Enhancing Privacy of Federated Identity Management Protocols

Anonymous Credentials in WS-Security

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

Thomas Gross
IBM Zurich Research Laboratory
tgr@zurich.ibm.com

Dieter Sommer
IBM Zurich Research Laboratory
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.

Categories and Subject Descriptors: H.4.3 [Information Systems Applications]: Communications Applications

General Terms: Security, Standardization.

Keywords: Security, privacy, federated identity management, XML Signature Standard, WS-Security, verifiable random function.

1. INTRODUCTION

Today's 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 in order to perform a sanity check on the important attributes.

Federated identity management (FIM) protocols have multiple advantages over this approach: the service provider can obtain precisely the attributes required by the business process and have a trusted identity provider vouch for the attributes. Many prevalent FIM protocols involve the identity provider in each transaction and have him issue a signed security token that is relayed to the service provider via the user's machine. FIM protocols can enhance the user's privacy by following the data minimization principle: only user data required for the service are requested and provided. No additional, typically identifying, attributes are required for doing a sanity check as the attributes are certified. Also, FIM protocols allow for unlinkable transactions. The privacy advantages of FIM hold in the weak trust model of a fully trusted identity provider.

Research in cryptography lead to even stronger methods for attribute exchange, so called *anonymous credential systems*. Their key property is that the *show transaction* is unlinkable to the *issuance transaction*. Newer systems even allow for that a credential, once obtained, can be used polynomially many times and all the transactions are still unlinkable. Furthermore, users can release a subset of attributes certified in a credential. We will refer to current-day generalizations of anonymous credential systems as *private certificate systems* as this characterizes them well due to their basic principle of operation: A *private certificate* is obtained once and then used many times to perform mutually unlinkable transactions. Overall, anonymous credential systems or private certificate systems allow for even stronger privacy protection while maintaining certified attributes being endorsed by a trusted identity provider.

The latest private certificate systems are based on (non-interactive) zero-knowledge proofs of knowledge. This is a very different mechanism compared to traditional signature schemes used in FIM protocols. Thus it is not straightforward to deploy private certificate systems within FIM protocols. The key difficulties in transferring the zero-knowledge proof semantics to the XML-DSig and Web Services message are: (1) the zero-knowledge proof must be securely bound to the WS message, such that only the person computing the proof can have done the binding; (2) multiple such bindings must be unlinkable to each other; (3) the proving party should not be able to reuse an existing signature key or use a signature key with multiple different zero-knowledge proofs.

1.1 Contributions

We have two main contributions: first, we present a general method to securely transfer the semantics of zero-knowledge-proof-based protocols to XML-DSig keys and signatures. Second, we sketch how this method can be applied to integrate private certificate systems—due to its practical importance—into WS-Security and thus as well into WS-Trust. This enables a new WS-Federation Active Requestor profile [13, 12] that uses private certificates instead of traditional tokens and therefore leverages all the advantages of the private certificate system. More background is given in the full version of the paper [4].

1.2 Paper Outline

We structure the remainder of this paper as follows: Section 2 sketches the preliminaries for our approach: XML-DSig, WS-Security, federated identity management, and private certificate systems. Section 3 contains the construction of our generic method applicable to XML-DSig. Section 4 sketches how the generic construction is applied to WS-Security and thus to WS-Trust and WS-Federation. We conclude the paper in Section 5.

2. PRELIMINARIES

In this section, we provide the background regarding the underlying technologies used in our approach.

2.1 Established Technologies and Standards

2.1.1 *XML Signature Standard*

The XML Signature Standard (XML-DSig) [10] 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 to message part is hashed and then the list of digests and reference information is signed by a traditional digital signature scheme like DSA or RSA, after applying a hash function.

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 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. These trust semantics are left to be extended by applications. Trust semantics are arbitrarily extensible in XML-DSig.

2.1.2 *WS-Security*

WS-Security [14] is a protocol framework for providing message security of Web Services messages. In particular, messages can carry claims expressed by security tokens contained 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 mechanism for associating a digital signature with a Web Services message is XML-DSig. The basic semantics used in WS-Security is the semantics of XML-DSig (basically, integrity). 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 an XML-DSig is used for our method of transferring the semantics of the proof to the semantics of the XML signature.

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 vouches, e.g., through issuing certified attribute statements, for 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* [13]. The latter is based on security assertions contained in WS-Security messages. Whenever the identity 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. However, 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.

2.2 Private Certificate Systems

A *private certificate system* is a generalization of *anonymous credential systems* in that more sophisticated statements are supported. A private certificate system allows for obtaining private certificates and for using them to make certified claims, both issuing and using are possible in a privacy-enhancing way. Examples for private certificate systems are Brands' system [3] and Camenisch and Lysyanskaya's system [6]. Generalizations of the latter one are also known as *idemix*. Contrary to Brands' system, the idemix system features multi-show unlinkability.

In a private certificate system, a user obtains private certificates and can then use them to release attribute information in a polynomial number of transactions and still all the transactions are unlinkable, unless attribute information allows for establishing linkability. When using a private certificate for asserting attribute information to a relying party, an *assertion* conveying identity information and a *proof* of the correctness of the assertion is created using the certificates the assertion makes statements about. Thus, the verifier can be assured that the prover has certificates with the stated attributes. The certificates themselves are never revealed, thus unlinkability can be reached. The identity provider can be *off-line* during the identity provisioning transaction. The system allows for *general attribute claims*, that is, releasing attribute information, to relying parties. Most noteworthy are the selective release of attributes, the partial release of attributes, or showing polynomial relations between attributes of multiple certificates, and verifiable encryption of attributes (conditional release).[2, 7] The last property allows for anonymous yet accountable transactions [1].

Private certificate systems cryptographically guarantee privacy in a strong trust model, that is, no honest identity provider is required: even if the 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 in this strong model, though FIM protocols protect privacy in the weak model of an honest identity provider.

Private certificate systems can not be easily fit into currently deployed FIM protocols such as WS-Security as those protocols typically rely on the stable and well-established XML-DSig standard for certifying claims. This standard has been built around the concept of traditional signature schemes such as RSA or DSA. As the idemix protocols are technically different—they are based on zero-knowledge proofs—they cannot be used as another signature scheme unless the standard's explicit and intended semantics is extremely widely interpreted and at some points violated.

3. GENERIC CONSTRUCTION

This section provides a generic construction for transferring the semantics of a non-interactive zero-knowledge-proof-based protocol for making certified claims, for example a private-certificate-based proof protocol, to an XML-DSig public key and thus to a signature computed with the corresponding private key. This construction generalizes to any conceivable signature standard with comparably extensible semantics.

3.1 Security Properties

Our security model does not require the signer to be trusted. Therefore, we enforce through the protocol that the signer either chooses the temporary signature key faithfully according to the protocol or that the verification of the signature fails. In particular, we require that the signer chooses a pseudo-random and fresh temporary signature key pair for each single signature to which the semantics of a zero-knowledge-proof-based protocol for proving correctness of an assertion is to be transferred.

These properties of the signature key are very important as a malicious requestor may otherwise attempt to use a temporary signature key multiple times, mis-bind keys of other principals, or bind a single key to multiple zero-knowledge-proof-based authentications and thus exploit potential vulnerabilities. The following list summarizes the properties of our construction:

(a) The party creating the signature has performed the zero-knowledge proof for proving the correctness of the stated assertion. For the example of a private certificate system, this means that the signer has private certificates that fulfill the assertion.

(b) The party holds the private DSA key K_s that corresponds to the public DSA key K.

(c) The DSA key pair (K_s, K) is pseudo-random and freshly generated for the zero-knowledge proof and security context associated with the signature.

(d) Multiple messages with the same assertion are unlinkable if other parts of the message and the assertion do not establish linkability.

Note that items (a) to (c) account for the security of the relying party, whereas (d) accounts for the privacy of the requestor.

3.2 Construction

Our construction binds claims from the assertion A that are proved by a non-interactive zero-knowledge-proof-based mechanism to a *temporary signature public key*. The corresponding private key can be used to create XML signatures with semantics extended by the proved claims.

For each statement made with the zero-knowledge-proof-based mechanism, 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 generating it and a proof of its correct generation by means of a *verifiable pseudo-random function*. 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.

The construction requires three new token types: an *assertion token* $\tilde{A}$, a *proof token* $\tilde{p}$, and a *temporary public key token* $\tilde{K}$. The temporary public key token is associated with claims regarding the key pair's properties and faithful generation. The assertion token provides additional claims—in case of using a private certificate system, 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 made in the assertion are proved.

We assume for the algorithms that the signer has all private and public parameters required to perform the non-interactive zero-knowledge proof for the correctness of the assertion. See [7] for what is required for a private certificate system. We assume that the verifier has all the required public parameters.

As our construction utilizes a *verifiable random function* (VRF) to generate a temporary DSA key pair, we briefly give the intuition behind this cryptographic mechanism. Intuitively, a VRF is much like a conventional pseudo-random function, with the difference that it additionally generates a proof that the pseudo-random value has been properly generated. A function family $F_{(\cdot)}$ is a family of VRFs if there exist algorithms VRFGen, VRFProve, and VRFVerify such that $\mathrm{VRFG\textsc{en}}_x(1^k)$ outputs a pair of keys (y, x). $\mathrm{VRFP\textsc{rove}}_x(m)$ computes $(F_x(m), p_x(m))$ where $F_x(m)$ is a pseudo-random value and $p_x(m)$ is the proof of correct generation of this value. $\mathrm{VRFV\textsc{erify}}_x(m, K, p)$ verifies a proof with respect to its corresponding value. We use a VRF based on the one of Dodis and Yampolskiy [9], but use a zero-knowledge proof as a correctness proof. The function has been proved to be *pseudo random* in the generic group model [5].

Subsequently, we describe the setup algorithm and our construction for generation and verification of an XML signature. The *signature generation* operation is performed by the signer, then the signed message is provided to the verifier who performs the *signature verification* operation.

3.2.1 System Setup

The following algorithm generates the public parameters of the system. The created values are distributed to each system participant. Note that (p, q, g) defines a VRF family $F_{(\cdot)}$.

```
Algorithm Setup
Input: Bit lengths for the public parameters. Those are to be chosen such that the generated parameters $(p, q, g)$ can be used as public parameters of a DSA signature scheme.
Output: Public parameters $(p, q, g)$.
1: Generate a prime order $q$ cyclic group $G$: Generate a prime $p$ and a prime $q$ such that $q|(p-1)$. Generate a generator $g$ of $G$.
2: return $(p, q, g)$
```

3.2.2 Signature Generation

The following algorithm is used to create a signature with the semantics of a zero-knowledge-proof-based protocol transferred to the public key and the signature. Note that the security context ctx contains any security-relevant context information such as time or a challenge. This depends on the application the construction is used for.

```
Algorithm Sign
Input: Items required for creating the non-interactive zero-knowledge proof of knowledge $p_A$ and the assertion $A$ (see [7] for an example); system parameters $(p, q, g)$; the context $ctx$;[1] a Web Services message $\tilde{\mu}$.
Output: Signed message $\tilde{\mu}^*$.
```

[1]To prevent replay in the setting of Section 4, a time stamp or a challenge can be contained in ctx.

```
1: Create a new verifiable random function $F_{(x,y)}$ from the function family $F_{(\cdot)}$: Choose the secret key $x$ of $F_{(x,y)}$ as $x \in_R \{1,...,q\}$. Compute the public key $y$ of $F_{(x,y)}$ as $y := g^x$.
2: Compute the non-interactive zero-knowledge proof $p_A$ for the correctness of the assertion $A$ with the public key $y$ of $F_{(x,y)}$ as input to be signed by using the Fiat-Shamir heuristic [11].
3: Let $m$ be defined as $m = H(A||p_A||ctx)$, with $H$ a cryptographically strong hash function.
4: Compute the temporary signature public key $K$ by computing the algorithm $\mathrm{VRFProve}_x(m)$ of $F_{(x,y)}$ on input $m$. Thus, $K$ is computed as $K := g^{\frac{1}{x+m} \pmod q}$. The invocation of the algorithm $\mathrm{VRFProve}_x(m)$ in addition computes the following non-interactive zero-knowledge proof of knowledge $p'_K$ to prove that $K$ has been computed correctly using $F_{(x,y)}$, specified using Camenisch and Stadler's approach [8]:
   $$SPK\{(\alpha,\beta) : y = g^\alpha \wedge K = g^\beta \wedge g = y^\beta g^{m\beta}\}(\epsilon)$$
   This non-interactive proof shows that $K$ has been computed according to the protocol. The variable $\alpha$ maps to the signer's secret $x$ and $\beta$ to the signer's secret $\frac{1}{x+m} \pmod q$. $\epsilon$ is the empty string.
5: The temporary secret key $K_s$ is $K_s := \frac{1}{x+m} \pmod q$.
6: Construct the XML token $\tilde{A}$ from the assertion $A$.
7: Construct the XML token $\tilde{p}$ from the proofs $p_A$ and $p_K := (p'_K, y)$.
8: Construct the XML token $\tilde{K}$ from the public key token containing $K$.
9: Construct the XML token $\tilde{\mu}'$ comprising $\tilde{\mu}$, the assertion token $\tilde{A}$, the proof token $\tilde{p}$, and the signature public key token $\tilde{K}$.
10: Create an enveloped XML signature $\tilde{\sigma}$ over $\tilde{\mu}'$.
11: Insert the XML signature $\tilde{\sigma}$ into the message $\tilde{\mu}'$ yielding $\tilde{\mu}^*$.
12: Return $\tilde{\mu}^*$.
```

Note that there are different possibilities to define the XML tokens for the cryptographic elements of our construction, for example, tokens can be aggregated into one token in order to better fit the overall processing at hand. Also note that if done properly, a non-enveloped signature can be used, covering all elements such that the security properties are retained.

A major benefit of our approach is that the XML-DSig library being used can remain completely unchanged.

3.2.3 *Signature Verification*

The following algorithm is executed by the verifier of the XML signature.

```
Algorithm Verify
Input: Items required for verification of the proof $p_A$ (see [7] for an example); system parameters $(p, q, g)$; the message $\tilde{\mu}^*$; the context $ctx$.
Output: true if and only if the basic signature validation of the extended-semantics signature has been successful, otherwise false.
1: Extract the signature element $\tilde{\sigma}$, the public key $\tilde{K}$, the proof $\tilde{p}$, and the assertion $\tilde{A}$, from the message $\tilde{\mu}^*$ by following the respective references.
2: Compute $m$ as $m := H(A||p_A||ctx)$.
3: Verify the correct generation of the temporary public signature key by computing $\mathrm{VRFVerify}_y(m, K, p'_k)$: This algorithm verifies the correctness of $K$ using the non-interactive zero-knowledge proof $p'_K$.
4: if verification of $p_K$ in the previous step succeeds then
5:     Verify the non-interactive zero-knowledge proof $p_A$ with respect to $A$ and $y$. This is done following the used non-interactive zero-knowledge proof system.
6:     if verification of $p_A$ succeeds then
7:         Verify $\tilde{\sigma}$ on $\tilde{\mu}^*$ with $K$.
8:         if verification of $\tilde{\sigma}$ succeeds then
9:             RETURN true
10:        end if
11:    end if
12: end if
13: return false
```

3.3 Discussion

We motivate why the construction fulfills the security properties stated in Section 3.1.

The following holds: (1) The proof p_A allows one to prove the assertion A, for example, to convey endorsed user attribute information in case of a private certificate system. (2) The verifiable random function $F_{(x,y)}$ comprising x and the commitment y to x is chosen freshly in each new execution of the protocol. This is required for the unlinkability of multiple transactions by the same party. (3) Binding the function $F_{(x,y)}$ to the proof p_A by signing the commitment y with the proof using the Fiat-Shamir heuristic securely binds $F_{(x,y)}$ to the proof. This prevents anyone who cannot compute the proof p_A from creating the overall signed message. (4) The temporary key pair (K_s, K) being computed via the verifiable random function from p_A, A, and ctx is a DSA key pair used for binding the zero-knowledge proof to the signed message. The discrete logarithm nature of the DSA signature key pair allows for the application of the chosen verifiable random function from the family $F_{(\cdot)}$ for its generation.

From (3) and (4) it follows, that the party having computed the zero-knowledge proof for the correctness of the assertion has created the temporary signing key pair and thus holds the secret key (Property (b)).

Property (c) is achieved as follows: The use of the verifiable random function $F_{(x,y)}$ in (4) assures that K is pseudo randomly derived from the proof p_A. The key pair (K_S, K) is freshly generated in the sense that it has been generated as a random function of the zero-knowledge authentication statement p_A. Furthermore, p_A is guaranteed to be different whenever a new zero-knowledge proof protocol is executed, unless the requestor deviates from the protocol and reuses randomness in the zero-knowledge proof which would make her linkable immediately. Note that the requestor is prevented from choosing the key pair in a way deviating from the algorithm due to the verifiability property. Many applications of XML-DSig 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, thus strengthening the freshness property.

The unlinkability property (Property (d)) follows from (1), the unlinkability of the proof, and (2), the freshly chosen function $F_{(x,y)}$.

Property (a) follows from (1) to (4) above, where it would not be necessary to generate the verifiable random function freshly to achieve (a).

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.

3.4 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 security context ctx 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 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 means that the party signing with the secret key associated to the public key provably has the properties as stated in the assertion.

4. CONSTRUCTION FOR WS-SECURITY

Using the generic construction for XML-DSig as defined in Section 3, we sketch in this section how to transfer the semantics of a non-interactive zero-knowledge-proof-based protocol, such as a private certificate system, to WS-Security. The construction fulfills the same basic properties as the generic construction in Section 3.

The construction provides for a secure binding of the semantics of the zero-knowledge-proof-based protocol to the WS-Security header and thus to the whole Web Services message. This binding is achieved by leveraging our basic construction of Section 3 and transferring the semantics of the zero-knowledge-proof-based protocol to the WS-Security message with the XML signature of the basic construction used as an enveloped signature over the Web Services message.

4.1 Construction

There are different ways in terms of syntax and semantics to do the construction for WS-Security. Every method conceptually requires an *assertion* token $\tilde{A}$, a *proof* token $\tilde{p}$, a *temporary public key* token $\tilde{K}$, and an XML digital signature $\tilde{\sigma}$. The constructions mainly differ (1) by the chosen syntax of the constructs and (2) by merging of certain tokens and only exposing the merged tokens and their (more comprehensive) semantics to the WS-Security processing algorithm.

The commonalities of all constructions are the following: WS-Security-conform tokens have to be used to smoothly extend WS-Security. The token $\tilde{K}$ is referenced from $\tilde{\sigma}$ using the WS-Security-defined `SecurityTokenReference` referencing mechanism [14] that extends the semantics of the XML signature with the semantics of the referred to token. The tokens $\tilde{K}$, $\tilde{A}$, and $\tilde{p}$ contain appropriate references to each other in order for a token referencing another token to include the semantics of the referenced token. The temporary private key K_s is used for making an XML signature, preferably—for security reasons—an enveloped signature over the full message.

4.2 Semantics

The semantics of the generic construction of Section 3 applies to constructions for WS-Security in addition to the WS-Security-specific semantics as outlined next.

The assertion $\tilde{A}$ makes claims about the message producer and relies on the proof token p_A contained in $\tilde{p}$ to prove those claims. The temporary key token $\tilde{K}$ contains the signature verification key K. The key semantics is that the key is temporary and freshly and randomly generated, but this semantics is not endorsed by the token itself, but by a referenced token. The referred to token also provides further third-party-endorsed claims through the proof p_A and the corresponding assertion $\tilde{A}$ it references or is being referenced from. The signature with the temporary private key K_s is a *proof of possession* of this key. The signature is also a *claim confirmation* as it can only be performed by the holder of the private key. This claim confirmation immediately leads to a cryptographic binding of the claims to the Web Service message.

5. CONCLUSION

Our approach of using a temporary signing key pair with certain properties, seems to be the only feasible one within the following requirements: (a) not to change the XML-DSig Standard as it is already stable and changes towards zero-knowledge proofs may complicate the XML-DSig semantics beyond manageability; and (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.

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

6. REFERENCES

[1] Backes, M., Camenisch, J., and Sommer, D. Anonymous yet accountable access control. In *Proceedings of the Workshop on Privacy in the Electronic Society 2005* (2005).

[2] Bangerter, E., Camenisch, J., and Lysyanskaya, A. A cryptographic framework for the controlled release of certified data. In *Twelfth International Workshop on Security Protocols 2004* (2004), LNCS, Springer Verlag.

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

[4] Camenisch, J., Gross, T., and Sommer, D. Enhancing privacy of federated identity management protocols – anonymous credentials in ws-security. Tech. rep., Purdue University, 2006.

[5] Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., and Meyerovich, M. How to win the clone wars: Efficient periodic n-times anonymous authentication. In *ACM CCS* (2006).

[6] Camenisch, J., and Lysyanskaya, A. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In *Advances in Cryptology — EUROCRYPT 2001* (2001), B. Pfitzmann, Ed., vol. 2045 of *LNCS*, Springer Verlag, pp. 93–118.

[7] Camenisch, J., Sommer, D., and Zimmermann, R. A general certification framework with applications to privacy-enhancing certificate infrastructures. In *SEC 2006* (2006).

[8] Camenisch, J., and Stadler, M. Efficient group signature schemes for large groups. In *Advances in Cryptology — CRYPTO '97* (1997), B. Kaliski, Ed., vol. 1296 of *LNCS*, Springer Verlag, pp. 410–424.

[9] Dodis, Y., and Yampolskiy, A. A verifiable random function with short proofs an keys. In *Public Key Cryptography* (2005), vol. 3386 of LNCS, pp. 416–431.

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

[11] Fiat, A., and Shamir, A. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — CRYPTO '86* (1987), A. M. Odlyzko, Ed., vol. 263 of *LNCS*, Springer Verlag, pp. 186–194.

[12] Kaler, C., and Nadalin, A. Web services federation language (ws-federation), version 1, July 2003.

[13] Kaler, C., and Nadalin, A. Ws-federation active requestor profile, version 1, July 2003.

[14] OASIS. Ws-security standard, 2004.