-hiding property for the generic construction.

Proof of Key-Hiding for the Construction in Section 6.1. We begin by describing the simulator $\mathcal{S}(\mathsf{pk}, m)$ which works as follows:

1. sample a uniformly random k-sized subset $\mathsf{chal} \subseteq [n]$
2. sample $\mathsf{crs}_{\mathsf{PC}} \leftarrow \mathsf{PC.Setup}(1^\lambda, 1^k)$ and $\mathsf{crs}_{\mathsf{VC}} \leftarrow \mathsf{VC.Setup}(1^\lambda, 1^n)$
3. commit to m using VC and compute opening proofs for all indices:
$$(\mathsf{com}_{\mathsf{VC}}, \mathsf{aux}_{\mathsf{VC}}) \leftarrow \mathsf{VC.Commit}(m)$$
$$\left\{(m_i, \pi_{\mathsf{VC},i}) \leftarrow \mathsf{VC.Open}(\mathsf{com}_{\mathsf{VC}}, \mathsf{aux}_{\mathsf{VC}}, i)\right\}_{i\in[n]}$$
4. sample a degree-k polynomial $f(X) \in \mathbb{Z}_p^{\leq k}[X]$ uniformly at random subject to $f(0) = 0$
5. compute $(\mathsf{com}_{\mathsf{PC}}, \mathsf{aux}_{\mathsf{PC}}) \leftarrow \mathsf{PC.Commit}(f(X))$ and the corresponding opening proofs
$$\{(f(i), \pi_{\mathsf{PC},i}) \leftarrow \mathsf{PC.Open}(\mathsf{com}_{\mathsf{PC}}, \mathsf{aux}_{\mathsf{PC}}, i)\}_{i\in\mathsf{chal}}$$
6. define a vector of quadruples L, where for $i \in [n]$:
$$L_i = \begin{cases} (i, m_i, \pi_{\mathsf{VC},i}, f(i)) & \text{if } i \in \mathsf{chal}, \\ (0,0,0,0) & \text{otherwise} \end{cases}$$
7. compute a hiding Merkle commitment to L as $(\mathsf{com}_{\mathsf{MT}}, \mathsf{aux}_{\mathsf{MT}}) \leftarrow \mathsf{MT.Commit}(L)$ and the corresponding opening proofs
$$\{((i, m_i, \pi_{\mathsf{VC},i}, f(i)), \pi_{\mathsf{MT},i}) \leftarrow \mathsf{MT.Open}(\mathsf{com}_{\mathsf{MT}}, \mathsf{aux}_{\mathsf{MT}}, i)\}_{i\in\mathsf{chal}}$$
8. run the NIZK simulator $\mathcal{S}_{\mathsf{NIZK}}(1^\lambda, (\mathsf{pk}, \mathsf{com}_{\mathsf{PC}}))$ to obtain $\mathsf{crs}_{\mathsf{NIZK}}$ and π_{NIZK}
9. output $\mathsf{crs} := (\mathsf{crs}_{\mathsf{PC}}, \mathsf{crs}_{\mathsf{VC}}, \mathsf{crs}_{\mathsf{NIZK}})$, pk, chal, $\mathsf{com} := (\mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{PC}}, \mathsf{com}_{\mathsf{VC}})$ and the proofs $\{\pi_i = (f(i), \pi_{\mathsf{PC},i}, \pi_{\mathsf{MT},i}, \pi_{\mathsf{VC},i}, \pi_{\mathsf{NIZK}})\}_{i\in\mathsf{chal}}$

To argue that this is a good simulator, we define a sequence of hybrids, starting with faithfully executing the specified algorithms and ending in a hybrid whose output is as produced by the simulator above, without any knowledge of secrets.

$\mathbf{H}_0$: Given $(\mathsf{pk}, \mathsf{sk})$ and m:

- sample $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
- compute $(\mathsf{com}, \mathsf{aux}_{\mathsf{com}}) \leftarrow \mathsf{Commit}(\mathsf{sk}, m)$
- sample a k-sized subset $\mathsf{chal} \subseteq [n]$,
- compute $\{(m_i, \pi_i) \leftarrow \mathsf{Open}(\mathsf{sk}, \mathsf{com}, \mathsf{aux}_{\mathsf{com}}, i)\}_{i\in\mathsf{chal}}$
- output $(\mathsf{crs}, \mathsf{pk}, \mathsf{com}, \mathsf{chal}, m, \{\pi_i\}_{i\in\mathsf{chal}})$

$\mathbf{H}_1$: This hybrid is identical to the previous hybrid, except that π_{NIZK} is simulated. This also changes the setup algorithm, and we output the $\mathsf{crs}_{\mathsf{NIZK}}$ at the end, as described in the simulator above. By zero knowledge of the NIZK-AoK, this hybrid is computationally indistinguishable from the previous hybrid.

$\mathbf{H}_2$: In this hybrid, we sample chal at the beginning of the protocol instead of after the commitment is created. This hybrid has an identical distribution to the previous hybrid.

$\mathbf{H}_3$: In this hybrid, we replace all unopened leaves of the Merkle tree with $(0,0,0,0)$ as described in the simulator above. This is now possible because we sampled the challenge before the commitment was created. This hybrid is computationally indistinguishable from the previous hybrid because of the hiding property of the Merkle commitment.

$\mathbf{H}_4$: This is the final hybrid, where we change the polynomial that is committed by the PC by setting $f(0) = 0$, instead of $f(0) = \mathsf{sk}$. The remaining evaluations at the k points in chal are left unchanged. This hybrid is computationally indistinguishable from the previous hybrid due to the hiding property of the polynomial commitment scheme.

Thus, the scheme in Section 6.1 satisfies key-hiding. □

We next prove the key-hiding property for the construction instantiated with KZG commitments.

Proof of Key-Hiding for the Construction in Section 6.2. We begin by describing the simulator $\mathcal{S}(\mathsf{pk}, m)$ which works as follows:

- sample a uniformly random k-sized subset $\mathsf{chal} \subseteq [n]$
- sample $\mathsf{crs}_{\mathsf{VC}} \leftarrow \mathsf{VC.Setup}(1^\lambda, 1^n)$
- commit to m using VC and compute opening proofs for all points:

$$(\mathsf{com}_{\mathsf{VC}}, \mathsf{aux}_{\mathsf{VC}}) \leftarrow \mathsf{VC.Commit}(m)$$
$$\left\{(m_i, \pi_{\mathsf{VC},i}) \leftarrow \mathsf{VC.Open}(\mathsf{com}_{\mathsf{VC}}, \mathsf{aux}_{\mathsf{VC}}, i)\right\}_{i\in[n]}$$

- sample a degree-k polynomial $f(X) \in \mathbb{Z}_p^{\leq k}[X]$ uniformly at random subject to $f(0) = 0$
- to program $\mathsf{crs}_{\mathsf{KZG}}$, sample two uniformly random integers $\tau, \Delta \leftarrow \mathbb{Z}_p$. Set

$$g^\eta := \left(\mathsf{pk}^1\right)^{\frac{1}{\Delta}} \qquad h^\eta := \left(\mathsf{pk}^2\right)^{\frac{1}{\Delta}}$$

  and compute the rest of $\mathsf{crs}_{\mathsf{KZG}}$ honestly using τ.
- sample a uniformly random integer $r \leftarrow \mathbb{Z}_p$ and compute $(\mathsf{com}_{\mathsf{PC}}, \mathsf{aux}_{\mathsf{PC}}) \leftarrow \mathsf{PC.Commit}(f(X); r)$ using the randomness r, along with the corresponding opening proofs:

$$\{(f(i), \pi_{\mathsf{PC},i}) \leftarrow \mathsf{PC.Open}(\mathsf{com}_{\mathsf{PC}}, \mathsf{aux}_{\mathsf{PC}}, i)\}_{i\in\mathsf{chal}}$$

- define a vector of quadruples L, where for $i \in [n]$:

$$L_i = \begin{cases} (i, m_i, \pi_{\mathsf{VC},i}, f(i)) & \text{if } i \in \mathsf{chal}, \\ (0,0,0,0) & \text{otherwise} \end{cases}$$

- compute a hiding Merkle commitment to L as $(\mathsf{com}_{\mathsf{MT}}, \mathsf{aux}_{\mathsf{MT}}) \leftarrow \mathsf{MT.Commit}(L)$ and the corresponding opening proofs

$$\{((i, m_i, \pi_{\mathsf{VC},i}, f(i)), \pi_{\mathsf{MT},i}) \leftarrow \mathsf{MT.Open}(\mathsf{com}_{\mathsf{MT}}, \mathsf{aux}_{\mathsf{MT}}, i)\}_{i\in\mathsf{chal}}$$

- sample a uniformly random integer $s \leftarrow \mathbb{Z}_p$ and compute $\pi_{\mathsf{KZG},0}$ as

$$\pi^1_{\mathsf{KZG},0} = g^{f(\tau)/\tau} \cdot (g^\eta)^s, \quad \pi^2_{\mathsf{KZG},0} = g^{(r-\Delta)-s\tau}$$

  Note that the simulator uses $r - \Delta$ instead of r while computing $\pi^2_{\mathsf{KZG},0}$, whereas for an honestly computed opening proof at 0, it would have used r.
- output $\mathsf{crs} := (\mathsf{crs}_{\mathsf{KZG}}, \mathsf{crs}_{\mathsf{VC}})$, pk, chal, $\mathsf{com} := (\mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{KZG}}, \mathsf{com}_{\mathsf{VC}})$ and the proofs

$$\{\pi_i = (f(i), \pi_{\mathsf{KZG},0}, \pi_{\mathsf{KZG},i}, \pi_{\mathsf{MT},i}, \pi_{\mathsf{VC},i})\}_{i \in \mathsf{chal}}$$

To argue that this is a good simulator, we again define a sequence of hybrids.

$\mathbf{H}_0$: This is the same as $\mathbf{H}_0$ from the proof of Key-Hiding for the generic construction in Section 6.1.

$\mathbf{H}_1$: This hybrid is identical to the previous hybrid, except that $\mathsf{crs}_{\mathsf{KZG}}$ is simulated with g^η and h^η as described above. Since sk/Δ is uniformly distributed over $\mathbb{Z}_p$, $\mathsf{crs}_{\mathsf{KZG}}$ has the same distribution as the honestly-sampled CRS for the KZG scheme, implying that this hybrid has the same distribution as the previous hybrid.

$\mathbf{H}_2$: This is the same as $\mathbf{H}_2$ from the proof of Key-Hiding for the generic construction in Section 6.1.

$\mathbf{H}_3$: This is the same as $\mathbf{H}_3$ from the proof of Key-Hiding for the generic construction in Section 6.1.

$\mathbf{H}_4$: This is the final hybrid, where we change the committed polynomial by setting $f(0) = 0$, instead of $f(0) = \mathsf{sk}$, and compute $\pi_{\mathsf{KZG},0}$ using $r - \Delta$ instead of r, as done by the simulator. Now, for $\pi_{\mathsf{KZG},0}$, it holds that

- $\pi^1_{\mathsf{KZG},0}$ has the same marginal distribution in both hybrids $\mathbf{H}_3$ and $\mathbf{H}_4$, which is uniform over $\mathbb{G}_1$ and is independent from all other $\pi^1_{\mathsf{KZG},i}$ for $i \neq 0$.
- Since in $\mathbf{H}_4$,

$$\begin{aligned}
\mathsf{e}\left(\frac{\mathsf{com}_{\mathsf{KZG}}}{\mathsf{pk}^1}, h\right) &= \mathsf{e}(g^{f(\tau)+r\eta-\mathsf{sk}}, h) \\
&= \mathsf{e}(g^{f(\tau)}, h) \cdot \mathsf{e}\left((\mathsf{pk}^1)^{\frac{r}{\Delta}-1}, h\right) = \mathsf{e}(g^{f(\tau)+(r-\Delta)\eta}, h) \\
&= \mathsf{e}(g^{f(\tau)+s\eta\tau+(r-\Delta)\eta-s\eta\tau}, h) = \mathsf{e}(\pi^1_{\mathsf{KZG},0}, h^\tau) \cdot \mathsf{e}(\pi^2_{\mathsf{KZG},0}, h^\eta)
\end{aligned}$$

  $\pi^2_{\mathsf{KZG},0}$ has the same distribution in both hybrids, which is

$$\left(\mathsf{com}_{\mathsf{KZG}}/((\pi^1_{\mathsf{KZG},0})^\tau \cdot \mathsf{pk}^1)\right)^{1/\eta}$$

  Therefore, $\pi_{\mathsf{KZG},0}$ in $\mathbf{H}_4$ has the same distribution as the honestly computed opening proof for 0 in $\mathbf{H}_3$. By a similar argument, in both hybrids, for any $i \in \mathbb{Z}_p$, $i \neq 0$, $\pi^1_{\mathsf{KZG},i}$ has uniform distribution over $\mathbb{G}_1$ and is independent from all other $\pi^1_{\mathsf{KZG},j}$, $j \neq i$, and $\pi^2_{\mathsf{KZG},i}$ is a deterministic function of $(\mathsf{com}_{\mathsf{KZG}}, i, g^{f(i)}, \pi^1_{\mathsf{KZG},i})$[5]. Moreover, the two collections of the k '$f(i)$' values revealed in $\mathbf{H}_3$ and $\mathbf{H}_4$ for the same indices i as part of the final proof are statistically indistinguishable. Hence, $\mathbf{H}_4$ is statistically indistinguishable from $\mathbf{H}_3$.

Thus, the scheme in Section 6.2 satisfies key-hiding. □

[5]This observation also implies the hiding property for the KZG scheme

Non-Delegatability. We begin by providing intuition for why our schemes satisfy non-delegatability with soundness error $\mu(k) = 2^{-\lambda/k} - k/n$. For the adversary to have a non-negligible probability of winning the $\mu(k)$-soundness game, over the choice of randomness used in sampling chal, $\mathbf{R}_2$ must successfully provide openings for a "large fraction" of points in $[n]$. In particular, if it can only provide openings for $n - t$ points, then its winning probability is at most $(1 - t/n)^k$. For all $t = \Theta(n)$, there exists a sufficiently large k so that this probability is negligible. We then show that, except with negligible probability, each leaf opened by $\mathbf{R}_2$ must appear in the joint set of queries made by $\mathbf{S}_1, \ldots, \mathbf{S}_\ell$ and $\mathbf{R}_1$ to the random oracle (Lemma 1).

Next, we argue that $\mathbf{R}_1$ must have made at least $n - t - k$ of these queries. If more than k queries were made by the set $\mathbf{S}_1, \ldots, \mathbf{S}_\ell$, then some $\mathbf{S}_j$ must have made a query outside the set chal (of size k). Together with the openings for chal, $\mathbf{S}_j$ thus has $k + 1$ points of the polynomial $f(X)$ and can reconstruct the secret $f(0)$. The extractor can do so too, by identifying the queries corresponding to the Merkle leaves by comparing against the vector commitment openings in each leaf node.

For intuition, we present an abridged version of the proof for the generic construction in Section 6.1. The full proof of non-delegatability for the constructions in Section 6.1 and Section 6.2 is presented in Appendix A.2. We start by showing that, in general, if an adversary is able to successfully produce Merkle opening proofs for a randomly chosen challenge (k-sized subset), then it must be the case that a large fraction of leaf nodes are queried at the time of creation of the commitment.

Lemma 1. *Consider the following game for a PPT adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$:*

- $\mathcal{B}_1(\mathsf{crs}) \to (\mathsf{com}_{\mathsf{MT}}, \mathsf{st})$
- *Uniformly sample a k-sized subset* $\mathsf{chal} \subseteq [n]$
- $\mathcal{B}_2(\mathsf{st}, \mathsf{chal}) \to \{M_i, \pi_{\mathsf{MT},i}\}_{i \in \mathsf{chal}}$
- *output* 1 *if* $\mathsf{MT.Verify}(\mathsf{com}_{\mathsf{MT}}, i, M_i, \pi_{\mathsf{MT},i}) = 1$ *for all* $i \in \mathsf{chal}$*, else output* 0

For all PPT $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that pass the above game with non-negligible probability, there exists a polynomial-time extractor $\mathsf{Ext}_{\mathsf{MT}}$*, which, given access to* $\mathsf{com}_{\mathsf{MT}}$ *and the random oracle queries $Q_{\mathcal{B}_1}$ made by $\mathcal{B}_1$, outputs* $\{M_i, \pi_{\mathsf{MT},i}\}_{i \in [n]} \leftarrow \mathsf{Ext}_{\mathsf{MT}}(\mathsf{crs}, Q_{\mathcal{B}_1}, \mathsf{com}_{\mathsf{MT}})$ *such that*

$$\Pr\left[\left|\left\{i \in [n] \mid \mathsf{MT.Verify}(\mathsf{com}_{\mathsf{MT}}, i, M_i, \pi_{\mathsf{MT},i}) = 1\right\}\right| \leq (1 - \nu(k))n\right] \leq \mathsf{negl}(\lambda)$$

for $\nu(k) = 1 - 2^{-\lambda/k}$.

Proof. The extractor follows a "top-down" approach, starting at the root $\mathsf{com}_{\mathsf{MT}}$ and making its way down to the leaf nodes. If at any point, it encounters an intermediate node that is not in $Q_{\mathcal{B}_1}$, then it must be the case that all leaf nodes that are children of this intermediate node cannot be opened. On all paths where it does not encounter such an intermediate node missing from $Q_{\mathcal{B}_1}$, the extractor manages to successfully identify the leaf nodes. At the end, it simply outputs all the reachable leaf nodes and their corresponding paths.

All that is left to argue is that the extractor outputs sufficiently many good leafs nodes (and paths). Let p be the fraction of unreachable leaf nodes. It is sufficient to show that if $p > \nu(k)$, then $\mathcal{B}$ cannot pass the game with non-negligible probability, which leads to a contradiction. The probability that none of the challenge points land on unreachable nodes is at most $(1 - p)^k$, which is $\leq 2^{-\lambda}$, and thus negligible, for all $p \geq 1 - 2^{-\lambda/k} = \nu(k)$.[6] □

Let $\mathcal{A} = (\mathbf{S}_1, \ldots, \mathbf{S}_\ell, \mathbf{R}_1, \mathbf{R}_2)$ be an adversary that wins the soundness game with non-negligible probability, thus $\mathbf{R}_2$ returns

$$(m'_i, \pi_i = \{y_i, \pi_{\mathsf{PC},i}, \pi_{\mathsf{MT},i}, \pi_{\mathsf{VC},i}, \pi_{\mathsf{NIZK},i}\}_{i \in \mathsf{chal}})$$

with valid π_i, which implies for all $i \in \mathsf{chal}$:

1. $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i, m_i, \pi_{\mathsf{VC},i}) = 1$

[6] As the number of queries k increase, we can detect progressively smaller fractions of bad leaves with overwhelming probability. The parameter λ determines the error, and k determines the fraction of messages that can be output at that error rate.

2. $\mathsf{PC.Verify}(\mathsf{com}_{\mathsf{PC}}, i, y_i, \pi_{\mathsf{PC},i}) = 1$
3. $\mathsf{NIZK.Verify}((\mathsf{pk}, \mathsf{com}_{\mathsf{PC}}), \pi_{\mathsf{NIZK},i}) = 1$

By knowledge soundness of NIZK, from (all) $\pi_{\mathsf{NIZK},i}$ we can extract (with overwhelming probability) a polynomial $f'(X)$ of degree at most k and randomness r' s.t.

4. $(\mathsf{com}_{\mathsf{PC}}, \mathsf{aux}_{\mathsf{PC}}) = \mathsf{PC.Commit}(f'(X); r')$ (for some $\mathsf{aux}_{\mathsf{PC}}$) and
5. $(f'(0), \mathsf{pk}) \in \mathcal{R}$

(By Binding of PC, all $\pi_{\mathsf{NIZK},i}$ must yield the same $f'(X)$.)

We now adapt Lemma 1 to the soundness game, letting $\mathcal{B}_1 := (\mathbf{S}_1, \ldots, \mathbf{S}_\ell, \mathbf{R}_1)$ and $\mathcal{B}_2 := (\mathbf{S}_1, \ldots, \mathbf{S}_\ell, \mathbf{R}_2)$ and $f'(X)$ be the polynomial extracted from the NIZK output by $\mathbf{R}_2$. If $\mathcal{A} = (\mathbf{S}_1, \ldots, \mathbf{S}_\ell, \mathbf{R}_1, \mathbf{R}_2)$ wins the soundness game with non-negligible probability, that is, $\mathbf{R}_2$'s output satisfies the first two bullet points in 6. in Fig. 1, then for a set I of size at least $(1 - \nu(k))n$, $\mathcal{B}_1$ must have made random oracle queries of the form

$$\{(i, \hat{m}_i, \hat{\pi}_{\mathsf{VC},i}, \hat{y}_i)\}_{i \in I},$$

so that the content satisfies the following:

6. $\hat{m}_i = m_i$
7. $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i, \hat{m}_i, \hat{\pi}_{\mathsf{VC},i}) = 1$

We moreover argue that for these queries:

8. $\hat{y}_i = f'(i)$

Assume $|I| < (1 - \nu(k))n$. Then, as argued in Lemma 1, with non-negligible probability, chal will contain an index i^* not in I.

Assume 8. does not hold for i^*. Then, running $\mathsf{PC.Open}$ using $\mathsf{aux}_{\mathsf{PC}}$ from 4. produces a proof π'_{PC,i^*} for $f'(i^*)$, while from 2. we have that π_{PC,i^*} is a valid proof for $\hat{y}_{i^*} \neq f'(i^*)$. Together this breaks PC evaluation binding.

We are now ready to construct the extractors $\mathsf{Ext}^{\mathbf{S}_j}$ and $\mathsf{Ext}^{\mathbf{R}_1}$.

$\mathsf{Ext}^{\mathbf{R}_1}(\mathsf{pk}, \mathsf{com}, \langle \mathbf{R}_1, (\mathbf{S}_1, \ldots, \mathbf{S}_\ell), \rangle) \to m''$: Unlike in the proof of Lemma 1, an extractor cannot follow a "top-down" approach to arrive at the leaf nodes, which identifies each leaf query. Indeed, $\mathbf{R}_1$ can for example delegate computation of the intermediate nodes in the Merkle tree without compromising the privacy of its secret key, as in a hiding Merkle commitment, the leaves are committed with randomness. However, when leaf nodes are contained in the random oracle queries made by $\mathbf{R}_1$, the extractor can identify them efficiently by relying on the VC proofs contained in the leaf nodes. More formally, $\mathsf{Ext}^{\mathbf{R}_1}$ does the following:

- Maintain a list $\mathcal{L}$ of tuples from $[n] \times \mathcal{M}$ containing the entries of m that have been recovered so far.
- Parse $\mathsf{com} = (\mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{PC}}, \mathsf{com}_{\mathsf{VC}})$.
- Let Q_R denote the random oracle queries made by $\mathbf{R}_1$. Try parsing each query as $(i, \bar{m}_i, \bar{\pi}_{\mathsf{VC},i}, \bar{y}_i)$, and if $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i, \bar{m}_i, \bar{\pi}_{\mathsf{VC},i}) = 1$, add it to the list $\mathcal{L} \leftarrow \mathcal{L} \cup \{(i, \bar{m}_i)\}$. If some $(i, \bar{m}'_i)$ with $\bar{m}'_i \neq \bar{m}_i$ already exists in the list, abort.
- Define m'' where for $i \in [n]$

$$m''_i = \begin{cases} \bar{m}_i & \text{if } (i, \bar{m}_i) \in \mathcal{L}, \\ \bot & \text{otherwise} \end{cases}$$

- Output m''.

By the Position-Binding of VC, it is infeasible to produce conflicting openings for $\mathsf{com}_{\mathsf{VC}}$ at the same position i. Thus, $\mathsf{Ext}^{\mathbf{R}_1}$ only aborts with negligible probability and moreover (due to the proofs $\hat{\pi}_{\mathsf{VC},i}$ (cf. 7) and $\bar{\pi}_{\mathsf{VC},i}$):

9. $\hat{m}_i = m''_i$ (whenever both of them are defined)

and by 6.:

10. $m_i = m''_i$ (whenever $m''_i \neq \perp$)

If $\left|\{i \in [n] \mid m_i = m''_i\}\right| > (1-\mu(k))n$ with $\mu(k) = 1 - \nu(k) - k/n$, then $\mathcal{A}$ loses the soundness game. We next show that if this is not the case, then one of $\{\mathsf{Ext}^{\mathbf{S}_j}\}_{j\in[\ell]}$ can recover the secret key.

$\mathsf{Ext}^{\mathbf{S}_j}(\mathsf{pk}, \mathsf{com}, \langle \mathbf{S}_j, \mathbf{R}_1\rangle, \langle \mathbf{S}_j, \mathbf{R}_2, \{(m'_i, \pi_i)\}_{i\in\mathsf{chal}}\rangle) \to \mathsf{sk}$: This extractor monitors the random oracle queries for valid leaf nodes by $\mathbf{S}_j$. If one of them is outside of chal, then using the collected $k+1$ points y_i output by $\mathbf{R}_2$, the interpolating polynomial reveals the secret key.

- Maintain a list $\mathcal{L}$ of tuples from $[n]\times\mathcal{M}\times\mathbb{Z}_p$ containing the set of evaluations that have been recovered.
- Parse $\mathsf{com} = (\mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{PC}}, \mathsf{com}_{\mathsf{VC}})$.
- For $i \in \mathsf{chal}$, parse $\mathbf{R}_2$'s output m'_i and $\pi_i = (y_i, \pi_{\mathsf{PC},i}, \pi_{\mathsf{MT},i}, \pi_{\mathsf{VC},i}, \pi_{\mathsf{NIZK},i})$.
- Let Q_S denote the random oracle queries made by $\mathbf{S}_j$. Try parsing each query as $(i, \bar{m}_i, \bar{\pi}_{\mathsf{VC},i}, \bar{y}_i)$, and if $i \notin \mathsf{chal}$ and $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i, \bar{m}_i, \bar{\pi}_{\mathsf{VC},i}) = 1$, add it to the list $\mathcal{L} \leftarrow \mathcal{L} \cup \{(i, \bar{m}_i, \bar{y}_i)\}$.
- If $\mathcal{L} = \emptyset$, abort.
- Else for all entries $(i, \bar{m}_i, \bar{y}_i) \in \mathcal{L}$: Interpolate a polynomial $f_i(X)$ through the point $(i, \bar{y}_i)$ as well as $\{(i, y_i)\}_{i\in\mathsf{chal}}$. If $(f_i(0), \mathsf{pk}) \in \mathcal{R}$ then stop and output $f_i(0)$.
- Else, abort.

Clearly, $\mathsf{Ext}^{\mathbf{S}_j}$ runs in polynomial time. Moreover, by 6. and 8., $(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$ and $\mathbf{R}_1$ made random oracle queries $\{(i, \hat{m}_i, \hat{\pi}_{\mathsf{VC},i}, \hat{y}_i)\}_{i\in I}$ for $|I| \geq (1-\nu(k))n$ with

11. $\hat{m}_i = m_i$ and
12. $\hat{y}_i = f'(i)$

If $|\{i \in [n] \mid m_i = m''_i\}| \leq (1-\nu(k))n - k$, then one of the queries i^* in I but outside of chal must have been made by some $\mathbf{S}_j$, and thus $(i^*, \hat{m}_{i^*}, \hat{y}_{i^*})$ will be in its list $\mathcal{L}$, implying that $f_{i^*}(X) = f'(X)$ and it will recover $f'(0)$, which is a secret key for pk. Hence, if $\mathsf{Ext}^{\mathbf{R}_1}$ outputs at most $(1-\nu(k)-k/n)n$ points, a value sk such that $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{R}$ can be extracted from $\mathbf{S}_j$'s random oracle queries except with negligible probability. This implies that $\mathcal{A}$ cannot pass the soundness game except with negligible probability (Fig. 1) and concludes the proof showing that the construction in Section 6.1 is a secure non-delegatable commitment scheme.

Remark 6. *To ensure that a polynomial-time extractor $\mathsf{Ext}^{\mathbf{S}_j}$ can identify the polynomial $f'(X)$, it is crucial that the degree of the polynomial is at most a constant k larger than the number of opened leaves.*

7 Implementation

We implemented the concrete construction in Section 6.2 in Rust using arkworks [ark] for the instantiation of the pairing-friendly curve BLS12-381 [BLS02].[7] All of the experiments were run on a 2024 MacBook Pro with an Apple M4 Pro processor (12 cores) and 24 GB of unified memory in single-threaded mode.

For the Merkle tree, we used the Winterfell library [Win] and the SHA-256 implementation of arkworks. The hiding KZG scheme in Section 4.1 was implemented using the non-hiding scheme in [Win] as the

[7] Code can be found here: https://github.com/nusret1996/non-delegatable-commitments

baseline, which simulates its own trusted setup.[8] The experiment data was obtained using the criterion benchmarking library [cri]. The reported numbers are the average of 100 trials.

Commitments. Each commitment consists of two Merkle tree commitments and one hiding KZG commitment. Each hiding KZG commitment is one $\mathbb{G}_1$ element of 48 bytes, and each Merkle tree commitment contains a Merkle tree root of 32 bytes and two 8 byte parameters for the tree depth and number of leaves. Thus, the total commitment size is 144 bytes. Since the tree depth and leaf count is the same for $\mathsf{com}_{\mathsf{MT}}$ and $\mathsf{com}_{\mathsf{VC}}$, and the former can be inferred from the latter, the commitment size can be reduced to 120 bytes in a more optimized implementation.

Table 1a on the left shows the time to create a commitment for different (n, k) tuples, where n is the number of 32-byte messages and k is the polynomial degree. As the number of public key operations scales only in k, it takes less than 2 seconds to commit to $\approx 100,000$ messages while allowing up to 512 openings to be made. Commitment times (asymptotically) scale as $O(n \log n)$ after accounting for the time to create the KZG commitment. This is because constructing the Merkle tree MT requires placing n $O(\log n)$-sized Merkle proofs from VC into its leaves.

Table 1: **a)** Time to create commitments with varying size of messages (n), and maximum number of opening queries in the $\mu(k)$-soundness game. **b)** Time to create a batch opening of queries (T_{open}) and verify them (T_{verify}) for $n = 131,072$. The polynomial degree k is set to be the same as the batch size. Total proof sizes ($|\pi|$), and their breakdown into Merkle proofs ($|\pi_{\mathsf{MT}}|$ and $|\pi_{\mathsf{VC}}|$). All times are in ms, and proof sizes in kB.

k \ n	2^{11}	2^{14}	2^{17}
2^6	10.85	68.21	557.40
2^7	15.37	92.01	739.06
2^8	24.84	142.17	1110
2^9	42.14	240.10	1840

(a) Commitment Benchmarks

Batch Size	T_{commit}	T_{open}	T_{verify}	$\lvert\pi\rvert$	$\lvert\pi_{\mathsf{MT}}\rvert$	$\lvert\pi_{\mathsf{VC}}\rvert$
1	380.29	0.39	4.81	1.40	0.61	0.57
10	407.78	0.61	6.09	11.10	5.41	5.08
20	436.81	0.98	7.35	20.52	10.08	9.43
30	466.25	1.51	8.79	29.56	14.56	13.59
40	494.26	2.41	10.92	38.09	18.78	17.50
50	524.54	3.43	12.56	46.49	22.94	21.34

(b) Open and Verify Benchmarks

Openings and Verification. Table 1b on the right shows the time to batch open (T_{open}) and verify (T_{verify}) proofs for ℓ messages, along with the total proof size ($|\pi|$) and its breakdown into the Merkle proof sizes, namely $|\pi_{\mathsf{MT}}|$ and $|\pi_{\mathsf{VC}}|$. These constitute the majority of the proof. There are two KZG proofs, namely the batch proof and the opening proof for point 0, each taking 96 bytes (two $\mathbb{G}_1$ elements). Besides the Merkle and KZG proofs, the final proof contains indices of 8 bytes and polynomial evaluations of 32 bytes for each message. The difference in $|\pi_{\mathsf{MT}}|$ and $|\pi_{\mathsf{VC}}|$ is due to the random 32-byte values, which was sampled for the leaves of the hiding Merkle tree MT and revealed as part of the Merkle proof. Together, π_{MT} and π_{VC} constitute over 85% of all proofs.

Comparison to Beholder Signatures. We provide a direct comparison to Beholder Signatures [DFK+25] by using their parameter ranges. The benchmarks for beholder signatures target an extraction rate of $\mu = 1/2$ except with probability $\epsilon \approx 3.6 \times 10^{-5}$. From Remark 5, given a target rate μ and $\ell \leq k$ openings selected uniformly at random, the non-delegatable commitment scheme achieves μ-soundness except with probability $\epsilon + \mathsf{negl}(\lambda)$, where $\epsilon = (1 - \mu - k/n)^{\ell}$. In Table 2, we compare the times to generate a beholder signature and ℓ opening proofs of a non-delegatable commitment scheme for $k = \ell$, $\mu = 1/2$, $\epsilon \approx 3.6 \times 10^{-5}$ and different values of n. To provide a fair comparison we ran the implementation provided by [DFK+25] to obtain benchmarks on the same machine as ours.

[8] As we only commit to polynomials of small degree, *e.g.*, ≤ 100, it is possible to run the setup via a secure multi-party computation, or use the CRS established by the Ethereum community.

Table 2: Comparison of Non-Delegatable Commitments (NDC) against Beholder Signatures (BS) both with a guarantee of extracting $\mu = 1/2$ fraction of data, except with probability $\epsilon < 3.6 \times 10^{-5}$ for a message with n chunks that are 32-bytes each. The total time to compute a BS is a sum of two values: T_{FK20} – time to compute KZG openings using [FK23], and T_{Mining} – time to perform the mining process as in Fischlin transform [Fis05]. For NDC, we report the commitment time (T_{commit}), opening time (T_{open}) and proof size ($|\pi|$) for $k = \ell$ and the given value of n. The total time to compute an NDC proof is the sum of T_{commit} and T_{open}. The proof sizes are in kB.

n	**Beholder Signatures**			ℓ	**Non-Delegatable Commitments**			
	T_{FK20}	T_{Mining}	T_{total}		T_{commit}	T_{open}	T_{total}	$\lvert\pi\rvert$
2^9	0.27 s	0.14 s	0.41 s	11	2.31 ms	0.55 ms	2.86 ms	6.30
2^{10}	0.56 s	0.29 s	0.85 s	12	3.74 ms	0.59 ms	4.33 ms	7.54
2^{11}	1.17 s	0.53 s	1.70 s	14	6.50 ms	0.64 ms	7.14 ms	9.50
2^{12}	2.45 s	1.05 s	3.50 s	15	12.45 ms	0.70 ms	13.15 ms	11.06
2^{13}	5.04 s	2.17 s	7.20 s	15	24.06 ms	0.71 ms	24.77 ms	12.03
2^{14}	10.26 s	4.32 s	14.59 s	15	48.93 ms	0.72 ms	49.65 ms	13.01
2^{15}	20.50 s	8.27 s	28.77 s	15	98.70 ms	0.75 ms	99.45 ms	13.98
2^{16}	42.63 s	16.82 s	59.45 s	15	204.42 ms	0.74 ms	205.16 ms	14.96

The total time to commit to the data and generate a proof is over 100 times smaller for non-delegatable commitments compared to the time to create beholder signatures. Moreover, most of the time is due to the commit operation, which must be done once when multiple proofs are generated. However, beholder signatures for the given parameters are 3.06 kB regardless of the data size, which is smaller than our proof sizes. We chose to optimize the time to commit and generate proofs rather than the proof size as a design principle; since our use-cases such as high-performance blockchains provide abundant block space (and can delete the proofs after their verification), yet, impose strict timing requirements (*e.g.*, Solana can support 100,000 transactions per second but impose a block time of 400 ms).

Acknowledgments. We thank Joachim Neu and Paul Resnick for several insightful discussions on this project. This work was performed in part during a research internship of the third author and a research visit of the first author at *a16z crypto*. The first author is supported by the Vienna Science and Technology Fund (WWTF) [10.47379/VRG18002] and by the Austrian Science Fund (FWF) [10.55776/F8515-N].

References

[AADG25] Prabhanjan Ananth, Gilad Asharov, Hila Dahari, and Vipul Goyal. Towards accountability in CRS generation. *J. Cryptol.*, 38(4):29, 2025. 7

[ABC+07] Giuseppe Ateniese, Randal C. Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary N. J. Peterson, and Dawn Xiaodong Song. Provable data possession at untrusted stores. In *CCS*, pages 598–609. ACM, 2007. 1

[ACH11] Gilad Asharov, Ran Canetti, and Carmit Hazay. Towards a game theoretic view of secure computation. In *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 426–445. Springer, 2011. 6

[ADGH06] Ittai Abraham, Danny Dolev, Rica Gonen, and Joseph Y. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In *PODC*, pages 53–62. ACM, 2006. 6

[AL10] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *J. Cryptol.*, 23(2):281–343, 2010. 6

[Anz25] Anza / Agave Documentation. Turbine block propagation. https://docs.anza.xyz/consensus/turbine-block-propagation, 2025. Accessed: 2025-10-02. 2

[APMT08] Giuseppe Ateniese, Roberto Di Pietro, Luigi V. Mancini, and Gene Tsudik. Scalable and efficient provable data possession. In *SecureComm*, page 9. ACM, 2008. 1

[ark] arkworks. .https://arkworks.rs,2022. 23

[BFL20] Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In *CRYPTO (2)*, volume 12171 of *Lecture Notes in Computer Science*, pages 121–151. Springer, 2020. 34

[BG17] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *CoRR*, abs/1710.09437, 2017. 6

[BGM+18] Christian Badertscher, Juan A. Garay, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. But why does it work? A rational protocol design treatment of bitcoin. In *EUROCRYPT (2)*, volume 10821 of *Lecture Notes in Computer Science*, pages 34–65. Springer, 2018. 6

[BK22] Dan Boneh and Chelsea Komlo. Threshold signatures with private accountability. In *CRYPTO (4)*, volume 13510 of *Lecture Notes in Computer Science*, pages 551–581. Springer, 2022. 6

[BKN17] Dan Boneh, Sam Kim, and Valeria Nikolaenko. Lattice-based DAPS and generalizations: Self-enforcement in signature schemes. In *ACNS*, volume 10355 of *Lecture Notes in Computer Science*, pages 457–477. Springer, 2017. 6

[BLR24] Eric Budish, Andrew Lewis-Pye, and Tim Roughgarden. The economic limits of permissionless consensus. In *EC*, pages 704–731. ACM, 2024. 6

[BLS02] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In *SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 257–267. Springer, 2002. 23

[BPR24a] Dan Boneh, Aditi Partap, and Lior Rotem. Accountability for misbehavior in threshold decryption via threshold traitor tracing. In *CRYPTO (7)*, volume 14926 of *Lecture Notes in Computer Science*, pages 317–351. Springer, 2024. 7

[BPR24b] Dan Boneh, Aditi Partap, and Lior Rotem. Proactive refresh for accountable threshold signatures. In *FC (2)*, volume 14745 of *Lecture Notes in Computer Science*, pages 140–159. Springer, 2024. 6

[BPR24c] Dan Boneh, Aditi Partap, and Lior Rotem. Traceable secret sharing: Strong security and efficient constructions. In *CRYPTO (5)*, volume 14924 of *Lecture Notes in Computer Science*, pages 221–256. Springer, 2024. 7

[BPR25] Dan Boneh, Aditi Partap, and Lior Rotem. Traceable verifiable random functions. In *CRYPTO (2)*, volume 16001 of *Lecture Notes in Computer Science*, pages 203–234. Springer, 2025. 7

[BPS17] Mihir Bellare, Bertram Poettering, and Douglas Stebila. Deterring certificate subversion: Efficient double-authentication-preventing signatures. In *Public Key Cryptography (2)*, volume 10175 of *Lecture Notes in Computer Science*, pages 121–151. Springer, 2017. 6

[BYK21] Osman Biçer, Burcu Yildiz, and Alptekin Küpçü. m-stability: Threshold security meets transferable utility. In *CCSW*, pages 73–82. ACM, 2021. 6

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 257–270. Springer, 1994. 7

[CFS22] Dario Catalano, Georg Fuchsbauer, and Azam Soleimanian. Double-authentication-preventing signatures in the standard model. *J. Comput. Secur.*, 30(1):3–38, 2022. 6

[CKBA08] Reza Curtmola, Osama Khan, Randal C. Burns, and Giuseppe Ateniese. MR-PDP: multiple-replica provable data possession. In *ICDCS*, pages 411–420. IEEE Computer Society, 2008. 1

[cri] Crate criterion. https://docs.rs/criterion/latest/criterion/. 24

[DFK+25] Stefan Dziembowski, Sebastian Faust, Paweł Kędzior, Marcin Mielniczuk, Susil Kumar Mohanty, and Krzysztof Pietrzak. Beholder signatures. Cryptology ePrint Archive, Paper 2025/1900, 2025. 7, 8, 24

[DFL23] Stefan Dziembowski, Sebastian Faust, and Tomasz Lizurej. Individual cryptography. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 547–579. Springer, Cham, August 2023. 7

[DHR00] Yevgeniy Dodis, Shai Halevi, and Tal Rabin. A cryptographic solution to a game theoretic problem. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 112–130. Springer, Berlin, Heidelberg, August 2000. 6

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 152–168. Springer, 2005. 25

[FK23] Dankrad Feist and Dmitry Khovratovich. Fast amortized KZG proofs. Cryptology ePrint Archive, Paper 2023/033, 2023. 25

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018. 4, 12, 16

[FKN10] Georg Fuchsbauer, Jonathan Katz, and David Naccache. Efficient rational secret sharing in standard communication networks. In *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 419–436. Springer, 2010. 6

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. ACM, 2009. 4

[GK06] S. Dov Gordon and Jonathan Katz. Rational secret sharing, revisited. In *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 229–241. Springer, 2006. 6

[GK12] Adam Groce and Jonathan Katz. Fair computation with rational players. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 81–98. Springer, 2012. 6

[GKM+13] Juan A. Garay, Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Rational protocol design: Cryptography against incentive-driven adversaries. In *54th FOCS*, pages 648–657. IEEE Computer Society Press, October 2013. 6

[GKS22] Chaya Ganesh, Bhavana Kanukurthi, and Girisha Shankar. Secure auctions in the presence of rational adversaries. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1173–1186. ACM Press, November 2022. 6

[GKTZ15] Juan A. Garay, Jonathan Katz, Björn Tackmann, and Vassilis Zikas. How fair is your protocol?: A utility-based approach to protocol optimality. In *PODC*, pages 281–290. ACM, 2015. 6

[GLR13] Ronen Gradwohl, Noam Livne, and Alon Rosen. Sequential rationality in cryptographic protocols. *ACM Trans. Economics and Comput.*, 1(1):2:1–2:38, 2013. 6

[GSS21] Vipul Goyal, Yifan Song, and Akshayaram Srinivasan. Traceable secret sharing and applications. In *CRYPTO (3)*, volume 12827 of *Lecture Notes in Computer Science*, pages 718–747. Springer, 2021. 7

[Hal11] Joseph Y. Halpern. Beyond nash equilibrium: Solution concepts for the 21st century. In *GameSec*, volume 7037 of *Lecture Notes in Computer Science*, pages 1–3. Springer, 2011. 6

[HT04] Joseph Y. Halpern and Vanessa Teague. Rational secret sharing and multiparty computation: extended abstract. In *STOC*, pages 623–632. ACM, 2004. 6

[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. 1

[KBD+24] Mahimna Kelkar, Kushal Babel, Philip Daian, James Austgen, Vitalik Buterin, and Ari Juels. Complete knowledge: Preventing encumbrance of cryptographic secrets. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 2415–2429. ACM Press, October 2024. 7

[KGP+25] Mahimna Kelkar, Aadityan Ganesh, Aditi Partap, Joseph Bonneau, and S. Matthew Weinberg. Breaking omertà: On threshold cryptography, smart collusion, and whistleblowing. Cryptology ePrint Archive, Paper 2025/1582, 2025. 7

[KN08] Gillat Kol and Moni Naor. Cryptography and game theory: Designing protocols for exchanging information. In *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 320–339. Springer, 2008. 6

[KT13] Aggelos Kiayias and Qiang Tang. How to keep a secret: leakage deterring public-key cryptosystems. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 943–954. ACM Press, November 2013. 7

[KT15] Aggelos Kiayias and Qiang Tang. Traitor deterring schemes: Using bitcoin as collateral for digital content. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 231–242. ACM Press, October 2015. 7

[KT24] Tohru Kohrita and Patrick Towa. Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments. *J. Cryptol.*, 37(4):38, 2024. 6, 11, 12, 16

[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. 6, 11

[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. 4

[NS24] Ngoc Khanh Nguyen and Gregor Seiler. Greyhound: Fast polynomial commitments from lattices. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 243–275. Springer, Cham, August 2024. 6

[PH10] Rafael Pass and Joe Halpern. Game theory with costly computation: formulation and application to protocol security. In *BQGT*, page 89:1. ACM, 2010. 6

[PS14] Bertram Poettering and Douglas Stebila. Double-authentication-preventing signatures. In *ESORICS (1)*, volume 8712 of *Lecture Notes in Computer Science*, pages 436–453. Springer, 2014. 6

[RKS15] Tim Ruffing, Aniket Kate, and Dominique Schröder. Liar, liar, coins on fire!: Penalizing equivocation by loss of bitcoins. In *CCS*, pages 219–230. ACM, 2015. 6

[SCW23] Elaine Shi, Hao Chung, and Ke Wu. What can cryptography do for decentralized mechanism design? In Yael Tauman Kalai, editor, *ITCS 2023*, volume 251, pages 97:1–97:22. LIPIcs, January 2023. 6

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997. 4

[Sol24] Solana Labs. Solana validator requirements. https://docs.solanalabs.com/operations/requirements, 2024. Accessed: 2025-10-02. 2

[SW13] Hovav Shacham and Brent Waters. Compact proofs of retrievability. *J. Cryptol.*, 26(3):442–483, 2013. 1

[SWN+21] Peiyao Sheng, Gerui Wang, Kartik Nayak, Sreeram Kannan, and Pramod Viswanath. BFT protocol forensics. In *CCS*, pages 1722–1743. ACM, 2021. 6

[Win] Winterfell merkle tree implementation. https://github.com/facebook/winterfell/tree/main/crypto/src/merkle. 23

[WWW25] Brent Waters, Hoeteck Wee, and David J. Wu. New techniques for preimage sampling: Improved nizks and more from LWE. In *EUROCRYPT (4)*, volume 15604 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2025. 6

[ZGK+17] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. A zero-knowledge version of vsql. *IACR Cryptol. ePrint Arch.*, page 1146, 2017. 11

[Zha25] Mark Zhandry. Optimal traitor tracing from pairings. In *EUROCRYPT (3)*, volume 15603 of *Lecture Notes in Computer Science*, pages 305–335. Springer, 2025. 7

A Appendix

A.1 Proof of Correctness

Theorem 4. *If the polynomial and vector commitment schemes* PC, MT *and* VC *satisfy correctness, both constructions in Section 6.1 and Section 6.2 satisfy correctness, where* PC *is the* KZG *scheme from Section 4.1 in the case of the construction in Section 6.2*

Proof. By the correctness of PC, MT, and VC, for any commitment com and proof π output by an honest receiver such that $\mathsf{com} \coloneqq (\mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{PC}}, \mathsf{com}_{\mathsf{VC}})$, and $\pi \coloneqq (f(i), f(x), \pi_{\mathsf{PC},i}, \pi_{\mathsf{PC},x}, \pi_{\mathsf{MT},i}, \pi_{\mathsf{VC},i}, \pi_{\mathsf{NIZK}})$, the

following holds with probability 1:

$$\begin{aligned}
&\mathsf{NIZK.Verify}(\mathsf{pk}, \mathsf{com}_{\mathsf{PC}}, \pi_{\mathsf{NIZK}}) = 1 \text{ for the generic construction,}\\
&\mathsf{PC.Verify}(\mathsf{com}_{\mathsf{PC}}, i, f(i), \pi_{\mathsf{PC},i}) = 1 \;\text{ for }\; i \in [k],\\
&\mathsf{MT.Verify}(\mathsf{com}_{\mathsf{MT}}, i, (i, m_i, \pi_{\mathsf{VC},i}, f(i)), \pi_{\mathsf{MT},i}) = 1,\\
&\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i, m_i, \pi_{\mathsf{VC},i}) = 1\\
&\mathsf{e}(\mathsf{com}_{\mathsf{KZG}}/\mathsf{pk}^1, h) = \mathsf{e}(\pi^1_{\mathsf{KZG}}, h^\tau) \cdot \mathsf{e}(\pi^2_{\mathsf{KZG}}, h^\eta) \text{ for the construction with KZG.}
\end{aligned}$$

Therefore, both constructions satisfy correctness. □

A.2 Proof of Non-delegatability

Theorem 2. *If* VC *is a computationally position-binding vector commitment scheme,* PC *is a computationally evaluation-binding polynomial commitment scheme, and the* NIZK *scheme satisfies knowledge soundness, the construction in Section 6.1 satisfies non-delegatability with* $\mu(k) = (2^{-\lambda/k} - k/n)$*-soundness in the random oracle model.*

Proof. Consider a PPT adversary $\mathcal{A} = (\mathbf{S}_1, \ldots, \mathbf{S}_\ell, \mathbf{R}_1, \mathbf{R}_2)$ such that for the messages m, commitment com and proof π output by $\mathcal{A}$, it holds that $m_i = m'_i$, and $\mathsf{Verify}(\mathsf{pk}, \mathsf{com}, i, m_i, \pi_i) = 1$ for all $i \in \mathsf{chal}$. Let $\mathsf{com} = (\mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{PC}}, \mathsf{com}_{\mathsf{VC}})$ and $\pi_i = (f(i), \pi_{\mathsf{PC},i}, \pi_{\mathsf{MT},i}, \pi_{\mathsf{VC},i}, \pi_{\mathsf{NIZK},i})$.

By the knowledge soundness of NIZK, a polynomial $f'_i(X)$ along with its randomness r'_i, can be extracted from each $\pi_{\mathsf{NIZK},i}$ using the expected-polynomial-time extractor for the following relation:

$$\mathcal{R}_{\mathsf{NIZK}} = \left\{ (\mathsf{pk}, \mathsf{com}_{\mathsf{PC}}; \mathsf{sk}, f(X), r) \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \in \mathcal{R} \\ f(X) \in \mathbb{Z}_p^{\leq k}[X] \\ \mathsf{com}_{\mathsf{PC}} = \mathsf{PC.Commit}(f(X); r) \\ f(0) = \mathsf{sk} \end{array} \right\}$$

Let $f'_i(X)$ and r'_i denote the output of this extractor, and note that as $\pi_{\mathsf{NIZK},i}$ verifies, $f'_i(X) \in \mathbb{Z}_p^{\leq k}[X]$, $\mathsf{com}_{\mathsf{PC}} = \mathsf{PC.Commit}(f'_i(X); r'_i)$, and $(\mathsf{pk}, f'_i(0)) \in \mathcal{R}$, except with negligible probability ϵ_{NIZK}. By the binding property of the PC scheme, all of the polynomials $f'_i(X)$ are equal, *i.e.*, $f'_i(X) = f'(X)$ for all $i \in \mathsf{chal}$.

We next prove the following lemma, which follows from similar arguments to Lemma 1. Recall that m denotes the sequence of messages output by $(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$.

Lemma 2. *Let* $m \in \mathcal{M}^n$ *be a fixed sequence, and consider the following game for a PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*:*

- $\mathcal{A}_1(\mathsf{crs}) \to (\mathsf{pk}, \mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{VC}}, \mathsf{com}_{\mathsf{PC}}, \mathsf{aux})$
- *Uniformly sample a* k*-sized subset* $\mathsf{chal} \subseteq [n]$
- $\mathcal{A}_2(\mathsf{chal}, \mathsf{aux}_{\mathsf{MT}}) \to \{(i, m'_i, \pi'_{\mathsf{VC},i}, y'_i), \pi'_{\mathsf{MT},i}, \pi'_{\mathsf{PC},i}, \pi'_{\mathsf{NIZK},i}\}_{i \in \mathsf{chal}}$
- *output* 1 *if*
  - $m'_i = m_i$*,*
  - $\mathsf{MT.Verify}(\mathsf{com}_{\mathsf{MT}}, i, (i, m'_i, \pi'_{\mathsf{VC},i}, y'_i), \pi'_{\mathsf{MT},i}) = 1$*,*
  - $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i, m'_i, \pi'_{\mathsf{VC},i}) = 1$*,*
  - $\mathsf{PC.Verify}(\mathsf{com}_{\mathsf{PC}}, i, y'_i, \pi'_{\mathsf{PC},i}) = 1$*.*
  - $\mathsf{NIZK.Verify}((\mathsf{pk}, \mathsf{com}_{\mathsf{PC}}), \pi'_{\mathsf{NIZK},i}) = 1$

  Else, output 0

For all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that pass the above game with non-negligible probability, there exists a polynomial-time extractor $\mathsf{Ext}_{\mathsf{MT}}$, which, given access to $\mathsf{com}_{\mathsf{MT}}$ and the random oracle queries $Q_{\mathcal{A}_1}$ made by $\mathcal{A}_1$, outputs

$$\{(i, \hat{m}_i, \hat{\pi}_{\mathsf{VC},i}, \hat{y}_i), \hat{\pi}_{\mathsf{MT},i}\}_{i\in[n]} \leftarrow \mathsf{Ext}_{\mathsf{MT}}(\mathsf{crs}, Q_{\mathcal{A}_1}, \mathsf{com}_{\mathsf{MT}})$$

such that for over $(1-\nu(k))n$ indices, where $\nu(k) = 1 - 2^{-\lambda/k}$, the following holds except with negligible probability:

1. $\mathsf{MT.Verify}(\mathsf{com}_{\mathsf{MT}}, i, (i, \hat{m}_i, \hat{\pi}_{\mathsf{VC},i}, \hat{y}_i), \hat{\pi}_{\mathsf{MT},i}) = 1$,
2. $\hat{m}_i = m_i$,
3. $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i, \hat{m}_i, \hat{\pi}_{\mathsf{VC},i}) = 1$,
4. $\hat{y}_i = f'(i)$ *for a degree* $(\leq k)$*-polynomial such that* $(\mathsf{pk}, f'(0)) \in \mathcal{R}$.

Proof. The extractor follows a "top-down" approach, starting at the root $\mathsf{com}_{\mathsf{MT}}$ and making its way down to the leaf nodes. If at any point, it encounters an intermediate node that is not in $Q_{\mathcal{A}_1}$, then it must be the case that all leaf nodes that are children of this intermediate node cannot be opened. On all paths where it does not encounter such an intermediate node missing from $Q_{\mathcal{A}_1}$, the extractor manages to successfully identify the leaf nodes. At the end, it simply outputs all the reachable leaf nodes and their corresponding paths. For the non-reachable nodes and paths, it outputs $\perp$.

We next argue that the extractor outputs over $(1-\nu(k))n$ leaf nodes (and Merkle opening proofs) such that for each of these leaf nodes $(i, \hat{m}_i, \hat{\pi}_{\mathsf{VC},i}, \hat{y}_i)$ and the corresponding opening proofs $\hat{\pi}_{\mathsf{MT},i}$, 1.–4. from above hold.

Consider an adversary $\mathcal{A}$ that passes the game above. Let $f'_i(X)$ and r'_i denote the output of the expected-polynomial-time extractor run on $\pi_{\mathsf{NIZK},i}$ for the relation $\mathcal{R}_{\mathsf{NIZK}}$. Note that as $\pi_{\mathsf{NIZK},i}$ verifies, $f'_i(X) \in \mathbb{Z}_p^{\leq k}[X]$, $\mathsf{com}_{\mathsf{PC}} = \mathsf{PC.Commit}(f'_i(X); r'_i)$, and $(\mathsf{pk}, f'_i(0)) \in \mathcal{R}$, except with negligible probability ϵ_{NIZK}. By the binding property of the PC scheme, all of the polynomials $f'_i(X)$ and randomness r'_i are equal, *i.e.*, $f'_i(X) = f'(X)$, $r'_i = r$ for all $i \in \mathsf{chal}$.

Suppose for $\ell_1 \geq \nu(k)n$ indices $(a_i)_{i\in[\ell_1]} \subseteq [n]$, at least one of the items above is not true. Then, either $\mathsf{chal} \cap (a_i)_{i\in[\ell_1]} = \emptyset$, which happens with probability at most $(1-\nu(k))^k = 2^{-\lambda}$, or there exists an index $i^* \in \mathsf{chal} \cap (a_i)_{i\in[\ell_1]}$, which implies the union of the following events:

Event 1: $\mathsf{MT.Verify}(\mathsf{com}_{\mathsf{MT}}, i^*, (i^*, \hat{m}_{i^*}, \hat{\pi}_{\mathsf{VC},i^*}, \hat{y}_{i^*}), \hat{\pi}_{\mathsf{MT},i^*}) \neq 1$. This implies i^* is an unreachable leaf node, which is a contradiction, since $\mathcal{A}_2$ outputs a leaf $(i^*, m'_{i^*}, \pi'_{\mathsf{VC},i^*}, y'_{i^*})$ and an opening proof π'_{MT,i^*} such that

$$\mathsf{MT.Verify}(\mathsf{com}_{\mathsf{MT}}, i^*, (i^*, m'_{i^*}, \pi'_{\mathsf{VC},i^*}, y'_{i^*}), \pi'_{\mathsf{MT},i^*}) = 1.$$

Therefore, this event cannot happen.

Event 2: $\hat{m}_{i^*} \neq m_{i^*}$. Then, as $\mathcal{A}$ passes the game, $m'_{i^*} = m_{i^*}$, and

- $\mathsf{MT.Verify}(\mathsf{com}_{\mathsf{MT}}, i^*, (i^*, m_{i^*}, \pi'_{\mathsf{VC},i^*}, y'_{i^*}), \pi'_{\mathsf{MT},i^*}) = 1$, and

However, in this case, $(i^*, m_{i^*}, \pi'_{\mathsf{VC},i^*}, y'_{i^*}) \neq (i^*, \hat{m}_{i^*}, \hat{\pi}_{\mathsf{VC},i^*}, \hat{y}_{i^*})$ both verify with respect to $\mathsf{com}_{\mathsf{MT}}$, which by the position-binding property of Merkle commitments, happens with probability at most $\epsilon_{\mathsf{MT}} = \mathsf{negl}(\lambda)$.

Event 3. $\hat{m}_{i^*} = m_{i^*}$ and $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i^*, \hat{m}_{i^*}, \hat{\pi}_{\mathsf{VC},i^*}) \neq 1$. Then, as $\mathcal{A}$ passes the game,

- $\mathsf{MT.Verify}(\mathsf{com}_{\mathsf{MT}}, i^*, (i^*, m_{i^*}, \pi'_{\mathsf{VC},i^*}, y'_{i^*}), \pi'_{\mathsf{MT},i^*}) = 1$, and
- $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i^*, m_{i^*}, \pi'_{\mathsf{VC},i^*}) = 1$.

Now, by the fact that $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i^*, \hat{m}_{i^*}, \hat{\pi}_{\mathsf{VC},i^*}) \neq 1$, and $\hat{m}_{i^*} = m_{i^*}$, it must be that $\hat{\pi}_{\mathsf{VC},i^*} \neq \pi'_{\mathsf{VC},i^*}$. However, in this case, $(i^*, m_{i^*}, \pi'_{\mathsf{VC},i^*}, y'_{i^*}) \neq (i^*, m_{i^*}, \hat{\pi}_{\mathsf{VC},i^*}, \hat{y}_{i^*})$ both verify with respect to $\mathsf{com}_{\mathsf{MT}}$, which by the position-binding property of Merkle commitments, happens with probability at most $\epsilon_{\mathsf{MT}} = \mathsf{negl}(\lambda)$.

Event 4. $\hat{y}_{i^*} \neq f'(i^*)$. Then, as $\mathcal{A}$ passes the game,

- $\mathsf{MT.Verify}(\mathsf{com}_{\mathsf{MT}}, i^*, (i^*, m'_{i^*}, \pi'_{\mathsf{VC},i^*}, y'_{i^*}), \pi'_{\mathsf{MT},i^*}) = 1$,
- $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i^*, m'_{i^*}, \pi'_{\mathsf{VC},i^*}) = 1$,
- $\mathsf{PC.Verify}(\mathsf{com}_{\mathsf{PC}}, i^*, y'_{i^*}, \pi'_{\mathsf{PC},i^*}) = 1$.

Now, consider the following two cases:

- $\hat{y}_{i^*} \neq y'_{i^*}$: In this case, $(i^*, m'_{i^*}, \pi'_{\mathsf{VC},i^*}, y'_{i^*}) \neq (i^*, \hat{m}_{i^*}, \hat{\pi}_{\mathsf{VC},i^*}, \hat{y}_{i^*})$ both verify with respect to $\mathsf{com}_{\mathsf{MT}}$, which by the position-binding property of Merkle commitments, happens with probability at most $\epsilon_{\mathsf{MT}} = \mathsf{negl}(\lambda)$.
- $\hat{y}_{i^*} = y'_{i^*}$: In this case, $y'_{i^*} = \hat{y}_{i^*} \neq f'(i^*)$, yet it holds that

  $$\mathsf{PC.Verify}(\mathsf{com}_{\mathsf{PC}}, i^*, y'_{i^*}, \pi'_{\mathsf{PC},i^*}) = 1 \text{ and } \mathsf{PC.Verify}(\mathsf{com}_{\mathsf{PC}}, i^*, f'(i^*), \pi_{\mathsf{PC},i^*}) = 1,$$

  where π_{PC,i^*} is the opening proof created with respect to $\mathsf{com}_{\mathsf{PC}}$ for the polynomial $f'(X)$ and randomness r. However, by the evaluation-binding property of the PC, this happens with probability at most $\epsilon_{\mathsf{PC}} = \mathsf{negl}(\lambda)$.

Consequently, event 4 happens with probability at most $\epsilon_{\mathsf{MT}} + \epsilon_{\mathsf{PC}}$.

Finally, we observe that the extractor outputs over $(1 - \nu(k))n$ leaf nodes (and Merkle opening proofs) that satisfy the points above except with probability $\mathsf{negl}(\lambda)$. This concludes the proof. □

Defining the adversaries $\mathcal{A}_1 := (\mathbf{S}_1, \ldots, \mathbf{S}_\ell, \mathbf{R}_1)$ and $\mathcal{A}_2 := (\mathbf{S}_1, \ldots, \mathbf{S}_\ell, \mathbf{R}_2)$, by Lemma 2, $(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$ and $\mathbf{R}_1$ together must have queried the random oracle with over $(1 - \nu(k))$-fraction of the leaf nodes before the challenge phase, and except with $\mathsf{negl}(\lambda)$ probability, for over $(1 - \nu(k))n$ queried leaves denoted by $\{(a_i, \hat{m}_{a_i}, \hat{\pi}_{\mathsf{VC},a_i}, \hat{y}_{a_i})\}_{i \in [\ell_1]}$, it holds that

- $\mathsf{MT.Verify}(\mathsf{com}_{\mathsf{MT}}, a_i, (a_i, \hat{m}_{a_i}, \hat{\pi}_{\mathsf{VC},a_i}, \hat{y}_{a_i}), \hat{\pi}_{\mathsf{MT},a_i}) = 1$,
- $\hat{m}_{a_i} = m_{a_i}$,
- $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, a_i, \hat{m}_{a_i}, \hat{\pi}_{\mathsf{VC},a_i}) = 1$,
- $\hat{y}_{a_i} = f'(a_i)$.

We next construct the extractors $\mathsf{Ext}^{\mathbf{S}_j}$ and $\mathsf{Ext}^{\mathbf{R}_1}$:

$\mathsf{Ext}^{\mathbf{R}_1}(\mathsf{pk}, \mathsf{com}, \langle \mathbf{R}_1, (\mathbf{S}_1, \ldots, \mathbf{S}_\ell), \rangle) \to m''$: Unlike in the proof of Lemma 1, an extractor cannot follow a "top-down" approach to arrive at the leaf nodes, which identifies each leaf query. Indeed, $\mathbf{R}_1$ can for example delegate computation of the intermediate nodes in the Merkle tree without compromising the privacy of its secret key, as in a hiding Merkle commitment, the leaves are committed with randomness. However, when leaf nodes are contained in the random oracle queries made by $\mathbf{R}_1$, the extractor can identify them efficiently by relying on the VC proofs contained in the leaf nodes. More formally, $\mathsf{Ext}^{\mathbf{R}_1}$ does the following:

- Maintain a list $\mathcal{L}$ of tuples from $[n] \times \mathcal{M}$ containing the entries of m that have been recovered so far.
- Parse $\mathsf{com} = (\mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{PC}}, \mathsf{com}_{\mathsf{VC}})$.
- Let Q_R denote the random oracle queries made by $\mathbf{R}_1$. Try parsing each query as $(i, \bar{m}_i, \bar{\pi}_{\mathsf{VC},i}, \bar{y}_i)$, and if $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i, \bar{m}_i, \bar{\pi}_{\mathsf{VC},i}) = 1$, add it to the list $\mathcal{L} \leftarrow \mathcal{L} \cup \{(i, \bar{m}_i)\}$. If some $(i, \bar{m}'_i)$ with $\bar{m}'_i \neq \bar{m}_i$ already exists in the list, abort.
- Define m'' where for $i \in [n]$

  $$m''_i = \begin{cases} \bar{m}_i & \text{if } (i, \bar{m}_i) \in \mathcal{L}, \\ \perp & \text{otherwise} \end{cases}$$

- Output m''.

By the position-binding property of VC, it is infeasible for the adversary to produce conflicting openings for $\mathsf{com}_{\mathsf{VC}}$ at the same position i, and hence $\mathsf{Ext}^{\mathbf{R}_1}$ does not abort except with negligible probability. Furthermore, it must be that $m''_{a_i} = m_{a_i}$ for all $i \in [\ell_1]$, $a_i \in \mathcal{L}$, except with negligible probability; since (i) $\hat{m}_{a_i} = m_{a_i}$ for all $i \in [\ell_1]$, and, (ii) as for all $i \in [\ell_1]$, $a_i \in \mathcal{L}$,

$$\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, a_i, m''_{a_i}, \bar{\pi}_{\mathsf{VC},a_i}) = \mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, a_i, \hat{m}_{a_i}, \hat{\pi}_{\mathsf{VC},a_i}) = 1,$$

by the position-binding property of VC, $m''_{a_i} = \hat{m}_{a_i}$ for these indices. Recalling that $\mu(k) = 1 - \nu(k) - k/n$, if $|\{i \in [\ell_1] \mid m''_{a_i} = m_{a_i}\}| > (1-\mu(k))n$, then $\mathcal{A}$ loses the $\mu(k)$-soundness game. We next show that if this is not the case, then one of $\{\mathsf{Ext}^{\mathbf{S}_j}\}_{j\in[\ell]}$ can recover the secret key.

$\mathsf{Ext}^{\mathbf{S}_j}(\mathsf{pk}, \mathsf{com}, \langle \mathbf{S}_j, \mathbf{R}_1\rangle, \langle \mathbf{S}_j, \mathbf{R}_2, \{(m'_i, \pi_i)\}_{i\in\mathsf{chal}}\rangle) \to \mathsf{sk}$: This extractor monitors the random oracle queries for valid leaf nodes by $\mathbf{S}_j$. If one of them is outside of chal, then using the collected $k+1$ points y_i output by $\mathbf{R}_2$, the interpolating polynomial reveals the secret key.

- Maintain a list $\mathcal{L}$ of tuples from $[n] \times \mathcal{M} \times \mathbb{Z}_p$ containing the set of evaluations that have been recovered.
- Parse $\mathsf{com} = (\mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{PC}}, \mathsf{com}_{\mathsf{VC}})$.
- For $i \in \mathsf{chal}$, parse $\mathbf{R}_2$'s output m'_i and $\pi_i = (y_i, \pi_{\mathsf{PC},i}, \pi_{\mathsf{MT},i}, \pi_{\mathsf{VC},i}, \pi_{\mathsf{NIZK},i})$.
- Let Q_S denote the random oracle queries made by $\mathbf{S}_j$. Try parsing each query as $(i, \bar{m}_i, \bar{\pi}_{\mathsf{VC},i}, \bar{y}_i)$, and if $i \notin \mathsf{chal}$ and $\mathsf{VC.Verify}(\mathsf{com}_{\mathsf{VC}}, i, \bar{m}_i, \bar{\pi}_{\mathsf{VC},i}) = 1$, add it to the list $\mathcal{L} \leftarrow \mathcal{L} \cup \{(i, \bar{m}_i, \bar{y}_i)\}$.
- If $\mathcal{L} = \emptyset$, abort.
- Else for all entries $(i, \bar{m}_i, \bar{y}_i) \in \mathcal{L}$: Interpolate a polynomial $f_i(X)$ through the point $(i, \bar{y}_i)$ as well as $\{(i, y_i)\}_{i\in\mathsf{chal}}$. If $(\mathsf{pk}, f_i(0)) \in \mathcal{R}$ then stop and output $f_i(0)$.
- Else, abort.

Clearly, $\mathsf{Ext}^{\mathbf{S}_j}$ runs in polynomial time.

As in the case of $\mathbf{R}_1$, by the position-binding property of the vector commitment scheme, the extractor does not abort when adding entries to $\mathcal{L}$ except with negligible probability. Moreover, as $(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$ and $\mathbf{R}_1$ together must have queried the random oracle with the Merkle tree leaves $\{(a_i, \hat{m}_{a_i}, \hat{\pi}_{\mathsf{VC},a_i}, \hat{y}_{a_i})\}_{i\in[\ell_1]}$, where $\hat{y}_{a_i} = f'(a_i)$ except with negligible probability, if $|\{i \in [\ell_1] \mid m_{a_i} = \hat{m}_{a_i}\}| \le (1-\mu(k))n$, $\mu(k) = 1-\nu(k)-k/n$, there exist a sender $\mathbf{S}_j$ such that one of the polynomials interpolated by $\mathbf{S}_j$ is $f_i(X) = f'(X)$. Thus, for the interpolated polynomial, it holds that $(\mathsf{pk}, f_i(0)) \in \mathcal{R}$. Hence, if $\mathsf{Ext}^{\mathbf{R}_1}$ outputs at most $(1-\mu(k))n$ points, a value sk such that $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{R}$ can be extracted from $\mathbf{S}_j$'s random oracle queries except with negligible probability. This implies that $\mathcal{A}$ cannot pass the soundness game except with negligible probability (Fig. 1) and concludes the proof showing that the construction in Section 6.1 is a secure non-delegatable commitment scheme. □

As the proof of Theorem 3 uses the q-Bilinear Strong Diffie-Hellman assumption, we define it below:

Definition 10. *The q-Bilinear Strong Diffie-Hellman assumption, denoted by q-BSDH and parameterized by $q \in \mathbb{N}$, states that for any PPT adversary $\mathcal{A}$,*

$$\Pr\left[t = \mathsf{e}(g,h)^{\frac{1}{\tau+c}} \;\middle|\; \tau \leftarrow \mathbb{Z}_p,\ (c,t) \leftarrow \mathcal{A}\big(g, g^{\tau}, \ldots, g^{\tau^q}, h\big)\right] = \mathsf{negl}(\lambda)$$

for any value of $c \in \mathbb{Z}_p \setminus \{-\tau\}$.

Theorem 3. *If VC is a computationally position-binding vector commitment scheme, and the $(k+1)$-DLOG assumption holds, then the construction in Section 6.2 satisfies non-delegatability with $\mu(k) = (2^{-\lambda/k} - k/n)$-soundness in the random oracle model against algebraic adversaries.*

Proof. Consider a PPT adversary $\mathcal{A} = (\mathbf{S}_1, \ldots, \mathbf{S}_\ell, \mathbf{R}_1, \mathbf{R}_2)$ such that for the messages m, commitment com and proof π output by $\mathcal{A}$, it holds that $m'_i = m_i$ and $\mathsf{Verify}(\mathsf{pk}, \mathsf{com}, i, m_i, \pi_i) = 1$ for all $i \in \mathsf{chal}$. Let $\mathsf{com} = (\mathsf{com}_{\mathsf{MT}}, \mathsf{com}_{\mathsf{PC}}, \mathsf{com}_{\mathsf{VC}})$ and $\pi_i = (f(i), \pi_{\mathsf{PC},i}, \pi_{\mathsf{MT},i}, \pi_{\mathsf{VC},i}, \pi_{\mathsf{NIZK}})$. Again by Lemma 1, we know that $(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$ and $\mathbf{R}_1$ together must have queried the random oracle with over $(1-\nu(k))$-fraction of the leaf nodes, which verify with respect to $\mathsf{com}_{\mathsf{MT}}$, before the challenge phase, except with probability $\epsilon_{\mathsf{MT}} = \mathsf{negl}(\lambda)$. Let $\{(a_i, \hat{m}_{a_i}, \hat{\pi}_{\mathsf{VC},a_i}, \hat{y}_{a_i})\}_{i\in[\ell_1]}$ denote the queried leaves, where $\ell_1 > (1-\nu(k))n$, and note that for all $i \in [\ell_1]$

$$\mathsf{MT.Verify}(\mathsf{com}_{\mathsf{MT}}, a_i, (a_i, \hat{m}_{a_i}, \hat{\pi}_{\mathsf{VC},a_i}, \hat{y}_{a_i}), \pi'_{\mathsf{MT},a_i}) = 1$$

By the extractability of the KZG scheme, a polynomial $f'(X)$ along with its randomness r', can be extracted from the KZG opening proof $\pi_{\mathsf{KZG},i}$ for some $i \in \mathsf{chal}$, using the expected-polynomial-time extractor for the following relation:

$$\left\{(\mathsf{com}_{\mathsf{PC}}, x, y) \quad | \quad \exists\, (f(X), r) \in \mathbb{Z}_p^{\leq k} \times \{0,1\}^* \;:\; \mathsf{PC.Commit}(f(X); r) = \mathsf{com}_{\mathsf{PC}} \wedge f(x) = y\right\}$$

Let $f'(X)$ denote the output of this extractor, and note that $f'(X) \in \mathbb{Z}_p^{\leq k}[X]$, $\mathsf{com}_{\mathsf{PC}} = \mathsf{PC.Commit}(f'(X); r')$, and $f'(i) = f(i)$.

The rest of the proof proceeds as in the proof of Theorem 2, except that for $f'(0)$, we prove that $(\mathsf{pk}, f'(0)) \in \mathcal{R}$ as follows: For contradiction, suppose $(\mathsf{pk}, f'(0)) \notin \mathcal{R}$, *i.e.*, $g^{f'(0)} = \overline{\mathsf{pk}}^1 \neq \mathsf{pk}^1$ with non-negligible probability. Define $(\overline{\pi}^1_{\mathsf{KZG},0}, \overline{\pi}^2_{\mathsf{KZG},0})$ as the opening proof of the polynomial $f'(X)$ with randomness r' at point 0. Recall the proof $\pi_{\mathsf{KZG},0} = (\pi^1_{\mathsf{KZG},0}, \pi^2_{\mathsf{KZG},0})$ output as part of the proofs π_i. Then,

$$\begin{aligned}
\mathsf{e}(\mathsf{com}_{\mathsf{KZG}}/\mathsf{pk}^1, h) &= \mathsf{e}(\pi^1_{\mathsf{KZG},0}, h^\tau) \cdot \mathsf{e}(\pi^2_{\mathsf{KZG},0}, h^\eta),\\
\mathsf{e}(\mathsf{com}_{\mathsf{KZG}}/\overline{\mathsf{pk}}^1, h) &= \mathsf{e}(\overline{\pi}^1_{\mathsf{KZG},0}, h^\tau) \cdot \mathsf{e}(\overline{\pi}^2_{\mathsf{KZG},0}, h^\eta)
\end{aligned}$$

By dividing both equations, $\mathcal{A}$ can obtain;

$$\begin{aligned}
&\mathsf{e}\left(\frac{\overline{\mathsf{pk}}^1}{\mathsf{pk}^1}, h\right) = \mathsf{e}\left(\frac{\pi^1_{\mathsf{KZG},0}}{\overline{\pi}^1_{\mathsf{KZG},0}}, h^\tau\right) \cdot \mathsf{e}\left(\frac{\pi^2_{\mathsf{KZG},0}}{\overline{\pi}^2_{\mathsf{KZG},0}}, h^\eta\right)\\
\Longrightarrow\; &\mathsf{e}\left(\frac{\overline{\mathsf{pk}}^1}{\mathsf{pk}^1}, h\right) \cdot \mathsf{e}\left(\frac{\overline{\pi}^2_{\mathsf{KZG},0}}{\pi^2_{\mathsf{KZG},0}}, h^\eta\right) = \mathsf{e}\left(\left(\frac{\pi^1_{\mathsf{KZG},0}}{\overline{\pi}^1_{\mathsf{KZG},0}}\right)^\tau, h\right)\\
\Longrightarrow\; &\mathsf{e}\left(\frac{\pi^1_{\mathsf{KZG},0}}{\overline{\pi}^1_{\mathsf{KZG},0}}, h\right) \cdot \mathsf{e}\left(\frac{\pi^2_{\mathsf{KZG},0}}{\overline{\pi}^2_{\mathsf{KZG},0}}, h^\eta\right)^{\frac{1}{\tau}} = \mathsf{e}\left(\frac{\overline{\mathsf{pk}}^1}{\mathsf{pk}^1}, h\right)^{\frac{1}{\tau}}
\end{aligned}$$

Finally, consider an adversary $\mathcal{B}$ that receives a powers-of-τ setup, samples a uniformly random $\alpha \leftarrow \mathbb{Z}_p$ and a uniformly-random $\mathsf{sk} \leftarrow \mathbb{Z}_p$, and runs $\mathcal{A}$ with $\mathsf{pk}^1 := g^{\mathsf{sk}}$, $\mathsf{pk}^2 := h^{\mathsf{sk}}$, and the $\mathsf{crs}_{\mathsf{KZG}}$ set to be the powers-of-τ along with $g^\eta := (g^\tau)^\alpha$ and $h^\eta := (h^\tau)^\alpha$ (this $\mathsf{crs}_{\mathsf{KZG}}$ has the same distribution as an honestly generated $\mathsf{crs}_{\mathsf{KZG}}$). Then, for the opening proofs found by $\mathcal{A}$, the expression above becomes

$$\begin{aligned}
&\mathsf{e}\left(\frac{\pi^1_{\mathsf{KZG},0}}{\overline{\pi}^1_{\mathsf{KZG},0}} \cdot \left(\frac{\pi^2_{\mathsf{KZG},0}}{\overline{\pi}^2_{\mathsf{KZG},0}}\right)^\alpha, h\right) = \mathsf{e}\left(\frac{\overline{\mathsf{pk}}^1}{\mathsf{pk}^1}, h\right)^{\frac{1}{\tau}}\\
\Longrightarrow\; &\mathsf{e}\left(\frac{\pi^1_{\mathsf{KZG},0}}{\overline{\pi}^1_{\mathsf{KZG},0}} \cdot \left(\frac{\pi^2_{\mathsf{KZG},0}}{\overline{\pi}^2_{\mathsf{KZG},0}}\right)^\alpha, h\right)^{1/(f'(0)-\mathsf{sk})} = \mathsf{e}\,(g,h)^{\frac{1}{\tau}},
\end{aligned}$$

where $\mathsf{sk} \neq f'(0)$ by assumption (which happens with non-negligible probability), and $\mathcal{B}$ can calculate the left-hand side to find $e(g,h)^{1/\tau}$. In other words, $\mathcal{A}$ breaks $(k+1)$-BSDH, which is implied by the $(k+1)$-DLOG assumption in AGM [BFL20]. Since this happens with negligible probability, a value $\mathsf{sk} := f'(0)$ such that $(\mathsf{pk}, \mathsf{sk}) \in \mathcal{R}$ can be extracted from $\mathbf{S}_j$'s random oracle queries with non-negligible probability. □

B Proof of Data Dissemination with Guaranteed Transmission

Definition 11 (Provable Data Dissemination Scheme with Guaranteed Transmission (PDD-GT))**.** *A PDD-GT scheme is characterized by the following eleven algorithms*

$$(\mathsf{Setup}, \mathsf{KeyGen}_{\mathbf{S}}, \mathsf{KeyGen}_{\mathbf{R}}, \mathsf{Digest}, \mathsf{ProveDigest}, \mathsf{Encode}, \mathsf{Decode}, \mathsf{Commit}, \mathsf{Challenge}, \mathsf{Prove}, \mathsf{Verify}).$$

Since Setup*,* $\mathsf{KeyGen}_{\mathbf{R}}$*,* Digest*,* ProveDigest*,* Challenge *and* Prove *are defined exactly as in Definition 5, their definitions are omitted below.*

- $\mathsf{KeyGen}_{\mathbf{S}}(1^\lambda; r) \to (\mathsf{pk}_{\mathbf{S}}, \mathsf{sk}_{\mathbf{S}})$: *The sender's key generation algorithm is a randomized algorithm run by the sender that takes a security parameter and uniformly-random bits r as input and outputs a key pair $(\mathsf{pk}_{\mathbf{S}}, \mathsf{sk}_{\mathbf{S}}) \in \mathcal{R}_{\mathbf{S}}$. Here, $\mathcal{R}_{\mathbf{S}}$ is an NP relation such that $\mathsf{sk}_{\mathbf{S}}$ is a witness attesting to the fact that $\mathsf{pk}_{\mathbf{S}}$ is a true statement.*
- $\mathsf{Encode}(\mathsf{sk}_{\mathbf{S}}, (m_{a_i})_{i\in[\ell]}) \to (c_{a_i})_{i\in[\ell]} \in (\mathcal{C})^n$: *The encoding algorithm is a randomized algorithm run by the sender that takes the sender's secret key $\mathsf{sk}_{\mathbf{S}}$ and (possibly only a subset of) the messages $(m_i)_{i\in[n]}$ as input and outputs a* locally-decodable *encoding $(c_i)_{i\in[n]}$ of the data. The encoding operation crucially relies on knowledge of the secret key $\mathsf{sk}_{\mathbf{S}}$.*
- $\mathsf{Decode}(\mathsf{pk}_{\mathbf{S}}, (c_{a_i})_{i\in[\ell]}) \to (m_{a_i})_{i\in[\ell]}$: *The decoding algorithm is a deterministic algorithm run by the receiver that takes the sender's public key $\mathsf{pk}_{\mathbf{S}}$ and (possibly only a subset of) the encoding $(c_i)_{i\in[n]}$ as input and outputs some data $(m_{a_i})_{i\in[\ell]}$.*
- $\mathsf{Commit}(\mathsf{sk}_{\mathbf{R}}, \mathsf{pk}_{\mathbf{S}}, \mathsf{dig}, (c_{a_i})_{i\in[\ell]}, \pi_{\mathsf{dig}}; r) \to (\mathsf{com}, \mathsf{data}_{\mathsf{com}})$: *The commitment algorithm is a randomized algorithm run by the receiver* after *receiving (possibly only a subset of) the encodings and the decoded messages. It outputs a commitment* com *to the sequence of ℓ encodings and a data structure* data*, both of which crucially depend on the knowledge of the secret key $\mathsf{sk}_{\mathbf{R}}$.*
- $\mathsf{Verify}(\mathsf{pk}_{\mathbf{S}}, \mathsf{pk}_{\mathbf{R}}, \mathsf{dig}, \mathsf{com}, \mathsf{chal}, \pi) \to \{0, 1\}$: *The verification algorithm is a deterministic algorithm that outputs 1 if the given proof π verifies against the public keys $\mathsf{pk}_{\mathbf{S}}$, $\mathsf{pk}_{\mathbf{R}}$, the digest* dig*, the commitment* com*, and the challenge* chal*.*

A PDD-GT scheme satisfies sender and receiver correctness, soundness, receiver privacy, sender privacy, succinctness, and guaranteed delivery.

Definition 12 (PDD-GT Correctness)**.** *For all $\lambda \in \mathbb{Z}^+$, $n = \mathsf{poly}(\lambda)$, messages $(m_i)_{i\in[n]} \in \mathcal{M}^n$, and subsets $(a_i)_{i\in[\ell]} \subseteq [n]$,*

$$\Pr\left[\begin{array}{l|l} \mathsf{Verify}(\mathsf{pk}_{\mathbf{R}}, \mathsf{dig}, \mathsf{com}, \mathsf{chal}, \pi) = 1 \\ \mathsf{Decode}(\mathsf{pk}_{\mathbf{S}}, (c_{a_i})_{i\in[\ell]}) = (m_{a_i})_{i\in[\ell]} & \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \\ (\mathsf{pk}_{\mathbf{R}}, \mathsf{sk}_{\mathbf{R}}) \leftarrow \mathsf{KeyGen}_{\mathbf{R}}(1^\lambda), \\ (\mathsf{pk}_{\mathbf{S}}, \mathsf{sk}_{\mathbf{S}}) \leftarrow \mathsf{KeyGen}_{\mathbf{S}}(1^\lambda), \\ \mathsf{dig} \coloneqq \mathsf{Digest}((m_i)_{i\in[n]}), \\ \pi_{\mathsf{dig}} \coloneqq \mathsf{ProveDigest}(\mathsf{data}_{\mathsf{dig}}, (a_i)_{i\in[\ell]}), \\ (c_i)_{i\in[n]} \leftarrow \mathsf{Encode}(\mathsf{sk}_{\mathbf{S}}, (m_i)_{i\in[n]}) \\ (\mathsf{com}, \mathsf{data}) \leftarrow \mathsf{Commit}(\mathsf{sk}_{\mathbf{R}}, \mathsf{pk}_{\mathbf{S}}, \mathsf{dig}, (c_{a_i})_{i\in[\ell]}, \pi_{\mathsf{dig}}), \\ \mathsf{chal} \leftarrow \mathsf{Challenge}(\mathsf{pk}_{\mathbf{R}}, \mathsf{dig}, \mathsf{com}) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{sk}_{\mathbf{R}}, \mathsf{data}_{\mathsf{com}}, \mathsf{chal}) \end{array} \end{array}\right] = 1$$

Definition 13 (PDD-GT Receiver Soundness (Guaranteed Reception))**.** *A PDD scheme is said to be receiver sound if for all $\lambda \in \mathbb{Z}^+$, $\mu > 0$, and PPT adversaries $\mathcal{A}$, there exists a polynomial-time extractor* Ext *such that*

$$\Pr\left[\begin{array}{l|l} \mathsf{Soundness}_{\mathcal{A}}(1^\lambda, 1^n) = 1 \wedge \\ (\mathsf{pk}_{\mathbf{R}}, \mathsf{sk}_{\mathbf{R}}) \notin \mathcal{R}_{\mathbf{R}} \wedge \\ (\mathsf{Decode}(\mathsf{pk}_{\mathbf{S}}, (c'_{b_i})_{i\in[\ell']}) \not\subseteq (m_i)_{i\in[n]} \vee \ell' < (1-\mu)\ell) & \begin{array}{l} \mathsf{Ext}^{\mathbf{R}}(\mathsf{pk}_{\mathbf{R}}, \mathsf{dig}, \mathsf{com}, \langle \mathbf{S}, \mathbf{R}\rangle) \to (c'_{b_i})_{i\in[\ell']} \\ \mathsf{Ext}^{\mathbf{S}}(\mathsf{pk}_{\mathbf{R}}, \mathsf{dig}, \mathsf{com}, \langle \mathbf{S}, \mathbf{R}\rangle) \to \mathsf{sk}_{\mathbf{R}} \end{array} \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where the probability is taken over the randomness used to sample chal *and the randomness tapes of the algorithms* $\mathcal{A}$*,* $\mathbf{S}$ *and* $\mathbf{R}$*. The soundness game is the same as* **??***, except that the adversary* $\mathcal{A}^{\mathsf{Setup}}$ *also outputs and sends* $\mathsf{pk}_{\mathbf{S}}$ *to the challenger* $\mathcal{C}$ *at step 2.*

Definition 14 (PDD-GT Receiver Privacy)**.** *There exists a PPT simulator* $\mathcal{S}$ *such that for all* $\lambda \in \mathbb{Z}^+$*,* $n = \mathsf{poly}(\lambda)$*, and PPT adversaries* $\mathcal{A}$*;*

$$\Pr\left[\mathcal{A}(\mathsf{crs}_b, \mathsf{pk}_{\mathbf{R},b}, \mathsf{com}_b, \pi_b) = b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ \mathsf{crs}_0 \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ (\mathsf{pk}_{\mathbf{R},0}, \mathsf{sk}_{\mathbf{R}}) \leftarrow \mathsf{KeyGen}_{\mathbf{R}}(1^\lambda) \\ (\mathsf{pk}_{\mathbf{S}}, \mathsf{dig}, (m_i)_{i\in[n]}, (c_i)_{i\in[n]}, \pi_{\mathsf{dig}}) \leftarrow \mathcal{A}(\mathsf{crs}_0) \\ (\mathsf{com}_0, \mathsf{data}_{\mathsf{com}}) \leftarrow \mathsf{Commit}(\mathsf{sk}_{\mathbf{R}}, (c_{a_i})_{i\in[\ell]}, (m_{a_i})_{i\in[\ell]}) \\ \mathsf{chal}_0 \leftarrow \mathsf{Challenge}(\mathsf{pk}_{\mathbf{R},0}, \mathsf{dig}, \mathsf{com}_0) \\ \pi_0 \leftarrow \mathsf{Prove}(\mathsf{crs}_0, \mathsf{sk}_{\mathbf{R}}, \mathsf{com}, \mathsf{data}, \mathsf{chal}), \\ \mathsf{crs}_1 \leftarrow \mathcal{S}(1^\lambda, 1^n), \\ (\mathsf{pk}_{\mathbf{R},1}, \mathsf{com}_1, \mathsf{chal}_1, \pi_1) \leftarrow \mathcal{S}(\mathsf{crs}_1, \mathsf{dig}, (m_{a_i})_{i\in[\ell]}, \mathsf{chal}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Sender Security $\mathsf{SenderSecurity}_{\mathcal{A}}(1^\lambda, 1^n)$

1. The challenger $\mathcal{C}$ runs $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$ and sends crs to the adversary $\mathcal{A}^{\mathsf{Setup}}$.
2. Upon receiving crs, $\mathcal{A}^{\mathsf{Setup}}$ outputs keys $\mathsf{pk}_{\mathbf{R}}$, $\mathsf{pk}_{\mathbf{S}}$, a digest dig, messages $(m_i)_{i\in[n]} \in \mathcal{M}^n$, a commitment com, and data $\mathsf{aux}_{\mathbf{S}}$ and $\mathsf{aux}_{\mathbf{R}}$. It sends $\mathsf{pk}_{\mathbf{R}}$, dig, com, and optionally the description of the algorithm Expand, *i.e.*, $\langle\mathsf{Expand}\rangle$ to $\mathcal{C}$.
3. $\mathcal{C}$ samples a challenge $\mathsf{chal} \leftarrow \mathsf{Challenge}(\mathsf{pk}_{\mathbf{R}}, \mathsf{dig}, \mathsf{com})$. If $\mathsf{chal} = \perp$, it outputs 0 and aborts the game. Otherwise, it sends chal to $\mathbf{S}$ and $\mathbf{R}$. Let $(a_i)_{i\in[\ell]}$ denote the subset of indices claimed by com when $\mathsf{chal} \neq \perp$.
4. $\mathbf{R}$ responds with a proof π, some messages $(m_{b_i})_{i\in[\ell']}$ and the associated opening proofs $(\pi_{\mathsf{dig},b_i})_{i\in[\ell']}$, and some input w to the algorithm Expand, computed via an interactive protocol: $(\pi, w) \leftarrow \langle\mathbf{S}(\mathsf{aux}_{\mathbf{S}}, \mathsf{chal}), \mathbf{R}(\mathsf{aux}_{\mathbf{R}}, \mathsf{chal})\rangle$.
5. $\mathcal{C}$ outputs 1 if
   (a) $\mathsf{Verify}(\mathsf{pk}_{\mathbf{S}}, \mathsf{pk}_{\mathbf{R}}, \mathsf{dig}, \mathsf{com}, \mathsf{chal}, \pi) = 1$.
   (b) $|w| \leq \beta(\lambda, k)\ell'$.
   (c) $\mathsf{Decode}(\mathsf{pk}_{\mathbf{S}}, \mathsf{Expand}(w)) = (m_{b_i})_{i\in[\ell']}$.
   (d) $\mathsf{VerifyDigest}(\mathsf{dig}, b_i, m_{b_i}, \pi_{\mathsf{dig},b_i}) = 1$ for all $i \in [\ell']$.

Figure 2: The guaranteed transmission game with an adversary $\mathcal{A} = (\mathcal{A}^{\mathsf{Setup}}, \mathbf{S}, \mathbf{R})$, where $\mathcal{M} = \{0,1\}^k$. It uses an additional algorithm $\mathsf{VerifyDigest}(\mathsf{dig}, i, m_i, \pi_{\mathsf{dig}})$. Altogether, Digest, ProveDigest and VerifyDigest constitute as secure VC scheme.

Definition 15 (PDD-GT Sender Privacy)**.** *There exists a PPT simulator* $\mathcal{S}$ *such that for all* $\lambda \in \mathbb{Z}^+$*,* $n = \mathsf{poly}(\lambda)$*,*

and PPT adversaries $\mathcal{A}$;

$$\Pr\left[\begin{array}{l}\mathcal{A}(\mathsf{crs}_b, \mathsf{pk}_{\mathbf{S},b}, \mathsf{dig}_b, \pi_{\mathsf{dig},b}, (c_{a_i,b})_{i\in[\ell]}) = b\end{array}\;\middle|\;\begin{array}{l}b \leftarrow \{0,1\}\\ \mathsf{crs}_0 \leftarrow \mathsf{Setup}(1^\lambda, 1^n)\\ (\mathsf{pk}_{\mathbf{S},0}, \mathsf{sk}_{\mathbf{S}}) \leftarrow \mathsf{KeyGen}(1^\lambda)\\ ((m_i)_{i\in[n]}, (a_i)_{i\in[\ell]}) \leftarrow \mathcal{A}(\mathsf{crs}_0)\\ (\mathsf{dig}_0, \mathsf{data}_{\mathsf{dig}}) \leftarrow \mathsf{Digest}((m_i)_{i\in[n]})\\ \pi_{\mathsf{dig},0} \leftarrow \mathsf{ProveDigest}(\mathsf{data}_{\mathsf{dig}}, (a_i)_{i\in[\ell]})\\ (c_{a_i,0})_{i\in[\ell]} \leftarrow \mathsf{Encode}(\mathsf{sk}_{\mathbf{S}}, (m_{a_i})_{i\in[\ell]})\\ \mathsf{crs}_1 \leftarrow \mathcal{S}(1^\lambda, 1^n),\\ (\mathsf{pk}_{\mathbf{S},0}, \mathsf{dig}_0, \pi_{\mathsf{dig},0}, (c_{a_i,0})_{i\in[\ell]}) \leftarrow \mathcal{S}(\mathsf{crs}_1),\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

C Our Provable Data Dissemination Scheme with Guaranteed Transmission

We describe our generic construction with a blackbox use of vector and polynomial commitment schemes along with an incompressible encoding scheme in Figure 3. The scheme can be instantiated using hiding Merkle trees and KZG polynomial commitments as in **??**.

Our Generic Construction with Guaranteed Transmission

$\mathsf{KeyGen}_{\mathbf{S}}(1^\lambda) \to (\mathsf{pk}_{\mathbf{R}}, \mathsf{sk}_{\mathbf{R}})$: The key generation algorithm run by the sender outputs a key pair $(\mathsf{pk}_{\mathbf{S}}, \mathsf{sk}_{\mathbf{S}}) \leftarrow \mathsf{IE.KeyGen}(1^\lambda)$.

$\mathsf{Encode}(\mathsf{sk}_{\mathbf{S}}, (m_{a_i})_{i\in[\ell]}) \to (c_{a_i})_{i\in[\ell]}$: The encoding algorithm run by the sender computes $c_{a_i} \leftarrow \mathsf{IE.Encode}(\mathsf{sk}_{\mathbf{S}}, m_{a_i})$ for $i \in [\ell]$ and outputs $(c_{a_i})_{i\in[\ell]}$.

$\mathsf{Decode}(\mathsf{pk}_{\mathbf{S}}, (c_{a_i})_{i\in[\ell]}) \to (m_{a_i})_{i\in[\ell]}$: The decoding algorithm run by the receiver computes $m_{a_i} \leftarrow \mathsf{IE.Decode}(\mathsf{pk}_{\mathbf{S}}, c_{a_i})$ for $i \in [\ell]$ and outputs $(m_{a_i})_{i\in[\ell]}$.

$\mathsf{Commit}(\mathsf{sk}_{\mathbf{R}}, \mathsf{pk}_{\mathbf{S}}, \mathsf{dig}, (c_{a_i})_{i\in[\ell]}, \pi_{\mathsf{dig}}; r) \to (\mathsf{com}, \mathsf{data}_{\mathsf{com}})$: The commitment algorithm run by the receiver first computes $(m_i)_{i\in[\ell]} \leftarrow \mathsf{Decode}(\mathsf{pk}_{\mathbf{S}}, (c_{a_i})_{i\in[\ell]})$ and then checks π_{dig} against dig and $(m_{a_i})_{i\in[\ell]}$: $\mathsf{VC}_1.\mathsf{Verify}(\mathsf{dig}, a_i, m_{a_i}, \pi_{\mathsf{VC}_1,a_i})$ for $i \in [\ell]$ If any of the checks fail, it outputs $\mathsf{com} = \perp$. Otherwise, it samples a degree-d polynomial $f(X) \in \mathbb{Z}_p^{\leq d}[X]$ uniformly at random, subject to the constraint $f(0) = \mathsf{sk}_{\mathbf{R}}$. For $i \in [k]$, it computes

$$(\mathsf{com}_{\mathsf{PC}}, \mathsf{data}_{\mathsf{PC}}) \leftarrow \mathsf{PC.Commit}(f(X); r)$$
$$(\mathsf{com}_{\mathsf{VC}_2}, \mathsf{data}_{\mathsf{VC}_2}) \leftarrow \mathsf{VC}_2.\mathsf{Commit}((H(c_{a_i} \| f(a_i)))_{i\in[\ell]}; r),$$

and outputs $\mathsf{com} := (\mathsf{com}_{\mathsf{VC}_2}, \mathsf{com}_{\mathsf{PC}}, (a_i)_{i\in[\ell]})$ and $\mathsf{data}_{\mathsf{com}} := ((c_{a_i}, \pi_{\mathsf{VC}_1,a_i})_{i\in[\ell]}, \mathsf{data}_{\mathsf{VC}_2}, \mathsf{data}_{\mathsf{PC}})$.

$\mathsf{Prove}(\mathsf{sk}_{\mathbf{R}}, \mathsf{com}, \mathsf{data}_{\mathsf{com}}, \mathsf{chal}; r) \to \pi$: If $\mathsf{com} = \perp$ or $\mathsf{chal} = \perp$, then it returns $\pi := \perp$. Let $\mathsf{chal} := (x_i)_{i\in[k]} \subseteq (a_i)_{i\in[\ell]}$ for $k \leq d$ (otherwise, the challenge is not well-formed). The proving algorithm run by the receiver first computes the following opening proofs for $i \in [k]$:

$$\pi_{\mathsf{PC},x_i} \leftarrow \mathsf{PC.Open}(\mathsf{data}_{\mathsf{PC}}, x_i; r)$$

It then computes a NIZK-AoK proof $\pi_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Prove}(x, w)$ for the following relation:

$$\{x = (\mathsf{pk}_{\mathbf{R}}, \mathsf{com}_{\mathsf{PC}}), w = (\mathsf{sk}_{\mathbf{R}}, \pi_{\mathsf{PC},0}) : (\mathsf{pk}_{\mathbf{R}}, \mathsf{sk}_{\mathbf{R}}) \in \mathcal{R}_{\mathbf{R}} \wedge \mathsf{PC.Verify}(\mathsf{com}_{\mathsf{PC}}, 0, \mathsf{sk}_{\mathbf{R}}, \pi_{\mathsf{PC},0}) = 1\}.$$

It finally outputs

$$\pi = (\pi_{\mathsf{NIZK}}, (x_i, c_{x_i}, f(x_i), \pi_{\mathsf{PC},x_i}, \pi_{\mathsf{VC}_1,x_i}, \pi_{\mathsf{VC}_2,x_i})_{i\in[k]}).$$

$\mathsf{Verify}(\mathsf{pk}_{\mathbf{S}}, \mathsf{pk}_{\mathbf{R}}, \mathsf{dig}, \mathsf{com}, \mathsf{chal}, \pi) \to \{0, 1\}$: If any of com, chal or π is $\perp$, it outputs 1. Let

$$\pi := (\pi_{\mathsf{NIZK}}, (x_i, m_{x_i}, y_{x_i}, \pi_{\mathsf{PC},x_i}, \pi_{\mathsf{VC}_1,x_i}, \pi_{\mathsf{VC}_2,x_i})_{i\in[k]}).$$

The verification algorithm run by the verifier checks the NIZK-AoK proof, *i.e.*, $\mathsf{NIZK.Verify}(\mathsf{pk}_{\mathbf{R}}, \mathsf{com}_{\mathsf{PC}}, \pi_{\mathsf{NIZK}})$, and verifies the opening proofs for the polynomial and vector commitments:

$$\mathsf{PC.Verify}(\mathsf{com}_{\mathsf{PC}}, x_i, y_{x_i}, \pi_{\mathsf{PC},x_i}) \quad \text{for} \quad i \in [k]$$
$$\mathsf{VC}_2.\mathsf{Verify}(\mathsf{com}_{\mathsf{VC}_2}, x_i, H(\mathsf{Decode}(\mathsf{pk}_{\mathbf{S}}, (c_{a_i})_{i\in[\ell]}) \| y_{x_i}), \pi_{\mathsf{VC}_2,x_i}) \quad \text{for} \quad i \in [k]$$

It outputs 1 if all checks verify, and 0 otherwise.

Figure 3: Our generic construction with guaranteed transmission using a NIZK-AoK, two vector commitment schemes VC_1 and VC_2 that support a proof of opening knowledge, a polynomial commitment scheme PC, and an incompressible encoding IE. Here, both VC_1 and VC_2 are randomized and satisfies the hiding property. Similarly, PC is a hiding PCS scheme. The Setup and Challenge algorithms remain the same and are omitted for brevity. The Digest and Open algorithms now take random bits r, but otherwise remain the same, and thus are also omitted.