Ramp secret sharing and secure information storage

Laszlo Csirmaz

Central European University
Renyi Institute

November 11–14, 2009

Contents

1 Secure Information Storage – the problem

2 Integredients

3 Encryption system: tweaking a block cipher

4 Secret sharing schemes

5 Open problems

How to store information

Basic requirements:

1. Diversity: don't rely on a single service
2. Security: never trust any third party, don't store data on the clear
3. Cost-effectiveness: minimize the total amount of distributed data

How to store information

Basic requirements:

1. Diversity: don't rely on a single service
2. Security: never trust any third party, don't store data on the clear
3. Cost-effectiveness: minimize the total amount of distributed data

Scenarios:

1. Remote: store data at several warehouses for security
2. On site: hot swappable hard drive cluster

Contents

1 Secure Information Storage – the problem

2 Integredients

3 Encryption system: tweaking a block cipher

4 Secret sharing schemes

5 Open problems

Secret Sharing

Shamir's threshold scheme

- Secret sharing: n participants (servers, storage units) hold shares of a secret (chunk of data)
- Accessibility: any k can recover the secret
- Perfect secrecy: $k-1$ shares do not release any information about the secret
- Ramp scheme: participants might gain some info

Secret Sharing

Shamir's threshold scheme

- Secret sharing: n participants (servers, storage units) hold shares of a secret (chunk of data)
- Accessibility: any k can recover the secret
- Perfect secrecy: $k-1$ shares do not release any information about the secret
- Ramp scheme: participants might gain some info

Other schemes

- Linear codes – span program
- Geometric constructions
- Elliptic curves

Encryption System

Requirements for encryption:

- Encrypt data before distribution
- Encryption should allow *random access*
- Use standard block ciphers (AES, Blowfish, 3DES, etc.)

Encryption System

Requirements for encryption:

- Encrypt data before distribution
- Encryption should allow *random access*
- Use standard block ciphers (AES, Blowfish, 3DES, etc.)

Therefore:

- Use XTS tweakable mode (Rogaway) 2004
- approved as IEEE 1619 standard for *cryptographic protection of data on block-oriented storage devices* 2007

Contents

1 Secure Information Storage – the problem

2 Integredients

3 Encryption system: tweaking a block cipher

4 Secret sharing schemes

5 Open problems

Tweaking a cipher

Data:

- data is split into *chunks*
- a chunk is split into *blocks*, blocks are accessed incrementally

Encryption:

- $E_k(m)$ is a secure block cipher with block lenght n
- $K1$, $K2$ are the *primary* and *secondary* keys
- a is a primitive element in $\mathbb{F}(2^n)$
- N is the *physical address* of the chunk
- the *chunk key* L is created as $L = E_{K2}(N)$
- the *i-th block key* is $\Delta_i = a^i L$ computed in $\mathbb{F}(2^n)$
- the i-th block is encypted as $C_i = E_{K1}(M_i \oplus \Delta_i) \oplus \Delta_i$

Properties of encryption

Efficiency

- one extra encryption for the chunk
- one multiplication in $\mathbb{F}(2^n)$ for each block when accessed incrementally

Security

- **if** E_K is resistent to chosen cipertext attack (CCA-secure)
- **then** this scheme is resistent as well (Rogaway 2004)

Contents

1 Secure Information Storage – the problem

2 Integredients

3 Encryption system: tweaking a block cipher

4 Secret sharing schemes

5 Open problems

Use it for . . .

Encryption keys

(You should not forget them, right?

- *security* — ESSENTIAL
- *efficiency* — does not matter, it is small compared to the data

Bulk data

- *security* — encryption takes care of it
- *efficiency* — ESSENTIAL

Use it for . . .

Encryption keys

(You should not forget them, right?

- *security* — ESSENTIAL
- *efficiency* — does not matter, it is small compared to the data

Bulk data

- *security* — encryption takes care of it
- *efficiency* — ESSENTIAL

Use a scheme tailored to the task

Shamir's Secret Sharing

- take the secret $s \in \mathbb{F}$
- give different non-zero labels from $\mathbb{F}$ to the participants
- pick $k-1$ random elements $a_1, \ldots, a_{k-1}$ from the field
- define the polynomial

$$p(x) = s + a_1 x + \cdots + a_{k-1} x^{k-1}$$

- give participant with label b the share $p(b)$

Shamir's Secret Sharing – secret recovery

Do it with *Lagrange interpolation*

- participants with labels $b_1 \dots b_k$ submit their values $v_1 \dots v_k$.
- define the polynomial $p_i(x)$ which takes zero at all b_j's except at b_i where is takes 1:

$$p_i(x) = \frac{(x-b_1)\dots(x-b_{i-1})(x-b_{i+1})\dots(x-b_k)}{(b_i-b_1)\dots(b_i-b_{i-1})(b_i-b_{i+1})\dots(b_i-b_k)}$$

- recover the polynomial $p(x)$ as

$$p(x) = v_1p_1(x) + v_2p_2(x) + \dots + v_kp_k(x)$$

(indeed, it has value v_i at b_i and has degree $\le k$)

- recover the secret as

$$\text{secret} = p(0) = v_1p_1(0) + v_2p_2(0) + \dots + v_kp_k(0)$$

Shamir's Secret Sharing – secret recovery, con't

Efficiency

- no need to compute $p_i(x)$, only the value $p_i(0)$.
- $p_1(0), \ldots, p_k(0)$ are field elements which can be precomputed beforehand
- the secret is a *linear combination* of the shares with predetermined coefficients from the field $\mathbb{F}$.
- the share is LARGE – equal to the size of the secret itself

Secrecy

- the BEST possible:
  even $k-1$ shares do not leak any information about the secret

Shamir's Secret Sharing – summary

Good

- for storing and distributing encryption keys

Bad

- for storing bulk data

Can we improve on data storage requirements?

The solution: ramp scheme

Y E S !

Use ramp sheme: relax on security and gain on efficiency

Shamir's Ramp Secret Sharing

Method

use the whole polynomial as the secret

Advantage

the secret is k times as long as the shares

Disatvantage

even a single share reveals something about the secret

Shamir's Ramp Secret Sharing

Method

use the whole polynomial as the secret

Advantage

the secret is k times as long as the shares

Disatvantage

even a single share reveals something about the secret
$\longrightarrow$ no problem if the data is encrypted beforehand!

How does it work?

Data distribution

- take the next k values $a_0, a_1, \ldots, a_{k-1}$ from the data stream as elements of $\mathbb{F}$
- define the polynomial
$$p(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1}$$
- give participant with label b the share $p(b)$

Data recovery

- collect k shares $v_1 \ldots v_k$ from participants with labels $b_1 \ldots b_k$
- recover the polynomial $p(x)$ using Lagrange interpolation

Properties

- each coefficient a_i requires a (predetermined) linear combination of the shares over $\mathbb{F}$
- any other share can be conputed without recovering the polynomial (restoring the content of a corrupted server)
- computation is over a finite field $\mathbb{F}$ – might be slow

Contents

1 Secure Information Storage – the problem

2 Integredients

3 Encryption system: tweaking a block cipher

4 Secret sharing schemes

5 Open problems

Flexibility?

We understand perfect secret sharing, but no so well ramp schemes

- how to balance load among participants (servers), if one is capable more work (higher capacity, faster, etc.) than the others?
- how to involve pricing constraints, how to minimize the total cost?
- what can be done if we do not want some servers to gain information even on the encrypted data? How does this distroys the efficiency?
- how to utilize other secret sharing methods? Can those methods be better both in efficiency and flexibility?

G. R. Blakley.
Safeguarding cryptographic keys
Proc.NCC AFIPS 1979, pp. 313–317

L. Csirmaz.
Secret sharing schemes: Solved and unsolved problems
8th Central European Conference on Cryptography Graz 2008
Available from http://www.renyi.hu/_csirmaz/talk08.pdf

H. Krawczyk.
Secret sharing made short,
CRYPTO'93, LNCS 773 (1993), pp. 136–146

P. Rogaway.
Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC,
ASIACRYPT'04, LNCS 3329 (2004) pp. 16–31

A. Shamir.
How to share a secret.
Commm. of ACM, volume 22 (1979), no 11, pp. 612–613

J. J.Wylie, M. W. Bigrigg, J. D. Strunk, G. R. Ganger, H. Killccote, P. K. Khoisa.
Survivable Information Storage Systems
IEEE Computer, volume 33 (2000), no 8, pp 61-68