U-Prove Designated-Verifier Accumulator Revocation Extension

Draft Revision 1

Microsoft Research

Authors: Lan Nguyen, Christian Paquin

September 11, 2013 (updated on February 26, 2014)

Summary

This document extends the U-Prove Cryptographic Specification [UPCS] by specifying an efficient revocation mechanism based on a dynamic accumulator. This scheme requires a designated verifier that shares the Revocation Authority's private key. Unlike many accumulator schemes based on bilinear pairings, this scheme is built using a prime-order group (like the ones defined in [UPCS]) and is therefore suitable for system that require standard constructions.

Contents

List of Figures

Change history

Version	Date	Description
Draft Revision 1	09/11/2013	Initial draft
	02/26/2014	Added clarification to Figure 3

1 Introduction

This document extends the U-Prove Cryptographic Specification [UPCS] by specifying an efficient revocation mechanism based on a dynamic accumulator. This scheme requires a designated verifier that shares the Revocation Authority's private key. Unlike many accumulator schemes based on bilinear pairings, this scheme is built using a prime-order group (like the ones defined in [UPCS]) and is therefore suitable for system that require standard constructions.

1.1 Notation

In addition to the notation defined in [UPCS], the following notation is used throughout the document.

$a \notin A$	Indicates that element a is not in set A.

The key words "MUST", "MUST NOT", "SHOULD", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119].

1.2 Feature overview

Revocation is an important feature of a credential system; this document specifies a scheme based on a dynamic accumulator using a standard prime order group instead of bilinear pairings.[1]

A cryptographic accumulator allows the aggregation of a large set of elements into one constant-size value, and the ability to prove that an element has been aggregated or not in the accumulator.

The accumulator scheme is built on the prime order groups defined in [UPCS]; it is therefore possible to use the scheme as a revocation mechanism for U-Prove tokens. The *Revocation Authority* is a new party that manages the revocation list and validates the users' non-revocation proofs. Each token encodes a unique user identifier UID; a non-revocation proof consists of proving that the value UID has not been accumulated in the accumulator. In order to create an efficient non-revocation proof, users periodically obtain revocation witnesses (computed on-demand by the Revocation Authority, or by users as the revocation list is updated); users can then compute in constant time the non-revocation proof.

We detail the DA revocation extension in five steps.

1. **Revocation Authority setup**: The Revocation Authority generates its public parameters and secret key, and makes the public parameters available to users.
2. **Token issuance**: The user obtains U-Prove tokens encoding her unique identifier UID from the Issuer.
3. **Revocation list management**: Periodically, the Revocation Authority updates the revocation accumulator, and the user obtains non-revocation witnesses from the Revocation Authority, or computes them using the revocation list update.
4. **Token presentation**: The user presents a U-Prove token to the Verifier, including a non-revocation proof (using the non-revocation witnesses). The Verifier validates the presentation proof.
5. **Revocation verification**: The Verifier sends the non-revocation proof to the Revocation Authority that verifies that the undisclosed UID does not appear on the current revocation list.

[1] Many accumulator-based schemes rely on bilinear pairings, common in the cryptographic literature, but not yet popular in industry systems and standards.

2 Protocol specification

2.1 Revocation Authority setup

The Revocation Authority generates its key and parameters as specified in Figure 1. The group G_q MUST match the one specified in the parameters of Issuers that will issue tokens revocable by this Revocation Authority.

RASetup( )

Input

Parameters: desc(G_q), g (the G_q generator)

Computation

Choose δ, r_1 and r_2 at random from $\mathbb{Z}_q^*$

Compute $P \coloneqq g, H \coloneqq g^{r_1}, K \coloneqq H^{\delta}, G \coloneqq g^{r_2}$

Output

Private key: δ

Public key: P, H, K, G

Figure 1: Function RASetup

2.2 Token issuance

Token issuance follows the same steps as in [UPCS]. One of the attributes, called the *revocation attribute* and denoted x_{id} (where id is an index value between 1 and the number of attributes in the tokens), is reserved for revocation.

2.3 Revocation list management

The Revocation Authority computes the accumulator corresponding to a set of revoked attribute values[2] (see Figure 2); the accumulator is re-computed when values are added or removed from the revocation list. Users periodically obtain revocation witnesses corresponding to their revocation attribute x_{id} allowing them to create non-revocation proofs. If the revocation list is secret, or for better efficiency, the witnesses are computed by the Revocation Authority (see Figure 3); otherwise, the witnesses are computed by users and updated when values are added or removed from the revocation list (see Figure 4).

[2] Initially, this set can be empty.

`ComputeAccumulator`( )

Input

Revocation Authority private key: $\delta \in \mathbb{Z}_q^*$
Revocation Authority public key: $P, H, K, G \in G_q$
List of revoked attribute values: $R = \{x_1, \ldots, x_m\} \in \mathbb{Z}_q - \{\delta\}$

Computation

$V := P^{\prod_{x \in R}(\delta + x)}$

Output

Return accumulator V

Figure 2: Function ComputeAccumulator

`ComputeWitness`( )

Input

Revocation Authority private key: $\delta \in \mathbb{Z}_q^*$
Revocation Authority public key: $P, H, K, G \in G_q$
List of revoked attribute values: $R = \{x_1, \ldots, x_m\} \in \mathbb{Z}_q - \{\delta\}$
Target attribute value: $x_{id} \notin R$
Current accumulator: $V \in G_q$

Computation

$d = f(\delta) \; mod \; (\delta + x_{id}) \in \mathbb{Z}_q$ where $f(\delta) = \prod_{x \in R}(\delta + x)$
$W = P^{(\prod_{x \in R}(\delta + x) - d)/(\delta + x_{id})}$
$Q = VW^{-x_{id}}P^{-d}$

Output

Revocation witness for target x_{id}: $(d, W, Q)_{x_{id}}$

Figure 3: Function ComputeWitness

Note that the computation of d is a polynomial division of polynomial $f(\delta) = \prod_{x \in R}(\delta + x)$ over polynomial $(\delta + x_{id})$ in polynomial ring $\mathbb{Z}_q[\delta]$. As the denominator is a polynomial of degree 1, the result d is a polynomial of degree 0, i.e. a constant, in the polynomial ring. So d can be computed from just the set R and does not depend on δ.

UpdateWitness( )

Input

Revocation Authority private key: $\delta \in \mathbb{Z}_q^*$
Revocation Authority public key: $P, H, K, G \in G_q$
Revocation attribute value: x_{id}
New revoked value: x'
Boolean: **add** (if true, x' will be added to the witness, otherwise it will be removed)
Old accumulator: $V \in G_q$
Old witness: $(d, W, Q)_{x_{id}}$
Updated accumulator: $V' \in G_q$

Computation

if **add** = true

$d' := d(x' - x_{id})$
$W' := VW^{(x'-x_{id})}$
$Q' := V'W'^{-x_{id}}P^{-d'}$

else

$d' := d(x' - x_{id})^{-1}$
$W' := (V'^{-1}W)^{(x'-x_{id})^{-1}}$
$Q' := V'W'^{-x_{id}}P^{-d'}$

Output

Updated revocation witness: $(d', W', Q')_{x_{id}}$

Figure 4: Function UpdateWitness

2.4 Token presentation

The presentation proof is generated according to the needs of the application following [UPCS], but additionally x_{id} is a committed attribute. The (public) output $\tilde{c}_{id} = g^{x_{id}} g_1{}^{o_{id}}$ and the (private) opening information (x_{id}, o_{id}), are input to the non-revocation proof generation defined in Figure 5.

GenerateNonRevocationProof( )

Input

Parameters: UID_P, desc(G_q), $\text{UID}_\mathcal{H}$, g_1
Commitment to x_{id}: $\tilde{c}_{id}$
Opening information: x_{id}, o_{id}
Revocation Authority public key: P, H, K, G
Revocation witness: $(d, W, Q)_{x_{id}}$

Computation

Generate $t_0, t_1, t_2, t_3, k_0, \ldots, k_8$ at random from $\mathbb{Z}_q$
$B := g^{k_0} g_1^{t_0}$
$X := WH^{t_1}$
$Y := QK^{t_1}$
$R := G^{t_1} H^{t_2}$
$S := G^{d^{-1}} H^{t_3}$
$T_1 := G^{k_1} H^{k_2}$
$T_2 := G^{k_7} H^{k_4} R^{-k_0}$
$T_3 := G^{k_6} H^{k_3}$
$T_4 := H^{k_8} S^{-k_5}$
$\Gamma := X^{-k_0} H^{k_7} K^{k_1} P^{-k_5}$
$c' := \mathcal{H}(g, g_1, P, H, K, G, \tilde{c}_{id}, B, X, Y, R, S, T_1, T_2, T_3, T_4, \Gamma)$
$r' := -c'^{\tilde{o}_{id}} + t_0 \bmod q$
$s_0 := -c' x_{id} + k_0 \bmod q$
for each $i \in \{1,2,3\}$, $s_i := -c' t_i + k_i \bmod q$
$s_4 := -c' t_2 x_{id} + k_4 \bmod q$
$s_5 := -c' d + k_5 \bmod q$
$s_6 := -c' d^{-1} + k_6 \bmod q$
$s_7 := -c' t_1 x_{id} + k_7 \bmod q$
$s_8 := -c' t_3 d + k_8 \bmod q$
Delete $t_0, t_1, t_2, t_3, k_0, \ldots, k_8$

Output

Return $c', r', s_0, \ldots, s_8, X, Y, R, S$

Figure 5: Function GenerateNonRevocationProof.

2.5 Revocation verification

The revocation verification function, defined in Figure 6, is run after the corresponding presentation proof has been successfully verified. Inputs are the commitment $\tilde{c}_{id}$ from the presentation proof and the non-revocation proof. If the presentation and non-revocation proofs are valid, then the verifier has assurance that $\tilde{c}_{id}$ is a valid commitment to the attribute x_{id} and that x_{id} is not in the revocation list.

VerifyNonRevocationProof( )

Input

Parameters: desc(G_q), g_1
Commitment to x_{id}: $\tilde{c}_{id}$
Non-revocation proof: $c', r', s_0, \ldots, s_8, X, Y, R, S$
Revocation Authority public key: P, H, K, G
Revocation Authority private key: δ

Computation

$B \coloneqq g^{s_0} g_1^{r'} \tilde{c}_{id}^{c'}$
$T_1 \coloneqq G^{s_1} H^{s_2} R^{c'}$
$T_2 \coloneqq G^{s_7} H^{s_4} R^{-s_0}$
$T_3 \coloneqq G^{s_6} H^{s_3} S^{c'}$
$T_4 \coloneqq G^{-c'} H^{s_8} S^{-s_5}$
$\Gamma \coloneqq X^{-s_0} H^{s_7} K^{s_1} P^{-s_5} (V^{-1} Y)^{c'}$
Verify that $c' = \mathcal{H}(g, g_1, P, H, K, G, \tilde{c}_{id}, B, X, Y, R, S, T_1, T_2, T_3, T_4, \Gamma)$
Verify that $Y = X^{\delta}$

Figure 6: Function VerifyNonRevocationProof

References

[RFC2119] Scott Bradner. *RFC 2119: Key words for use in RFCs to Indicate Requirement Levels*, 1997. ftp://ftp.rfc-editor.org/in-notes/rfc2119.txt.

[UPCS] Christian Paquin, Greg Zaverucha. *U-Prove Cryptographic Specification V1.1 (Revision 2)*. Microsoft, April 2013. http://www.microsoft.com/u-prove.