U-Prove Inequality Proof Extension

Draft Revision 1

Microsoft Research

Author: Mira Belenkiy

June 2014

Summary

This document extends the U-Prove Cryptographic Specification [UPCS] by specifying proofs of inequality. This allows proving that U-Prove token attribute value is not equal to another target value.

Contents

List of Figures

Change history

Version	Description
Revision 1	Initial draft

1 Introduction

This document extends the U-Prove Cryptographic Specification [UPCS] by specifying inequality proofs. This allows proving that U-Prove token attribute value is not equal to another target value. We consider two types of proofs.

In the Known Value proof, the Prover and Verifier have as common input a commitment C_X, a pair of generators $g, h \in G_q$, and a value v. The Prover wants to show that C_X is not a Pedersen Commitment to v

$$\pi = SPK\{\alpha, \beta \middle| C_X = g^\alpha h^\beta \cap \alpha \neq v\}$$

In the Unknown Value proof, the Prover and Verifier have as common input a commitment C_X, a pair of generators $g, h \in G_q$,

The Prover and Verifier have as common input commitments C_X and C_Y and a pair of generators $g, h \in G_q$. The Prover wants to show that C_X and C_Y are Pedersen Commitments to different values.

$$\pi = SPK\{\alpha, \beta, \delta, \gamma \middle| C_X = g^\alpha h^\beta \cap C_X = g^\delta h^\gamma \cap \alpha \neq \delta\}$$

In both cases, the Prover knows assignments to the unknown values that would satisfy the specified relation. For the Known Value proof, the Prover knows the opening of C_X: (x, z). In the Unknown Value, the Prover also knows the opening of C_Y: (y, w). The Prover will create a special honest-verifier non-interactive zero-knowledge proof of knowledge using its witnesses.

In the case of Known Value proofs, the Prover will generate a new Pedersen Commitment A to a randomly chosen value $a \neq 0$. Then, the Prover will compute $B = g^{(x-v)a}$. The Prover will create an equality proof [EXEQ] to show that A and B are formed correctly, and the Verifier will check that $B \neq 1$. Unknown Value proofs can be reduced to Known Values proofs by computing $\tilde{C}_X \coloneqq C_X / C_Y$ and $v \coloneqq 0$.

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 either C_X or C_Y for the inequality proof. The Issuance and Token Presentation protocols are unaffected by this extension. The Prover may choose to create an inequality proof after these two protocols complete.

The committed value in C_X or C_Y MAY be hashed, and v MAY be the result of a hash. If any of these values are U-Prove token attributes, the attributes MAY be hashed.

1.1 Notation

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

v	Prover will show to Verifier that opening of C_X is not equal to v.
C_X	Value of the Prover's Pedersen Commitment to x.
C_Y	Value of Prover's Pedersen Commitment to y.
x	Committed value of Pedersen Commitment C_X.
z	Opening of Pedersen Commitment C_X.
y	Committed value of Pedersen Commitment C_Y.

w	Opening of Pedersen Commitment C_Y.
A	Commitment to random value a.
a	Committed value of Pedersen Commitment A.
r	Opening of Pedersen Commitment A
B	Helper value in the proof: $B = g^{(x-v)a}$.
M	Map for Equality Proof
$\bar{\bar{A}}$	Array of values of DL equations for equality proof.
$\bar{\bar{g}}$	Array of lists of generators for DL equations for equality proof.
$\bar{\bar{x}}$	Witnesses for equality proof.
$a \leftarrow A$	Choose a uniformly at random from 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

To create the Known Value Inequality Proof, the Prover will choose random $a \leftarrow Z_q^*$ and $r \leftarrow Z_q$ and compute the following two values: $A = g^a h^r$ and $B = g^{(x-v)a}$. As long as $x \neq v$ the value of B is not one. The Prover will create an equality proof [EXEQ] showing that A and B are formed correctly:

$$\pi = SPK\{\alpha, \beta, \delta, \gamma, \lambda, \kappa | C_X = g^\alpha h^\beta = \cap A = g^\delta h^\gamma \cap B = g^\lambda \cap B = (C_X g^{-v})^\delta h^\kappa\}$$

The Verifier will check the proof and verify that $B \neq 1$.

The Unknown Value Inequality Proof can be transformed into a Known Value Inequality Proof. The Prover will compute a new Pedersen Commitment $\tilde{C}_X \coloneqq C_X / C_Y$ and show that the committed value $\tilde{x} = (x - y)$ in $\tilde{C}_X$ is not equal to $\tilde{v} = 0$.

2 Protocol specification

As the inequality 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 $\mathcal{H}$. For Known Value proofs, the Prover and Verifier have common input C_X and v where C_X MAY be generated by the Prover, while either the Prover or Verifier MAY choose v. For Unknown Value proofs, the Prover and Verifier have common input C_X and C_Y where either value MAY be generated by the Prover

2.1 Known Value Presentation

The presentation protocol for Known Value Inequality Proofs is given below.

InequalityProve( )

Input

Parameters: desc(G_q), $\mathrm{UID}_{\mathcal{H}}$, g, h
Known value: v
Commitment to x: C_X
Opening information: x, z

Computation

$a \leftarrow \mathbb{Z}_q^*$
$r \leftarrow \mathbb{Z}_q$
$A \coloneqq g^a h^r$
$B \coloneqq g^{(x-v)a}$
$\bar{\bar{A}}, \bar{\bar{g}}, \mathcal{M} \coloneqq \mathrm{KVParam}(desc(G_q), g, h, v, C_X, A, B)$

$// \ C_X = g^x h^z$
$\bar{\bar{x}}_{0,0} \coloneqq x$
$\bar{\bar{x}}_{0,1} \coloneqq z$

$// \ A = g^a h^r$
$\bar{\bar{x}}_{1,0} \coloneqq a$
$\bar{\bar{x}}_{1,1} \coloneqq r$

$// \ B = g^{(x-v)a}$
$\bar{\bar{x}}_{2,0} \coloneqq (x-v)a$

$// \ B = (C_X \cdot g^{-v})^a h^{-za}$
$\bar{\bar{x}}_{3,0} \coloneqq a$
$\bar{\bar{x}}_{3,1} \coloneqq -za$

$// \ x_{1,0} = x_{3,0}$
$\pi \coloneqq \mathrm{EqualityProve}(desc(G_q), \mathrm{UID}_{\mathcal{H}}, \bar{\bar{A}}, \bar{\bar{g}}, \mathcal{M}, \bar{\bar{x}})$

Output

Return A, B, π

Figure 1: InequalityProve

KVParam( )

Input

Parameters: desc(G_q), g, h
Known value: v
Commitment to x: C_X
Helper values: A, B

Computation

$// \ C_X = g^x h^z$
$\bar{\bar{A}}_0 \coloneqq C_X$
$\bar{\bar{g}}_{0,0} \coloneqq g$
$\bar{\bar{g}}_{0,1} \coloneqq h$

$// \ A = g^a h^r$
$\bar{\bar{A}}_1 \coloneqq A$
$\bar{\bar{g}}_{1,0} \coloneqq g$
$\bar{\bar{g}}_{1,1} \coloneqq h$

$// \ B = g^{(x-v)a}$
$\bar{\bar{A}}_2 \coloneqq B$
$\bar{\bar{g}}_{2,0} \coloneqq g$

$// \ B = (C_X \cdot g^{-v})^a h^{-za}$
$\bar{\bar{A}}_3 \coloneqq B$
$\bar{\bar{g}}_{3,0} \coloneqq C_X \cdot g^{-v}$
$\bar{\bar{g}}_{3,1} \coloneqq h$

$// \ \bar{\bar{x}}_{1,0} = \bar{\bar{x}}_{3,0}$
$\mathcal{M} \coloneqq \emptyset$
$\mathcal{M}.Add((\text{"delta"}, 0), (1, 0))$
$\mathcal{M}.Add((\text{"delta"}, 0), (3, 0))$

Output

Return $\bar{\bar{A}}, \bar{\bar{g}}, \mathcal{M}$

Figure 2: KVParam

2.2 Known Value Verification

The Verifier verifies the set membership and equality proofs.

InequalityVerify( )

Input

Parameters: *desc*(G_q), $\text{UID}_{\mathcal{H}}$, g, h
Known value: v
Commitment to x: C_X
Proof: A, B, π

Computation

$pass \coloneqq true$
If $C_X = 1$ **then** $pass \coloneqq false$
If $B = 1$ **then** $pass \coloneqq false$
$\bar{\bar{A}}, \bar{\bar{g}}, \mathcal{M} \coloneqq \text{KVParam}(desc(G_q), g, h, v, C_X, A, B)$
If $\text{EqualityVerify}(desc(G_q), \text{UID}_{\mathcal{H}}, \bar{\bar{A}}, \bar{\bar{g}}, \mathcal{M}, \pi) = false$ **then** $pass \coloneqq false$

Output

Return *pass*

Figure 3: InequalityVerify

2.3 Unknown Value Presentation

The presentation protocol for Unknown Value Inequality Proofs reduce to the Known Value presentation protocol.

UKVInequalityProve( )

Input

Parameters: *desc*(G_q), $\text{UID}_{\mathcal{H}}$, g, h
Known value: v
Commitment to x: C_X
Opening information: x, z
Commitment to y: C_Y
Opening information: y, w

Computation

$\tilde{C}_X \coloneqq C_X / C_Y$
$\tilde{x} \coloneqq x - y$
$\tilde{z} \coloneqq z - w$
$\tilde{v} \coloneqq 0$
$A, B, \pi \coloneqq \text{InequalityProve}(\text{desc}(G_q), \text{UID}_{\mathcal{H}}, g, h, \tilde{v}, \tilde{C}_X, \tilde{x}, \tilde{z})$

Output

Return A, B, π

2.4 Unknown Value Verify

The Unknown Value verification protocol invokes the Known Value verification protocol.

UKVInequalityVerify( )

Input

Parameters: *desc*(G_q), $\mathrm{UID}_{\mathcal{H}}$, g, h
Known value: v
Commitment to x: C_X
Commitment to y: C_Y
Proof: A, B, π

Computation

$\tilde{C}_X \coloneqq C_X / C_Y$
$\tilde{v} \coloneqq 0$
$pass \coloneqq \mathrm{InequalityVerify}(\mathrm{desc}(G_q), \mathrm{UID}_{\mathcal{H}}, g, h, \tilde{v}, \tilde{C}_X, A, B, \pi)$

Output

Return $pass$

3 Security considerations

The inequality proof protocols rely on the security of the equality 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

[EXEQ] Mira Belenkiy. *U-Prove Equality Proof Extension*. Microsoft, June 2014. http://www.microsoft.com/u-prove.

[EXSM] Mira Belenkiy. *U-Prove Set Membership Proof Extension*. Microsoft, June 2014. http://www.microsoft.com/u-prove.

[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.