Enhancing Privacy in Identity Federation

Anonymous Credentials Ensure Unlinkability in WS-Security

Jan Camenisch
IBM Zurich Research Lab
jca@zurich.ibm.com

Thomas Groß
IBM Zurich Research Lab
tgr@zurich.ibm.com

Dieter Sommer
IBM Zurich Research Lab
dso@zurich.ibm.com

Abstract

Federated Identity Management (FIM) allows for securely provisioning certified user identities and attributes to relying parties. It establishes higher security and data quality compared to user-asserted attributes and allows stronger user privacy protection than technologies based upon user-side attribute certificates. Therefore, industry pursues the deployment of FIM solutions as one cornerstone of the WS-Security framework. Current research proposes even more powerful methods for security and privacy protection in identity management with so called anonymous credential systems*. Being based on new, yet well-researched, signature schemes and cryptographic zero-knowledge proofs, these systems have the potential to improve the capabilities of FIM by superior privacy protection, user control, and multiple use of single credentials. Unfortunately, anonymous credential systems and their semantics being based upon zero-knowledge proofs are incompatible with the XML Signature Standard which is the basis for the WS-Security and most FIM frameworks. We put forth a general construction for integrating anonymous credential systems with the XML Signature Standard and FIM protocols. We apply this method to the WS-Security protocol framework and thus obtain a very flexible WS-Federation Active Requestor Profile with strong user control and superior privacy protection.*

1 Introduction

Today's most important on-line transactions require a user to provide her identity or certain attributes to the service provider. As a prevalent approach users enter uncertified attributes in web forms, which implies that service providers request more attributes than required for the business process itself. They use the additional information to check the provided attributes for consistency.

Federated identity management (FIM) protocols have multiple advantages over this approach: the service provider can obtain precisely the attributes required and have these attributes certified by a trusted identity provider. The FIM protocols follow the schema to send a requestor acting on behalf of a user to and identity provider where the user authenticates and obtains a credential for the attributes requested by the service provider. In the optimal case, the FIM approach enhances the user's privacy by following the data minimization principle: only user data required for the service are transmitted.

As shown by recent research, FIM protocols such as SAML [19], Liberty [17], or WS-Federation [15] excel approaches relying on uncertified attributes in terms of both security [12, 13] and privacy [21, 20]. Additionally, FIM protocols have privacy advantages compared to user-side attribute certificates as those certificates require that the full certificate be released in an interaction, even if only a subset of the attributes needs to be disclosed. Thus, FIM provides widely-deployed and standardized protocols with benefits in terms of security of attributes and privacy of the users.

Research in cryptography provides even stronger methods for identity management and attribute exchange: so called *anonymous credential systems*. The key property of this concept introduced by Chaum [9] is: the user may selectively show attributes stored in a single credential and stay completely unlinkable over multiple transactions.

This allows a user to keep an anonymous credential secret and use it multiple times to provision certified attribute data to relying parties.

Therefore, this concept allows for even stronger privacy protection while maintaining certified attributes being endorsed by a trusted identity provider. Anonymous credential systems have been pursued in different research threads by, e.g., [2] or a formal definition in [18], and reached a breakthrough in terms of features and efficiency in the Camenisch and Lysyanskaya system [4, 5, 6]. A variant of their system, the Direct Anonymous Attestation (DAA) protocol [3], has been standardized within the Trusted Computing Group (TCG).

Camenisch and Lysyanskaya also generalized the anonymous credential systems to so-called *private certificate systems* that support more differentiated attributes, the show of arbitrary logical formulas over those attributes, and the integration of verifiable encrypted attributes. When asserting attributes with this system, the identity provider need not be involved in the transaction. The user obtains her private certificates ahead of time and stores them locally. Whenever the user wants to provide certified attributes to a service provider, she uses the private certificate to generate a new proof to show a logical formula over the attributes in her private certificate. A private certificate can be leveraged multiple times in this manner with the same or different relying parties. Unless the attribute information itself inherently breaks linkability (e.g., by revealing name and date of birth), all transactions involving a private certificate are unlinkable to each other.

The cryptography of the private certificate systems allows that even if an adversary obtains the transaction transcripts of all identity providers *and* all relying parties (collusion) the transactions can still not be linked unless attribute information allows for this. In FIM protocols the user's privacy can be attacked by tracing the user's transactions over multiple service and identity providers.

The property that a credential, once obtained, can be used many times with (different) relying parties allows for cheaper (amortized) operation cost of the overall system. Furthermore, new scenarios with unreachable or off-line identity providers become possible, like mobile scenarios without connectivity to the identity provider, but only to some local relying party.

However, the building block of private certificate systems enabling these powerful features in terms of privacy protection and off-line identity provider also render the integration in FIM or WS-Security frameworks a serious challenge: the zero-knowledge proof of knowledge. In particular, the stable and well-established standard for XML Signatures, which is the basis for most FIM and WS-Security frameworks, was designed without taking the complex semantics of such a cryptographic primitive into account and only supports traditional digital signature schemes. As any approach to extend the XML Signature Standard itself is endangered by the standardization process and may jeopardize the clear semantics of the XML Signature Standard, we follow a different approach.

We have two main contributions: first, we show a general method that securely transfers the semantics of zero-knowledge proof-based protocols to XML-DSig keys and signatures. In a nutshell, our method uses a verifiable random function to bind a temporary XML signature key to a zero-knowledge proof of knowledge. We show that binding is secure in terms of that a recipient of a token can verify that the temporary key was generated faithfully.

Second, we demonstrate this method by integrating the semantic of a strong private certificate system into a particular FIM framework. We use, due to its practical importance, WS-Security [14] as a running example to be enhanced with those proof semantics. This enables a new WS-Federation Active Requestor profile [15, 16] that uses private certificates instead of traditional identities and therefore leverages all the advantages of the private certificate system.

Paper outline. We structure the remainder of this paper as follows. Section 2 explains the technologies involved in our solution. We first introduce the WS-Security area and underlying XML Signature Standards, and secondly explain the properties of private certificate systems. In Section 3, we reason why we need private certificate systems in FIM protocols and why this is a hard task to achieve. Section 4 contains the construction of our abstract method as well as its application to a WS-Security token type. We conclude the paper in Section 5.

2 Preliminaries

In this section, we provide the background regarding the XML Signature standard, WS-Security and its use for federated identity management, and private certificate systems. In particular, we motivate the properties of

traditional FIM approaches and private certificate systems.

2.1 Established Technologies and Standards

We provide the required background to established standards and technologies which the contributions in this paper are based on. These are the XML Signature Standard, the WS-Security protocol framework, WS-Trust, and WS-Federation.

2.1.1 XML Signature Standard

The XML Signature Standard (XML-DSig) [1] defines how to do a signature on parts of an XML document or, more generally, on any web resource. The standard completely defines the syntax and also the basic semantics for XML signatures.

Without going into details, a signature's message is defined by a list of references, each to a web resource or part of an XML document. Each referred message part is hashed and then the list of digests and reference information is signed by a traditional digital signature scheme like DSA [23] or RSA [22], after applying a hash function.

Typical examples for applications of XML signatures include security assertion languages and protocol frameworks like WS-Security.

As the approach of XML-DSig has not been designed to account for advanced cryptographic mechanisms like zero-knowledge proofs, it cannot work with those mechanisms in a straightforward way.

Semantics. The XML Signature standard defines only basic semantics for signatures, that is, the semantics of associating a cryptographic signature with a message. This semantics in particular includes the integrity and message authentication properties of the cryptographic signature. This boils down to the property of only the holder of the private signing key that corresponds to the public key being able to compute the signature and thus the signed message parts are protected against unauthorized changes. Signer authentication is supported by XML-DSig, but out of scope of the standard as this is trust semantics that is left to extensions by applications. This and other trust semantics are to be defined by applications of XML-DSig. XML-DSig's extensibility allows for extension of semantics by arbitrary trust semantics.

2.1.2 WS-Security

WS-Security is a protocol framework for providing message security of web services messages. In particular, messages can carry claims contained in security tokens in the message. Claims typically are statements about the requestor for authenticating her, or, more generally, to establish the required kind of identity with the relying party.

The basic mechanisms for associating a signature to the message is XML Signature. The basic semantics used in WS-Security is the semantics of XML Signature. Further semantics can be associated to the signature either in a WS-Security profile or by the applications that further process the message. The extensibility of the semantics of the signature is key for our method of transferring the semantics of the proof to the semantics of the XML-based signature.

An XML signature used by a WS-Security message takes over the basic signature semantics of XML Signature (basically, integrity) and extends it. An extension is the proof of knowledge of a confirmation key for a security token done by a signature. The security token can be a public key certificate, trust semantics in which is left out of the specification. WS-Security profiles define extensions to the basic semantics as fits the intended application area for the profile.

2.1.3 Federated Identity Management

Federated identity management (FIM) protocols are protocols between three types of players. A *user*, whose identity is to be federated, together with her *requestor* machine, an *identity provider* who certifies the identity, and a *relying party* who is the recipient of the user's identity. Today's standardized and established FIM protocols typically make use of signed assertion tokens being conveyed from the identity provider to the requestor and from the requestor to the relying party. A prominent example for such a protocol is the WS-Federation Active Requestor Profile [16]. The latter is based on security assertions contained in WS-Security messages. Whenever the identity

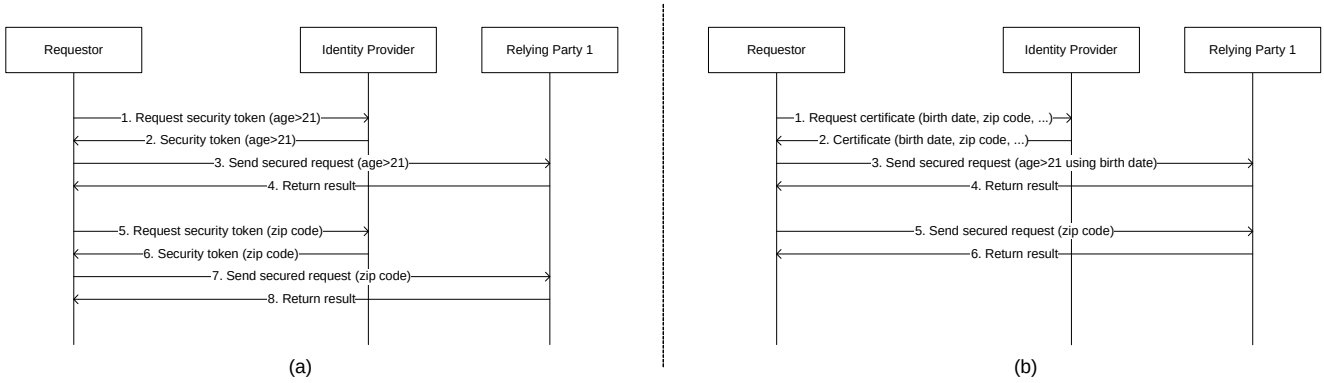

Figure 1. Message flows for traditional and idemix-based FIM protocols

of the requestor is required, a security assertion is obtained by the requestor from the identity provider and then relayed to the relying party. This allows for the properties that *i)* the attributes are certified by the identity provider and *ii)* exactly the required attributes are conveyed to the relying party. The first property provides security for the relying party, the second one privacy for the requestor. Though, the privacy only holds under the assumption that identity providers cannot be controlled by the adversary, e.g., in a passive attack in which they collaborate with relying parties to obtain profile information on the user's activities. See Figure 1 (a) for a simplified federation flow in WS-Federation Active Requestor Profile without showing policy-related messages. The example shows two identity federation transactions, in the first one the requestor's age is being established to be over 21, in the second one the zip code of the requestor is conveyed. For each of those transactions, a new security token has to be fetched from the identity provider, containing exactly the attribute information being shown.

2.2 Private Certificate Systems

A private certificate system is a generalization of anonymous credential systems in that sophisticated attribute statements are supported. A private certificate system allows for obtaining private certificates and use them to make certified claims, both issuing and using being possible in a privacy-enhancing way. The *idemix* private certificate system, whose key building block are the signature protocols of Camenisch and Lysyanskaya [5, 6], is the most advanced private certificate system that is currently available. The system allows for very general ways of making attribute claims, that is, releasing attribute information, to relying parties.

In the next few paragraphs, a general overview of private certificate systems is given. In the appendix, we give more details on the properties of private certificate systems for the interested reader.

A user obtains private certificates from *identity providers* (IPs) and holds these certificates locally. Certificates can have long lifetimes, e.g., multiple years.[1] A certificate, once obtained, is never sent to a relying party. Whenever the user needs to provide attribute information to a relying party, she uses one or multiple of her private certificates to release partial information on their third party-endorsed attributes in a controlled way. This release does neither involve the identity provider, nor a sending of the certificates. The data release protocol is based on a zero-knowledge proof of knowledge (holdership) of certificates with appropriate attributes. Figure 1 (b) shows the protocol flows for obtaining a private certificate and showing it to a verifier twice, the first time establishing that her age is greater than 21 using the birth date attribute, the next time showing her zip code attribute. Is is easy to see that the private certificate can leverage one certificate many times and each time release partial information.

As already mentioned, it is a key feature of the identity provisioning protocol, that the user can *release partial attribute information* of the involved certificates, e.g., can use the birth date attribute of one of her certificates to prove that her age is greater than 21 without releasing any other information on the birth date attribute or other attributes. The user can *combine attributes of multiple certificates* in one proof that verifies with respect to multiple issuers' certificate verification keys and state properties about them. Furthermore, *attributes can provided in encrypted form* together with a proof that the ciphertext is an attribute of a particular certificate, that is, the

[1] It is worth noting that there are certificate revocation mechanisms available; revocation is an important feature when considering the long lifetime of idemix certificates.

recipient can be sure that the requestor does not cheat in what she encrypts. This concept is denoted verifiable encryption and allows one to realize accountability of a transaction while still maintaining anonymity. A private certificate system allows to *use the same certificate in multiple transactions* and release a (different) subset of the attribute information of the certificate in each transaction while all transactions, including the issuing one, are unlinkable to each other.

Semantics. The combined semantics of an idemix assertion and proof is the following: The assertion makes claims regarding attributes of the user's certificates and about encryptions and commitments contained in the assertion. The proof provides a cryptographic proof of the assertion, that is, if the proof can be verified by the relying party, the formula in the assertion holds; that is, the verifier can be assured that the user has certificates with attributes as stated and the encryptions and commitments in the assertion have been computed as specified.

3 Discussion

Considering the properties of private certificate systems, using such systems for identity federation has the following advantages over traditional established FIM protocols:

(a) The identity provider is not involved in can be off-line during the identity provisioning transaction to the relying party. This reduces the number of messages being sent and reduces the amortized cost of a single identity transaction over many reuses of the same certificates. This holds under the (realistic) assumption that computation cycles of identity providers are the most expensive ones and are amortized over many uses of a once-obtained certificate whereas in traditional FIM protocols the identity provider is involved in each transaction.

(b) An identity provisioning transaction can involve attribute information from certificates of different identity providers and allows for logical formulas being expressed over attributes over multiple certificates, e.g., $bankstatement_1.balance + bankstatement_2.balance \geq 4000$.

(c) Attribute information can be provably encrypted without involvement of any party other than the requestor and relying party. This means that the ciphertext is accompanied by a proof that shows that the corresponding plaintext is the one claimed, e.g., the serial number of an electronic passport to allow for conditional anonymity.

(d) Even if the identity providers and relying parties are controlled by an active attacker, that is they can all arbitrarily deviate from the protocol in an orchestrated manner, they cannot link transactions unless attribute information allows for this. Given that the attribute semantics are well defined such that attribute cannot be used as covert channels between identity provider and relying parties, the unlinkability holds in this very strong attacker model. Therefore, the privacy properties hold in a stronger attacker model than the model for traditional FIM protocols.

Considering those properties of private certificate systems, their potential in identity federation is huge as they even improve on the privacy that currently deployed FIM protocols based on traditional signature mechanisms offer. Unfortunately, private certificate systems cannot be easily fit into currently deployed identity federation protocols as those protocols typically rely on XML-DSig for certifying claims which has been built around the concept of traditional signature schemes. As the idemix protocols are technically quite different—they are based on zero-knowledge proofs—they cannot be used just as another signature scheme unless the standard's explicit and intended semantics is extremely widely interpreted and at some points violated. As one example, the standard does not provide a facility for giving an assertion as input when doing a signature, but idemix requires an assertion and the private certificates being input.

4 Construction

We provide a generic construction for transferring the semantics of a zero-knowledge proof-based protocol for making certified claims, like the idemix protocol, to an XML Signature public key and a signature computed with

the corresponding secret key in Section 4.1. This construction generalizes to any conceivable signature standard with sufficiently extensible semantics. As for the importance of the WS-Security protocol framework, we show in Section 4.2 how the construction is carried over to a WS-Security protected message.

4.1 Generic Construction

This section provides our construction to integrate an authentication with a private credential system into a standard signature scheme and XML-DSig. The idea is that we transfer the authentication with the private credential into the authentication with a temporary standard signature key with which an XML-DSIG can be generated.

For this authentication transfer our security model requires that the requestor is not trusted. Therefore, we need to enforce that the requestor either chooses the temporary signature key faithfully according to the protocol or that the verification aborts. In particular, we require that the requestor always chooses a pseudo-random and fresh temporary signature key for each single certificate show.

This property of the signature key is very important as a malicious requestor may attempt to use a temporary signature key multiple times, mis-bind keys of other principals, or bind a single key to multiple credentials.

Therefore, we transform an authentication statement with a private credential system (e.g., the party has the attributes att = "'is older than 21"') to the following statement:

(a) This party has the attributes att (e.g., "'is older than 21"').

(b) This party holds the secret key K_s that corresponds to a temporary signature verification key K.

(c) The key pair (K_s, K) is pseudo-random and freshly generated for this authentication statement.

For each statement made with a zero-knowledge proof-based protocol, we guarantee that a new temporary key pair is freshly and pseudo-randomly generated. We enforce that the requestor chooses this key pair faithfully by means of a *verifiable pseudo-random function* and a corresponding zero-knowledge proof stating that the temporary signature verification key was computed correctly. This model is natural for our setting as the one-timeness of keys is required for the unlinkability of transactions and there is no need for the requestor to choose a key itself.

In terms of attribute claims our construction shows the following relationships: it binds claims that are proved by a zero-knowledge proof-based mechanism to a *temporary signature key pair* created for a single transaction. This temporary key pair can be used to create XML signatures with semantics extended by the proved claims. The claims are manifest in three new token types: an *assertion token* A, a *proof token* p, and a *temporary public key token*. The temporary public key token is associated with claims regarding the key pair's properties and faithful generation. The assertion token provides additional claims about a user's attributes that are associated with the key pair. Finally, the proof token holds a two-fold proof for those claims: on the one hand, the properties and faithful generation of the keys are proved; on the other hand, the additional claims of the assertion about the user's attributes are proved.

We will elaborate our method on the example of the idemix certificate proof protocol and idemix assertion, but the same method applies to any zero-knowledge proof-based protocols that can be represented as a set of security tokens, thus our construction is extremely general.

Setup. Let G be a prime order q cyclic group and g a generator of it, both g and the group G are system parameters. G and g are chosen such that they can be used as public parameters of a DSA signature scheme used by all system participants. We assume that the signer has all private certificates required for the protocol to be executed. We also assume that the parties have the certificate verification keys that are required as inputs to the protocol.

Subsequently, we describe our construction which is a protocol between a requestor (the signer) and a relying party (the verifier). The protocol proceeds by a *Generation* done by the requestor, sending the signed message to a relying party, and a *Verification* done by a relying party. The generation and verification are explained in detail below.

4.1.1 Signature Generation

Create an idemix assertion A and compute the idemix proof p_A for the correctness of the assertion following the idemix protocol CertificateProof for releasing data using private certificates.[2] Compute the temporary signature public key K as a *verifiable random function* with the idemix proof p_A and the security context as input as follows: Choose an $x \in_R \{1, ..., q\}$ and compute m as $m = H(p_A||context)$. The variable *context* can contain other security-relevant context information such as time or a nonce; this depends on the application the construction is used for. Compute the value y as $y := g^x$ and the temporary public key K using the random function as $K := g^{\frac{1}{x+m} \pmod q}$. The temporary secret key K_s is $K_s := \frac{1}{x+m} \pmod q$. The verifiability is accounted for by computing the non-interactive proof p_K that is specified as follows, using the widely-used zero-knowledge proof specification language introduced by Camenisch and Stadler in [8]:

$$SPK\{(\alpha, \beta) : y = g^\alpha \wedge K = g^\beta \wedge g = y^\beta g^{m\beta}\} \tag{1}$$

This non-interactive proof shows that y and K have been computed according to the protocol. The variable α maps to the signer's secret x and β to the signer's secret $\frac{1}{x+m} \pmod q$.

As next steps, the XML tokens for the assertion, the proofs, and the temporary public key have to be built. We do not give the complete syntax for these tokens as it is analogous to the syntax we use in the method for WS-Security as shown further below in Section 4.2, but independent of WS-Security. The semantics of the tokens and references is outlined further below.

Construct the assertion token such that it encapsulates the assertion A. Construct the proof token such that it contains the proofs p_A and p_K. The proof token references the assertion token. The public key token contains the key K and a reference to the proof token.

Compute an XML signature with the private signing key K_s over the message to be signed and the security tokens defined above. The result is a signed document with the semantics of an idemix certificate proof being transferred to the verification key and thus also the XML signature.

4.1.2 Signature Verification

The verification of a message with signature received from a requestor is performed by the relying party as follows: extract the reference to the temporary public key token from the XML signature. Obtain the reference to the proof token from the public key token. Verify the correct generation of the temporary public key by verifying the zero-knowledge proof p_K. For this, compute m as $m = H(p_A||context)$ where the context can be derived in a well-determined way. Verify the proof p_K using p_K, m, and K as input. If the proof verifies, follow the reference to the assertion A and verify the proof p_A with respect to A and the public keys of the certificate issuers. If successful, the public key is valid and has the semantics outlined below. Then the XML signature is verified with the temporary public key. If the signature is valid, any further processing depends on the application and its trust semantics.

Discussion. The temporary key pair (K_s, K) being computed as verifiable random function of p_A and the security context is a DSA key pair. The discrete logarithm nature of the DSA signature key pair allows for the application of the verifiable random function for its generation. As DSA is a required signature algorithm of the XML Signature Standard every implementation of XML-DSig can be used for our extensions.

The exponent x in is chosen freshly in each new execution of the protocol. The verifiable random function assures that K is pseudo random and derived from the proof p_A without the requestor being able to choose the key pair herself. The key is fresh as p_A is guaranteed to be different whenever a new proof is executed, unless the prover deviates from the protocol and reuses randomness in the idemix proof which would make her linkable immediately. In addition, many applications allow the security context to be different in each execution of the protocol, e.g., by including the current time or a challenge from the relying party. The private key K_s remains, of course, secret to the prover and computing it from the values known to the relying party is computationally intractable. We refer the reader to [10] for a security model and security proof for the verifiable random function we use in our construction.

[2] The proof p_A is the non-interactive version of the certificates show protocol including corresponding zero-knowledge proofs.

Semantics. The temporary public key token has the semantics that the contained public key and corresponding private key is freshly generated from p_A and a $securitycontext$ without the possibility of the requestor choosing either part of the key pair. Additionally, further claims proved by a proof token referenced from the key token and specified by the assertion token are associated with the temporary key. The semantics of the reference in the public key token does not restrict in any way on whether one or multiple tokens are used to contain the proof and further claims. The public key token references the proof with the semantics of the proof being a proof for the claims about the key properties and additional claims carried by the assertion and being proved by the proof being associated to the temporary key.

This immediately gives rise to the new comprehensive semantics of the temporary signing key by the combined semantics of the key properties and the identity claims regarding the key holder associated to the key as stated in the assertion. This semantics transfers to the signatures that are made with the temporary key. A signature represents a proof of possession of the private key corresponding to the public key and thus securely binds the claims associated to the public key to the signed content.

For the application semantics this boils down to the party signing with the secret key associated to the public key provably has the properties as stated in the idemix assertion.

4.2 WS-Security

Using the above generic construction for XML-DSig as a template, we transfer the idemix semantics to XML signatures for web services messages of the WS-Security protocol framework. This allows us to bind claims authenticated by any zero-knowledge proof-based protocol to web services messages. When the idemix assertion token and proof token are placed as security tokens inside a WS-Security header, this alone could already convey the basic semantics of the idemix proof to the relying party. Though, there would be no secure binding of the semantics to the web service message, thus the conveyed semantics would not be of much use in a WS-Security context.

Thus it is required to have a secure binding of the idemix semantics to the WS-Security header. This binding can be achieved by leveraging our basic construction of Section 4.1 and transferring the semantics to the WS-Security-protected message with an XML signature. The assertion, proof, and public key tokens are again used in this construction, this time with WS-Security-specific syntax and semantics. The semantics remains basically the same, it only needs to be mapped to WS-Security terminology.

The temporary private key is used for making an enveloped signature over the whole web services message. This step performs the secure binding of the proof to the web services message.

4.2.1 Message Generation

We again require an *assertion* token, a *proof* token, and a *temporary public key* token for our method (see Figure 2 and Appendix A). F or WS-Security integration, they have to be designed specifically as WS-Security security tokens. The temporary private key is used for making an enveloping XML signature over the complete web services message. The WS-Security specification defines the `SecurityTokenReference` mechanism for referencing security tokens which extends the semantics of the XML signature by t he semantics of the security token. We use this reference mechanism in the XML signature in order to use our custom temporary public key token for the signature.

The temporary public key token contains a key claim regarding the key and its properties, but no further associated claims. The token provides for a mechanism for referencing another security token (that can again reference others or be referenced from others) that contains a proof of the claimed key properties and that can provide further claims and proofs thereof. These further claims are associated with the temporary key pair.

The referenced security token (the proof token) contains the proof of the key properties and a proof of additional claims made by the assertion referenced from the proof.

Semantics. The semantics of the basic construction for the proof with assertion and the temporary key remains unchanged and defines the key claim and associated claims. That is, the temporary signing key is endorsed by the proof, that is, third party-endorsed claims are securely bound to the key. The signature with the temporary key is

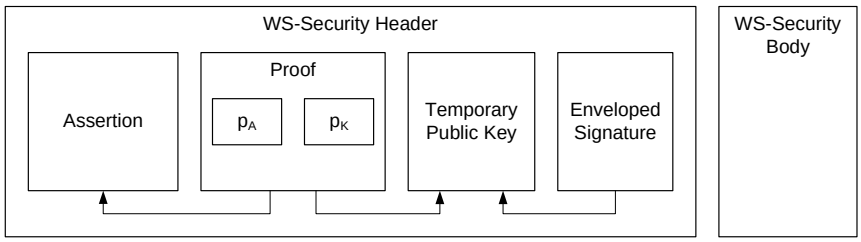

Figure 2. Structure of a WS-Security token with a zero-knowledge proof

a proof-of-possession of the temporary secret key. The signature with the temporary signing key is also a claim confirmation with the semantics defined in WS-Trust as it can only be performed by the holder of the secret key. This claim confirmation immediately leads to a cryptographic binding of the claims to the web service message.

The temporary key token contains the key material for the signature verification key. The key semantics is that the key is temporary and freshly and randomly generated. The token is not endorsing this claimed key semantics. The endorsement and further claims are contained in a token referenced within the `AssociatedEndorsedClaims` element. The semantics is that the referenced token provides a proof for the claimed key properties. The token may also provide further third-party endorsed claims that are bound to the temporary public key. In our case, the referenced token is the idemix proof token.

The idemix proof token contains a reference to an idemix assertion token. The proof provides a proof of the statements of the assertion. The proof token also carries the proof for the properties of the key (freshly randomly generated).

The assertion makes claims about the message producer, but relies on the proof token to prove the claims such that the proof can be verified against the public keys of the certificate issuers for the certificates being used in the proof.

5 Conclusion

We presented a generic construction for transferring the semantics of zero-knowledge proof-based protocols like private certificate systems to XML-DSig keys and signatures.

We demonstrated the use of our method by seamlessly integrating a private certificate system into WS-Security and therefore enabling a new WS-Federation Active Requestor profile with superior privacy properties.

To our knowledge, our approach is the only feasible one within the following requirements:

(a) not to change the XML-DSig Standard as it is already stable and changes toward zero-knowledge proofs may complicate the XML-DSig semantics beyond manageability.

(b) not to resort to an extremely wide interpretation of standard parts beyond its original intensions and even to violations of its intended semantics.

To our judgment both requirements are mandatory to guarantee a wide acceptance and secure deployment of XML-DSig-based protocols employing private certificate systems.

The extensions of XML-DSig and WS-Security use well-defined extensibility points are completely orthogonal to the standards. That is, all existing XML-DSig and WS-Security deployments may stay untouched. Our extensions compose seamlessly with existing standards semantics and, therefore, promote a easy application in an industrial WS-Security environment.

References

[1] D. E. 3rd, J. Reagle, and D. Solo. XML-Signature syntax and processing, Mar. 2002. `http://www.w3.org/TR/xmldsig-core/`.

[2] S. Brands. *Rethinking Public Key Infrastructure and Digital Certificates— Building in Privacy*. PhD thesis, Eindhoven Institute of Technology, Eindhoven, The Netherlands, 1999.

[3] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 132–145, New York, NY, USA, 2004. ACM Press.
[4] J. Camenisch and A. Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer Verlag, 2001.
[5] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In G. Persiano, editor, *Third Conference on Security in Communication Networks*, volume 2576 of *LNCS*, pages 274–295. Springer Verlag, 2002.
[6] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology — CRYPTO 2004*, LNCS. Springer Verlag, 2004.
[7] J. Camenisch, D. Sommer, and R. Zimmermann. A general certification framework with applications to privacy-enhancing certificate infrastructures. Technical Report 3629, IBM Research, November 2005.
[8] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In B. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1296 of *LNCS*, pages 410–424. Springer Verlag, 1997.
[9] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, Oct. 1985.
[10] Y. Dodis and A. Yampolsky. A Verifiable Random Function with Short Proofs an Keys. In *Public Key Cryptography*, volume 3386 of LNCS, pages 416–431, 2005.
[11] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86*, volume 263 of *LNCS*, pages 186–194. Springer Verlag, 1987.
[12] T. Groß and B. Pfitzmann. Proving a WS-Federation Passive Requestor profile. In *2004 ACM Workshop on Secure Web Services (SWS)*, Washington, DC, USA, Oct. 2004. ACM Press.
[13] T. Groß, B. Pfitzmann, and A.-R. Sadeghi. Proving a WS-Federation Passive Requestor profile with a browser model. In *2004 ACM Workshop on Secure Web Services (SWS)*, Alexandria, VA, USA, Nov. 2005. ACM Press.
[14] P. Hallam-Baker, C. Kaler, R. Monzillo, and A. Nadalin. Web services security: SOAP message security, 2003.
[15] C. Kaler and A. N. (ed.). Web Services Federation Language (WS-Federation), Version 1.0, July 2003. BEA and IBM and Microsoft and RSA Security and VeriSign, `http://www-106.ibm.com/developerworks/webservices/library/ws-fed/`.
[16] C. Kaler and A. N. (ed.). WS-Federation: Active Requestor Profile, Version 1.0, Jul 2003. BEA and IBM and Microsoft and RSA Security and VeriSign, `http://www-128.ibm.com/developerworks/library/specification/ws-fedact/`.
[17] Liberty Alliance Project. Liberty Phase 2 final specifications, Nov. 2003. `http://www.projectliberty.org/`.
[18] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H. Heys and C. Adams, editors, *Selected Areas in Cryptography*, volume 1758 of *LNCS*. Springer Verlag, 1999.
[19] OASIS Standard. Security assertion markup language (SAML) V2.0, Mar. 2005.
[20] B. Pfitzmann. Privacy in enterprise identity federation - policies for Liberty 2 single signon. *Elsevier Information Security Technical Report (ISTR)*, 9(1):45–58, 2004. `http://www.sciencedirect.com/science/journal/13634127`.
[21] B. Pfitzmann and M. Waidner. Privacy in browser-based attribute exchange. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 52–62, Washington, USA, Nov. 2002.
[22] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb. 1978.
[23] US Department of Commerce and National Institute of Standards and Technology. Digital signature standard (dss). Federal Information Processing Standards Publication, 2000.

A Details of the Token Format

We give an example of the skeleton of the WS-Security header of a web services message. Note that in the example, the namespace `wsse` is the one of WS-Security, and `cred` the namespace for our extensions.

```
...
<wsse:Security>
  <cred:ProofToken wsu:Id="endorsed_claims"
      xmlns:cred="www.ibm.com/IdemixProofToken"
    <AssertionFormat format="www.ibm.com/IdemixAssertionFormat"
    <cred:Proof>
      ...
    </cred:Proof>
    <!-- Reference to the assertion -->
```

```
    <AssertionReference URI="#assertion_token"/>
  </cred:ProofToken>

  <cred:Assertion wsu:Id="assertion_token">
    ...
  </cred:Assertion>

  <cred:TemporaryPublicKey wsu:Id="temp_public_key">
    <cred:AssociatedEndorsedClaims>
      <wsse:Reference URI="#endorsed_claims"/>
    </cred:AssociatedEndorsedClaims>
    <cred:KeyMaterial>
      ...
    </cred:KeyMaterial>
  </cred:TemporaryPublicKey>

  <ds:Signature>
    <ds:SignedInfo>
      ...
      <!-- signature over message -->
    </ds:SignedInfo>
    <ds:SignatureValue>
      ...
    </ds:SignatureValue>
    <ds:KeyInfo>
      <!-- Reference to key -->
      <wsse:SecurityTokenReference>
        <wsse:Reference URI="#temp_public_key"/>
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </ds:Signature>
</wsse:Security>
...
```

B Private Certificate Systems

B.0.2 Certificate Proof

A certificate proof protocol is an interactive protocol CERTIFICATEPROOF between a *user* and a *relying party*. The protocol conveys an assertion from the user to the relying party that specifies claims made by the user to the relying party. The interactive steps of the protocol provide a proof of the assertion to the relying party by the means of cryptographic zero-knowledge proofs.

A non-interactive variant of the protocol can be obtained by applying the Fiat-Shamir heuristic [11] to the interactive zero-knowledge proofs of the protocol. The non-interactive protocol fits into a single message which is the reason why we will only consider the non-interactive variant for our constructions. It requires that the user create two tokens, the *assertion* token and the *proof* token, that are both sent to the relying party. The two tokens together convey claims (in the assertion) and a proof of these claims (in the proof) to the relying party in a single message exchange. The relying party checks the correctness of the proof by executing the verification algorithm on the proof, the assertion, and the appropriate public keys of certificate issuers as governed by the assertion.

Assertion. The assertion specifies the claims that are made by the user to the relying party. These claims are claims about attributes of certificates the user has and about encryptions and commitments contained in the assertion and their relation to attributes of certificates. The assertion consists of three sections, a *logical formula* specifying the claims, *certificate information* providing information on the types of the involved certificates and the public key of their issuers, and *encryptions and commitments*.

The *logical formula* is a formula in predicate logic without without negation and without quantifiers. It connects *predicates* with the logical AND and OR operators. Predicates are expressed over *variables* that can be instantiated

with different types of objects. A variable can either refer to an *attribute* of a certificate the user has (knows), or to an *encryption* or *commitment* contained in the third section of the assertion. A predicate makes a statement on variables – predicates that express polynomial relations on the variables are supported. For integer attributes, the comparison operators $>$, $\geq$, $<$, $\leq$, $=$, and $\neq$ are available, for string attributes only $=$, and $\neq$.

When a variable refers to an *attribute*, we mean the value of the specified attribute of a certificate the user possesses. The certificate is specified by its type, its issuer, and verification key in the second section of the assertion. The semantics is that the user has a certificate that contains an attribute with a value that fulfills the properties as stated by the predicate that uses the variable. For example, a predicate can state that $bankstatement.balance \geq 4000$ where $bankstatement.balance$ is the variable referring to the value of the $balance$ attribute of the user's bank statement.[3]

A variable referencing an *encryption* refers to the plaintext value of a particular ciphertext that is contained in the third section of the assertion together with information regarding the key it was encrypted under. The semantics is that the ciphertext is an encryption of the plaintext under a specified third party's key and the statements for the plaintext hold as expressed by the predicate. For example, $enc_1 = bankstatement.balance$ specifies that variable enc_1 appearing in the third section of the assertion is an encryption of the value of the balance attribute of the user's bank statement certificate. The variable $bankstatement.balance$ again refers to the balance attribute of the bank statement certificate of the user.

Each ciphertext has a decryption condition cryptographically bound to it. This condition is agreed upon by both user and relying party and defines under what conditions a decryption of the ciphertext may be requested from the third party by the relying party. The third party must verify the decryption condition and only decrypt in case the decryption condition is fulfilled. The semantics and verification of the decryption condition is out of the scope of the cryptographic system.

A variable referring to a *commitment* refers to the committed value, not the cryptographic commitment itself, analogous to ciphertexts. The semantics is that the commitment is a commitment of a value as specified by the predicate. The cryptographic commitment is contained in the third section in the assertion.

Example assertion:

Formula	$passport.bdate \leq current\_date - 21years \wedge passport.holder = idcard.holder \wedge$ $idcard.zipcode = 7890 \wedge enc_1 = passport.snr$
Key information	$(passport, PK_{USPPAuth})$, $(idcard, PK_{USIDAuth})$
Encryptions and commitments	(enc_1, cryptographic value; PKE_T; cond = "When the user does not comply to the general terms of the service, the ciphertext may be decrypted for legal actions.")

In this example, the birth date attribute of the electronic passport of the user is used to establish that the user's age is greater than or equal to 21 years. The zip code attribute of the idcard certificate is released and a prove is given that the holder of the passport and the idcard is the same person. Within the encryption enc_1 the serial number of the user's passport is encrypted. The second section of the assertion contains the key information for the two certificates used in the formula. The third section contains the ciphertext referenced by enc_1, the corresponding public key information, and the decryption condition.

A more advanced feature of what can be expressed in an assertion is a logical *OR* relation between predicates, e.g., that the user establishes her age with either a US passport or a Swiss one without revealing the information on which passport she actually holds. See [7] for more details on these features and their semantics.

Proof. The proof is a cryptographic proof that the assertion is certified, that is, that the requestor has private certificates fulfilling the attribute statements in the assertion and that the ciphertexts and commitments in the assertion have been computed correctly. The proof either is an interactive protocol or, when applying the Fiat-Shamir heuristic, it can be expressed in a single token.

Generally, zero-knowledge proofs allow for proving statements without revealing any further information. The idemix protocols build on zero-knowledge proofs for proving that one knows (has) a signature on a set of attributes (that is, a certificate) without revealing the signature and without revealing the attributes. In addition to this, known zero-knowledge proof techniques can be used to prove properties of attributes, e.g., that the balance attribute is

[3] A real-world use case would typically require that the user also prove that the bank statement is a recent one. We keep this out of our example in order to keep it simple.

greater than 4000, without revealing the attribute. All these protocols are computationally very efficient and thus the idemix system is highly practical.

B.0.3 Certificate issuing

A private certificate is obtained by the interactive protocol ISSUECERTIFICATE between the *receiver* of the certificate and the *issuer* (identity provider). As a prerequisite we assume that the receiver has provided cryptographic commitments of attribute information of certificates in a prior protocol with the identity provider (the roles in this protocol were user and relying party). This approach of doing a PROOFCERTIFICATE protocol first fits, for example, into the WS-Trust security token exchange paradigm. What attributes have to be committed to is defined via the policy of the issuer. It is worth noting that these commitments being made do not reveal any information (information theoretically) regarding the committed value to the issuer, but allow that the issuer can include the committed values into the certificate to be issued.

Each attribute of the certificate to be issued is either *known to the issuer*, *jointly randomly generated*, *committed by the receiver*, or *unknown to the receiver*. The case of the attribute being known to the issuer is the standard case of determining the attribute, this is the way as signed security tokens are being issued today, e.g., X.509 certificates. Jointly randomly chosen attributes are useful for realizing e-coins by using the random number as serial number of the e-coin, but preventing either of the parties determining it and the relying party from learning it. Committed attributes are key for realizing a general private certificate system in which certain prerequisite certificates are shown, and based on this new certificates are issued, possibly including attributes of the shown certificates, but without the issuer learning those. This feature is important for binding private certificates to the user, e.g., via including a user's master secret and pseudonym as an attribute in each certificate without the issuer learning this attribute. The case of an attribute unknown to the issuer is a niche case with little importance.

Compared to traditionally used certification schemes based on signature mechanisms like RSA [22], idemix allows for much stronger privacy properties in the certification process as attributes from other certificates can be taken into the certificate without letting the issuer learn them; the issuer only learns that the receiver has certificates from trusted issuers on them. We will not consider the issuing in more detail in this paper as our focus are the proof protocols and their application within XML-Dsig and WS-Security. It shall be mentioned that the issuing can be realized within the WS-Trust framework by using its multi-round token exchange features.