Summary
This document extends the U-Prove Cryptographic Specification [UPCS] by specifying an ID escrow capability.
Contents

Summary ............................................................................................................................................... 1
1 Introduction .......................................................................................................................................... 3
  1.1 Notation .......................................................................................................................................... 3
  1.2 Feature overview .............................................................................................................................. 3
2 Protocol specification .......................................................................................................................... 5
  2.1 Auditor Setup ................................................................................................................................... 5
  2.2 Token Issuance ................................................................................................................................. 5
  2.3 Presentation ...................................................................................................................................... 5
  2.4 Verification ...................................................................................................................................... 6
  2.5 Decryption ....................................................................................................................................... 7
3 Security Considerations ...................................................................................................................... 8
References ............................................................................................................................................... 9

List of Figures

Figure 1: Function AuditorSetup ........................................................................................................ 5
Figure 2: Function IEProve .................................................................................................................... 6
Figure 3: Function IEFverify .................................................................................................................. 7
Figure 4: Function AuditorDecrypt .................................................................................................... 7

Change history

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draft Revision 1</td>
<td>09/11/2013 Initial draft</td>
</tr>
</tbody>
</table>

1 Introduction

This document extends the U-Prove Cryptographic Specification [UPCS] by specifying an identity escrow mechanism.

The U-Prove identity escrow feature allows a U-Prove token to be presented anonymously to a Verifier, but the presentation contains the token holder’s identity in an encrypted form. This allows the presentation proof to be de-anonymized if needed. The decryption key needed for de-anonymization is ideally held at arm’s length from the system, by an entity distinct from the Issuer and Verifier called the Auditor. The Auditor is only involved in the rare occasions when de-anonymization is required.

To prevent users from encrypting junk data instead of their identity, the presentation proof with ID escrow includes an additional proof that the encryption of the identity is valid. In this context, valid means i) the ciphertext was computed correctly, following the encryption algorithm, and ii) the plaintext is an attribute from the U-Prove token. For example, the Issuer will ensure that the 2nd attribute in the token always contains a user ID. The proof will give the Verifier assurance that the ciphertext is an encryption of this attribute. This type of encryption is sometimes called verifiable encryption, since parties without the decryption key may verify some property of the plaintext. U-Prove’s ID escrow is however, not a general purpose verifiable encryption scheme, since the plaintext must have a special form to allow efficient decryption.

1.1 Notation

This document will reference the U-Prove cryptographic specification [UPCS]. Notation will be re-used when possible. In protocol descriptions, the statement “Verify X” indicates that an error should be returned and the protocol aborted if X does not hold. 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 identity will be escrowed by encrypting a pseudonym for the user, using Elgamal encryption. Since the pseudonym is a group element, it is a valid plaintext for the Elgamal scheme, and since it has a special form, the user can prove that it corresponds to an attribute from the token. Let \( H \) be an element of \( G_q \), the public key of the Auditor, and let \( x_b \) be an attribute from the user’s token used to create a pseudonym \( P_b = g^{x_b} \) (where \( g \) is the group generator). During presentation, the user will encrypt \( P_b \), as \((E_1, E_2) = (g^r, P_b H^r)\), then prove that \((E_1, E_2)\) is a valid encryption of \( P_b \) and that \( P_b \) was created correctly.

The ciphertext, along with the associated proof of correctness, the token \( T \), and the U-Prove presentation proof (where \( x_b \) is undisclosed), will form the presentation proof with identity escrow. The ciphertext can be stored at the Verifier, and if necessary, the Auditor can decrypt it, using the private key associated with \( H \). Depending on how the pseudonym is formed, it may be enough to identify the user, or additional information from the Issuer may be required to link the pseudonym to an identity.

The ID escrow feature is designed as a U-Prove extension that will take a commitment to \( x_b \) as input, denoted \( C_{x_b} \). The extension will produce a ciphertext encrypting the pseudonym \( P_b = g^{x_b} \) along with proof that \( P_b \) is consistent with \( C_{x_b} \). This commitment will be part of the U-Prove presentation proof (which links \( C_{x_b} \) to the ID escrow attribute). We detail the identity escrow extension in five steps.

i) **Auditor setup**: key generation by the Auditor.

---

1 The User is referred to as the Prover in [UPCS].
ii) **Issuance:** the U-Prove issuance protocol is mostly unchanged, but some information is logged by the Issuer for use during decryption/audit.

iii) **Presentation:** standard U-Prove proof generation step including a commitment to the ID escrow attribute, then the ID escrow ciphertext is created along with proof that it is valid.

iv) **Verification:** the Verifier includes additional steps to verify the ciphertext is valid (after performing the standard U-Prove verification step).

v) **Decryption: (in case of abuse)** The Auditor receives the ciphertext and decrypts it, after performing the required checks.

The system’s decryption key may easily be divided amongst two or more Auditors, such that no single Auditor may decrypt ID escrow ciphertexts without the cooperation of one or more of the other Auditors. This is possible since the encryption scheme is based on Elgamal, which allows the private key to be split using either (i) a simple additive secret sharing scheme (for $k$-of-$k$ decryption), or (ii) a polynomial secret sharing scheme such as Shamir’s (for $k$-of-$n$ decryption, where $n \geq k$).
2 Protocol specification

2.1 Auditor Setup
The Auditor generates an Elgamal key pair as setup. The group parameters \( \text{desc}(G_q) \) MUST match the Issuer parameters that will be used to create tokens. It is also recommended that the Auditor create a policy, describing the values required with a ciphertext, and the conditions under which identity will be revealed. This policy should be bound to the ciphertext during creation.

\[
\text{AuditorSetup}() \\
\text{Input} \\
\text{Parameters: desc}(G_q) \\
\text{Computation} \\
\text{Choose } x \text{ at random from } \mathbb{Z}_q^* \\
\text{Compute } H := g^x \\
\text{Store } x \text{ securely} \\
\text{Output} \\
\text{Return } H
\]

Figure 1: Function AuditorSetup.

2.2 Token Issuance
Token issuance follows the same steps as in [UPCS]. One of the attributes, called the identity escrow attribute and denoted \( x_b \), is reserved for identity escrow. It is RECOMMENDED that the value \( x_b \) be chosen at random by the Prover. The identity escrow pseudonym is defined as \( P_b = g^{x_b} \). The Issuer must know \( P_b \). The Issuer MAY keep a table of \((P_b, \text{ID})\) pairs, so when presented a \( P_b \) value later by the Auditor it can look up the true identity associated with the pseudonym.

In an alternative implementation where the Issuer does not have to store anything, the attribute \( x_b \) is chosen from a relatively small set of possible values (for example, of size less than \( 2^{60} \)) that uniquely identify the user, such as a social security number. This allows \( \log g P_b \) to be computed, revealing \( x_b \) and identifying the user. This approach also allows possible \( x_b \) values to be tested against a given \( P_b \) (we can efficiently test whether \( P_b = g^{x_b} \)).

2.3 Presentation
The presentation proof is generated according the needs of the application following [UPCS], but additionally \( x_b \) is a committed attribute. The (public) output \( C_b = g^{x_b} g_1^{o_b} \) and the (private) opening information \((x_b, o_b)\), are input to the function IEPprove (Figure 2). The additional information field info allows arbitrary data to be bound to the ciphertext, since it is required to verify the presentation proof. Examples include: the entire presentation proof (if it will be sent to the Auditor), a certificate for the Auditor public key, or the Auditor’s decryption policy.
**IEProve**

**Input**
- Parameters: $\text{UID}_p$, $\text{desc}(G_q)$, $\text{UID}_T$, $g_1$
- Commitment to $x_b$: $C_b$
- Opening information: $x_b$, $o_b$
- Token identifier: $\text{UID}_T$
- Auditor public key: $H$
- Additional information: $\text{info}$

**Computation**
- Choose $r, r', x_b'$ at random from $\mathbb{Z}_q^*$
- $E_1 := g^r$
- $E_2 := g^{x_b}H^r$
- $C_b' := g^{r'}g_1^{o_b'}$
- $E_1' := g^{r'}$
- $E_2' := g^{x_b}H^{r'}$
- $c := H(\text{UID}_p, \text{UID}_T, H, C_b, E_1, E_2, C_b', E_1', E_2', \text{info})$
- $r_b := x_b' - cr_b \pmod{q}$
- $r_r := r' - cr \pmod{q}$
- $r_o := o_b' - co_b \pmod{q}$

**Output**
- Return $(E_1, E_2, c, r_b, r_r, r_o)$

---

The first two fields of the hash, $\text{UID}_p$ and $\text{UID}_T$, serve to uniquely identify the Issuer parameters and the token, respectively.

### 2.4 Verification

The verification function of the identity escrow module is called **IEVerify** (Figure 3). The verify step is run after the presentation proof has been successfully verified. Inputs to **IEVerify** are the commitment $C_b$ from the presentation proof and the output of **IEProve**. If the presentation proof is valid, **IEVerify** has assurance that $C_b$ is a valid commitment to the attribute $x_b$. Note that $r_{x_b}$ is used in two places, to prove that the same value is used in $E_2$ and $C_{x_b}$.
IEVerify( )

Input
Parameters: UID_p, desc(G_q), UID_T, g_1
Commitment to x_b: C_b
Token identifier: UID_T
Auditor public key: H
Additional information: info
Output of IEProve(): (E_1, E_2, c, r_b, r_r, r_o)

Computation
Verify that E_1 and E_2 are valid elements of G_q
Verify that r_b, r_r, r_o are valid elements of Z_q
Verify that c has the correct length
C_b'' := (g^{r_b}g_1^{r_o})(C_b)^c
E_1'' := (g^{r_r})(E_1)^c
E_2'' := (g^{r_o}H^{r_o})(E_2)^c
E'' := H(UID_p, UID_T, H, C_b, E_1, E_2, C_b'', E_1'', E_2'', info)

Output
Verify that c = c''

Figure 3: Function IEVerify.

2.5 Decryption
To de-anonymize a Prover, the Verifier sends the output of IEProve to the Auditor, and at least UID_T, but optionally the token and the entire presentation proof. The Auditor verifies proof, and reads any policy information encoded in the info field of the proof. If the request satisfies the decryption policy, the Auditor runs the IEVerify, to ensure that the ciphertext is valid, and consistent with the presentation proof. If valid, the Auditor decrypts the ciphertext (E_1, E_2) as shown in Figure 4.

AuditorDecrypt( )

Input
Inputs to IEVerify: UID_p, desc(G_q), UID_T, g_1, C_b, UID_T, H, info, (E_1, E_2, c, r_b, r_r, r_o)
Auditor secret key: x

Computation
Verify that info is valid (RECOMMENDED)
Verify the ID escrow proof with IEVerify
P_b := E_2E_1^{-x}

Output
Return P_b

Figure 4: Function AuditorDecrypt.

How the output P_b is then used to uniquely identify a user is implementation dependent, and is not specified here.
3 Security Considerations

In addition to the security considerations of [UPCS], implementers should also consider the following.

Auditor Setup

For security, in the case when $G_q$ is an elliptic curve group, it MUST NOT have a pairing which can be computed efficiently. It is recommended to use the U-Prove parameters defined in [UPRPP] which do not have this issue.

Provers will need an authentic copy of $H$ and the policy. Note that the Issuer should not act as an authority to certify $H$, since we want separation between the Issuer and Auditor. Distributing $H$ with authenticity is the usual bootstrapping problem, and may be done, for example, with a CA/PKI, or $H$ may be embedded in Prover devices, or distributed as part of an authenticated software package.

Token Issuance

Based on the policy set by the Auditor, the Issuer should regularly purge old values from the table. For example, if the policy states that the identity may be revealed for a period of up to one month following the use of the token, and the token expires in one month, then the Issuer should delete the table entry after two months. Keeping a minimal amount of data in this table limits the potential for abuse in the event of the Auditor's secret key being exposed.

Choosing a different $x_b$ value for each token also limits the damage of an Auditor compromise, since transactions will then only be linkable by the Issuer. If a single $x_b$ value is used for all of user's tokens, then $P_b$ is the same for all presentation proofs, allowing them to be linked, given the Auditor secret key. To simplify storage and management of one $x_b$ value per token, $x_b$ can be computed as a function of the token secret key (and other token-specific information, like an index/counter). The Issuer will have to keep all the $P_b$ values corresponding to a given identity.

Presentation

No comments specific to presentation.

Decryption

In terms of information relayed from the Verifier to the Auditor, at the very least, the Auditor should know when the proof was created and which Verifier it was presented to (which may differ from the Verifier who is presenting the ciphertext to the Auditor). It is also recommended that the policy information encoded in info, be checked by the Auditor. Without these checks the Auditor becomes a simple decryption oracle, and Verifiers alone are trusted to decide whether a user should be de-anonymized, making the system open to abuse by dishonest or compromised Verifiers.
References

