Middleware for a Re-configurable Distributed Archival Store Based on Secret Sharing

Shiva Chaitanya[1] Dharani Vijayakumar[2] Bhuvan Urgaonkar[3]
Anand Sivasubramaniam[3]

[1] Netapp Inc.
[2] VMware Inc.
[3] The Pennsylvania State University

Abstract — Modern storage systems are often faced with complex trade-offs between the confidentiality, availability, and performance they offer their users. Secret sharing is a data encoding technique that provides information-theoretically provable guarantees on confidentiality unlike conventional encryption. Additionally, secret sharing provides quantifiable guarantees on the availability of the encoded data. We argue that these properties make secret sharing-based encoding of data particularly suitable for the design of increasingly popular and important distributed archival data stores. These guarantees, however, come at the cost of increased resource consumption during reads/writes. Consequently, it is desirable that such a storage system employ techniques that could dynamically transform data representation to operate the store within required confidentiality, availability, and performance regimes (or budgets) despite changes to the operating environment. Since state-of-the-art transformation techniques suffer from prohibitive data transfer overheads, we develop a middleware for dynamic data transformation. Using this, we propose the design and operation of a secure, available, and tunable distributed archival store called FlexArchive. Using a combination of analysis and empirical evaluation, we demonstrate the feasibility of our archival store. In particular, we demonstrate that FlexArchive can achieve dynamic data re-configurations in significantly lower times (factor of 50 or more) without any sacrifice in confidentiality and with a negligible loss in availability (less than 1%).

Keywords: Secret sharing, archival storage, confidentiality, performance, availability.

1 Introduction

The last decade has witnessed a deluge of digital data that need to be safely archived for future generations [**?**]. Rapid increase in sensitive online data such as health-care, customer, and financial records has contributed to this unprecedented growth. The challenges facing such archival data stem from the need to ensure their long-term confidentiality and availability. Many factors mandate these requirements, ranging from preservation, retrieval, and security properties demanded by legislation to long lifetimes expected for cultural and family heritage data. To address data confidentiality, modern storage systems typically employ encryption-based techniques (see survey paper [**?**]). The use of data encryption for archival lifetimes, however, introduces problems that have been well-documented [**?**,**?**]. The primary drawback is that data secured using

keyed cryptography are only *computationally secure*—they are decipherable via cryptanalysis given sufficient computing power/time.

Secret sharing is a data encoding technique that offers the promise of overcoming these shortcomings of an encryption-based archival storage. Secret sharing with parameters (m, n) breaks a data block into n fragments (each of the same size as the original data block) in such a manner that at least m fragments must be obtained to re-construct the original block. These fragments are stored in n different storage nodes and an adversary has to obtain access to at least m fragments to decipher the original data - any set of fewer than m fragments provides no information about the original block. This property provides a quantitative notion of data confidentiality. Additionally, the original data item is resilient to the loss of fragments in the following manner: it can be re-constructed even when $(n - m)$ fragments are lost. This provides a quantitative measure of the availability properties of encoded data.

In this paper, we address the important problem of dynamically re-configuring the secret sharing parameters (m, n) used to encode data in a distributed archival store. The need to re-configure could arise as a result of one or more of the following scenarios. First, a subset of the storage nodes comprising the archival store might become unavailable or unreliable due to some form of security compromise or component failure. The data fragments at affected nodes must be considered lost and the archival system must be reverted back to its original settings. Second, there could be infrastructural changes to the storage network (e.g., addition of new nodes) which are likely to happen quite frequently relative to the lifetime of the archival data. Finally, the secrecy, availability, or performance needs of an archival store might change with time (e.g., due to changes in regulations or societal changes resulting in the stored data becoming more sensitive). Existing archival systems that have incorporated secret sharing to achieve the goals of secure long-term preservation of data have either (i) neglected this problem of re-configuration (e.g., Potshards [**?**]), or (ii) proposed inefficient techniques (e.g., PASIS [**?**]). Whereas some key aspects of the problem of dynamically re-configuring secret sharing parameters have been studied [**?**,**?**,**?**,**?**], the approaches emerging out of this body of work have severe drawbacks when used to build a practical archival storage system. In particular, they suffer from the following two main drawbacks:

- **High data access overhead.** Existing re-configuration techniques require access to m fragments for every data object stored using a (m, n) configuration. Many archival storage systems store data across wide-area networks (often with components that need to be accessed via congested or inherently slow links) and use cheap storage media technologies wherein reads to the original data can be quite slow. As we will observe later in this paper, for archival storage, it is desirable for the value of m to be close to n. Thus, the data traffic resulting from a re-configuration can become a limiting factor.
- **High computational overhead.** Existing re-configuration techniques suffer from high computational overheads. These overheads could be prohibitive in the context of archival systems that deal with very large volumes of data. It is desired that a re-configuration technique complete fast enough so the archival system spends a small amount of time in an unstable (and hence, potentially vulnerable) configuration.

1.1 Research Contributions

Our contribution is twofold.

- We propose a re-configuration technique called Multi-share Split that is both lightweight in terms of (i) the computational and I/O overheads it imposes on the archival storage system as well as (ii) tunable in terms of the trade-offs it offers between confidentiality, availability, and performance. We expect that Multi-share Split would enable administrators of archival stores to make appropriate choices to best satisfy the requirements (e.g, completion time targets, network resource constraints) of their systems.
- Using our re-configuration technique, we design and implement a middleware that is used by nodes comprising a distributed archival storage system called FlexArchive. We analyze the security and availability properties offered by FlexArchive and conduct an empirical evaluation of the feasibility and efficacy of FlexArchive using a prototype networked storage system.

1.2 Road-map

The rest of this paper is organized as follows. We discuss some background material in Section 2. We introduce the proposed FlexArchive system in Section **??** and describe the re-configuration algorithm employed by FlexArchive in Section **??**. We develop analytical techniques for characterizing the availability offered by FlexArchive in Section **??**. We conduct an empirical evaluation of the efficacy of FlexArchive in Section **??**. Finally, we present concluding remarks in Section **??**.

2 Background and Related Work

In this section, we provide basic background on secret sharing and its appropriateness for archival storage.

2.1 Basics of Secret Sharing

An (m, n) secret sharing scheme, where $m \leq n, m > 0$, creates n fragments from a data item with the following properties: given any m fragments, one can re-construct the data item; however, fewer than m fragments provide no information about the original data item. Such classes of secret sharing techniques are "perfectly secure" in the sense that they exhibit information-theoretic security. The size of each fragment for secret sharing schemes is provably the same as that of the original data item. Hence, the storage needs are n times the size of the original data.

A number of secret sharing techniques have been proposed that differ very slightly in their computational complexity. We use a secret sharing scheme due to Shamir (often called "Shamir's threshold scheme" [**?**]). The key idea behind Shamir's threshold scheme is that m points are needed to define a polynomial of degree $(m - 1)$ (e.g., two points for a line, three points for a hyperbola, four points for a third-degree polynomial, and so forth). Shamir's threshold scheme, for representing a data item S

with secret sharing parameters (m, n), chooses uniformly random $(m-1)$ coefficients $a_1, \cdots, a_{m-1}$, and lets $a_0 = S$. It then builds the polynomial $f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{m-1} x^{m-1}$. Finally, it computes the values taken by this polynomial for n distinct values of x comprising the set $\{x_1, \cdots, x_n\}$. The n shares of the secret S are now given by the pairs $(x_i, f(x_i))$. Given any m of these pairs, one can find the coefficients of the polynomial $f(.)$ by interpolation, and then evaluate the secret $S = a_0$. Geometrically, on a $X - Y$ plane, one can think of the secret S as being the Y-intercept of the curve defined by the polynomial $f(.)$; the shares are the Y-values at $x_1, \cdots, x_n$. Note that since we are dealing with finite values (the secret and the shares are data values and represented by say, q bits), the $X - Y$ plane is a finite field with the range of values 0 to $2^q - 1$. Since the participants holding the shares could implicitly define the n indices for which the shares are computed (e.g., based on their unique names/identities), each share is simply the value $f(x_i)$ and hence can be represented using q bits, same as those needed for the secret S.

2.2 Long Term Data Confidentiality

Two fundamental classes of mechanisms for enforcing data secrecy are those based on encryption and secret sharing, respectively. Many systems such as OceanStore [**?**], FARSITE [**?**], SNAD [**?**], Plutus [**?**], and e-Vault [**?**] address file secrecy, but rely on the explicit use of keyed encryption. Keyed encryption may work reasonably well for short-term secrecy needs but it is less than ideal for the long-term security problem that the current work addresses. Keyed cryptography is only computationally secure, so compromise of an archive of encrypted data is a potential problem regardless of the encryption algorithm used. An adversary who compromises an encrypted archive need only wait for cryptanalysis techniques to catch up with the encryption used at the time of the compromise. If an insider at a given archive gains access to all of its data, he can decrypt any desired information even if the data is subsequently re-encrypted by the archive, since the insider will have access to the new key by virtue of his internal access. Encrypted data can be deciphered by anyone, given sufficient CPU cycles and advances in cryptanalysis. Furthermore, future advances in quantum computing have the potential to make many modern cryptographic algorithms obsolete. For long-lasting applications, encryption also introduces the problems of lost keys, compromised keys, and even compromised crypto-systems. Additionally, the management of keys becomes difficult because data might experience many key rotations and crypto-system migrations over the course of several decades. This must all be done without user intervention because the user who stored the data may be unavailable.

Moving away from encryption to secret sharing enables an archival storage system to rely on the more flexible and secure authentication realm. Unlike encryption, authentication need not be done by a computer and authentication schemes can be easily changed in response to new vulnerabilities. Secret sharing improves the security guarantees by forcing an adversary to breach *multiple* archival sites to obtain meaningful information about the data. Several recently proposed archival systems such as POTSHARDS [**?**], PASIS [**?**,**?**], and GridSharing [**?**] employ secret sharing schemes for this reason.