U-Prove WS-Trust Profile V1.0

Draft Revision 1

Microsoft Corporation

Author: Christian Paquin

February 2011

Summary

This specification defines a profile detailing how U-Prove tokens, defined in [UPCS], can be obtained and presented using [WS-Trust14]. This document is intended for developers and architects who wish to design applications that interoperate.

Contents

List of tables

List of figures

1 Introduction

The U-Prove technology specified in [UPCS] provides many security, privacy, and scalability benefits to identity systems: users disclose the minimal amount of claim information encoded in short- or long-lived U-Prove tokens for a particular transaction, and this without connecting to the Claim Provider at presentation time. For more information about the U-Prove technology, see [UPTO].

This specification defines how U-Prove tokens can be obtained and presented using [WS-Trust14]. Two classes of U-Prove tokens are define herein: "claim" tokens that can encode arbitrary claims, and "ID" tokens that specify globally unique, secure, and privacy-protecting persistent identifiers for users.

This document is intended for developers and architects who wish to design applications that interoperate.

1.1 U-Prove technology overview

This section summarizes the U-Prove features specified in [UPCS].

A U-Prove token (UPT) is a cryptographically protected container of claim information that is issued to a Prover (the client) by an Issuer (the Claim Provider), and presented to a Verifier (the Relying Party). Each UPT corresponds to a private key needed to present the token, and contains an Issuer's signature that attests to its origin and integrity.

A UPT is conceptually similar to a X.509 certificate or SAML assertion, with two major differences:

1. A UPT is generated jointly by the Prover and the Issuer in an interactive issuance protocol. It contains no correlation handles identifiable by the Issuer outside the certified claim values. In particular, its public key and Issuer's signature have been randomized[1] by the Prover in the issuance protocol; as such, these values are never seen by the Issuer. Consequently, the Prover cannot be tracked on the basis of these values when using the UPT, even if the Issuer and the Verifiers collude (even if they are the same entity).
2. When presenting a UPT, the Prover can hide any subset of the encoded claims, without invalidating the Issuer's signature generated on all the claims.[2] In particular, the Prover can hide all the claims (merely proving ownership of the UPT) or disclose all of them (like presenting a signed SAML assertion or a X.509 certificate).

As illustrated in Figure 1, a Prover typically gets multiple UPTs certifying the same set of claims in one instance of the issuance protocol (multiple UPTs are obtained to preserve unlinkability between Verifiers). To present any subset of the certified claims to a Verifier (immediately after issuance for on-demand tokens, or a later time for long-lived tokens), the Prover creates a presentation proof for a selected UPT by applying the corresponding private key to a cryptographic challenge.

[1] This does not affect the security of the certified claims; they cannot be forged, modified, etc.
[2] This is not achieved by obfuscation or encryption techniques; it is not possible to brute force the UPT (and the presentation proof) to learn the undisclosed values (or anything about them).

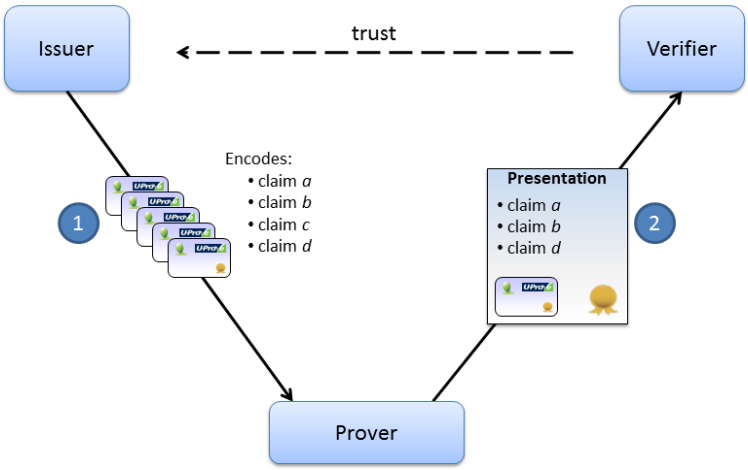

Figure 1: U-Prove token data flow

Optionally, an Issuer can issue a UPT to a Prover in such a manner that the Prover cannot use the token without the assistance of a trusted Device (e.g., a smartcard, a mobile phone, or an online server). The Device can efficiently protect multiple tokens issued by any number of Issuers, and can dynamically (i.e., at token use time) enforce policies on behalf of the Issuer, Verifiers, or third parties — all without being able to compromise the Prover's privacy and without needing to interact with the Issuer.

For more information about the U-Prove technology, see [UPTO].

2 Terminology and notation

2.1 Notational conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “RECOMMENDED”, and “MAY” in this document are to be interpreted as described in [RFC2119].

This specification uses the following syntax to define outlines for XML data:

- The syntax appears as an XML instance, but values in italics indicate data types instead of literal values.
- Characters are appended to elements and attributes to indicate cardinality:
  - “?” (0 or 1)
  - “*” (0 or more)
  - “+” (1 or more)
- The character “|” is used to indicate a choice between alternatives.
- The characters “(“ and “)” are used to indicate that contained items are to be treated as a group with respect to cardinality or choice.
- XML namespace prefixes (see Table 1) are used to indicate the namespace of the element being defined.

XML elements and Attributes defined by this specification are referred to in the text of this document using XPath 1.0 expressions.

2.2 Namespaces

The base XML namespace URI used by the definitions in this profile is as follows:

```
http://schemas.xmlsoap.org/ws/2011/02/u-prove
```

The U-Prove schema [UPS] for this namespace defines the artifacts described in this document. Table 1 lists the XML namespaces used in this document.

Prefix	XML namespace	Specification
xs	http://www.w3.org/2001/XMLSchema	[XMLSchema2]
ds	http://www.w3.org/2000/09/xmldsig#	[XMLDSIG]
up	http://schemas.xmlsoap.org/ws/2011/02/u-prove	This document
wst	http://docs.oasis-open.org/ws-sx/ws-trust/200512	[WS-Trust14]

Table 1: XML namespaces

3 U-Prove Cryptographic Specification profile

This section defines an XML syntax and processing for the artifacts defined in the U-Prove Cryptographic Specification V1.1 [UPCS].

3.1 Encoding rules

3.1.1 Binary values

The following rules define how octet string values defined in [UPCS] are encoded into XML:

- Mathematical elements (elements of $\mathbb{Z}_q$ and G_q, and the values p and q in the description of G_q) are encoded in XML elements of type `ds:CryptoBinary`. Before applying the base64 encoding, the binary value is conditionally zero-extended to make its length a multiple of 8 bits (the value 0 is zero-extended to a full 8 bits). The bytes forming the extended value are then encoded as an octet string, leading with the most-significant byte (big endian); e.g., the number 254666256150 is encoded as 0x3b4b4aaf16.
- The Issuer parameters' field S, the U-Prove token's field TI, and the presentation message MUST be canonicalized before being treated as an octet string. The octet string is obtained by taking the exclusive canonical form of the XML element with a null InclusiveNamespaces PrefixList,[3] as defined in [XML-EXC-C14N].
- Elements of type *xs:anyURI* are encoded using UTF-8 encoding. For example, the URI "http://www.contoso.com" is encoded as 0x687474703a2f2f7777772e636f6e746f736f2e636f6d.

3.1.2 Time values

All time values encoded in XML elements of type `xs:dateTime` MUST be expressed in the canonical timezone representation (i.e., UTC with 'Z' as the timezone).

3.2 Issuer parameters

3.2.1 Issuer parameters encoding

The Issuer parameters

$$\mathrm{UID}_P, (p, q, g), \mathrm{UID}_{\mathcal{H}}, (g_0, g_1, \ldots, g_n, g_t), (e_1, \ldots, e_n), (z_0, z_1, \ldots, z_n, z_t), S$$

as defined in Section 2.3.1 of [UPCS], and the optional value z_d if the Issuer supports issuance of Device-protected tokens, are encoded as follows:

[3] There is no need to preserve unused namespace prefixes as the resulting octet string is not interpreted in the cryptographic protocols.

```
<up:IssuerParameters Version="1.0" N="xs:unsignedShort">
  <up:UIDP>xs:anyURI</up:UIDP>
  <up:Group>
  (
    <up:GroupDescription GroupType="">
      <up:P>ds:CryptoBinary</up:P>
      <up:Q>ds:CryptoBinary</up:Q>
      <up:Gen>ds:CryptoBinary</up:Gen>
    </up:GroupDescription> |
    <up:GroupName>xs:anyURI</up:GroupName>
  )
  </up:Group>
  <up:HashAlgorithm>xs:anyURI</up:HashAlgorithm>
  <up:Gvalues GeneratorsName="xs:anyURI">
    <up:G>ds:CryptoBinary<up:G>+
  </up:Gvalues>
  <up:Evalues>
    <up:E>xs:unsignedbyte</up:E>*
  </up:Evalues>
  <up:Zvalues>
    <up:Z>ds:CryptoBinary</up:Z>+
  </up:Zvalues>
  <up:Zd>ds:CryptoBinary</up:Zd>?
  <up:Specification>
    <up:Issuer>xs:anyURI</up:Issuer>
    <up:NotBefore>xs:dateTime</up:NotBefore>
    <up:NotOnOrAfter>xs:dateTime</up:NotOnOrAfter>
    <up:AttributeDescriptions>
      <up:AttributeDescription ClaimType="xs:anyURI"/>*
    </up:AttributeDescriptions>
  </up:Specification>
</up:IssuerParameters>
```

The following describes the attributes and elements listed in the schema outlined above:

`/up:IssuerParameters`

This element contains a set of Issuer parameters.

`/up:IssuerParameters/@Version`

This attribute indicates the version of this specification. The value MUST be "1.0".

`/up:IssuerParameters/@N`

This attribute encodes the value n.

`/up:IssuerParameters/up:UIDP`

This element contains a URI that uniquely identifies the parameters. The value $\mathrm{UID_P}$ is defined as the UTF-8 encoding of this value.

`/up:IssuerParameters/up:Group`

This element specifies the mathematical group. It contains either a `up:GroupDescription` element specifying a group, or a `up:GroupName` element referencing a group defined externally.

`/up:IssuerParameters/up:Group/up:GroupDescription`

This element contains a group description (p, q, g).

`/up:IssuerParameters/up:Group/up:GroupDescription/up:P`

This element contains the base64 encoding of p.

`/up:IssuerParameters/up:Group/up:GroupDescription/up:Q`

This element contains the base64 encoding of q.

`/up:IssuerParameters/up:Group/up:GroupDescription/up:Gen`

This element contains the base64 encoding of g.

`/up:IssuerParameters/up:Group/up:GroupName`

This element contains a URI referencing a group description defined externally. An implementation SHOULD support the groups defined in [UPRPP]. Parties that do not understand the URI MUST reject the Issuer parameters.

`/up:IssuerParameters/up:HashAlgorithm`

This element contains a URI value for $\mathrm{UID}_{\mathcal{H}}$. It is RECOMMENDED to use well-known URIs, such as those defined in [XMLDSIG], [XMLENC] and [RFC4051]. Parties that do not understand the URI MUST reject the Issuer parameters.

`/up:IssuerParameters/up:Gvalues`

This element contains either the list $(g_0, g_1, \ldots, g_n, g_t)$, encoded in order in the $n + 2$ child elements, or if the `GeneratorsName` attribute is present, only the value g_0 encoded in one child element.

`/up:IssuerParameters/up:Gvalues/@GeneratorsName`

This optional attribute contains a URI referencing a list of $(g_1, \ldots, g_n, g_t)$ values defined externally. It MUST only be present if the `up:Gvalues` element contains only the g_0 value. An

implementation SHOULD support the Issuer generators defined in [UPRPP]. Parties that do not understand the URI MUST reject the Issuer parameters.

`/up:IssuerParameters/up:Gvalues/up:G`

This element contains the base64 encoding of a value g_i.

`/up:IssuerParameters/up:Evalues`

This element contains the list $(e_1, \dots, e_n)$, encoded in order in the n child elements. It is empty if $n = 0$.

`/up:IssuerParameters/up:Evalues/up:E`

This element contains a value e_i.

`/up:IssuerParameters/up:Zvalues`

This element contains the list $(z_0, z_1, \dots, z_n, z_t)$, encoded in order in the $n + 2$ child elements.

`/up:IssuerParameters/up:Zvalues/up:Z`

This element contains the base64 encoding of a value z_i.

`/up:IssuerParameters/up:Zd`

This optional element contains the value z_d defined in Section 2.3.2 of [UPCS]. It is only present if the Issuer supports the issuance of Device-protected tokens.

`/up:IssuerParameters/up:Specification`

The element contains the specification for the U-Prove tokens issued under these Issuer parameters. The value S is defined as the exclusive canonical form of this element with a null InclusiveNamespaces PrefixList, as defined in [XML-EXC-C14N].

`/up:Specification/up:Issuer`

This element contains an application-specific identifier for the Claim Provider.

`/up:Specification/up:NotBefore`

This element indicates the instant from which the parameters are valid. U-Prove tokens MUST only be issued and presented on or after this instant.

`/up:Specification/up:NotOnOrAfter`

This element indicates the instant at which the parameters become invalid. U-Prove tokens MUST only be issued and presented before this instant. A U-Prove presentation MAY be verified after the expiration of the parameters if it was generated within the validity period.

`/up:Specification/up:AttributeDescriptions`

This element contains the descriptions of the attributes issued using these parameters, encoded in order in the n child elements. It is empty if $n = 0$.

`/up:Specification/up:AttributeDescriptions/up:AttributeDescription`

This element contains the description of one token attribute.

`…/up:AttributeDescriptions/up:AttributeDescription/@ClaimType`

This attribute contains the unique identifier of the claim type encoded into the corresponding token attribute.

3.2.2 Creation

The Claim Provider generates and makes available a set of Issuer parameters that supports all the claims it issues. Let n be the number of claim types to support; the Issuer parameters for these n claim types MUST be generated using the procedure described in Section 2.3.1 of [UPCS]; the $\{e_i\}$ values (encoded into the `/up:IssuerParameters/up:Evalues` element) MUST all be 0x01.

The `up:Group` element MUST either contain a `up:GroupName` element that references a well-known group defined externally (e.g., in a profile such as [UPRPP]), or a `up:GroupDescription` element. If the $(g_1, \dots, g_n, g_t)$ values are defined externally (in a profile such as [UPRPP]), then the `up:Gvalues` element only encodes g_0 and the `GeneratorsName` attribute refers to the generators name; otherwise the `up:Gvalues` element contains all the generators.

3.2.3 Trust management

How Issuer parameters are transmitted to the client and the Relying Parties is out of scope of this document. Application SHOULD however make sure they are obtained in a secure and authenticated fashion, for example using a direct channel to the Claim Provider, or distributing them in a signed form using a trusted PKI certificate.

Relying Parties MUST verify that the `Specification/up:Issuer` element value is consistent with the Claim Provider's identity.

If the Issuer parameters contains the `up:Group/up:GroupDescription` element, the client and the Relying Parties MUST verify the parameters using the procedure defined in Section 2.4.1 of [UPCS].

3.3 U-Prove token

A client, acting as the Prover, obtains U-Prove tokens, as defined in Section 2.2.2 of [UPCS], from a Claim Provider using the protocol defined in Section 4.1, and presents them to Relying Parties using the protocol defined in Section 4.3.

3.3.1 Token information field

A UPT's token information field TI contains the token's expiration date and indicates if the token is Device-protected. This element appears in the first issuance response (see Section 4.2.3) and in U-Prove tokens (see Section 3.3.4); the XML fragment MUST be copied in the tokens as it was received from the

Claim Provider (to make sure that the canonicalization of the element is consistent). The element is encoded as follows (line breaks and indentation are only present here for readability):

```
<up:TI>
  <up:Gd GdName="xs:anyURI"/>?
  <up:NotOnOrAfter>xs:date</up:NotOnOrAfter>
</up:TI>
```

The `up:TI` element MUST NOT contain insignificant white spaces; the client MAY reject the element if this is the case.[4] The following describes the elements listed in the schema outlined above:

`/up:TI`

This element contains the token information field. The value TI is defined as the exclusive canonical form of this element with a null InclusiveNamespaces PrefixList, as defined in [XML-EXC-C14N].

`/up:TI/up:Gd`

This optional element specifies the Device generator g_d. It MUST only be present if the token is Device-protected (if $\mathbf{d}$ = true).

`/up:TI/up:Gd/@GdName`

This attribute contains a URI referencing the Device generator g_d of the Device protecting this token, defined externally. An implementation SHOULD support the Device generators defined in [UPRPP] and using their URN name. Parties that do not understand the URI MUST reject the token. Parties MUST reject the token if the referenced g_d is not part of the group specified in the corresponding Issuer parameters.

`/ up:TI/up:NotOnOrAfter`

This element specifies the instant in UTC (see Section 3.1.2) when the U-Prove token becomes invalid. This date MUST be within the validity period of the corresponding Issuer parameters. Because this value might be used as a correlation handle to trace users, a client MAY reject values that might not meet its privacy settings.[5]

3.3.2 Prover information field

The Prover information field PI is unused in this profile and is always null, it is not represented in the token encoding.

3.3.3 Token attributes

A list of U-Prove token attributes $\{A_i\}$ appears in the first issuance response (see Section 4.2.3) and in presentation tokens (see Section 4.3.1). This element is encoded as follows:

[4] White spaces could be used to "mark" tokens to defeat the untraceability property.
[5] The more coarse-grained the values, the more privacy preserving they are. For example, granularity to the month (fixing the day field to 1, and the time to 00:00:00) is statistically more neutral than granularity to the day.

```
<up:Attributes>
  <up:Attribute Index="xs:short">…</up:Attribute>+
</up:Attributes>
```

The following describes the attributes and elements listed in the schema outlined above:

`up:Attributes`

This element contains a list of U-Prove token attributes $\{A_i\}$.

`Up:Attributes/up:Attribute`

This element contains a U-Prove token attribute A_i. If the element is empty, then the value A_i is null; otherwise it is defined as the UTF-8 encoding of the element's content.

`Up:Attributes/up:Attribute/@Index`

This attribute indicates the index i of the U-Prove token attribute A_i. It MUST be a value greater than 0 and no greater than the number n of attributes in the token. The indices MUST appear in sorted increasing order.

3.3.4 U-Prove token encoding

A U-Prove token

$$\mathrm{UID_P}, h, TI, \sigma'_z, \sigma'_c, \sigma'_r, \mathbf{d}$$

as defined in Section 2.4.3 of [UPCS], is encoded as follows:

```
<up:Token Version="1.0">
  <up:UIDP>xs:anyURI</up:UIDP>
  <up:H>ds:CryptoBinary</up:H>
  <up:TI>…</up:TI>
  <up:SigmaZPrime>ds:CryptoBinary</up:SigmaZPrime>
  <up:SigmaCPrime>ds:CryptoBinary</up:SigmaCPrime>
  <up:SigmaRPrime>ds:CryptoBinary</up:SigmaRPrime>
</up:Token>
```

The following describes the attributes and elements listed in the schema outlined above:

`/up:Token`

This element contains a U-Prove token.

`/up:Token/@Version`

This attribute indicates the version of this specification. The value MUST be "1.0".

`/up:Token/up:UIDP`

This element contains a URI that identifies the Issuer parameters. The value $\mathrm{UID_P}$ is defined as the UTF-8 encoding of this value.

`/up:Token/up:H`

This element contains the base64 encoding of the value h.

`/up:Token/up:TI`

This element contains the token information field, see Section 3.3.1.

`/up:Token/up:SigmaZPrime`

This element contains the base64 encoding of the value σ_z'.

`/up:Token/up:SigmaCPrime`

This element contains the base64 encoding of the value σ_c'.

`/up:Token/up:SigmaRPrime`

This element contains the base64 encoding of the value σ_r'.

4 WS-Trust profile

This section specifies how UPTs are obtained and presented using [WS-Trust14].

4.1 U-Prove token classes

A U-Prove token can encode any number of claims of any type. In many cases, it is desirable to establish a persistent identifier at a Relying Party, for the user to be recognized in repeat visits. The token identifier (see Section 2.3.7 of [UPCS]) of a UPT can be used to generate such a persistent identifier in a secure and privacy-protecting way: the Claim Provider never sees this value (supporting the untraceability property), and only the user can prove ownership of this identifier.

Since claim values might change over time and persistent identifiers must be constant, two classes of UPTs are defined.

- A "claim" UPT encodes the claim values specified by the corresponding Issuer parameters, and typically has a short validity period. A "claim" UPT is used either once (for an anonymous presentation) or repeatedly with a particular Relying Party (for a pseudonymous presentation).
- An "ID" UPT encodes no claim values and is used to compute a persistent identifier; it has a long validity period,[6] and is reused in repeat visits to a Relying Party.

Both classes of tokens can be issued and presented in a similar fashion.

4.2 U-Prove issuance protocol

The U-Prove issuance protocol, specified in Section 2.3 of [UPCS], consists of a 4-message exchange between a client (the Prover) and a Claim Provider (the Issuer). At the end of the protocol, the client generates multiple UPTs, each of which encodes an identical set of claim values.

The protocol is performed over WS-Trust using the challenge pattern and the issuance binding to support the 4-message exchange. In one run of the protocol, multiple instances of the U-Prove issuance protocol can be performed, each to obtain a batch of UPTs of a particular class. For example, a client could obtain 100 "claim" UPTs, and 20 "ID" UPTs. Figure 2 illustrates the message exchange.

[6] Changing the "ID" UPT changes the corresponding token identifier value. It is therefore desirable to keep them as long as possible.

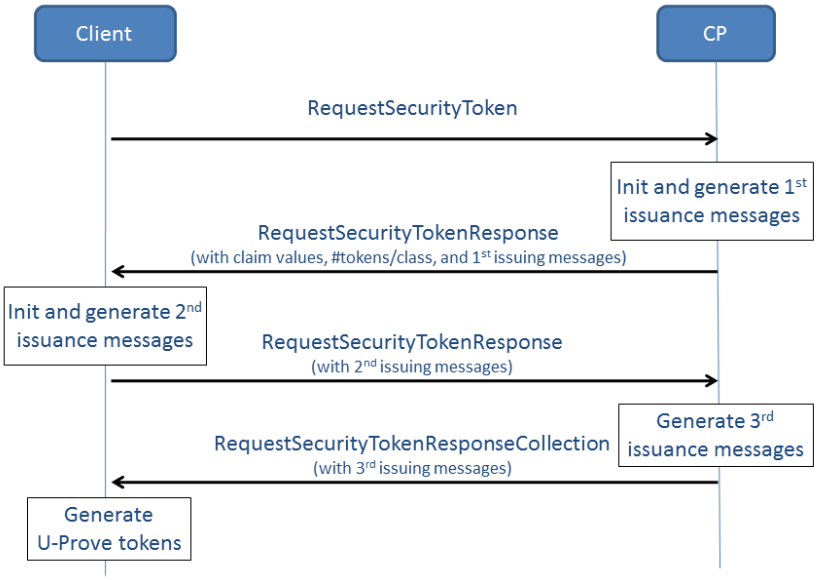

Figure 2: WS-Trust challenge pattern issuance

A summary of the exchange follows. The client initiates the issuance by sending to the Claim Provider a token request.

The Claim Provider authenticates the client, validates the request, initiates the U-Prove issuance protocol instances, and returns the first token response containing the number and contents of the UPTs of each class and the corresponding first issuance messages.

The client validates the response, initiates the U-Prove issuance protocol instances, and returns the second token response containing the second message of each issuance instance.

The Claim Provider replies with the third and final token response containing the third message of each issuance instance. Finally, for each issuance instance, the client generates the UPTs using the exchanged cryptographic material.

4.2.1 Message exchange

The message exchange uses the issuance binding mechanism described in [WS-Trust14]. The following sections describe each message in detail. Some notes:

- The endpoint to contact, and its authentication requirements, are application specific. The issuance protocol SHOULD be done over a secure channel to protect the confidentiality of the claim values.
- The final response does not contain a `wst:RequestedSecurityToken` element; the client generates the UPTs based on the exchanged cryptographic material.
- Since the exchange is multi-leg, the Claim Provider must keep the cryptographic state of each issuance instance between the message exchanges. The state MUST be securely protected (as strongly as the Issuer's private key). As explained in Section 2.3 of [UPCS], the Issuer state MUST

NOT be sent (even encrypted) to the client to implement a stateless Issuer. Farm deployments must either use "sticky sessions" or use a database to commit the Issuer state in a secure manner.

- When Device-protected tokens are issued, the client and the Claim Provider must obtain the Device parameters g_d and h_d to start their computations. To optimize the message exchange, the Device parameters are sent from the client to the Claim Provider in the first request; the client obtains them from the Device. If the Device parameters are inconsistent with the Issuer parameters, then the Claim Provider MUST return an error. The client also needs an extra value ζ_d computed by the Device using the value z_d encoded in the Issuer parameters of the Claim Provider.

4.2.2 Token request

The client initiates the U-Prove issuance protocol by sending a `wst:RequestSecurityToken` (RST) message to the Claim Provider. The request MAY specify a number of UPTs to be issued of each class; however the Claim Provider makes the final decision on how many to issue. If the tokens are to be Device-protected, then the request MAY contain the Device parameters; alternatively, these parameters MAY already be known to the Claim Provider. The syntax for the element is:

```
<wst:RequestSecurityToken Context="…">
  <wst:TokenType>…/2011/02/u-prove/token</wst:TokenType>?
  <wst:RequestType>…/Issue</wst:RequestType>
  <up:Instances>
    <up:Instance TokenClass="xs:token" TokenCount="xs:unsignedShort">+
  </up:Instances>?
  <up:DeviceParameters Id="xs:anyURI">
    <up:Gd GdName="xs:anyURI"/>
    <up:Hd>…</up:Hd>
    <ds:Signature>…</ds:Signature>
  </up:DeviceParameters>?
</wst:RequestSecurityToken>
```

The following describes the attributes and elements listed in the schema outlined above:

`/wst:RST/@Context`

The request MUST contain a `Context` attribute. The Claim Provider MUST reject requests with context values already in use.

`/wst:RST/wst:TokenType`

The value of this optional element MUST be "http://schemas.xmlsoap.org/ws/2011/02/u-prove/token".

`/wst:RST/wst:RequestType`

The value of this element MUST be "http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue".

`/wst:RST/up:Instances`

This optional element describes the requested U-Prove issuance instances.

`/wst:RST/up:Instances/up:Instance`

This element describes one requested U-Prove issuance instance.

`/wst:RST/up:Instances/up:Instance/@TokenClass`

This attribute describes the requested class of the UPTs. It MUST be either “claim” or “ID”. The Claim Provider MAY refuse to issue tokens of a particular class depending on its policy.

`/wst:RST/up:Instances/up:Instance/@TokenCount`

This attribute describes the number of requested UPTs of the class specified by the `TokenClass` attribute. If 0, then the client explicitly requests no token of this class. The Claim Provider MAY issue a different number of tokens depending on its policy.

`/wst:RST/up:DeviceParameters`

This optional element contains the Device parameters. It MUST be present only if the tokens to issue are to be Device-protected.

`/wst:RST/up:DeviceParameters/@Id`

Identifier for this element, referenced by the `ds:Signature` element.

`/wst:RST/up:DeviceParameters/up:Gd`

This element specifies the Device generator g_d. This element MUST be copied in the `up:TI` element of the issued UPTs.

`/wst:RST/up:DeviceParameters/up:Gd/@GdName`

This attribute contains a URI referencing a Device generator g_d defined externally. An implementation SHOULD support the Device generators defined in [UPRPP]. Claim Providers that do not understand the URI MUST reject the request. The Claim Provider MUST reject the request if the referenced g_d is not part of the group specified in its Issuer parameters.

`/wst:RST/up:DeviceParameters/up:Hd`

This element contains the base64 encoding of the Device public key h_d.

`/wst:RST/up:DeviceParameters/ds:Signature`

This element contains the enveloped signature that protects the value of the Device parameters. The signature is created by an entity trusted by the Claim Provider, typically the Device issuer; the Claim Provider MUST reject the request if it cannot validate the signature. The following rules apply:

- The signature MUST contain only one `ds:Reference` element with a `URI` attribute set to the value of the `up:DeviceParameters/@Id` attribute.
- It is RECOMMENDED to use the RSA signature method (identified by the URI http://www.w3.org/2001/04/xmldsig-more#rsa-sha256).
- The signature MUST only use the exclusive canonicalization transform (identified by the URI http://www.w3.org/2001/10/xml-exc-c14n#) and the enveloped signature transform (identified by the URI http://www.w3.org/2000/09/xmldsig#envelopedsignature).
- It is RECOMMENDED to use SHA-256 (identified by the URI http://www.w3.org/2001/04/xmlenc#sha256).

4.2.3 First response

After the Claim Provider authenticates the client and validates the token request, it initiates the U-Prove issuance protocol instances using the claim values corresponding to the authenticated user. It then responds by sending a `wst:RequestSecurityTokenResponse` (RSTR) message to the client. The syntax for the element is:

```
<wst:RequestSecurityTokenResponse Context="…">
  <wst:TokenType>…/2011/02/u-prove/token</wst:TokenType>
  <up:Instances>
    <up:Instance TokenClass="xs:token" TokenCount="xs:unsignedShort">
      <up:Attributes>…</up:Attributes>?
      <up:TI>…</up:TI>
      <up:IssuanceMessage1>
        <up:SigmaPair>
          <up:SigmaA>ds:cryptoBinary</up:SigmaA>
          <up:SigmaB>ds:cryptoBinary</up:SigmaB>
        </up:SigmaPair>+
      </up:IssuanceMessage1>
    </up:Instance>+
  </up:Instances>
  <up:UIDP>…</up:UIDP>
</wst:RequestSecurityTokenResponse>
```

The following describes the attributes and elements listed in the schema outlined above:

`/wst:RSTR/@Context`

This attribute value MUST match the initial request's `Context` value.

`/wst:RSTR/wst:TokenType`

The value of this element MUST be "http://schemas.xmlsoap.org/ws/2011/02/u-prove/token".

`/wst:RSTR/up:Instances`

This element contains a sequence of U-Prove issuance instances. The Claim Provider decides how to form the `up:Instances` element given the token request.

`/wst:RSTR/up:Instances/up:Instance`

This element describes one U-Prove issuance instance.

`/wst:RSTR/up:Instances/up:Instance/@TokenClass`

This attribute describes the class of the issued UPTs. It MUST be either "claim" or "ID".

`/wst:RSTR/up:Instances/up:Instance/@TokenCount`

This attribute describes the number of issued UPTs of the class specified by the `TokenClass` attribute. The value MUST be greater than zero.

`/wst:RSTR/up:Instances/up:Instance/up:Attributes`

This optional element MUST be present if and only if `up:Instance/@TokenClass` is set to "claim". It contains the list of attribute values for a "claim" UPT.[7] There MUST be as many child elements as the number of claims supported by the Issuer parameters, see Section 3.3.3.

`/wst:RSTR/up:Instances/up:Instance/up:TI`

This element contains the token information field, see Section 3.3.1 for details.

`/wst:RSTR/up:Instances/up:Instance/up:IssuanceMessage1`

This element contains the first message of the issuance instance containing ordered `up:SigmaPair` elements, one per issued token. There MUST be as many child elements as the number specified in the `up:Instance/@TokenCount` attribute.

`/wst:RSTR/up:Instances/up:Instance/up:IssuanceMessage1/up:SigmaPair`

This element contains a (σ_a, σ_b) pair.

`/wst:RSTR/up:Instances/up:Instance/up:IssuanceMessage1/up:SigmaA`

This element encodes the value σ_a.

`/wst:RSTR/up:Instances/up:Instance/up:IssuanceMessage1/up:SigmaB`

This element encodes the value σ_b.

[7] All the attributes of an "ID" UPT are implicitly null; this element is therefore absent.

`/wst:RSTR/up:UIDP`

This element contains the UID of Issuer parameters used by the Claim Provider, as defined in Section 3.2.1. This element MUST be copied into the corresponding element of issued UPTs.

4.2.4 Second response

The client validates the first token response. It initiates the U-Prove issuance protocol and then responds by sending a `wst:RequestSecurityTokenResponse` message to the Claim Provider. The syntax for the element is:

```
<wst:RequestSecurityTokenResponse Context="…">
  <up:Instances>
    <up:Instance>
      <up:IssuanceMessage2>
        <up:SigmaC>ds:cryptoBinary</up:SigmaC>+
      </up:IssuanceMessage2>
    </up:Instance>+
  </up:Instances>
</wst:RequestSecurityTokenResponse>
```

The following describes the attributes and elements listed in the schema outlined above:

`/wst:RSTR/@Context`

This attribute value MUST match the initial request's `Context` value.

`/wst:RSTR/up:Instances`

This element contains a sequence of U-Prove issuance instances. There MUST be a matching number of ordered child `up:Instance` elements as in the first response.

`/wst:RSTR/up:Instances/up:Instance`

This element describes one U-Prove issuance instance.

`/wst:RSTR/up:Instances/up:Instance/up:IssuanceMessage2`

This element contains the second message of the issuance instance. There MUST be a matching number of child elements as in the first response's corresponding `up:IssuanceMessage1` element; the i^{th} `up:SigmaC` element corresponds to the first response's i^{th} `up:SigmaPair` element.

`/wst:RSTR/up:Instances/up:Instance/up:IssuanceMessage2/up:SigmaC`

This element encodes the value σ_c.

4.2.5 Third response

The Claim Provider validates the second token response and responds by sending a `wst:RequestSecurityTokenResponseCollection (RSTRC)` message containing one `wst:RequestSecurityTokenResponse (RSTR)` to the client. The syntax for the element is:

```
<wst:RequestSecurityTokenResponseCollection>
  <wst:RequestSecurityTokenResponse Context="…">
    <up:Instances>
      <up:Instance>
        <up:IssuanceMessage3>
          <up:SigmaR>ds:cryptoBinary</up:SigmaR>+
        </up:IssuanceMessage3>
      </up:Instance>+
    </up:Instances>
  </wst:RequestSecurityTokenResponse>
</wst:RequestSecurityTokenResponseCollection>
```

The following describes the attributes and elements listed in the schema outlined above:

`/wst:RSTRC/wst:RSTR/@Context`

This attribute value MUST match the request's `Context` value.

`/wst:RSTRC/wst:RSTR/up:Instances`

This element contains a sequence of U-Prove issuance instances. There MUST be a matching number of ordered child `up:Instance` elements as in the first response.

`/wst:RSTRC/wst:RSTR/up:Instances/up:Instance`

This element describes one U-Prove issuance instance.

`/wst:RSTRC/wst:RSTR/up:Instances/up:Instance/up:IssuanceMessage3`

This element contains the third message of the issuance instance. There MUST be a matching number of child elements as in the second response's corresponding `up:IssuanceMessage2` element; the i^{th} `up:SigmaR` element corresponds to the second response's i^{th} `up:SigmaC` element.

`/wst:RSTRC/wst:RSTR/up:Instances/up:Instance/up:IssuanceMessage3/up:SigmaR`

This element encodes the value σ_r.

4.2.6 Example

This example illustrates the issuance of three claim UPTs and five ID UPTs.

The Issuer parameters:

```
<up:IssuerParameters Version="1.0">
```

```
  <up:UIDP>http://fabrikam.com/ip1</up:UIDP>
  <up:Group>
    <up:GroupDescription>
      <up:P>mARqW5gJ2hGJNgPYu8...</up:P>
      <up:Q>//u9ZguUQSrmHq2cKQa...</up:Q>
      <up:Gen>ZY/qS3f2qmXBzrjKwhn...</up:Gen>
    </up:GroupDescription>
  </up:Group>
  <up:HashAlgorithm>http://www.w3.org/2001/04/xmlenc#sha256</up:HashAlgorithm>
  <up:Gvalues>
    <up:G>he4yzdZEKS8+oj8wiTdxn6D...</up:G>
    <up:G>fuNsPDWWPZePI3ohVqcEKS...</up:G>
    <up:G>NL+6MxBT/XfRllNdOqbRXL...</up:G>
    <up:G>MlzSghqWH3dB39IRCWTZG...</up:G>
    <up:G>Ai0Ht/qC9+UnwExjtvFjr8b...</up:G>
  </up:Gvalues>
  <up:Evalues>
    <up:E>1</up:E>
    <up:E>1</up:E>
    <up:E>1</up:E>
  </up:Evalues>
  <up:Zvalues>
    <up:Z>CFssS3ZkJ3Syv57PVJwlTO...</up:Z>
    <up:Z>ZRV6X5gC8k9JlWGYdlrRHj...</up:Z>
    <up:Z>dzZ6uDK8SuicGFDKPB6faZ...</up:Z>
    <up:Z>OvuO+Lab0s1LvfdbJeGC1...</up:Z>
    <up:Z>kl0JDCO8jmLH4riKxlNjUM...</up:Z>
  </up:Zvalues>
  <up:Specification>
    <up:Issuer>http://fabrikam.com</up:Issuer>
    <up:NotBefore>20090416T00:40:02.781Z</up:NotBefore>
    <up:NotOnOrAfter>20100401T12:00:00Z</up:NotOnOrAfter>
    <up:AttributeDescriptions>
      <up:AttributeDescription
            ClaimType="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"/>
      <up:AttributeDescription
            ClaimType="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname"/>
      <up:AttributeDescription
            ClaimType="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/gender"/>
    </up:AttributeDescriptions>
  </up:Specification>
</up:IssuerParameters>
```

The token request:

```
<wst:RequestSecurityToken Context="uuid54a545f734574c59b93d02445f28c495">
  <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</wst:RequestType>
  <wst:TokenType>http://schemas.xmlsoap.org/ws/2011/02/uprove/token
     </wst:TokenType>
</wst:RequestSecurityToken>
```

Figure 3: Token request example

The first issuance response:

```
<wst:RequestSecurityTokenResponse Context="uuid54a545f734574c59b93d02445f28c495">
  <wst:TokenType>http://schemas.xmlsoap.org/ws/2011/02/uprove/token</wst:TokenType>
  <up:Instances>
    <up:Instance TokenClass="claim" TokenCount="3">
      <up:Attributes>
        <up:Attribute Index="1">bob@contoso.com</up:Attribute>
        <up:Attribute Index="2">Bob</up:Attribute>
        <up:Attribute Index="3">1</up:Attribute>
      </up:Attributes>
```

```
    <up:TI><up:NotOnOrAfter>20100401T00:00:00Z</up:NotOnOrAfter></up:TI>
    <up:IssuanceMessage1>
     <up:SigmaPair>
      <up:SigmaA>YooTfNOgFbRPjT34UW0p...</up:SigmaA>
      <up:SigmaB>cM/BGFUeLTNLtPJbUfw8...</up:SigmaB>
     </up:SigmaPair>…</up:SigmaPair>
     <up:SigmaPair>…</up:SigmaPair>
    </up:IssuanceMessage1>
   </up:Instance>
   <up:Instance TokenClass="ID" TokenCount="5">
    <up:TI><up:NotOnOrAfter>20100401T00:00:00Z</up:NotOnOrAfter></up:TI>
    <up:IssuanceMessage1>
     <up:SigmaPair>
      <up:SigmaA>ZxOeoVV5nM7HOBiYLZlA...</up:SigmaA>
      <up:SigmaB>QlAELZHB3VP1WtCGE5DF...</up:SigmaB>
     </up:SigmaPair>
     <up:SigmaPair>…</up:SigmaPair>
     <up:SigmaPair>…</up:SigmaPair>
     <up:SigmaPair>…</up:SigmaPair>
     <up:SigmaPair>…</up:SigmaPair>
    </up:IssuanceMessage1>
   </up:Instance>
  </up:Instances>
"" <up:UIDP>http://fabrikam.com/ip1</up:UIDP>
""""""</wst:RequestSecurityTokenResponse>
```

Figure 4: First issuance response example

The second issuance response:

```
<wst:RequestSecurityTokenResponse Context="uuid54a545f734574c59b93d02445f28c495">
 <up:Instances>
  <up:Instance>
   <up:IssuanceMessage2>
    <up:SigmaC>WukyBnOt1J6Qzi8Qeo+G5a4SC0Tfi03bh2XMCjDjKHM=</up:SigmaC>
    <up:SigmaC>iMbTiKEdvEJyuF9L6ATZzJtx3L4WgPO3iDMZ0eKOlwg=</up:SigmaC>
    <up:SigmaC>HQpD6TjEpJyTspSPIH2GsweQdT6pzzUNyHHB37QpyZY=</up:SigmaC>
   </up:IssuanceMessage2>
  </up:Instance>
  <up:Instance>
   <up:IssuanceMessage2>
    <up:SigmaC>STK4NVt0c1MuK+SnvMgnEG2b37+DQLIYLGoWM59sih4=</up:SigmaC>
    <up:SigmaC>kgjIEFvJXhd8ZeOT24nx520Ua+S3r2Z8IpzlySC6H54=</up:SigmaC>
    <up:SigmaC>UnwSYrjV22KFSUvTHsBk5lmRfYXmymDprV3RmrW5a4w=</up:SigmaC>
    <up:SigmaC>q4xkz+Yolu9taHRWokWm696NPwccj4Ns6LunhUpVrjM=</up:SigmaC>
    <up:SigmaC>jR+e2P5NctWlU49oi5yl0QBZ3vadxxGTXHr6MvDQr/I=</up:SigmaC>
   </up:IssuanceMessage2>
  </up:Instance>
 </up:Instances>
</wst:RequestSecurityTokenResponse>
```

Figure 5: Second issuance response example

The third issuance response:

```
<wst:RequestSecurityTokenResponseCollection>
 <wst:RequestSecurityTokenResponse Context="uuid54a545f734574c59b93d02445f28c495">
  <up:Instances>
   <up:Instance>
    <up:IssuanceMessage3>
     <up:SigmaR>XRzG4sEM4jG+79K/y7CNujhLNIfMf7fN/2u9Gah4VvI=</up:SigmaR>
     <up:SigmaR>8/zXLatrIJA0hdBmGA0qY6qZQYGRUBPPFchUMcuXbl0=</up:SigmaR>
     <up:SigmaR>9BbLS/HBb2C4qu3EV584vL7lkGSeIZlBmQTYGrM2/fE=</up:SigmaR>
    </up:IssuanceMessage3>
   </up:Instance>
   <up:Instance>
```

```
    <up:IssuanceMessage3>
     <up:SigmaR>Za1SqV9FhCaqU/wf+LSDfN0C5a8BXHBziygp3hCn+lg=</up:SigmaR>
     <up:SigmaR>wry2vKIif+scfM+MwZ43PW3QdmlwY0Wwe2phQfsWT10=</up:SigmaR>
     <up:SigmaR>RyEpz9EEVZkM+OKxflgAweGNE6+bBTMtWevxGGfaSl8=</up:SigmaR>
     <up:SigmaR>RGRhSZgKikVB5TEEk3KrZC7W0euOpTsumFLQgcnQ95o=</up:SigmaR>
     <up:SigmaR>CekFMP8dKKq72Y0mi4SnmuTSjMDd3qGbE6domnkTXyQ=</up:SigmaR>
    </up:IssuanceMessage3>
   </up:Instance>
  </up:Instances>
 </wst:RequestSecurityTokenResponse>
</wst:RequestSecurityTokenResponseCollection>
```

Figure 6: Third issuance message example

4.3 U-Prove presentation protocol

To present a U-Prove token, the client creates a U-Prove presentation token and sends it to the Relying Party. One presentation token MAY hold presentations from one or more UPTs (e.g., from two “claim” UPTs and one “ID” UPT), each being used to sign a message. The message, specified by either the Prover or the Verifier, is composed of a cryptographic challenge and application-specific data. If a token is Device-protected, then the corresponding Device must interact with the client before presenting the token.

4.3.1 Presentation Token

A U-Prove presentation token is a WS-Trust security token generated by the client containing presentations of one or more UPTs.

The U-Prove presentation token is identified by the following URI:

```
http://schemas.xmlsoap.org/ws/2011/02/u-prove/token
```

The syntax for the element is:

```
<up:PresentationToken Version="1.0">
  <up:Message>
    <up:Nonce>xs:string</up:Nonce>
    <up:SignedData>xs:string</up:SignedData>?
  </up:Message>
  <up:Presentations>
    <up:Presentation TokenClass="xs:token">
      <up:Token>…</up:Token>
      <up:SubsetProof>
        <up:Attributes>…</up:Attributes>?
        <up:A>xs:base64Binary</up:A>
        <up:R0>ds:cryptoBinary</up:R0>
        <up:Responses>
          <up:R>ds:crypto64Binary</up:R>+
        </up:Responses>?
        <up:Rd>ds:cryptoBinary</up:Rd>?
      </up:SubsetProof>
    </up:Presentation>+
  </up:Presentations>
</up:PresentationToken>
```

The following describes the attributes and elements listed in the schema outlined above:

`/up:PresentationToken`

This element contains a U-Prove presentation token.

`/up:PresentationToken/Version`

This attribute indicates the version of this specification. The value MUST be "1.0".

`/up:PresentationToken/up:Message`

This element contains the message that will be signed by a UPT's private key to generate a presentation proof. The U-Prove presentation protocol message m is the exclusive canonical form of the element with a null InclusiveNamespaces PrefixList, as defined in [XML-EXC-C14N].

`/up:PresentationToken/up:Message/up:Nonce`

This element contains a cryptographic challenge. If specified by the Relying Party, it SHOULD be unpredictable. If specified by the user, it MUST be the UTC representation of the presentation token creation time, followed by a comma, followed by a unique identifier of the Relying Party (for example, "20090427T22:11:57.444Z,https://www.contoso.com").

`/up:PresentationToken/up:Message/up:SignedData`

This optional element contains application-specific data that will be signed by the user. The client SHOULD display this text to the user before creating the presentation token.

`/up:PresentationToken/up:Presentations`

This element contains one or more token presentations.

`/up:PresentationToken/up:Presentations/up:Presentation`

This element contains one token presentation.

`/up:PresentationToken/up:Presentations/up:Presentation/@TokenClass`

This attribute specifies the class of the presented UPT. It MUST be either “claim” or “ID”.

`/up:PresentationToken/up:Presentations/up:Presentation/up:Token`

This element contains the UPT of the specified class used to generate the presentation, see Section 3.3.4 for details.

`…/up:Presentation/up:SubsetProof`

This element contains a subset presentation proof $\{A_i\}_{i\in D}, a, r_0, \{r_i\}_{i\in U}$, as defined in Section 2.4 of [UPCS]. The total number of `up:Attributes/up:Attribute` and `up:Responses/up:R` elements MUST match the number of attributes in the corresponding token.

`…/up:Presentation/up:SubsetProof/up:Attributes`

This optional element contains the list of disclosed attributes $\{A_i\}_{i\in D}$. If no attributes are disclosed, or if the `up:Presentation/@TokenClass` has value “ID”, then it MUST be absent; see Section 3.3.3 for details.

`…/up:Presentation/up:SubsetProof/up:A`

This element encodes the value a.

`…/up:Presentation/up:SubsetProof/up:R0`

This element encodes the value r_0.

`…/up:Presentation/up:SubsetProof/up:Responses`

This optional element contains the ordered list of responses $\{r_i\}_{i\in U}$ for the undisclosed elements. It MUST be absent if all the attributes are disclosed.

`…/up:Presentation/up:SubsetProof/up:Responses/up:R`

This element contains a response for an undisclosed attribute.

`…/up:Presentation/up:SubsetProof/up:Rd`

This optional element encodes the value r_d; it is only present if the token is Device-protected (i.e., if the UPT’s `up:TI` element contains a `up:Gd` element).

4.3.2 Creating a U-Prove presentation token

A presentation token is created by the client. The following steps MUST be followed.

1. The client selects one or more UPTs to present to the Relying Party:
    a. If a persistent identifier is not requested, then a set of fresh, unassociated "claim" UPTs are selected that satisfy the Relying Party's policy. To maintain unlinkability among the UPTs, they SHOULD be deleted after use.
    b. If a persistent identifier is requested, then an existing pair of "claim" and "ID" UPTs associated with the Relying Party is selected. If none is found, then a fresh pair of "claim" and "ID" UPTs is selected and associated with the Relying Party.
    c. In any case, if a UPT of a specific class is not available, the client MUST either obtain a fresh batch from the Claim Provider using the protocol defined in Section 4.1 or fail.
2. If the message is specified in by Relying Party (e.g., in its policy), it is used directly. If not, the `up:PresentationToken/up:Message/up:Nonce` element is set to the current time in UTC format (see Section 3.1.2), followed by a comma, followed by the Relying Party identifier (e.g., URL where the presentation token is sent).
3. For each selected UPT, one presentation proof is generated using the protocol defined in Section 2.5 of [UPCS]:
    a. For a "claim" UPT, the list of disclosed attributes is set from the corresponding set of disclosed claims. For an "ID" token, all the attributes (null) are disclosed.
    b. The message m is obtained as described in Section 4.3.1.
    c. If the UPT is Device-protected, then:
        i. The message m_d is unused in this profile and MUST be set to null.
        ii. The client interacts with the Device to compute the response r_d and encodes it in the presentation proof.
4. Each generated presentation proof is encoded in a `up:PresentationToken/up:Presentations/up:Presentation` element.

4.3.3 Verifying a U-Prove presentation token

A presentation token can be verified by the Relying Party who requested the presentation or by an auditor at a later time. For a presentation to be deemed valid all the following checks MUST be valid, otherwise the presentation token MUST be rejected.

1. If the `up:Message` is specified by the Relying Party, it MUST match the one specified in the presentation token. If not, the message is parsed to obtain a presentation time and a Relying Party Identifier; the time MUST be within an acceptable range from the time the token was received (the current time for an immediate verification), and the Relying Party identifier MUST identify the Relying Party.
2. For each `up:Presentation` element in the `up:Presentations` element:
    a. Validate the UPT class. The value of the `TokenClass` attribute MUST be either "claim" or "ID".
    b. Verify the UPT specified in the `up:Token` element:

i. The `up:Token/up:UIDP` value MUST correspond to a trusted set of Issuer parameters. Moreover, the Issuer parameters' `up:Specification/up:Issuer` value MUST meet the Relying Party's issuer requirement.

ii. The `up:Token/up:TI/up:NotOnOrAfter` value MUST be the same or before the Issuer parameters' `up:Specification/up:NotOnOrAfter` value and MUST be later than the time when the token was received. Moreover, the time when the token was received MUST be the same or after the Issuer parameters' `up:Specification/up:NotBefore` value.

c. Verify the subset proof specified in the `up:SubsetProof` element:

i. Verify the presentation proof following the procedure defined in Section 2.5 of [UPCS] using the Issuer parameters identified in the `up:Token/up:UIDP` element, the UPT defined in the `up:Token` element, the message m obtained as described in Section 4.3.1, and if the UPT is Device-protected, the message m_d set to null and g_d obtained from the UPT's `up:TI` element. The UPT's signature verification MUST NOT be skipped.

ii. If the `up:Message` is specified by the client, verify that the proof is fresh by checking if the value `up:A` was previously received. It is sufficient to keep the set of `up:A` values indexed by the value of the timestamp encoded in the `up:Message/up:Nonce` element for as long as the timestamp is deemed valid.

d. Extract the presented claims:

i. If the value of the `up:TokenClass` attribute is "claim", each `up:SubsetProof/up:Attributes/up:Attribute` element encodes the value for a claim of the type defined in the `up:Attribute/@Index`[th] element of the Issuer parameters' `up:Specification/up:AttributeDescriptions/ up:AttributeDescription/@ClaimType` attribute.

ii. If the value of the `up:TokenClass` attribute is "ID", the persistent identifier value is obtained by computing the base64 encoding of the token identifier of the "ID" UPT, as defined in Section 2.2.6 of [UPCS].

4.3.4 Examples

The following example lists a presentation token for Contoso.com containing a presentation of one claim UPT (disclosing one attribute), and one ID UPT (both obtained from Fabrikam) using a message specified by the client.

```
<up:PresentationToken Version="1.0" xmlns:up="http://schemas.xmlsoap.org/ws/2011/02/uprove">
 <up:Message>
  <up:Nonce>20090427T22:11:57.444Z,https://www.contoso.com </up:Nonce>
 </up:Message>
 <up:Presentations>
  <up:Presentation TokenClass="claim">
   <up:Token Version="1.0">
    <up:UIDP>http://fabrikam.com/ip1</up:UIDP>
    <up:H>Vuv9TIPe+rcAPxcpX20FjwZut6aQ...</up:H>
    <up:TI><up:NotOnOrAfter>20100401T00:00:00Z</up:NotOnOrAfter></up:TI>
```

```
      <up:SigmaZPrime>Z/cSeGXhvcjTIAQw36fq...</up:SigmaZPrime>
      <up:SigmaCPrime>Pjwa3US2c1U99TOMRE...</up:SigmaCPrime>
      <up:SigmaRPrime>0WJHTMjiwAq8Ganeyeu...</up:SigmaRPrime>
    </up:Token>
    <up:SubsetProof>
      <up:Attributes>
        <up:Attribute Index="2">Bob</up:Attribute>
      </up:Attributes>
      <up:A>MszDfj0WmUghDdEv8rIbPQBwgN2s...</up:A>
      <up:R0>KWVDRNEyJG19p6AfuDkwMUhiwQ...</up:R0>
      <up:Responses>
        <up:R>fRvB3Fni09vkgnHGd8m6Ofxk5RiKSk...</up:R>
        <up:R>Tm8XfvpZq5+9iVkdtRntk9vKGniwz7...</up:R>
      </up:Responses>
    </up:SubsetProof>
  </up:Presentation>
  <up:Presentation TokenClass="ID">
    <up:Token Version="1.0">
      <up:UIDP>http://fabrikam.com/ip1</up:UIDP>
      <up:H>j5ep7KiPuxdODBZOD8nB83eUgJcyib...</up:H>
      <up:TI><up:NotOnOrAfter>20100401T00:00:00Z</up:NotOnOrAfter></up:TI>
      <up:SigmaZPrime>Hs5WUOkTCcYDACffscK9Zl...</up:SigmaZPrime>
      <up:SigmaCPrime>tfmzhCjnNfM2mS/RsnMxxa...</up:SigmaCPrime>
      <up:SigmaRPrime>X/almkFwx4r3lNM93UsCAQ...</up:SigmaRPrime>
    </up:Token>
    <up:SubsetProof>
      <up:A>s6eUwtsfXtt478/s6nUFSeFaJ7+udFV...</up:A>
      <up:R0>fyQLY758BjqVRugvph4Wa10zTkFLb...</up:R0>
    </up:SubsetProof>
  </up:Presentation>
 </up:Presentations>
</up:PresentationToken>
```

Figure 7: Presentation token example with client-specified message

The following example lists a presentation token for Contoso.com containing a presentation of one Device-protected claim UPT (disclosing one attribute), using a message specified by the Relying Party.

```
<up:PresentationToken Version="1.0" xmlns:up="http://schemas.xmlsoap.org/ws/2011/02/uprove">
 <up:Message>
  <up:Nonce>dGhlc2VhcmVyYW5kb21ieXRlcw==</up:Nonce>
  <up:SignedData>I accept the terms and conditions of Contoso</up:SignedData>
 </up:Message>
 <up:Presentations>
  <up:Presentation TokenClass="claim">
    <up:Token Version="1.0">
      <up:UIDP>http://fabrikam.com/ip1</up:UIDP>
      <up:H>Vuv9TIPe+rcAPxcpX20FjwZut6aQ...</up:H>
      <up:TI><up:Gd GdName="urn:oid:1.3.6.1.4.1.311.75.1.1.1"/>
              <up:NotOnOrAfter>20100401T00:00:00Z</up:NotOnOrAfter></up:TI>
      <up:SigmaZPrime>Z/cSeGXhvcjTIAQw36fq...</up:SigmaZPrime>
      <up:SigmaCPrime>Pjwa3US2c1U99TOMRE...</up:SigmaCPrime>
      <up:SigmaRPrime>0WJHTMjiwAq8Ganeyeu...</up:SigmaRPrime>
    </up:Token>
    <up:SubsetProof>
      <up:Attributes>
        <up:Attribute Index="3">1</up:Attribute>
      </up:Attributes>
      <up:A>MszDfj0WmUghDdEv8rIbPQBwgN2s...</up:A>
      <up:R0>KWVDRNEyJG19p6AfuDkwMUhiwQ...</up:R0>
      <up:Responses>
        <up:R>fRvB3Fni09vkgnHGd8m6Ofxk5RiKSk...</up:R>
        <up:R>Tm8XfvpZq5+9iVkdtRntk9vKGniwz7...</up:R>
```

```
      </up:Responses>
      <up:Rd>J4KScM/4BGFUeL6TNLAtPJgbUfmw8...</up:Rd>
    </up:SubsetProof>
  </up:Presentation>
 </up:Presentations>
</up:PresentationToken>
```

Figure 8: Presentation token example of a Device-protected token

References

[IMI1.0] OASIS Standard. *Identity Metasystem Interoperability Version 1.0*, July 1st 2009. http://docs.oasis-open.org/imi/identity/v1.0/identity.html.

[RFC2119] Scott Bradner. *RFC 2119: Key words for use in RFCs to Indicate Requirement Levels*, March 1997. http://www.rfc-editor.org/rfc/rfc2119.txt.

[RFC3066] H. Alvestrand. *RFC 3066: Tags for the Identification of Languages*, January 2001. http://www.rfc-editor.org/rfc/rfc3066.txt.

[RFC4051] D. Eastlake 3rd. *RFC 4051: Additional XML Security Uniform Resource Identifiers (URIs)*, April 2005. http://www.rfc-editor.org/rfc/rfc4051.txt.

[UPCS] Christian Paquin. *U-Prove Cryptographic Specification V1.1* Microsoft, February 2011. http://www.microsoft.com/u-prove.

[UPRPP] Christian Paquin. *U-Prove Recommended Parameters Profile V1.0*. Microsoft, February 2011. http://www.microsoft.com/u-prove.

[UPS] Christian Paquin. *U-Prove WS-Trust Profile Schema V1.0*. Microsoft, February 2011. http://www.microsoft.com/u-prove.

[UPTO] Christian Paquin. *U-Prove Technology Overview V1.1*. Microsoft, March 2011. http://www.microsoft.com/u-prove.

[WS-Trust14] OASIS Standard. *WS-Trust 1.4*, February 2nd 2009. http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html.

[XML-EXC-C14N] W3C Recommendation. *Exclusive XML Canonicalization Version 1.0*, 18 July 2002. http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/.

[XMLDSIG] W3C Recommendation, *XML Signature Syntax and Processing (Second Edition)*, 10 June 2008. http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/.

[XMLENC] W3C Recommendation, *XML Encryption Syntax and Processing*, 10 December 2002. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/.

[XMLSchema2] W3C Recommendation, *XML Schema Part 2: Datatypes Second Edition*, 28 October 2004. http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/.