Long-Term Secure Archival Using Proactively Secret-Shared Datastores

Research proficiency exam

Christopher Smith

Supervisors: Professor Erez Zadok, Professor Omkant Pandey

Department of Computer Science

State University of New York at Stony Brook

September 2024

Technical Report FSL-24-03

Abstract

Secure archival systems seek to efficiently and reliably preserve data confidentiality, integrity, and availability for long periods of time (on the order of a human lifespan or more). Such long-lived data is often valuable and highly sensitive; the consequences of information leakage or corruption could be catastrophic. To ensure long-term data security, we must contend with the combined threats of cryptographic obsolescence and Harvest Now, Decrypt Later attacks. These threats render inadequate traditional approaches to secure storage (e.g., encrypt then apply a message authentication code). We propose the use of information-theoretic techniques whose security does not depend on unproven assumptions of computational hardness. Specifically, we envision a geo-distributed and decentralized storage system that encodes data with secret-sharing schemes. Further, the system should support proactive share renewal with dynamic committees in the presence of Byzantine faults. In this work we explore the challenges that motivate our approach, describe the key ideas behind the associated techniques, and highlight promising research avenues that may lead to more efficient and secure long-term archival systems.

Keywords: Archival, secure storage, cryptography, information-theoretic security, proactive secret sharing, leakage-resilience, side-channel attacks, Byzantine fault-tolerance

Contents

1 Introduction

What is the safest and most reliable way of keeping a secret for a very long time? This is one of the central motivating questions behind the study of secure archival systems, and it is the focal point of this work.

1.1 The Problem of Secure Archival.

Within the context of computing, archival refers to the long-term preservation of digital data. In this work we take "long-term" to be on the order of a human lifetime or more. Because the primary purpose of archives is preservation, *reliability* is the most basic requirement for archival systems: user data should never be lost or corrupted due to system failure, bit rot, negligence, etc. Another basic concern for archival systems is *storage efficiency*. This is because archives are large—potentially on the order of several exabytes or more [Goo19]. Combine this with the fact that the cost of data storage is driven by quantity, duration, and access latency, and one quickly realizes that archives cannot afford to use storage inefficiently. Interestingly, archival systems are often willing to sacrifice latency: data is archived with the intention of accessing it eventually—exactly when is not so important.

A pressing third requirement for archival systems—and possibly the most important—is *security*, by which we mean the protection of data confidentiality (secrecy), integrity, and availability (i.e., the CIA triad [And03]) against an intelligent adversary. This is because archival data is often highly valuable and/or sensitive. Consider as examples public historical records, private medical data, trade secrets, or classified government secrets. These may all be targets for adversaries willing to spend vast resources to attack an archive over the course of many years.

Given what might be at stake, it is desirable to know how to make a long-term archive as secure as possible. When asking questions about the possibilities or impossibilities of data security, it makes sense to begin by inspecting the mathematical guarantees at the level of bits and encodings—the domain of cryptography and information theory. Secure storage systems typically use encryption for confidentiality, message authentication codes (or digital signatures) for integrity, and replication or erasure coding for reliability and availability. On the time scale of archival systems, however, we must contend with the possibility of cryptographic obsolescence. That is, as time progresses, it becomes more likely that cryptographic schemes in use today will be broken by future cryptanalytic advances. This danger is unavoidable unless—at the very least— we resolve long-standing open questions in complexity theory like $\mathbb{P}$ vs. $\mathbb{NP}$ (and the question must be resolved in a positive manner, i.e., $\mathbb{P} \neq \mathbb{NP}$, see Section 2).

A natural response to the threat of cryptographic obsolescence is to shrug and suggest re-encrypting (for confidentiality) or re-signing (for integrity) data. In the case of confidentiality, this is clearly insufficient when one is worried about so-called "Harvest Now, Decrypt Later" attacks [Noo23]. In such an attack, an adversary steals encrypted data from a system with the hopes of gleaning information about the stolen data possibly decades into the future with the help of improved cryptanalysis (or better yet, poor key management). Concern for Harvest Now, Decrypt Later is more than just gratuitous paranoia—the threat is being taken seriously by industry and government alike with the prospects of cryptographically viable

quantum computers on the horizon [Lag23, Noo23, Tow22].

1.2 An Information-Theoretic Solution

How, then, can we hope to attain security—especially secrecy—in the face of both cryptographic obsolescence and Harvest Now, Decrypt Later attacks? We propose the use of *secret sharing*, a method of encoding a secret into n secret "shares" with a threshold $t \leq n$, such that any subset of t shares suffices to recover the secret, but any subset of fewer than t shares reveals no information about the secret. Slightly more formally, we require that any subset of fewer than t shares is statistically independent of the secret. This requirement of statistical independence is an *information-theoretic* security guarantee, meaning it holds against computationally unbounded attackers, and is therefore immune to cryptographic obsolescence.

Secret sharing lends itself well to a decentralized secure storage system because a user can secret-share their data, then upload each share to a geographically distinct and administratively independent storage provider. With this decentralized approach, the user avoids a single point of trust or failure, and can additionally resist Harvest Now, Decrypt Later attacks. For example, consider a storage provider (e.g., AWS) who unfairly hikes their prices, loses a share (admittedly unlikely since storage providers place a premium on data reliability), or is suspected of stealing a share from—or colluding with another provider (e.g., Azure). Assuming at least t providers are still honest, and fewer than t providers have been corrupted, the user can retrieve shares from all honest providers, ask the honest providers to delete their shares, then reconstruct and reshare the secret among a new set of storage providers that the user believes is uncorrupted. This share "refresh" functionality is critical to long-term secrecy. Without it, given enough time an adversary would eventually be able to steal enough shares to reconstruct the secret—no cryptanalysis required.

This user-driven approach is reminiscent of the "multi-cloud" paradigm, wherein a user transparently leverages the services of multiple clouds through a single interface [BLM20]. We note that multi-cloud is really an umbrella term referring to any solution that leverages the services of multiple clouds [Wik24], but here (as we believe is common) we take it to mean the weakest form of multi-cloud where integration/interoperability between clouds is virtually nonexistent. The main drawback of the multi-cloud approach is the excessive burden it places on the user. They must monitor each provider for price hikes, data loss, security events, etc., and then carry out reconstruction and resharing. It is highly unrealistic to expect users to do this, especially when considering the long-lived nature of the system. Of course, these tasks could be outsourced to some third-party application provider like a cloud broker [LTM+11]. But again, this introduces a single point of trust, and now we must worry about the broker itself going offline, raising prices, going rogue, etc.

To improve upon the (weak) multi-cloud approach, we instead envision a fully decentralized, self-maintaining system that does not require user intervention to keep data secure. In particular, the system needs to be able to refresh its secret shares without user intervention. This can be accomplished through *Proactive Secret Sharing* (PSS): a special case of secure multi-party computation (MPC) that emulates the functionality described above where a user reconstructs their secret and redistributes fresh shares. When the set of parties can change between refreshes, the PSS scheme is called *dynamic*: no party learns any information about the secret

or anyone else's share, but everyone ends up with fresh shares of the same secret. Ideally, the system is configured to execute PSS periodically (e.g., once a year), or whenever some security event is triggered, or upon user prompting.

In summary, we propose the use of a decentralized, dynamic, proactive, secret-shared storage system as the core component behind a long-term secure archive. After exploring the nature of cryptographic obsolescence (Section 2) and introducing a system and threat model (Section 3), we describe the key technical ideas behind secret sharing and PSS (Sections 4 and 5). Finally, we cover related work (Section 6) and conclude with a discussion of future research directions that address the shortcomings of existing secret-shared archives. (Section 7).

2 Cryptographic Obsolescence

2.1 Cryptographic Hardness Assumptions

Cryptographic obsolescence refers to the expiration of a cryptographic algorithm's security due to new cryptanalytic attacks. This is quite a rare phenomenon compared to all the other disasters that could befall a secure system, but it happens enough that there are several examples to pull from. Recall the demise of the MD5 hash [SSA+09], or that of DES encryption [BS93]. Shor's algorithm [Sho99] has theoretically rendered discrete-log and factoring based cryptosystems insecure against quantum adversaries—a breakthrough that is still driving modern research efforts on plausibly post-quantum cryptography.

Granted, if we are worrying about secure storage, then we should focus on the state of the art and practice in symmetric key encryption schemes: the Rijndael algorithm, or Advanced Encryption Standard (AES) [DBN+01]. After more than twenty years of cryptanalysis, AES remains (to public knowledge) as a battle-tested and well-trusted encryption solution with no even remotely practical attacks on any full-round versions. In fact, a prevailing notion among the cryptology community is that the security of symmetric crypto (i.e., private-key encryption, hashes, etc.) is more or less a solved problem security-wise [Aum19].

Nevertheless, it is currently impossible to prove that *any* cryptographic primitive will not be broken in the future. This is because cryptography is basically a field built on unproven mathematical assumptions of computational hardness. For example, Diffie-Hellman key exchange assumes the hardness of computing discrete logarithms [DH76], RSA encryption assumes the hardness of factoring RSA moduli [RSA78], several post-quantum cryptographic schemes assume the hardness of the Module Learning with Errors (MLWE) problem [BDK+18], and—like we said above—breaking AES is assumed to be hard because of two decades of failed cryptanalysis. A natural question to ask is why all these assumptions remain unproven. In short, the answer is that all of cryptography currently implies a single, minimal assumption: the existence of one-way functions [Imp95]. Informally, a one-way function is a function that is easy to compute in the forward direction, but difficult to invert. It is a straightforward exercise to show that cryptographic primitives such as hash functions and encryption schemes imply one-way functions. The existence of one-way functions is a strictly stronger assumption than $\mathbb{P} \neq \mathbb{NP}$ [AB09], so unconditional proofs of security for any cryptographic algorithms depending on hardness assumptions depends on the resolution of one of the most intractable problems in

all of mathematics. In other words, the possibility of cryptographic obsolescence is unavoidable for the foreseeable future.

2.2 Information-Theoretic Security

As we alluded to in Section 1, there is, in fact, a way of unconditionally proving the security of some non-cryptographic schemes. Non-cryptographic schemes enjoy the property of *information-theoretic* security, i.e. provable security that makes no computational hardness assumptions. Contrasted with the *computational* security of cryptography, information-theoretic security is much stronger as it holds against computationally unbounded adversaries, but information-theoretic schemes tend to suffer from poor efficiency.

The historic canonical example of this is the One-Time Pad (OTP): an encryption scheme that XORs the plaintext (in bits) with a uniformly random key to produce the ciphertext. Even if an adversary in possession of the ciphertext were to brute force through every possible key, they would fail to learn any additional information about the plaintext beyond that which they already knew (i.e. the posterior equals the prior). The efficiency drawback is that the key must be the of the same length as the message. This makes the scheme impractical in many situations. For one, if Alice wishes to securely send Bob an n-bit message with the OTP, they must both first securely agree upon a uniformly random n-bit key, which is very similar to the problem they were trying to solve in the first place. From a storage perspective, if Alice uses the OTP to encrypt her secret files, she has now doubled her storage problems, as losing the key effectively destroys the information content of the message. These impracticalities motivated the search for secure encryption schemes with short key lengths. Computational hardness assumptions allow circumvention of unsavory lower bounds on key lengths for information-theoretic encryption, and thus the setting of computational security was introduced. Information-theoretic approaches took a backseat to the flourishing field of cryptography, as a surprising number of impractical or impossible tasks in the information-theoretic setting (far beyond short encryption keys) suddenly become possible with cryptographic assumptions.

Of course, as we have addressed above, one thing that does not become possible with cryptographic assumptions is long-term secrecy. In fact, this may be one of the only application scenarios in which information-theoretic security is preferred over the computational setting. As such, relatively few works on secure storage adopt information-theoretic approaches (see Section 6). When they do, they always turn to approaches based on secret sharing, which we cover in Section 4.

3 System and Threat Model

We consider a decentralized storage system comprised of geographically distributed and administratively independent storage providers. Ideally, each provider is itself a reputable geo-distributed storage system, like AWS. Beyond this, however, the internal architecture of each storage provider is arbitrary. This is convenient for a few reasons. For one, architectures may differ greatly between providers. On the archival time scale, architecture within even a single provider will likely change, and we should expect different providers to come and go (i.e. dynamic committees).

Leaving the design of each provider unspecified lets us discuss the system as a set of n abstract computing processes (nodes) communicating over a network of point-to-point links. We assume for now that the (logical, not necessarily physical) network topology is a clique. We also assume a secure public-key infrastructure (PKI) that ensures (computationally) secure private and authenticated communication channels between parties.

With such a system model in mind, we turn to threat modeling, taking inspiration from the mobile adversary model of Ostrovsky and Yung [OY91]. In order to violate data confidentiality, integrity, or availability, an adversary must clearly exert control over some nodes or their communication links. Because we abstracted so heavily over the internals of each node, we assume all-or-nothing Byzantine corruptions. That is, nodes are either honest or malicious (i.e. Byzantine). Honest nodes follow protocol and are completely free of any faults or adversarial influence. Malicious nodes are wholly corrupted by an adversary with access to all private keys, admin privileges, storage, memory, etc., and thus may deviate arbitrarily from protocol and learn any of the node's secret data. Adversaries are assumed to run in probabilistic polynomial time (PPT). Obviously we cannot allow an adversary to corrupt all n nodes, so we assume an adversary can control at most $f < n$ nodes at any given point in time. Typically, $f < \lfloor n/3 \rfloor$ or $f < \lfloor n/2 \rfloor$. This subset of f malicious nodes can be arbitrarily and adaptively changed by the adversary throughout the lifetime of the system. A threat model for a distributed system should also address network synchrony models, as the network can itself be viewed as an adversary that controls delivery of all messages. Note that control of message delivery also implies the ability to eavesdrop on—and tamper with—messages, but this is not an issue as long as (a) our assumption of a secure PKI holds, and (b) the adversary is a PPT machine. The synchrony model imposes certain restrictions on the network adversary. Broadly speaking, the network could be synchronous, asynchronous, or somewhere in between. In a synchronous network, each message is guaranteed to be delivered within some known delay upper bound Δ. As a consequence, we can assume that a system proceeds sequentially in epochs $e_0, e_1, e_2, ...$ (this is actually called lock-step synchrony, and it can be achieved with a clock synchronization protocol [ADD$^+$19]). This makes it easier to design and analyze synchronous protocols. In a fully asynchronous network, messages can be arbitrarily and indefinitely delayed. The asynchronous model is fraught with impossibility results, and protocols in this model are generally more difficult to design and analyze. Partial synchrony [DLS88] is a popular model that sits in between synchrony and asynchrony. In this model, the network can be fully asynchronous up until some unknown time GST, after which the network becomes fully synchronous. The rationale behind the partial synchrony assumption is that although the real world is technically asynchronous, in practice messages are not delayed arbitrarily since protocols like TCP provide reliable delivery. The synchrony model is quite relevant in the context of secure distributed protocols, such as proactive and verifiable secret sharing (Section 5). Choice of synchrony model affects protocol design, security, and efficiency. For the purposes of this work, however, the exact choice of synchrony model is not important as we do not commit ourselves to (nor do we design or analyze the security of) any distributed protocols.

4 Secret Sharing

Secret sharing is the fundamental cryptographic primitive underlying our secure archival system. In this section we will provide formal definitions for secret sharing, and constructions of both additive secret sharing and Shamir's secret sharing. These constructions assume familiarity with finite fields, see Appendix A for an overview. Proofs of security are deferred to Appendix B. It is worth noting we exclusively focus on information-theoretically secure secret sharing. Computational secret sharing schemes are their own field of study, and some of them have even been used in secure archival systems [RP11].

4.1 Definitions

All of the formal definitions in this section are adpated or pulled verbatim from an excellent survey on secret sharing schemes by Amos Beimel [Bei11]. We begin with definitions of *access structures* and *distribution schemes*, and then use both of these to define secret sharing.

Definition 4.1 (Access Structure [Bei11])**.** Identify a set of n parties by their indices $[n] \coloneqq \{1, ..., n\}$. A collection $A \subseteq 2^{[n]}$ is monotone if $B \in A$ and $B \subseteq C$ imply that $C \in A$. An access structure is a monotone collection $A \subseteq 2^{[n]}$ of non-empty subsets of $[n]$. Sets in A are called authorized, and sets not in A are called unauthorized.

The threshold access structure is perhaps the most popular example of an access structure. If $[n]$ is our set of parties, then a threshold access structure A_t is parameterized by some integer $t \leq n$, and consists of all subsets of $[n]$ of size t or greater. That is, $A_t = \{B \subseteq [n] : |B| \geq t\}$. A threshold access structure with n parties and threshold t is often written as a (t, n) access structure.

Definition 4.2 (Distribution Scheme [Bei11])**.** Let $[n]$ be a set of parties. A distribution scheme $\Sigma = (\Pi, \mu)$ with domain of secrets K is a pair, where μ is a probability distribution on some finite set R called the set of random strings, and Π is a mapping from $K \times R$ to a set of n-tuples $K_1 \times \cdots \times K_n$, where K_j is called the domain of shares of party j. A dealer distributes a secret $k \in K$ according to Σ by first sampling a random string $r \in R$ according to μ, computing a vector of shares $\Pi(k, r) = (s_1, ..., s_n)$, and privately communicating each share s_j to party j. For a set $I \subseteq [n]$, we denote $\Pi(k, r)_I$ as the restriction of $\Pi(k, r)$ to its I-entries.

The probability distribution μ is very often the uniform distribution, so we frequently leave it unspecified and assume μ is uniform by default. We now present two equivalent definitions of secret sharing. Proof of their equivalence can be found in Claim 1 of Beimel [Bei11].

Definition 4.3 (Secret Sharing [Bei11])**.** Let K be a finite set of secrets, where $|K| \geq 2$. A distribution scheme (Π, μ) with domain of secrets K is a secret-sharing scheme realizing an access structure A if the following two requirements hold:

1. **(Correctness)** The secret k can be reconstructed by any authorized set of parties. That is, for any set $B \in A$ where $B = \{i_1, ..., i_{|B|}\}$, there exists a reconstruction function $\textsc{Recon}_B \colon K_{i_1} \times \cdots \times K_{i_{|B|}} \to K$ such that for every $k \in K$,
$$\Pr[\textsc{Recon}_B(\Pi(k, r)_B) = k] = 1$$

2. **(Perfect Privacy)** Every unauthorized set cannot learn anything about the secret (in the information-theoretic sense) from their shares. Formally, for any set $T \notin A$, for every two secrets $a, b \in K$, and for every possible vector of shares $(s_j)_{j \in T}$:
$$\Pr[\Pi(a, r)_T = (s_j)_{j \in T}] = \Pr[\Pi(b, r)_T = (s_j)_{j \in T}]$$

The second definition of secret sharing makes use of the information-theoretic entropy function H. Let X be a random variable with support $\mathcal{X}$. The entropy function is defined as $H(X) \coloneqq \sum_{x \in \mathcal{X}} \Pr[X = x] \log(1/\Pr[X = x])$.

Definition 4.4 (Secret Sharing - Alternative Definition [Bei11])**.** A distribution scheme (Π, μ) is a secret sharing scheme realizing an access structure A with respect to a given probability distribution on the secrets, denoted by a random variable S, if the following conditions hold.

1. **(Correctness)** For every authorized set $B \in A$, $H(S|\Pi(S, R)_B) = 0$ (where we abuse notation and use R to denote the random variable according to μ over the finite set of random strings used by Π, also denoted R). In other words, $\Pi(S, R)_B$ implies the value of S.
2. **(Privacy)** For every unauthorized set $T \in A$, $H(S|\Pi(S, R)_T) = H(S)$. Equivalently, the random variables S and $\Pi(S, R)_T$ are independent.

Instead of using (Π, μ) to specify a secret sharing scheme, we will often find it useful (and more readable) to specify a secret sharing scheme as a pair of algorithms $(\textsc{Share}, \textsc{Recon})$, where $\textsc{Share}$ is just alternative notation for the distribution scheme Π, and $\textsc{Recon}$ is the corresponding reconstruction algorithm.

4.2 Additive Secret Sharing

We now present a construction of additive secret sharing. Let $\mathbb{F}_q$ be a finite field, and $[n]$ be a set of parties such that $n < q$. Let $K = K_1 = \cdots = K_n = \mathbb{F}_q$ be the set of secrets and share domains, let $R = \mathbb{F}_q^{n-1}$ be the set of random strings, and let μ be the uniform distribution on R. The distribution scheme Π is as follows. On input $k \in K, (r_1, ..., r_{n-1}) \in R$, compute $r_n \coloneqq k - r_1 - \cdots - r_{n-1}$, and return vector of shares $(r_1, ..., r_n)$. This scheme is perfectly secret and realizes an (n, n) access structure (i.e., every party is needed to reconstruct the secret). This statement is formalized and proved as Claim B.1.

As one may suspect, additive secret sharing is not so ideal for a secure storage system. Its major drawback is an inability to tolerate the loss of even a single share, for this would information-theoretically destroy the original secret. More desirable would be a threshold secret sharing scheme for which the threshold was configurable and not fixed to n. This is accomplished by the well-known threshold secret sharing scheme of Adi Shamir from 1979 [Sha79], which we present next.

4.3 Shamir's Secret Sharing

As before, let $\mathbb{F}_q$ be a finite field, and $[n]$ a set of parties such that $q < n$. Let $K = K_1 = \cdots = K_n = F_q$ be the set of secrets and share domains. Let $2 \leq t \leq n$ be the threshold, $R = \mathbb{F}_q^{t-1}$ the set of random strings, and μ the uniform distribution on R. On input $k \in K, (r_1, ..., r_{t-1}) \in R$, Π constructs the degree $t - 1$ polynomial

$p(x) = k + r_1 x + \cdots + r_{t-1} x^{t-1}$, chooses any n distinct but non-zero evaluation points $x_1, ..., x_n \in F_q - \{0\}$, and returns vector of shares $((x_1, p(x_1)), ..., (x_n, p(x_n)))$ consisting of all the points on the polynomial associated with the n evaluation points. Correctness intuitively follows because any t shares represent t points on the polynomial, which suffice to uniquely interpolate the degree $t-1$ polynomial p and recover the constant term k (see Theorem B.1). Privacy intuitively follows because if only a set of $t-1$ points on the polynomial (i.e., shares) are known, then considering any secret as an additional point (specifically the point at $x = 0$) of the polynomial results in a unique polynomial, and the probability that this polynomial is the correct polynomial is only as likely as the a priori distribution on the secret. See Claim B.2 for the full proof.

Any secret sharing scheme—including Shamir's—enjoying perfect secrecy must pay the price in storage inefficiency. Notice in Shamir's we started with a single field element as the message, and we produced n field elements as output. In coding-theoretic terms, we can view the n shares as a single codeword resulting from applying the scheme to our message. This means the *rate* of the code (ratio of message size to codeword size) is $O(1/n)$. Contrast this with AES, which asymptotically has optimal rate $O(1)$. Informally, this is because the quality of information-theoretic security relies on how much entropy (i.e. randomness) is used in the scheme. Formal proofs of lower bounds on share size for perfectly secret secret sharing schemes can be found in Section 5 of Beimel [Bei11].

5 Proactive Secret Sharing

Recall in Section 1.2 we sketched the naïve approach to a secret-shared storage system wherein a user secret shares their data and uploads each share to a "committee" of geographically distinct and administratively independent storage providers. To ensure the security of their data against Harvest Now, Decrypt Later attacks, the user should periodically retrieve shares from the storage providers (and request their deletion), reconstruct the secret, generate fresh secret shares of the secret, and re-upload them (possibly to a different set of storage providers). The drawback of this approach is the burden of requiring the user to facilitate the refresh. This can be avoided with proactive secret sharing (PSS): a secure distributed protocol that emulates the refresh process in a decentralized and (optionally) Byzantine fault-tolerant manner.

Many PSS protocols have been designed over the past few decades. These protocols may vary by threat model, desired features, and level of sophistication. Full descriptions and analyses of even the simplest PSS protocols can be a bit involved.

As a result, in this section we only aim to build an intuition for PSS protocols and identify key techniques common across several PSS protocols. We do this by sketching the seminal PSS protocol of Herzberg et al. [HJKY95] (which we refer to as "Herzberg's PSS" for convenience), first in the semi-honest case and then in the malicious case. In doing so, we will cover the fundamental zero-sharing trick, along with verifiable secret sharing and its subcomponents. Following Herzberg's PSS, we briefly cover the (equally fundamental) resharing trick of Desmedt and Jajodia [DJ97] that initiated the study of dynamic PSS (DPSS) protocols which allow redistribution of fresh shares to a new access structure. The section concludes with a high-level overview of the state-of-the-art in DPSS.

5.1 Herzberg's Semi-Honest PSS

To the best of our knowledge, the work of Herzberg et al. [HJKY95] was the first to coin the term proactive secret sharing, and the first to provide a full construction. Herzberg's PSS assumes a (lock-step) synchronous system of n nodes, private (see Sections 6 and 7 for more on this assumption) authenticated point-to-point channels between all pairs of nodes provided by a secure PKI, and the existence of a "secure" broadcast channel. Note that in practice, this broadcast channel would have to be implemented over the point-to-point channels. This is trivial in the semi-honest case we consider here where every node seeks to learn private information but does not deviate from protocol: the broadcaster simply sends the same message to every other party individually using the point-to-point channels. In the malicious (i.e. Byzantine) case however, implementation of the broadcast channel becomes quite nontrivial. Discussion of secure broadcast implementation is a rabbit hole that drifts out of scope for this work, but we will say a few more words about this later in Section 5.2.

Before describing the semi-honest protocol, we introduce the key technique enabling the generation of fresh shares used in Herzberg's and other PSS protocols. It is sometimes called "zero-sharing" [MZW+19], or "commit to zero" [NN04], as it involves parties sending each other fresh secret shares of the zero field element, and adding received shares of zero to their existing local share. Combined with the fact that any unauthorized subset of Shamir's secret shares is distributed uniformly and independently at random (see Claim 1 in Chandran et al. [CKOS21]), a homomorphic property of Shamir's secret sharing ensures that every party ends up with fresh shares that are statistically independent of the old shares, and yet the underlying secret remains unchanged. The homomorphic property of Shamir's secret sharing is really a homomorphic property of polynomial interpolation, which we state precisely and prove in Claim B.3. Put plainly, the homomorphic property implies that the "sum of shares equals shares of the sum". We assume Shamir's secret sharing for the rest of the section unless specified otherwise. The upshot is that if $\textsc{Share}(m, r)$ is a secret sharing of some message m with randomness r, then we can generate secret shares $\textsc{Share}(0, r')$ of the zero field element, and their sum $\textsc{Share}(m, r + r')$ remains a secret sharing of m. Further, this new secret sharing (intuitively) "looks" uniformly random and independent from $\Pi(m, r)$ since we essentially masked each share of $\textsc{Share}(m, r)$ with an independent and uniformly random value (i.e., a share of $\textsc{Share}(0, r')$).

We now describe Herzberg's PSS in the semi-honest case where corrupted parties collect and share information with the aim of learning the secret, but do not deviate from protocol in order to e.g., tamper with the integrity of the secret. Suppose a secret s has been $(k + 1, n)$ secret shared so that party $1 \leq i \leq n$ holds share s_i. To refresh the shares, each party i generates a random degree k polynomial δ_i such that $\delta_i(0) = 0$, and then sends $\delta_i(j)$ to party j. Upon receiving $\{\delta_j(i)\}_{j \neq i}$, party i sets $s'_i := s_i + \sum_{j \neq i} \delta_j(i)$, and deletes their old share. Notice each $\delta_j(i)$ is an i-th Shamir secret share of 0, so the resulting share s'_i is independent of s_i (as it has been masked several times), but still corresponds to the original secret s.

While this simple semi-honest example due to Herzberg et al. represents the core technique behind many subsequent PSS works, there are two major issues to address before such a scheme could hope to be responsibly deployed in practice. The first is

that we have only considered semi-honest adversaries. A single malicious adversary can destroy the secret by sending garbage (for example) instead of the appropriate $\delta_j(i)$. This issue is addressed with a *verifiable secret sharing* (VSS) subprotocol. The second issue is that the set of parties is static, so shares of zero are being sent right back to the malicious parties. The threat model of Herzberg et al. justifies this by assuming adversarial corruption is removed from each corrupted party before the refresh via some magical "reboot". This is unrealistic. A storage provider may not know how to properly remove an adversary, or even know an adversarial corruption has occurred in the first place. Further, the storage provider itself may be under total adversarial control (e.g., they went rogue). What we desire instead is a PSS protocol supporting *dynamic* access structures, wherein the newly minted shares can be given to a new set of parties, possibly of different size and with different threshold. We address the problem of malicious parties next.

5.2 Herzberg's Malicious PSS

The key idea enabling security against malicious parties is the incorporation of a verifiable secret sharing (VSS) subprotocol. Like PSS, (VSS) is a secure distributed protocol that builds upon an existing secret sharing scheme (we again assume Shamir's secret sharing unless specified otherwise). Informally, the goal of a VSS protocol is to protect shareholders against a malicious dealer who tries to deal inconsistent shares to the shareholders (e.g., a dealer who sends each party uniformly random garbage). At first glance, this may seem like a strange security goal because, in the context of our discussion on archival, we might associate the dealer with a user, and a user has no incentive to deal inconsistent shares. This is indeed a reasonable intuition, but it arises from the assumption that the dealer is always the user. Recalling Herzberg's semi-honest PSS from the prior section, each party is itself a dealer when it generates shares of zero to send to every other party. Should a party not honestly generate shares of zero, the secret could be destroyed or altered. Hence, the use of VSS makes sense within the context of a PSS protocol to protect against shareholders who act as dealers during share renewal. For completeness, we provide a formal definition of VSS adapted from Das et al. [DXT+23].

Definition 5.1 (Verifiable Secret Sharing [DXT+23])**.** A verifiable secret sharing (VSS) protocol consists of two phases: sharing and reconstruction. During the sharing phase, a dealer D shares a secret $s \in \mathbb{F}$ (for $\mathbb{F}$ a finite field). During the reconstruction phase, nodes interact to recover the secret. We say a VSS protocol is k-resilient if the following properties hold with probability $1 - \text{negl}(\lambda)$ (where λ is the security parameter) against any PPT adversary $\mathcal{A}$ that corrupts up to k nodes:

1. **Correctness.** If D is honest and has a secret s, then the sharing phase will result in all honest nodes eventually outputting a share of s. Once the sharing phase finishes, if all honest nodes start the reconstruction phase, they will output s.

2. **Completeness.** If any honest node outputs in the sharing phase, then there exists a secret $\tilde{s} \in \mathbb{F}$ such that all honest nodes eventually output a share of $\tilde{s}$. Also, $\tilde{s}$ is guaranteed to be reconstructed during the reconstruction phase.

3. **Secrecy.** If D is honest, there exists a PPT simulator $\mathcal{S}$ which interacts with

an ideal functionality $\mathcal{F}_{VSS}$ and outputs a view of $\mathcal{A}$, such that $\mathcal{A}$'s view in the real-world protocol and the simulated protocol are indistinguishable.

4. **Termination.** All honest nodes will eventually terminate the sharing phase.

We make a few comments on this definition before sketching a VSS construction. First, the termination property, as stated, only makes sense in synchronous systems. For asynchronous systems, the authors replace the termination with an asynchronous termination property stating *if* any honest node outputs in the sharing phase, then all honest nodes will eventually terminate. Second, the language in the secrecy property belongs to the simulatability paradigm for security proofs, and very loosely says that an adversary cannot do better in the real world than they can against some ideal execution of the protocol that leverages a trusted third party (i.e., $\mathcal{F}_{VSS}$). Third, if the secrecy property holds with probability 1, (i.e. $\text{negl}(\lambda) = 0$), then we have perfect secrecy.

The specific VSS construction used in Herzberg's PSS protocol is due to Pedersen [Ped91]. Two salient features of Pedersen's VSS are that it (a) unconditionally (i.e., information-theoretically) hides the secret, and (b) is non-interactive (i.e., requires only one round of communication between dealer and parties). The first feature is desirable for long-term security, while the second is desirable because it saves on communication complexity. The key idea of the protocol is for the dealer to *broadcast* a Pedersen commitment to the random polynomial used for Shamir's secret sharing, then send to each party their share and some additional opening information that lets each party verify that their share was generated according to the dealer's random polynomial. Information-theoretic secrecy is preserved by the unconditional hiding property of the Pedersen commitment. A homomorphic property of the Pedersen commitment allows parties to non-interactively verify their shares. See Appendix C for the relevant details of commitment schemes that make this possible.

One caveat is that the binding property of the Pedersen commitment depends on hardness of the discrete log assumption, so a dealer that efficiently computes discrete logarithms can succeed in dealing inconsistent shares. Fortunately, this is not much of an issue, since data *integrity* does not suffer from the same Harvest Now, Decrypt Later threat as data secrecy. Integrity remains uncompromised as long as the dealer cannot efficiently compute discrete logarithms at the time of protocol execution. Of course, as mentioned in Section (2), in theory, discrete logarithms can already be computed in polynomial time with Shor's algorithm on a quantum computer. If the threat of cryptographically viable quantum computers seems imminent, we could swap out the Pedersen commitment with another suitable unconditionally hiding commitment whose binding depends on a secure post-quantum hardness assumption, such as the one in Cabarcas et. al [CDG+15].

It is worth noting that a commitment scheme cannot be both unconditionally hiding and binding. Intuitively, this is because unconditional binding implies that the commitment can only correspond to one element (or some small subset) of the message space, while unconditional hiding implies that the commitment could correspond to any element (or some large subset) of the message space. These implications directly contradict each other. Thus, when selecting commitment schemes for a VSS protocol, we must choose between unconditional hiding / computational binding, computational hiding / unconditional binding, or computational hiding and

binding. Because our focus in this work is long-term security against Harvest Now, Decrypt Later attacks, we opt for unconditional hiding. Computational binding is acceptable for use in PSS since we need only worry about integrity violations during protocol execution, so we can simply use whatever computational assumption is currently believed to be secure.

A second caveat is—as mentioned in the section preamble—the assumption of a secure broadcast channel. The naïve approach to broadcast—send the same message individually to each party—is trivially insecure against a malicious broadcaster who sends different messages to each party. A secure broadcast channel can be realized with Byzantine broadcast (BB) in synchronous systems, and Byzantine reliable broadcast (RB) in asynchronous systems. The difference between the two is identical to the difference in Definition 5.1: BB requires that all parties terminate, while RB requires that if one honest party terminates, then all parties terminate [LRU22]. We do not discuss any protocols for realizing either of these or their costs, but will point out that their communication complexity is necessarily higher than that of the semi-honest case, as parties must communicate with each other to weed out misinformation.

Armed with a high-level understanding of Pedersen's VSS, we provide a sketch of Herzberg's full protocol with malicious security. Assume a user has already honestly dealt Shamir shares of a secret to n parties with threshold $k+1$, such that party i holds share s_i. Herzberg's protocol assumes a synchronous system with private authenticated channels, so the lifetime of the system can be divided into epochs, and each party can be sure that messages are reliably delivered, untampered, from the party that claims to have sent the message, in a timely manner. At the end of each epoch t, parties begin a share renewal phase. Each honest party i generates n shares of zero using a random polynomial δ_i as described above in Section 5.1, sends each share to its corresponding party, and broadcasts a Pedersen commitment to δ_i. When party j receives all of its shares of zero, it verifies the i-th share against the public commitment to δ_i. If all verifications pass, then party j broadcasts an acceptance message to all other parties. If the verification for the $\tilde{i}$-th share fails, then party j broadcasts an accusation against party $\tilde{i}$. Party $\tilde{i}$ must then broadcast the share it sent to party j, at which point every other party can check whether this share indeed failed verification. Parties then agree on a set of "bad" parties, and add to their local share those shares of zero belonging to "good" parties. This completes the share renewal phase, and the epoch is set to $t+1$.

5.3 The Resharing Trick for Dynamic-Committee PSS (DPSS)

We now address the second issue with Herzberg's PSS (Section 5.1) of static parties. The study of PSS with dynamic access structures (DPSS) was initiated by the work of Desmedt and Jajodia in 1997 [DJ97]. This work showed that linear secret sharing schemes satisfying certain properties (Shamir's among them) admit a "resharing" trick, whereby an authorized subset of parties in the original committee generates "subshares" of their local shares using an appropriate secret sharing scheme. These subshares are then sent to corresponding parties in the new committee. Each new party uses its received subshares to compute a new local share. This new share belongs to a sharing of the original secret under a new access structure (e.g., possibly different threshold). Desmedt and Jajodia only considered the semi-honest case

(passing mention of a robust protocol variant is made in Section 6 [DJ97]). Wong, et al., [WWW02] use a straightforward application of VSS techniques to support malicious parties, but incur exponential communication costs in the worst case, and assume all parties in the new committee are honest.

To provide slightly more detail on the resharing technique, suppose a secret $s \in \mathbb{F}$ was $(k+1, n)$ secret shared as $(s_1, ..., s_n)$ such that, for any authorized set B, there exist weights $(b_i)_{i \in B}$ such that $s = \sum_{i \in B} b_i s_i$ (in the case of Shamir's secret sharing, $b_i = \prod_{j \in B, j \neq i} x_j/(x_j - x_i)$), where $x_1, ..., x_n \in \mathbb{F}$ are the evaluation points). B wishes to produce fresh shares for a new set of parties $[n']$ with a new threshold $k'+1$. To do this, each party $i \in B$ generates a new degree k' random polynomial $p'_i(x) = s_i + a_{i,1}x + \cdots + a_{i,k'}x^{k'}$, and computes *subshares* $\hat{s}_{i,j} = p'_i(x_j)$ for all $j \in [n']$. Each party $i \in B$ then sends $\hat{s}_{i,j}$ to party $j \in [n']$. Now each new party $j \in [n']$ upon receiving its collection of subshares $\{\hat{s}_{i,j}\}$ from all $i \in B$, computes its final new local share $s'_j = \sum_{i \in B} b_i \hat{s}_{i,j}$. The fact that the collection of new shares $\{s'_j\}_{j \in [n']}$ are secret shares of s under the new $(k'+1, n')$ access structure follows from the main result of Desmedt and Jajodia [DJ97].

5.4 Overview of State-of-the-art in DPSS

A string of subsequent works [SLL10, MZW+19, VAFB22, GKM+22, YXD22, YXXM23] on dynamic-committee proactive secret sharing (DPSS) incrementally achieve improved security and practical efficiency through the use of a variety of additional techniques, including (but not limited to): distributed ledgers, bivariate polynomials, asynchronous networks, hyperinvertible matrices, distributed polynomial generation, non-interactive zero-knowledge proofs, KZG polynomial commitments, and multi-valued validated Byzantine agreement. As a result, these DPSS systems have become significantly more complex than the historical PSS protocols described above. As such, we opt to briefly provide an overview of the "HoneyBadger" DPSS protocol of Yurek et al. [YXXM23], and the COBRA DPSS protocol of [VAFB22], which we believe to be the current state-of-the-art in DPSS. Interestingly, none of these works are concerned specifically with long-term archival, instead citing blockchain issues like MPC in the face of node churn, or confidential Byzantine fault-tolerant state machine replication, as their target applications.

Honeybadger DPSS assumes an asynchronous network with authenticated private channels between each pair of nodes, and considers the problem of resharing a secret s from a committee C to a new committee C'. At most t nodes are corrupted in C, and at most t' nodes are corrupted in C', subject to the constraints $t < |C|/3$ and $t' < |C'|/3$. It is possible that nodes in C overlap with those in C'. The main tools used in the protocol are an asynchronous complete secret sharing scheme (ACSS), Byzantine reliable broadcast (RB), and multi-valued Byzantine agreement (MVBA). ACSS is similar to VSS, in that parties can verify that everyone received a share consistent with a broadcasted commitment. Their specific ACSS construction relies on RB, Pedersen commitments, Paillier encryption, and zero-knowledge proofs of knowledge (ZKPoK). In short, for each secret share s_i, the dealer broadcasts a Pedersen commitment to s_i, a Paillier encryption of s_i under the i-th party's public key, and a ZKPoK proving the dealer knows s_i given the commitment and the encryption. It is worth noting their ACSS supports a distinction between a correctness threshold t representing the maximum number of Byzantine parties, and

a privacy threshold $d+1$ representing the number of shares needed to reconstruct the secret. Additionally, we observe that this ACSS construction is not information-theoretically hiding, as every party must know the Paillier encryption of every other share. As such, this protocol would not be suitable for our purposes, but we describe it anyway as an example of the current state-of-the-art in DPSS. In MVBA, every party has an input value v, and the goal is to agree upon a single value satisfying a public predicate $f : \{0,1\}^{|v|} \to \{0,1\}$. At a high level, the protocol strategy is as follows. C uses ACSS to reshare (in the sense of Desmedt and Jajodia [DJ97]) shares to C'. Because of asynchrony, C' needs to agree on a set of $d+1$ parties from C that successfully completed the ACSS resharing; this is achieved via MVBA.

COBRA DPSS considers the same problem setting as HoneyBadger DPSS, except in a partially synchronous network. The main tool used in COBRA is distributed polynomial generation, which is built over verifiable secret sharing and Byzantine consensus. Distributed polynomial generation allows a group of parties to create a random degree t polynomial Q encoding a specific point (x, y). For the purposes of dynamic resharing, COBRA designs a modified distributed polynomial generation algorithm allowing the old committee C to generate two random polynomials Q, Q' subject to the constraint that $Q(0) = Q'(0) = q$, where q is a random value. Each party $i \in C$ holds a share $Q(i)$ (in addition to their original local share s_i), and each party $j \in C'$ holds a share $Q'(j)$. The key idea behind the resharing is as follows. Each party $i \in C$ will add the point $Q(i)$ to their original local share s_i, and sends the new *blinded* share $s_i + Q(i)$ to each $j \in C'$. Upon receiving at least $t+1$ correct blinded shares from C, each $j \in C'$ reconstructs the blinded secret $z := (s + Q(0))$. They then compute $s'_j := z - Q'(j)$ as their new share. Observe that the $\{s'_j\}$ are points on the polynomial $z - Q'$, and the free term of this polynomial is $z - Q'(0) = (s + Q(0)) - Q'(0) = s$, so the secret is preserved under the new refreshed shares. This describes the semi-honest case; VSS and Byzantine consensus are used in straightforward ways to lift this to the malicious case. Unlike HoneyBadger DPSS, this protocol is information-theoretically hiding, and therefore more suitable for long-term secure archival.

6 Related Work

Gridsharing [SB05], POTSHARDS [SGMV09], and PASIS [GKP05] (which subsumes the work of Wong et al. [WWW02]) were among the first works to propose a secret-shared storage system. That is, storage systems where the data is directly protected with information-theoretic secret sharing. These systems stand in contrast with others that use traditional encryption methods for data, and secret sharing for other purposes, like protecting server private keys [MS04]. POTSHARDS uses Shamir's secret sharing along with some disaster recovery techniques, but does not support any proactive share renewal capabilities. Gridsharing attempts to reduce computational overheads associated with Shamir's secret sharing and VSS by using XOR (i.e., additive) secret sharing and quorum techniques, but incurs higher storage overhead as a result. Further, while they mention the ability to perform proactive share renewal, they never actually incorporate it into their system. PASIS provides a general framework for survivable storage systems with several options for data encoding—one of them being the DPSS scheme of Wong et al. As discussed previously in Section 5, this early DPSS scheme suffers from exponential communication

overhead in the worst case, and assumes all parties in the new committee are honest during the share renewal phase.

Whereas these early works provide long-term confidentiality of data at rest via information-theoretic secret sharing, the 2016 work of LINCOS [BBD+17] and its follow-ups [BDF+20, MGA+20, GKKB18] belong to a more modern thread of secure archival that take a more hardline stance towards the issue of cryptographic obsolescence by additionally considering long-term confidentiality of data in transit, as well as long-term integrity. These works establish information-theoretically private communication channels by having parties agree on a One-Time Pad key with quantum key distribution (QKD). The main drawback is that QKD is a fledgling technology with high infrastructure costs. In particular, LINCOS throughput is severely limited by the 40KB/s rate at which they could generate fresh key material from the Tokyo QKD system. Long-term integrity is achieved with a chain of digitally signed timestamps. Intuitively, integrity of a digital signature can be prolonged indefinitely by signing an old timestamped signature with a new timestamped signature. As long as the new (presumably secure) signature is added before the old one is broken, the security of the entire chain is effectively renewed. Curiously, while LINCOS (and its aforementioned follow-ups) suggest the use of proactive secret sharing for long-term confidentiality, they forego its implementation and instead have the user manually reshare the secret periodically. To the best of our knowledge, no secure archival systems to date have actually implemented PSS/DPSS. The systems that do rigorously implement and evaluate PSS/DPSS focus on more general blockchain applications. We also note that *none* of the DPSS works from Section 5 consider information-theoretic channels. While these systems can be considered related work, we have already mentioned them in Section 5, and focus here on the specific problem of secure archival, especially in the face of cryptographic obsolescence and Harvest Now, Decrypt Later attacks.

That said, we briefly mention two approaches to secure archival in the computationally secure setting. The first is AONT-RS [RP11], a secure storage system designed around a computationally secure secret sharing scheme (constructed using encryption schemes, hash functions, and Reed-Solomon erasure codes). The approach of AONT-RS found its way into commercial use in the Cleversafe storage system, which was eventually acquired by IBM. The second approach is that of *cascade ciphers*: wrapping data in multiple layers of different encryption schemes to hedge against the cryptanalysis of any particular scheme. ArchiveSafeLT [SS22] is the only archival system we are aware of that uses this approach. Of course, since both of these approaches are only computationally secure, they are both susceptible to Harvest Now, Decrypt Later attacks.

We conclude the section by pointing out some pertinent surveys and position papers. In 2006, a position paper by Storer et al. [SGM06] holistically reasons about threats to long-term secure archives, but says nothing about information-theoretic security or secret-shared archives. A 2014 survey by Braun et al. [BBMW14] explores information-theoretic techniques for achieving long-term confidentiality of data at rest and in transit. A 2015 survey by Vigit et al. [VBC+15] complements that of Braun et al. by focusing on long-term integrity (and related) concerns. Finally, a recent position paper by Smith et al. [STD+24] focuses on challenges and approaches to building long-term secure archives in the face of cryptographic obsolescence and Harvest Now, Decrypt Later attacks This work can be considered an extension of

that work with a more pointed focus on using DPSS systems to achieve the same goal.

7 Discussion

Recall the main requirements we postulated in Section 1 for archival systems: security, reliability, and storage efficiency. Our proposed solution is a decentralized, geo-distributed, dynamic proactively secret shared storage system. While we believe this is the right approach for a long-term archive that must contend with the combined threats of cryptographic obsolescence and Harvest Now, Decrypt Later, there remain several areas for improvement.

A glaring issue is the poor storage efficiency incurred by information-theoretic secret sharing due to hard lower bounds on share size (see Section 4.3). Instead of trying to remedy this by using a different encoding—such as packed secret sharing [FY92], or entropically secure encryption [DS05]—we could directly try to make storage cheaper with a new generation of archival storage media. Two promising candidates are DNA and glass. DNA storage boasts a theoretical density of 1EB per cubic millimeter (8 orders of magnitude greater than tape), and centuries of durability. While not quite as dense, at 429TB per cubic inch, glass requires very little maintenance, and can survive for millenia [ZČD+16]. Of the two, glass appears to be much closer to widespread adoption: Microsoft's Project Silica [pro24] is a prototype archive based on glass. They choose glass over DNA due to the high costs and low throughputs of DNA synthesis and sequencing. One could still argue that regardless of storage media, asymptotically the storage cost of secret-sharing will not scale for massive archives. We do not refute this argument, but instead point out that our target workload for secret-shared archives is highly sensitive long-lived data for which the user does not even wish to risk the possibility of a Harvest Now, Decrypt Later attack years down the line. In these cases, the value of the data likely outweighs the storage cost. As one example, suppose a government organization stores petabytes of classified information in an archive. What should they do with the encryption keys, which are now as valuable as the cumulative value of every piece of data that was encrypted under said keys? Such a secret-shared archive is a strong contender for an extremely secure key management system. Further, encryption keys are short, so the storage cost in this case should be negligible compared to the size of the encrypted archive.

A security concern that has thusfar gone completely unaddressed by secret-shared systems is the relatively recently discovered vulnerability of many secret sharing schemes to leakage attacks. In a leakage attack, an adversary might leak only a few bits of information about a share via some hidden side-channel. Notice that the traditional threat model for secret sharing only considers adversaries that steal *whole* shares. A 2017 work on the exact repair problem for Reed-Solomon codes [GW17] was shown by Benhamouda et al. [BDIR21] to translate into a leakage attack on Shamir's secret sharing over characteristic 2 fields. In the worst case, leaking only one bit of information from each share (no need to steal any share in its entirety) can lead to complete reconstruction of the secret. In response to this vulnerability, several recent works [BDIR21, CKOS22, KK23, MNPC+22, MPCSW21] have proposed new leakage-resilient secret sharing (LRSS) schemes. These schemes can be broadly classified according to two axes: leakage model and linearity. The leakage model

determines the class of side-channel attacks that a scheme can resist. Strong leakage models protect against larger classes of attacks, but tend towards greater share sizes. Weaker leakage models have smaller share sizes, but protect against fewer attacks. Linear LRSS schemes are more compatible with existing secret-sharing based protocols (like PSS), but require restrictive parameter choices (e.g., field size, threshold) in order to maintain security. Nonlinear schemes do not have these restrictions, but the constructions are more complex, less efficient, and are not directly compatible with existing protocols. Evaluating the LRSS landscape in the context of archival systems is an open problem.

As mentioned in Section 6, LINCOS uses QKD for information-theoretic private channels. An alternative is the use of schemes in the Bounded Storage Model (BSM) [Mau92]. Cryptography in the BSM relies on the assumption that an adversary's storage capacity is limited. While unorthodox, this assumption is not of a computational nature, so schemes in the BSM still enjoy information-theoretic security. As an example, in the BSM honest parties can agree on a One-Time Pad key by leveraging a large source of public randomness that an adversary (with a much larger storage capacity than the honest parties) cannot fully capture and store. The only systems-level (i.e. non-theoretical) evaluation of the BSM we are aware of is from 2005 [DNLA05]. Since then, new theoretical results have expanded the possibilities of the BSM. Namely, the necessary gap between honest and adversarial storage has been improved from quadratic to exponential for important cryptographic primitives like key agreement, oblivious transfer, and general multi-party computation (at the cost of increased round and communication complexity) [Raz18, GZ19, DQW23].

In summary, dynamic proactively secret-shared datastores are a leading candidate for the core architecture of long-term secure archival systems that must contend with cryptographic obsolescence and Harvest Now, Decrypt Later attacks. DPSS has been studied for many years, and the latest systems have become quite sophisticated and increasingly practical due to interest in enabling confidential data storage and MPC applications in the blockchain space. By drawing on the state-of-the-art in DPSS and pursuing the open directions outlined above, we aim to develop highly secure, reliable, and efficient long-term archival systems that can safeguard humanity's most valuable data far into the future.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009.

[ADD+19] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. Synchronous byzantine agreement with expected o(1) rounds, expected communication, and optimal resilience. In *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers*, page 320–334, Berlin, Heidelberg, 2019. Springer-Verlag.

[And03] James M. Anderson. Why we need a new definition of information security. *Comput. Secur.*, 22(4):308–313, 2003.

[Aum19] Jean-Philippe Aumasson. Too much crypto. Cryptology ePrint Archive, Paper 2019/1492, 2019. *https://eprint.iacr.org/2019/1492*.

[BBD+17] Johannes Braun, Johannes Buchmann, Denise Demirel, Matthias Geihs, Mikio Fujiwara, Shiho Moriai, Masahide Sasaki, and Atsushi Waseda. Lincos: A storage system providing long-term integrity, authenticity, and confidentiality. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pages 461–468, New York, NY, USA, 2017. Association for Computing Machinery. *https://dl.acm.org/doi/10.1145/3052973.3053043*.

[BBMW14] Johannes Braun, Johannes Buchmann, Ciaran Mullan, and Alex Wiesmaier. Long term confidentiality: a survey. *Designs, Codes and Cryptography*, 71:459–478, 2014. *https://eprint.iacr.org/2012/449.pdf*.

[BDF+20] Johannes Buchmann, Ghada Dessouky, Tommaso Frassetto, Ágnes Kiss, Ahmad-Reza Sadeghi, Thomas Schneider, Giulia Traverso, and Shaza Zeitouni. Safe: A secure and efficient long-term distributed storage system. In *Proceedings of the 8th International Workshop on Security in Blockchain and Cloud Computing*, SBC '20, pages 8–13, New York, NY, USA, 2020. Association for Computing Machinery.

[BDIR21] Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. *Journal of Cryptology*, 34:1–65, 2021. *https://link.springer.com/article/10.1007/s00145-021-09375-2*.

[BDK+18] Joppe Bos, Leo Ducas, Eike Kiltz, T Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehle. Crystals - kyber: A cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367, 2018.

[Bei11] Amos Beimel. Secret-sharing schemes: A survey. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology*, pages 11–46, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[BLM20] Robert Bohn, Craig Lee, and Martial Michel. The nist cloud federation reference architecture, 2020. *https://doi.org/10.6028/NIST.SP.500-332*.

[BS93] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993.

[CDG+15] Daniel Cabarcas, Denise Demirel, Florian Göpfert, Jean Lancrenon, and Thomas Wunderer. An unconditionally hiding and long-term binding post-quantum commitment scheme. Cryptology ePrint Archive, Paper 2015/628, 2015. *https://eprint.iacr.org/2015/628*.

[CKOS21] Nishanth Chandran, Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Adaptive extractors and their application to leakage resilient secret sharing. In *Annual International Cryptology Conference*, pages 595–624. Springer, 2021.

[CKOS22] Nishanth Chandran, Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Short leakage resilient and non-malleable secret sharing schemes. In *Annual International Cryptology Conference*, pages 178–207. Springer, 2022. *https://eprint.iacr.org/2022/216.pdf*.

[DBN+01] Morris Dworkin, Elaine Barker, James Nechvatal, James Foti, Lawrence Bassham, E. Roback, and James Dray. Advanced encryption standard (aes), 2001-11-26 2001.

[DF03] D.S. Dummit and R.M. Foote. *Abstract Algebra*. Wiley, 2003.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DJ97] Yvo Desmedt and Sushil Jajodia. Redistributing secret shares to new access structures and its applications. Technical report, George Mason University, 1997. Technical Report ISSE TR-97-01, Vol. 148.

[DLS88] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, apr 1988.

[DNLA05] Timothy John Draelos, William Douglas Neumann, Andrew J Lanzone, and William Erik Anderson. Key management and encryption under the bounded storage model. Technical report, Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA ..., 2005.

[DQW23] Yevgeniy Dodis, Willy Quach, and Daniel Wichs. Speak much, remember little: Cryptography in the bounded storage model, revisited. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 86–116. Springer, 2023.

[DS05] Yevgeniy Dodis and Adam Smith. Entropic security and the encryption of high entropy messages. In *Theory of Cryptography Conference*, pages 556–577. Springer, 2005.

[DXT+23] Sourav Das, Zhuolun Xiang, Alin Tomescu, Alexander Spiegelman, Benny Pinkas, and Ling Ren. Verifiable secret sharing simplified. Cryptology ePrint Archive, Paper 2023/1196, 2023.

[FY92] Matthew Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '92, page 699–710, New York, NY, USA, 1992. Association for Computing Machinery.

[GKKB18] Matthias Geihs, Nikolaos Karvelas, Stefan Katzenbeisser, and Johannes Buchmann. Propyla: Privacy preserving long-term secure storage. SCC '18, pages 39–48, New York, NY, USA, 2018. Association for Computing Machinery.

[GKM+22] Vipul Goyal, Abhiram Kothapalli, Elisaweta Masserova, Bryan Parno, and Yifan Song. Storing and retrieving secrets on a blockchain. In *IACR International Conference on Public-Key Cryptography*, pages 252–282. Springer, 2022.

[GKP05] Gregory R Ganger, Pradeep K Khosla, and CARNEGIE-MELLON UNIV PITTSBURGH PA. Pasis: A distributed framework for perpetually available and secure information systems, 2005. *https://apps.dtic.mil/sti/citations/ADA436245*.

[Goo19] Phil Goodwin. Tape and cloud: Solving storage problems in the zettabyte era of data. White paper, International Data Corporation (IDC), 2019.

[GW17] Venkatesan Guruswami and Mary Wootters. Repairing reed-solomon codes. *IEEE Transactions on Information Theory*, 63(9):5684–5698, 2017. *https://ieeexplore.ieee.org/document/7922614*.

[GZ19] Jiaxin Guan and Mark Zhandary. Simple schemes in the bounded storage model. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38*, pages 500–524. Springer, 2019.

[HJKY95] Amir Herzberg, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *Advances in Cryptology—CRYPT0'95: 15th Annual International Cryptology Conference Santa Barbara, California, USA, August 27–31, 1995 Proceedings 15*, pages 339–352. Springer, 1995. *https://link.springer.com/chapter/10.1007/3-540-44750-4_27*.

[Imp95] R. Impagliazzo. A personal view of average-case complexity. In *Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference*, pages 134–147, 1995.

[KK23] Ohad Klein and Ilan Komargodski. New bounds on the local leakage resilience of shamir's secret sharing scheme. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 139–170, Cham, 2023. Springer Nature Switzerland. *https://link.springer.com/chapter/10.1007/978-3-031-38557-5_5*.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[Lag23] David Lague. U.s. and china race to shield secrets from quantum computers. *Reuters*, 2023.

[LRU22] Christophe Levrat, Matthieu Rambaud, and Antoine Urban. Breaking the $t < n/3$ consensus bound: Asynchronous dynamic proactive secret sharing under honest majority. Cryptology ePrint Archive, Paper 2022/619, 2022.

[LTM+11] Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Mark Badger, and Dawn Leaf. Nist cloud computing reference architecture, 2011-09-08 00:09:00 2011.

[Mau92] Ueli M Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5:53–66, 1992.

[MGA+20] Philipp Muth, Matthias Geihs, Tolga Arul, Johannes Buchmann, and Stefan Katzenbeisser. Elsa: efficient long-term secure storage of large datasets (full version). *EURASIP Journal on Information Security*, 2020:1–20, 2020.

[MNPC+22] Hemanta K. Maji, Hai H. Nguyen, Anat Paskin-Cherniavsky, Tom Suad, Mingyuan Wang, Xiuyu Ye, and Albert Yu. Leakage-resilient linear secret-sharing against arbitrary bounded-size leakage family. In Eike Kiltz and Vinod Vaikuntanathan, editors, *Theory of Cryptography*, pages 355–383, Cham, 2022. Springer Nature Switzerland. *https://link.springer.com/chapter/10.1007/978-3-031-22318-1_13*.

[MPCSW21] Hemanta K Maji, Anat Paskin-Cherniavsky, Tom Suad, and Mingyuan Wang. Constructing locally leakage-resilient linear secret-sharing schemes. In *Annual International Cryptology Conference*, pages 779–808. Springer, 2021.

[MS04] Michael A Marsh and Fred B Schneider. Codex: A robust and secure secret distribution system. *IEEE Transactions on Dependable and secure Computing*, 1(1):34–47, 2004.

[MZW+19] Sai Krishna Deepak Maram, Fan Zhang, Lun Wang, Andrew Low, Yupeng Zhang, Ari Juels, and Dawn Song. Churp: Dynamic-committee proactive secret sharing. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2369–2386, New York, NY, USA, 2019. Association for Computing Machinery.

[NN04] Ventzislav Nikov and Svetla Nikova. On proactive secret sharing schemes. In *Proceedings of the 11th International Conference on Selected Areas in Cryptography*, SAC'04, page 308–325, Berlin, Heidelberg, 2004. Springer-Verlag.

[Noo23] Greg Noone. Are harvest now, decrypt later cyberattacks actually happening? *Tech Monitor*, 2023.

[OY91] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks (extended abstract). In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '91, pages 51–59, New York, NY, USA, 1991. Association for Computing Machinery. *https://doi.org/10.1145/112600.112605*.

[Ped91] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.

[pro24] Project silica, April 2024.

[Raz18] Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. *Journal of the ACM (JACM)*, 66(1):1–18, 2018.

[RP11] Jason K. Resch and James S. Plank. Aont-rs: blending security and performance in dispersed storage systems. In *Proceedings of the 9th USENIX Conference on File and Stroage Technologies*, FAST'11, page 14, USA, 2011. USENIX Association.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978.

[SB05] Arun Subbiah and Douglas M. Blough. An approach for fault tolerant and secure data storage in collaborative work environments. In *Proceedings of the 2005 ACM Workshop on Storage Security and Survivability*, StorageSS '05, page 84–93, New York, NY, USA, 2005. Association for Computing Machinery.

[SGM06] Mark W. Storer, Kevin M. Greenan, and Ethan L. Miller. Long-term threats to secure archives. In Ethan L. Miller and Erez Zadok, editors, *Proceedings of the 2006 ACM Workshop On Storage Security And Survivability, StorageSS 2006, Alexandria, VA, USA, October 30, 2006*, pages 9–16. ACM, 2006.

[SGMV09] Mark W. Storer, Kevin M. Greenan, Ethan L. Miller, and Kaladhar Voruganti. Potshards—a secure, recoverable, long-term archival storage system. *ACM Trans. Storage*, 5(2), jun 2009. *https://dl.acm.org/doi/10.1145/1534912.1534914*.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, nov 1979. *https://doi.org/10.1145/359168.359176*.

[Sho99] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.*, 41(2):303–332, 1999.

[SLL10] David Schultz, Barbara Liskov, and Moses Liskov. Mpss: Mobile proactive secret sharing. *ACM Trans. Inf. Syst. Secur.*, 13(4), dec 2010.

[SS22] Moe Sabry and Reza Samavi. Archivesafe lt: Secure long-term archiving system. ACSAC '22, pages 936–948, New York, NY, USA, 2022. Association for Computing Machinery.

[SSA+09] Marc Stevens, Alexander Sotirov, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, and Benne de Weger. Short chosen-prefix collisions for md5 and the creation of a rogue ca certificate. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 55–69, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[STD+24] Christopher Smith, Maliha Tabassum, Soumya Chowdary Daruru, Gaurav Kulhare, Arvin Wang, Ethan L. Miller, and Erez Zadok. Secure archival is hard... really hard. In *Proceedings of the 16th ACM Workshop on Hot Topics in Storage and File Systems*, HotStorage '24, page 38–46, New York, NY, USA, 2024. Association for Computing Machinery.

[Tow22] Kevin Townsend. Solving the quantum decryption 'harvest now, decrypt later' problem. *SecurityWeek*, 2022.

[VAFB22] Robin Vassantlal, Eduardo Alchieri, Bernardo Ferreira, and Alysson Bessani. Cobra: Dynamic proactive secret sharing for confidential bft services. In *2022 IEEE symposium on security and privacy (SP)*, pages 1335–1353. IEEE, 2022.

[VBC+15] Martín Vigil, Johannes Buchmann, Daniel Cabarcas, Christian Weinert, and Alexander Wiesmaier. Integrity, authenticity, non-repudiation, and proof of existence for long-term archiving: A survey. *Computers & Security*, 50:16–32, 2015.

[Wik24] Wikipedia contributors. Multicloud — Wikipedia, the free encyclopedia. *https://en.wikipedia.org/w/index.php?title=Multicloud&oldid=1232791299*, 2024. [Online; accessed 19-July-2024].

[WWW02] T.M. Wong, Chenxi Wang, and J.M. Wing. Verifiable secret redistribution for archive systems. In *First International IEEE Security in Storage Workshop, 2002. Proceedings.*, pages 94–105, 2002. *https://ieeexplore.ieee.org/document/1183515*.

[YXD22] Yunzhou Yan, Yu Xia, and Srinivas Devadas. Shanrang: Fully asynchronous proactive secret sharing with dynamic committees. *Cryptology ePrint Archive*, 2022.

[YXXM23] Thomas Yurek, Zhuolun Xiang, Yu Xia, and Andrew Miller. Long live the honey badger: Robust asynchronous {DPSS} and its applications. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 5413–5430, 2023.

[ZČD+16] Jingyu Zhang, A Čerkauskaitė, Rokas Drevinskas, Aabid Patel, Martynas Beresna, and Peter G Kazansky. Eternal 5d data storage by ultrafast laser writing in glass. In *Laser-based Micro-and Nanoprocessing X*, volume 9736, pages 163–178. Spie, 2016.

A Algebra Preliminaries

Informally, a *field* is a set of elements that support all the usual arithmetic operations of addition, subtraction, multiplication and division (by non-zero elements). Popular examples include $\mathbb{Q}, \mathbb{R}$, and $\mathbb{C}$. A *finite* field is a field where the set of elements is finite. The canonical example is $\mathbb{Z}_p$: the set $\{0, 1, ..., p-1\}$ with integer addition and multiplication modulo a prime p. Many cryptographic algorithms, such as Shamir's secret sharing, perform arithmetic over a finite field instead of some infinite field like the reals. There are many reasons why this is the case. For Shamir's secret sharing, use of a field allows polynomial interpolation. Use of a finite field allows for a uniform distribution (it is easy to prove that the uniform distribution cannot exist over any infinite set). Shamir's secret sharing is covered below (Section 4), in this section we provide the requisite algebra background for understanding finite fields and computing over them.

We begin with a formal definition of a field.

Definition A.1 (Field [DF03])**.** A *field* is a set $\mathbb{F}$ together with two binary operations $+$ and $\cdot$ (called addition and multiplication, respectively) on $\mathbb{F}$ such that the following properties hold:

1. $(\mathbb{F}, +)$ is an abelian group (call its identity 0)
2. $(\mathbb{F} - \{0\}, \cdot)$ is also an abelian group
3. The following distributive law holds for all $a, b, c \in \mathbb{F}$:

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

Clearly, one must understand what an abelian group is in order to understand Definition A.1.

Definition A.2 ((Abelian) Group [DF03])**.** A *group* is an ordered pair $(G, \cdot)$, where G is a set and $\cdot$ is a binary operation on G (i.e., a map $G \times G \to G$) satisfying the following axioms:

1. $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for all $a, b, c \in G$ i.e., G is associative.
2. There exists an element $e \in G$, called an identity of G, such that for all $a \in G$ we have $a \cdot e = e \cdot a = a$.
3. For each $a \in G$ there exists an element $a^{-1} \in G$, called the inverse of a, such that $a \cdot a^{-1} = a^{-1} \cdot a = 1$.

The group $(G, \cdot)$ is called abelian, or commutative, if $a \cdot b = b \cdot a$ for all $a, b \in G$. For convenience we abuse notation and refer to G as the group $(G, \cdot)$ when it is clear from context what $\cdot$ is.

Common examples of an abelian group include $(\mathbb{Z}, +)$ and $(\mathbb{Z}_n, +_n)$, where $+$ refers to the usual integer addition, $\mathbb{Z}_n$ refers to the set of integers $\{0, 1, ..., n-1\}$, and $+_n$ refers to the usual integer addition followed by taking the remainder modulo n. It is important to note that the integers under integer multiplication (or $\mathbb{Z}_n$ under integer multiplication modulo n) do not form a group as zero has no multiplicative inverse. Even if we consider $\mathbb{Z}_n - \{0\}$ under multiplication modulo n we still do

not get a group *in the general case* (see that the element $2 \in \mathbb{Z}_4 - \{0\}$ has no multiplicative inverse modulo 4). If we restrict ourselves to a prime modulus p, however, all of a sudden $(\mathbb{Z}_p - \{0\}, \times_p)$ is an abelian group. This popular group, denoted $\mathbb{Z}_p^*$, is important enough for our purposes that we supply a quick proof of this fact.

Claim A.1. *Z_p^* is an abelian group.*

Proof. We must verify that Z_p^* satisfies the properties of an abelian group from Definition A.2. First, the operation of multiplication modulo p is associative and commutative because modular arithmetic is associative and commutative. Second, $1 \in \mathbb{Z}_p$ is obviously the identity. Third, every element $a \in \mathbb{Z}_p$ has an inverse because $\gcd(a, p) = 1$, so there exist integers x, y (Bézout coefficients, computable via extended Euclidean algorithm) such that $xa + yp = 1$, implying that $xa = 1 \pmod{p}$, and therefore $a^{-1} = x \pmod{p}$ □

Taken together with the (trivial) fact that $(\mathbb{Z}_p, +_p)$ is an abelian group, we arrive at an easily verifiable corollary stated at the beginning of the section that $\mathbb{Z}_p$ is a finite field (just need to show distributivity, which follows from distributivity over $\mathbb{Z}$, which can be proven by induction).

Here we have only discussed the basic definition of a finite field, and provided only the construction of prime fields $\mathbb{Z}_p$. This is sufficient for purposes of this work. It is important to note that many applications use more advanced *extension* fields such as $GF(2^8)$. For more information we refer the reader to any standard algebra text [DF03].

B Secret Sharing Proofs

Claim B.1 (Security of Additive Secret Sharing)**.** *The distribution scheme described in 4.2 is a secret-sharing scheme realizing an (n, n) threshold access structure.*

Proof. We first prove correctness (using Definition 4.3. There is only one authorized set $B = [n] \in A$. Consider the reconstruction function $\textsc{Recon}_B : K_1 \times \cdots \times K_n \to K$ that works as follows. On input shares $(r_1, ..., r_n) \coloneqq \Pi(k, r)_B$, return $r_1 + \cdots + r_n =: k'$. Clearly $k = k'$ by construction.

Now we prove privacy (using Definition 4.4). Let $T \subsetneq [n]$ be any unauthorized set. It suffices to show that S and $S_T \coloneqq \Pi(S, R)_T$ are independent. Let S_i be the random variable corresponding to the secret share held by party i. To prove independence we will show that, for arbitrary secret a and secret shares $(r_i)_{i \in T}$:

$$\Pr[S = a \mid S_T = (r_i)_{i \in T}] = \Pr[S = a] \tag{B.1}$$

Observe that $S = \sum_{i\in[n]} S_i$ and $S_T = (S_i)_{i\in T}$, so we obtain from the LHS of (B.1):

$$\Pr\left[\sum_{i\in[n]} S_i = a \;\middle|\; S_i = r_i\ \forall i \in T\right] \tag{B.2}$$

$$= \Pr\left[\sum_{i\notin T} S_i = a - \sum_{i\in T} r_i\right] \tag{B.3}$$

$$= \Pr\left[S_{i^*} = a - \sum_{i\in T} r_i - \sum_{i\notin T, i\neq i^*} S_i\right] \tag{B.4}$$

$$= \sum_{(r'_i)_{i\notin T, i\neq i^*}} \Pr\left[S_i = r'_i\ \forall i \notin T, i \neq i^*\right]$$

$$\cdot \Pr\left[S_{i^*} = a - \sum_{i\in T} r_i - \sum_{i\notin T, i\neq i^*} S_i \;\middle|\; S_i = r'_i\ \forall i \notin T, i \neq i^*\right] \tag{B.5}$$

$$= \sum_{(r'_i)_{i\notin T, i\neq i^*}} \Pr\left[S_i = r'_i\ \forall i \notin T, i \neq i^*\right]$$

$$\cdot \Pr\left[S_{i^*} = a - \sum_{i\in T} r_i - \sum_{i\notin T, i\neq i^*} r'_i\right] \tag{B.6}$$

$$= \Pr[S = a] \sum_{(r'_i)_{i\notin T, i\neq i^*}} \Pr\left[S_i = r'_i\ \forall i \notin T, i \neq i^*\right] \tag{B.7}$$

$$= \Pr[S = a], \tag{B.8}$$

where the i^* appearing in (B.4) is any arbitrary party index not in T, and (B.7) follows because the event $S_{i^*} = a - \sum_{i\in T} r_i - \sum_{i\notin T, i\neq i^*} r'_i$ occurs if and only if $S = a$. This is because there are q possibilities for S_{i^*} and q possibilities for S, so only one value of S_{i^*} results in a being the secret, and conversely each value of S uniquely determines the value of S_{i^*} (given the conditioning of the formula in (B.7). □

Theorem B.1 (Lagrange Interpolation Theorem)**.** *For every field $\mathbb{F}$, every t distinct points $x_1, ..., x_t$, and any t values $y_1, ..., y_t$, there exists a unique polynomial p of degree at most $t-1$ with coefficients over $\mathbb{F}$ (i.e., $p \in \mathbb{F}[X]^{\leq t-1}$) such that $p(x_i) = y_i$ for every $1 \leq i \leq t$.*

Proof. First we show such an interpolating polynomial exists. For each $x_i \in \{x_1, ..., x_t\}$, let $\ell_i(x) \coloneqq \prod_{j\neq i} \frac{(x-x_j}{x_i-x_j}$ be the i-th Lagrange basis polynomial. See that $\ell_i(x) = 0$ if $x \neq x_1$, and $\ell_i(x) = 1$ if $x = x_i$ (so ℓ_i is an indicator function for x_i). Each basis polynomial has degree $t-1$, so the sum $p(x) = \sum_{1\leq 1\leq t} y_t \ell_i(x)$ is an at most degree $t-1$ polynomial satisfying $p(x_i) = y_i$ for every $1 \leq i \leq t$.

Now we show this polynomial is unique. Assume we have two polynomials p and q of degree at most $t-1$ such that $p(x_i) = q(x_i) = y_i$ for every $1 \leq i \leq t$. Consider the polynomial $h \coloneqq p - q$. This polynomial has degree at most $t-1$, but it also has t roots (namely, $x_1, ..., x_t$)). Thus, h must be identically 0, so $p = q$, and therefore the interpolating polynomial is unique. □

Claim B.2 (Security of Shamir's Secret Sharing)**.** *Shamir's distribution scheme is a secret-sharing scheme realizing a (t, n) threshold access structure.*

Proof. First we prove correctness. Let $B \subseteq [n]$ be any authorized set of parties holding a set of shares $\Pi(k, r)_B =: ((x_{i_1}, p(x_{i_1})), ..., (x_{i_{|B|}}, p(x_{i_b})))$ of secret $k \in \mathbb{F}_q$. Since $|B| \geq t$, we can take any t of these shares and apply Lagrange interpolation (Theorem B.1) to recover the unique polynomial p that produced these shares. We can actually directly recover the secret without fully interpolating p by evaluating each Lagrange basis polynomial at 0: $\ell_i(0) = \prod_{j \neq i} \frac{x_j}{x_i - x_j}$, and computing $k = p(0) = \sum_i p(x_i)\ell(0)$.

The proof of privacy is analogous to that of additive secret sharing. Let $T \subseteq [n]$ be any unauthorized set. Let S denote the random variable corresponding to the secret, and $S_T \coloneqq \Pi(S, R)_T$ be the random variable corresponding to the shares held by parties in T. Shamir's secret shares are of the form $(x_i, p(x_i))$, but the x_i are public so randomness is only over the second coordinate, which we will denote by Y_i (i.e., $S_T = (Y_i)_{i \in T}$). It is common to leave evaluation points implicit like this and identify a share with its second coordinate. Finally, let λ_i denote the evaluation of the i-th Lagrange basis polynomial at 0 ($\lambda_i \coloneqq \ell_i(0)$). It suffices to show that S and S_T are independent. Observe $S = \sum_{i \in [n]} \lambda_i Y_i$, so we have, for any secret $a \in K$ and any shares $(y_i)_{i \in T}$:

$$\Pr\left[S = a \mid S_T = (y_i)_{i \in T}\right] \tag{B.9}$$

$$= \Pr\left[\sum_{i \in [n]} \lambda_i Y_i = a \;\middle|\; Y_i = y_i \; \forall i \in T\right] \tag{B.10}$$

$$= \Pr\left[\sum_{i \notin T} \lambda_i Y_i + \sum_{i \in T} \lambda_i y_i = a\right] \tag{B.11}$$

$$= \Pr\left[\lambda_{i^*} Y_{i^*} + \sum_{i \in T} \lambda_i y_i + \sum_{i \notin T, i \neq i^*} \lambda_i Y_i = a\right] \tag{B.12}$$

$$= \sum_{(y'_i)_{i \notin T, i \neq i^*}} \Pr\left[Y_i = y'_i \; \forall i \notin T, i \neq i^*\right] \tag{B.13}$$

$$\cdot \Pr\left[\lambda_{i^*} Y_{i^*} + \sum_{i \in T} \lambda_i y_i + \sum_{i \notin T, i \neq i^*} \lambda_i y'_i = a\right] \tag{B.14}$$

$$= \Pr[S = a] \cdot \sum_{(y'_i)_{i \notin T, i \neq i^*}} \Pr\left[Y_i = y'_i \; \forall i \notin T, i \neq i^*\right] \tag{B.15}$$

$$= \Pr[S = a] \tag{B.16}$$

□

Claim B.3. *Let $\mathbb{F}_q$ be a finite field, let $\vec{X} \coloneqq (x_1, ..., x_{k+1})$ be a set of $k < q$ non-zero distinct field elements, and let $\textsc{Recon}_{\vec{X}} \colon \mathbb{F}_q^{k+1} \to \mathbb{F}_q[x]^{\leq k}$ be the function that interpolates $k+1$ field elements into a polynomial over $\mathbb{F}_q$ of degree at most k. Then $\textsc{Recon}_{\vec{X}}$ is an additive homomorphism.*

Proof. Let $(f_1, ..., f_{k+1})$ and $(g_1, ..., g_{k+1})$ each be an arbitrary list of points in $\mathbb{F}_q^{k+1}$. Let $f \coloneqq \text{RECON}_{\vec{X}}(f_1, ..., f_{k+1})$ and $g \coloneqq \text{RECON}_{\vec{X}}(g_1, ..., g_{k+1})$ be their corresponding interpolated polynomials, and define $h \coloneqq \text{RECON}_{\vec{X}}((f_1, ..., f_{k+1}) + (g_1, ..., g_{k+1})) = \text{RECON}_{\vec{X}}(f_1+g_1, ..., f_{k+1}+g_{k+1})$. See that $h(x_i) = f_i+g_i = f(x_i)+g(x_i) = (f+g)(x_i)$ for $1 \leq i \leq k+1$. Because h agrees with $f+g$ on $k+1$ points, by the interpolation theorem (B.1) $h = f + g$. In other words, $\text{RECON}_{\vec{X}}((f_1, ..., f_{k+1}) + (g_1, ..., g_{k+1})) = \text{RECON}_{\vec{X}}(f_1, ..., f_{k+1}) + \text{RECON}_{\vec{X}}(g_1, ..., g_{k+1})$, as desired. □

C Commitments

Informally, a commitment scheme is a two-phase (commit and open) protocol between a sender and a receiver, where the sender can send a commitment to a value to the receiver, and later open the commitment to the value upon the receiver's request. Commitment schemes should satisfy the *hiding* and *binding* properties. Hiding means that the receiver should learn no information about the committed value from the commitment it received. Binding means that the sender cannot open the commitment to a value different from the one it committed to. As discussed in Section 5.2, at most one of the hiding and binding properties can be information-theoretically secure. We now provide the construction of the Pedersen commitment central to the VSS scheme in Section 5.2.

Let G be a finite group of size $|G| = q$ (assume for simplicity q is prime) with generator g where the discrete logarithm problem is considered hard. That is, given g^x, where $x \in \mathbb{Z}_q$ is chosen uniformly at random, any PPT adversary with access to G, q, g, g^x has negligible probability of guessing x. For a more formal definition of the discrete logarithm problem, see Katz and Lindell [KL14]. The Pedersen commitment scheme is a tuple of algorithms $(\text{SETUP}, \text{COMMIT}, \text{OPEN}, \text{VERIFY})$ that operate as follows:

- (**Sender**) $\text{SETUP}(1^\lambda) \to pp$: Output public parameters $pp = (G, q, g, h)$ (held by sender and receiver), where G is a group of prime size q, g and h are randomly chosen generators (so $\log_g h$ is unknown), and the discrete logarithm problem is hard in G. The security parameter is λ.
- (**Sender**) $\text{COMMIT}(pp, x \in \mathbb{Z}_q) \to (c, w)$: Sample $r \in \mathbb{Z}_q$ uniformly at random, and output (c, w), where $c \coloneqq g^x h^r \in G$ is the commitment sent to the receiver, and $w = (x, r)$ is the witness to be used by the sender during the open phase.
- (**Sender**) $\text{OPEN}(pp, w) \to w$: Output $w = (x, r)$ and send to receiver.
- (**Receiver**) $\text{VERIFY}(pp, c, w = (x, r)) \to \{0, 1\}$: Output 1 if $g^x h^r = c$, else 0.

Pedersen commitments provide perfect hiding, and computational binding.

Claim C.1 (Security of Pedersen Commitment)**.** *The Pedersen commitment scheme described above satisfies perfect hiding, i.e. a receiver holding $pp = (G, q, g, h)$ and $c = g^x h^r$ learns no information about x, and computational binding, i.e. a sender who cannot solve the discrete log problem in G cannot open c to a value other than (x, r) and pass verification.*

Proof. For hiding, it suffices to show that c is distributed uniformly in G. See that $\Pr[c = g^x h^r] = \Pr[h^r = c \cdot g^{-x}] = \Pr[r = \log_h(c \cdot g^{-x})] = 1/q$, as r is distributed uniformly in G.

For binding, assume that the sender was able to open c to a value $(x', r') \neq (x, r)$ such that $g^{x'}h^{r'} = g^x h^r$, and let $\alpha = \log_g h$. Then the sender can set $g^{x'-x} = g^{\alpha(r-r')}$, and because the sender knows x, x', r, r', they can solve for $\alpha = (x - x')(r - r')^{-1}$ (recall q is prime so this inverse always exists), thus solving the discrete logarithm problem for g and h. □

Pedersen commitments enjoy a homomorphic property that allow them to be extended to a *polynomial commitment scheme*. As the name suggests, the sender can commit to an entire polynomial. Additionally, the receiver can query the sender for evaluations of the polynomial at a particular point, and use the commitment to verify that the received point is indeed a point on the committed polynomial. The way this is done is very straightforward. Let $(G, q, g, h) \leftarrow \textsc{Setup}(1^\lambda)$, and $p(x) = a_0 + a_1 x + \cdots + a_d x^d$ be a degree d polynomial with coefficients in $\mathbb{Z}_q$ A sender can commit to p by sampling a random degree d polyomial $r(x) = r_0 + r_1 x + \cdots + r_d x^d$, and Pedersen committing to the i-th coefficient of p using the i-th coefficient of r as randomness, yielding a commitment $c \coloneqq (g^{a_0}h^{r_0}, ..., g^{a_d}h^{r_d})$. If the receiver queries the sender for the evaluation of p at a point $i \in \mathbb{Z}_q$, an honest sender can respond with $(p(i), r(i))$. It follows that a receiver can check whether the response is correct by computing $g^{p(i)}h^{r(i)}$, and testing for equality against $(g^{a_0}h^{r_0}) \cdot (g^{a_1}h^{r_1})^i \cdots (g^{a_d}h^{r_d})^{i^d}$, since homomorphic properties of exponents mean this is equal to $g^{a_0+a_1(i)+\cdots+a_d(i^d)}h^{r_0+r_1(i)+\cdots+r_d(i^d)} = g^{p(i)}h^{r(i)}$.