U-Prove Designated-Verifier Accumulator Revocation Extension

Draft Revision 2

Microsoft Research

Authors: Lan Nguyen, Christian Paquin

May 2015

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 used in the U-Prove protocol.

Contents

Summary 1
1 Introduction 3
1.1 Notation 3
1.2 Feature overview 3
2 Protocol specification 4
2.1 Revocation Authority setup 4
2.2 Token issuance 4
2.3 Revocation list management 4
2.4 Token presentation 6
2.4.1 Revocation verification 7
References 8

List of Figures

Figure 1: Function RASetup 4
Figure 2: Function ComputeAccumulator 5
Figure 3: Function ComputeWitness 5
Figure 4: Function UpdateWitness 6
Figure 5: Function GenerateNonRevocationProof. 7
Figure 6: Function VerifyNonRevocationProof 8

Change history

Version	Date	Description
Draft Revision 1	09/11/2013	Initial draft
	02/26/2014	Added clarification to Figure 3
Draft Revision 2	06/19/2014	Improved scheme performance
	05/15/2015	Clarified nature of the revocation set

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*[2] 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 for a public set of revoked identifiers, 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] The Revocation Authority is a role that can be played by other U-Prove protocol participants. For examples, in systems where the Issuer both issues and verifies tokens, it is natural to have the Issuer also implement the Revocation Authority role.

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 be one defined in [UPRPP] and token Issuers MUST use the same group in their Issuer Parameters in order to support revocation services from this Revocation Authority.

RASetup( )

Input

Parameters:[3] desc(G_q), g , g_1, g_t

Computation

Choose δ at random from $\mathbb{Z}_q^*$

Compute $K \coloneqq g^\delta$

Output

Private key: δ

Public key: K

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[4] (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 Authority is trusted, and for better efficiency, the witnesses can be computed by the Revocation Authority (see Figure 3);[5] otherwise, the witnesses are computed by users and updated when values are added or removed from the revocation list (see Figure 4).

[3] The parameters g, g_1, g_t are defined in [UPRPP] for supported groups G_q. These three values have been chosen because they are 1) randomly generated, 2) always present in Issuer Parameters of revocable tokens with at least one attribute (minimum needed for revocation), and 3) the first two correspond to the bases in U-Prove commitment values.

[4] Initially, this set can be empty.

[5] The user can always recalculate its witness given the current revocation set, to verify the one calculated by the Revocation Authority.

ComputeAccumulator( )

Input

Revocation Authority private key: $\delta \in \mathbb{Z}_q^*$
Revocation parameters: $g_t \in G_q$
Set of revoked attribute values: $R = \{x_1, \dots, x_m\} \in \mathbb{Z}_q - \{-\delta\}$

Computation

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

Output

Return accumulator V

Figure 2: Function ComputeAccumulator

ComputeWitness( )

Input

Revocation Authority private key: $\delta \in \mathbb{Z}_q^*$
Revocation parameters: $g_t \in G_q$
Set of revoked attribute values: $R = \{x_1, \dots, x_m\} \in \mathbb{Z}_q - \{-\delta\}$
Target attribute value: $x_{id} \notin R$
Current accumulator: $V \in G_q$

Computation

$d := \prod_{x \in R}(x - x_{id})$
$W := g_t^{(\prod_{x \in R}(\delta + x) - d)(\delta + x_{id})^{-1}}$
$Q := V W^{-x_{id}} g_t^{-d}$

Output

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

Figure 3: Function ComputeWitness

Note that the computation of d is equivalent to 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 parameters: $g_t \in G_q$
Revocation attribute value: x_{id}
Updated 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

$d' \coloneqq d(x' - x_{id})$
$W' \coloneqq VW^{(x'-x_{id})}$
$Q' \coloneqq V'W'^{-x_{id}}g_t^{-d'}$

else

$d' \coloneqq d(x' - x_{id})^{-1}$
$W' \coloneqq (V'^{-1}W)^{(x'-x_{id})^{-1}}$
$Q' \coloneqq V'W'^{-x_{id}}g_t^{-d'}$

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], and additionally x_{id} is a committed attribute. The (public) output $\tilde{c}_{id} = g^{x_{id}}g_1{}^{\tilde{o}_{id}}$ and the (private) opening information $(x_{id}, \tilde{o}_{id})$, are input to the non-revocation proof generation defined in Figure 5.

GenerateNonRevocationProof( )

Input

Revocation parameters: desc(G_q), $\mathrm{UID}_{\mathcal{H}}$, g, g_1, g_t
Commitment to x_{id}: $\tilde{c}_{id} \in G_q$
Opening information: $x_{id}, \tilde{o}_{id}$
Revocation Authority public key: $K \in G_q$
Revocation witness: $(d, W, Q)_{x_{id}}$

Computation

Generate $t_1, t_2, k_1, k_2, k_3, k_4, k_5, k_6$ at random from $\mathbb{Z}_q$
$X \coloneqq W g^{t_1}$
$Y \coloneqq Q K^{t_1}$
$C_d \coloneqq g_t^d g_1^{t_2}$
$w \coloneqq d^{-1} \bmod q$
$z \coloneqq t_1 \tilde{o}_{id} - t_2 \bmod q$
$z' \coloneqq -t_2\, w \bmod q$
$T_1 \coloneqq X^{k_1} (\tilde{c}_{id} K)^{-k_2} g_1^{k_3}$
$T_2 \coloneqq g^{k_1} g_1^{k_4}$
$T_3 \coloneqq C_d^{k_5} g_1^{k_6}$
$c' \coloneqq \mathcal{H}(g, g_1, g_t, K, \tilde{c}_{id}, X, Y, C_d, T_1, T_2, T_3)$
$s_1 \coloneqq -c' x_{id} + k_1 \bmod q$
$s_2 \coloneqq -c' t_1 + k_2 \bmod q$
$s_3 \coloneqq -c' z + k_3 \bmod q$
$s_4 \coloneqq -c' \tilde{o}_{id} + k_4 \bmod q$
$s_5 \coloneqq -c' w + k_5 \bmod q$
$s_6 \coloneqq -c' z' + k_6 \bmod q$
Delete $t_1, t_2, k_1, k_2, k_3, k_4, k_5, k_6, w, z, z', T_1, T_2, T_3$

Output

Return $c', s_1, s_2, s_3, s_4, s_5, s_6, X, Y, C_d$

Figure 5: Function GenerateNonRevocationProof.

2.4.1 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

Revocation parameters: desc(G_q), $\mathrm{UID}_{\mathcal{H}}$, g, g_1, g_t
Commitment to x_{id}: $\tilde{c}_{id} \in G_q$
Non-revocation proof: $c', s_1, s_2, s_3, s_4, s_5, s_6, X, Y, C_d$
Revocation Authority public key: $K \in G_q$
Revocation Authority private key: δ
Revocation accumulator: $V \in G_q$

Computation

$T_1 \coloneqq (VY^{-1}(C_d)^{-1})^{c'} X^{s_1}(\tilde{c}_{id}K)^{-s_2} g_1^{s_3}$
$T_2 \coloneqq \tilde{c}_{id}{}^{c'} g^{s_1} g_1^{s_4}$
$T_3 \coloneqq g_t^{c'}(C_d)^{s_5} g_1^{s_6}$
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 $Y = X^{\delta}$

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 3)*. Microsoft, December 2013. http://www.microsoft.com/u-prove.

[UPRPP] Christian Paquin. *U-Prove Recommended Parameters Profile V1.1 (Revision 2)*. Microsoft, April 2013. http://www.microsoft.com/u-prove.