

# 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)**

© 2014 Microsoft Corporation. All rights reserved. This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

## 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

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
2.5 Revocation verification .....	7
References .....	9

## 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

## 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.<sup>1</sup>

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.

---

<sup>1</sup> 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.

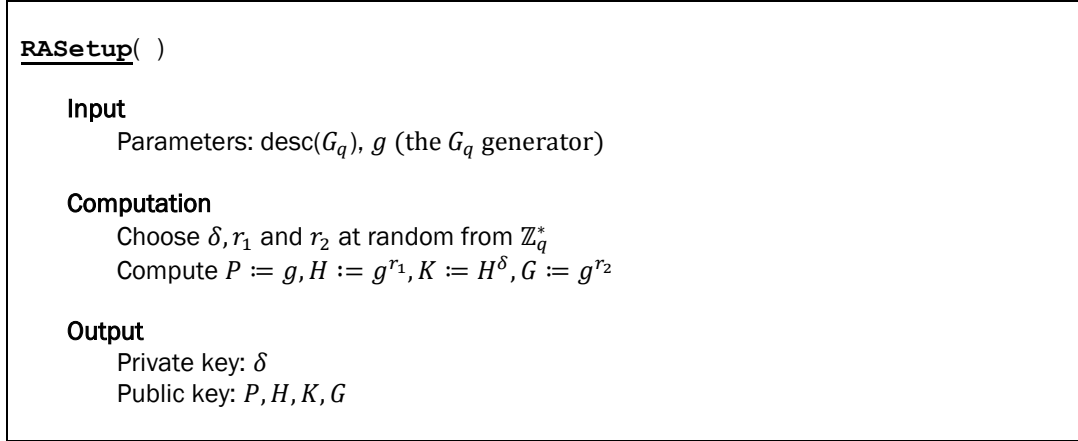


Figure 1: Function RASetup

### 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<sup>2</sup> (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).

---

<sup>2</sup> Initially, this set can be empty.

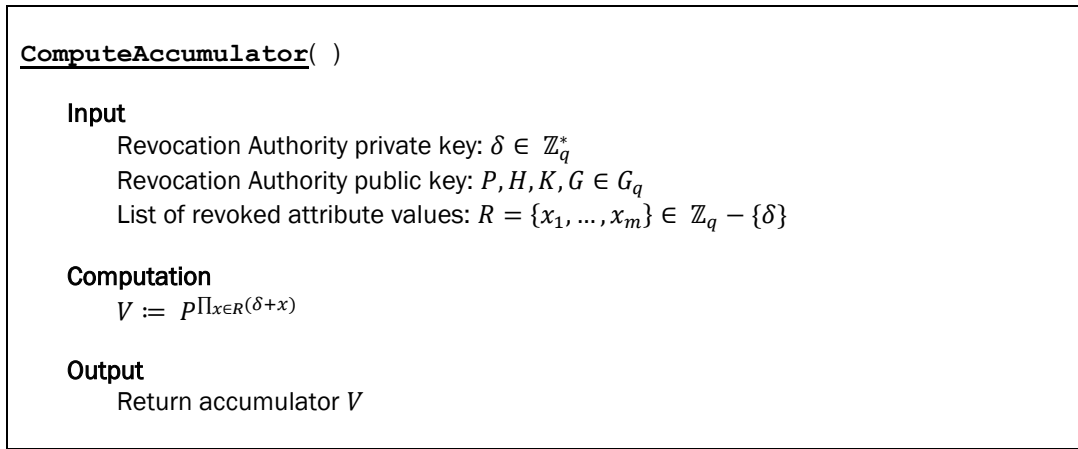


Figure 2: Function ComputeAccumulator

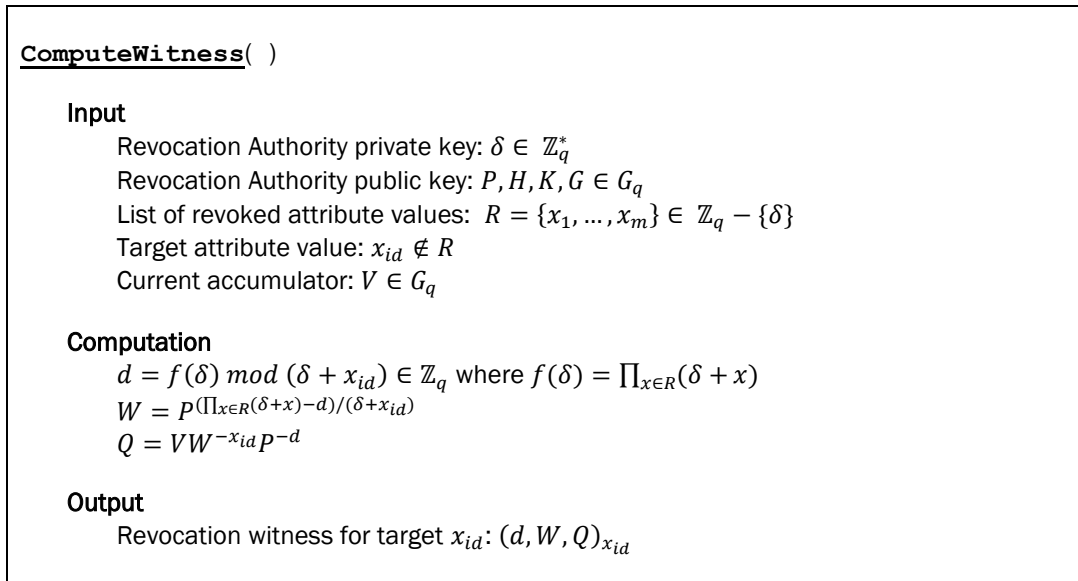


Figure 3: Function ComputeWitness

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  $\delta$ .

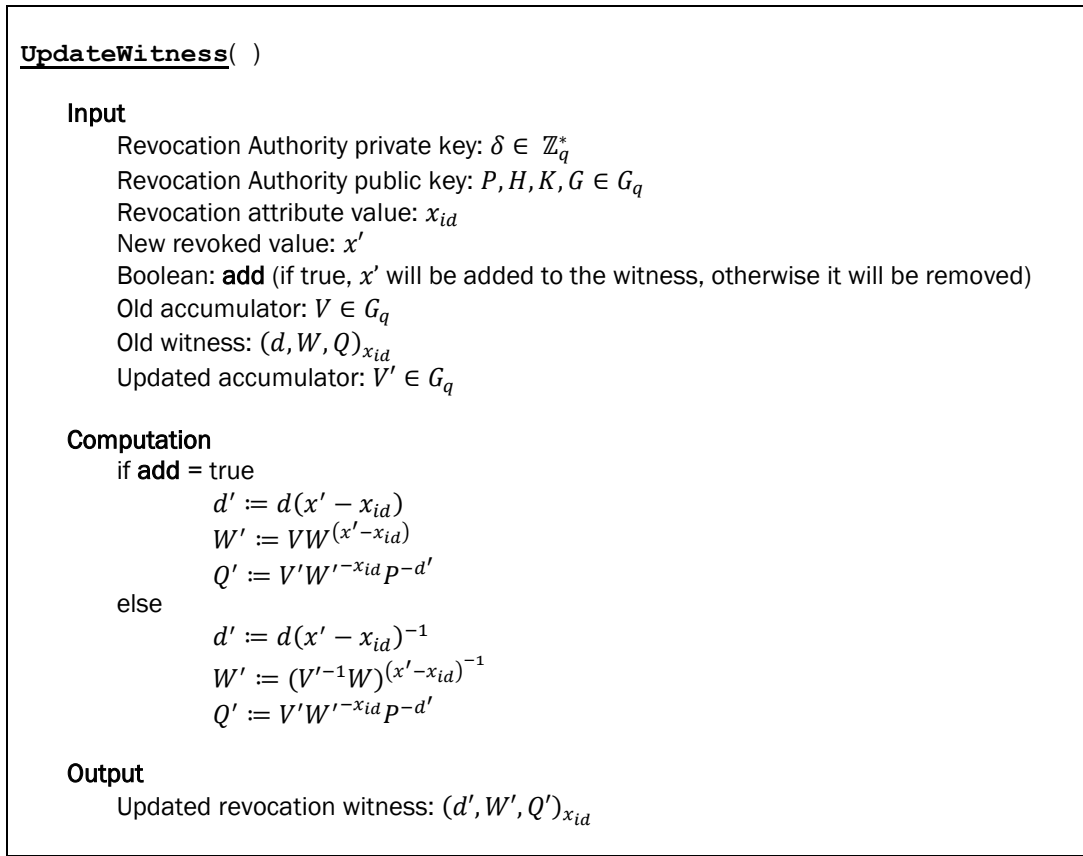


Figure 4: Function UpdateWitness

## 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.

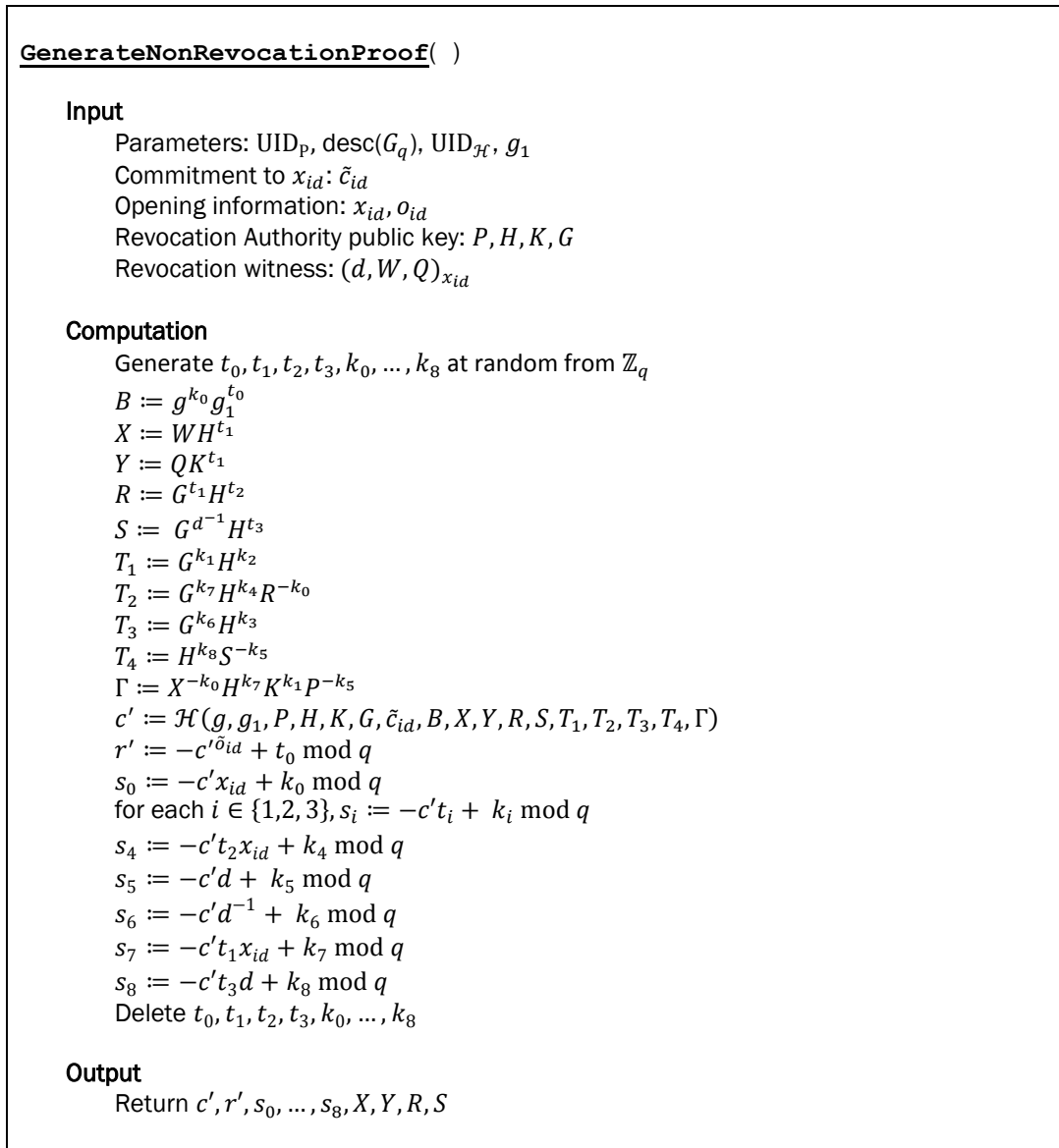


Figure 5: Function GenerateNonRevocationProof.

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

**Computation**

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