Proof of Possession for Cloud Storage via Lagrangian Interpolation Techniques

Łukasz Krzywiecki, Mirosław Kutyłowski

Institute of Mathematics and Computer Science
Wrocław University of Technology

NSS 2012
Wuyi Shan

Cloud storage

Motivation

- keeping data locally might be secure (in theory), but in practice ... all kind of stupid mistakes occur
- it is much cheaper to keep data in a cloud, especially if the business has to adopt quickly

Problem

How do you know that the data uploaded to the cloud is really kept in the cloud storage?

Problem

Typical scenario

- you upload data
- you download data from time to time:
  - just to use it
  - or to check if the cloud is performing what has been promised

If nothing nasty happens over a long time you start to **believe** that the data is really kept in the storage.

Problem

Professional cheaters

- be honest in business over a long time
- invest in your reputation
- . . . until the people really believe you
- then escape with money ...

Vertrauen ist gut, Kontrolle ist besser!

Trust is good, but control is better!

Problem

Professional cheaters

- be honest in business over a long time
- invest in your reputation
- . . . until the people really believe you
- then escape with money ...

Vertrauen ist gut, Kontrolle ist besser!
Trust is good, but control is better!

Problem

Control limitations

What makes the problem difficult

1. the data are uploaded to the cloud gradually at a small rate, nevertheless the volume grows and grows
2. the connection to the cloud is slow, the throughput is too low for downloading the whole data back to the client
3. the client does not keep a copy of data *(the cloud service has to free the client from burden of keeping the data)*
4. the client may have no storage space for fetching back the data

Challenge

how to check that the original data is in the cloud if the client has no copy?

Proof of Possession

Design scenario

Data

- a client retains some small size data S_M corresponding to the data M stored in the cloud
- the cloud stores data M together with tags T_M corresponding to M

Verification

the client may check if cloud stores M by executing verification procedure:

- the client computes a challenge based on S_M
- the cloud server computes an answer using M, T_M and the challenge
- the client checks the answer using the challenge and S_M
  M is not used for verification!

Main requirements

the volume of challenge and the answer should be low

Proof of Possession

Naïve solution

- hash $H(M)$ of data M is retained by the client
- for verification the cloud has to hash the data M stored and to present the result
- the client compares the value received with the hash stored locally

Attack

- the cloud stores $H(M)$ instead of M
- no problem to answer the challenge

Extension

if the client can ask for $H_1(M), H_2(M), \ldots, H_k(M)$, each of them can be used effectively at most once.

Previous work

- different methods to create tags and to check them
- RSA, bilinear mappings, ...
- clever methods to relate the tags and the data so that the cloud server cannot manipulate them and that the only way to answer correctly is just to keep the data

some papers

- *Provable Data Possession at Untrusted Stores*, Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, Dawn Song, ACM CCS, 2007
- *Compact proofs of retrievability*, Hovav Shacham, Brent Waters, ASIACRYPT 2008
- *Towards efficient provable data possession*, Jia Xu, Ee-Chien Chang, ASIACCS, 2012
- . . .

Lagrangian Interpolation in the Exponent

Given

- $L : \mathbb{Z}_q \to \mathbb{Z}_q$ be a polynomial of degree z.
- $A' = \langle (x_0, g^{rL(x_0)}), \ldots, (x_z, g^{rL(x_z)}) \rangle$, where $x_1, \ldots, x_z$ are all different

Interpolation in the Exponent

$$LI_{EXP}(x, A') \overset{\text{def}}{=} \prod_{i=0,(x_i,.)\in A'}^{z} \left(g^{rL(x_i)}\right)^{\prod_{j=0,j\neq i}^{z}\left(\frac{x-x_j}{x_i-x_j}\right)}.$$

Note that

$$LI_{EXP}(x, A') = g^{r\sum_{i=0,(x_i,.)\in A'}^{z}\left(L(x_i)\prod_{j=0,j\neq i}^{z}\left(\frac{x-x_j}{x_i-x_j}\right)\right)} = g^{rL(x)} .$$

Interpolation in the Exponent

basic application

Revocation scheme

- a secret polynomial L of degree z
- a user i holds $(i, L(i))$
- in order to broadcast a key K to all users but $i_1, \ldots, i_z$, the broadcaster makes a header

$$g^r, \quad (i_1, g^{rL(i_1)}), \ldots, (i_k, g^{rL(i_z)}), \quad x_0, K \cdot g^{rL(x_0)}$$

- a user i such that $i \notin \{i_1, \ldots, i_z\}$, uses Lagrangian Interpolation in the exponent to compute $g^{rL(x_0)}$ and thereby derive encryption key K
- ability to derive $g^{rL(i)}$

Security of the scheme is based on the fact that ability to derive $(g^r)^{L(x)}$ for any x does not help to derive $(g^{r'})^{L(x)}$ if r, r' are given by $g^r, g^{r'}$ only.

Scheme based on LI for a single block

setting

A block

- block i of m consists of z elements from $\mathbb{Z}_p$, say

$$m_i = m_{i,1} \dots m_{i,z}, \qquad m_{i,1} \dots m_{i,z} \in \mathbb{Z}_p$$

Parameters

- SK – master secret key of the user,
- filename ID_f, block number i

Block polynomial

- SK, ID_f, i (and may be some other parameters like block version) used to compute
  - a seed s_1
  - a polynomial value t_0
- $t_1, \dots, t_z$ obtained via a PRNG from s_1
- secret polynomial of degree z:
  $L_i(x)$ that has values $t_0, t_1, \dots, t_z$ at points $0, m_{i,1}, \dots, m_{i,z}$

Proof of Possession via Lagrangian Interpolation

Krzywiecki, Kutyłowski

Introduction

Previous work

Preliminaries

Scheme

Comparison

Scheme based on LI for a single block

creating tags for a block

Data computed by the cloud client

- compute a value $t_c = L_i(x_c)$
- g^{t_c}, and its ciphertext $E(g^{t_c})$ computed with a secret key of the client

Data stored in a cloud

- data $m_{i,1}, \dots, m_{i,z}$
- tags:
  - s_1
    (for reconstruction of $t_1, \dots, t_z$)
  - $E(g^{t_c})$
    (to be returned to the client at the time of Proof of Possession)

Scheme based on LI for a single block

Proof of Possession

Computing a challenge by a client

- choose r at random
- compute g^r, $g^{rL(0)}$ and send to the cloud

Responding a challenge by a cloud

- recompute $t_{i,1}, \ldots, t_{i,z}$
- compute $(g^r)^{t_{i,1}}, \ldots, (g^r)^{t_{i,z}}$
  (they are equal to $g^{rL_i(m_{i,1})}, \ldots, g^{rL_i(m_{i,z})}$)
- use LI in the exponent to compute $g^{rL(x_c)}$
  ($m_{i,1}, \ldots, m_{i,z}$ must be used)
- return $g^{rL(x_c)}$ and $E(g^{t_c})$

Verification of the answer

- decrypt $E(g^{t_c})$
- compute $(g^{t_c})^r$ and check if the result is the same as the value $g^{rL(x_c)}$ returned by the cloud

Scheme based on LI for a single block

Intuition

Necessity to use m_i

- for LI the cloud needs to know all $m_{i,j}$
  they explicitly appear in the formulas
- even if each value t_j of L_i is easy to determine, using it for a different argument than $m_{i,j}$ via LI provides a wrong result – this is a basic property of polynomials

Possibility to reuse answers

- the response for g^r does not help to create the answer for $g^{r'}$

  otherwise we would have a way to solve the problem of equality of discrete logarithms

Security

formal approach

Computational Diffie-Hellman

input g^a and g^b, compute g^{ab}

Model

- we assume that the attacker cloud forgets $m_{i,1}$ but has a way to answer a challenge correctly
- the memory of the cloud may include all answers shown so far

Proof strategy

- use the attack to solve CDH Problem
- challenge: create a virtual situation for the attack

Security

using the attack to solve CDH

Parameters

- g^a will play the role of the challenge g^r
- g^b plays the role of $g^{L(m_{i,1})}$
- the values $L(0)$, $L(m_{i,2}), \ldots, L(m_{i,z})$ given explicitly

Creating history

- choose r_k at random
- compute the values of L in the exponent: $(g^b)^{r_k}$ and $g^{r_k L(0)}$, $g^{r_k L(m_{i,2})}, \ldots,$ $g^{r_k L(m_{i,z})}$
- use LI in the exponent to compute $g^{r_k L(x_c)}$

Exploiting faked answer

- let the attack deliver $g^{aL(x_c)}$ for challenge g^a
- compute directly $(g^a)^{L(0)}$, $(g^a)^{L(m_{i,2})}, \ldots, (g^a)^{L(m_{i,z})}$
- use LI in the exponent to compute $(g^a)^{L(m_{i,1})} = g^{ab}$

Checking many blocks at once

It is possible:

- for each block use a polynomial with the same fixed value at 0
- therefore the same challenge serves for all polynomials
- for the answer: multiply the results from the blocks

Computational efficiency

Comparison with Xu, J., Chang, E.C.: “Towards efficient provable data possession” IACR ePrint 574 (2011) 574, ASIACCS, 2012

space overhead - cloud server

XCh :1 group element per block and z group elements per client

LI : 1 group element per block and 1 secret per client

space overhead - client

XCh,LI: independent of the number of blocks

Computational efficiency

creating tags

XCh, LI: no exponentiations

challenge & verification

XCh : 2 exponentiations in total

LI : 3 exponentiations per block

creating a proof

XCh : $z - 1$ exponentiations in total,

LI : $z + 1$ exponentiations per block

Modifying a file for Xu & Cheng scheme

Construction of a tag:

$$t_i := \mathtt{PRF}_{\mathtt{seed}}(\mathtt{id}, i) + \tau \sum_{j=0}^{s-1} m_{i,j} \cdot \alpha^j \bmod p$$

where $\mathtt{seed}$ and α are secrets

Attack by re-computation of a tag:

step 1: make the user to encode the file once more, for $m'_{i,j} = m_{i,j}$ except for $m'_{1,1} = m_{1,1} + 1 \bmod p$

step 2: the server derives α based on the observation that the new tag t'_1 equals $t_i + \alpha$

step 3: the server can modify the files: without changing t_i it can change: $m_{i,j} := m_{i,j} + \delta$, $m_{i,j'} := m_{i,j'} - \delta/(\alpha^{j'-j}) \bmod p$. Indeed:

$$\begin{array}{lclcl} \ldots + & m_{i,j} \cdot \alpha^j & + \ldots + & m_{i,j'} \cdot \alpha^{j'} & + \ldots = \\ \ldots + & (m_{i,j} + \delta) \cdot \alpha^j & + \ldots + & (m_{i,j'} - \delta/\alpha^{j'-j}) \cdot \alpha^{j'} & + \ldots \end{array}$$

Modifying a file for Lagrangian Interpolation scheme

- on each block the polynomial is chosen independently
- after a modification the polynomial within the block must be chosen anew,
  otherwise the cloud server can replace all blocks via Lagrangian interpolation (in the exponent)

Thanks

Thank You