Tracing Attacks on U-Prove with Revocation Mechanism

Lucjan Hanzlik, Przemysław Kubiak, Mirosław Kutyłowski

Wrocław University of Technology, Poland

ACM ASIA CCS 2015, Singapore

Privacy/Personal Data Protection/Anonymity

principle of minimal information disclosure

traditional systems:

1. user identification
2. user authentication
3. granting rights

security problem:

- identification+authentication **secures proper assignment of rights**
- but... reveals a lot of data that can be used by a malicious parties

too much information is a security threat [a]

[a] like writing PIN on an ATM card

Privacy/Personal Data Protection/Anonymity

anonymous credentials idea

anonymous credentials idea:

1. presenting anonymous credentials
2. granting rights

anonymous credential:

- presents chosen attributes of the user
- presents a proof that a trusted party has confirmed these attributes for this user without **revealing identity of the person**

Anonymous Credential System

Issuing a witness

- user interacts with a Trusted Party who can verify the attributes
- the user gets a **witness** – cryptographic data that is used to create credentials

Presenting credentials

- a user takes the witness, and chooses a subset of attributes
- creates a **credential** on (a formula about) selected attributes

Credentials verification

a verifier takes the credentials and checks that

- the attributes have been confirmed by the Trusted Party
- the formula on the attributes holds

Anonymous Credential System
major products

Products

UProve: Microsoft, based on Brands idea

Idemix: IBM, mainly based on Camenisch, Lysyanskaya techniques

Problems

- relatively “heavy” computations
- hard to understand
- not really suited for smart cards, only some subprocedures in a secure environment of smart cards
- application scope?

many versions of anonymization possible: a similar product is **Restricted Identification** on German personal identity cards

Revocation Scheme

Problem

- a user may loose some attributes
- strong anonymity may enable the user to use the outdated credentials – **we have to prevent this, otherwise the system is useless for most practical applications**

Revocation Scheme

extra functionality:

- the user proves that his attributes have not been revoked by the Revocation Authority

Revocation Challenges

Principles and requirements

- revocation must not reveal identity
- revocation is not a penalty - user's privacy has to be protected

Accumulator concept

traditional approach: revocation list

a list of all revoked users

Cryptographic accumulator

a single value, such that

- one can **put** a value into the accumulator
- one can prove that **a given value is not in the accumulator**

Presenting credentials with a non-revocation proof

a user creates a proof that

1. the chosen attributes have been confirmed by the Trusted Party
2. they have not been inserted into the accumulator (the current accumulator value is used)

The accumulator is computed and published by a Revocation Authority.

General concept for UProve
FC'2013

Paper:

Acar, T., Chow, S.S.M., Nguyen, L.: Accumulators and U-Prove Revocation. Financial Cryptography. LNCS 7859 (2013)

Strategy

- on-top of Uprove as a "plug-in"
- it uses accumulator concept
- it reuses the standard mechanisms of anonymous credentials schemes

Problems
FC'2013

Problems

1. bilinear mappings used, but standard embedded devices do not support bilinear groups
2. complicated
3. no formal proof – flaws might exists [a]

[a] like the one indicated by our FC 2014 paper (already corrected by the authors)

UProve Revocation -Microsoft
general assumptions

Main features

- generally the same approach
- however: the bilinear mappings are **eliminated**[a]
- simplifications
- published in technical drafts, versions change
- no security proof, no motivation/justification at all
- support/ good conditions for creating third party products based on UProve

[a] this leads to problems

UProve Revocation

Procedure `RSSetup()`

Initialization of the system

Input:	
U-Prove parameters:	group G_q of a prime order q generators g, g_1, g_t
Computation:	
	choose $\delta \in \mathbb{Z}_q$ at random $K := g^{\delta}$
Output:	private key δ, public key K

UProve Revocation

Procedure `ComputeAccumulator()`

Computing the accumulator V of revoked users

Input:	
RA private key:	$\delta \in \mathbb{Z}_q$
Revocation parameter:	g_t
Set of revoked attribute values:	$R = \{x_1, \ldots, x_m\} \subseteq \mathbb{Z}_q \backslash \{-\delta\}$
Computation:	
	$V := g_t^{\prod_{i=1}^{m}(\delta + x_i)}$
Output: accumulator value V	

UProve Revocation

Procedure `ComputeWitness()`

Creating a witness by the system

Input:	
RA private key:	$\delta \in \mathbb{Z}_q$
Revocation parameter:	g_t
List of revoked attribute values:	$R=\{x_1, \ldots, x_m\} \in \mathbb{Z}_q \setminus \{-\delta\}$
Target user's revocation attribute:	$x_{id} \notin R$
Current accumulator:	$V \in G_q$
Computation:	
$d := \prod_{x \in R}(x - x_{id}) \bmod q$	
$W := g_t^{(\prod_{x \in R}(\delta + x) - d)/(\delta + x_{id})}$	
$Q := VW^{-x_{id}} g_t^{-d}$	
Output:	
Revocation witness for target user holding x_{id}:	(d, W, Q)

UProve Revocation

Procedure `UpdateWitness( )` - updating the witness by the user himself

Input:	
Revocation parameter:	$g_t \in G_q$
The revocation attribute of the user:	x_{id}
Revocation attribute to be added or removed from R:	x'
Boolean value indicating whether x' has to be added to R:	`add`
Old accumulator:	$V \in G_q$
Old witness of the user holding x_{id}:	(d, W, Q)
Updated accumulator:	$V' \in G_q$
Computation: if `add` = `true`	(x' added to R)
$d' := d(x' - x_{id}) \bmod q$	
$W' := VW^{x'-x_{id}}$	
$Q' := V'W'^{-x_{id}} g_t^{-d'}$	
else	(x' removed from R)
...	
Output: updated witness (d', W', Q') for x_{id}	

UProve Revocation

Procedure `UpdateWitness( )` — updating the witness by the user himself

Input:	
Revocation parameter:	$g_t \in G_q$
The revocation attribute of the user:	x_{id}
Revocation attribute to be added or removed from R:	x'
Boolean value indicating whether x' has to be added to R:	`add`
Old accumulator:	$V \in G_q$
Old witness of the user holding x_{id}:	(d, W, Q)
Updated accumulator:	$V' \in G_q$
Computation: if `add` = `true`	(x' added to R)
...	
else	(x' removed from R)
$d' := d(x' - x_{id})^{-1} \bmod q$	
$W' := ((V')^{-1} W)^{(x' - x_{id})^{-1}}$	
$Q' := V' W'^{-x_{id}} g_t^{-d'}$	
Output:	updated witness (d', W', Q') for x_{id}

UProve Revocation

Procedure `GenerateNonRevocationProof()` - a user proves to be not revoked

Input:

Revocation parameters:	G_q, hash function $\mathcal{H}$, g, g_1, g_t
Commitment to x_{id}:	$\tilde{c}_{id}$, where $\tilde{c}_{id} = g^{x_{id}} g_1^{\tilde{o}_{id}}$
Opening information:	x_{id} , $\tilde{o}_{id}$
RA public key:	K
Revocation witness:	(d, W, Q)

Computation:

generate $t_1, t_2, k_1, \ldots, k_6 \in \mathbb{Z}_q$ at random

$X := Wg^{t_1}$ $\quad$ $Y := QK^{t_1}$

$C_d := g_t^d g_1^{t_2}$ $\quad$ $w := d^{-1} \bmod q$

$z := t_1 \tilde{o}_{id} - t_2 \bmod q$ $\quad$ $z' := -t_2 w \bmod q$

$T_1 := X^{k_1} (\tilde{c}_{id} K)^{-k_2} g_1^{k_3}$, $\quad$ $T_2 := g^{k_1} g_1^{k_4}$, $T_3 := C_d^{k_5} g_1^{k_6}$

$c' := \mathcal{H}(g, g_1, g_t, K, \tilde{c}_{id}, X, Y, C_d, T_1, T_2, T_3)$

$s_1 := -c' x_{id} + k_1 \bmod q$ $\quad$ $s_2 := -c' t_1 + k_2 \bmod q$

$s_3 := -c' z + k_3 \bmod q$ $\quad$ $s_4 := -c' \tilde{o}_{id} + k_4 \bmod q$

$s_5 := -c' w + k_5 \bmod q$ $\quad$ $s_6 := -c' z' + k_6 \bmod q$

delete $t_1, t_2, k_1, ..., k_6, w, z, z', T_1, T_2, T_3$

Output:

non-revocation proof for x_{id}:	$(c', s_1, \ldots, s_6, X, Y, C_d)$

UProve Revocation

Procedure `VerifyNonRevocationProof()`

verification of the `NonRevocationProof`

Input:

Revocation parameters:	$G_q, \mathcal{H}, g, g_1, g_t$
Commitment to x_{id}:	$\tilde{c}_{id}$
Non-revocation proof:	$c', s_1, \ldots, s_6, X, Y, C_d$
Revocation Authority public key:	K
Revocation Authority private key:	δ
Revocation accumulator:	V

Computation:

$T_1 := (VY^{-1}(C_d)^{-1})^{c'} X^{s_1} (\tilde{c}_{id} K)^{-s_2} g_1^{s_3}$

$T_2 := \tilde{c}_{id}^{c'} \, g^{s_1} g_1^{s_4}$

$T_3 := g_t^{c'} (C_d)^{s_5} g_1^{s_6}$

verify that $c' = \mathcal{H}(g, g_1, g_t, K, \tilde{c}_{id}, X, Y, C_d, T_1, T_2, T_3)$

verify that $Y = X^{\delta}$

UProve Revocation

Procedure `VerifyNonRevocationProof()`

problems with verification of $Y = X^{\delta}$

- δ is known neither to the prover not to the verifier so it cannot be checked directly
- $K = g^{\delta}$ as well, so is there is a workaround, then we can solve the equality of discrete logarithms problem.
- for bilinear groups it would be easy

Solution

the equality $Y = X^{\delta}$ is checked by the Revocation Authority holding the key δ.

Disadvantages

- the system is not distributed one anymore
- RA gets traffic data
- potentially RA may recognize each single user while the primary goal was to hide the identity of the user

UProve Revocation
attack mechanism

Overall idea

1. Revocation Authority manipulates some parameters or data
2. ... so that nobody can see the difference
3. but any time when the verifier presents X, Y for checking that $X^{\delta} = Y$, the Revocation authority learns **who presents the credentials to this verifier**

fully automatic Big Brother on large scale

UProve Revocation

attack 1: Creating a corrupted initial witness

Input:

RA private key:	$\delta \in \mathbb{Z}_q$
Revocation parameter:	g_t
List of revoked attribute values:	R
New user's revocation attribute:	$x_{id} \notin R$
Current accumulator:	V
Auxiliary database:	$\mathcal{T}$

Computation:

1. compute d and W via `ComputeWitness` for R and x_{id}
2. choose u at random
3. $\underline{d} := d + u \bmod q$
4. $\underline{Q} := VW^{-x_{id}} g_t^{-\underline{d}}$

Output:

insert (g_t^u, x_{id}) in the database $\mathcal{T}$

give $(\underline{d}, W, \underline{Q})$ to the user holding x_{id}

UProve Revocation
attack 1: verification of X, Y

Revocation Authority gets X, Y for which:

$$X^{\delta} = W^{\delta} g^{t_1 \delta} = QK^{t_1} = \underline{Q} g_t^u K^{t_1} = Y g_t^u \neq Y \ .$$

Revocation Authority searches for an entry (Z, x_{id}) such that $X^{\delta} = YZ$.

- If there is one, then the answer is `correct` and as a side effect the Revocation Authority learns x_{id}.
- Otherwise, the answer is `false`.

UProve Revocation
attack 1

how to compute (Q_1, W_1, d_1) from (manipulated) (Q, W, d)?

- According to `UpdateWitness` $\underline{d_1}$, the new value of the parameter d, equals

$$\underline{d_1} = \underline{d}(x' - x_{id}) = (d + u)(x' - x_{id}) = d_1 + u(x' - x_{id})$$

  where $d_1 = d(x' - x_{id})$ is the correct value for the correct initial witness.
- W will updated correctly since no manipulated value is applied for the update.
- The new value of Q equals

$$\underline{Q_1} = VW^{-x_{id}} g_t^{-\underline{d_1}} = VW^{-x_{id}} g_t^{-d_1 - u(x' - x_{id})} = Q_1 g_t^{-u(x' - x_{id})},$$

  where Q_1 is the value of Q computed for the correct d_1.

If the verifier presents a pair (X, Y) created by the user holding x_{id}, then

$$X^{\delta} = W_1^{\delta} g_t^{t_1 \delta} = Q_1 K^{t_1} = \underline{Q_1} g_t^{u(x' - x_{id})} K^{t_1} = Y g_t^{u(x' - x_{id})}.$$

Defense

learn the IDs of all revoked users

... and compute the witness yourself

the MS technical draft suggests that the revocation attributes of the revoked users can be hidden:

> *If the revocation list is secret, or for better efficiency, the witnesses are computed by the Revocation Authority . . .*

Moreover, initially the set of revoked users may be large and complicated (artificial users due to system testing and initialization).

Solution

see IACR eprint 108/2015: idea+implementation

UProve Revocation
attack 2

Main properties

- compute the witnesses according to the specification
- .. but manipulate K: now $K = g^{\tilde{\delta}}$ where $\tilde{\delta} \neq \delta$
  - but still use δ for accumulator, witnesses, ...
  - recognizing that $K \neq g^{\delta}$ would require solving DDH Problem
- there are many elements depending on δ that potentially could disclose a deviation from the protocol - nevertheless the users cannot see any difference

UProve Revocation
attack 2

Main points

- $K = g^{\tilde{\delta}}$
- now Revocation Authority gets X and Y such that in general $X^{\delta} \neq Y$
- Revocation Authority computes
  1. $\widetilde{W} := Y \cdot X^{-\tilde{\delta}}$.
     (then $\widetilde{W} = Q \cdot W^{-\tilde{\delta}} = W^{\delta - \tilde{\delta}}$.)
  2. check $W \stackrel{?}{=} \widetilde{W}^{\eta}$, where $\eta = (\delta - \tilde{\delta})^{-1} \bmod q$ and W has been obtained for no-manipulated calculations and a tested concrete revocation attribute x_{id}

UProve Revocation
attack 2

Patch

request a proof of equality of discrete logarithms
– the same δ must be used at different places

unfortunately, the attacker can create a more involved attack

UProve Revocation
attack 3

Manipulations on V

- use $\delta := (\alpha_1 + \alpha_2) \cdot 2^{-1} \bmod q$
- For $i = 0, 1, 2, \ldots$ let:

$$\Lambda_i := g_t^{(\alpha_1^i + \alpha_2^i) \cdot 2^{-1} \bmod q} \quad \text{and} \quad \Delta_i := g_t^{\delta^i} \ .$$

- the regular computation:

$$V = \prod_{i=0}^{m} (\Delta_i)^{a_i}$$

where

$$\prod_{i=1}^{m} (\delta + x_i) = \sum_{i=0}^{m} a_i \cdot \delta^i$$

- attack: take

$$V = \prod_{i=0}^{m} (\Lambda_i)^{a_i}$$

UProve Revocation
attack 3

key property:

the updates of the witnesses are consistent with the manipulated accumulator no matter who makes updates:

- Revocation Authority
- the users themselves

tracing:

It turns out that X^{δ}/Y can be recomputed with α_1, α_2 and the x_{id} of the user issuing non-revocation proof
– a quite technical and tedious proof

Patches?

a proof of correctness of V
the proof length is linear in the number of revoked users??

Conclusions

Lesson learnt

- **the procedures must be fully specified – under-specification enables creating malicious systems with insecurity-by-design**
- **serious problems with UProve Revocation system as specified by MS**
  - $\Rightarrow$ many patches necessary, no efficient patch for the last attack in case of a large scale system
  - $\Rightarrow$ the system more and more complicated
  - $\Rightarrow$ a complicated system is more vulnerable as it is easier to overlook attack scenarios,
  - $\Rightarrow$ no serious analysis as it is costly, time-consuming, boring, unattractive for academia, less effective than marketing propaganda ...
- maybe UProve revocation should **backtrack to FC'2013** solution by Acar, Chow, Nguyen (with bilinear mappings)

Thanks for your attention!

Contact data

1. Miroslaw.Kutylowski@pwr.edu.pl
2. http://kutylowski.im.pwr.edu.pl
3. http://ki.pwr.edu.pl