Integrity Protections for Secure Long-term Storage

Maliha Tabassum

Department of Computer Science

Stony Brook University

Doctoral Program

Research Proficiency Exam

December 2024

Technical Report FSL-24-05

Contents

Abstract

Long-term digital data preservation is often confronted with security challenges that persist for decades. The security assurance of most storage solutions available today rely on practical yet unproven mathematical assumptions. Therefore, these solutions fall short of meeting the protection requirements of data that demand to be safeguarded for longer than a human lifespan. The three key components of long-term data security—confidentiality, integrity, and availability—are in danger now more than ever due to the rapid advancements in technology (*e.g.*, the emergence of powerful quantum computers). Although researchers have explored the long-term confidentiality and availability of highly sensitive information, data integrity in the long run has received inadequate attention. As a result, conventional approaches to ensure long-term integrity have remained vulnerable to cryptographic obsolescence. In this technical report, we investigate techniques that have the potential to serve as the central component of a robust data-integrity mechanism for long-lived data and discuss their application to secure storage systems.

Chapter 1

Introduction

Digital archival systems provide an efficient solution for securely storing data for a long time by allowing easy access and requiring less space. The trade-off is that storing data in the digital format expands the attack surface and makes data more vulnerable to stealing and tampering attacks. While several techniques to ensure the security of data in digital format have existed for a few decades [Sha84, GPSW06, Pai99, ABC+07, SW08], the problem arises when data must be preserved for a *really* long time (*e.g.*, decades or even centuries). Computationally secure cryptographic primitives that are currently used for data security rely on practical yet unproven mathematical assumptions and, therefore, do not meet the requirements of long-term archival systems that aim to store data securely for at least a hundred years.

Readers may wonder at this point about the type of data that must be securely stored for so long. There are plenty of instances where data demand to be safeguarded for decades or centuries. For example, the oldest continuing agricultural experiment, The Broadbalk Wheat Experiment, started in 1843 and is still ongoing at Rothamsted Research, UK [PJG+24]. Although the results of the experiment are published periodically, maintaining the authenticity of data over time and preventing any inadvertent or malicious alterations are crucial. Another example is trade secrets such as Coca-Cola's recipe, that has been kept hidden for more than a hundred years even from most of their employees [coc24]. In the case of trade secrets, data must remain both confidential and unchanged. Public health records that have particular significance to the study of the causes of disease are yet another type of data that must be shielded against any tampering attacks. Global weather/climate history, presidential records, legal documents, court orders, property deeds, birth and population records, and family medical history are some other examples of data that call for secure long-term preservation.

The security of long-lived digital data comprises three key components: confidentiality, integrity, and availability, also known as the CIA triad. Confidentiality refers to the hiding property of data that prevents an unauthorized party from gleaning information. Integrity ensures the trustworthiness of data and protects against data tampering attacks. Availability means that data is accessible whenever needed regardless of network failures, system crashes, or other unanticipated failure events. The confidentiality and availability of long-lived data have managed to capture the interest of researchers over the years [WWW02, GKP05, SGMV09, BBD+17, BDF+20, ZWGG23]. The long-term integrity of data, however, remained largely unexplored and is the primary focus of our work [STD+24].

We now present a formal definition of data integrity, followed by a real-world example of data tampering attempts. According to National Institute of Standards and Technology (NIST), data integrity is the property that data has not been altered in an unauthorized manner in transit, during processing, or at rest [GGF17]. An infamous attack example on the integrity of long-term data is the modification of a US presidential record. In 2011, the US National Archives discovered that an established Lincoln researcher altered a presidential pardon, part of the archive's permanent records, in order to support a claim in his book [nat11].

President Abraham Lincoln signed 128 pardons during his presidency, including the pardon of Patrick Murphy, who was sentenced to death. The pardon statement contained the signature of the president and the date it was signed: April 14, 1865, the day the president was later assassinated [HSL21]. The number "5" in the year, however, looked like a darker shade of ink than the rest of the date and raised suspicion. Moreover, transcriptions of the document, published in 1953, listed the date as April 14, 1864, which added credence to the concern. It was later determined that the original date of the pardon was indeed April 14, 1864, a year prior to the president's demise. The researcher changed the date so that he could support his claim about the president's commitment to kindness till death [nat11]. The archivists were able to detect the tampering using forensic examination methods such as visual observations of the physical document under visible and ultraviolet radiation [HSL21]. If the document were stored only in digital format, which is the case for many important documents, detecting such an attack would have been almost impossible without a robust data-integrity mechanism, let alone recovering the original data.

In 2015, motivated by the need for long-term data preservation, Vigil *et al.* conducted a survey of the existing long-term integrity solutions and found that none of them provided protection in the long run [VBC+15]. The common limitation of the solutions was the failure to properly address the obsolescence of cryptographic primitives due to algorithmic advances or increases in computational power. For example, the DES encryption algorithm with 56-bit keys was considered secure in the research community for 20 years until the *Deep Crack* machine was introduced that could find a single DES key in 22 hours [ST21]. The MD5 hash function with 128-bit output length was believed to be collision resistant for many years until a team of Chinese cryptanalysts demonstrated an explicit collision [KL14]. The SHA-1 algorithm was adopted in 1995 as a replacement in the aftermath of collision attacks on MD5 but was practically broken by the *SHAttered* attack in 2017 [wik24].

Two other solutions for long-term integrity were proposed after the aforementioned survey that aimed to mitigate the limitations of the existing approaches. In 2017, Braun *et al.* introduced COPRIS, a data-integrity scheme, that provides long-term integrity by renewing the timestamps and commitments used in the protocol [BBD+17]. But the security of the scheme relies on the assumption that the cryptographic primitives used are computationally secure during their usage period. In 2020, Muth *et al.* presented another long-term integrity solution for secure storage that uses renewable vector commitments and timestamps [MGA+20]. Nonetheless, the primitives used are assumed to be secure throughout the duration of their employment. These two solutions addressed the aging of cryptographic primitives to some extent, but they cannot thwart tampering attacks if the algorithms are suddenly broken during their usage period.

In this technical report, we explore techniques that can potentially ensure the integrity of digital data for a long time even when the adversary is computationally unbounded. To provide background information on this work, we first outline the desired characteristics of a secure long-term storage system. We extend our study to different data integrity solutions such as digital signature schemes, commitment schemes, secret sharing schemes, and blockchain technology. Based on our work, we direct attention to promising research areas for further investigation.

The rest of the report is organized as follows. Chapter 2 discusses various types of adversaries, expands on two distinct notions of security, and describes the desired characteristics of a secure long-term archival system. Chapter 3 presents two system models for data storage, explores potential mechanisms for long-term data integrity, and discusses their application to secure storage systems. Chapter 4 includes related work. We conclude in Chapter 5 with some future research directions.

Chapter 2

Background

In this chapter, we discuss various types of adversaries that imperil the integrity of long-lived data and distinguish between two important notions of security. In addition, we describe the desired characteristics of a long-term secure archival system that aims to store data for centuries.

2.1 Threat Modeling

Adversary Types. Malicious entities can be divided into two categories according to their computational power: computationally bounded adversary and computationally unbounded adversary. The runtime of a computationally bounded adversary, also known as a probabilistic polynomial-time adversary (PPT), is upper bounded by a polynomial in the input size. Such adversaries can have significant computational advantages (*e.g.*, access to a powerful quantum computer) over honest parties. Most cryptographic primitives used for short-term storage are considered secure against PPT adversaries, assuming the time to break the primitive is much longer than the lifetime of data. Nevertheless, these adversaries can pose a serious threat to storage systems if the lifetime of data is longer than the primitive used to ensure security. Computationally unbounded adversaries, on the other hand, are not limited by their computational power and, therefore, can take advantage of parallel computing. Long-term archival systems are more vulnerable to attacks from this type of adversaries since computationally secure integrity mechanisms cannot prevent an attacker from tampering with data for an extended period of time.

Adversaries can also be classified into different types based on their motives. The goal of an attacker can be to tamper with data or to destroy it completely, causing an archival system to lose valuable information for good. Another motive of a malicious intruder could be to alter data without being detected; for example, changing digital images and audio files to portray the opposite of reality as true. The last category of adversaries in this classification comprises the ones that are concerned with monetary gain. Ransomware attacks are a common example of attacks from this type of adversaries, where a group of attackers encrypt user information with their private key to make it inaccessible and demand money from the user in return of the decryption key. An adversary, threatening the integrity of long-term storage systems, can be motivated by any of the aforementioned reasons though ransomware attacks are usually targeted at highly valuable short-lived data.

Data tampering attacks can be launched by both malicious outsiders, who first need to bypass the storage system's access control mechanism, or dishonest insiders, who already have access to the storage nodes. For example, distributed storage systems are susceptible to Byzantine failures, where a set of rogue storage nodes violate the security protocols on purpose and send incorrect information to other storage nodes [LSP19]. Other than various types of adversaries, unanticipated events such as natural disasters and media obsolescence can also result in data loss. Hardware/software glitches and human errors are additional threats that can affect the integrity of data in long-term storage systems.

Security Notions. Based on the computational power of the adversary, the security assurance of a cryptographic primitive can be divided into two types. A cryptographic scheme is considered *computationally* secure if it is secure against computationally bounded adversaries. The security guarantee of such schemes depends on unproven mathematical assumptions such as finding the discrete logarithm of a number is hard. So long as these assumptions hold true, the security of the schemes will remain intact.

In contrast, a scheme is called *information-theoretically* secure if it provides security against computationally unbounded adversaries. The security of such schemes is derived from information theory, that is the scheme can only be broken by stealing enough information about the data. In the case of information-theoretic security, no algorithmic advances can threaten the security of a scheme; therefore, the security is everlasting provided an adversary cannot steal the information needed to break the scheme. We give a formal definition of both notions below using message authentication code as an example.

Message Authentication Code [KL14, Def. 4.1, §4.2]. A message authentication code (MAC) consists of three probabilistic polynomial-time algorithm $(Gen, Mac, Vrfy)$ such that:

1. The key-generation algorithm Gen takes as input the security parameter 1^n and outputs a key k with $|k| >= n$.
2. The tag-generation algorithm Mac takes as input a key k and a message $m \in \{0,1\}^*$, and outputs a tag t. Since this algorithm may be randomized, we write this as $t \leftarrow Mac_k(m)$.
3. The deterministic verification algorithm $Vrfy$ takes as input a key k, a message m, and a tag t. It outputs a bit b, with $b = 1$ meaning *valid* and $b = 0$ meaning *invalid*. We write this as $b := Vrfy_k(m,t)$.

It is required that for every n, every key k output by $Gen(1^n)$, and every $m \in \{0,1\}^*$, it holds that

$$Vrfy_k(m, Mac_k(m)) = 1.$$

Computationally Secure MAC [KL14, pp. 112-113]. The message authentication experiment Mac-forge$_{A,\Pi}(n)$ is defined as follows:

1. A key k is generated by running $Gen(1^n)$.
2. The adversary A is given input 1^n and oracle access to $Mac_k(\cdot)$. The adversary eventually outputs (m,t). Let Q denote the set of all queries that A asked its oracle.
3. A succeeds if and only if $Vrfy_k(m,t) = 1$ and $m \notin Q$. In that case, the output of the experiment is defined to be 1.

A message authentication code $\Pi = (Gen, Mac, Vrfy)$ is computationally secure, if for all probabilistic-polynomial time adversaries A, there is a negligible function $negl$ such that:

$$Pr[\text{Mac-forge}_{A,\Pi}(n) = 1] \leq negl(n).$$

Information-theoretically Secure MAC [KL14, p. 142]. The one-time message authentication experiment Mac-forge$^{\text{1-time}}_{A,\Pi}$ is defined as follows:

1. A key k is generated by running Gen.
2. The adversary A outputs a message m' and is given in return a tag $t' \leftarrow Mac_k(m')$.
3. A outputs (m,t).

4. The output of the experiment is defined to be 1 if and only if $Vrfy_k(m,t) = 1$ and $m \neq m'$.

A message authentication code $\Pi = (Gen, Mac, Vrfy)$ is information-theoretically secure, if for all adversaries A, there is a negligible function ϵ such that:

$$\Pr[\text{Mac-forge}^{\text{1-time}}_{A,\Pi} = 1] \leq \epsilon.$$

Note that the message authentication experiments for computationally secure MAC and information-theoretically secure MAC are different in that the adversary can query the oracle only once in the latter [KL14].

In other words, a computationally secure MAC is secure *only* against computationally bounded adversaries, whereas an information-theoretically secure MAC is secure against *all* adversaries regardless of their computational power. The integrity of data in transit can be ensured using a computationally secure MAC as long as the scheme is secure throughout the duration of the protocol. Another solution is to use an information-theoretically secure MAC. But in both cases, the sender and the recipient must agree on a shared secret first [KL14]. In this article, we only focus on the integrity of data at rest, and hence, do not expand on the integrity of data in transit further.

2.2 Desired Characteristics

Since long-term archival systems are vulnerable to attacks from adversaries with unlimited computational power, the security guarantees of traditional storage systems fall short of meeting their requirements. This is not to say that long-term archival systems do not share attributes with short-term storage systems. The security assurances of long-term archives are expected to be stronger. Long-term archives differ in their protection goals as well. Below are some desired characteristics of a storage system that stores data for decades or even centuries.

Confidentiality

Data confidentiality translates to the practice of keeping data hidden from the public eye. Most storage systems store sensitive data in an encrypted form to prevent malicious entities from obtaining it. The security guarantees of these encryption schemes rely on the infeasibility of some mathematically hard problems that cannot be solved using the most advanced machines we have today. Cryptographic obsolescence, however, can make limited-access data public without giving enough time to secure it [STD+24]. Keeping the possibility of new cryptanalytic attacks in mind, some storage systems use information theoretic solutions like secret sharing. Secret sharing schemes encode data into shares such that a threshold number of shares is needed to reconstruct it. This approach doesn't depend on any unproven assumptions, but it incurs high storage cost [SGMV09].

Integrity

Data integrity is an umbrella term that covers many security features. Informally, data integrity refers to the accuracy and consistency of data. In secure archival systems, the accuracy and consistency of data also depend on the correctness of the metadata. Hence, ensuring the integrity of both data and metadata is essential. To guarantee data integrity, a storage system needs to periodically check for intrusions by malevolent insiders and outsiders. The scope of assurance of data integrity solutions can be classified into avoidance, detection, and correction [SWZ05]. Data integrity mechanisms for long-lived data should be able to resist tampering, detect unauthorized modification attempts, and recover the original data in the event of a successful intrusion.

- **Data Auditing**
  The process of evaluating the integrity of remote data is called data auditing. It can be performed by the data owner (private auditing) or by a trusted third party auditor (public auditing). Public

auditing removes the burden of periodic verification from a data owner with limited computational capacity [ZKM+17]. However, data auditing brings along additional challenges. For example, auditors should be able to perform the verification without accessing the entire data, also known as blockless verification [ABC+07]. Auditing requests should be stateless, that is the results from previous auditing requests should not be needed for subsequent verifications. For confidential data, auditing should reveal no information about the data or the owner to the public auditors [ZKM+17]. There should be no restrictions on how many verification requests can be sent to the storage service provider [ABC+07]. Data owners, however, should not be able to retrieve data while pretending to verify data integrity as frequently accessed data has a higher service charge and a dishonest data owner might try to access data this way without paying the cost. Additionally, auditing schemes should be able to verify that a storage service provider is indeed storing data as agreed upon and has not deleted some information to save space [SM06].

- **Provenance Tracking**
  Provenance is the history of ownership and authorized modifications of data [HSW09]. Provenance metadata can facilitate non-repudiation, which means a data owner cannot deny the origin of data, and by extension, his or her participation in an authorized operation on data [VBC+15]. Provenance and non-repudiation protect honest storage service providers from false accusations by data owners. A user may accidentally or intentionally modify or delete data and then wrongfully seek compensation from the storage system. Archival systems need to verify the successful completion of an authorized modification request and store the information securely. Provenance data also supports intrusion detection and improves the reliability of storage [PSR23].

Availability

Data availability means that data is accessible and retrievable from storage systems whenever needed. Hard-disk failures, software crashes, network issues, viruses, and malware are some instances that can temporarily or permanently make data inaccessible [FLP+10]. Data unavailability can also result from data deletion or tampering attacks [ZKM+17]. Failure to reach a Byzantine consensus in distributed systems can generate inconsistency in metadata, leading to data unavailability [FLP85]. Popular data availability solutions used in storage systems include replication and erasure coding. Replication simply makes copies of the data and stores it in different locations to avoid a single-point of failure. Erasure coding is similar to secret sharing in that data is encoded into data blocks and stored across multiple servers. Any subset of a threshold number of data blocks can reconstruct the data. So, an erasure coding based solution can survive the failure of a certain number of storage servers at a given time [FLP+10]. They can also detect and recover from errors that affect a small portion of data [KL14].

Access Control

Access control regulates who can view, retrieve, or modify data [ZFY+18]. For long-term storage, access control is of great consequence as the ownership of data might change over time. An adversary with malicious intentions can impersonate the data owner and request to modify or delete data. Archives should be able to distinguish between impersonators and authorized users to avert data stealing and unauthorized alterations. In the case of shared data, if there is a change in the authorized set, storage systems need to revoke access-permission of all users that are no longer part of the set [LJW+16].

Dynamic Data Handling

Contrary to popular belief, archived data is not necessarily static in nature. Archival systems must permit data modification though it may occur infrequently. Data owners should be able to insert, delete, and update information without compromising the integrity and confidentiality of data [ZKM+17]. Moreover, upon successful completion of the modification requests, stale data must be deleted securely to prevent data

leakage [TLLP12]. In addition, data consistency should be maintained, that is all authorized users should see the same copy of the modified data.

Efficiency

Storage systems that store data for decades or even centuries can accumulate high storage, communication, and computation overheads. Storage overhead is the cost of storing data and metadata, that is required to ensure confidentiality, integrity, and availability. Communication overhead is the cost of periodically sending auditing requests to the storage service provider and receiving a response. Computation overhead refers to the time and resources needed to perform various operations on data [ZKM+17]. Long-term archival systems should be affordable in the long run since the lifetime of data is rather lengthy. Deduplication is a technique that can identify and delete duplicate and redundant data, thus, improving storage utilization [ZKM+17]. Integrity verification schemes with a short proof size can significantly lower the communication overhead [SM06]. Storage systems should be able to perform batch auditing, that is processing multiple auditing requests from different users at the same time, to reduce its computational burden [WWRL10].

Scalability

Storage systems should be scalable in terms of number of users and services, amount of data stored, and rates of processing, among others. Scalability entails the ability to extend the service efficiently, while upholding its quality [JW00]. Archival systems store a copious amount of data, and the size of an archive is ever-growing since most data is not deleted. Long-term storage systems should scale according to the storage demand to accommodate new clients [JW00].

Chapter 3

Integrity of Digital Archives

In this chapter, we survey various techniques that can potentially guarantee information-theoretic integrity of long-lived data.

3.1 System Overview

We begin by describing two system models that take into account the *confidentiality* and *availability* characteristics of long-term storage systems. However, we do not consider *access control and authenticity*, *dynamic data handling*, and *efficiency and scalability* in this article and leave them as future work. There are three entities in the system:

1. Data Owner (DO): owns data and outsources it to the storage service provider.
2. Authorized User (AU): can access and view data. AUs can also serve as a third party auditor (TPA), who verifies the integrity of data at the DO's request.
3. Storage Service Provider (SSP): stores data and is responsible for its confidentiality, integrity, and availability. The SSP uses a distributed and decentralized system to store data as distributed systems provide better availability and decentralization averts the single-point of failure problem.

Computational Confidentiality Model (CCM). The secrecy requirement of this model is less than one hundred years since some data does not need to be kept secret for longer than the human lifespan. It is ideal for scenarios when the confidentiality of data is not the prime concern. CCM employs Advanced Encryption Standard (AES-256) for confidentiality and erasure coding, also known as forward error correction codes, for availability.

AES is a symmetric block cipher that can be used to encrypt electronic data [NIS01]. The AES algorithm has a block length of 128 bit and can use 128-, 192-, or 256-bit keys, with 256-bit key being the most secure. AES is considered highly secure since no practical cryptanalytic attacks better than an exhaustive search for the key have been found till to date. It is free, standardized, and an efficient encryption solution for data confidentiality [KL14].

Erasure coding breaks data down into smaller data blocks and generates additional parity blocks for data recovery. In an (m, n) erasure coding, data is split into m data blocks and using those m data blocks $n - m$ parity blocks are generated. All n blocks are then stored across multiple drives or servers. To reconstruct the data, only m blocks are needed; thus, erasure coding enables fault tolerance of up to $n - m$ servers simultaneously. Erasure coding has gained popularity in storage systems because of its improved fault tolerance and reduced storage overhead.

The combination of AES and erasure coding provides confidentiality and availability while keeping the storage cost low. For short-term storage, any computationally secure integrity scheme such as Secure Hash

Algorithm (SHA-2) can be used with this model to ensure data confidentiality, integrity, and availability. SHA-2 is a NIST-approved cryptographic hash function that takes a message as input and produces a fixed-length hash value as output. In this technical report, we only focus on long-term integrity and do not explore short-term integrity any further. The computational confidentiality model is designed for data that must remain accurate and available for longer than a century, but not necessarily secret.

Information-theoretic Confidentiality Model (ICM). This model is suitable for data that must remain confidential for at least a century. ICM deploys threshold secret sharing for both confidentiality and availability. A (t, n) threshold secret sharing scheme encodes a secret into n shares and distributes them to n parties such that any t shares can reconstruct the secret but less than t shares provide no information about the secret. Shamir's secret sharing scheme is a well-known threshold secret sharing scheme, based on polynomials over a finite field. It provides information-theoretic secrecy and fault tolerance of up to $n-t$ shares. There is an inevitable storage overhead of Shamir's secret sharing that comes with the information-theoretic secrecy guarantee, nonetheless [Sha79].

ICM can be used with both computationally secure and information-theoretically secure integrity schemes. However, since computationally secure schemes depend on unproven cryptographic hardness assumptions, combining them with an information-theoretic confidentiality scheme might compromise the secrecy. For instance, a computationally secure auditing scheme can reveal additional information about the secret-shared data even if the data cannot be reconstructed in its entirely. Care must be taken while designing a system that integrates both computationally secure schemes and information-theoretically secure schemes so as not to weaken the information-theoretic security assurances. The information-theoretic confidentiality model is ideal for data that must remain accurate, available, and confidential for longer than a century.

3.2 Integrity: Data Auditing

We now turn our attention to some cryptographic primitives that have the potential to act as the cornerstone of data integrity auditing mechanism for long-term storage.

3.2.1 Digital Signatures

Similar to message authentication codes, digital signatures are used to verify the integrity of transmitted messages. But digital signatures do not require the sender to share a distinct secret key with each potential recipient. A recipient can verify the accuracy and authenticity of a message by only using public information about the sender [KL14].

Digital Signature Schemes [KL14, Def. 12.1, §12.2]. A digital signature scheme consists of three probabilistic polynomial-time algorithms $(Gen, Sign, Vrfy)$ such that:

1. The key-generation algorithm Gen takes as input a security parameter 1^n and outputs a pair of keys (pk, sk). These are called the $public\ key$ and the $private\ key$, respectively. We assume that pk and sk each has length at least n, and that n can be determined from pk or sk.

2. The signing algorithm $Sign$ takes as input a private key sk and a message m from some message space (that may depend on pk). It outputs a signature σ, and we write it as

$$\sigma \leftarrow Sign_{sk}(m).$$

3. The deterministic verification algorithm $Vrfy$ takes as input a public key pk, a message m, and a signature σ. It outputs a bit b, with $b = 1$ meaning *valid* and $b = 0$ meaning *invalid*. We write this as

$$b := Vrfy_{pk}(m, \sigma).$$

It is required that except with negligible probability over (pk, sk) output by $Gen(1^n)$, it holds that

$$Vrfy_{pk}(m, Sign_{sk}(m)) = 1$$

for every (legal) message m. If there is a function l such that for every (pk, sk) output by $Gen(1^n)$, the message space is $\{0,1\}^{l(n)}$, then we say that $(Gen, Sign, Vrfy)$ is a signature scheme for messages of length $l(n)$. We call σ a *valid signature* on a message m with respect to some public key pk understood from the context if

$$Vrfy_{pk}(m, \sigma) = 1.$$

Security of Digital Signature Schemes [KL14, p. 443]. Let $\Pi = (Gen, Sign, Vrfy)$ be a signature scheme, and consider the following experiment for an adversary A and parameter n :

The signature experiment Sig-forge$_{A,\Pi}(n)$:

1. $Gen(1^n)$ is run to obtain keys (pk, sk).
2. Adversary A is given pk and access to an oracle $Sign_{sk}(\cdot)$. The adversary then outputs (m, σ). Let Q denote the set of all queries that A asked its oracle.
3. A succeeds if and only if $Vrfy_{pk}(m, \sigma) = 1$ and $m \notin Q$. In this case, the output of the experiment is defined to be 1.

A signature scheme $\Pi = (Gen, Sign, Vrfy)$ is secure, if for all probabilistic-polynomial time adversaries A, there is a negligible function $negl$ such that:

$$Pr[\text{Sig-forge}_{A,\Pi}(n) = 1] \leq negl(n).$$

Therefore, a signature scheme is considered secure if an adversary cannot output a valid signature σ on a message m without the private key. The adversary's runtime is assumed to be polynomially bounded [KL14]. Below we provide the construction of Lamport's Signature Scheme, that uses a one-way function, as a concrete example.

One-Way Functions [KL14, p. 243]. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a function. Consider the following experiment defined for any algorithm A and any value n for the security parameter.

The inverting experiment, $Invert_{A,f}(n)$:

1. Choose uniform $x \in \{0,1\}^n$ and compute $y := f(x)$.
2. A is given 1^n and y as input, and outputs x'.
3. The output of the experiment is defined to be 1 if $f(x') = y$, and 0 otherwise.

Note that A need not find the original preimage x; it suffices for A to find any value x' for which $f(x') = y = f(x)$. Also, A runs in time polynomial in the security parameter n, regardless of the length of y.

A function $f : \{0,1\}^* \to \{0,1\}^*$ is one-way if the following two conditions hold:

- *(Easy to Compute.)* There exists a polynomial-time algorithm M_f computing f; that is, $M_f(x) = f(x)$ for all x.
- *(Hard to Invert.)* For every probabilistic polynomial-time algorithm A, there is a negligible function $negl$ such that

$$Pr[Invert_{A,f}(n) = 1] \leq negl(n).$$

Lamport's Signature Scheme [KL14, pp. 462-463]. Let $H : \{0,1\}^* \to \{0,1\}^*$ be a one-way function. Construct a signature scheme for messages of length $l = l(n)$ as follows:

- Gen: on input 1^n, proceed as follows for $i \in \{1, ..., l\}$:
  1. Choose uniform $x_{i,0}, x_{i,1} \in \{0,1\}^n$.
  2. Compute $y_{i,0} := H(x_{i,0})$ and $y_{i,1} := H(x_{i,1})$.

  The public key pk and the private key sk are,

$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & \dots & y_{l,0} \\ y_{1,1} & y_{2,1} & \dots & y_{l,1} \end{pmatrix}$$

$$sk = \begin{pmatrix} x_{1,0} & x_{2,0} & \dots & x_{l,0} \\ x_{1,1} & x_{2,1} & \dots & x_{l,1} \end{pmatrix}.$$

- $Sign$: on input a private key sk as above and a message $m \in \{0,1\}^l$ with $m = m_1 \dots m_l$, output the signature

$$(x_{1,m_1}, \dots, x_{l,m_l}).$$

- $Vrfy$: on input a public key pk as above, a message $m \in \{0,1\}^l$ with $m = m_1 \dots m_l$, and a signature $\sigma = (x_1, \dots, x_l)$, output 1 if and only if

$$H(x_i) = y_{i,m_i}$$

  for all $1 \leq i \leq l$.

This construction is a one-time secure signature scheme, which means the private key can be used only once to securely sign a message. Here, l is any polynomial [KL14].

To securely send a message using a signature scheme, a user generates a pair of public-private keys (pk, sk) using $Gen(1^n)$ and makes pk publicly available such that a recipient can access it as needed. The user then signs a message m with pk and sends the message along with the signature to the recipient who can verify the origin and integrity of message using the verification algorithm $Vrfy(m, \sigma) \stackrel{?}{=} 1$ [KL14].

Remote storage integrity can be established in a similar way: a data owner generates a pair of public-private keys (pk, sk) and shares the public key with the storage server. The data owner then obtains a signature on the data and sends both the data and the signature to the storage server. The storage service provider can verify the integrity of the data using the sender's public key. When the data owner retrieves the data after some time, he or she can use the verification algorithm to verify the integrity of data. Digital signature schemes as defined above do not allow blockless verification as the entire data needs to be downloaded to verify its integrity. Moreover, the integrity guarantees of most digital signature schemes are computational though NIST has recently standardized three digital signature algorithms, namely *CRYSTALS-Dilithium, FALCON, and SPHINCS+*, that can withstand attacks by an adversary who has access to a quantum computer [NIS24].

3.2.2 Commitment Schemes

A commitment scheme can be viewed as a sealed envelope that contains a value that remains hidden until the envelope is opened, and the envelope can only be opened to the original value. So, the sealed envelope guarantees both confidentiality and integrity of the message inside [KL14].

Commitment Schemes [KL14, p. 188]. A (non-interactive) commitment scheme is defined by a randomized algorithm Gen that outputs public parameters $params$ and an algorithm Com that takes $params$ and a message $m \in \{0,1\}^n$ and outputs a commitment com, where the randomness used by Com is denoted by r. A sender commits to m by choosing uniform r, computing

$$com := Com(params, m; r),$$

and sending it to a receiver. The sender can later decommit com and reveal m by sending m, r to the receiver; the receiver verifies this by checking that

$$Com(params, m; r) \stackrel{?}{=} com.$$

Commitment schemes have two properties: hiding and binding. Hiding means that com reveals nothing about m; binding means that it is impossible to output a commitment com that can be opened in two different ways. These properties are formally defined below.

The commitment hiding experiment, $Hiding_{A,Com}(n)$:

1. Parameters $params \leftarrow Gen(1^n)$ are generated.
2. The adversary A is given input $params$, and outputs a pair of messages $m_0, m_1 \in \{0,1\}^n$.
3. A uniform $b \in \{0,1\}$ is chosen and $com \leftarrow Com(params, m_b; r)$ is computed.
4. The adversary A is given com and outputs a bit b'.
5. The output of the experiment is 1 if and only if $b' = b$.

The commitment binding experiment, $Binding_{A,Com}(n)$:

1. Parameters $params \leftarrow Gen(1^n)$ are generated.
2. A is given input $params$ and outputs (com, m, r, m', r').
3. The output of the experiment is defined to be 1 if and only if $m \neq m'$ and

$$Com(params, m; r) = com = Com(params, m'; r').$$

Security of Commitment Schemes [KL14, Def. 5.13, §5.6] A commitment scheme Com is secure if for all probabilistic polynomial time adversaries A, there is a negligible function $negl$ such that,

$$Pr[Hiding_{A,Com}(n) = 1] \leq \frac{1}{2} + negl(n)$$

$$Pr[Binding_{A,Com}(n) = 1] \leq negl(n).$$

Commit schemes can either be *computationally* hiding and *information-theoretically* binding or *information-theoretically* hiding and *computationally* binding, but not both [Dam98]. Because in order for a scheme to be information-theoretically binding, the commitments should map one-to-one to the message space. That means there is a unique commitment for each message. Otherwise, the sender can open the commitment to two different messages, rendering the information-theoretic binding property inert. On the other hand, a commitment scheme is information-theoretically hiding if a commitment can be mapped to more than one message from the point of view of the recipient regardless of his or her computational power. That means the commitment reveals no information about the message at all.

Now if a scheme is information-theoretically binding, then the commitment maps to a unique value, and thus, potentially reveals some information about the message. Similarly, if a scheme is information-theoretically hiding, then the commitment can possibly open to more than one message, but in that case, the binding property is violated. So, only one property can be information-theoretically secure at a time.

An information-theoretically binding interactive commitment scheme can be constructed from secure Pseudorandom Generators (PRGs) [BS23]. We describe one such commitment scheme that can commit a single-bit message next.

Pseudorandom Generators [KL14, Def. 3.14, §3.3] Let l be a polynomial and let G be a deterministic-polynomial time algorithm such that for any n and any input $s \in \{0,1\}^n$, the result $G(s)$ is a string of length $l(n)$. We say that G is a pseudorandom generator if the following conditions hold:

- (*Expansion.*) For every n, it holds that
$$l(n) > n.$$
- (*Pseudorandomness.*): For any probabilistic polynomial time (PPT) algorithm D, there is a negligible function $negl$ such that
$$|Pr[D(G(s)) = 1] - Pr[D(r) = 1]| \leq negl(n),$$
where the first probability is taken over uniform choice of $s \in \{0,1\}^n$ and the randomness of D, and the second probability is taken over uniform choice of $r \in \{0,1\}^{l(n)}$ and the randomness of D.

We call l the expansion factor of G.

Bit Commitments from secure PRGs [BS23, p. 89] Let $G : R \to P$ be a secure PRG, where $|P| \geq |R|^3$ and $P = \{0,1\}^n$ for some n. To commit to a bit $b_0 \in \{0,1\}$:

1. The recipient chooses a random $p \in P$ and sends p to the sender.
2. The sender chooses a random $r \in R$ and computes
$$com \leftarrow Com(p, b_0; r),$$
where $Com(p, b_0; r)$ is the following function:
$$Com(p, b_0; r) := \begin{cases} G(r) & \text{if } b_0 = 0, \\ G(r) \oplus p & \text{if } b_0 = 1. \end{cases}$$

The sender sends com as the commitment string. To open the commitment, the sender sends (b_0, r) to the recipient. The recipient accepts if $com = Com(p, b_0; r)$ and rejects otherwise. The binding property holds unconditionally as long as $\frac{1}{|R|}$ is negligible [BS23].

Some applications of commitment schemes are: coin flipping, zero-knowledge proofs, and verifiable secret sharing. We can also use commitment schemes as part of data integrity mechanisms. For the sake of illustration, we next discuss a naive approach of using an information-theoretically binding commitment scheme for data integrity. A data owner can use an interactive commitment scheme, where the recipient generates $params$, as follows: the storage server generates params-1 using the generation algorithm and shares it with the data owner. The data owner generates a commitment to the data to be stored using params-1 and obtains an additional parameter params-2 using Gen. The data owner then sends the data, the randomness used by Com, the commitment, and params-2 to the storage server. The storage server verifies the commitment to make sure the data owner is honest and to avoid repudiation. Next, the storage server generates a new commitment using params-2 and sends the commitment and the randomness used to the data owner. Upon receiving the new commitment, the data owner confirms that the storage server has indeed received and stored his data intact. The data owner locally stores the commitment and the random value from the storage server for later verification.

When the data owner later retrieves the data, he can generate a new commitment to the retrieved information using the stored random value and match it against the stored commitment from the storage server. The information-theoretic binding of the commitment scheme guarantees that the storage server cannot tamper with data without changing the value of the commitment he sent previously. Hence, the data owner can detect any tampering attempts. The protocol must be run twice to obtain commitments from both the data owner and the storage server as commitment schemes are designed to protect the recipient from the sender with respect to the binding property. We emphasize that this is a rather simple example of how a commitment scheme can be used for data storage. In reality, these schemes are never used on their own; instead, they are used in conjunction with other cryptographic primitives to protect data.

3.2.3 Secret Sharing Schemes

A secret sharing scheme splits a secret s into n shares such that only an authorized set of shares can reconstruct the secret but an unauthorized set does not reveal any information about the secret. Secret sharing schemes come in different flavors, and most of them do not rely on any mathematical unproven assumptions.

Sharing Function [GK18, Def. 2.4, §2.2]. Let $[n] = \{1, 2, \ldots, n\}$ be a set of identities of n parties and M be the domain of secrets. A sharing function $Share$ is a randomized mapping from M to $S_1 \times S_2 \times \cdots \times S_n$, where S_i is the domain of shares of party i. A dealer distributes a secret $m \in M$ by computing the vector

$$Share(m) = (s_1, s_2, \ldots, s_n)$$

and privately communicating each share s_i to party i. For a set $S \subseteq [n]$, we denote $Share(m)_S$ to be a restriction of $Share(m)$ to S.

Secret Sharing Scheme [GK18, Def. 2.5, §2.2]. Let M be a finite set of secrets, where $|M| \geq 2$. Let $[n] = \{1, 2, \ldots, n\}$ be a set of identities of n parties. A sharing function $Share$ with domain M is a (t, n, ϵ)-Secret Sharing Scheme if the following two properties hold:

- *(Correctness.)* The secret can be reconstructed from any t out of n shares. That is, for any set $T \subseteq [n]$ where $|T| \geq t$, there exists a deterministic reconstruction function $Rec : \otimes_{i \in T} S_i \to M$ such that for every $m \in M$,
$$Pr[Rec(Share(m)_T) = m] = 1$$
(over the randomness of the sharing function).
- *(Statistical Privacy.)* Any collusion of less than t parties have "almost" no information about the underlying secret. Formally, for any unauthorized set $U \subseteq [n]$, where $|U| < t$, and for every pair of secrets $m_1, m_2 \in M$, for any distinguisher D with output in $\{0, 1\}$, the following holds:
$$\left| Pr_{shares \leftarrow Share(m_1)}[D(shares_U) = 1] - Pr_{shares \leftarrow Share(m_2)}[D(shares_U) = 1] \right| \leq \epsilon.$$
The special case of $\epsilon = 0$ is known as perfect secrecy [GK18].

A secret sharing scheme is called linear if the share-generation and secret-reconstruction steps are linear transformations [Bei11]. Shamir's secret sharing scheme is a popular linear secret sharing scheme. Below we describe Shamir's secret sharing scheme, preceded by two algebraic structures that will be used in Shamir's.

Groups [DF04, pp. 16–17]. A *group* is an ordered pair $(G, \star)$, where G is a set and $\star$ is a binary operation on G, satisfying the following axioms:

1. the binary operation $\star$ is associative, meaning
$$a \star (b \star c) = (a \star b) \star c;$$
2. there exists an element e in G, called an *identity* of G, such that for all $a \in G$, we have
$$a \star e = e \star a = a; \text{ and}$$
3. for each $a \in G$, there exists $a^{-1} \in G$, called an *inverse* of a, such that
$$a \star a^{-1} = a^{-1} \star a = e.$$

If the operation $\star$ is commutative, then G is called an *abelian* group [DF04].

Fields [DF04, p. 34]. A *field* is a set F together with two binary operations $+$ and $\cdot$ on F such that $(F, +)$ is an abelian group (identity 0) and $(F - \{0\}, \cdot)$ is also an abelian group. Moreover, the following distributive law holds:

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

for all $a, b, c \in F$. If the set F is finite, then it is called a *finite field*.

Shamir's $(\mathbf{t}, \mathbf{n})$-threshold Secret Sharing Scheme [KL14, p. 503] Let F be a finite field that contains the domain of possible secrets, and with $|F| > n$. Let $x_1, x_2, \ldots, x_n \in F$ be distinct, non-zero elements that are fixed and publicly known (such elements exist since $|F| > n$). The scheme works as follows:

- *Sharing.* Given a secret $s \in F$, the dealer chooses uniform $a_1, a_2, \ldots, a_{t-1} \in F$ and defines the polynomial

  $$p(X) \stackrel{def}{=} s + \sum_{i=1}^{t-1} a_i X^i.$$

  This is a uniform degree-$(t-1)$ polynomial with constant term s. The share of party P_i is

  $$s_i := p(x_i) \in F.$$

- *Reconstruction.* Say t users $P_{i_1}, P_{i_2}, \ldots, P_{i_t}$ pool their shares $s_{i_1}, s_{i_2}, \ldots, s_{i_t}$. Using polynomial interpolation, they compute the unique degree-$(t-1)$ polynomial p' for which

  $$p'(x_{i_j}) = s_{i_j}$$

  for $1 \leq j \leq t$. The secret is $p'(0)$.

The reconstruction works since $p' = p$ and $p(0) = s$ [KL14].

Shamir's secret sharing scheme offers information-theoretic confidentiality and allows dynamic addition and removal of shares as long as the threshold is fixed [Sha79]. Secret sharing schemes are usually used in storage systems for data privacy and availability, but they can also provide data integrity. Since any threshold number of shares can reconstruct the secret, a data owner can take all possible combinations of t shares and run the reconstruction algorithm on each subset. If the resulting secrets are not identical, then at least one share has been altered either by the storage service provider or a malicious attacker. The data can be recovered as long as fewer than t shares are corrupted. But if t shares have been compromised, then the data is lost forever. This method of integrity verification is very similar to error correcting codes. Error correction techniques, however, can only detect and recover from random errors that affect a small portion of data [KL14]. The drawback of secret sharing schemes is that they do not allow blockless verification. Similar to the other techniques, secret sharing schemes need to be adapted for long-term integrity of remote storage systems.

3.3 Integrity: Provenance Tracking

Now that we have seen some techniques for data auditing, we review a potential solution for long-term provenance tracking. Data about data such as the origin of data and information about the ownership and authorized modifications can protect both data owner and storage server from unfair allegations. Metadata is usually static in nature and takes much less space compared to the data itself. Secure archival systems must collect metadata in a timely manner, establish its accuracy, and store it securely throughout the entire lifetime of data.

Blockchain Technology. A blockchain is a distributed and immutable ledger of transactions, where each transaction is verified by the majority of the participating nodes in the system. In a peer-to-peer network, nodes are the same as the peers and are considered equal, but they can take on different roles. There

are two types of nodes: full nodes, that maintain a complete copy of the blockchain, and miner nodes, that add new blocks to the chain. The blockchain framework can be divided into three layers: the data layer, network layer, and application layer. The data layer consists of data structures and algorithms, the network layer includes a decentralized network and a consensus mechanism used for distributed agreement among the nodes, and the application layer contains applications to facilitate various blockchain operations [GHY18].

The data structure used in a blockchain can be viewed as a linked list of blocks that are connected and secured through cryptographic hash functions. Blocks usually have a fixed size and comprise two parts: a header and a body. The block header contains metadata about the block such as the hash of the previous block header, the Merkle hash root of the included transactions, a timestamp, and a nonce. The body of the block contains all the transactions in the block, the number of which depends on the block size. Blockchain technology uses consensus mechanisms such as Proof of Work, Proof of Stake, and Practical Byzantine Fault Tolerance to reach a consensus among the full nodes regarding the validity of the transactions contained in a block [GHY18].

To illustrate how blocks are added to a chain, we briefly describe Bitcoin, the first cryptocurrency leveraging blockchain technology. Bitcoin uses a peer-to-peer network to record electronic transactions as follows: new transactions are broadcast to all nodes that collect them into a block. Each node then tries to solve a difficult proof-of-work, that is finding the correct nonce for its block. Once a solution is found by a node, it broadcasts the block to all the other nodes. Nodes accept the block only if all the transactions contained are valid and not already spent. The hash of the newly accepted block is then used as the hash of the previous block in the header of the next block to be added to the chain [Nak08].

Blockchain technology is widely used for security and provenance owing to its fault tolerance, transparency, and tamper resistance [PSR23]. Since the addition of a new block is subject to a majority nodes' approval, errors in a small number of nodes do not affect the ledger much. Moreover, each full node stores a complete copy of the ledger, so an attacker must tamper with all the copies to destroy a blockchain [GHY18]. In addition, changing a past block requires redoing the proof-of-work for that block and all the blocks after it while keeping up with the addition of newer blocks to the chain. As the number of blocks a computationally bounded attacker has to catch up with increases, the success probability of such attacks drops exponentially [Nak08]. Long-term secure archival systems can employ a blockchain to store metadata for provenance and non-repudiation purposes. For instance, authorized data modification requests can be treated as transactions and stored on a blockchain. Computationally secure cryptographic primitives, however, can render a blockchain insecure in the face of a computationally unbounded adversary. Another potential solution for provenance tracking is briefly mentioned in Chapter 5.

3.4 Discussion

The security approaches discussed in this chapter, unfortunately, do not encompass the necessary features of data integrity mechanisms that can protect data for a long time. All three data auditing techniques — digital signature schemes, commitment schemes, and secret sharing schemes— lack support for blockless verification. Blockchain technology is susceptible to majority attacks, where more than half of the nodes collude and introduce erroneous blocks to the chain [GHY18].

The auditing techniques also fail to cover the scope of assurance of data integrity solutions. Digital signatures and commitment schemes can only detect data tampering, but they cannot avoid data corruption or recover the original data. Secret sharing schemes, on the other hand, can avoid data corruption in the sense that a certain number of shares need to be altered in order to modify data. They can also recover data if a threshold number of shares are available. Detecting data tampering attempts using secret sharing is not straightforward as a data owner needs to reconstruct the polynomial for each subset of shares that has size t [HL09]. These approaches, therefore, cannot be used as standalone solutions for long-term data

integrity. To illustrate how they can be part of a complete solution, we now briefly describe some systems that employ one or more of these approaches though they do not consider them in the context of long-term information-theoretic integrity.

POTSHARDS is an archival storage system that provides long-term data confidentiality using secret sharing schemes [SGMV09]. Integrity verification in POTSHARDS is performed using hash functions and algebraic signatures as follows. In the preprocessing phase, data is split into secure *shards* using secret-splitting and sent to independent archives. When an archive receives a shard, it randomly selects a block to store the shard and places it in the last available slot in the block. A hash value, stored in the block header, is updated accordingly at this time. Archives periodically verify the integrity of its data by checking the hash stored in the header of each block on disk. To perform inter-archive verification, POTSHARDS uses a scheme based on algebraic signatures that have the property that the parity of the signatures is the same as the signatures of the parity. The scheme exhibits attractive features such as verification of random data blocks and high resistance to coordinated attempts to modify data.

LINCOS is a storage system that considers the long-term integrity, authenticity, and confidentiality of data though the integrity guarantee is computational [BBD+17]. It uses a novel integrity scheme based on information-theoretic hiding commitment schemes for confidentiality-preserving integrity and authenticity protection. The integrity scheme in LINCOS generates commitments to data being stored and timestamp the commitments. Since the commitment schemes are computationally binding, the commitments and timestamps are renewed from time to time. In order to verify the integrity of a document, the decommitment values are disclosed and the corresponding sequence of timestamps are used to establish the existence of the commitments. Assuming the commitment schemes and time stamp schemes used are secure during their usage period, the integrity scheme provides long-term unforgeability. Moreover, experimental evaluation shows that the scheme has good performance and negligible time and space overheads, compared to hash-based solutions. For long-term confidentiality, LINCOS relies on Shamir's secret sharing and requires the data owner to proactively renew the shares.

PASIS is a survivable storage system that aims to ensure everlasting security and availability [GKP05]. In the PASIS framework, the presence of compromised entities is viewed as the common case rather than the exception. Security is achieved through a decentralized storage architecture and secret sharing schemes, which allow users to trade off among performance, availability, and confidentiality. Some secret sharing schemes compatible with the PASIS framework are: Shamir's secret sharing, Blakley's threshold scheme, proactive secret sharing scheme, and verifiable secret sharing scheme. Verifiable secret sharing is a secret sharing scheme that uses a commitment scheme to protect honest share holders from a corrupted data dealer. A dealer or data owner uses a polynomial-based secret sharing scheme to generate shares and a commitment scheme to produce commitments to the coefficients of the polynomial. The shares are then sent to the respective parties and the commitment values are broadcast among all parties. Using the broadcast values, the share holders or storage servers can verify that their received shares are correct without communicating with other parties.

ProvChain is a blockchain-based provenance solution that offers security features such as tamper-proof provenance, user privacy, and reliability [LST+17]. It is a decentralized architecture that collects and verifies cloud data provenance. In ProvChain, provenance records are hashed and added to the leaf nodes of a Merkle tree, where the Merkle root is mapped to a transaction in the blockchain network. The transactions are then collected into a block and added to the chain after external verification [PSR23]. ProvChain protects against tampering attacks on provenance data since in order to modify a record, an attacker needs to locate the transaction and the block, create a new block with the modified record, and present a longer chain of blocks containing the new block. Additionally, ProvChain hashes the user ID to conceal user information from the network nodes and provenance auditors [LST+17].

Chapter 4

Related Work

In this chapter, we refer to some similar surveys on the subject matter. Storage security, especially the security of cloud storage services, has been investigated for decades [VBC+15, LYZC15, ZKM+17, ZFY+18, LCY+22]. These works are mostly focused on short-term storage security given that most data stored in the cloud is not classified information. Among the works cited above, only Vigil *et al.* survey the integrity, authenticity, non-repudiation, and proof of existence for long-term archives [VBC+15]. They compare existing solutions that offer the desired protection goals with respect to their functionalities, trust assumptions, and performance. The authors also shed light on the deficiencies of current systems in achieving these goals and draw attention to open problems. Our work is different from the survey in that we do not compare the existing solutions; instead, we focus on the underlying techniques that provide some of the integrity assurances in order to find the most suitable option. Moreover, we consider both computational and information-theoretic integrity of long-term archives, whereas the paper only discusses computationally secure solutions.

Liu *et al.* provide an analysis of authenticator-based data integrity verification techniques for cloud and Internet of Things data [LYZC15]. They partition the life cycle of a remote integrity verification scheme with dynamic data handling as follows: setup and data upload, authorization for trusted third parties, challenge for integrity proof, proof integration, proof verification, updated data upload, updated metadata upload, and verification of updated metadata. The paper focuses on the integrity verification techniques that support provable data possession (PDP), that is the verification is done only by checking a small number of blocks. While the PDP setting is efficient and practical for short-term storage, it cannot detect data integrity breaches in the long term. Besides, the representative schemes presented in the paper use computationally secure cryptographic primitives at best.

A survey by Zafar *et al.* presents common security concerns associated with cloud storage and discusses the significance of integrity schemes in the life cycle of outsourced data [ZKM+17]. The authors define a taxonomy based on the attributes of data integrity schemes mentioned in the paper and identify some design challenges. They also provide a comparative analysis of the existing solutions with respect to the identified characteristics and conclude with potential research threads. The paper is mostly concerned with the integrity of cloud storage but does not include long-term integrity solutions. The goal of our work is to survey techniques that can ensure the integrity of long-term storage systems, not only the integrity of short-term cloud storage services.

Zhou *et al.* review the state of the art data integrity verification schemes for cloud storage and classify existing approaches into four types according to user mode and storage type [ZFY+18]. In addition, they point out some major challenges that can compromise the safety and efficiency of data integrity verification schemes and discuss available techniques that can mitigate these issues. This paper mentions many desired characteristics and their associated challenges that pertain to long-term archival systems. However, the

security guarantee of all the schemes surveyed in the paper is only computational.

Li *et al.* list ten criteria for evaluating integrity auditing schemes for both single copy and multiple replicas of data in cloud storage [LCY+22]. The paper reviews a wide range of integrity auditing schemes based on these criteria and calls attention to some open research problems. Similar to most other surveys, this paper focuses on computationally secure integrity schemes for cloud storage. In contrast, we explore both computationally and information-theoretically secure techniques that can protect the integrity of long-term archival systems.

Chapter 5

Conclusion and Future Directions

As mentioned before, existing integrity solutions cannot provide protection against computationally unbounded adversaries. But the techniques explored in this report have the potential to ensure long-term integrity and can be combined to extend their scope of assurance. For example, the amalgamation of secret sharing schemes and digital signature schemes can avoid data corruption, detect tampering, and recover the original data. Goyal and Kumar recently proposed a non-malleable secret sharing scheme, where an adversary can only destroy the secret but cannot modify it without being detected [GK18]. If a corrupted share is used in reconstruction, the algorithm will return a string that is completely "unrelated" to the secret. Non-malleable secret sharing can be used in secure archival systems to easily uncover the presence of a corrupted share in a set of shares. For provenance tracking, blockchain-based solutions can replace computationally secure hash functions with commitment schemes, that have the unconditional-binding property. Maniatis and Baker introduced *Timeweave*, a framework for timeline entanglement based on two novel data structures [MB02]. A timeline is a historic record of the states that a system goes through during its operational history. Timeline entanglement merges different timelines maintained at independent and mutually distrusted systems by linking the past of one timeline to the future of another. Long-term storage systems can use timeline entanglement to create undeniable temporal orderings of provenance related events across distributed servers.

In this technical report, we consider techniques for long-term integrity of data at rest. The integrity of data in transit in the face of an unbounded adversary is still under investigation [KL14]. Besides, long-term confidentiality solutions often come with high storage and communication overheads [STD+24]. Access control, authenticity, dynamic data handling, and efficiency are some other challenges that an archival system must address in order to protect valuable data for generations to come.

Bibliography

[ABC+07] Giuseppe Ateniese, Randal C. Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary N. J. Peterson, and Dawn Xiaodong Song. Provable data possession at untrusted stores. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 598–609. ACM, 2007.

[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.

[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.

[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 - Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings*, volume 6639 of *Lecture Notes in Computer Science*, pages 11–46. Springer, 2011.

[BS23] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. 2023. https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_6.pdf.

[coc24] Coca-cola's formula is at the world of coca-cola, the secret's out, c2024. https://www.coca-colacompany.com/about-us/history/coca-cola-formula-is-at-the-world-of-coca-cola.

[Dam98] Ivan Damgård. Commitment schemes and zero-knowledge protocols. In Ivan Damgård, editor, *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*, volume 1561 of *Lecture Notes in Computer Science*, pages 63–86. Springer, 1998. https://doi.org/10.1007/3-540-48969-X_3.

[DF04] David S. Dummit and Richard M. Foote. *Abstract Algebra, Third Edition*. Willy and Sons, 2004.

[FLP85] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

[FLP+10] Daniel Ford, François Labelle, Florentina I. Popovici, Murray Stokely, Van-Anh Truong, Luiz Barroso, Carrie Grimes, and Sean Quinlan. Availability in globally distributed storage systems. In Remzi H. Arpaci-Dusseau and Brad Chen, editors, *9th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2010, October 4-6, 2010, Vancouver, BC, Canada, Proceedings*, pages 61–74. USENIX Association, 2010.

[GGF17] Paul A. Grassi, Michael E. Garcia, and James L. Fenton. Digital identity guidelines, 2017. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf.

[GHY18] Weichao Gao, William G. Hatcher, and Wei Yu. A survey of blockchain: Techniques, applications, and challenges. In *27th International Conference on Computer Communication and Networks, ICCCN 2018, Hangzhou, China, July 30 - August 2, 2018*, pages 1–11. IEEE, 2018. https://doi.org/10.1109/ICCCN.2018.8487348.

[GK18] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 685–698. ACM, 2018. https://doi.org/10.1145/3188745.3188872.

[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.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006*, pages 89–98. ACM, 2006.

[HL09] Lein Harn and Changlu Lin. Detection and identification of cheaters in (t , n) secret sharing scheme. *Des. Codes Cryptogr.*, 52(1):15–24, 2009.

[HSL21] Jennifer K. Herrmann, Yoonjoo Strumfels, and Kathy Ludwig. Examination of date tampering on abraham lincoln's pardon of patrick murphy, rg 153 entry 15, case mm761 (arc identifier: 1839980). *Forensic Science International: Synergy*, 3:100210, 2021. https://www.sciencedirect.com/science/article/pii/S2589871X21000802.

[HSW09] Ragib Hasan, Radu Sion, and Marianne Winslett. The case of the fake picasso: Preventing history forgery with secure provenance. In Margo I. Seltzer and Richard Wheeler, editors, *7th USENIX Conference on File and Storage Technologies, February 24-27, 2009, San Francisco, CA, USA. Proceedings*, pages 1–14. USENIX, 2009.

[JW00] Prasad Jogalekar and C. Murray Woodside. Evaluating the scalability of distributed systems. *IEEE Trans. Parallel Distributed Syst.*, 11(6):589–603, 2000.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014. https://www.crcpress.com/Introduction-to-Modern-Cryptography-Second-Edition/Katz-Lindell/p/book/9781466570269.

[LCY+22] Angtai Li, Yu Chen, Zheng Yan, Xiaokang Zhou, and Shohei Shimizu. A survey on integrity auditing for data storage in the cloud: From single copy to multiple replicas. *IEEE Trans. Big Data*, 8(5):1428–1442, 2022. https://doi.org/10.1109/TBDATA.2020.3029209.

[LJW+16] Zechao Liu, Zoe Lin Jiang, Xuan Wang, Siu-Ming Yiu, Chunkai Zhang, and Xiaomeng Zhao. Dynamic attribute-based access control in cloud storage systems. In *2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, August 23-26, 2016*, pages 129–137. IEEE, 2016.

[LSP19] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. In Dahlia Malkhi, editor, *Concurrency: the Works of Leslie Lamport*, pages 203–226. ACM, 2019. https://doi.org/10.1145/3335772.3335936.

[LST+17] Xueping Liang, Sachin Shetty, Deepak K. Tosh, Charles A. Kamhoua, Kevin A. Kwiat, and Laurent Njilla. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2017, Madrid, Spain, May 14-17, 2017*, pages 468–477. IEEE Computer Society / ACM, 2017.

[LYZC15] Chang Liu, Chi Yang, Xuyun Zhang, and Jinjun Chen. External integrity verification for outsourced big data in cloud and iot: A big picture. *Future Gener. Comput. Syst.*, 49:58–67, 2015.

[MB02] Petros Maniatis and Mary Baker. Secure history preservation through timeline entanglement. In Dan Boneh, editor, *Proceedings of the 11th USENIX Security Symposium, San Francisco, CA, USA, August 5-9, 2002*, pages 297–312. USENIX, 2002. http://www.usenix.org/publications/library/proceedings/sec02/maniatis.html.

[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.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf.

[nat11] National archives discovers date change on lincoln record, 2011. https://www.archives.gov/press/press-releases/2011/nr11-57.html#:~:text=Lowry%20admitted%20to%20changing%20the,Ford's%20Theatre%20in%20Washington%2C%20DC.

[NIS01] NIST. Advanced encryption standard (aes), 2001. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf.

[NIS24] NIST. Post-quantum cryptography: Additional digital signature schemes, 2024. https://csrc.nist.gov/projects/pqc-dig-sig.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[PJG+24] Paul R. Poulton, A. Edward Johnston, Margaret J. Glendining, Rodger P. White, Andrew S. Gregory, Suzanne J. Clark, Wendy S. Wilmer, Andy J. Macdonald, and David S. Powlson. Chapter four - the broadbalk wheat experiment, rothamsted, uk: Crop yields and soil changes during the last 50 years, 2024. https://www.sciencedirect.com/science/article/pii/S0065211323001165.

[PSR23] Bofeng Pan, Natalia Stakhanova, and Suprio Ray. Data provenance in security and privacy. *ACM Comput. Surv.*, 55(14s):323:1–323:35, 2023.

[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, 1979. https://doi.org/10.1145/359168.359176.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[SM06] Thomas J. E. Schwarz and Ethan L. Miller. Store, forget, and check: Using algebraic signatures to check remotely administered storage. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS 2006), 4-7 July 2006, Lisboa, Portugal*, page 12. IEEE Computer Society, 2006.

[ST21] Nigel P. Smart and Emmanuel Thomé. History of cryptographic key sizes. *IACR Cryptol. ePrint Arch.*, page 894, 2021.

[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 2024, Santa Clara, CA, USA, July 8-9, 2024*. ACM, 2024.

[SW08] Hovav Shacham and Brent Waters. Compact proofs of retrievability. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, volume 5350 of *Lecture Notes in Computer Science*, pages 90–107. Springer, 2008.

[SWZ05] Gopalan Sivathanu, Charles P. Wright, and Erez Zadok. Ensuring data integrity in storage: techniques and applications. In *Proceedings of the 2005 ACM Workshop on Storage Security and Survivability*, StorageSS '05, page 26–36, New York, NY, USA, 2005. Association for Computing Machinery. https://doi.org/10.1145/1103780.1103784.

[TLLP12] Yang Tang, Patrick P. C. Lee, John C. S. Lui, and Radia Perlman. Secure overlay cloud storage with access control and assured deletion. *IEEE Trans. Dependable Secur. Comput.*, 9(6):903–916, November 2012.

[VBC+15] Martín A. Gagliotti 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 and Security*, 50:16–32, 2015. https://doi.org/10.1016/j.cose.2014.12.004.

[wik24] Sha-1, Oct 2024. https://en.wikipedia.org/wiki/SHA-1.

[WWRL10] Cong Wang, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-preserving public auditing for data storage security in cloud computing. In *INFOCOM 2010. 29th IEEE International*

Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA, pages 525–533. IEEE, 2010.

[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.

[ZFY+18] Lei Zhou, Anmin Fu, Shui Yu, Mang Su, and Boyu Kuang. Data integrity verification of the outsourced big data in the cloud environment: A survey. *Journal of Network and Computer Applications*, 122:1–15, 2018. https://doi.org/10.1016/j.jnca.2018.08.003.

[ZKM+17] Faheem Zafar, Abid Khan, Saif Ur Rehman Malik, Mansoor Ahmed, Adeel Anjum, Majid Iqbal Khan, Nadeem Javed, Masoom Alam, and Fuzel Jamil. A survey of cloud computing data integrity schemes: Design challenges, taxonomy and future trends. *Computers and Security*, 65:29–49, 2017. https://doi.org/10.1016/j.cose.2016.10.006.

[ZWGG23] Yifang Zhang, Mingyue Wang, Yu Guo, and Fangda Guo. Towards dynamic and reliable private key management for hierarchical access structure in decentralized storage. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, page 3371–3380, New York, NY, USA, 2023. Association for Computing Machinery.