Data Privacy in Remote Data Integrity Checking for Secure Cloud Storage

Yong Yu

Centre for Computer and Information Security

University of Wollongong

Australia

- Yong Yu, Man Ho Au, Yi Mu, Willy Susilo et al. Enhanced Privacy of a Remote Data Integrity Checking Protocol for Secure Cloud Storage. **International Journal of Information Security**, accepted, 17 August, 2014.

- Xinyu Fan, Guomin Yang, Yi Mu and Yong Yu, On Indistinguishability in Remote Data Integrity Checking, **The Computer Journal**, Oxford (accepted, 19 Oct. 2013), (online version: doi: 10.1093/comjnl/bxt137)

Outline

- Cloud computing and cloud storage
- Idea of Ateniese et al's PDP
- Hao et al.'s scheme
- Privacy in RIC protocols
- Definition of privacy – Zero Knowledge Privacy
- Our RIC protocols with Zero Knowledge Privacy
- Conclusion

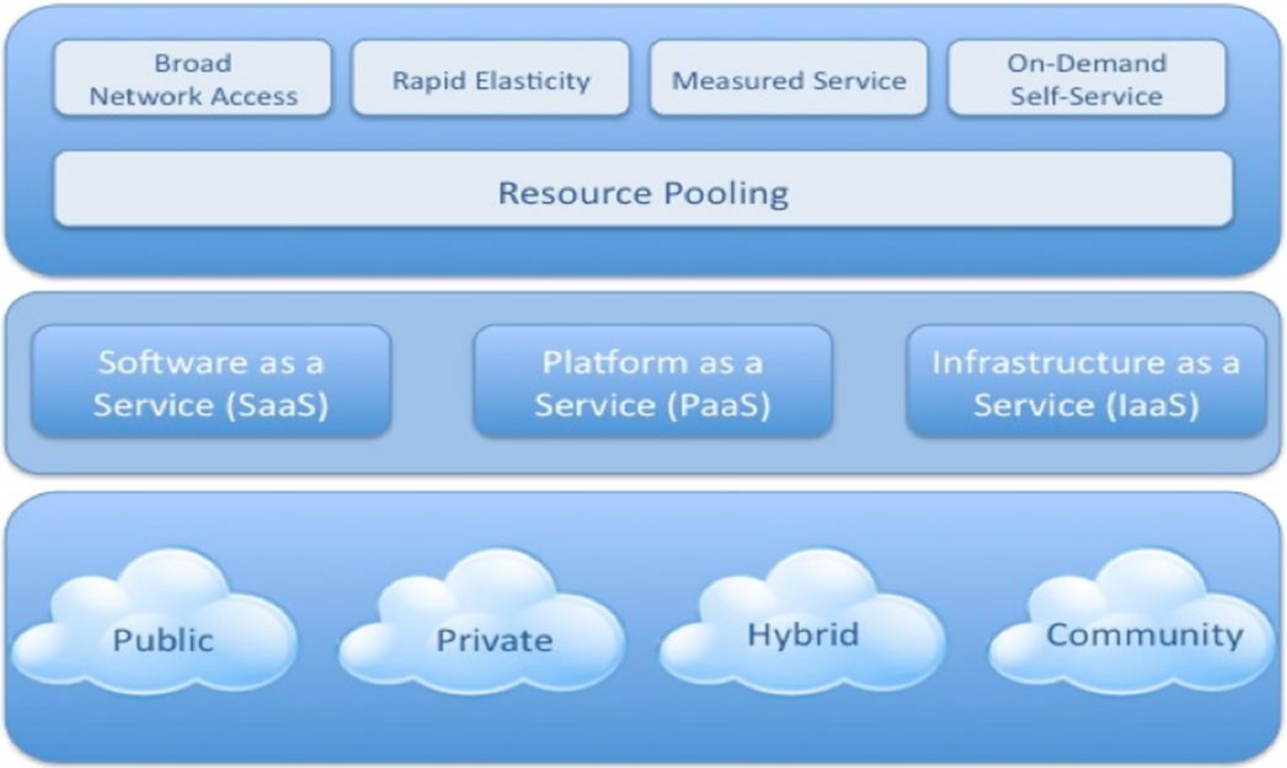
UOW

Cloud Computing: Advantages

– Cloud computing enjoys a "pay-per-use model for enabling available, convenient and on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." – NIST

Cloud Characteristics

Visual Model Of NIST Working Definition Of Cloud Computing
http://www.csrc.nist.gov/groups/SNS/cloud-computing/index.html



Essential Characteristics

Service Models

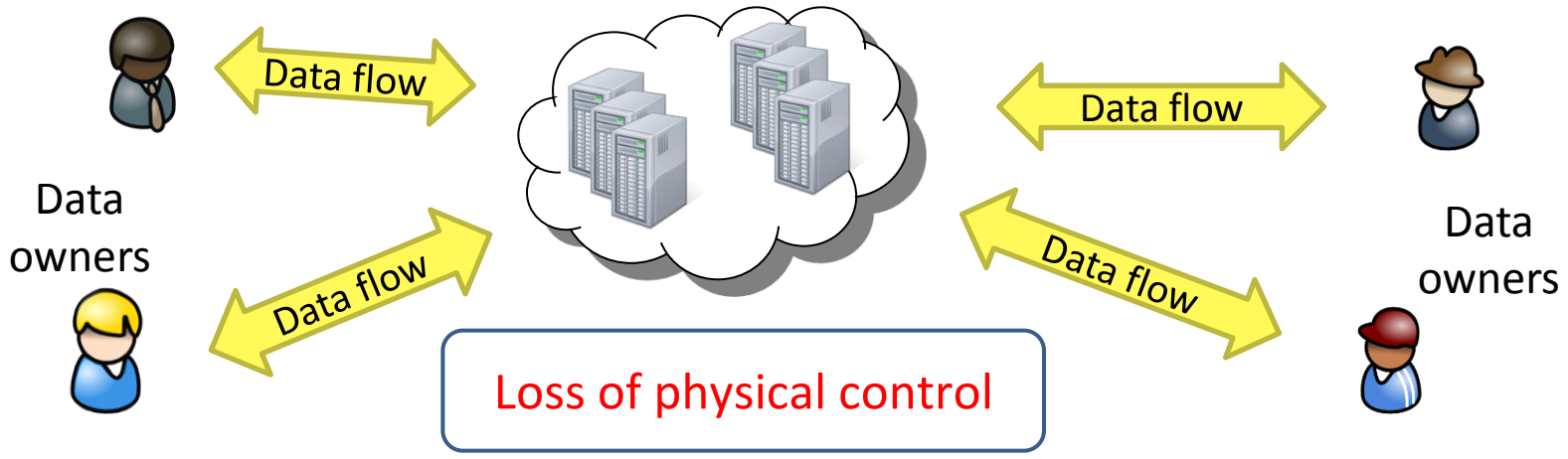
Deployment Models

UOW

Cloud Storage vs. Data Integrity

- Cloud storage service allows owners to outsource their data to cloud servers for storage and maintenance.
  - Low capital costs on hardware and software, low management and maintenance overheads, universal on-demand data access, etc
  - E.g., Amazon S3.

Cloud Storage vs. Data Integrity



- However, data outsourcing also eliminates owners' ultimate control over their data.
- The cloud server is not fully trusted.
  - Try to hide data loss incidents in order to maintain their reputation.
  - Might discard the data that have not been or are rarely accessed for monetary reasons.

Remote Data Integrity Checking

- Demand efficient storage correctness guarantee without requiring local data copies.
  - Traditional methods for integrity can not be directly adopted.
  - Retrieving massive data for checking is unpractical. (large bandwidth)
- Allow meaningful tradeoffs between security and overhead.
  - Communication and computation costs should be low.
  - Integrity checking cost should not outweigh its benefits.

UOW

RSA based PDP (Atenises et al, CCS2007)

- RSA 101
  - N=pq, p=2p'+1, q=2q'+1

$$ed \equiv 1 \bmod \phi(\mathrm{N})$$

  - pk=(e,N)
  - sk=d

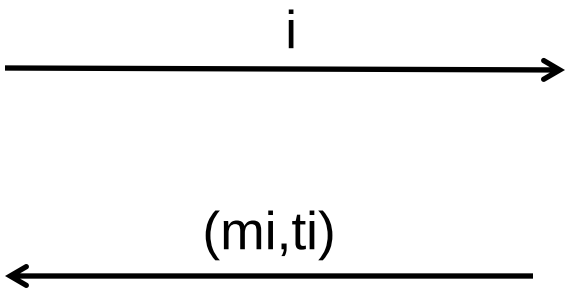
$$Sign : \sigma = H(\mathrm{m})^d \bmod N$$

$$Verify : \sigma^e = H(\mathrm{m}) \bmod N$$

- RSA-based Tag

$$t_i = (\mathrm{H}(\mathrm{W}_i) \cdot \mathrm{g}^{m_i})^d \bmod N$$

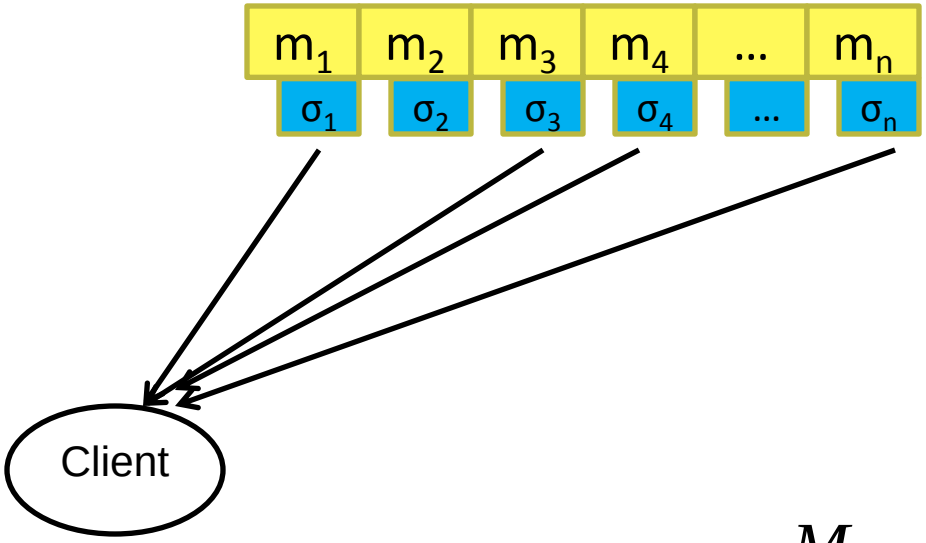
- Single Block



$$t_i = (\mathrm{H}(\mathrm{W}_i) \cdot \mathrm{g}^{m_i})^d \bmod N$$

$$0 \le m_i < e; \frac{t_i^e}{H(\mathrm{W_i})} = g^{m_i}$$

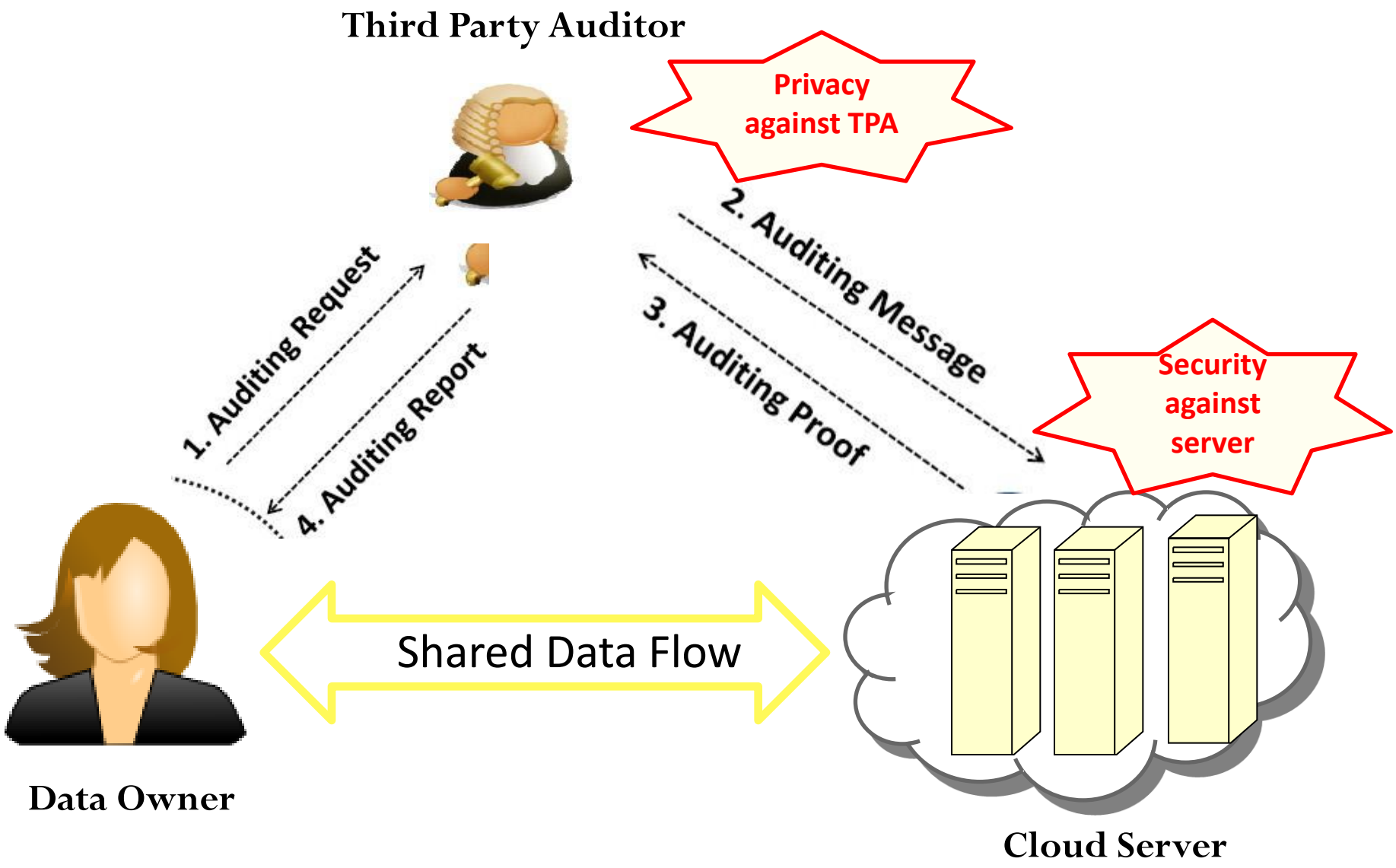
- Challenge-Response



$$T = \sigma_1^{a_1} \sigma_3^{a_3} \sigma_4^{a_4} \sigma_n^{a_n}$$

$$M = a_1 m_1 + a_3 m_3 + a_4 m_4 + a_n m_n$$

$$0 \le M < e; \frac{T^e}{H(\mathrm{W}_1)^{a_1} H(\mathrm{W}_3)^{a_3} H(\mathrm{W}_4)^{a_4} H(\mathrm{W}_n)^{a_n}} = g^M$$



Publicly verifiable RIC

Privacy Analysis of Ateniese et al.'s PDP

$\{1, a_1, 3, a_3, 4, a_4, \mathrm{n}, a_n,\} \rightarrow$

$$T = t_1^{a_1} t_3^{a_3} t_4^{a_4} t_n^{a_n}$$

$$M = a_1 m_1 + a_3 m_3 + a_4 m_4 + a_n m_n$$

(T,M) $\leftarrow$

The adversary can recover by challenging the same blocks many times with different a_i

$$\begin{cases} M_1 = a_{11} m_1 + a_{31} m_3 + a_{41} m_4 + a_{n1} m_n \\ M_2 = a_{12} m_1 + a_{32} m_3 + a_{42} m_4 + a_{n2} m_n \\ M_3 = a_{13} m_1 + a_{33} m_3 + a_{43} m_4 + a_{n3} m_n \\ M_4 = a_{14} m_1 + a_{34} m_3 + a_{44} m_4 + a_{n4} m_n \end{cases}$$

UOW

An improvement due to Hao et al.

– Setup. Let $N = pq$ be a publicly known RSA modulus [25], in which $p = 2p' + 1, q = 2q' + 1$ are large primes. p', q' are two large primes as well. QR_N denotes the multiplicative cyclic group of the quadratic residues modulo N. g, whose order is $p'q'$, is a generator of QR_N. The public key of the data owner is $pk = (N, g)$ while the secret key is $sk = (p, q)$.

– TagGen. For each file block $m_i, i \in [1, n]$, the data owner computes the block tag as

$$D_i = g^{m_i} \pmod{N}.$$

Zhuo Hao, Sheng Zhong, A Privacy-Preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability. IEEE Trans. Knowl. Data Eng. 23(9): 1432-1437 (2011)

An improvement due to Hao et al.

– Challenge. To check the integrity of the file m, the TPA generates a random key $r \in [1, 2^k - 1]$ and a random group element $s \in Z_n \setminus \{0\}$, computes $g_s = g^s \pmod N$ and sends $chal = (r, g_s)$ to the cloud server.

– GenProof. Upon receiving the challenge $chal = (r, g_s)$, the server generates a sequence of block indexes $a_1, \ldots, a_n$ by calling $f_r(i)$ for $i \in [1, n]$ iteratively, computes

$$R = g_s^{\sum_{i=1}^{n} a_i m_i} \pmod N,$$

and sends R to the verifier.

UOW

An improvement due to Hao et al.

– CheckProof. Upon receiving R from the server, the TPA generates $\{a_i\}_{i=1,\ldots,n}$ firstly and then computes P and R' as follows:

$$P = \prod_{i=1}^{n} (D_i^{a_i}) \pmod{N},$$

$$R' = P^s \pmod{N}.$$

If $R = R'$, this algorithm outputs "success" to indicate the data are kept virgin; Otherwise, outputs "failure".

UOW

Privacy Analysis of the Scheme

The verifier can determine whether the client is storing a file block m^* from the public key (N, g) and meta-data $\{D_i\}_{i=1}^{n}$ by evaluating the following equations,

$$D_i \stackrel{?}{=} g^{m^*} \quad \text{for } i = 1 \text{ to } n.$$

Dictionary Attack!!

UOW

Our improvement—System Components

- Setup. On input a security parameter (k), this algorithm generates the public key (pk) and secret key (sk) for the data owner. pk is public to everyone but sk is kept secret by the data owner.
- TagGen. On input the key pair (pk, sk) and a data block (m_i), this algorithm outputs a tag (D_{m_i}) for the block, which will be used for public verification of data integrity.

Our improvement—System Components

- Challenge. TPA generates a challenge *chal* to request for the integrity proof of the file by sending *chal* to the server.
- GenProof. The server computes response R using *chal*, the file and the tags, and returns R to TPA.
- CheckProof. TPA validates response R using *chal*, the tags and public key *pk*. Secret key *sk* is not required in a publicly verifiable data integrity-checking scheme.

Our improvement—Soundness

- Setup. The challenger runs Setup algorithm to generate the public-secret key pair (pk, sk), forwards pk to the adversary and keeps sk secret.
- Query. The adversary adaptively selects some data blocks $m_i (i = 1, \ldots, n)$ and makes tag queries. The challenger computes the corresponding tag $D_i (i = 1, \ldots, n)$ for these blocks and sends them back to the adversary.
- Challenge. The challenger generates a challenge $chal$ and requests the adversary to respond a proof of possession for the challenged blocks.
- Forge. The adversary computes a response R for the data blocks indicated by $chal$ and returns it to the challenger.

UOW

Our improvement—Soundness

We say the adversary wins the game if $CheckProof(pk, chal, D_i(i = 1, \ldots, n), R)$ succeeds. We say that a public remote data integrity-checking scheme is secure against the server if for all polynomial-time adversary that wins the above game, there exists another polynomial-time algorithm Σ, denoted as knowledge extractor, that is capable of extracting the file blocks. The rationale of the model is that for any adversary that can win the game, it can employ Σ to compute the data blocks in polynomial time. In other words, any algorithm that can answer the challenge must be in possession of the underlying file blocks stored in one form or another.

UOW

Our improvement—Zero Knowledge Privacy

- Setup. The challenger runs Setup algorithm to generate the public-secret key pair (pk, sk), forwards pk to the adversary and keeps sk secret.
- Query. The adversary submits two equal-length files, $m_0 = (m_{0,1}, \ldots, m_{0,n})$ and $m_1 = (m_{1,1}, \ldots, m_{1,n})$ to the challenger. The challenger chooses a random bit $b \in_R \{0, 1\}$ and returns

$$D_i \leftarrow \text{tagGen}(m_{b,i}, pk, sk) \quad \text{for } i = 1 \text{ to } n.$$

- Challenge. The adversary sends *chal* to the challenger.
- GenProof. The challenger computes response R using m_b, *chal* and the tags $\{D_i\}$.
- Guess. The adversary outputs a guess bit b'. It wins the game if $b' = b$.

Our improvement—Scheme description

– Setup. Let k, l be two security parameters. $N = pq$ is a public RSA modulus, in which $p = 2p' + 1$, $q = 2q' + 1$ are large primes. p', q' are two large primes as well. QR_N denotes the multiplicative cyclic group of the quadratic residues modulo N. g, h are two generators of QR_N. The public key of the data owner is $pk = (N, g)$ while the secret key is $sk = (p, q)$. A file m is divided into n blocks $m_1, \ldots, m_n$. $H_1, H_2 : \{0,1\}^* \rightarrow Z_N$ are secure cryptographic hash functions. $f : \{0,1\}^k \times \{0,1\}^{\log_2(n)} \rightarrow \{0,1\}^l$ denotes a pseudo-random function and $\pi : \{0,1\}^k \times \{0,1\}^{\log_2(n)} \rightarrow \{0,1\}^{\log_2(n)}$ represents a pseudo-random permutation.

Our improvement—Scheme description

TagGen: $m = m_1 m_2 m_3 \cdots m_n, t \in Z_n$

$$D_i = g^{m_i} h^{H_1(m_i, t)} \pmod{N}$$

stores $(m||t, D_i (1 \leq i \leq n))$ into the server.

Our improvement—Scheme description

TagGen: $m = m_1 m_2 m_3 \cdots m_n, t \in Z_n$

$$D_i = g^{m_i} h^{H_1(m_i, t)} \pmod{N}$$

stores $(m||t, D_i (1 \leq i \leq n))$ into the server.

Ivan Damg ård, Eiichiro Fujisaki: A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order. ASIACRYPT 2002: 125-142

UOW

Our improvement—Scheme description

– Challenge. To check the integrity of file m, the verifier picks a random positive integer c and two keys k_1, k_2 for f and π respectively and sends the challenge $chal = (c, k_1, k_2)$ to the server.

The Verifier		Cloud Server
1. Retrieve file tags D_i and verify its signature. Quit if fail.		
2. Generate a random challenge $chal = \{c, k_1, k_2\}$	$\xrightarrow{chal}$	3. Determine indices i_j and the corresponding coefficients a_j of the challenged blocks.
		4. Compute $A = \sum_{j=1}^{c} a_j m_{i_j}$, $B = \sum_{j=1}^{c} a_j h_j$, $\pi = PK\{(A, B) : (\prod_{j=1}^{c} D_{i_j}^{a_j}) = g^A h^B\}$
5. Check the proof π.	$\xleftarrow{\pi}$	

Our improvement—Scheme description

– GenProof. Upon receiving $chal = (c, k_1, k_2)$, the server computes a response as follows.

1. For $1 \le j \le c$, compute $i_j = \pi_{k_2}(j)$ as the indices of the challenged blocks and computes $a_j = f_{k_1}(j)$ as coefficients.
2. Compute $A = \sum_{j=1}^{c} a_j m_{i_j}$, and $B = \sum_{j=1}^{c} a_j h_j$ where $h_j = H_1(m_{i_j}, t)$.
3. Randomly pick $\rho_1, \rho_2 \in Z_N$, compute $T = g^{\rho_1} h^{\rho_2} \pmod N$, $\xi = H_2(T, chal)$ and $z_1 = \rho_1 - \xi A$, $z_2 = \rho_2 - \xi B$.
4. Send $R = (\xi, z_1, z_2)$ as the response to the verifier.

Check:

$$\xi \stackrel{?}{=} H_2\left(g^{z_1} h^{z_2} \left(\prod_{j=1}^{c} D_{i_j}^{a_j}\right)^{\xi} \pmod N, chal\right).$$

UOW

Our improvement—Soundness Proof

Proof What we are going to prove here is that for any PPT adversary $\mathcal{A}$ who wins the soundness game of some file blocks, there exists a challenger $\mathcal{B}$ that can construct a simulator $\mathcal{S}$ to extract these blocks. Otherwise, the challenger can solve an instance of the factorization problem. $\mathcal{B}$ is given a large integer N, the product of two large primes p and q, and simulates the environment as follows.

UOW

Our improvement—Soundness Proof

- Setup. $\mathcal{B}$ generates two random generators g, h of QR_N and sends $pk = (N, g, h)$ as the public key to the adversary $\mathcal{A}$.
- Queries. $\mathcal{A}$ adaptively selects some blocks m_i and a tag salt t_j to query $\mathcal{B}$ the tags of these blocks. $\mathcal{B}$ computes $h_{ij} = H_1(m_i, t_j)$ and $D_i = g^{m_i} h^{h_{ij}}$, and sends D_i to $\mathcal{A}$.
- Challenge. $\mathcal{B}$ picks an integer c, two keys $k_1, k_2 \in \{0, 1\}^k$ and generates a challenge $chal = (c, k_1, k_2)$ for c file blocks and sends it to $\mathcal{A}$.
- Response. $\mathcal{A}$ generates a response $R = (\xi, z_1, z_2)$ and sends it back to $\mathcal{B}$ as integrity proof of the requested blocks.

UOW

Our improvement—Soundness Proof

If the response can pass the verification,i.e.,

$$\xi = H_2(g^{z_1} h^{z_2} \left(\prod_{j=1}^{c} D_{i_j}^{a_j} \right)^{\xi} \pmod{N}, chal),$$

Using the oracle replay technique and forking lemma, replay H2 to generate a new response R'; then we can get two pairs of collision for H2, we have

$$g^{z_1} h^{z_2} \left(\prod_{j=1}^{c} D_{i_j}^{a_j} \right)^{\xi} = g^{z'_1} h^{z'_2} \left(\prod_{j=1}^{c} D_{i_j}^{a_j} \right)^{\xi'} \pmod{N}.$$

UOW

that is

$$\left(g^{\sum_{j=1}^{c} a_j m_{i_j}} h^{\sum_{j=1}^{x} H_1(m_{i_j}, t) a_j} \right)^{\xi - \xi'} = g^{z_1' - z_1} h^{z_2' - z_2} \pmod{N}.$$

According to the elegant conclusion due to Damgård and Fujisaki [28], $\xi - \xi'$ divides both $z_1' - z_1$ and $z_2' - z_2$ with significant probability. As a consequence, we have

$$g^{\sum_{j=1}^{c} a_j m_{i_j}} h^{\sum_{j=1}^{x} H_1(m_{i_j}, t, i) a_j} = g^{\frac{z_1' - z_1}{\xi - \xi'}} h^{\frac{z_2' - z_2}{\xi - \xi'}} \pmod{N}.$$

Based on the discrete logarithm assumption, $\mathcal{B}$ can extract

$$\sum_{j=1}^{c} a_j m_{i_j} = \frac{z_1' - z_1}{\xi - \xi'}.$$

UOW

$\mathcal{B}$ then generates c challenges $(c, k_1^1, k_2), \ldots, (c, k_1^c, k_2)$ to challenge the same blocks using the approach described above. For each challenge, $\mathcal{B}$ can get an equation of the blocks. $\mathcal{B}$ can select such $k_1^i (1 \leq i \leq c)$ that

$$\begin{vmatrix} a_1^1 & a_2^1 & \cdots & a_c^1 \\ a_1^2 & a_2^2 & \cdots & a_c^2 \\ \vdots & \vdots & \vdots & \vdots \\ a_1^c & a_2^c & \cdots & a_c^c \end{vmatrix} \neq 0$$

where $a_j^i = f_{k_i}(j)(1 \leq i, j \leq c)$.

When the determinant is not zero, the following system of linear equations has a unique solution.

$$\begin{cases} a_1^1 m_{1_j} + a_2^1 m_{2_j} + \cdots + a_c^1 m_{c_j} = \frac{{z_1}^{1'} - z_1^1}{\xi^1 - \xi^{1'}} \pmod{p'q'} \\ a_1^2 m_{1_j} + a_2^2 m_{2_j} + \cdots + a_c^2 m_{c_j} = \frac{{z_1}^{2'} - z_1^2}{\xi^2 - \xi^{2'}} \pmod{p'q'} \\ \vdots \\ a_1^c m_{1_j} + a_2^c m_{2_j} + \cdots + a_c^c m_{c_j} = \frac{{z_1}^{c'} - z_1^c}{\xi^c - \xi^{c'}} \pmod{p'q'} \end{cases}$$

When the determinant is not zero, the following system of linear equations has a unique solution.

$$\begin{cases} a_1^1 m_{1_j} + a_2^1 m_{2_j} + \cdots + a_c^1 m_{c_j} = \frac{{z_1}^{1'} - z_1^1}{\xi^1 - \xi^{1'}} \pmod{p'q'} \\ a_1^2 m_{1_j} + a_2^2 m_{2_j} + \cdots + a_c^2 m_{c_j} = \frac{{z_1}^{2'} - z_1^2}{\xi^2 - \xi^{2'}} \pmod{p'q'} \\ \vdots \\ a_1^c m_{1_j} + a_2^c m_{2_j} + \cdots + a_c^c m_{c_j} = \frac{{z_1}^{c'} - z_1^c}{\xi^c - \xi^{c'}} \pmod{p'q'} \end{cases}$$

Our improvement—ZK privacy proof

In the random oracle model, the new protocol achieves "zero-knowledge privacy". To prove this property, we construct a simulator $\mathcal{S}$ who plays the role of the challenger in the game. For any public key (N, g, h) and system parameters (k, l, H_1, H_2, f, π), $\mathcal{S}$ interacts with the adversary $\mathcal{A}$ as follows.

UOW

Our improvement—ZK privacy proof

- Setup. $\mathcal{S}$ forwards the public key to the adversary.
- Query. The adversary submits two equal-length files, $m_0 = (m_{0,1}, \ldots, m_{0,n})$ and $m_1 = (m_{1,1}, \ldots, m_{1,n})$ to $\mathcal{S}$. For $i = 1$ to n, $\mathcal{S}$ picks $D_i \in_R \mathbb{N}^*$ and returns $\{D_i\}$ to $\mathcal{A}$.
- Challenge. The adversary sends $chal = (c, k_1, k_2)$ as the challenge to $\mathcal{S}$.

Our improvement—ZK privacy proof

– GenProof. $\mathcal{S}$ simulates the response as follows.

1. Pick a random tag-salt $t \in Z_N$ for m and for each block m_i in m, compute

   $$D_i = g^{m_i} h^{H_1(m_i,t)} \pmod{N},$$

   as the tag of m_i.
2. For a challenge (c, k_1, k_2), compute the indices $i_j = \pi_{k_2}(j)$ of the challenged blocks and coefficients $a_j = f_{k_1}(j)$.
3. Pick a random $\xi^* \in Z_n$ and two integers z_1^* and z_2^*, and compute $T^* = g^{z_1^*} h^{z_2^*} (\prod_{j=1}^{c} D_{i_j}^{a_j})^{\xi^*} \pmod{N}$.
4. Take H_2 as a random oracle and set the

   $$H_2(T^*, chal) \rightarrow \xi^*.$$

5. Out $R^* = (\xi^*, z_1^*, z_2^*)$ as the response.

– Guess. The adversary outputs a guess bit b'.

Conclusions

- Cloud computing has posed new challenges to data integrity
- Privacy issues in existing RIC protocols is a big issue.
- Zero Knowledge Privacy was introduced
- RSA based publicly verifiable RIC protocols fails to achieve Zero Knowledge privacy
- An improved scheme with ZK privacy was given

UOW

Thanks to All