Control and Cybernetics

vol. **42** (2012) No. 2

Homomorphic linear authentication schemes from ϵ-ASU$_2$ functions for proofs of retrievability [1][2]

by

Shengli Liu[1,2], **Kefei Chen**[1]

[1]Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai 200240, China
[2]State Key Laboratory of Information Security,
Institute of Software, Chinese Academy of Sciences

Abstract: Proof of Retrievability (POR) refers to interactive auditing protocols executed between a storage server and clients, so that clients can be convinced that their data is available at the storage server, ready to be retrieved when needed. In an interactive POR protocol, clients initiate challenges to the server, and the server feedbacks responses to clients with the help of the stored data. Retrievability means that it should be possible for a client to extract his/her data from the server's valid responses. An essential step leading to retrievability is the server's unforgeability of valid responses, i.e, any server coming up with valid responses to a client's challenges is actually storing the client's data with overwhelming probability. Unforgeability can be achieved with authentication schemes like MAC, Digital Signature, etc. With Homomorphic Linear Authentication (HLA) schemes, the server's several responses can be aggregated into one, hence reducing the communication complexity. In this paper, we explore some new constructions of ϵ-almost strong universal hashing functions (ϵ-ASU$_2$), which are used to build homomorphic linear authenticator schemes in POR to provide unforgeability. We show the HLA scheme involved in Shacham and Waters' POR scheme (see Shacham and Waters, 2008) is just an employment of a class ϵ-ASU$_2$ functions. Using another class of ϵ-ASU$_2$ for authentication in POR results in a new HLA scheme, which enjoys unforgability, the same shortest responses as the SW scheme, but reduces the local storage from $O(n+s)$ to $O(n)$ for information soundness, and from $O(s)$ to $O(1)$ for knowledge-soundness.

Keywords: authentication code, proof of retrievability

[1]Submitted: November 2011; Accepted: April 2012

[2]Funded by NSFC (Nos. 61170229, 60970111, 61133014), Innovation Project (No. 12ZZ021) of Shanghai Municipal Education Commission, and Specialized Research Fund (No. 20110073110016) for the Doctoral Program of Higher Education.

1. Introduction

Nowadays "cloud storage" is able to provide storage services at both personal and business level. However, outsourced storage puts clients' data totally in the control of the storage center. Hence, storing data in a remote and unreliable server is of full risk and clients may worry about the availability of their data at the storage center. Hence it is of great importance for the storage center to be able to convince its clients that their data is right there, ready to be retrieved when needed.

The ability of a storage system to generate proofs of possession of client's files, without having to retrieve the whole file, is the so-called "proofs of data possession"(PDP). If the client's data can be extracted from proofs of data procession, then the PDP scheme is developed to be a "Proof of Retrievability" (POR) scheme. A POR scheme involves an interactive POR protocol between a storage server (prover) and a client (verifier). The client issues a series of queries and the storage provider feedbacks the corresponding responses. The client checks whether the responses are valid or not. Proof of retrievability requires that the server who passes verification checks is actually storing clients' data, and there exists an extraction algorithm which can extract the stored data via valid responses from the server.

The POR schemes fall into two categories, one with public verification and the other with private verification. Public verification means that everyone can perform the role of verifier in the POR protocol, while private verification means that only the client who owns the data is able to play the role of verifier.

1.1. Related work

There are many studies devoted to POR, among which Naor and Rothblum (2005) and Juels and Kaliski (2007) are the first to consider the security model. Dodis, Vadhan and Wichs (2009) identified POR problems with different variants, like bounded-use vs. unbounded-use, knowledge-soundness vs. information-soundness. They abstracted a notion of POR codes and applied POR codes in various constructions. Most of the constructions of POR, like the constructions of Ateniese et al. (2007), Shacham and Waters (2008), or Schwarz and Miller (2006), are combinations of erasure codes and an authentication scheme. When authentication is achieved with a digital signature scheme, one gets a POR scheme with public verification.

To decrease both the computational and communication complexity of the interacitve POR protocol, Ateniese et al. (2007) proposed to use homomorphic verifiable tags and constructed a RSA-based POR scheme with public verification. Because of the homomorphic property, tags computed for multiple file blocks can be combined into a single value.

The first POR schemes with full proofs of security against arbitrary adversaries were given by Shacham and Waters (2008). The POR schemes proposed

by Shacham and Waters (2008) also used homomorphic tags, one with private verification and the other with public verification.

The POR schemes with homomorphic tags, proposed by Ateniese et al. (2007) and Shacham and Waters (2008), result in short responses among available POR schemes. These schemes were identified as homomorphic linear authenticator schemes (HLA schemes for short) by Dodis, Vadhan and Wichs (2009). The idea is first to use an erasure code to expand the original file M' to an encoded version M, such that the retrieval of ρ fraction of M is able to extract the whole file M. Then the authentication scheme is applied to the blocks $\{M_1, M_2, \cdots, M_n\}$ of the coded file M to get tags $\vec{\sigma} = (\sigma_1, \sigma_2, \cdots, \sigma_n)$ with an authentication key K. The final file M^*, which is M attached with the tags $\vec{\sigma}$, is stored in a storage server. The client can issue some indices of the blocks of M, and the server responds with the corresponding block-tag pairs. The client then verifies the validity of those block-tag pairs with K, and accepts the blocks if those pairs pass verification. The valid blocks can be fed to an extraction algorithm, which is a decoder of erasure codes, to extract the whole file M. Let $(v_1, v_2, \cdots, v_n)$ be the challenge vector, the prover (server) can generate an aggregated pair of message and tag i.e., $(\mu = \sum_{i=1}^{n} v_i M_i, \sigma = \sum_{i=1}^{n} v_i \sigma_i)$. Given the prover's response (μ', σ'), the verifier (client) is able to use the authentication key K to check whether $\mu' = \sum_{i=1}^{n} v_i M_i$ with overwhelming probability. Homomorphic linear authenticator (HLA) schemes enable the prover (server) to provide correct response with only one block-tag pair (μ, σ). In fact, the two POR schemes proposed by Shacham and Waters (2008) are just homomorphic linear authenticator schemes and enjoy the shortest responses from the server among all the available POR schemes.

In this paper, we study POR with private verification. We will bench Shacham and Waters' POR scheme with private verification, named the SW scheme hereafter. The SW scheme based on an erasure code, a Pseudo-Random Function (PRF) and a homomorphic linear authenticator (HLA) scheme to produce homomorphic tags.

- An erasure code is used to encode some redundancy to the original file M' to get M, so that the data owner is able to extract the whole file M from some verified blocks obtained from interactions in the POR protocol.
- A Pseudo-Random Function (PRF) uses a seed k_{prf} to generate keys: $f_{k_{\text{prf}}}(i)$, $i = 1, 2, \cdots$.
- A homomorphic linear authenticator (HLA) scheme is employed to create homomorphic tags. The encoded file M is divided into n blocks of same sizes, $M_1, M_2, \cdots, M_n$ and each block M_i is further divided into s sectors of same sizes, $(m_{i1}, m_{i2}, \cdots, m_{is}) \in GF(q)^s$. A systematic authentication code is applied to M, block by block, to obtain tags $(\sigma_1, \sigma_2, \cdots, \sigma_n)$ with $(\alpha_1, \alpha_2, \cdots, \alpha_s, f_{k_{\text{prf}}}(1), f_{k_{\text{prf}}}(2), \cdots, f_{k_{\text{prf}}}(n))$ being the authentica-

tion keys,

$$\sigma_i \leftarrow f_{k_{\text{prf}}}(i) + \sum_{j=1}^{s} \alpha_j m_{ij}, \quad i = 1, 2, \cdots, n,$$

and all the operations are over $GF(q)$.

- The authentication key for block i is $(\alpha_1, \alpha_2, \cdots, \alpha_s, f_{k_{\text{prf}}}(i))$. Hence all the blocks share the same key $(\alpha_1, \alpha_2, \cdots, \alpha_s)$, but each one has its individual key $f_{k_{\text{prf}}}(i)$ for authentication.
- Each tag σ_i is attached to its block M_i for storage. The tags are homomorphic verifiable with respect to the blocks. For queries $(v_1, v_2, \cdots, v_n)$, the storage server is able to aggregate the sectors of the blocks with a linear combination $\mu_j = \sum_{i=1}^{n} v_i m_{ij}, j = 1, 2, \cdots, s$, and feedbacks the responding block $(\mu_1, \mu_2, \cdots, \mu_s)$, together with an aggregated tag computed by $\sigma = \sum_{i=1}^{n} v_i \sigma_i$.
- Thanks to the homomorphic properties of the tags, the file owner is able to verify the validity of the aggregated block and tag with his/her own secret key by testing whether the following holds or not

$$\sigma = \sum_{i=1}^{n} v_i f_{k_{\text{prf}}}(i) + \sum_{j=1}^{s} \alpha_j \mu_j.$$

In the SW scheme, the client will store $(s+1)\lceil \log_2 q \rceil$ bits locally, and the storage server will store $(n+1)(s+1)\lceil \log_2 q \rceil$ bits. The use of Pseudo-Random Function makes the POR scheme only computationally secure, hence "knowledge soundness" according to Dodis, Vadhan and Wichs (2009).

1.2. Authentication codes and ϵ-ASU_2

The goal of authentication is to investigate a coding method such that the receiver will detect the opponent's active attack. Authentication codes (A-codes) work as follows: the transmitter and receiver first agree on an authentication key k, taken from a finite set $\mathcal{K}$. Each authentication key k determines an encoding rule $E_k(\cdot)$, which encodes a piece of plaintext m, to a message $c = E_k(m)$. If $c = E_k(m) = (m, \sigma)$ with $\sigma = e_k(m)$, we call σ the tag or the authenticator, and these codes are called system authentication-codes.

The active attacks by the opponent can be classified into two categories according to his cheating strategies. The first is call impersonation attack in which the opponent introduces a fraudulent message to the channel, hoping it to be accepted by the receiver. The other is called substitution attack, in which the opponent intercepts a message-tag pair (m, σ) and modifies it to a different one (m', σ'), hoping it to be accepted by the receiver. Let P_I and P_S be

the probabilities of a successful impersonation attack, respectively substitution attack.

Stinson (1992) considered how to construct authentication codes from ϵ-*almost strongly universal*$_2$ (see Section 2 for the definition). Given an ϵ-*ASU*$_2$: $\mathcal{H} : \mathcal{A} \to \mathcal{B}$, let $\mathcal{A}$ be the set of messages to be authenticated, $\mathcal{B}$ be the set of tags, and the index of hash functions $H \in \mathcal{H}$ uniquely determined by the authentication key. Then an ϵ-ASU$_2$ determines a systematic authentication code with $P_I = 1/|\mathcal{B}|$ and $P_S \leq \epsilon$ (see Stinson, 1992).

The essential part of a POR system is the design of a homomorphic linear authentication (HLA) scheme with short homomorphic tags and negligible probability of successful active attacks. Short tags results in small storage expansion. Homomorphic tags make the server's response short by aggregating multiple tags to a single one, reducing the communication complexity of POR. The resistance to active attacks ensures that it is impossible for a cheating storage server to forge any valid aggregated tag without the knowledge of the file and its tags. We will show in this paper that some ϵ-ASU$_2$ imply HLA scheme for POR.

1.3. Our contribution

In this paper, we make two contributions.

1. We give two new constructions for ϵ-Almost Strong Universal (ϵ-ASU$_2$) family of hash functions, one is a $1/q$-ASU$_2$ family and the other an s/q-ASU$_2$, each ϵ-ASU$_2$ implying an HLA scheme with information-theoretic security. The HLA scheme from $1/q$-ASU$_2$ requires that clients store an authentication key of length $(n + s)\lceil \log_2 q \rceil$, and the HLA scheme from s/q-ASU$_2$ needs an authentication key of length $(n + 1)\lceil \log_2 q \rceil$, where n is the number of blocks in a file and s the number of sectors in each block.
2. We show that the SW scheme by Shacham and Waters (2008) is just a homomorphic linear authenticator scheme instantiated with the $1/q$-ASU$_2$ family, with a Pseudo-Random Function (PRF) generating authentication keys for each block. The use of a PRF reduces the local storage of clients to $(1+s)\lceil \log_2 q \rceil$ with price of computational security. We substitute the s/q-ASU$_2$ family for the $1/q$-ASU$_2$ used in the SW scheme to obtain another HLA scheme with computational security. The new scheme enjoys the shortest responses and the same security just as the SW scheme. However, the HLA scheme from s/q-ASU$_2$ needs a shorter authentication key, hence the local storage of clients, which is mainly used to store the authentication key, is reduced to $2\lceil \log_2 q \rceil$, compared to $(1+s)\lceil \log_2 q \rceil$ of the SW scheme, with s the number of sectors in each block.

The rest of paper is organized as follows. In Section 2, we will investigate ϵ-Almost Strong Universal (ϵ-ASU$_2$) hash functions, and give some constructions. In Section 3, we apply the ϵ-ASU$_2$ obtained in Section 2 to HLA schemes and give analysis. Section 4 concludes the paper.

2. ϵ-Almost strong universal hash functions and constructions

Universal classes of hash functions were first introduced by Carter and Wegman (1979) for storage and retrieval on keys. Later, Wegman and Carter (1981) studied the applications of those hash functions in authentication. Stinson (1994) extended universal hash functions to *ϵ-almost strongly universal$_2$* (ϵ-ASU$_2$) hash functions and showed how to construct authentication codes from ϵ-ASU$_2$.

Let $\mathcal{A}$ and $\mathcal{B}$ be finite sets. For a hash function $h : \mathcal{A} \to \mathcal{B}$, for $a_1, a_2 \in \mathcal{A}, a_1 \neq a_2$, define

$$\delta_h(a_1, a_2) = \begin{cases} 1, & \text{if } h(a_1) = h(a_2), \\ 0, & \text{otherwise.} \end{cases}$$

For a finite set $\mathcal{H}$ of hash functions, all from $\mathcal{A}$ to $\mathcal{B}$, define

$$\delta_{\mathcal{H}}(a_1, a_2) = \sum_{h \in \mathcal{H}} \delta_h(a_1, a_2).$$

Then, $\delta_{\mathcal{H}}(a_1, a_2)$ counts the number of hash functions, for which a_1 and a_2 collide.

DEFINITION 1 *Let $\epsilon > 0$. $\mathcal{H} : \mathcal{A} \to \mathcal{B}$ is ϵ-almost strongly-universal$_2$ (or ϵ-ASU$_2$) if*

(a) *for every $a \in \mathcal{A}$ and for every $b \in \mathcal{B}$, $|\{h \in \mathcal{H} : h(a) = b\}| = |\mathcal{H}|/|\mathcal{B}|$;*

(b) *for every $a_1, a_2 \in \mathcal{A}$ $(a_1 \neq a_2)$ and for every $b_1, b_2 \in \mathcal{B}$, $|\{h \in \mathcal{H} : h(a_1) = b_1,\ h(a_2) = b_2\}| \leq \epsilon|\mathcal{H}|/|\mathcal{B}|$.*

DEFINITION 2 *If $\mathcal{H}$ is ϵ-ASU$_2$ with $\epsilon = 1/|\mathcal{B}|$, then $\mathcal{H}$ is called strongly universal$_2$ (or strongly universal, SU$_2$ for short).*

Here we show some well-known constructions for ϵ-ASU$_2$.

CONSTRUCTION 1 (Stinson, 1994) *For some positive integer s, let $\mathcal{A} = \{\vec{\boldsymbol{a}} = (a_1, a_2, \ldots, a_s) \in GF(q)^s\}$, $\mathcal{B} = GF(q)$. Let $\mathcal{H} = \{(h_1, h_2, \ldots, h_{s+1}) \in GF(q)^{s+1}\}$. Then $\mathcal{H} : \mathcal{A} \to \mathcal{B}$ defines an SU$_2$ with*

$$H(\vec{\boldsymbol{a}}) = h_1 \cdot a_1 + h_2 \cdot a_2 + \cdots + h_s \cdot a_s + h_{s+1},$$

where $(h_1, h_2, \ldots, h_{s+1}) \in \mathcal{H}$, $(a_1, a_2, \ldots, a_s) \in \mathcal{A}$, $H(\vec{\boldsymbol{a}}) \in \mathcal{B}$, and all operations are over $GF(q)$.

CONSTRUCTION 2 (den Boer, 1993) *Let $\mathcal{A} = \{\vec{\boldsymbol{a}} = (a_1, a_2, \ldots, a_s) \in GF(q)^s\}$. For any $\vec{\boldsymbol{a}} = (a_1, a_2, \ldots, a_s) \in \mathcal{A}$, define a polynomial $\vec{\boldsymbol{a}}(x) = a_1 x + a_2 x^2 + \cdots + a_s x^s$. It is a mapping from $GF(q)$ to itself. Let $\mathcal{H} = \{(h_1, h_2) \in GF(q)^2\}$. Then $\mathcal{H} : \mathcal{A} \to \mathcal{B}$ defines an ϵ-ASU$_2$ with $\epsilon = s/q$, and*

$$H(\vec{\boldsymbol{a}}) = h_1 + \vec{\boldsymbol{a}}(h_2) = h_1 + a_1 \cdot h_2 + a_2 \cdot (h_2)^2 + \cdots + a_s \cdot (h_2)^s,$$

where $(h_1, h_2) \in \mathcal{H}$ and $\vec{\boldsymbol{a}} \in \mathcal{A}$.

Now we extend the inputs of SU$_2$ in Construction 1 and s/q-ASU$_2$ in Construction 2 to matrices, and obtain new constructions of ϵ-ASU_2.

CONSTRUCTION 3 *For some positive integers n, s, let*

$$\mathcal{A} = \left\{ A_{n\times s} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1s} \\ a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{ns} \end{pmatrix} \in GF(q)^{n\times s} \right\},$$

$\mathcal{B} = \{\vec{\boldsymbol{b}} = (b_1, b_2, \cdots, b_n)^T \in GF(q)^n\}$. *Let* $\mathcal{H} = \{(\vec{\boldsymbol{h}}, \vec{\boldsymbol{h'}}) \mid \vec{\boldsymbol{h}} = (h_1, h_2, \ldots, h_s)^T \in GF(q)^s, \vec{\boldsymbol{h'}} = (h'_1, h'_2, \cdots, h'_n)^T \in GF(q)^n\}$. *Then $\mathcal{H} : \mathcal{A} \to \mathcal{B}$ defines a family of functions with*

$$H(A_{n\times s}) = A_{n\times s} \cdot \vec{\boldsymbol{h}} + \vec{\boldsymbol{h'}}.$$

LEMMA 1 *The family of the hash functions $\mathcal{H} : \mathcal{A} \to \mathcal{B}$ in Construction 3 is a $\frac{1}{q}$-ASU_2.*

Proof. . Given $A_{n\times s}$ and the hash value $\vec{\boldsymbol{b}} = (b_1, b_2, \cdots, b_n)^T$, for each $\vec{\boldsymbol{h}} = (h_1, h_2, \cdots, h_s)^T$, there exists a unique $\vec{\boldsymbol{h'}} = (h'_1, h'_2, \cdots, h'_n)^T$ such that

$$\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = A_{n\times s} \cdot \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_s \end{pmatrix} + \begin{pmatrix} h'_1 \\ h'_2 \\ \vdots \\ h'_n \end{pmatrix}. \tag{1}$$

There are in total q^s choices for $\vec{\boldsymbol{h}}$, hence there are totally q^s hash functions mapping $A_{n\times s}$ to $\vec{\boldsymbol{b}}$, and $q^s = |\mathcal{H}|/|\mathcal{B}| = q^{n+s}/q^n$.

Given $A_{n\times s}, \widetilde{A}_{n\times s}$, with $A_{n\times s} \neq \widetilde{A}_{n\times s}$, $\vec{\boldsymbol{b}} = (b_1, b_2, \cdots, b_n)^T$, and $\widetilde{\vec{\boldsymbol{b}}} = (\widetilde{b_1}, \widetilde{b_2}, \cdots, \widetilde{b_n})^T$, we count how many hash functions in $\mathcal{H}$ satisfy both (1) and (2).

$$\begin{pmatrix} \widetilde{b_1} \\ \widetilde{b_2} \\ \vdots \\ \widetilde{b_n} \end{pmatrix} = \widetilde{A}_{n\times s} \cdot \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_s \end{pmatrix} + \begin{pmatrix} h'_1 \\ h'_2 \\ \vdots \\ h'_n \end{pmatrix}. \tag{2}$$

Define $\Delta\vec{\boldsymbol{b}} = \vec{\boldsymbol{b}} - \widetilde{\vec{\boldsymbol{b}}}$, and $(\Delta A)_{n\times s} = A_{n\times s} - \widetilde{A}_{n\times s}$. Subtracting (2) from (1) gives

$$\Delta\vec{\boldsymbol{b}} = (\Delta A)_{n\times s} \cdot \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_s \end{pmatrix}. \tag{3}$$

We only have to determine the number of hash functions in $\mathcal{H}$ satisfying both (1) and (3).

Let r be the rank of matrix $(\Delta A)_{n\times s}$, then the null space of $(\Delta A)_{n\times s}$ has dimension $s-r$. $(\Delta A)_{n\times s} \neq 0$ implies that $r \geq 1$.

Among all the q^s free choices of $(h_1, h_2, \cdots, h_s)$ (each choice determines a unique $(h'_1, h'_2, \cdots, h'_n)$) satisfying (1), only q^{s-r} of them also satisfy (3). Hence the total number of hash functions in $\mathcal{H}$ for both (1) and (3) is q^{s-r}, which is upper-bounded by

$$q^{s-1} = \frac{1}{q} \cdot \frac{|\mathcal{H}|}{|\mathcal{B}|}.$$

Consequently, Construction 3 results in a $\frac{1}{q}$-ASU_2 family. ∎

Construction 4 *For some positive integers n, s, let*

$$\mathcal{A} = \left\{ A_{n\times s} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1s} \\ a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots & a_{ns} \end{pmatrix} \in GF(q)^{n\times s} \right\},$$

$\mathcal{B} = \{\vec{\boldsymbol{b}} = (b_1, b_2, \cdots, b_n)^T \in GF(q)^n\}$. *Let* $\mathcal{H} = \{(h, \vec{\boldsymbol{h'}}) \mid h \in GF(q), \vec{\boldsymbol{h'}} = (h'_1, h'_2, \cdots, h'_n)^T \in GF(q)^n\}$. *Then $\mathcal{H} : \mathcal{A} \to \mathcal{B}$ defines a family of functions with*

$$H(A_{n\times s}) = A_{n\times s} \cdot \begin{pmatrix} h \\ h^2 \\ \vdots \\ h^s \end{pmatrix} + \begin{pmatrix} h'_1 \\ h'_2 \\ \vdots \\ h'_n \end{pmatrix}.$$

Lemma 2 *The family of the hash functions $\mathcal{H} : \mathcal{A} \to \mathcal{B}$ in Construction 4 is an $\frac{s}{q}$-ASU$_2$.*

Proof. . Given $A_{n\times s}$ and the hash value $\vec{\boldsymbol{b}} = (b_1, b_2, \cdots, b_n)^T$, for each $h \in GF(q)$, there exists a unique $\vec{\boldsymbol{h'}} = (h'_1, h'_2, \cdots, h'_n)^T$ such that

$$\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = A_{n\times s} \cdot \begin{pmatrix} h \\ h^2 \\ \vdots \\ h^s \end{pmatrix} + \begin{pmatrix} h'_1 \\ h'_2 \\ \vdots \\ h'_n \end{pmatrix}. \tag{4}$$

Hence there are totally q hash functions mapping $A_{n\times s}$ to $\vec{\boldsymbol{b}}$, and $q = |\mathcal{H}|/|\mathcal{B}| = q^{n+1}/q^n$.

Given $A_{n\times s}, \widetilde{A}_{n\times s}$, with $A_{n\times s} \neq \widetilde{A}_{n\times s}$, $\vec{\boldsymbol{b}} = (b_1, b_2, \cdots, b_n)^T$, and $\widetilde{\vec{\boldsymbol{b}}} = (\widetilde{b_1}, \widetilde{b_2}, \cdots, \widetilde{b_n})^T$, we count the number of hash functions in $\mathcal{H}$ satisfying both (4) and (5).

$$\begin{pmatrix} \widetilde{b_1} \\ \widetilde{b_2} \\ \vdots \\ \widetilde{b_n} \end{pmatrix} = \widetilde{A}_{n\times s} \cdot \begin{pmatrix} h \\ h^2 \\ \vdots \\ h^s \end{pmatrix} + \begin{pmatrix} h'_1 \\ h'_2 \\ \vdots \\ h'_n \end{pmatrix}. \tag{5}$$

Define $\Delta\vec{\boldsymbol{b}} = \vec{\boldsymbol{b}} - \widetilde{\vec{\boldsymbol{b}}}$, and $(\Delta A)_{n\times s} = A_{n\times s} - \widetilde{A}_{n\times s}$. Subtraction of (5) from (4) gives

$$\Delta\vec{\boldsymbol{b}} = (\Delta A)_{n\times s} \cdot \begin{pmatrix} h \\ h^2 \\ \vdots \\ h^s \end{pmatrix}. \tag{6}$$

Let $(\Delta A)_{n\times s} = (a_{ij})_{n\times s}$. (6) is equivalent to

$$b_i - \widetilde{b_i} = \sum_{j=1}^{s} a_{ij} h^j, \quad i = 1, 2, \cdots, n. \tag{7}$$

Since $(\Delta A)_{n\times s} \neq 0$, there exists at least one non-zero row, say $\vec{\boldsymbol{a}}_{\boldsymbol{k}}{}^T = (a_{k1}, a_{k2}, \cdots, a_{ks})$, $1 \leq k \leq n$, in $(\Delta A)_{n\times s}$, resulting in a nonzero $(b_k - \widetilde{b_k})$, i.e., $b_k - \widetilde{b_k} = \sum_{j=1}^{s} a_{kj} h^j$. Equation

$$(b_k - \widetilde{b_k}) = a_{k1}x + a_{k2}x^2 + \cdots + a_{ks}x^s \tag{8}$$

has at most s roots in $GF(q)$.

Consequently, among the q free choices of h (which uniquely determine the value of $(h'_1, h'_2, \cdots, h'_n)$), satisfying (4), at most s of them also satisfy (7). Hence, the total number of hash functions in $\mathcal{H}$ for both (4) and (7) is at most s.

Since

$$s = \frac{s}{q} \cdot q = \frac{s}{q} \cdot \frac{|\mathcal{H}|}{|\mathcal{B}|},$$

Construction 4 is an $\frac{s}{q}$-ASU$_2$ family. ∎

3. Proof of retrievability scheme

3.1. Homomorphic linear authentication schemes

We recall the definition of Homomorphic Linear Authentication Schemes, which is used to achieve POR, and the corresponding security model in the work of Dodis, Vadhan and Wichs (2009).

A homomorphic linear authentication scheme consists of four randomized algorithms as follows.

$\mathsf{Kg}(1^\lambda)$. The key generation algorithm takes as input the security parameter λ, and generates an authentication key sk.

$\vec{\boldsymbol{\sigma}} \leftarrow \mathbf{LinTag}(sk, M)$. The algorithm takes as input the file M to be stored and the secret key sk, divides the M into n blocks $(M_1, M_2, \cdots, M_n)$, then computes a tag σ_i for each block M_i with the authentication key sk. It outputs a vector-tag $\vec{\boldsymbol{\sigma}} = (\sigma_1, \cdots, \sigma_n)$.

$(\mu, \sigma) \leftarrow \mathbf{LinAuth}(M, \vec{\boldsymbol{\sigma}}, \vec{\boldsymbol{v}})$. The algorithm takes as input the file M and the vector-tag $\vec{\boldsymbol{\sigma}}$, which is the output of algorithm LinTag, and a challenge vector $\vec{\boldsymbol{v}} = (v_1, v_2, \cdots, v_n)$. It computes $\mu = \sum_{i=1}^{n} v_i M_i$ and a tag σ, then outputs (μ, σ).

$b \leftarrow \mathbf{LinVer}(sk, \vec{\boldsymbol{v}}, (\mu', \sigma'))$. The LinVer algorithm takes as input the authentication key sk, a challenger vector $\vec{\boldsymbol{v}}$, a block-tag pair (μ', σ'). It computes a bit $b \in \{0, 1\}$, indicating whether it accepts or rejects.

The correctness of the HLA scheme requires that for all secret key sk outputted by Kg, for all file $M \in \{0,1\}^*$, for all $\vec{\boldsymbol{\sigma}}$ output by $\mathsf{LinTag}(sk, M)$, and all (μ, σ) output by $\mathsf{LinAuth}(M, \vec{\boldsymbol{\sigma}}, \vec{\boldsymbol{v}})$, the algorithm $\mathsf{LinVer}(sk, \vec{\boldsymbol{v}}, (\mu, \sigma))$ always outputs 1.

The unforgeability game between an adversary $\mathcal{A}$ and a challenger is defined below.

1. The adversary $\mathcal{A}$ chooses a message M.
2. The challenger computes an authentication key $sk \leftarrow \mathsf{Kg}(1^\lambda)$, and computes a vector-tag $\vec{\boldsymbol{\sigma}} \leftarrow \mathsf{LinTag}(sk, M)$. It sends $\vec{\boldsymbol{\sigma}}$ to $\mathcal{A}$.
3. $\mathcal{A}$ produces a non-zero vector-challenge $\vec{\boldsymbol{v}} = (v_1, v_2, \cdots, v_n)$ and a tuple (μ', σ').
4. If $\mathsf{LinVer}(sk, \vec{\boldsymbol{v}}, (\mu', \sigma')) = 1$ and $\mu' \neq \sum_{i=1}^{n} v_i M_i$, the adversary $\mathcal{A}$ wins.

Definition 3 *If there exists no adversary $\mathcal{A}$, who wins the above game with non-negligible probability, then the HLA scheme is unforgeable with "information theoretic security". If adversaries are limited to be of probabilistic-polynomial-time (ppt), the scheme is unforgeable with "computationally security".*

3.2. POR scheme from Homomorphic Linear Authentication scheme

A homomorphic linear authentication (HLA) scheme can be used to construct a POR scheme in the following ways.

Key Generation. The client generates an authentication key by $\mathsf{Kg}(1^\lambda)$.

File coding. The client encodes the original file F for storage.

- Erasure Correctoin Coding: The original file F is encoded to M with an erasure correction code, which ensures that ρ fraction of M suffices to recover M.
- Authentication Coding: $\vec{\boldsymbol{\sigma}} \leftarrow \mathsf{LinTag}(sk, M)$. The encoded file is $M^* = M||\vec{\boldsymbol{\sigma}}$.

The client submits M^* to the server for storage.

Auditing: The server (prover) $\mathcal{P}$ and the client (verifier) $\mathcal{V}$ define an interactive POR protocol together for the file retrievability.

1. $\vec{\boldsymbol{v}} \leftarrow \mathcal{V}$. The verifier $\mathcal{V}$ outputs a randomized challenge vector $\vec{\boldsymbol{v}} = (v_1, v_2, \cdots, v_n)$.
2. The prover $\mathcal{P}$ uses $\mathsf{LinAuth}(M, \vec{\boldsymbol{\sigma}}, \vec{\boldsymbol{v}})$ algorithm to obtain the aggregated block-tag pair (μ, σ). $\mathcal{P}$ sends the block-tag pair (μ, σ) to $\mathcal{V}$.
3. Let (μ', σ') be the pair received by the verifier $\mathcal{V}$. $\mathcal{V}$ computes $b \leftarrow \mathsf{LinVer}(sk, \vec{\boldsymbol{v}}, (\mu', \sigma'))$. If $b = 1$, $\mathcal{V}$ outputs 1, convinced that $\mu = \mu'$, otherwise outputs 0.

File Extraction. An extraction algorithm $\mathsf{Extr}(sk, \mathcal{P})$ takes as input the secret key sk, is given non-black-box access to $\mathcal{P}$, and outputs file M.

The soundness of the POR scheme requires that any cheating prover who convinces the verification algorithm that it is storing a file M actually has the file. And there exists an extraction algorithm $\mathsf{Extr}(sk, \mathcal{P}')$, which takes as input the secret key sk, is given non-black-box access to $\mathcal{P}'$, and outputs the file M. The soundness of the POR scheme is guaranteed by the unforgeability of the involved HLA scheme. As long as the output (μ', σ') of $\mathcal{P}'$ passes verification in the auditing protocol, the unforgeability of the authentication scheme ensures the correctness of μ' with overwhelming probability. Executions of the POR protocol during auditing output correct μs, which can be used to recover a fraction of M. As long as ρ fraction of M is obtained, the extractor $\mathsf{Extr}(sk, \mathcal{P}')$ can recover the whole M with erasure decoding.

The full security proof of a POR scheme is in three parts (see Shacham and Waters, 2008). The first part involves the proof of unforgeability of the HLA scheme, the second part is the proof of ϵ-soundness, which uses combinatoric techniques, and the last part is the existence of an extraction algorithm to reconstruct the original file M with ρ fraction of it, which uses erasure correction technique.

An HLA scheme with "information-theoretic security" results in a POR scheme with "information" soundness, while a HLA scheme with "computational security" results in a POR scheme with "knowlege" soundness (see Dodis, Vadhan and Wichs, 2009). We will focus on how to implement an efficient HLA scheme in the POR scheme and prove its unforgeability.

3.3. HLA Scheme 1 from the $\frac{s}{q}$-ASU$_2$ of Construction 4

We present a new HLA scheme for POR, which employs the $\frac{s}{q}$-ASU$_2$ of Construction 4 to generate homomorphic tags. We call the scheme HLA Scheme 1 hereafter. All operations are over $GF(q)$. Below we describes our proposal for a homomorphic linear authenticator scheme.

$\mathsf{Kg}(1^\lambda)$. Randomly choose $\alpha, k_i \in GF(q)$ with $i = 1, 2, \cdots, n$. Output $sk = (\alpha, k_1, k_2, \cdots, k_n)$.

$\vec{\boldsymbol{\sigma}} \leftarrow \mathsf{LinTag}(sk, M)$. M is divided into n blocks and s sectors in each block, i.e.,

$$M = \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1s} \\ m_{21} & m_{22} & \cdots & m_{2s} \\ \vdots & & & \\ m_{n1} & m_{n2} & \cdots & m_{ns} \end{pmatrix}, \quad m_{ij} \in GF(q).$$

Compute the tags $\vec{\boldsymbol{\sigma}}$ with

$$\vec{\boldsymbol{\sigma}} = \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_n \end{pmatrix} = M \cdot \begin{pmatrix} \alpha^1 \\ \alpha^2 \\ \vdots \\ \alpha^s \end{pmatrix} + \begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{pmatrix}. \tag{9}$$

The final file to be stored is $M^* = M||\vec{\boldsymbol{\sigma}}$.

$(\vec{\boldsymbol{\mu}}, \sigma) \leftarrow \mathsf{LinAuth}(M, \vec{\boldsymbol{\sigma}}, Q)$. Parse M^* as

$$M^* = M||\vec{\boldsymbol{\sigma}} = \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1s} & \sigma_1 \\ m_{21} & m_{22} & \cdots & m_{2s} & \sigma_2 \\ \vdots & & & & \\ m_{n1} & m_{n2} & \cdots & m_{ns} & \sigma_n \end{pmatrix}.$$

The vector challenge is obtained in this way: randomly pick up some block indices to obtain the index-subset $I \subseteq \{1, 2, \cdots, n\}$. For each $i \in I$, randomly choose v_i from $GF(q)$. Let $Q = \{(i, v_i) \mid i \in I\}$, and let $\vec{\boldsymbol{v}} = (v_1, v_2, \cdots, v_n)^T$ be a vector with $v_i = 0$ for $i \notin I$. Compute

$$(\mu_1, \mu_2, \cdots, \mu_s, \sigma) = \vec{\boldsymbol{v}}^T \cdot M^* = (v_1, v_2, \cdots, v_n) \cdot \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1s} & \sigma_1 \\ m_{21} & m_{22} & \cdots & m_{2s} & \sigma_2 \\ \vdots & & & & \\ m_{n1} & m_{n2} & \cdots & m_{ns} & \sigma_n \end{pmatrix}. \tag{10}$$

Return $\vec{\boldsymbol{\mu}} = (\mu_1, \mu_2, \cdots, \mu_n)$ and σ.

$b \leftarrow \mathsf{LinVer}(sk, \vec{\boldsymbol{v}}, (\mu', \sigma'))$. Parse $\vec{\boldsymbol{\mu}}$ as $(\mu_1, \mu_2, \cdots, \mu_s)^T \in GF(q)^s$ and $\sigma \in GF(q)$.

Check whether the following relation holds or not.

$$\sigma = (v_1, v_2, \cdots, v_n) \cdot \begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{pmatrix} + (\mu_1, \mu_2, \cdots, \mu_s) \cdot \begin{pmatrix} \alpha \\ \alpha^2 \\ \vdots \\ \alpha^s \end{pmatrix}. \tag{11}$$

If so, output 1, otherwise output 0.

3.4. Correctness

The correctness of the protocol is given as follows. (10) implies $\vec{\boldsymbol{\mu}}^T || \sigma = \vec{\boldsymbol{v}}^T \cdot (M || \vec{\boldsymbol{\sigma}})$, hence $\vec{\boldsymbol{\mu}}^T = \vec{\boldsymbol{v}}^T \cdot M$ and $\sigma = \vec{\boldsymbol{v}}^T \cdot \vec{\boldsymbol{\sigma}}$. According to (9), we know that

$$\begin{aligned} \sigma &= \vec{\boldsymbol{v}}^T \cdot \left[M \cdot \begin{pmatrix} \alpha^1 \\ \alpha^2 \\ \vdots \\ \alpha^s \end{pmatrix} + \begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{pmatrix} \right] = \left(\vec{\boldsymbol{v}}^T \cdot M \right) \cdot \begin{pmatrix} \alpha^1 \\ \alpha^2 \\ \vdots \\ \alpha^s \end{pmatrix} + \vec{\boldsymbol{v}}^T \cdot \begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{pmatrix} \\ &= (\mu_1, \mu_2, \cdots, \mu_s) \cdot \begin{pmatrix} \alpha^1 \\ \alpha^2 \\ \vdots \\ \alpha^s \end{pmatrix} + (v_1, v_2, \cdots, v_n) \cdot \begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{pmatrix}, \end{aligned}$$

which means that (11) holds, given that the prover and the verifier behave according to the POR protocol.

3.5. Security

Here, we show that the HLA scheme 1 from the s/q-ASU$_2$ of Construction 4 is unforgeable in an information-theoretic sense. The POR scheme with the HLA scheme (see Subsection 3.2) is unforgeable too.

THEOREM 1 *The HLA scheme 1 from the s/q-ASU$_2$ of Construction 4 and the POR scheme obtained from the HLA scheme 1 are both unforgeable with information-theoretic security.*

Proof. If the adversary gives a response for the challenge $Q = \{(i, v_i)\}$ that passes the verification algorithm but is not what would have been computed by an honest prover, then the adversary wins. We will analyze the probability that the adversary wins.

Let

$$M^* = \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1s} & \sigma_1 \\ m_{21} & m_{22} & \cdots & m_{2s} & \sigma_2 \\ \vdots & & & & \\ m_{n1} & m_{n2} & \cdots & m_{ns} & \sigma_n \end{pmatrix} = M || \vec{\boldsymbol{\sigma}}$$

be the stored file M together with block signatures $\vec{\boldsymbol{\sigma}} = (\sigma_1, \sigma_2, \cdots, \sigma_n)^T$, issued by the challenger.

Suppose the challenge $Q = \{(i, v_i) \mid i \in I\}$ determines the query vector $\vec{\boldsymbol{v}} = (v_1, v_2, \cdots, v_n)^T$ with $v_i = 0$ for $i \notin I$.

Let $(\mu_1, \mu_2, \cdots, \mu_s, \sigma)$ be the expected response from an honest prover, i.e.,

$$(\mu_1, \mu_2, \cdots, \mu_s, \sigma) = \vec{\boldsymbol{v}}^T \cdot M^* = (v_1, v_2, \cdots, v_n) \cdot \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1s} & \sigma_1 \\ m_{21} & m_{22} & \cdots & m_{2s} & \sigma_2 \\ \vdots & & & & \\ m_{n1} & m_{n2} & \cdots & m_{ns} & \sigma_n \end{pmatrix}, \tag{12}$$

where

$$\begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_n \end{pmatrix} = M \cdot \begin{pmatrix} \alpha^1 \\ \alpha^2 \\ \vdots \\ \alpha^s \end{pmatrix} + \begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{pmatrix}. \tag{13}$$

It follows that $\sigma = \sum_{i \in I} v_i k_i + \sum_{j=1}^{s} \alpha^j \mu_j$. Hence this is exactly an s/q-ASU_2 with $(\alpha, k_1, k_2, \cdots, k_n)$ as the hash index, i.e., the authentication key for the corresponding authentication code.

Let $(\tilde{\mu}_1, \tilde{\mu}_2, \cdots, \tilde{\mu}_s, \tilde{\sigma})$ be the adversary's forgery that passes the verification. Then we have

$$\tilde{\sigma} = \sum_{i \in I} v_i k_i + \sum_{j=1}^{s} \alpha^j \tilde{\mu}_j. \tag{14}$$

Let $\Delta\sigma = \tilde{\sigma} - \sigma$ and $\Delta\mu_j = \tilde{\mu_j} - \mu_j$, we have that

$$\Delta\sigma = (\Delta\mu_1, \Delta\mu_2, \cdots, \Delta\mu_n) \cdot \begin{pmatrix} \alpha^1 \\ \alpha^2 \\ \vdots \\ \alpha^s \end{pmatrix}. \tag{15}$$

Since $(\tilde{\mu}_1, \tilde{\mu}_2, \cdots, \tilde{\mu}_s, \tilde{\sigma}) \neq (\mu_1, \mu_2, \cdots, \mu_s, \sigma)$, it follows that there exists at least one j such that $\mu_j \neq \tilde{\mu_j}$, indicating that $(\Delta\mu_1, \Delta\mu_2, \cdots, \Delta\mu_n)$ is a matrix of rank 1.

According to Lemma 1 and the properties of s/q-ASU_2, among the q possible choices of $(\alpha, k_1, k_2, \cdots, k_n)$ satisfying (13), at most s choices also satisfy (15).

Due to the randomness of $(\alpha, k_1, k_2, \cdots, k_n)$, we know that the probability that the adversary forges a valid response $(\tilde{\mu}_1, \tilde{\mu}_2, \cdots, \tilde{\mu}_s, \tilde{\sigma})$ is s/q.

As to the POR scheme constructed from the HLA scheme, the auditing process may involve many executions of the interactive protocol between Prover $\mathcal{P}'$

(played by the adversary) and Verifier $\mathcal{V}$ (who will call $b \leftarrow \mathsf{LinVer}(sk, \vec{\boldsymbol{v}}, (\mu', \sigma'))$ for each execution).

The first execution of the interactive protocol with failure verification will eliminate s choices from the total of q choices. Hence, the adversary succeeds in forgery during the second execution of the interactive protocol with probability $s/(q-s)$.

Similarly the adversary's success probability during $k+1$-th execution is at most $s/(q-ks)$.

Suppose that there are totally q_e executions, and the adversary's success probability in POR is at most

$$1-(1-\frac{s}{q})\cdot(1-\frac{s}{q-s})\cdots(1-\frac{s}{q-(q_e-1)s})\approx\frac{q_e s}{q-q_e s}.$$

Here q is the size of the finite field, which is exponential to the security parameter λ, and q_e is the number of of interactions between Prover $\mathcal{P}'$ and Verifier $\mathcal{V}$, which is polynomial to λ. Obviously, the probability $\frac{q_e s}{q-q_e s}$ is negligible. ∎

3.6. HLA Scheme 2 from the $\frac{1}{q}$-ASU$_2$ of Construction 3

Similarly, the $\frac{1}{q}$-ASU$_2$ of Construction 3 results in another HLA scheme, namely, HLA scheme 2. HLA scheme 2 is obtained with the following modifications in HLA scheme 1 in Subsection 3.3.

- $\mathsf{Kg}(1^\lambda)$ is modified to output $(\alpha_1, \alpha_2, \cdots, \alpha_s, k_1, k_2, \cdots, k_n)$, with α_i randomly chosen from $GF(q)$ for $i = 1, 2, \ldots, s$. The local storage of clients is $(n+s)\lceil \log_2 q \rceil$ bits.
- The vector $(\alpha, \alpha^2, \cdots, \alpha^s)^T$ in (9) and (11) should be replaced by $(\alpha_1, \alpha_2, \cdots, \alpha_s)^T$.

3.7. HLA schemes with computational security

As to the HLA scheme 1 from Construction 4, if the authentication key is generated by a Pseudo-Random Function (PRF) $f : \{0,1\}^* \times GF(q) \to GF(q)$, the local storage of clients can be reduced. But the price is that the information-theoretic security is reduced to computational security. With the following modifications, HLA scheme 1 with information-theoretic security is transformed into HLA scheme 1' with computational security.

- $\mathsf{Kg}(1^\lambda)$: Randomly choose $\alpha, k_{prf} \in GF(q)$. Return (α, k_{prf}).
- The authentication keys $(k_1, k_2, \cdots, k_n)$ used in (9) and (11) are computed by $k_i \leftarrow f_{k_{prf}}(i)$ for $i = 1, 2, \cdots, n$.

This reduces the local storage of clients of HLA Scheme 1 from $(n+1)\lceil \log_2 q \rceil$ bits to $2\lceil \log_2 q \rceil$ bits.

Scheme	Local storage(bits)	I.T.S.	C.S.
HLA scheme 1	$(n+1)\lceil \log_2 q \rceil$	$\surd$	$\surd$
HLA scheme 2	$(n+s)\lceil \log_2 q \rceil$	$\surd$	$\surd$
HLA scheme 1'	$2\lceil \log_2 q \rceil$	$\times$	$\surd$
HLA scheme 2'	$(s+1)\lceil \log_2 q \rceil$	$\times$	$\surd$

Table 1: Comparison of the SW scheme and ours.

The above modifications apply to HLA scheme 2 to result in HLA scheme 2' with computational security. And the local storage of clients is reduced to $(s+1)\lceil \log_2 q \rceil$ bits.

Note that HLA scheme 2' is exactly the HLA scheme used in the SW scheme (see Shacham and Waters, 2008).

The comparison of HLA schemes is given in Table 1, where I.T.S. denotes Information-Theoretic Security and C.S. denotes Computational Security.

4. Conclusion

The essence of the POR scheme is an authentication scheme, which ensures that the storage server succeeds in cheating any client by forging a valid response without the client's data at hand with negligible probability. If an HLA scheme gives homomorphic verifiable tags, the server is able to give aggregated responses, thus reducing the computational and communication complexity. This paper exploits the application of some ϵ-ASU_2 to HLA scheme in POR. We give two constructions of ϵ-ASU_2, one being $1/q$-ASU_2 and the other s/q-ASU_2. Both of the two ASU_2 give homomorphic linear authenticator schemes with information-theoretic security, where the s/q-ASU_2 results in a shorter authentication key than the $1/q$-ASU_2. To further reduce the local storage of authentication key, a Pseudo-Random Function(PRF) can be applied to generate authentication keys. But the HLA schemes from ASU_2s turn out to be computationally secure. The HLA scheme constructed from the $1/q$-ASU_2 and a PRF turns out to be the SW scheme (see Shacham and Waters, 2008), the local storage of clients being $O(s)$. The HLA scheme constructed from the s/q-ASU_2 and a PRF needs local storage $O(1)$. Both schemes enjoy shortest responses from the server. Our future work will explore the HLA constructions to deal with adaptive adversaries.

5. References

Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z. and Song, D. (2007) Provable data possession at untrusted stores. In: De Capitani di Vimercati, S., Syverson, P., eds., *CCS '07 Pro-*

ceedings of the 14th ACM Conference on Computer and Communications Security, ACM Press, New York, 598-609.

Carter, J.L. and Wegman, M.N. (1979) Universal classes of hash functions. *Journal of Computer and System Sciences* **18**(2), 143-154.

Deswarte, Y., Quisquater, J.-J. and Saïdane, A. (2004) Remote integrity checking. In: S. Jajodia, L. Strous, eds., *Proceedings of IICIS 2003, IFIP*140, 1-11. Kluwer Academic, Dordrecht.

den Boer, B. (1993) A simple and key-economical unconditional authentication scheme. *Journal of Computer Security* **2**(1), 65-71.

Dodis, Y., Vadhan S. and Wichs, D. (2009) Proofs of retrievability via hardness amplification. *Theory of Cryptography*, **LNCS 5444**, 109-127.

Filho, D. and Barreto, P. (2006) Demonstrating data possession and uncheatable data transfer. *Cryptology ePrint Archive, Report 2006/150*, http://eprint.iacr.org/

Gilbert, E., MacWilliams, F.J. and Sloane, N. (1974) Codes which detect deception. *The Bell System Technical Journal* **53**(3), 405-424.

Juels, A. and Kaliski, B. (2007) PORs: proofs of retrievability for large files. In: De Capitani di Vimercati, S., Syverson, P. , eds., *Proceedings of CCS 2007*. ACM Press, New York, 584-597.

Naor, M. and Rothblum, G. (2005) The complexity of online memory checking. In: E. Tardos, ed., *Proceedings of FOCS 2005*. IEEE Computer Society, Los Alamitos, 573-584.

Schwarz, T. and Miller, E. (2006) Store, forget, and check: Using algebraic signatures to check remotely administered storage. In: M. Ahamad, L. Rodrigues, eds., *Proceedings of ICDCS 2006*. IEEE Computer Society, Los Alamitos, 12-12.

Shacham, H. and Waters, B. (2008) Compact proofs of retrievability. In: *Proceedings of Asiacrypt 2008*, **LNCS 5350**, Springer-Verlag, 90-107.

Sommons, G.J. (1984) Authentication theory/coding theory. *Advances in Cryptology, Proc. Crypto'84*, **LNCS 196**, Springer-Verlag, New York, 411-431.

Sommons, G.J. (1992) A game theory model of digital message Authentication. *Congr. Numer.* 34, 413-424.

Stinson, D.R. (1992) Universal hashing and authentication codes. *Advances in Cryptology-CRYPTO'91*, **LNCS 576**, 74-85.

Stinson, D.R. (1994) Combinatorial techniques for universal hashing. *Journal of Computer and System Sciences* **48**(2), 337-346.

Stinson, D.R. (1994) Universal hashing and authentication codes. *Designs, Codes and Cryptography* **4**(4), 369-380.

Wegman, M.N. and Carter, J.L. (1998) New hash functions and their use in authentication and set equality. *J. Computer and System Sci.* 22, 265-279.