Towards Efficient Provable Data Possession*

Jia Xu Ee-Chien Chang
{xujia,changec}@comp.nus.edu.sg

Abstract

Provable Data Possession ($\mathcal{PDP}$) allows data owner to periodically and remotely audit their data stored in a cloud storage, without retrieving the file and without keeping a local copy. Ateniese *et al.* (CCS 07) proposed the first $\mathcal{PDP}$ scheme, which is very efficient in communication and storage. However their scheme requires a lot of group exponentiation operations: In the setup, one group exponentiation is required to generate a tag per each data block. In each verification, (equivalently) $(m + \ell)$ group exponentiations are required to generate a proof, where m is the size of a data block and ℓ is the number of blocks accessed during a verification. This paper proposed an efficient $\mathcal{PDP}$ scheme. Compared to Ateniese *et al.* (CCS 07), the proposed scheme has the same complexities in communication and storage, but is more efficient in computation: In the setup, no group exponentiations are required. In each verification, only m group exponentiations are required to generate a proof. The security of the proposed scheme is proved under Knowledge of Exponent Assumption and Factoriztion Assumption.

1 Overview

Ateniese *et al.* [ABC+07] proposed the first Provable Data Possession ($\mathcal{PDP}$ for short) scheme. Their scheme is very efficient in communication and storage: the size of a proof is independent on the number of blocks accessed during a verification and the storage overhead due to authentication tags is a fraction[1] of the size of the original data. However, their scheme requires a large number of modular exponentiation in both setup phase and verification phase, and is thus relative expensive in computation.

In this paper, we will propose a new $\mathcal{PDP}$ construction named POS, which requires no modular exponentiation in the setup phase and a smaller number of group exponentiations in verification phase, without sacrificing in communication or storage aspects. We remark that both Ateniese *et al.* [ABC+07, ABC+11] and the proposed scheme in this paper support only private key verification.

*This is the full version of the PDP scheme described in the Appendix of Cryptology ePrint Archive, Report 2011/362.

[1]This fraction is a configurable system parameter.

1.1 A Brief Description of POS

Setup Phase.

Suppose Alice wants to backup her file $\mathtt{F}$ into a cloud storage server provided by Bob. Alice encodes file $\mathtt{F}$ with some error erasure code to obtain data blocks $(\mathtt{F}_0, \ldots, \mathtt{F}_{n-1})$. Alice chooses a RSA modulus $N = pq$, a secret seed, denoted as seed, of a pseudorandom function PRF, and a secret number τ. Let $\phi(N) = (p-1)(q-1)$. With the secret private key $sk = (\phi(N), \mathsf{seed}, \tau)$, Alice produces an authentication tag σ_i for each block $\mathtt{F}_i$:

$$\sigma_i := \tau \mathtt{F}_i \ + \ \mathsf{PRF}_{\mathsf{seed}}(i) \mod \phi(N). \tag{1}$$

We emphasize that the generated authentication tag σ_i is much shorter than a data block $\mathtt{F}_i$. At the end of setup, Alice sends data blocks and tags $\{(i, \mathtt{F}_i, \sigma_i) : i \in [0, n-1]\}$ together with a public key $pk = (N)$ to Bob.

Audit.

Later, Alice may remotely verify the integrity of her data file stored with Bob periodically. In each verification session, Alice randomly selects a subset $C \subset [0, n-1]$ of indices and selects a random weights ν_i for each $i \in C$. Alice sends $\{(i, \nu_i) : i \in C\}$ as challenge to Bob. Bob then finds all data blocks $\mathtt{F}_i$'s and authentication tags σ_i's with index $i \in C$, and apply the linear homomorphism to compute an aggregated message-tag pair $(\mathtt{M}, \sigma)$ as below:

$$\mathtt{M} \ := \ \sum_{i \in C} \nu_i \mathtt{F}_i; \tag{2}$$

$$\sigma \ := \ \sum_{i \in C} \nu_i \sigma_i. \tag{3}$$

We emphasize that the above two equations are computed over integer domain, and thus the bit-length of the linear combination $\mathtt{M}$ (the aggregated authentication tag σ, respectively) is slightly larger than a data block $\mathtt{F}_i$ (an authentication tag σ_i, respectively).

Instead of sending the large message block $\mathtt{M}$ together with authentication tag σ directly to the verifier Alice, Bob is able to produce a *shorter* data-tag pair with the help of Alice. Alice generates a pair of public token pt and secret token st per each verification, where the public token pt is sent to the prover Bob and the secret token st is kept private. With pt and $(\mathtt{M}, \sigma)$, Bob is able to generate a *shorter* message-tag pair, which can be verified by the verifier Alice with the private key and the secret token st.

Illustration Picture

Figure 1 illustrates the scheme POS briefed above. In Figure 1, a rectangle represents a data block, and a circle represents a short authentication tag corresponding to the data block represented by the rectangle that lies above. Those shaded rectangles represent data

blocks that are generated by the error erasure code. In our scheme POS, a data block is treated as a single large integer (much larger than the RSA modulus N). Figure 1 shows an example where a data block is about three times larger than a tag, by dividing each rectangle with dashed lines.

In a verification, a subset of three pairs of blocks and tags are selected, which are aggregated into a single pair of block and tag through linear homomorphism. Since the linear combination is computed over integer domain, the aggregated block (tag, respectively) is slightly larger than an original data block (tag, respectively). With the help of the public token pt provided by the verifier, a shorter block-tag pair can be generated from the long aggregated block-tag pair. The verifier can verify the short block-tag pair using a secret token st.

1.2 Organization

The rest of this chapter is organized as below. The next Section 2 describes the definition of $\mathcal{PDP}$. Section 3 presents the construction of our $\mathcal{PDP}$ scheme POS. Then we analyze the performance of proposed scheme in Section 4 and security in Section 5. At the end, Section 6 closes this chapter.

2 Provable Data Possession: Definition and Formulation

A $\mathcal{PDP}$ scheme consists of five polynomial algorithms (KeyGen, $\mathsf{DEncode}$, $\mathsf{Challenge}$, Prove, Verify), which are described as below.

- $\mathsf{KeyGen}(1^\lambda) \rightarrow (pk, sk)$: Given security parameter λ, the randomized key generating algorithm outputs a public-private key pair (pk, sk).
- $\mathsf{DEncode}(sk, \mathtt{F}) \rightarrow (\mathsf{id}_{\mathtt{F}}, n, \hat{\mathtt{F}})$: Given the private key sk and a data file $\mathtt{F}$, the data encoding algorithm $\mathsf{DEncode}$ produces a unique identifier $\mathsf{id}_{\mathtt{F}}$, file size n (in term of number of blocks) and the encoded file $\hat{\mathtt{F}}$.
- $\mathsf{Challenge}(sk, \mathsf{id}, n) \rightarrow (\mathsf{pt}, \mathsf{st}, \mathsf{Chall})$: The probabilistic algorithm $\mathsf{Challenge}$ takes as input the private key sk, a file identifier id and the file size n (in term of number of blocks), and outputs a public token pt, a private token st and a query Chall.
  Note: (1) The public/secret token pair (pt*,* st*) is generated independently per each verification. (2) In both [ABC+07] and the scheme* POS *that will be presented later in this Chapter,* Chall *is just a subset of indices (in the range* $[0, n-1]$*) and weights (from some group), and has no secret information involved.*
- $\mathsf{Prove}(pk, \mathsf{id}_{\mathtt{F}}, \hat{\mathtt{F}}, \mathsf{pt}, \mathsf{Chall}) \rightarrow \psi$: Given the public key pk, an identifier $\mathsf{id}_{\mathtt{F}}$, an encoded file $\hat{\mathtt{F}}$, a public token pt and a challenge query Chall, the prover algorithm Prove produces a proof ψ.

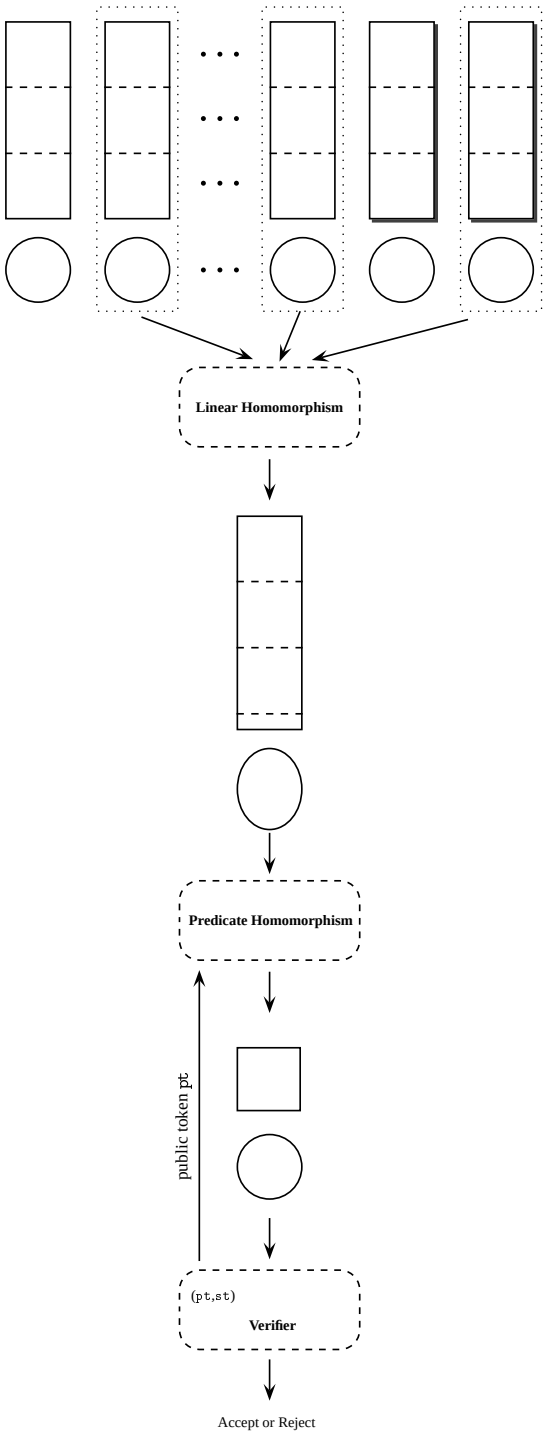

Figure 1: An Efficient $\mathcal{PDP}$ scheme POS

- $\mathsf{Verify}(sk, \mathsf{id_F}, \mathsf{st}, \mathsf{Chall}, \psi) \rightarrow$ `accept` or `reject`: Given the private key sk, an identifier $\mathsf{id_F}$, the secret token st, a challenge query Chall, and a proof ψ, the deterministic

verifying algorithm Verify will output either `accept` or `reject`.

Remark 1 *Compared to the $\mathcal{POR}$, in the above description for $\mathcal{PDP}$, the prover algorithm* Prover *takes as an additional input a public token* pt *and the verifier algorithm* Verify *takes a corresponding secret token* st *as an additional input, where the public/secret token pair* (pt, st) *is generated online by the verifier for each verification session.*

3 POS: An Efficient $\mathcal{PDP}$ Scheme

Like the first $\mathcal{PDP}$ scheme proposed by Ateniese [ABC+07], in our construction below the prover requires a public token to generate a short proof and the verifier requires a corresponding secret token to verify the short proof, where this pair of public-secret tokens are generated by the verifier online per each verification session. Therefore, the homomorphic cryptography component of the below construction is not a MAC scheme, and we do not separate it out as a standalone component.

The description of scheme POS = (KeyGen, DEncode, Challenge, Prove, Verify) is as below.

POS.KeyGen$(1^\lambda) \rightarrow (pk, sk)$

Choose at random a λ bits RSA modulus $N = pq$, such that all of $p, q, p' = (p-1)/2, q' = (q-1)/2$ are primes and the bit-lengths of p and q are the same. Without loss of generality, assume $p' < q'$. Let $\phi(N) = (p-1)(q-1) = 4p'q'$. Let $\mathcal{QR}_N$ denote the subgroup of quadratic residues modulo N. Choose at random a generator g of the subgroup $\mathcal{QR}_N$. Choose at random $\tau \xleftarrow{\$} \mathbb{Z}_{\phi(n)}$. Choose at random a seed, denoted as seed, from the key space of the pseudorandom function $\mathsf{PRF} : \{0,1\}^{2\lambda} \rightarrow \mathbb{Z}_{\phi(N)}$. The public key is $pk = (N, g)$ and the private key is $sk := (g, p, q, \tau, \mathsf{seed})$.
Note: Then size of subgroup $\mathcal{QR}_N$ equals to $\frac{1}{4}\phi(N) = p'q'$.

POS.DEncode$(sk, \mathtt{F}) \rightarrow (\mathsf{id}, \{(i, \mathtt{F}_i, \sigma_i) : i \in [0, n-1]\})$

Let $\rho \in (0,1)$ be a system parameter. Apply rate-ρ error erasure code on data file $\mathtt{F}$ to generate blocks $(\mathtt{F}_0, \ldots, \mathtt{F}_{n-1})$, such that each block $\mathtt{F}_i \in \{0,1\}^{m\lambda}$ and any ρ-fraction of blocks $\mathtt{F}_i$'s can recover the original file $\mathtt{F}$. Choose a unique identifier id for the file $\mathtt{F}$. For each data block $\mathtt{F}_i, i \in [0, n-1]$, the data owner computes an authentication tag σ_i:

$$\sigma_i := \tau \mathtt{F}_i + \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id} \| i) \mod \phi(N). \tag{4}$$

Output $(\mathsf{id}, \{(i, \mathtt{F}_i, \sigma_i) : i \in [0, n-1]\})$.

$\underline{\mathsf{POS.Challenge}(sk, \mathsf{id}, n) \rightarrow (\mathsf{pt}, \mathsf{st}, \{(i, \nu_i) : i \in C\})}$

Find at random a secret value $d \xleftarrow{\$} \mathbb{Z}_{\phi(N)}$, and computes $g_d := g^d \mod N$. The public token is $\mathsf{pt} := g_d$ and the secret token is $\mathsf{st} := d$. Chooses a subset $C \subset [0, n-1]$ at random and choose weight $\nu_i \xleftarrow{\$} \mathbb{Z}_{\phi(N)}$ at random for each $i \in C$. Output $(\mathsf{pt}, \mathsf{st}, \{(i, \nu_i) : i \in C\})$.

$\underline{\mathsf{POS.Prove}(pk, \mathsf{id}, \hat{\mathtt{F}}, \mathsf{pt}, \{(i, \nu_i) : i \in C\}) \rightarrow (\psi_1, \psi_2)}$

Find all selected blocks $\mathtt{F}_i$'s and tags t_i's, and compute (π_1, π_2) as below over integer domain:

$$\pi_1 := \sum_{i \in C} \nu_i \mathtt{F}_i; \tag{5}$$

$$\pi_2 := \sum_{i \in C} \nu_i t_i. \tag{6}$$

Compute (ψ_1, ψ_2) as below

$$\psi_1 := g_d^{\pi_1} \mod N; \tag{7}$$

$$\psi_2 := g^{\pi_2} \mod N. \tag{8}$$

Send (ψ_1, ψ_2) to the verifier.

$\underline{\mathsf{POS.Verify}(sk, \mathsf{id}, \mathsf{st}, \{(i, \nu_i) : i \in C\}, \psi_1, \psi_2) \rightarrow \mathtt{accept}\ \textbf{or}\ \mathtt{reject}}$

With the private key $sk = (g, p, q, \tau, \mathsf{seed})$ and the secret token $\mathsf{st} = d$, check whether $\psi_1 \in \mathcal{QR}_N$ is a quadratic residue modulo N and the following equality holds.

$$(\psi_1)^\tau \stackrel{?}{=} \left(\frac{\psi_2}{g^{\sum_{i \in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id} \| i)}} \right)^d \mod N \tag{9}$$

If both verifications succeed, then output `accept`; otherwise output `reject`.

We remark that in the above scheme, we can change[2] the proof from (ψ_1, ψ_2) to $(\psi_1, \mathtt{SHA256}(\psi_2))$ to reduce the size from 2λ bits to $(\lambda{+}256)$ bits using a secure hash function `SHA256` [NIS02], similar to Ateniese *et al.* [ABC+07].

[2]Meantime, change the range of the secret token d from $\mathbb{Z}_{\phi(N)}$ to $\mathbb{Z}^*_{\phi(N)}$, in order to recover ψ_2 from ψ_1 through Equation (9).

Table 1: Comparison between Ateniese *et al.* [ABC$^+$07] and POS proposed in this chapter w.r.t. a 1GB file. The setting is described in Example 1.

Scheme	Group Size	Communication bits	Computation (Data Preprocess)	Computation (Prove)
Ateniese [ABC$^+$07]	$\lambda = 1024$	$2\lambda + 160 + 256 = 2464$	2^{23} `exp.` over $\mathbb{Z}_N^*$	$(100 + \ell)$ `exp.` over $\mathbb{Z}_N^*$
POS	$\lambda = 1024$	$2\lambda + 160 + 256 = 2464$	2^{23} `mul.` over $\mathbb{Z}_N^*$	102 `exp.` over $\mathbb{Z}_N^*$

4 Performance Analysis

TODO: Compared to PDP

The proposed scheme is efficient in storage, communication and computation. The storage overhead due to authentication tags is $1/m$ of the file size (after error erasure encoding). The proof size in a verification is 2λ bits. In the setup, the data encoding algorithm DEncode requires n number of pseudorandom function evaluations, modular additions/multiplications. In a verification session, the computation of the prover algorithm Prove is dominated by the exponentiation with large integer exponent in Equation (7), which is equivalent to m number of group exponentiation in $\mathbb{Z}_N^*$. The verifier algorithm Verify requires one modular division, three modular exponentiation in $\mathbb{Z}_N^*$, ℓ number of additions/multiplications in $\mathbb{Z}_{\phi(N)}$, and ℓ number of pseudorandom function evaluations, where $\ell = |C|$ is the number of indices selected during a verification.

4.1 Comparison

We compare the proposed scheme POS and Ateniese *et al.* [ABC$^+$07,ABC$^+$11] in Table 1 in the setting specified in the below example. We remark that both Ateniese *et al.* [ABC$^+$07, ABC$^+$11] and the proposed scheme supports only private key verification.

Example 1 *After erasure encoding, the file size is 1GB, block size is* $m = 100$*, and storage overhead due to authentication tags is about 10MB for both schemes. For both schemes, we assume that, during a verification, the challenge query* $\{(i, \nu_i) : i \in C\}$ *is represented by two 80 bits PRF seeds. System parameter* ℓ *represents the size of set* C*. All computation times are represented by the corresponding dominant factor.* ***exp*** *and* ***mul*** *denote the group exponentiation and group multiplication respectively in the corresponding group. Note that one 1024 bits modular exponentiation takes roughly 5 millisecond in a standard PC.*

5 Security Analysis of Scheme POS

5.1 Security Formulation

We review the Provable Data Possession formulation proposed by Ateniese *et al.* [ABC$^+$07, ABC$^+$11]. The $\mathcal{PDP}$ security game between a *probabilistic polynomial time* (PPT) adver-

sary $\mathcal{A}$ and a PPT challenger $\mathcal{C}$ w.r.t. a $\mathcal{PDP}$ scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{DEncode}, \mathsf{Challenge}, \mathsf{Prove}, \mathsf{Verify})$ is as below.

Setup: The challenger $\mathcal{C}$ runs the key generating algorithm KeyGen to obtain public-private key pair (pk, sk). The challenger $\mathcal{C}$ gives the public key pk to the adversary $\mathcal{A}$ and keeps the private key sk securely.

Learning: The adversary $\mathcal{A}$ adaptively makes queries, where each query is one of the following:

- Store query ($\mathtt{F}$): Given a data file $\mathtt{F}$ chosen by $\mathcal{A}$, the challenger $\mathcal{C}$ responses by running data encoding algorithm $(\mathsf{id}, \hat{\mathtt{F}}) \leftarrow \mathsf{DEncode}(sk, \mathtt{F})$ and sending the encoded data file $\hat{\mathtt{F}}$ together with its identifier id to $\mathcal{A}$.
- Verification query (id): Given a file identifier id chosen by $\mathcal{A}$, if id is the (partial) output of some previous store query that $\mathcal{A}$ has made, then the challenger $\mathcal{C}$ initiates a $\mathcal{PDP}$ verification with $\mathcal{A}$ w.r.t. the data file $\mathtt{F}$ associated to the identifier id in this way:
  - $\mathcal{C}$ runs the algorithm $\mathsf{Challenge}$ to generate a pair of public-secret tokens $(\mathsf{pt}, \mathsf{st})$ and the a challenge query Chall, and sends $(\mathsf{pt}, \mathsf{Chall})$ to the adversary $\mathcal{A}$ and keeps st safely. The secret token st will be used in the verifier Verify algorithm.
  - $\mathcal{A}$ produces a proof ψ w.r.t. the challenge Chall;
    Note: adversary $\mathcal{A}$ may generate the proof in an arbitrary method rather than applying the algorithm Prove.
  - $\mathcal{C}$ verifies the proof ψ by running algorithm $\mathsf{Verify}(sk, \mathsf{id}, \mathsf{Chall}, \psi)$. Denote the output as b.

  $\mathcal{C}$ sends the decision bit $b \in \{\mathtt{accept}, \mathtt{reject}\}$ to $\mathcal{A}$ as feedback. Otherwise, if id is not the (partial) output of any previous store query that $\mathcal{A}$ has made, $\mathcal{C}$ does nothing.

Commit: Adversary $\mathcal{A}$ chooses a file identifier id^* among all file identifiers she obtains from $\mathcal{C}$ by making store queries in **Learning** phase, and commit id^* to $\mathcal{C}$. Let $\mathtt{F}^*$ denote the data file associated to identifier id^*.

Retrieve: The challenger $\mathcal{C}$ initiates one $\mathcal{PDP}$ verifications with $\mathcal{A}$ w.r.t. the data file $\mathtt{F}^*$, where $\mathcal{C}$ plays the role of verifier and $\mathcal{A}$ plays the role of prover, as in the **Learning** phase. Suppose the challenger $\mathcal{C}$ asks $\mathcal{A}$ to check all data blocks $\mathtt{F}_i$ of $\mathtt{F}^*$ with index $i \in \mathbf{C}$. The challenger $\mathcal{C}$ extracts file blocks $\{\mathtt{F}'_i : i \in \mathbf{C}\}$ from $\mathcal{A}$'s storage by applying a PPT knowledge extractor. The adversary $\mathcal{A}$ wins this $\mathcal{PDP}$ security game, if the challenger $\mathcal{C}$ accepts $\mathcal{A}$'s response in the verification. The challenger $\mathcal{C}$ wins this game, if the extracted blocks $\{(i, \mathtt{F}'_i) : i \in \mathbf{C}\}$ are identical to the original $\{(i, \mathtt{F}_i) : i \in \mathbf{C}\}$.

Definition 1 ([ABC+07]) *A $\mathcal{PDP}$ scheme is sound if for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the above $\mathcal{PDP}$ security game is negligibly close to the probability that $\mathcal{C}$ wins the same security game. That is*

$$\mathsf{Pr}[\mathcal{A} \textit{ wins } \mathcal{PDP} \textit{ game }] \leq \mathsf{Pr}[\mathcal{C} \textit{ wins } \mathcal{PDP} \textit{ game }] + negl. \tag{10}$$

5.2 KEA Assumption

The Knowledge of Exponent Assumption (KEA) is introduced by Damgård [Dam92] and subsequently appears in many works [HT98, BP04a, BP04b, Kra05, Den06]. The below is a variant version of KEA in the RSA ring given by Ateniese *et al.* [ABC$^+$07].

Assumption 1 (Knowledge of Exponent Assumption [Dam92, BP04a]) *Let $N = pq$ be a RSA modulus, $g \in \mathbb{Z}_N^*$ and s be an positive integer. For any PPT algorithm $\mathcal{A}$ that takes (N, g, g^s) as input and r as random coin and returns (C, Y) such that $Y = C^s \mod N$, there exists a PPT* extractor *algorithm $\bar{\mathcal{A}}$ which, given (N, g, g^s, r) as input, outputs x such that $g^x = C \mod N$.*

Remark 2

- *Note that the extractor $\bar{\mathcal{A}}$ has access to $\mathcal{A}$'s input (N, g, g^s) and $\mathcal{A}$'s random coin r, thus $\bar{\mathcal{A}}$ can replay step by step the process how $\mathcal{A}$ computes (C, Y) from (N, g, g^s).*
- *This assumption has been shown to hold in generic group by Abe and Fehr [AF07].*

Assumption 2 (Factorization Assumption [RSA78]) *We say an integer N is a RSA modulus, if $N = pq$ and all of $p, q, \frac{p-1}{2}, \frac{q-1}{2}$ are prime numbers and bit-lengths of p and q are equal. Then for any PPT adversary $\bar{\mathcal{A}}$, the probability that $\mathcal{A}$ can factorize a randomly chosen λ bits RSA modulus, is negligible in λ.*

5.3 Security Proof

Theorem 5.1 (POS is Sound) *If the Knowledge of Exponent Assumption 1 holds and the pseudorandom function* PRF *is secure, then the proposed scheme* POS *is sound.*

The proof below is similar to the proof of Ateniese *et al.* [ABC$^+$07] in the high level: If an adversary $\mathcal{A}$ wins the security game, then a knowledge extractor (as in the assumption KEA) can find a linear combination of data blocks (See Equation 5). Then each individual block can be obtained by solving a linear equation system.

Like in previous chapters, at first in Lemma 5.2, we prove Theorem 5.1 in a simplified no-feedback setting, where all decisions (acceptance or rejection) are kept secret from the adversary in the $\mathcal{PDP}$ security game.

Lemma 5.2 *Suppose the Factorization Assumption 2 holds and the pseudorandom function* PRF *is secure. Then the proof (ϕ_1, ϕ_2) in the proposed scheme* POS *is unforgeable in the no-feedback setting, where all acceptance or rejection decisions are kept secure from the adversary in the $\mathcal{PDP}$ security game. More precisely, let $(\hat{\phi}_1, \hat{\phi}_2)$ denote the adversary $\mathcal{A}$'s response in the* ***Retrieve*** *phase of the $\mathcal{PDP}$ security game w.r.t.* POS*. The probability*

$$\Pr[\textit{Verifier accepts } (\hat{\phi}_1, \hat{\phi}_2) \quad \wedge \quad (\hat{\phi}_1, \hat{\phi}_2) \neq (\phi_1, \phi_2)] \leq negl(\lambda) \tag{11}$$

is negligible.

Proof of Lemma 5.2:
Game 1 The first game is the same as the $\mathcal{PDP}$ security game, except that

- All acceptance or rejection decisions are kept secure from the adversary $\mathcal{A}$. Essentially, the challenger in the $\mathcal{PDP}$ security game does not answer verification queries made by the adversary.
- Adversary $\mathcal{A}$ wins in **Game 1**, if $\mathcal{A}$'s forgery proof $(\hat{\phi}_1, \hat{\phi}_2)$ is accepted and it is different from the genuine proof. Formally, let id, $\{(i, \nu_i) : i \in C\}$ and $(\mathsf{pt}, \mathsf{st})$ denote the file identifier, challenge query, and public-secret tokens respectively in the **Retrieve** phase of the $\mathcal{PDP}$ security game, let (ψ_1, ψ_2) denote the corresponding genuine proof and (pk, sk) be the public-private key pair. Adversary $\mathcal{A}$ wins in **Game 1**, if

$$\mathsf{Verify}(sk, \mathsf{id}, \mathsf{st}, \{(i, \nu_i) : i \in C\}, \hat{\psi}_1, \hat{\psi}_2) = \mathtt{accept} \text{ and } (\hat{\psi}_1, \hat{\psi}_2) \neq (\psi_1, \psi_2). \quad (12)$$

Game 2 The second game is the same as **Game 1**, except that the pseudorandom function PRF outputs true randomness. Precisely, the function $\mathsf{PRF}_{\mathsf{seed}}$ is evaluated in the following way:

- The challenger keeps a table to store all previous encountered input-output pairs $(v, \mathsf{PRF}_{\mathsf{seed}}(v))$.
- Given an input v, the challenger lookups the table for v, if there exists an entry (v, u), then return u as output. Otherwise, choose u at random from the range of $\mathsf{PRF}_{\mathsf{seed}}$, insert $(v, \mathsf{PRF}_{\mathsf{seed}}(v) := u)$ into the table and return u as output.

Game 3 The third game is the same as **Game 2**, except that:

- The range of the function PRF is changed from $\mathbb{Z}_{\phi(N)}$ to $\mathbb{Z}_N$. Note that in this game, PRF is evaluated in the same way as in **Game 2**;
- The range of the authentication tag is also changed from $\mathbb{Z}_{\phi(N)}$ to $\mathbb{Z}_N$. More precisely, the Equation (4) (on page 5) is replaced by the following equations

$$\sigma_i := \tau \mathtt{F}_i + \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id} \| i) \mod N. \quad (13)$$

We remark that in **Game 3**, the challenger is not able to verify adversary's response, and the challenger does not need to do verification either, since in the no-feedback setting, the challenger will not answer verification queries made by the adversary.

Claim 5.1 *If there is a non-negligible difference in a PPT adversary $\mathcal{A}$'s success probability between* **Game 1** *and* **Game 2**, *then there exists another PPT adversary $\mathcal{B}$ that can break the security of the pseudorandom function* PRF. *More precisely,*

$$|\mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game 1}] - \mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game 2}]| \leq N_{\mathsf{PRF}} \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{PRF}},$$

where N_{PRF} is the number of distinct evaluations of pseudorandom function PRF *required and* $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{PRF}}$ *denotes the probability that* $\mathcal{B}$ *can distinguish the output of* PRF *from true randomness.*

The above Claim 5.1 can be proved using a standard hybrid argument [Gol06]. Here we save the details.

Claim 5.2 *For any computationally unbounded adversary* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *can find the secret value* τ *after interacting in* **Game 2***, is* $\frac{1}{\phi(N)}$.

Proof (Proof Sketch of Claim 5.2): In **Game 2**, the function PRF outputs true random numbers in $\mathbb{Z}_{\phi(N)}$ and thus the secret value τ is hidden perfectly. Therefore, the probability that an (computationally unbounded) adversary $\mathcal{A}$ can find τ is $\frac{1}{\phi(N)}$. Recall that τ is chosen at random from group $\mathbb{Z}_{\phi(N)}$. □

Claim 5.3 *For any PPT adversary* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *can factorize* N *after interacting in* **Game 3** *is negligible.*

Proof (Proof Sketch of Claim 5.3): Recall that in **Game 3**, the authentication tag σ_i for each block is a group element chosen at random from $\mathbb{Z}_N$. Suppose a PPT adversary $\mathcal{A}$ factorizes the RSA modulus N after interacting in **Game 3**.

Based on $\mathcal{A}$, we construct a PPT adversary $\mathcal{B}$ to factorize N. Given only the RSA modulus N, the adversary $\mathcal{B}$ can play the role of challenger to setup[3] the $\mathcal{PDP}$ security game w.r.t. scheme POS, and answer store queries made by the adversary $\mathcal{A}$ by sampling uniform random number from $\mathbb{Z}_N$ as the authentication tag σ_i. Thus,

$$\mathsf{Pr}[\mathcal{A} \text{ factorizes } N \text{ in } \textbf{Game 3}] \leq \mathsf{Pr}[\mathcal{B} \text{ factorizes } N] = \mathsf{Adv}_{\mathcal{B}}^{\text{FACT}}. \tag{14}$$

□

Claim 5.4 *For any PPT adversary* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *can factorize* N *after interacting in* **Game 2** *is negligible.*

Proof of Claim 5.4: We will show that any PPT adversary cannot distinguish between **Game 2** and **Game 3**. As a result, Claim 5.3 can imply Claim 5.4.

Now we study the *statistical difference* [SV03] between uniform random variables over $\mathbb{Z}_{\phi(N)}$ and over $\mathbb{Z}_N$.

[3]From the input N, $\mathcal{B}$ can generate the public key and simulate the algorithm DEncode. In the no-feedback setting, $\mathcal{B}$ does not need to do verification, so secret key is not be necessary.

Let X be a uniform random variable over $\mathbb{Z}_{\phi(N)}$ and Y be a uniform random variable over $\mathbb{Z}_N$. The statistical difference [SV03] between X and Y is

$$\begin{aligned}
\mathsf{SD}(X,Y) &\stackrel{\text{def}}{=} \frac{1}{2}\sum_{a}|\mathsf{Pr}[X=a]-\mathsf{Pr}[Y=a]| \\
&= \frac{1}{2}\sum_{a\in\mathbb{Z}_{\phi(N)}}|\mathsf{Pr}[X=a]-\mathsf{Pr}[Y=a]| \;+\; \frac{1}{2}\sum_{a\in\mathbb{Z}_N\setminus\mathbb{Z}_{\phi(N)}}|\mathsf{Pr}[X=a]-\mathsf{Pr}[Y=a]| \\
&= \frac{1}{2}\left(\frac{1}{\phi(N)}-\frac{1}{N}\right)\times\phi(N) \;+\; \frac{1}{2}\left(\frac{1}{N}-0\right)\times(N-\phi(N)) \\
&= 1-\frac{\phi(N)}{N} \\
&= 1-(1-\frac{1}{p})(1-\frac{1}{q}) \\
&= \frac{1}{p}+\frac{1}{q}-\frac{1}{pq}.
\end{aligned}$$

Let N_0 be a positive integer. Let $X_i, i=1,2,\ldots,N_0$, be independently and identically distributed uniform random variables over $\mathbb{Z}_{\phi(N)}$, and $Y_i, i=1,2,\ldots,N_0$, be independently and identically distributed uniform random variables over $\mathbb{Z}_N$. According to Fact 2.1 and Fact 2.3 of Sahai and Vadhan [SV03], we have

$$\mathsf{SD}((X_1,\ldots,X_{N_0}),\ (Y_1,\ldots,Y_{N_0})) \le \sum_{i\in[1,N_0]}\mathsf{SD}(X_i,\ Y_i). \tag{15}$$

The right hand side of the above Equation (15) is

$$\sum_{i\in[1,N_0]}\mathsf{SD}(X_i,\ Y_i) = N_0\times\mathsf{SD}(X,\ Y) = N_0\left(\frac{1}{p}+\frac{1}{q}-\frac{1}{pq}\right). \tag{16}$$

Suppose the adversary $\mathcal{A}$ obtains exactly N_{PRF} authentication tags σ_i (σ'_i respectively) for N_{PRF} different indices i's in **Game 2** (**Game 3** respectively). Since σ_i's are independently and identically distributed uniform random variables over $\mathbb{Z}_{\phi(N)}$ and σ'_is are independently and identically distributed uniform random variables over $\mathbb{Z}_N$, the difference of the adversary's views[4] in **Game 2** and **Game 3** is bounded as below

$$\mathsf{SD}(\textsc{View}_{\mathcal{A}}^{\textbf{Game 2}},\ \textsc{View}_{\mathcal{A}}^{\textbf{Game 3}}) \le N_{\mathsf{PRF}}\left(\frac{1}{p}+\frac{1}{q}-\frac{1}{pq}\right). \tag{17}$$

The adversary $\mathcal{A}$ is polynomially bounded, which implies N_{PRF} is polynomially bounded. Therefore, the statistical difference $\mathsf{SD}(\textsc{View}_{\mathcal{A}}^{\textbf{Game 2}},\ \textsc{View}_{\mathcal{A}}^{\textbf{Game 3}})$ is negligible in $\lambda \approx \log N \approx 2\log p \approx 2\log q$, and there is no adversary can distinguish between **Game 2** and **Game 3**.

[4] Adversary's view is a transcript of all messages received.

Combining with Claim 5.3, we conclude that the probability

$$\mathsf{Pr}[\mathcal{A} \text{ factorizes } N \text{ in } \textbf{Game 2}] \leq \mathsf{Pr}[\mathcal{A} \text{ factorizes } N \text{ in } \textbf{Game 3}] + N_{\mathsf{PRF}} \left(\frac{1}{p} + \frac{1}{q} - \frac{1}{pq}\right)$$

is negligible in $\lambda \approx \log N$. The proof for Claim 5.4 is complete. □

Claim 5.5 *Let $(\hat{\psi}_1, \hat{\psi}_2)$ denote the adversary $\mathcal{A}$'s output in the* **Game 2** *and (ψ_1, ψ_2) be the corresponding genuine output which shares the same values $\{(i, \nu_i) : i \in C\}$ with the forgery output. Then,*

$$\mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game 2} \wedge (\hat{\psi}_1, \hat{\psi}_2) \neq (\psi_1, \psi_2)] \leq \frac{1}{p'}. \tag{18}$$

$$\mathsf{Pr}[\mathcal{A} \textit{ wins } \textbf{Game 2}] \leq \frac{1}{p'}. \tag{19}$$

Proof of Cliam 5.5: Suppose the adversary $\mathcal{A}$ wins **Game 2**, then $\mathcal{A}$'s forged proof $(\hat{\psi}_1, \hat{\psi}_2)$ is accepted and is different from the genuine output (ψ_1, ψ_2): $(\hat{\psi}_1, \hat{\psi}_2) \neq (\psi_1, \psi_2)$. Since both the forged proof and genuine proof are accepted by the verifier w.r.t. $\{(i, \nu_i) : i \in C\}$ and satisfy the Equation (9) (on page 6), we have

$$\left(\hat{\psi}_1\right)^{\tau} = \left(\frac{\hat{\psi}_2}{g^{\sum_{i \in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id} \| i)}}\right)^{d} \mod N \tag{20}$$

$$(\psi_1)^{\tau} = \left(\frac{\psi_2}{g^{\sum_{i \in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id} \| i)}}\right)^{d} \mod N \tag{21}$$

Dividing Equation (20) with Equation (21), we have

$$\left(\frac{\hat{\psi}_1}{\psi_1}\right)^{\tau} = \left(\frac{\hat{\psi}_2}{\psi_2}\right)^{d} \mod N \tag{22}$$

Recall that the verifier algorithm Verify accepts only if $\psi_1, \hat{\psi}_1 \in \mathcal{QR}_N$. Thus $\frac{\hat{\psi}_1}{\psi_1} \in \mathcal{QR}_N$ is also a quadratic residue. For any element $x \in \mathcal{QR}_N$, $x^{\frac{1}{4}\phi(N)} = 1$, and the multiplicative order of x modulo N will be a factor of $\frac{1}{4}\phi(N) = p'q'$. Since $\frac{\hat{\psi}_1}{\psi_1} \neq 1$, the multiplicative order, denoted with φ, of $\frac{\hat{\psi}_1}{\psi_1}$ modulo N is at least $\min\{p', q'\} = p'$. Thus a computationally unbounded adversary $\mathcal{B}$ can invoke the adversary $\mathcal{A}$ to obtain the above Equation (22) and find the value $(\tau \mod \varphi)$ from Equation (22) by solving a discrete log problem with $\frac{\hat{\psi}_1}{\psi_1}$ as base. The probability that $(\tau \mod \varphi) = \tau$ is

$$\mathsf{Pr}[(\tau \mod \varphi) = \tau] = \mathsf{Pr}[\tau \in \mathbb{Z}_{\varphi}] = \frac{\varphi}{\phi(N)}. \tag{23}$$

By Claim 5.2, we have the probability

$$\begin{aligned}\mathsf{Pr}[\mathcal{A} \text{ wins } \textbf{Game 2} \wedge (\hat{\psi}_1, \hat{\psi}_2) \neq (\psi_1, \psi_2)] &\leq \frac{\mathsf{Pr}[\mathcal{B} \text{ finds } \tau \text{ in } \textbf{Game 2}]}{\mathsf{Pr}[(\tau \mod \varphi) = \tau]} \\ &= \frac{\frac{1}{\phi(N)}}{\frac{\varphi}{\phi(N)}} \\ &= \frac{1}{\varphi} \leq \frac{1}{p'}\end{aligned}$$

is negligible in $\lambda \approx \log N \approx 2 + 2\log p'$. The proof of Claim 5.5 is complete. □

Thus, Lemma 5.2 is proved. □

Lemma 5.3 *Suppose the Factorization Assumption 2 holds and the pseudorandom function* PRF *is secure. Then the proof* (ϕ_1, ϕ_2) *in the proposed scheme* POS *is unforgeable in the feedback setting, where all acceptance or rejection decisions are provided to the adversary in the* $\mathcal{PDP}$ *security game.*

Proof (Proof Sketch of Lemma 5.3): In the simulated security game, the challenger does not have all information of private key and thus cannot answer verification queries.

However, the challenger can construct a simulated verifier: The challenger keeps a local copy of all files and tags and computes the genuine proof by himself/herself. The challenger accepts a proof provided by the adversary, if and only if the received proof is identical to the genuine proof.

We can show that the difference between the simulated verifier and the real verifier is negligible, by using Lemma 5.2. □

Now it is the time to prove the main Theorem 5.1 in this chapter.

Proof of Theorem 5.1: Lemma 5.3 states that the proof in the scheme POS is unforgeable. Since for random value $d \in \mathbb{Z}_{\phi(N)}$, $\mathcal{A}$ can win $\mathcal{PDP}$ security game with non-negligible probability. Then for many values d_i's, $\mathcal{A}$ can compute $(\psi_{i,1} = g_{d_i}^{\pi_1}, \psi_{i,2} = g^{\pi_2})$ correctly. Let us just consider d_1 and d_2 among these d_i's. Let $c = \frac{d_2}{d_1} \mod \phi(N)$. Given input $(g^{d_1}, g^{d_2} = \left(g^{d_1}\right)^c)$, the adversary $\mathcal{A}$ can output $(g^{d_1\pi_1}, g^{d_2\pi_1} = \left(g^{d_1\pi_1}\right)^c)$. By Knowledge of Exponent Assumption (KEA [Dam92]), there exists an extractor that can find M, such that $g^{d_1\pi_1} = g^{d_1 M} \mod N$.

Case 1: $M \neq \pi_1$ If the two integers M and π_1 are distinct (*Caution: Here we treat* M, π_1 *as large integer instead of group elements from* $\mathbb{Z}_{\phi(N)}$), then the difference $M - \pi_1$ has to be a multiple of $\frac{1}{4}\phi(N)$, from which the factorization of N can be found using Miller's result [Mil75].

Case 2: $M = \pi_1$ In this case, the extractor finds π_1, as desired. Recall that the large integer $\pi_1 = \sum_{i \in C} \nu_i \mathtt{F}_i$ (Yes, integer, not group element) is linear equation of file blocks $\mathtt{F}_i$'s. Similar to the proof in Ateniese's PDP [ABC+07], by choose independent weights ν_i's in $|C|$ number of executions of the protocol, we obtain $|C|$ independent linear equations in the unknowns $\mathtt{F}_i, i \in C$. Thus these file blocks $\mathtt{F}_i, i \in C$, can be found by solving the linear equation system over integer domain.

Thus, Theorem 5.1 is proved.

□

6 Summary

In this paper, we proposed a $\mathcal{PDP}$ scheme POS which is very efficient in communication, storage and computation. Compared to Ateniese *et al.* [ABC+07], POS is much more efficient in computation, and equally efficient in communication and storage.

References

[ABC+07] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In *CCS '07: ACM conference on Computer and communications security*, pages 598–609, 2007.

[ABC+11] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Osama Khan, Lea Kissner, Zachary Peterson, and Dawn Song. Remote data checking using provable data possession. *ACM Trans. Inf. Syst. Secur.*, 14:12:1–12:34, 2011.

[AF07] Masayuki Abe and Serge Fehr. Perfect NIZK with adaptive soundness. In *TCC '07: Theory of Cryptography Conference*, 2007.

[BP04a] Mihir Bellare and Adriana Palacio. The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In *CRYPTO '04: Annual International Cryptology Conference on Advances in Cryptology*, pages 273–289, 2004.

[BP04b] Mihir Bellare and Adriana Palacio. Towards Plaintext-Aware Public-Key Encryption Without Random Oracles. In *ASIACRYPT '04: International Conference on the Theory and Application of Cryptology and Information Security*, pages 48–62, 2004.

[Dam92] Ivan Damgård. Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. In *CRYPTO '91: Annual International Cryptology Conference on Advances in Cryptology*, pages 445–456, 1992.

[Den06] Alexander W. Dent. The Cramer-Shoup Encryption Scheme Is Plaintext Aware in the Standard Model. In *EUROCRYPT '06: Annual International Conference on Advances in Cryptology*, pages 289–307, 2006.

[Gol06] O. Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, New York, NY, USA, 2006.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the Existence of 3-Round Zero-Knowledge Protocols. In *CRYPTO '98: Annual International Cryptology Conference on Advances in Cryptology*, pages 408–423, 1998.

[Kra05] Hugo Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *CRYPTO '05: Annual International Cryptology Conference on Advances in Cryptology*, pages 546–566, 2005.

[Mil75] G. L. Miller. Riemann's hypothesis and tests for primality. In *7th ACM symp. on the Theory of Computing*, pages 234–239, 1975.

[NIS02] NIST. National Institute of Standards and Technology. Secure hash standard (SHS). FIPS 180-2, August 2002.

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

[SV03] Amit Sahai and Salil Vadhan. A Complete Problem for Statistical Zero Knowledge. *Journal of the ACM*, 50:196–249, 2003.