U-Prove Set Membership Proof Extension

Draft Revision 1

Microsoft Research

Author: Mira Belenkiy

June 2014

Summary

This document extends the U-Prove Cryptographic Specification [UPCS] by specifying set membership proofs. This allows proving that a U-Prove attribute value is within a set of values without disclosing which one.

Contents

Summary 1
1 Introduction 2
1.1 Notation 3
1.2 Feature overview 3
2 Protocol specification 4
2.1 Presentation 4
2.2 Verification 5
3 Security considerations 6
References 6

List of Figures

Figure 1: SetMembershipProve 5
Figure 2: SimulateProof ..... Error! Bookmark not defined.
Figure 3: SetMemberhship Verify 5

Change history

Version	Description
Revision 1	Initial draft

1 Introduction

This document extends the U-Prove Cryptographic Specification [UPCS] by specifying set membership proofs. The Prover will prove to the Verifier that a committed value is a member of a Verifier-specified set $S = \{s_1, \dots, s_n\}$.

The Prover will create a special honest-verifier non-interactive zero-knowledge proof of knowledge. The proof technique is a standard OR combination of multiple Sigma protocols. The Prover creates n separate sub-proofs that $x = s_1, x = s_2, \dots, x = s_n$. (Obviously only one of these statements can be true). The "challenge" for the set membership protocol is equal to the sum of the individual challenges for the sub-proofs. Thus, the Prover is able to fake $n - 1$ of the proofs using the special honest-verifier simulator, while creating only one real proof. The Verifier is unable to distinguish the fake proofs from the real proofs (due to the special honest-verifier zero knowledge property). As a result, all the Verifier learns is that the committed value x is equal to one of the members of the set S.

The size of the set membership proof is $2|S| - 1$ group/field elements, where $|S|$ is the number of elements in the set S. Creating the proof and verifying it requires $|S|$ multi-exponentiations.

The U-Prove Cryptographic Specification [UPCS] allows the Prover, during the token presentation protocol, to create a Pedersen Commitment and show that the committed value is the equal to a particular token attribute. The Prover MAY use this Pedersen Commitment as input for the set membership proof. The Issuance and Token

Presentation protocols are unaffected by this extension. The Prover may choose to create a set membership proof after these two protocols complete. The Verifier does not need to specify the set S until the Prover is ready to execute the set membership proof. Specifically, the set S can be determined after Issuance and Token Presentation.

The Prover's committed value x and every member of the set S will be encoded as elements of the field $\mathbb{Z}_q$. The specific encoding format is not relevant to the set membership proof. If x is an attribute of a U-Prove token, it MAY be hashed.

1.1 Notation

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

C	Value of the Prover's Pedersen Commitment.
x	Committed value of Pedersen Commitment C.
y	Opening of Pedersen Commitment C.
S	Verifier specified set of elements in $\mathbb{Z}_q$
s_i	The i[th] element in Verifier specified set S.
c	Challenge
a_i	Part of set membership proof: "commitment".
c_i	Part of set membership proof: "challenge". Also referred to as sub-challenge.
r_i	Part of set membership proof: "response".

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

The Prover knows the opening of a Pedersen Commitment $C = g^x h^y$ and needs to show that x is in the Verifier-specified set $S = \{s_1, \dots, s_n\}$. The Prover will create a special-honest verifier zero-knowledge proof of knowledge that the Prover knows a pair of values (x, y) such that:

1 $C = g^x h^y$
2 The value x is a member of the set $S = \{s_1, \dots, s_n\}$.

Suppose S is a set containing just one element s_1. In this case, the Prover can perform a three round Sigma protocol to show he knows the discrete logarithm representation of Cg^{-s_1} in terms of h using standard techniques:

1. Choose a random value w from $\mathbb{Z}_q$ and compute commitment $a := h^w$.
2. Compute the challenge $c := \mathcal{H}(C, S, a)$.
3. Compute the response $r := cy + w \bmod q$.

The Prover sends the Verifier the proof (a, r). The Verifier would independently compute the challenge $c := \mathcal{H}(C, S, a)$. Then the Verifier would check that $h^r = C^c g^{-cs_1} a$.

For sets $|S| > 1$, the Prover will use a simulator to fake $n-1$ of the proofs. Suppose $x = s_i$. For all $j \neq i$, the Prover does the following:

1. Choose a random challenge c_j from $\mathbb{Z}_q$.
2. Choose a random response r_j from $\mathbb{Z}_q$.
3. Compute commitment $a_j = h^{r_j} g^{s_j + c} C^{-c}$.

The Prover will then have a list of fake proofs (a_j, c_j, r_j) for all $j \neq i$. Next, the Prover will create the real proof for $x = s_i$. The Prover chooses a random w from $\mathbb{Z}_q$ and computes commitment $a := g^w$ as usual. For the challenge, the Prover computes a global challenge $c = \mathcal{H}(C, S, a_1, \ldots, a_n)$. Then the Prover computes the sub-challenge for the real proof as $c_i := c - \sum_j c_j$. Then the Prover computes the response $r_i := c_i y + w \bmod q$ as usual. The full proof consists of $(a_1, \ldots, a_n, c_1, \ldots, c_{n-1}, r_1, \ldots, r_n)$. The last sub-challenge c_n is omitted as the Verfier can compute it independently from the challenge c.

The Verifier takes the proof $(a_1, \ldots, a_n, c_1, \ldots, c_{n-1}, r_1, \ldots, r_n)$ and independently computes the challenge $c := \mathcal{H}(C, S, a_1, \ldots, a_n)$ and $c_n := c - sum\, c_i$. Then the Verifier checks each proof (a_i, c_i, r_i) using the same verification equation as for the single discrete logarithm representation proof.

2 Protocol specification

As the set membership proof can be performed independently of the U-Prove token presentation protocols, the common parameters consist simply of the group G_q, two generators g and h, and a cryptographic function $\mathcal{H}$. It is RECOMMENDED to use the generators g and g_1 associated with a group defined in [UPRPP] as the generators g and h, respectively. The set S is chosen by the Verifier. The commitment C and its opening MAY be generated by the Prover.

2.1 Presentation

Figure 1 shows how the Prover computes the set membership proof. The Prover finds an element s_i in the set S such that $x = s_i$, the Prover then uses the special honest-verifier zero-knowledge simulator to generate fake sub-proofs for all $s_j \neq x$, and computes the real proof for s_i.

SetMembershipProve( )

Input

Parameters: desc(G_q), $\text{UID}_{\mathcal{H}}$, g, h
Commitment to x: C
Opening information: x, y
Verifier-specified set: $S = \{s_1, \ldots, s_n\}$.

Computation

Compute index i such that $x = s_i$
For all $j \neq i$
Choose c_j, r_j at random from $\mathbb{Z}_q^*$
$a_j := h^{r_j} g^{s_j c_j} C^{-c_j}$
end
Choose w at random from $\mathbb{Z}_q^*$
$a_i := h^w$
$c := \mathcal{H}(desc(G_q), g, h, \langle S \rangle, C, \langle a_1, \ldots, a_n \rangle)$
$c_i := c - \sum_{j \neq i} c_j \bmod q$
$r_i := c_i y + w \bmod q$

Output

Return $(a_1, \ldots, a_n, c_1, \ldots, c_{n-1}, r_1, \ldots, r_n)$

Figure 1: SetMembershipProve

2.2 Verification

The Verifier independently computes the challenge c and uses it to compute c_n. Then, the Verifier verifies each tuple (a_i, c_i, r_i). Whenever SetMembershipVerify indicates to "Check that…" assume the protocol returns *fail* if the check fails. Otherwise, the protocol will return *pass*.

SetMembershipVerify( )

Input

Parameters: desc(G_q), $\text{UID}_{\mathcal{H}}$, g, h
Commitment to x: C
Verifier-specified set: $S = \{s_1, \ldots, s_n\}$
Proof $(a_1, \ldots, a_n, c_1, \ldots, c_{n-1}, r_1, \ldots, r_n)$

Computation

$c := \mathcal{H}(desc(G_q), g, h, \langle S \rangle, C, \langle a_1, \ldots, a_n \rangle)$
$c_n := c - \sum_{j \neq n} c_j \bmod q$
For all i **in** $1 \ldots n$
Check that $h^{r_i} = C^{c_i} g^{-c_i s_i} a_i$
end

Output

Return *pass*

Figure 2: SetMemberhship Verify

3 Security considerations

The set membership proof protocol is a standard Sigma protocol transformed using the Fiat-Shamir heuristic into a non-interactive proof. The following restriction apply:

- The Prover and the Verifier MUST NOT know the relative discrete logarithm $\log_g h$ of the generators g and h. This is not an issue if the generators are chosen from the list of U-Prove recommended parameters.

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, December 2013. http://www.microsoft.com/u-prove.