How to Construct State Registries Matching Undeniability with Public Security

Mirosław Kutyłowski
joint work with Przemysław Kubiak and Jun Shao*

Wrocław University of Technology
Pennsylvania State University*

ACIIDS-2010, Hue, 24.03.2010

State Registry for Personal Information
purpose

Reference database for e-ID

1. official source of basic personal data (birth date, parents, citizenship, issued ID documents)
2. accessible online for checking validity of these data

Purpose

1. high quality reference data that can be assumed to be true in the legal sense,
2. source of necessary data for other e-government systems,

Security Requirements

Requirements

1. each single (digital) record must be authenticated in a strong way
2. adding new records possible only through appending them to the database
3. corrections of old records only by adding correcting records

Cryptographic tools
Hash functions, chains

Cryptographic hash function H

- computing $H(x)$ for a given x is easy
- finding an x such that $H(x) = y$ for a given y is infeasible
- finding $x_1 \neq x_2$ such that $H(x_1) = H(x_2)$ is infeasible

Examples: SHA-256, RIPEMD, ...

Cryptographic tools
Hash functions, chains

Cryptographic hash function H

- finding $x_1 \neq x_2$ such that $H(x_1) = H(x_2)$ is infeasible

Hash chain

- given records $m_1, m_2, \ldots, m_k$ to be linked
- we compute the values H_i according to the formula

$$H_{i+1} = H(H_i, m_{i+1}) \text{ for } i < k$$

so we construct:

$$H_1 := H(IV, m_1),\ H_2 := H(H_1, m_2),\ H_3 := H(H_2, m_3), \ldots$$

- it is impossible to remove, add or modify a record without changing H_k

Cryptographic tools
Merkle tree

Merkle tree

1. a labeled tree
2. the leaves are labeled with data items $m_1, \ldots, m_k$
3. label $L(a)$ of a node a having children b, c in the tree is computed as

$$L(a) := H\big(L(a), L(b)\big)$$

4. label of the root is a fingerprint of all values in the leaves
5. for proving that a label is in some leaf of a tree with label h in the root: it is enough to show some hashes from the tree (an easy reconstruction)

Architecture based on Merkle trees

System architecture

1. form a Merkle tree from the records of one day
2. keep linking the roots of the Merkle trees in a single hash chain
3. leave physical traces: print, sign (traditionally) and store safely the root values,
   publish the root values each day in a newspaper

Features

1. a digital evidence for existence in the database: data for reconstructing the values on the path from a leaf to the root of some Merkle tree,
2. the trees need not to be published, only their roots! (automatic personal data protection)

Problems

The security requirements are in fact different:

1. in certain situations it is necessary to create in the past some records of the registry
2. creation of new identities for:
   - witness protection programs
   - creating identities for agents of security authorities
   - ...

Merkle trees are not well suited:

1. strong properties of the tree prevents creation of ID's by security agencies
2. agent ID's would have to be created in advance.

Our solution
actors

Registrar

1. Registrar is an authorized public body
2. Registrar can create entries in the registry only in the “append” mode only
3. no entry can be removed or modified after insertion so that it remains undetected

Our solution
actors

Security Agency

4. Security Agency has possibility to break the rules 1-2 and insert additional entries with past date
5. it is impossible to distinguish the entries created according to rule 4 from the regular entries, even with private keys used to create the entries
6. another authority, called Supervisor, has extra private keys and using them may reveal if a given entry in the database has been created by Registrar or by Security Agency

Cryptographic building blocks
hash function

Trapdoor hash function

1. H is one-way, collision resistant function: it is infeasible to find *any* $(x, s) \neq (x', s')$ such that $H(x, s) = H(x', s')$
2. there is a secret trapdoor $\mathcal{S}$, so that given $\bar{z}$, $\bar{s}$, and the trapdoor secret $\mathcal{S}$ one can find $\bar{x}$ such that $H(\bar{x}, \bar{s}) = \bar{z}$

Example

Let E be encryption with a a public key. Let

$$H(x, s) = E(E(x) \text{ xor } s)$$

- with a decryption function and a signature s it is easy to find a value x such that $H(x, s) = z$
- inverting H would mean breaking E: given a ciphertext c, find x, s such that $D(c) = E(x) \text{ xor } s$
- a collision for H would mean finding x' such that $E(x) \text{ xor } E(x') = s \text{ xor } s'$. s and s' must be signatures, so one has to find a pair of plaintexts yielding a given difference of ciphertexts

Cryptographic building blocks
group signatures

Requirements

1. an upper bound on the number of group members (for instance 2)
2. the group manager cannot become a group member
3. the group manager can prove that a signature was created by a given person with a zero knowledge proof (so that it is not transferable)
4. a group member cannot prove to a third party that a given signature has been created by himself (or somebody else)

Cryptographic building blocks
Verifiable randomness

Verifying random strings for randomness

If Alice wishes to determine a “random value”, then

- she chooses a random value x,
- she computes an undeniable signature $\tilde{s}$ of x with designated verifier Bob. The underlying designated signature scheme should be non-delegateable.

Creating Merkle tree by Registrar
Registrar

Creating a Merkle tree by Registrar

1. for the entries $m_1, \ldots, m_k$ created during day t Registrar creates signatures $s_1, \ldots, s_k$ using the key K_G
2. Registrar chooses $x_1, \ldots, x_k$ *at random*, then for $i \leq k$ computes $y_i = H(x_i, s_i)$, the values x_i, s_i get stored together with m_i in the database
3. for $k < j \leq L$ Registrar creates pseudo-random values y_j using a key K_U

Creating Merkle tree by Registrar
Registrar

Creating the Merkle tree by Registrar

1. Registrar contacts Security Agency , then:
   - Registrar shows $y_{k+1}, \ldots, y_L$ and performs together with Security Agency the verification procedure, additionally, for each y_i Registrar presents the hash proof p_i,
   - Registrar shows $x_1, \ldots, x_k$ and performs together with Security Agency verification procedure, additionally, Registrar also shows to Security Agency corresponding signatures $s_1, \ldots, s_k$, to prove that $x_1, \ldots, x_k$ were really used to create leaves,
2. Registrar creates a hash tree with the leaves $y_1, \ldots, y_L$
3. Registrar signs the root and archives it,
4. for each m_i Registrar creates a hash tree proof p_i and sends the authentication data to the entitled person(s),

Creating entries by Security Agency

Inserting a fake record

1. Security Agency chooses some y that has been shown by Registrar and proved as pseudo-random value not corresponding to any real entry,
2. Security Agency creates a signature s of m using the key $\bar{K}_G$ and the group signature scheme,
3. Security Agency uses the trapdoor K_H to find x such that $y = H(x, s)$.

Summary

Properties

1. a strong cryptographic proof that a record is in the registry
2. only append operation
3. also insert operation for special user
4. a supervisor can check who created a given record...
5. but the proof is non-transferable

the technique can be extended

Current work

implementation as a “proof of concept”
choice of cryptographic primitives - fine tuning the algorithms to specific needs

Thanks for your attention!

Contact data

1. Miroslaw.Kutylowski@pwr.wroc.pl
2. http://kutylowski.im.pwr.wroc.pl
3. +48 71 3202109, fax: +48 71 320 2105