2006 IEEE WEB SERVICES SECURITY SYMPOSIUM

by IEEE Web Services Security Symposium

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Online Proceedings

IEEE Web Services Security Symposium (WSSS) 2006

May 21, 2006, Berkeley, California, USA

FOREWORD

The 2006 IEEE Workshop on Web Services Security was held May 21, 2006, in Oakland, California, USA. The workshop provided a forum for the presentation, discussion, and dissemination of new results on security challenges presented by the Web Services. It was organized in conjunction with the 2006 IEEE Symposium on Security and Privacy.

The program committee selected 6 papers for inclusion into the proceedings. Each submission was reviewed by at least 3 members of the Program Committee. The Program Committee meeting was held electronically. We would like to thank all the authors for submitting to WSSS.

The one day workshop comprised of presentations, followed by discussions of the accepted papers. In addition to the research program, the workshop featured 2 invited talks and panel discussion.

We would like to acknowledge the efforts of those who reviewed the submitted manuscripts and helped realize this workshop.

Sincerely,

Abhilasha Bhargav-Spantzel

Program Chairs
Elisa Bertino
Anoop Singhal

INDEX

AMPol: Adaptive Messaging Policy

Raja N. Afandi, Jianqing Zhang, Munawar Hafiz and Carl A. Gunter[*]
University of Illinois Urbana-Champaign

April 3, 2006

Abstract

Large scale distributed systems often cannot impose a uniform policy on all participants. This can be addressed by making it possible for parties to use diverse policies and adapt to the policies of others in their interactions. In this paper we investigate how to achieve this adaptiveness in messaging systems using a service-oriented architecture called *AMPol (Adaptive Messaging Policy)*, which provides services for expressing policies, finding them, and carrying out system extensions to adapt to them. We implement this approach with a web service middleware that allows parties to use policies for features like attachments, payment, encryption, and signatures. Our implementation demonstrates how AMPol can enhance the function of email messaging by enabling automatic deployment and use of features like cycle exhaustion puzzles, reverse Turing tests and identity based encryption without the need for global deployment or changes to the baseline messaging system.

1 Introduction

Service Oriented Architectures (SOAs) use middleware with standardized interfaces, languages, and protocols to provide interoperability between heterogeneous systems with loose coupling. One challenge for for this objective is support for non-functional (QoS) features like security, availability or reliability without breaking the interoperability of the system. In a highly dynamic and varying environment these features and their constraints may change frequently with each change affecting interoperability and flexibility. Supporting non-functional features in a service-oriented environment is more complex than traditional distributed computing environments (based on DCOM/COM+, J2EE and CORBA) since such behaviors cannot be assumed by client applications and there may be no coupling between requestor and provider. Such systems can be made to function on a large scale in an interoperable way *by allowing participants to dynamically adapt to the requirements of others with whom they need to interact*. In order to fully implement non-functional features, we need an end-to-end adaptive mechanism to support and ensure them on both requestor and provider ends. Also frequent changes and additions of non-functional features require automatic and potentially frequent system updates. SOAs based on web services with meta data specifications offer a promising platform for realizing this kind of scalability and interoperability.

While a variety of studies have explored aspects of how the promise of end-to-end adaptive interoperable systems can be realized, new case studies are needed to reveal more of the requirements and design alternatives. This paper describes one such effort in which we explore an adaptive architecture for large-scale messaging systems such as Internet email, instant messaging, chat rooms, list servers, Wiki pages, blogs, bulletin boards, and so on. In many cases these protocols lack basic facilities for adaptation and this breaks their functionality with the introduction of new requirements. For instance, Internet email messages are

[*]Contact Information: Afandi, Zhang, Hafiz, afandi@illinoisalumni.org, {jzhang24, mhafiz}@uiuc.edu, Gunter http://cs.uiuc.edu/cgunter

often discarded by email relays for reasons that are unknown to senders. In some cases these policies are secrets of the relays or recipients, such as many anti-spam filtering techniques, but, in other cases, they could have been advertised to potential correspondents to facilitate reliable messaging. Examples include rules for the allowed sizes of messages, types of attachments, origin guarantees such as DNS listing, required signatures or encryption, and so on. If these policies can be communicated to senders and they can adapt to them easily, then the overall messaging system can be made more secure and efficient without sacrificing convenience.

In this paper, we propose a reference architecture for adaptive interoperable messaging based on advertisable QoS requirements in a form of policies and a case study of this approach for email based on web services. Our architecture is called *Adaptive Messaging Policy (AMPol)* and it is based on the idea that message senders should learn and adapt to the policies of potential recipients. This is achieved through three fundamental architectural components: (1) a way to describe QoS requirements and constraints of the entities' adaptive behaviors in form of policies (such as the allowed size of messages) called the *policy model*, (2) protocols to publish, discover and merge such policies (such as retrieving them from servers) called *policy discovery*, and (3) means to add capabilities (*e.g.*download and install plug-ins) and enforce policy rules on messages called *Extension and Enforcement (EE)*. Our validation case study realizes AMPol for WSEmail, an approach to Internet email in which legacy protocols such as SMTP, IMAP, and S/MIME are replaced by families of web service (SOAP) calls and email messages are XML documents. We aim to validate AMPol by showing how potentially useful email services could be facilitated by the AMPol architecture to support QoS features in an end-to-end adaptable manner. In particular, our implementation is able to automatically support addition of new QoS requirements for availability and security by deploying and using plug-ins for puzzles (to raise burdens for spammers) and identity-based encryption (which allows senders to encrypt mail for recipients based on email addresses). All this can be done with secure and seamless operations that can realize adaptively in a large-scale messaging system.

AMPol is an end-to-end adaptive middleware based solution for supporting dynamic QoS features while maintaining the interoperability of the system. AMPol's main contributions are its end-to-end solution, adaptive and generic distributed policy framework, reference architecture for adaptive middleware for messaging systems, and application of this middleware for email system based on web services. There have been a number of attempts to exploit service-oriented architectures to make distributed applications more adaptive and interoperable. AMPol differs from these studies in its focus on exploring an end-to-end solution that incorporates all of the necessary adaptive support features. There are few attempts which claim to propose an end-to-end solution for adaptive support of QoS features, but either these efforts are not end-to-end or do not have the flexibility of AMPol.

The paper is divided into seven sections. Section 2 provides some background including some motivation, basics of puzzles and IBE, and an overview of our WSEmail middleware platform. Sections 3, 4, 5 are the core of the paper. They describe our policy model, policy discovery, and EE components respectively by providing the designs of the components and reviewing these with respect to our implementation of the case study. Section 6 discusses some of the work related to adaptive messaging and policy. Section 7 concludes.

2 Background

Consider the challenge of telling potential email correspondents some rules concerning the email you wish to receive. For example, you may wish to specify that attachments must be less than 500KB in size and must be of certain extension types, and that messages from certain parties, like banks and mutual fund companies, must be encrypted using IBE. This is difficult to do with current Internet email facilities because there is no standardized format for expressing these requirements or widely accepted protocols for senders to deal with them. However, such policies could be quite helpful in increasing security, decreasing spam, and avoiding a lot of annoying email mysteries arising from policy conflicts. Quite a bit can be done by filtering, and there are limits on what can be advertised (for instance, it makes little sense to tell spammers what criteria you are using to identify them as such), but, in many cases, it would be helpful to just let potential correspondents know what protocol you would like them to respect in sending you a message. Dually, it would be useful to learn the policies of potential recipients and conform to these (when acceptable) with as little fuss as possible. In short, it would be handy to have more adaptive messaging systems.

Our validation of AMPol as an adaptive messaging system is based on our prototype for the case study of adaptive messaging policy with Puzzles and IBE for a WSEmail messaging system. We describe the background for these here. Their design and implementation in AMPol will be described in subsequent sections.

WSEmail [13] uses the emerging suite of W3C standards and service-oriented computing concepts as a foundation for messaging, rather than trying to design on top of the existing SMTP legacy protocols. It provides for a service-oriented Mail User Agent (MUA) client and Mail Transfer Agent (MTA) server which are designed to provide extensible messaging with plugins that works for both the MUAs and the MTAs. The existing system does not satisfy important requirements of messaging such as a protocol to discover policies or a policy model for messages. One of our key objects was to supply the functional components of adaptive messaging with very few extensions of the WSEmail platform.

Puzzles [10] are a mechanism to prevent DoS attack; in particular, puzzle-based anti-spam email systems have been studied for many years [8]. They aim to increase the cost of sending email in order to exploit an asymmetry between valid senders and spammers. There are two general type of puzzles. One type is a *cycle exhaustion puzzle* such as *hashcash* (`www.hashcash.org`). Another type of puzzle is a *Reverse Turing Test (RTT)*. If the recipient demands puzzle based anti-spam, the sender needs to know what kind of puzzle is required, what the puzzle problem is and where it can get the puzzle plug-in package to resolve the problem. AMPol is to provide for mechanisms to address all of these issues.

Identity Based Encryption [5] is a technique for addressing some of the burdens of key distribution that have made public key encryption of messages less widely used than PKI vendors had hoped. As in the scenario of the puzzle based anti-spam, the sender needs to know that the message should be encrypted by IBE and what IBE tool the sender needs to install. As with puzzles, our goal is to show how AMPol can aid the deployment of IBE without requiring universal adoption of IBE by users.

Our case study uses Voltage's IBE package (`www.voltage.com`) for IBEs, uses a RTT package based on CAPTCHA [19], and a cycle exhaustion puzzle system based on hashcash [3]. All packages are wrapped as COM components. However, the aim of AMPol is to facilitate the deployment of any technique that users think will be effective without the need for a global consensus or changes to the baseline messaging system. For instance, we could also have explored the use of various 'postage' schemes [1, 18] that have been proposed.

3 Policy Model

To achieve the adaptive messaging, there has to be an up-front, declarative way of specifying the behavior that an entity follows and expects from other entities it interact with. Policy model is such a framework for specifying a set of rules or constraints which describe entities' requirements during the communication. To achieve strong expressiveness and unambiguity, the policy constructs should be distinct or modular and different types of rule combination logics should be supported. In a distributed environment, the policy model should be able to specify which entities the policy is applied to and which entity enforce the policy. Only by this, the message conformance (with policy) logic and policy enforcement logic for each individual quality requirement can be taken out of the core policy engine and the true adaptivity is realized. The last point is that policy languages should be generic enough so that the policy schema and core policy engine do not need to be modified by addition of new assertions. The rest of section shows how AMPol's policy model satisfy these requirements.

The basic unit of the policy is the construct called a Rule. Each rule describes an operation or process requirement for a message, *e.g.*encryption and signature. Each rule has two main parts. The first is the *action* that specifies the operation (*e.g.*encryption). The second is the *property* that specifies parameters of this operation (*e.g.*IBE encryption). The Rule's are combined into a RuleSet with connectives AND, OR, EXACTLY-ONE, *etc.*. One or more RuleSet's form a Policy. A Policy is associated with the application level usage (using the App attribute). The policies are specified as ingress policies, egress or local policies which apply to messages in a local domain. Finally, the policies forms the PolicySet which is a set of the policy constructs and storage unit of AMPol policy model.

Policies are classified as static or dynamic. Static policy is defined by each entity itself before the communication occurs. It defines rules that are affective for all conversations. Dynamic policy, which is based on static policy, is a set of actual rules for a particular conversation session. If there are multiple entities involved in a session, each entity needs to know the static policies of the others and create or obtain a set of rules based on all the static policies from every involved entity. The final instantiated rules are dynamic policies; they are generated during policy negotiation. We shall give more details about how to obtain dynamic policies in Section 4.

Existing policy specification languages do not have the concept of meta-specification, *i.e.* "the policies of policy". For example, each Rule or RuleSet needs some meta-information, including who will *perform* the required operation and who will *check* whether the requirement is satisfied. In AMPol policy model, the Subject is the entity the rule or rules set will be applicable to and the Target is the entity enforcing the rule or rules set. In a distributed system, the creator of the rule or the policy might not be the entity who will check the enforcement of the policy. So it is necessary to indicate the target explicitly. When conform or enforce the rules or assertions of a policy, the entity may need some extended functionality to perform the operation. We define Transformation for a Rule, which contains all the information to identify and download a particular plug-in. Policy engine could parse the transformation information and pass it to system extensibility module. The latter can download and execute the plug-in. Thus, if different domain specific rules and corresponding plug-ins are introduced, the policy engine as well as system extensibility module do not need change, *i.e.*, the true adaptivity is realized. All of above meta-information is included in the MetaSpecification, which has extra properties to indicate the visibility and the priority of the rule. One aspect of meta-specifications is that they are applicable at different granularities, both at the rule level and the rules set level.

Our case study is based on a set of rules called *APES* (*A*ttachment, *P*ayment, *E*ncryption and *S*ignature). In APES, there are rules Encryption and Signature to specify the cryptographic parameters (key size, encryption algorithm, version of the encryption algorithm *etc.*) used for encryption or signature. For availability, spam is the main concern. There is a Payment rule that specifies the type of payment imposed on the message sender like RTT or hashcash. Another important security concern is the attachment, which is the primary medium for spreading viruses among email client hosts [14]. There is a Attachment rule that specifies the patterns of the content. An example rule might say that the recipient does not accept an attachment that has a '.pif' extension. (This does not necessarily mean that the sender will not send a .pif file without this particular extension, but proper security would prevent .pif processing of the attachment.)

4 Policy Discovery

After specifying policies individually, the entities need to exchange them and negotiate a mutual acceptable policy set for the current session. Policy Discovery component fulfills this work. This component should be able to publish each entity's policy to everyone involved in the session and ensure the published policies could be accessible remotely. Each entity should be able to query other's static policy or the final dynamic policy. Policy merging of two or multiple policies is required. It also should have a potential to support multi node negotiation. Our design comprises three functional sub-components, which provide the ability to advertise, query and merge policies.

Policy advertisement is used to publish static policies to other entities. The issues relevant to publishing are what to publish, where to publish, and how to publish. The publishable policies are determined using a meta level attribute of the policy model. The published policy is a subset of the static policy of the publishing entity. For the location of policy publication, the main requirement is the remote accessibility and availability. Policies of a particular client can be published at the client node, or at a dedicated policy server or at any remote server. In email systems policies for email clients can be published at a mail server. The entity that is publishing the policy also concerns how the policies are uploaded and maintained in a policy publishing server. This can be done in either the push or pull mode. In pull mode, policy server periodically (or when required) uploads/updates the policy for a particular client. In push mode, policies are uploaded/updated by individual clients. From the policy consumer's perspective, the main issue is the protocol and the location to find the published policies. The protocol for finding policies is implementation specific (*e.g.* HTTP, SSL, Web Service SOAP interface, LDAP, or custom built protocols). The location of a policy hosting node can be known in advance or can be learned by different types of discovery protocols such as UDDI or DNS.

Policy Merging is required to reconcile the policies of diverse parties in PQP exchanges (noted late). The propagated static policies are merged to create the dynamic policy. The message sender is the interested party to get the message recipient's policy and create the dynamic policy. The merging happens at different nodes of the system that lie in the path of the message sender and the recipient. The Policy and RuleSet constructs of the AMPol model have the *CombinationAlgorithm* structure, which specifies policy combination algorithms. If we have two policies with AND combination algorithm, the result of merging is a policy that has all the rulesets combined with an AND combination algorithm.

Policy Query Protocol (PQP) is the fundamental protocol for generating dynamic policies from static policies when a message transmission is required. We classify the entities involved in communication as the initiator and the respondent. The initiator is actually the message sender but in this context it is initiating the protocol. We present PQP for 'four-node messaging' as illustrated in Figure 1.

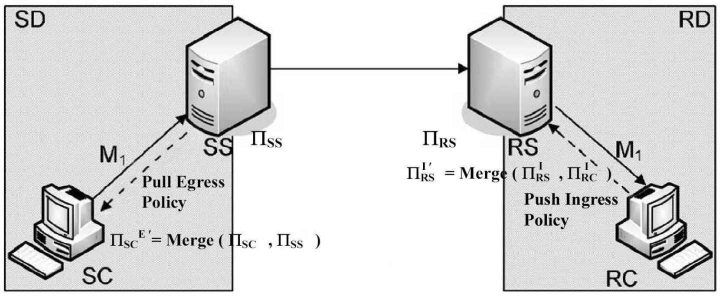

SD
RD
SS
Π_{SS}
Π_{RS}
RS
M_1
M_1
Pull Egress Policy
$\Pi_{RS}^{I'}$ = Merge (Π_{RS}^{I} , Π_{RC}^{I})
Push Ingress Policy
$\Pi_{SC}^{E'}$ = Merge (Π_{SC} , Π_{SS})
SC
RC

Figure 1: Policy exchange and merging

SC is the message sending client. It is connected with SS or the sending server. These two entities constitute the message sending domain. At the message recipient domain, RS is the recipient server that receives the message and forwards it to receiving client RC. The four entities involved have ingress and egress policies specified by the policy model. We denote the ingress policy specification of receiving client (RC) as Π^{I}_{RC}, the egress policy of sending server as Π^{E}_{SS} and so on.

In the policy advertisement mechanism for four node system, the policies can be advertised at the server node. AMPol proposes either pull or push mecahnsim to load ingress policies from clients to a email server which serves as a policy server. The policy server then merges its ingress policy with the client ingress policies. This results in the merged ingress policy for the receiving server and client. It is denoted as: $\Pi^{I'}_{RS} = \textit{Merge}(\Pi^{I}_{RS}, \Pi^{I}_{RC})$. These policies are stored per receiving client and can queried by using a unique identifier which can be fully qualified email address. This whole process is a part of policy advertisement.

The pull and push mechanism and merging of policies in the server is done in the policy advertisement step to achieve improved performance. Another advantage of this is that it does not require RC to be remotely accssible all the time. The clients can also involve in a pull mechanism for added performance. The clients pull egress policies from the server and merge them with their own static policies. This step is done at the start of the PQP protocol. The merging of policies at the client node reduces the requirement of any merging done during PQP. It is denoted as: $\Pi^{E'}_{SC} = \textit{Merge}(\Pi^{E}_{SC}, \Pi^{E}_{SS})$, where $\Pi^{E'}_{SC}$ is the merged egress policy of SC and SS. The equilibrium state reached after exchange of these policies is shown in Figure 1.

The policy query phase can now be described. Step 1: Client SC wants to communicate with RC so it queries the advertised ingress policies for the RC and RS. It sends this query to its own server. SC $\rightarrow$ SS : Query for Π^{I}_{RS} and Π^{I}_{RC}. Step 2: Server SS relays the request to the server of the recipient. SS $\rightarrow$ RS : Query for Π^{I}_{RS} and Π^{I}_{RC}. Step 3: Server RS sends the merged policy $\Pi^{I'}_{RS}$ to the sender's server. RS $\rightarrow$ SS : $\Pi^{I'}_{RS}$. Step 4. Server SS sends the merged policy to the sender client. SS $\rightarrow$ SC : $\Pi^{I'}_{RS}$. Step 5. Client SC merges the received policies with the sending domain's egress policies ($\Pi^{E'}_{SC}$) and sends messages complying with $\Pi^{I'}_{RS}$ and $\Pi^{E'}_{SC}$ via SS. This goes through RS to RC and RC accepts it because it is compliant to its ingress policy. SC $\rightarrow$ RC : Message complying with $\Pi^{I'}_{RS}$ and $\Pi^{E'}_{SC}$.

The protocol above is the base case in which the communication has no cached values for policies. There is a stream-lined version when there is caching since this improves performance. To support caching, the policies have lifetimes. After these expire, policies will be renegotiated.

The implementation of the policy discovery model consists of three main modules: *policy publisher*, *policy merger* and *PQP handler*. The policy publisher service is implemented as a C# .NET Web Service. It reads the stored static policy file(s). The loaded policy is mapped into policy objects and maintained in a key-value map with key as a unique policy identifier. The sender side PQP module initiates the protocol on behalf of the sender client. It retrieves dynamic policy for a particular conversation by calling a *policyQuery* web method of a policy server. It then calls policy conformance module of the EE component for conforming a message to the policy. There is a policy merger module at both server and client nodes as policy merging is required at all nodes.

The AMPol modules are integrated to a messaging application by application-specific hooks. The hooks are placed at appropriate message entry and exit points in an application to intercept these messages and only allow them to proceed further if they are successfully processed by AMPol underlying services. These hooks can either be directly integrated into a source code of the application or plugged into the application if it provides a mechanism for adding plug-ins.

5 Enforcement and Extension

Once mutual acceptable policies have been negotiated, the entities need to determine how to conform to the policies (if it is the sender) or how to enforce the policies (if it is the recipient). AMPol's Enforcement and Extension model deals with extensions to the system of the sender to accommodate recipient requirements if it has determined to do so, and enforcement processing by the recipient to assure conformance to policy. This component should not modify the core base implementation of each entity. It should have meta-level control over the adaptation process itself in order to ensure that changes are carried out effectively. Each extension should be described by policies and be able to run in terms of the policies after interpreting them. Each extension must be implemented as a separate module that can be incrementally added to and removed from the core application by addition or removal of a rule/assertion from the policies. Next, we'll show how AMPol fulfills these requirements and achieve the true adaptability.

The Enforcement and Extension model has three sub-components, each heavily reliant on Extensions. The *policy conformance component* is active at the sender side and the the *policy enforcement component* is active at the recipient side. The *system extension component* is active at both sides. Extensions are realized as third party plug-ins and extensible components that can be dynamically added or removed from the AMPol system. The novel idea behind this model is that we have extracted the logic of conforming a message to a policy rule (policy conformance and message compliance) and logic for enforcing a policy rule (policy enforcement) in an independent and dynamically pluggable process. This pluggable process is called an *extension*. The mechanism to locate, load and execute these extensions is called *system extension*. The interpretation of policy on the sender side is the execution of a series of extensions on a message to conform to the dynamic policy. For policy enforcement at the recipient's end, the recipient enforces the policy by executing the corresponding policy verification functionality on the messages. The policy conformance and the policy enforcement mechanism follows classic pipes and filters architectural style, with the operations for enforcement applied in reverse order.

Each extension application modifies or appends something to the input message. An extension for an RTT puzzle will generate an output message that will contain an original message appended with a puzzle result. Similarly an extension for IBE will generate a modified message which contains the encrypted original message appended with an encrypted symmetric key. These output message formats are defined in meta specification information of both extension and the Transformation part of the meta-specification for a static policy rule.

The enforcement and extension component is implemented using three corresponding modules: *policy conformance*, *policy enforcement* and *system extension*. We have designed and implemented an *AMPol extension framework* inspired by the WSEmail plugin framework [13]. It is a white-box framework and is extended by inheritance. Figure 2 shows the steps followed by the conformance, extension and enforcement

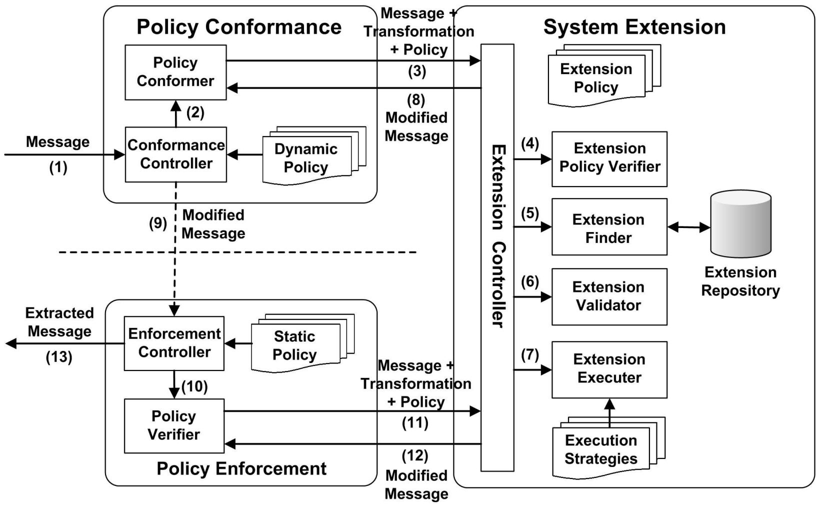


Figure 2: AMPol Conformance, Enforcement and Extensions Components

components of the sender and recipient. They are important parts of PQP.

The PQP module, after retrieving the dynamic policy, invokes the *policy conformance* module by calling the *conformance controller*, which co-ordinates all the processing steps. The controller first identifies the transformations for property rules using the *policy conformer*, then invokes the *extension controller* and passes it the list of transformations along with the policy and a message.

The *extension controller* verifies the transformations against system extension policy. If the verification succeeds, the *extension controller* calls the *extension finder* to find the required extensions. If required, it downloads the extension using the meta-information in the transformations. Then the *extension controller* calls *extension validator* to validate the authenticity of all required extensions. Finally, the controller calls the *extension executor* to execute the extensions. If any extension fails then the whole process aborts with failure. The resultant modified message is returned to the *conformance controller* which returns it to the PQP module.

The message returned to the PQP module is sent to the recipient by the underlying application specific transport mechanism. At the recipient end, the *policy enforcement* module enforces all of the constraints on a received message. The *enforcement controller* co-ordinates all the policy enforcement tasks. It first performs the same policy compliance check as the sender node. The identified list of transformations is passed to the *enforcement controller*, which extracts the original message from the received message.

We have implemented all the AMPol modules and AMPol extension framework in C# .NET and packaged the code in DLLs. We have integrated AMPol modules to WSEmail through hooks developed for WSEmail servers and clients. To implement case studies for RTT and hashcash puzzles we have implemented two extensions for each type of puzzle by extending from the AMPol extension framework. The RTT Extension displays a dialogue box and user is challenged to write the displayed image text in a provided text box. The hashcash extension solves the hash collision problem and appends the result to the input message. In both extensions, the puzzle execution result is appended at the end of the message and the resultant message is returned back to the caller. At the recipient end the same type of extensions are again used for verifying the results. For the IBE case study we implemented the IBE extension in a similar way. The IBE extension first generates an AES symmetric key, and then encrypts the message using this key. The key is then encrypted using the IBE encryption algorithm using the recipient email address as an IBE encryption key. On the other end, the same IBE extension is used to verify and extract the original message. The IBE encrypted symmetric key is decrypted using the recipient's email address. The symmetric key is then used to decrypt the encrypted original message.

6 Related Work

There have been a number of attempts to exploit service-oriented architectures to make distributed applications more adaptive. Meta data approaches [12, 15] support the description, discovery and composition of services using languages such as XACML, WSPL, WS-Policy, and OWL-S to describe QoS requirements and policy constraints. Associated policy processing frameworks are used to enforce requirements for individual entities. Adaptivity is achieved by adding, customizing or replacing entities such as aspects [11], components [2] or concerns [9]. AMPol aims to integrate and extend these types of mechanisms to achieve an end-to-end solution that works in at least the domain of messaging systems.

Dynamic adaptation has been explored [17, 16, 7] in a service-oriented framework to deal with entities that have different QoS requirements on a per session basis. This work uses WS-Policy to describe the QoS requirements; policies are enforced by a framework that supports dynamic binding of non-functional features (such as security and performance constraints) with applications. It provides middleware to achieve cooperation and agreement of requirements between entities but does not provide concrete negotiation protocols and does not explicitly specify which system entity will enforce the policy. There is work on using policy framework and system extensibility to achieve end-to-end adaptability [4] but this work does not support negotiation of requirements and focuses more on system extensibility and policy framework. DySOA [6] is another effort to achieve an adaptive system. It provides a framework for monitoring the application system, evaluating acquired data against the QoS requirements, and adapting the application configuration at runtime. It has a simple manual policy negotiation between the requester and the provider but does not support runtime negotiation. It does not address system extensibility and only re-configures the alternative variables for system parameters. GlueQoS [20] proposes a declarative language based on WS-Policy to specify QoS features and a policy mediation meta-protocol for exchanging and negotiating QoS features. One obvious limitation of GlueQoS is that it does not support dynamic system extensibility; it assumes that both ends have the capability to perform the required operations to fulfill QoS requirements.

AMPol differs from these studies in its focus on exploring an end-to-end solution that incorporates all of the necessary adaptive support features with the aim of customizing the solution to a specific domain (messaging). The goal is to improve validation and explore the extent to which adaptivity is constrained by domain-specific issues. We find that many important issues can be viewed as general concerns not particular to messaging systems, but the domain-specific focus leads us to appreciate the need for new features in the general solutions. For instance, all of the existing systems focus on adaptivity between pairs of nodes, whereas our messaging scenarios call for protocols that adapt the policies of four nodes in a manner that is not just a generalization of the two node protocols. While we are able to provide a treatment of this four node case (PQP), a fully general treatment is still lacking. Aside from this multi-node issue, the main distinctions of AMPol lie in its added flexibility.

Existing efforts assume a built-in logic to enforce policy constraints (QoS requirements) or have a static binding with external processing components to handle policy rules. In AMPol we have proposed an adaptive policy framework by taking out the message conformance (with policy) logic and policy enforcement logic for each individual policy rule out of the core policy engine and have provided this logic in a form of pluggable extensions. The binding logic of a policy rule with a processing plug-in is specified in a policy rule itself instead of hard coding it in a core policy engine. This enables AMPol to be more flexible in its adaptability than other approaches.

The efforts discussed above use WS-Policy framework. which is not generic and adptive enough to support new types of QoS costraints. WS-Policy language is an assertion based declarative language which is based on domain-specific policy specifications such as WS-PolicyConstraints, WS-SecurityPolicy and WS-ReliableMessagingPolicy. These policy specifications have domain specific vocabulary elements and in order to provide a support for new type of assertion (non-functiona constraint), the policy schema needs to be updated and each underlying policy engine must install a new code module to understand and process the semantics of the new assertion type. In contrast, AMPol provides a generic model for rule specifications in which a policy rule is defined as an attribute-value pair. Rules are domain independent so parties extend the specification language without modifying the policy schema and underlying policy engine. Only the policy conformance and enforcement logic at the end nodes needs to be updated.

7 Conclusion

We have introduced an architecture to support adaptive messaging, explained a detailed design, and implemented a prototype to illustrate the concepts and usefulness of the idea. Our architecture, AMPol, is based on functional components for expressing policies, discovering them, and facilitating extensions to conform to and enforce them. Our case study validates the approach and implementation by showing how to deploy puzzles and IBE over a WSEmail platform. This provides one of the most complete studies to date of a proof-of-concept adaptive system based on web services. Our future and current work includes support for multiple recipients (that is, mailing lists), improved security measures such as sandbox protection, features to facilitate error handling, more sophisticated negotiation for policies, and performance testing. Our project web site `seclab.uiuc.edu/ampol` includes a video demonstration of adaptive messaging based on the implementation described in this paper.

References

[1] M. Abadi, A. Birrell, M. Burrows, F. Dabek, and T. Wobber. Bankable postage for network services. In *Proceedings of the 8th Asian Computing Science Conference*, 2003.

[2] M. Aksit, K. Wakita, J. Bosch, and L. Bergmans. Abstracting object interactions using composition filters. *j-LECT-NOTES-COMP-SCI*, 791, 1994.

[3] A. Back. Hash cash - a denial of service counter-measure, 1997. `http://www.hashcash.org/papers/hashcash.pdf`.

[4] F. Baligand and V. Monfort. A concrete solution for web services adaptability using policies and aspects. In *WISE03: Proceedings of the Fourth International Conference on Web Information Systems Engineering*. IEEE Computer Society, 2004.

[5] D. Boneh and M. Franklin. Identity based ecncryption from the Weil pairing. *SIAM J. of Computing*, 32(3):586–615, 2003.

[6] I. Bosloper, J. Siljee, J. Nijhuis, and D. Hammer. Creating self-adaptive service systems with dysoa. In *ECOWS05: Proceedings of the Third European Conference on Web Services*, 2005.

[7] F. Curbera and N. Mukhi. Metadata-driven middleware for web services. In *WISE03: Proceedings of the Fourth International Conference on Web Information Systems Engineering*. IEEE Computer Society, 2003.

[8] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In E. F. Brickell, editor, *Proc. CRYPTO 92*, pages 139–147. Springer-Verlag, 1992.

[9] W. Hürsch and C. V. Lopes. Separation of concerns. Technical Report NU-CCS-95-03, College of Computer Science, Northeastern University, Boston, Massachusetts, Feb.24 1995. URL: `ftp://www.ccs.neu.edu/pub/people/crista/papers/separation.ps`.

[10] A. Juels and J. Brainard. Client puzzles: A cryptographic defense against connection depletion attacks. In S. Kent, editor, *Proceedings of NDSS '99 (Networks and Distributed Security Systems)*, pages 151–165, 1999.

[11] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. In M. Aksit and S. Matsuoka, editors, *Proceedings ECOOP '97*, volume 1241 of *LNCS*, pages 220–242, Jyvaskyla, Finland, June 1997. Springer-Verlag.

[12] L. V. S. Lakshmanan, F. Sadri, and I. N. Subramanian. SchemaSQL — A language for interoperability in relational multi-database systems. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *Proceedings of the twenty-second international Conference on Very Large Data Bases, September 3–6, 1996, Mumbai (Bombay), India*, pages 239–250, Los Altos, CA 94022, USA, 1996. Morgan Kaufmann Publishers.

[13] K. D. Lux, M. J. May, N. L. Bhattad, and C. A. Gunter. WSEmail: Secure internet messaging based on web services. In *International Conference on Web Services (ICWS '05)*, Orlando FL, July 2005. IEEE.

[14] M. McDowell and A. Householder. Cyber Security Tip ST04-010: Using Caution with Email Attachments. `http://www.us-cert.gov/cas/tips/ST04-010.html`.

[15] S. A. McIlraith, T. C. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.

[16] N. Mukhi, P. Plebanni, I. Silva-Lepe, and T. Mikalsen. Supporting policy-driven behaviors in web services: Experiences and issues. In *ICSOC04*. IEEE Computer Society, 2004.

[17] N. K. Mukhi, R. Konuru, and F. Curbera. Cooperative middleware specialization for service oriented architectures. In *WWW'04*. IEEE Computer Society, 2004.

[18] F. Rideau. Stamps vs spam: Postage as a method to eliminate unsolicited commercial email. `http://fare.tunes.org/articles/stamps_vs_spam.html`.

[19] L. von Ahn, M. Blum, N. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In *Proceedings of Eurocrypt*, pages 294–311, 2003.

[20] E. Wohlstadter, S. Tai, T. Mikalsen, I.Rouvellou, and P. Devanbu. Glueqos: Middleware to sweeten quality-of-service policy interaction. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*. IEEE Computer Society, 2004.

Enhancing Privacy of Federated Identity Management Protocols

Anonymous Credentials 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 for stronger user privacy protection than technologies based upon traditional 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 [21], Liberty [19], or WS-Federation [17] excel approaches relying on uncertified attributes in terms of both security [14, 15] and privacy [23, 22]. 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 [10] 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., [1] or a formal definition in [20], and reached a breakthrough in terms of features and efficiency in the Camenisch and Lysyanskaya system [3, 5, 6]. A variant of their system, the Direct Anonymous Attestation (DAA) protocol [2], has been standardized within the Trusted Computing Group (TCG).

Anonymous credential systems have been generlized 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.

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 could 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 zero-knowledge proofs. The following straightforward approaches fail: (a) Placing the proof token into the WS message without any cryptographic binding. (b) Generating a temporary signature key pair and signing the WS message with the proof token fails as anyone can take the proof and re-bind it to another message. The key idea is that it must be guaranteed that the signature key pair can only be generated by the prover depending on the proof and that it be pseudo random.

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 [16] as a running example to be enhanced with those proof semantics. This enables a new WS-Federation Active Requestor profile [18, 17] that uses private certificates instead of traditional identities and therefore leverages all the advantages of the private certificate system.

Our approach seems to be the first 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.

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. We reason why we need private certificate systems in FIM protocols and why this is a hard task to achieve. Section 3 contains the construction of our abstract method as well as its application to a WS-Security token type. We conclude the paper in Section 4.

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) [12] 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 [25] or RSA [24], 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: Zero-knowledge-proof-based protocols are inherently different to traditional digital signatures.

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. It is trust semantics that is left to be extended by applications. This and other trust semantics are to be defined by applications of XML-DSig.

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 with the message is XML-DSig. The basic semantics used in WS-Security is the semantics of XML-DSig. 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 signature is key for our method of transferring the semantics of the proof to the semantics of the XML signature.

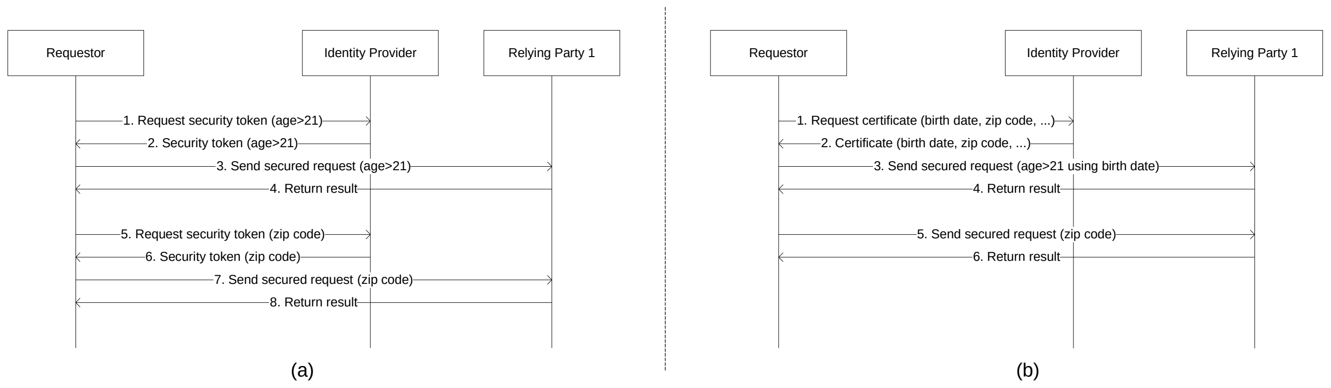
An XML signature used by a WS-Security message takes over the basic signature semantics of XML Signature (basically, integrity) and further extends it. One extension, for example, is the proof of knowledge of a confirmation key for a security token performed with a signature. The security token is typically a public key certificate, however its trust semantics is not specified. 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 [18]. 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. 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



Requestor
Identity Provider
Relying Party 1
1. Request security token (age>21)
2. Security token (age>21)
3. Send secured request (age>21)
4. Return result
5. Request security token (zip code)
6. Security token (zip code)
7. Send secured request (zip code)
8. Return result
(a)
Requestor
Identity Provider
Relying Party 1
1. Request certificate (birth date, zip code, ...)
2. Certificate (birth date, zip code, ...)
3. Send secured request (age>21 using birth date)
4. Return result
5. Send secured request (zip code)
6. Return result
(b)

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

elaborate 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, an overview of private certificate systems is given. Section 2.2.1 outlines the proof protocol for private certificates, Section 2.2.2 their issuance.

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. It is easy to see that the private certificate system 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 be provided in encrypted form* together with a proof that the ciphertext is an attribute of a particular certificate, that is, the recipient can be sure that the requestor does not cheat in what she encrypts. This concept is denoted verifiable encryption [7] 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.

2.2.1 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.

[1]It is worth noting that there are certificate revocation mechanisms available, see for example [4]; revocation is an important feature when considering the long lifetime of idemix certificates.

A non-interactive variant of the protocol can be obtained by applying the Fiat-Shamir heuristic [13] 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 propositional logic without negation and with predicates. 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.[2]

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:

[2]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.

Formula	$passport.bdate \leq current\_date - 21 years \wedge passport.holder = idcard.holder \wedge idcard.zipcode = 8004 \wedge enc_1 = passport.snr$
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 [8] 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 greater than 4000, without revealing the attribute. All these protocols are computationally very efficient and thus the idemix system is highly practical.

2.2.2 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 CERTIFICATEPROOF 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 [24], 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.

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.

2.3 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, that is it 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 verifiably 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 like RSA or DSA. As the idemix protocols are technically quite different—they are based on (non-interactive) zero-knowledge proofs of knowledge—they cannot be used as yet another signature scheme within XML-DSig unless the standard's explicit and intended semantics is extremely widely interpreted and at some points violated. As one example, the XML Signature Standard does not provide a facility for inputting an assertion to the signature generation algorithm, but the idemix private certificate system requires an assertion and also private certificates as algorithm input. This shows only one of the many incompatibilities of the traditional XML-DSig standard and idemix.

3 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 3.1. This construction generalizes to any conceivable signature standard with comparably extensible semantics. As for the importance of the WS-Security protocol framework, we show in Section 3.2 how the construction is carried over to a WS-Security-protected message.

3.1 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 semantics from an authentication with a private credential system into an authentication with a *temporary* DSA signature key pair (K_S, K) with which an XML-DSIG can be generated and verified.

3.1.1 Security Properties

For transferring the authentication, our security model does not require the requestor to be trusted. Therefore, we 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 chooses a pseudo-random and fresh temporary signature key for each single zero-knowledge-proof-based authentication.

The 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.

The following list summarizes the properties our construction achieves:

(a) The party creating the signature has attributes as asserted by the zero-knowledge-proof-based authentication.

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

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

(d) Multiple authentications with different zero-knowledge authentication statements are unlinkable if the zero-knowledge authentication statements are unlinkable.

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

3.1.2 Overview of the Construction

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.

For each statement made with a zero-knowledge-proof-based protocol, we guarantee that a new temporary key pair be 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 be able to determine the key in a different manner.

The claims are manifest in 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, e.g. 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, typically 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 we stress the same method applies to any non-interactive zero-knowledge-proof-based protocols that can be used to sign a message. Thus our construction is quite general.

3.1.3 System Setup

The system setup is accomplished by executing Algorithm 1 and creates the system parameters.

Algorithm 1 Setup

```
1: Generate a prime order $q$ cyclic group $G$ with $g$ as generator as follows: Generate a prime $p$ and a prime $q$
   such that $q|p-1$. Note that $p$ and $q$ are chosen to also fulfill the requirements of the DSA signature scheme
   [25]. Generate a generator $g$ as follows: choose a random element $r \in_R Z_p$ and compute $g := r^{(p-1)/q}$. The
   elements $\langle g \rangle$ are the elements of $G$.
2: RETURN $(p, q, g)$
```

Note that G and g are chosen such that they can be used as public system parameters of a DSA signature scheme used by all system participants. Arithmetic in the group is done modulo p. Note that (G, g) defines a verifiable random function (VRF) family $F_{(\cdot)}$ as defined in [11]. A VRF is much like a conventional pseudo-random function, only that it in addition generates a proof of correct generation of the pseudo-random elements. A verifiable random function family comprises the three algorithms VRFGEN, VRFPROVE, and VRFVERIFY. VRFGEN generates a new verifiable random function of the family, VRFPROVE creates a pseudo-random element and a proof of its correct generation, and VRFVERIFY verifies such a proof on an element. For details on VRFs see [11].

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 SIGN operation of a message performed by a party, sending the signed message to a verifier, and a VERIFY operation done by the verifier. The SIGN and VERIFY algorithms for the signature are explained in detail below. Let $H()$ be a cryptographically secure hash function.

3.1.4 Signature Generation

Algorithm 2 generates an XML signature on an XML message $\tilde{\mu}$ and binds the assertion and proof securely to the message using the extensibility of XML-DSig.

The output of the algorithm is a signed document with the semantics of an idemix certificate proof being transferred to the verification key and thus also the XML signature. Figure 2 depicts the structure of the resulting token when a WS-Security message is being signed with the temporary key using the *enveloped signature transform* as specified in [12].

Note that the value y acts as a commitment to the verifiable random function F_x. The values x and y are the secret respectively public key of the function F_x. The non-interactive zero-knowledge proof of knowledge p'_K shows that y and K have been computed according to the protocol and thus accounts for the verifiability aspect of the verifiable random function. The proof p_A is the non-interactive version of the proof protocol for the holdership of certificates including corresponding zero-knowledge proofs to prove properties of attributes. The variable ctx can contain other security-relevant context information such as time or a nonce; this depends on the application the construction is used for.

We do not give the complete syntax for the XML tokens here as the syntax is not constrained by XML-DSig and it is analogous to the syntax we use in the method for WS-Security as shown further below in Section 3.2, but

Algorithm 2 Sign

```
1:  Create a new verifiable random function $F_x$ from the function family $F_{(\cdot)}$ as follows: Choose the secret key of $F_x$ as $x \in_R \{1, ..., q\}$. Compute the public key $y$ of $F_x$ as $y := g^x$.
2:  Create an idemix assertion $A$ as required for the transaction.
3:  Compute the idemix proof $p_A$ for the correctness of the assertion $A$ following the idemix protocol CERTIFICATEPROOF (see Appendix 2.2.1) for releasing data using private certificates. During the computation of the non-interactive idemix proof the value $y$ of the verifiable random function is used as message to be signed by the proof using the well-known Fiat-Shamir heuristics. This Fiat-Shamir signature securely binds the verifiable random function to the proof.
4:  Let $m$ be defined as $m = H(A||p_A||ctx)$ where $ctx$ is the security context.
5:  Compute the temporary signature key $K$ by using the algorithm $\text{VRFPROVE}_x(m)$ of the verifiable random function $F_x$. In more detail, $K$ is computed as $K := g^{\frac{1}{x+m} \pmod q}$ and a non-interactive zero-knowledge proof $p'_K$ attesting that $K$ has been computed correctly via $F_x$ is computed as well. The proof being computed is the following, using the widely-used zero-knowledge proof specification language introduced by Camenisch and Stadler in [9]:
    $$SPK\{(\alpha, \beta) : y = g^\alpha \wedge K = g^\beta \wedge g = y^\beta g^{m\beta}\}() \quad (1)$$
    The variable $\alpha$ maps to the signer's secret $x$ and $\beta$ to the signer's secret $\frac{1}{x+m} \pmod q$. Note that the construction of the proof differs from Dodis and Yampolski's constuction [11].
6:  Compute the temporary secret key $K_s$ as $K_s := \frac{1}{x+m} \pmod q$.
7:  Construct the assertion token $\tilde{A}$ such that it encapsulates the assertion $A$.
8:  Construct the proof token $\tilde{p}$ such that it contains the proofs $p_A$ and $p_K = (p'_K, y)$. Add a reference ref$[\tilde{A}]$ to the proof token.
9:  Create the public key token $\tilde{K}$ containing the key $K$. Add a reference ref$[\tilde{p}]$.
10: Compute an XML signature $\tilde{\sigma}$ with the private signing key $K_s$ over the input message $\tilde{\mu}$ and the security tokens $\tilde{A}$, $\tilde{p}$, and $\tilde{K}$, add a reference to the public key token $\tilde{K}$ to the signature, and add the signature to the document resulting in $\tilde{\mu}^*$.
11: RETURN $\tilde{\mu}^*$
```

not constrained by WS-Security. The semantics of the tokens and references is outlined further below in Section 3.1.7.

The signature generation can make use of an unchanged library for XML-DSig generation. This is a major benefit of our solution.

3.1.5 Signature Verification

The verification of a message $\tilde{\mu}^*$ with signature received from a requestor is performed with Algorithm 3.

Algorithm 3 Verify

```
 1: Extract the signature element $\tilde{\sigma}$ from the message $\tilde{\mu}^*$.
 2: Extract the reference ref$[\tilde{K}]$ to the temporary public key token from the XML signature $\tilde{\sigma}$.
 3: Obtain the reference ref$[\tilde{p}]$ to the proof token from the public key token.
 4: Compute the value $m$ as $m = H(A||p_A||ctx)$. Note that the security context ctx can be derived in a well-
    determined way and must match the context having been used by the requestor.
 5: Verify the correct generation of the temporary public key by using the $\mathrm{VRFV}\textsc{erify}_x$ algorithm. This algo-
    rithm verifies the zero-knowledge proof $p'_K$ from $\tilde{p}$ using $p_K$, $m$, and $K$ as input.
 6: if verification of $p_K$ succeeds then
 7:   Follow the reference ref$[\tilde{A}]$ in $\tilde{p}$ to the assertion token $\tilde{A}$.
 8:   Verify the proof $p_A$ with respect to $A$ and $y$.
 9:   if verification of $p_A$ succeeds then
10:     Verify the XML signature $\tilde{\sigma}$ on $\tilde{\mu}^*$ with the temporary public key $K$ using the process as defined in [12].
11:     if verification of $\tilde{\sigma}$ succeeds then
12:       RETURN true
13:     end if
14:   end if
15: end if
16: RETURN false
```

If Algorithm 3 returns true, the public key K is valid and has the semantics as outlined below. In this case, any further processing depends on the trust semantics of the application layer that goes beyond the semantics of XML-DSig. In case the algorithm returns false, the web services message is invalid.

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 [11] for details on the verifiable random function we use in our construction.

3.1.6 Discussion

We motivate how the construction fulfills the security properties stated in Section 3.1.1.

(0) The proof p_A allows for securely conveying attributes of the requestor to the relying party. (1) The verifiable random function F_x 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. (2) Binding the function F_x to the proof p_A by signing the commitment y with the proof securely binds F_x to the proof. This prevents anyone who cannot compute the proof p_A from creating the overall signed message. (3) The temporary key pair (K_s, K) being computed via the verifiable random function F_x from p_A, A, and the security context 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. As DSA is a signature algorithm required by the XML Signature Standard, every standard conform implementation of XML-DSig can be used for our extensions.

From (2) and (3) it follows, that the party having computed the zero-knowledge authentication statement 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 in (3) 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 (0), the unlinkability of the proof, and (1), the freshly chosen function F_x.

Property (a) follows from (0) to (3) above, where it would not be necessary to generate the verifiable random function freshly to achieve (a).

3.1.7 Semantics

The temporary public key token has the semantics that the contained public key and corresponding private key are freshly generated pseudo-randomly from A, p_A, and a security context ctx. That is, the requestor cannot have freely chosen either part of the key pair, but is constrained by the algorithm. Further claims proved by a proof token p_A contained in $\tilde{p}$ and specified by the assertion A 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 a signature that is 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 has proved the assertion A.

3.2 Construction for 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.

3.2.1 Properties

This construction allows for securely binding claims authenticated by any non-interactive zero-knowledge-proof-based protocol to web services messages. The secure binding fulfills the same properties as the generic construction in Section 3.1.1 does, but with semantics specific to WS-Security.

When the idemix assertion token and proof token were just 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 of the web services message. This binding can be achieved by leveraging our basic construction of Section 3.1 and transferring

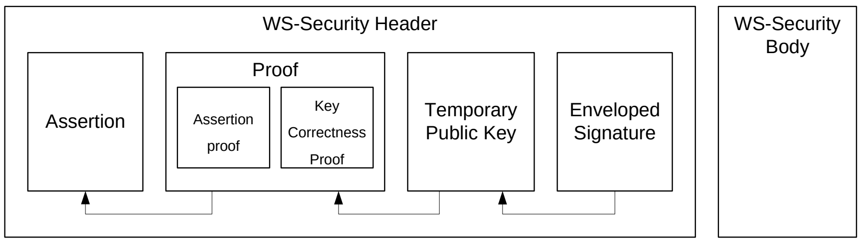


Figure 2. Structure of a WS-Security token with a zero-knowledge proof

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, that is, further constrained and more specific. 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.

3.2.2 Message Generation

The generation of a secured web services message proceeds like the SIGN algorithm, with the key differences pointed out in the following: We again require to construct an *assertion* token, a *proof* token, and a *temporary public key* token for our method (see Figure 2 for a graphical representation and Appendix A for an XML skeleton). For WS-Security integration, they have to be designed specifically as WS-Security security tokens to account for the syntactical and semantical requirements of WS-Security.

The temporary public key token $\tilde{K}$ contains a key claim regarding the key and its properties, but no further associated claims. With the `AssociatedEndorsedClaims` element of WS-Security, the token references the proof token $\tilde{p}$.[3]

Our newly generated security tokens are added to the web service message $\tilde{\mu}$.

The temporary private key K_s is used for making an enveloped XML signature over the complete web services message $\tilde{\mu}$. The signature is extended with a `SecurityTokenReference` element for referencing the public key token.

Semantics. The semantics of the basic construction for the proof with assertion and the temporary key remains unchanged with respect to the semantics in Section 3.1.7 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 a *proof-of-possession* in the WS-Security sense 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 WS-Security specification defines the `SecurityTokenReference` mechanism for referencing security tokens which extends the semantics of the XML signature by the 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 key token itself does not endorse its claimed key semantics of being freshly and pseudo-randomly generated. The endorsement and further claims are contained in the token referenced from within the `AssociatedEndorsedClaims` element. The semantics is that the referenced token (possibly jointly with further tokens) provides a proof for the claimed key properties and possibly also further third-party endorsed claims that

[3]The referenced security token can again reference others or be referenced from others, thus allowing for flexibility and extensibility.

are bound to the temporary public key. In our construction, the referenced token is the proof token. The proof token $\tilde{p}$ contains the proof of the claimed key properties in p_K and a proof of additional claims made by the assertion A.

4 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 intentions 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 and 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 an easy application in an industrial WS-Security environment.

References

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

[2] 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.

[3] 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.

[4] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In M. Yung, editor, *Advances in Cryptology — CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer Verlag, 2002.

[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 and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In D. Boneh, editor, *Advances in Cryptology — CRYPTO 2003*, LNCS, 2003.

[8] 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.

[9] 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.

[10] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, Oct. 1985.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] P. Hallam-Baker, C. Kaler, R. Monzillo, and A. Nadalin. Web services security: SOAP message security, 2003.

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

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

[19] Liberty Alliance Project. Liberty Phase 2 final specifications, Nov. 2003. `http://www.projectliberty.org/`.

[20] 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.

[21] OASIS Standard. Security assertion markup language (SAML) V2.0, Mar. 2005.

[22] 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`.

[23] 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.

[24] 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.

[25] 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 -->
    <SecurityTokenReference 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>
...
```

Semantic-Aware Data Protection in Web Services

Csilla Farkas[1]
Amit Jain[1]
Duminda Wijesekera[2]
Anoop Singhal[3]
Bhavani Thuraisingham [4]

[1] Center for Information Assurance Engineering, Dept of Computer Science and Eng.,
University of South Carolina, Columbia, SC-29208,
[2] Center for Secure Information Systems, Dept of Info. and Software Eng.,
George Mason University, Fairfax, VA 22030–4444,
[3] National Institute of Standards and Technology,
[4] University of Texas at Dallas
email:{farkas|jain}@cse.sc.edu,dwijesek@gmu.edu,psinghal@nist.gov,
bhavani.thuraisingham@utdallas.edu

Abstract

This paper presents a method to remove the dependency of XML access control models on the syntactic representation of the XML trees. We propose a semantics-based approach, expressing XML access control on ontologies that describe the XML documents. The semantics-based model is transformed to the syntactic representation on the actual XML instance. Our method supports the uniform enforcement of an authentication model on syntactically different but semantically equivalent XML documents.

Contact Author: Csilla Farkas, Dept. of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208.
Telephone: 803-576-5762, Fax: 803-777-3767, email:`farkas@cse.sc.edu`

1 Introduction

Current trends of Web applications indicate that Web Services will become a fundamental technology for Web-based applications. Although Web Services (WS)are rapidly developing with a set of standards, like WSDL [3], UDDI [4], SOAP [7], and WS-Policy, current WS capabilities are limited to express application specific semantics. Further, a crucial aspect of WS is to achieve high-assurance security. Several standards, like SOAP message security [15], SAML, XACML have been developed along this goal. Specifications like [6, 14] are some other initiatives taken by organizations like IBM, Verisign, Microsoft to develop policy languages for WS security. XML is a fundamental technology used across the various components of WS, conveying application specific semantics. Several XML access control models have been developed [5], [2], [13] to provide controlled accesses to XML formatted data. However, XML access control models are based on XML syntactic constructs without incorporating data or application semantics.

Based on the use of XML to convey application specific semantics, we propose an approach to secure XML documents based on these semantics. Access control requirements are defined on ontologies, then propagated to the XML documents. Our approach supports not only semantics-aware access control but also protects against security violation via XML rewrite attacks. The syntax-based approach provides limited support for fully machine understandable processing and even simple manipulations of an XML file can bypass the security requirements. For example, the same XML file might be restructured when passed between two WSs. In the example shown in Figure 1, the data originating from a patient may be presented in a different syntactic form when it is shared between the two Web Services. The XML structure and syntax-based access control policy over the original data may not be correctly represented on the modified XML files.

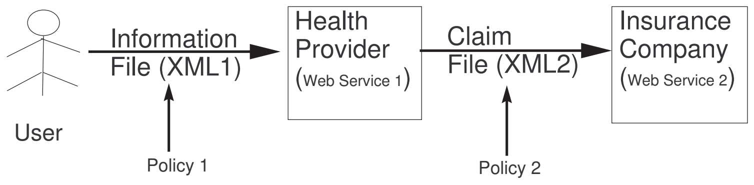

User
Information File (XML1)
Policy 1
Health Provider (Web Service 1)
Claim File (XML2)
Policy 2
Insurance Company (Web Service 2)

Figure 1: Web Service Data Sharing scenario

One of the features of XML is that several XML trees can be developed to represent the same original data. For example changing the element nesting, cardinality constraints, or replacing the element names with abbreviated text or synonyms are user defined and different preferences lead to different XML syntax [1]. Similarly a data item in XML can be designed either in the element or attribute form [16], based on the application requirements. Authorizations defined for an XML document would not work if the structure were changed syntactically even if it would correspond to the same original data. To ensure correct enforcement of the XML access control, it is necessary to provide transformation of policies over different syntactic forms. Performing these transformations by human expert is time consuming and may lead to errors in the case of complex policies. Automated tools that are capable of identifying restructured data and verify the satisfaction of original access control requirement on the restructured data are needed.

Our approach is based on the data semantics as defined by the mappings from XML documents to ontologies. None of the existing XML access control models utilize these mappings to express access control requirements in a syntax independent manner. Even simple modification of data structure would allow a malicious (or careless) user to violate the original access control requirements. Syntactic variations make access control in one XML file unusable or incorrect in a different XML structure over the same data. In [17] Qin and Atluri propose a concept-level access control for Web data, where access control is defined on ontological concepts and instances of these concepts inherit the access control of the concepts they belong to. They incorporate relationships supported by the ontology. However, they do not define how the instances are associated with the ontological concepts and how to enforce the access control requirements on these concepts.

In this paper we provide a framework to enable semantics-based access control model using the existing XML access control technology. The basis of our work is to "‘enforce"’ uniform policy requirements on semantically equivalent but syntactically different XML trees. Semantic units are identified by the mappings from the XML trees to a common ontology.

Our approach is the generalization of the works of Kodali et al. [10], [8], [9], [11], [12] on SMIL formatted multimedia security. They propose the concept of SMIL Normal Form for the representation of SMIL equivalence classes. These classes constitute syntactically different SMIL fragments with the same operational semantics.

[1]Note, that under XML syntax we consider syntax of the element and attribute names, and the structure of the XML tree

We incorporate data and application specific semantics, represented as an ontology that is associated with the XML tree, to define access control requirements. Consider patient information given in both XML documents in Figure 2 and Figure 3 from the previous example in Figure 1. Both trees were created from the same original data. Since XML access control policies are expressed on the XML tree syntax, different policies need to be developed for the two different representations. Further, no verification can be performed that semantically equivalent components have the same access control requirements.

```xml
<Information db="dbXML" network="HMO">
   <Dept id="1559" branch="Cola" specialist="Int Med">
      <PatientList>
         <PatInfo>
            <Patient ssn="1237435" dob="01/01/1950">
               <Name>
                  <First>John</First>
                  <Last>Fisher</Last>
               </Name>
               <Visit no="2" date="06/12/2004">
                  <Problems>Headache</Problems>
                  <Provider hospital="Baptist">
                     <Physician>Dr. Mark Winderip></Physician>
                  </Provider>
                  <Diagnosis>
                     <Prescription>Tylenol</Prescription>
                     <Allergy>Sugar</Allergy>
                  </Diagnosis>
               </Visit>
            </Patient>
         </PatInfo>
      </PatientList>
   </Dept>
</Information>
```

Figure 2: Data in XML Format 1

```xml
<Information db="dbXML" network="HMO">
   <Dept id="1559" " specialist="Int Med">
      <Branch>Cola</Branch>
      <PatientList>
         <PatInfo>
            <Patient dob="06/12/2004>
              <Ssn>1237435</Ssn>
               <Name>
                  <First>John</First>
                  <Last>Fisher</Last>
               </Name>
               <Visit no="2">
                  <Date>06/12/2004</Date>
                  <Problems>Headache</Problems>
                  <Provider hospital="Baptist">
                     <Physician>Dr. Mark Winderip></Physician>
                  </Provider>
                  <Diagnosis>
                     <Prescription>Tylenol</Prescription>
                     <Allergy>Sugar</Allergy>
                  </Diagnosis>
               </Visit>
            </Patient>
         </PatInfo>
      </PatientList>
   </Dept>
</Information>
```

Figure 3: Data in XML Format 2

For example, assume that the access control policy requires that a subject John is permitted to read the SSN of the patients. Expressing this requirement would use different XPath expressions in the two XML documents, because in XML format 1, SSN is represented as an attribute of the tag patients, while in XML format 2, it is represented as sub-element of the tag patient. Different examples are given in Figures 4 and 5. These trees have the same information data but they differ in their syntax. Similar to the previous example, two different access control policies need to be developed to secure the two XML trees.

We define access control policies on semantic data units, and develop automated transformation of these policies to the different syntactic forms.

The organization of the paper is as follows. The section 2 explains syntactic manipulation of XML trees and their mappings to ontologies. Section 3 describes the ontology-based access control framework for XML documents. We conclude in Section 4 and recommend future work direction.

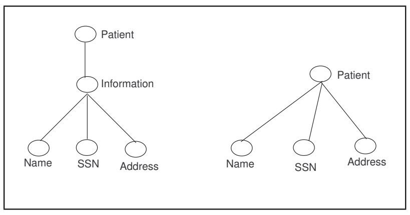


Figure 4: XML Data restructured

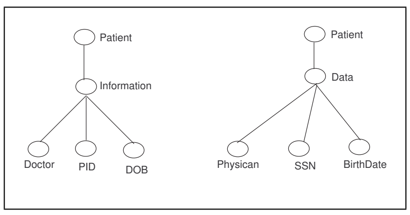


Figure 5: Different Data Syntax

2 Syntactic Manipulation of XML trees

In this section we provide definition of ontologies, XML trees, and mapping between them. We adopt the formalism of An et al. [1] defining the mappings between XML documents (schemas) and corresponding ontologies. We also propose an XML tree format, called element-only XML tree, that is equivalent to the general XML tree but will simplify our arguments. We provide transformation from an arbitrary XML document to its element only form. The aim of this transformation is to be able to define syntactic constructs that belong to the same equivalence class in a uniform manner. First, we define our XML trees.

Definition 2.1 (XML document tree) *An XML document consisting of elements and attributes is defined recursively as an ordered labeled-tree as follows:*

1. *The Empty set { } is a tree, called empty-tree.*
2. *A single node $\{n\}$ is a tree where n is a node or an attribute label, or an element or attribute value.*
3. *If t_1, t_2, ... t_k are trees then $\{n \rightarrow \{t_1, t_2, ..., t_k\}\}$ is a tree, where n is the label of the root node and has outgoing edges to subtrees $t_1, t_2, ...t_k$.*

In general XML trees nodes may correspond to elements or attributes. For uniform representation, we propose that all attributes $(a_1, \ldots, a_k)$ of an element n are treated as subelements of n, with the cardinality restriction, that a_i $(i = 1, \ldots, k)$ occurs 0 or 1 time. The transformation of a regular XML document to its element-only form is trivial and we omit it. Figure 6 shows the Element Only Form of the XML files that were given in Figures 2 and 3.

Ontologies provide the semantics XML documents convey. Our access control is based on this meaning to define the security requirements. For

```
<Information>
  <db>dbXML</db>
  <network>HMO</network>
  <Dept>
    <id>1559</id>
    <specialist>Int Med</specialist>
    <Branch>Cola</Branch>
    <PatientList>
      <PatInfo>
        <Patient>
          <dob>06/12/2004</dob>
          <Ssn>1237435</Ssn>
           <Name>
             <First>John</First>
             <Last>Fisher</Last>
           </Name>
           <Visit>
             <no>2</no>
             <Date>06/12/2004</Date>
             <Problems>Headache</Problems>
             <Provider>
               <hospital>Baptist</hospital>
               <Physician>Dr. Mark Winderip></Physician>
             </Provider>
             <Diagnosis>
               <Prescription>Tylenol</Prescription>
               <Allergy>Sugar</Allergy>
             </Diagnosis>
           </Visit>
        </Patient>
      </PatInfo>
    </PatientList>
  </Dept>
</Information>
```

Figure 6: Data in XML element only form

this, we define a mapping from the XML trees to their corresponding ontologies. Since these mappings are based on the data and application semantics, rather than the original XML syntax, semantically equivalent XML documents will map to the same ontological components.

An ontology is a 6-tuple $(C, P, C_H, P_H, dom, range)$ where C is a set of class names, P is a set of property names (including data type properties), C_H is the class hierarchy, P_H is the property hierarchy, dom is the domain of a property in P, and $range$ is the range of each property in P.

We represent ontologies as directed, node and edge-labeled graphs. Each node corresponds to a class or a data type, and each edge corresponds to a property or the subclass hierarchy. We say that the domain of a property p is $dom(p) = \bigcup c_i$ for all c_i such that 1) there is an edge labeled p in the ontology originating from c_i or 2) there is an edge originating from c' and labeled p in the ontology, such that $c_i \leq_{C_H} c'$. For each class c in O the properties of c, denoted as $p(c)$, are the union of all properties p such that

$c \in dom(p)$ or $c' \in dom(p)$, where $c \leq_{C_H} c'$. Similarly, we say that the range of a property p is $range(p) = \bigcup c_j$ for all c_j such that 1) there is an edge labeled p in the ontology pointing to c_j or 2) there is an edge pointing to c' and labeled p in the ontology, such that $c_i \leq_{C_H} c'$. For simplicity, each graph is converted into directed, edge and node labeled tree [1].

Definition 2.2 (Ontology) *Let C denote the set of classes, P the properties, τ the data types, and X a set of variables. An ontology O is represented as the conjunction of literals, where literals are defined as follow:*

1. *$c(x)$, where $c \in C$ and $x \in X$*
2. *$p(c_i, c_j)$, where $p \in P$ and c_i, c_j are either classes in O or variables*
3. *data types f are represented as "artificial" nodes with the special property f labeling the edge to the node*
4. *$dom(p) = \bigcup c_i | \exists p(c_i, c_j)$*
5. *$range(p) = \bigcup c_j | \exists p(c_i, c_j)$, where c_j may be a literal*

To incorporate data semantics in our XML access control model, we need to define a mapping ν, from XML trees to partial ontologies. ν maps each node of the XML tree to a class, a data type, or an edge of the ontology. Edges of the XML are mapped to paths of the ontology. We use this mapping to assign access control requirements to the syntactic XML trees, based on the access control specified over the corresponding ontology.

Definition 2.3 (Mapping Correspondence) *Let ν be a mapping from and XML tree T to ontology O. Then, the correspondence between the elements and edges of T and the elements and edges of O is defined as*

1. *$T.e \rightarrow_{corr} O.c$: an element e of T, identified by $T.e$, where T is a path expression that leads to e, corresponds to a class c in O.*
2. *$T.E \rightarrow_{corr} O.p(c_i, c_j)$: an element e of T, identified by $T.e$, where T is a path expression that leads to e, corresponds to an edge labeled p, originating from c_i to c_j in O*
3. *$(T.e_1, T.e_2) \rightarrow_{corr} \{O.p(c_1, c_2) \rightarrow \ldots \rightarrow O.p(c_n, c_{n+1})\}$: an edge in T between $T.e_1$ and $T.e_2$, identified as a pair of nodes $(T.e_1, T.e_2)$, corresponds to a path originating at $O.c_1$ and terminating at $O.c_{n+1}$ in O.*

We require that all nodes of an XML tree are mapped to either nodes or edges of the ontology, however, the ontology may have nodes, edges that do not participate in the mapping.

The mapping from an XML subtree maps to the classes and properties of the ontology, such that each element name is mapped to either a class name or to a property name. The access restrictions for a node in the XML tree are based on the access restrictions on the corresponding ontology components.

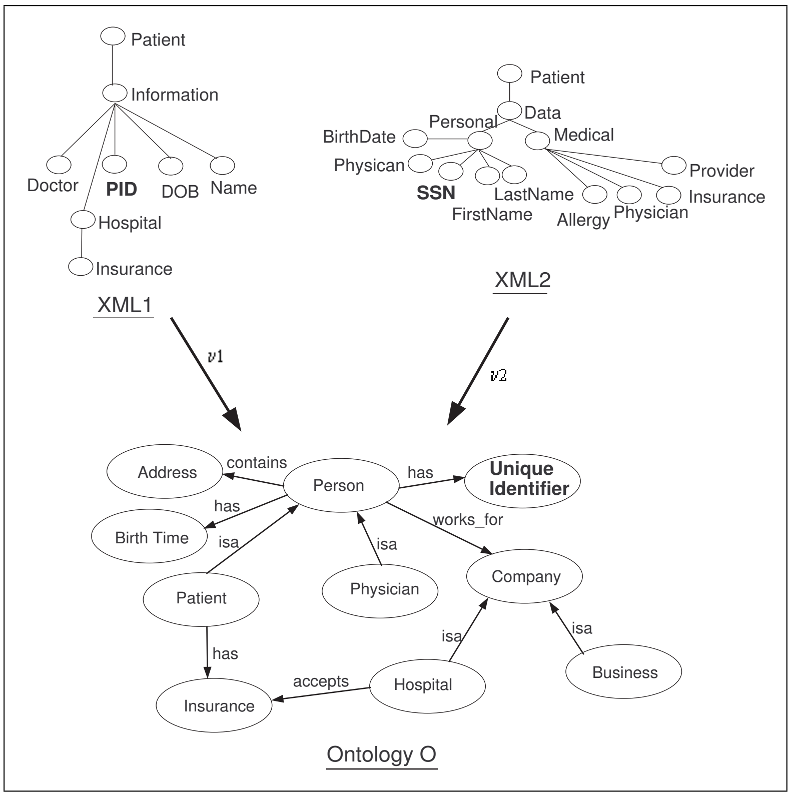


Figure 7: XML to Ontology Mapping

XML documents with different syntax and semantics may be mapped to the same partial ontology (see Figure 7).

3 Semantics-Based XML Access Control Architecture

We use the mapping between XML subtrees and partial ontologies to define access control for an XML document. Our approach allows the use of existing access control mechanisms, like [2], while supporting access control specification based on data and application semantics.

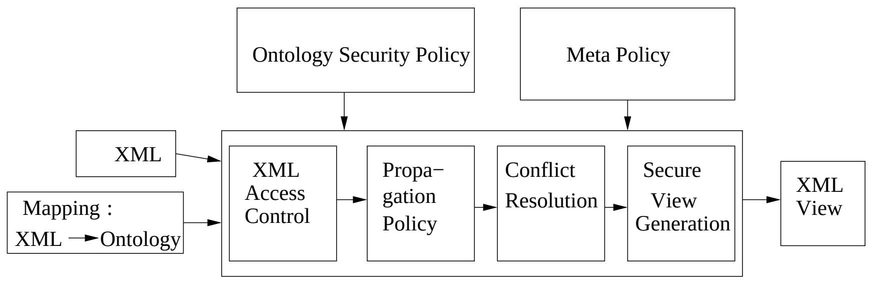


Figure 8: Access Control Mapping Architecture

Figure 8 shows the proposed architecture to assign access control requirements to XML documents. The access control module accepts an XML document, an ontology and its security policy, and the mapping from the XML document to the ontology as the input. A security policy, applicable to the XML document is constructed using the mapping and the security policy for the ontology. Additional input, containing propagation policy, conflict resolution and XML restructuring requirements, may be used to aid complex security policies.

For the purpose of this paper, we show how our framework handles Discretionary Access Control. More advanced access control models can be similarly expressed by our model.

Definition 3.1 (Access Control Policy over Ontology) *Access control is defined as a set of triples of the form $(s, o, < sign > a)$, where s is the subject, o is the object, and $< sign > a)$ is a signed action type.*

Definition 3.2 (Security Object) *Let O be an ontology. A security object is either a class $c \in C$ or a property p of a class c.*

Definition 3.3 (Security Assignment) *Let Pol define the security policy for an ontology O, and ν be the mapping from an XML tree T to O. For all nodes $T.n$, if $\nu(T.n) = O.o$, where $O.o$ is a security object in O, then create a triple $(s, T.n, < sign > a)$ for all triples $(s, O.o, < sign > a) \in Pol$.*

Clearly, because there is an association between all components of an XML and some of the components of an ontology, there is an access control rule for each XML component. However, the access control requirements for the components may not follow current XML access control requirements, like increasing restrictions traversing down the XML tree. Stoica et al. addressed this problem in [18].

4 Conclusion and Future Work

In this paper we presented a syntax independent method to define access control needs of XML trees. Our aim is to enforce the same authorization permissions for syntactic variations of the same XML document and to incorporate data and application specific semantics in XML access control. Our approach is based on defining access control on ontologies data and application semantics. Mappings from the XML trees to these ontologies enable to enforce access control on the syntactic representation. Our current approach aims to be built on top of XML access control mechanisms and to support presentation independence of XML developers.

Our future work includes formal definition of semantically equivalent XML documents based on specific applications, enhanced conflict resolution, policy verification, and addressing ontological inferences.

References

[1] Y. An, A. Borgida, and J. Mylopoulos. Constructing complex semantic mappings between xml data and ontologies. In *International Semantic Web Conference*, pages 6–20, 2005.

[2] E. Bertino, S. Castano, E. Ferrari, and M. Mesiti. Specifying and enforcing access control policies for xml document sources. *World Wide Web*, 3(3):139–151, May 2000.

[3] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl, March 2001.

[4] L. Clement, A. Hately, C. von Riegen, and T. Rogers. Universal description, discovery and integration (UDDI) v3.0. http://uddi.org/pubs/uddi-v3.0.2-20041019.htm, October 2004.

[5] E. Damiani, S. D. C. D. Vimercati, S. Paraboschi, and P. Samarati. A fine-grained access control system for xml documents. *ACM Transactions on Information and System Security TISSEC*, 5(2):139–151, 2002.

[6] G. Della-Libera, M. Gudgin, P. Hallam-Baker, M. Hondo, H. Granqvist, C. Kaler, H. Maruyama, M. McIntosh, A. Nadalin, N. Nagaratnam, R. Philpott, H. Prafullchandra, J. Shewchuk, D. Walter, and R. Zolfonoon. *Web Services Security Policy Language V1.1 specification* , July 2005. ftp://www6.software.ibm.com/software/developer/library/ws-secpol.pdf.

[7] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, and H. F. Nielsen. SOAP version 1.2 part 0: Primer. http://www.w3.org/TR/soap12-part1/, June 2003.

[8] N. Kodali, C. Farkas, and D. Wijesekera. Enforcing semantics aware security in multimedia surveillance. *To appear in the Journal on Data Semantics (Springer LNCS)(Invited)*, September 2004.

[9] N. Kodali, C. Farkas, and D. Wijesekera. Metadata for multimedia access control. *To appear in the International Journal of Computer Systems, Science and Engineering special issue on trends in XML Technology. (Invited)*, 2004.

[10] N. Kodali, C. Farkas, and D. Wijesekera. Rdf meta-structures for multimedia security. *To appear in the Journal of Digital Libraries*, 2004.

[11] N. B. Kodali, C. Farkas, and D. Wijesekera. Mls-smil for surveillance of facilities with multi-level security requirements. In *Proceedings of the IFIP 11.5 conference on Security and Integrity-2003*, pages 67–83, 2003.

[12] N. B. Kodali, C. Farkas, and D. Wijesekera. A rdf meta-structure for secure multimedia. In *In Proc. Workshop on Metadata for Security, in connection to ODBASE*, 2003.

[13] M. Kudo and S. Hada. Xml document security based on provisional authorization. In *Proceedings of the 7th ACM Conference on Computer and Communications Security, Athens, Greece*, pages 87–96, November 2000.

[14] Microsoft Corporation. *Web Service Enhancements (WSE 3.0)*, 2005. http://msdn.microsoft.com/webservices/webservices/building/wse/default.aspx.

[15] A. Nadalin, C. Kaler, H.-B. Phillip, and R. Monzillo. *Web Services Security: SOAP Message Security 1.0 (WS-Security 2004) OASIS Standard200401*. OASIS, March 2004. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.

[16] U. Ogbuji. When to use elements versus attributes. http://www-106.ibm.com/developerworks/xml/library/x-eleatt.html, March 2004.

[17] L. Qin and V. Atluri. Concept-level access control for the semantic web. In *Proceedings of the 2003 ACM workshop on XML security*, pages 94–103. ACM Press, 2003.

[18] A. Stoica and C. Farkas. Secure xml views. In *16th IFIP WG11.3 Working Conference on Database and Application Security*, pages 133–146, 2002.

Authorization Strategies for Virtualized Environments in Grid Computing Systems [*]

Xinming Ou   Anna Squicciarini   Sebastien Goasguen   Elisa Bertino
Purdue University

Abstract

The development of adequate security solutions, and in particular of authentication and authorization techniques, for grid computing systems is a challenging task. Recent trends of service oriented architectures (SOA), where users access grids through a science gateway — a web service that serves as a portal between users of a virtual organizations (VO) and the various computation resources, further complicate the authorization problem. Currently, the security component developed as part of the Globus Toolkit, the de facto standard for grid infrastructures, is not fully equiped with the capabilities to meet those challenges. The main drawback of the current approach is that it relies on a low level identity-based authorization scheme. A low-level access control policy maps a user's identity (distinguished name) to a local account. This approach does not scale and is hard to manage in a distributed environment. The goal of this paper is to make a first step towards new authorization solutions that better fit novel grid infrastructures characterized by virtual organizations and science gateways. We review and analyze several solutions proposed, in particular GridShib and the VO Privilege Project, as they represent the most innovative techniques currently under development to achieve attribute-based authorization. We then propose several solutions for grid authorization through science gateways, and discuss how those existing projects can be leveraged to implement the solutions we propose.

1 Introduction

Grid computing [15, 16] represents an important infrastructure that makes it possible for multiple institutions to pool their computing resources and to collaborate in order to solve computationally intensive problems. As more organizations and users are becoming interested in using grid computing systems in a variety of application domains, security becomes a key issue. Designing a scalable authorization strategy for such a context is a complex problem. Although authorization in distributed systems has been extensively investigated, not much work has been done to address authorization problems facing real large distributed systems such as grids. The current de facto solution, as represented by the GSI component [7] included in the Globus toolkit [2], adopts a simple low-level approach based on user identities. An access control list (grid-mapfile) that maps a user's global identity (distinguished name, or DN) to a local account has to be set up at *every* grid site. Users whose DN appears on the list is authorized to use machines in the site, with privileges associated with the local account. This simple approach is in essence the same authorization mechanism used for a single machine, for example the Unix "/etc/passwd" file. In a distributed system like a grid, there are thousands of users and it is not realistic to base authorization decisions on individual users' identities. Not only the size of the grid-mapfile will far exceed human managibility, but also

[*]This work was supported by the National Science Foundation through nanoHUB NMI integration and deployment grant SCI-0438246 and through the TeraGrid Resource Provider grant SCI-0503992.

it unnecessarily replicate many information across a wide domain. In reality, access decisions in large systems are typically based not on user's unique identity, but on his *attributes*, such as being a member of an insitution or involved in a particular computation project. Thus, it has been widely acknowledged in the grid-computing community that attribute-based authorization should be the direction of development for grid security. A number of projects are going on in this direction, such as the VO Privilege Project [10], GridShib [4], and PERMIS [13]. However, there are quite a few decision dimensions when it comes about designing an attribute-based authorization scheme for grid computing. Recent years have seen grid computing moving towards a more "virtualized" environments. The usage of computational resources is less divided by institutional boundaries, and virtual communities are formed that often include members from multiple organizations. It is also becoming more popular to access grid computing resources through a web-service based front-end, sometimes called science gateways. In this paper we would like to explore design options for attribute-based authorization in grid that will better suit the need in such virtualized environments.

The rest of the paper is organized as follows. Section 2 provides an overview of the main requirements grid systems currently need to face due to the emerging trend of virtualized environment. We then focus on the authorization requirement, and present in Section 3 the existing approaches and main challenges. In Section 4 we propose three possible authorization operation modes for accessing grid through science gateways. In Section 5 we elaborate on the case study of the NanoHub science gateway. We conclude the paper in Section 6.

2 Virtual organizations and service oriented architectures

A virtual organization (VO) is a community of individuals that transcends traditional organizational boundaries. Many computational projects conducted in the grid infrastructures span multiple institutions. It is not always feasible to designate any one of the physical institutions as having the administrative rights over the VO; or the members of the VO do not want to bother with the physical organization's administration to help them maintain the VO membership information. The implication of this on grid security is that information needed for authorization decisions may not always be available within the administration of a particular physical organization. For example, in typical settings, to determine whether a user has certain rights to use some resource, the resource provider can query a user database maintained by the user's home institution (e.g. an LDAP server) to retrieve the user's relevant information. However, if VOs are involved, it is not always clear where to retrieve relevant user information for authorization purposes. The authorization policy thus must be flexible enough to specify different trust relationships: for certain kinds of user information the resource provider trusts the user's home institution; but for user's membership regarding a VO, the resource provider may trust a database maintained by the VO's members.

In parallel, recent years have seen trends of grid computing moving towards a *service oriented architecture* (SOA), where a user does not directly interact with grid infrastructures but rather access the grid through a *service provider*. There are several reasons why this architecture has become popular, the most important one being that the service provider can maintain a collection of applications of particular interest to a user community. The users of the service provider can directly use those applications without having to obtain the application code and upload them to the grid sites. We distinguish service providers and resource providers even though it is understood that resource providers who actually operate and maintain the hardware resources necessary to execute services could also be service providers. We intentionally separated the SP and RP to tackle the case when a scientific community may build its own infrastructure and offer tailored services to its user community. These services in turn may use other services offered by the RP such as job execution, file transfer and so on. Indeed, there is a trend to move service offerings one level up in the hands of the actual scientific communities who know best what their needs are and how they want to interact with their services to enable more science. This trend is illustrated by the TeraGrid science gateway program [9] whose aim is to outreach to communities as a whole and build generic access to

TeraGrid resources so that a large number of scientists in the nation may benefit from the TeraGrid resources.

In some sense the challenges caused by the addition of VO and science gateways to grid security are related. Both require the authorization system to handle a large group of users. This requires the back-end grid infrastructures to have the ability to authorize a community of users in an effective and scalable way. However, the current authorization mechanisms in place in grids do not properly address this requirement yet, as we illustrate in the next section.

3 Authorization in grid

Grid is inherently a federated environment, where every local site wants to retain its authority on determining who can use its resource. In Globus toolkit's GSI component, authorization is done by consulting an access control list called "grid-mapfile". The file contains mapping from a DN, a globally unique name assigned to a grid user, to a local account. Only DN's that has an entry on a grid site's grid-mapfile can use the site's computational resource. Once the user is authenticated, it is assigned to a local account according to the grid mapfile and runs its job as the local account. Further access control can be done through that account in the local system.

Grid authorization based on grid-mapfile is not scalable, because it requires the resource provider to maintain authorization state for every potential user. In the emergence of virtual organizations and science gateways, this is hard to do not only because the number of users associated with a VO or service provider may be huge, but also because the membership of users in a VO or service provider may change dynamically and it is not realistic to require the resource provider to keep track of this membership change. Currently, a number of projects are going on to address this problem [4, 10] and they all adopt *attribute-based authorization*.

3.1 Attribute-based authorization

In real life, authorization decisions are typically based on a user's role in an organization, rather than his unique identity. For example, a policy regarding the usage of a computer may contain a statement like "every faculty member of the university can use the machine", instead of listing the names of every faculty member. "Faculty member" is an attribute associated with a user describing the user's role in an organization. Policies written in terms of attributes rather than individual user's identity more accurately capture the high-level access control intention. For example, when John Smith is no longer a faculty member, the policy does not need to be modified to revoke his privileges. Similarly, when a new faculty joins, the policy does not need to change to allow the new faculty to use the resource. Certainly, an authority needs to provide the attributes for every user that may use the resource. Such attributes should be provided by an attribute server trusted by the resource provider (for example, a server maintained by the university's HR department can provide attributes regarding faculty membership). There are a number of dimensions to choose when building an attribute-based authorization system. Different options result in different attribute collection process.

push mode vs. pull mode. Attribute-based authorization can be implemented according to two different strategies: *push* and *pull*. The push strategy requires the user to contact an attribute authority service to obtain attribute certificates and "push" them to the target service when submitting a request. This approach allows the user to select the specific roles he wants to be authorized with. The pull strategy, on the other hand, does not require the users to present any attribute. The attributes are directly retrieved by the resource provider on behalf of the user. The pull strategy makes attribute retrieval transparent to users.

IdP for institution vs. IdP for VO In an attribute-based authorization system, an attribute server is also referred to as an IdP (Identity Provider). Some IdP's are associated with a physical organization such as a university. IdP's may also be associated with a VO. An important question raised in this context is how a user's attributes should be

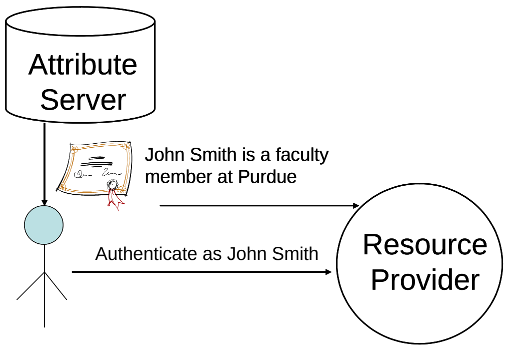

Attribute Server
John Smith is a faculty member at Purdue
Authenticate as John Smith
Resource Provider

Figure 1. Push mode

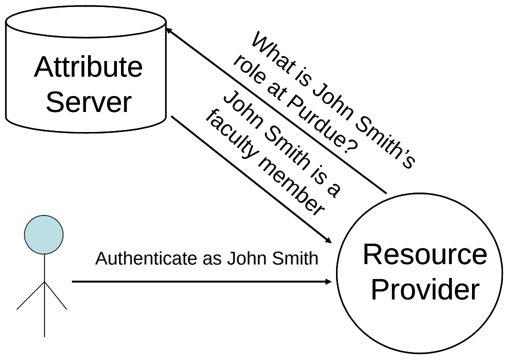

Attribute Server
What is John Smith's role at Purdue?
John Smith is a faculty member
Authenticate as John Smith
Resource Provider

Figure 2. Pull mode

maintained given the diversity of attributes. A user's home institution may seem to be the only entity entitled to provide a user's attributes. However, when VO's are involved, it is often the case that a user's attributes may not be related to his home institution at all. Thus, it is necessary to support flexible attribute maintenance schemes. The resource provider must be able to specify which IdP it trusts for certain kinds of attributes. Requiring the local site to subscribe to a fixed trust relationship in regard to attribute retrieval, (for example by only allowing a user's home institution to provide his attributes) is not likely to be adequate with the emergence of the virtualized environment in grid computing.

3.2 Enforcement of policies

In an identity-based authorization system, an access control policy is typically a list consisting of entries of (subject, object, operation), where "subject" is the user's identity and "object" represents the requested resource. For example, a grid-mapfile is such an access control list, where subject is a user's DN, object is a local account, and the implicit operation is "submit a job on behalf of the local account". In an attribute-based authorization system, user's attributes, instead of an id, are used in the "subject" field. The resource provider's policy must also include statements on who has the authority to assert a user's attributes. This is called the *trust management policy*. This is useful especially in environments in which the same attributes could be provided by different authorities. It is the resource provider's responsibility to decide who to trust for specific attributes. Without a formal trust management policy, such trust will have to be hardcoded in the implementation in an ad hoc way, which may result in security breaches.

Once the policy is defined, a *policy decision procedure* on the resource provider side consumes users' attributes from appropriate authorities and evaluate the policy. Since attribute assertions are from different sources, it is important that these assertions be transformed into a uniform format that has well defined semantics, for example in the XCAML [17] language. The policy decision point can then evaluate the policy with the retrieved attributes and render a decision. An example of authorization mechanism evaluating such policies is the Shibboleth [8] project, which is explained in the following section.

3.3 Current projects investigating attribute-based grid authorization

A number of projects are on going to implement different attribute-based authorization schemes for grids. We now briefly describe two of them: the VO Privilege Project [10] and GridShib [4].

- *Privilege Project.*

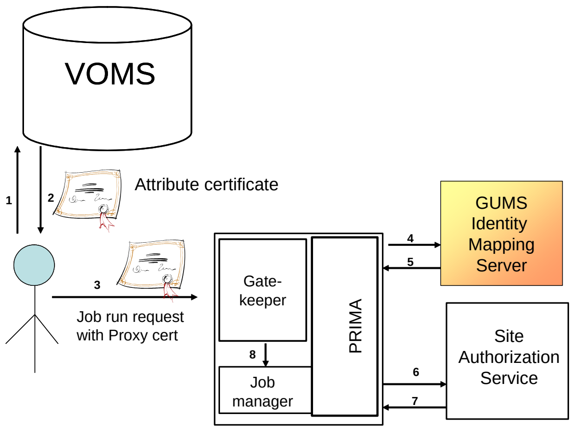


Figure 3. VO Privilege Project

The VO Privilege Project [10] has been developed by US CMS and US ATLAB in order to implement fine grained authorization for access to grid-enabled resources and services. The implemented access control mechanism is illustrated in Figure 3[1]. As shown, the VO Privilege Project implements a push mode mechanism. VOMS is an IdP that is associated with a VO. A user, before accessing the grid service, first obtains an attribute certificate from VOMS and embeds it into his grid proxy certificate[2]. The user then presents the proxy certificate to the site when submitting a job or initiating a file transfer. Operatively, when the proxy is forwarded to a gatekeeper, instead of consulting the grid mapfile, the gatekeeper contacts the site-wide GUMS [5] service to map the request to a local account. The PRIMA module retrieves the attribute information from the user's proxy certificate and communicates it to GUMS, and GUMS uses the attribute information to decide whether the request is authorized and if so which local account to use. As a last step, the gatekeeper contacts the Site Authorization Service to enforce other site-specific access control policies.

- *GridShib.*

  GridShib [4, 19, 11] is an example of a pull mode authorization system. The idea in GridShib is to develop an authorization system for grids by integrating the technologies in Shibboleth [8]. Shibboleth is an infrastructure for cross-identity authentication, and it exploits the concept of federated identity information to federated user attributes. In Shibboleth, when a user at one institution tries to use a resource at another, Shibboleth sends attributes about the user to the remote institution, rather than making the user log in to that

[1]This is a simplified version of the diagram at http://computing.fnal.gov/docs/products/voprivilege/
[2]A proxy certificate is used to authenticate a user with a grid site and is valid for a relatively short period of time.

institution. The receiver can check whether the attributes satisfy its access control policy. The IdP in the Shibboleth architecture has all the user attributes and user privacy preferences which are taken into account when this IdP gives information to a service provider.

In its current version, GridShib is implemented as a plug-in for Globus Toolkit 4.0 (GT4). The plug-in implements a policy decision point based on attributes obtained from a Shibboleth attribute authority. In a nutshell, the underlying idea in GridShib is to authenticate grid users using the existing GSI module in Globus, determining the address of the Shibboleth attribute server in the process, and then obtain from the Shibboleth service the selected user attributes that the Grid service is authorized to access. Differently from other approaches, in GridShib the clients of services are not directly affected and do not even need to know Shibboleth is involved in their decision-making. Unlike the full-fledged Shibboleth deployment, which adopts handle-based authentication, GridShib uses X.500 distinguished names to identify principals, thus the current PKI infrastructure in the grids does not need to be touched. The service provider receives a proxy certificate in place of the handle typically issued by a Shibboleth IdP. Like Shibboleth, GridShib's plan is to associate an IdP with a user's physical institution. This scheme may be too restrictive as users will have multiple affiliations and some of those affiliations will be virtual organizations. An approach that supports flexible IdP association is more desirable.

As illustrated, the goals of the two projects are very similar as they both aim to develop a flexible and efficient attribute-based authorization mechanism in grid systems. The approach to the problem is different mainly with respect to the strategies used to collect user attributes and whether the attribute server is associated with a VO or a user's physical institution, two design dimensions mentioned in Section 3.1. In both cases user authentication is based on X.509 [18] certificates. The possibility of a user to push his attributes as in the Privilege Project, let him select the role he wants to use for executing the request, though it requires additional operations at the user's side. The pull mode adopted by GridShib is however easier to deploy, since clients of services are not affected and can submit jobs regardless the underlying authorization system used. Currently it is not possible to determine which of them will better address the access control requirement we have outlined, as both projects are still work in progress. In particular, GridShib is at an infant stage and it still needs to be tested on real case scenarios. GridShib in its final version will include both push and pull mode.

3.4 Attribute-based authorization in a virtualized environment

The attribute-based authorization discussed in this section is better suited to provide security solutions for the virtualized environment described in Section 2. For example, the VO Privilege Project is particular aimed at managing authorization at the VO level. GridShib, by incorporating Shibboleth authorization mechanism which has been widely deployed for web-based access control, is well positioned to meet the security problems brought about by the science gateways. In the next section, we propose several scenarios for enabling attribute-based authorization for accessing grid through science gateways, and discuss how the existing projects may be leveraged to implement them.

4 Proposed Grid authorization solutions for science gateways

Currently, command line access is the most common operation mode for grid applications. A user first authenticates himself to a grid site using his X.509 certificate. The site then makes its access control decisions either based on the grid-mapfile or the user's attributes and its local policy. When accessing grids through a science gateway, a user does not necessarily authenticates himself directly to the grid site. Often times the authentication is between a user and a service provider, which uses one or more existing grid infrastructures as its computational backend. This architecture poses a challenge to authorization: without the user's identity, how can the grid site know who is using its resources and whether to grant access or not? In this section we propose three possible authorization modes for this service-oriented architecture. The different scenarios are all characterized by three main entities: a user, a service provider (the science gateway), and a resource provider (the grid site). Since the user does not interact directly with the resource provider (RP hereafter), the RP will have to rely on the service provider (SP hereafter) to perform some of the authorization tasks. The difference among the three scenarios largely arise from the different levels of trust between SP and RP.

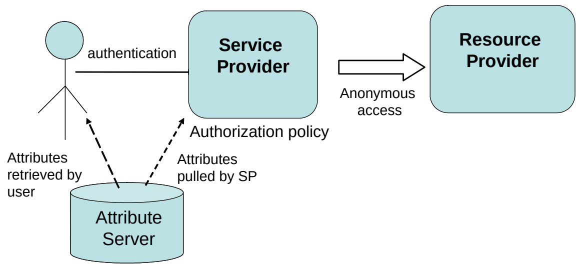


Figure 4. Complete trust

- **Complete Trust.** In this case, the SP is trusted completely by the RP. The analyzed scenario is sketched in Figure 4. The SP enforces both authorization and authentication, while the RP is not involved at all in the process. The user is essentially anonymous at the RP side — the only information the RP gathers from the request is related to the specific job to execute. The SP negotiates resource allocations with the grid site and its users do not need to be aware that they are using the grid as the computational backend. Ideally, authorization at SP should be based on users' attributes. Logically the attribute server and the SP are two different entities, but physically they could either be at the same site or belong to different organizations. Once authentication and authorization is completed, the job request is then forwarded to the RP. The request is typically forwarded as one single grid DN, and from the resource provider's point of view it is not possible to identify who the real user is. Actually many users of the SP may not even have a grid account. They may not even be aware that the SP is using one of the grids to perform computation. The advantage of this approach is that a user does not need to go through the lengthy application process to obtain a grid account allocation while still able to enjoy the computational power provided by the distributed infrastructure. The disadvantage is that the grid site cannot control its own access policy (e.g. differentiated service levels) based on users' credentials. For example, the resource provider cannot provide priority scheduling for members involved in a particular project if the job request is issued through the service provider, since such job requests will be uniformly mingled with all the other requests from the same SP and the RP has no way to know which request is from the particular group of users. In order to provide such priority scheduling, the RP will have to instruct SP to differentiate its service levels for that particular group.

- **Medium trust.** In this case, the SP still performs authentication, but the final authorization decision is made by the RP. A sketch of the process is given in Figure 5. The identity token passed from SP to RP does not necessarily contain the user's unique identity, but rather a user's attributes or a handle that can be used later to retrieve a user's attributes. These attributes should be provided either by the SP or another trusted third party. For example, a SP may provide attributes reflecting a user's paid premium and request the RP to schedule jobs on different priorities based on the premiums. Or the SP can provide attributes reflecting a user's trustworthiness — an anonymous user is less trustworthy than a registered user. The RP can then apply different access control policies based on a user's trustworthiness. The advantage of this approach, compared with the complete-trust model, is that the RP can have its own policy to differentiate its service based on users' attributes and can have better control on how to allocate its computation resource. Note also

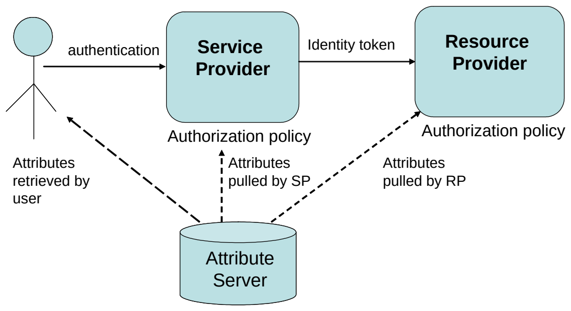


Figure 5. Medium trust

that under this mode, the SP can also have its own authorization policy. A request may already be filtered out by the SP's policy before reaching the RP. The SP can also depend its authorization policy on users' attributes and retrieve attributes from trusted IdP's.

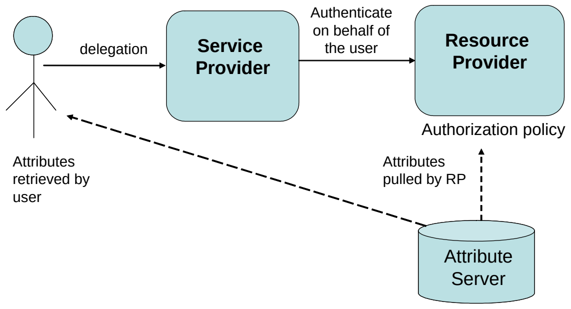


Figure 6. No trust

- **No trust.** In this scenario both authentication and authorization are executed by the RP. A user who already has a resource allocation at RP can access the grid through the SP using his own grid credentials. The advantage of this approach is that the RP can independently track individual users using their resources. In the previous two cases the RP needs the coordination and information from the SP to trace back who is behind a job running on its machines. The disadvantage is that only users with allocation on the RP can use the service provided by the SP, which may be too inflexible in practice. To implement this approach, a mechanism to securely delegate a user's grid authentication credential to the SP is necessary. Users are in this case aware that they are using certain grid resources and the service provider forwards job requests to the proper resource provider.

4.1 Implementation schemes

All the three cases discussed above adopt an attribute-based authorization scheme. Since the VO Privilege Project and GridShib are two projects being actively developed for enabling attribute-based authorization in grid, we would like to explore ways that one can use them for science gateways.

In case 1 the SP replaces the resource provider for the role of performing authentication and authorization. If the SP uses GSI for this purpose, then both the VO Privilege Project and GridShib can be directly used here. However, it may not be the case that the SP adopts GSI as its security mechanism (e.g. the NanoHUB example that will be introduced in Section 5). In this case the SP can still implement its own attribute-based authorization but will not be able to directly leverage the techniques developed by these projects.

Both the push mode and the pull mode could be used in attribute retrieval, and Shibboleth can play a role. For example, the SP may rely on a third-party Shibboleth IdP to provide the identity service and base its authorization decisions on the attributes retrieved from the IdP.

Case 2 has the most flexibility in terms of how to leverage the existing projects for implementation. One natural solution is to use GridShib and here again we have a number of possibilities. One is that the service provider is also an IdP. The SP can serve as a weak CA — a certificate authority entitled to issue short-term user credentials for authentication purposes. The DN's in such certificate could be meaningless, but they allow the resource provider to recognize different users. This kind of short-term certificates are sometimes called "junk certificate". A junk DN certificate can then serve as the identity token in Figure 5. A Shibboleth IdP will be running at SP site and the RP invokes the GridShib plug-in to retrieve attributes by presenting the junk DN to the IdP at SP site. Since the junk DN is issued by the SP's CA, the SP can link it to one of its users and retrieve his attributes. The other possible way of applying GridShib is that the SP relies on another IdP to provide authentication and attribute service. One of the developing projects in the GridShib use cases has this flavor [3]. On the other hand, the VO Privilege Project could also be applied to implement case 2. The service provider can obtain an attribute certificate from VOMS on behalf of the user and forward it to the RP. Currently the VOMS service uses a DN to identify a user, which means the user needs to establish a globally meaningful grid DN. This may be less flexible in terms of usability, but it allows greater flexibility in terms of attribute retrieval. The attribute servers do not need to be tightly associated to the authentication service (for example, the service provider), as long as both recognize the same set of DN's.

For case 3, the only difference from the current standard grid access method is that the service provider acts on behalf of a user. Both GridShib and VO Privilege Project can be readily applied at the RP side. The only additional requirement is to put a credential delegation service (such as MyProxy [12]) on the SP.

4.2 Grid interoperability

As more grid infrastructures emerge, there is an increasing need for different grids to interoperate seamlessly. Currently, user credentials for one grid cannot be directly used to access resources at another. A user will need to maintain multiple credentials if he wants to use multiple grids.

The emerging trends of virtual organizations and science gateways provide an opportunity for more seamless grid interoperability. A VO or science gateway can negotiate resources with different grids and users can be saved the burden of requiring grid credentials by themselves. To achieve this in practice requires the attribute retrieval protocols be standardised among the various grids. For exmaple, the VO Privilege Project uses ASN.1 format to embed an attribute certificate in a user's X.509 certificate, wheareas GridShib uses SAML assertions to retrieve user attributes. This means a grid infrastructure that adopts one of these technologies will not be able to understand attribute assertions from another grid that uses the other technology. A single standard format for expressing attributes and a single standard protocol for retrieving them must be defined to achieve grid interoperability.

5 Case Study: NanoHUB

One example that demonstrates the move towards service oriented architectures in grid computing is the nanoHUB project [6], which is becoming the de facto cyberinfrastructure for the computational nanotechnology community with 1,600 annual users accessing the nanoHUB simulation service and a total of 9,000 users accessing other services such as collaboration tools and educational modules. NanoHUB is a TeraGrid science gateway and is also an Open Science Grid VO.

It is necessary to study the authentication and authorization solutions for the nanoHUB cyberinfrastructure and how it will interoperate with the TeraGrid and OSG resources and policies. The nanoHUB infrastructure is based on the InVIGO middleware [14] and has integrated the use of Condor-C and Condor-G [1] to submit jobs to grid resources. InVIGO's innovation lies in the use of virtualization techniques. The nanoHUB makes use of virtual machines based on Xen 3.0 and VMware. Virtual machines can provide execution jail but also offer a level of isolation from the physical infrastructure. Virtualization is the solution to build a cyberinfrastructure for a community on top of one organization administrative domain. VMs together with virtual networking and virtual file system are the basis of a virtual environment where users of a community can be isolated. At a first order the nanoHUB is only computational oriented and does not face data transfers, replication and publication issues. The default nanoHUB user is loosely known with only a valid email address to identify them and no X.509 certificate therefore it differs from the model of standard TeraGrid users and OSG VO members. TeraGrid users have received an allocation on the resource providers and can use GSI authentication to access the resources, authorization is then done thorugh grid mapfile and local unix accounts. This approach won't scale for the science gateways as the number of users will be several thousands like in the nanoHUB case. It is unrealistic not only to maintain a grid mapfile with thousands of entries but it is also unrealistic to have thousands of grid accounts created for the nanoHUB: one per user. A more realistic approach is to use a VO account like in the OSG model but that approach breaks because it still requires every user who will access the resource to have its DN mapped to the VO account on the remote resource. Additionally the nanoHUB users are not issued a trusted X.509 certificate nor does the nanoHUB operates its own CA and vouches on the identity of its users. When a user accesses a nanoHUB service he is being authenticated on the nanoHUB portal through standard login/password mechanism, then his roles are looked up in an LDAP database which will give or deny access to simulation tools.

When a job is being run on the local virtual machines, it runs as the user using standard UNIX AA techniques through the PAM module. If a grid job submission is needed, then Condor-C is being used to forward the job to a machine on which users cannot login and where a nanoHUB proxy certificate is created to talk to RPs on TeraGrid or OSG using Condor-G. Condor-C is used to map job requests from nanoHUB users to a trusted certificate on remote grid resources. This nanoHUB certificate has been issued by a TG-trusted CA and is also trusted by OSG. The difference stems from the fact that no individual users DN will be mapped to this service account on the remote sites. The identity of the user submitting the job is only known from the nanoHUB, i.e the SP. Additionally the user does not know where the job is running either. This is essentially the complete-trust scenario described in section 4. However some nanoHUB users may already have valid certificates that they may want to use to run nanoHUB services on TG or OSG. These users may actually have their own allocation on TeraGrid and may decide to use nanoHUB services on their allocation. This is the situation discussed in the third scenario of Section 4. Managing all these different use cases and authorization criteria could become quite complex. Therefore we see that the nanoHUB and most probably more Virtual community cyberinfrastructures are in the need for an authentication and authorization model that offers hybrid models and delegated trust models.

Future efforts in the nanoHUB will focus on deployment of an authentication/authorization model. The model will be a hybrid of the three use cases described in section 4. Default users will be given limited privileges, i.e limited access to services. In the short term nanoHUB may maintain its own attribute server while in the long term it may also leverage other Shibboleth deployment at U.S. institutions relying on a distributed authentication system to grant access to nanoHUB services. Currently we are planning to base our implementation on the GridShib technology. But other approaches such as VO Privilege Project are also alternatives.

6 Conclusion

In this paper we discussed the need for attribute-based authorization technology in grid computing to acco-modate the emerging trend of virtualized environment, where users access grid resources as a virtual community through a service provider. We presented several design dimensions in building an attribute-based authorization system for grids and discussed their perspective advantages and disadvantages with regard to this trend. We proposed several solutions for attribute-based authorization for accessing grids through science gateways, and discussed how existing/onging grid authorization projects in this area could be leveraged to build such systems. As a case study, we described nanoHUB, a science gateway for the computational nanotechnology community, the authorization challenges faced by it, and how the framework outlined in this paper can solve the problems.

References

[1] Condor, High Throughpout Computing. http://www.cs.wisc.edu/condor.

[2] Globus Toolkit. http://www.Globus.org.

[3] Grid portal use case for gridshib. https://authdev.it.ohio-state.edu/twiki/bin/view/GridShib/GridPortalUser.

[4] The Gridshib project. http://gridshib.globus.org/.

[5] The Gridshib project. http://grid.racf.bnl.gov/GUMS/.

[6] NanoHUB. http://www.nanohub.org.

[7] Overview of the grid security infrastructure. http://www.globus.org/security/overview.html.

[8] The Shibboleth. http://shibboleth.internet2.edu/.

[9] The Teragrid project. http://www.teragrid.org.

[10] The VO Privilege Project. http://computing.fnal.gov/docs/products/voprivilege/.

[11] T. Barton, J. Basney, T. Freeman, T. Scavo, F. Siebenlist, V. Welch, R. Ananthakrishnan, B. Baker, and K. Keahey. Identity federation and attribute-based authorization through the globus toolkit, shibboleth, gridshib, and myproxy. In *5th Annual PKI R&D Workshop*, October 2005.

[12] J. Basney, M. Humphrey, and V. Welch. The MyProxy Online Credential Repository. In *Software: Practice and Experience*, volume 35(8), 2005.

[13] D. Chadwick. Authorisation in Grid Computing. *Information Security Technical Report*, 10(1):33–40, unknown 2005.

[14] J. Fortes, R. Figueiredo, and M. Lundstrom. Virtual computing infrastructures for nanoelectronics simulation. *Proceedings of the IEEE*, 93(10), August 2005.

[15] I. Foster and C. Kesselman, editors. *The grid: blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[16] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150:1–??, 2001.

[17] OASIS. Xacml 2.0 approved as oasis standard.

[18] W. F. R. Housley, W. Polk and D. Solo. Internet X.509 Public Key Certificate and certificate revocation list (CRL). RFC, 3280, network working group, April 2002.

[19] V. Welch, T. Barton, K. Keahey, and F. Siebenlist. Attributes, anonymity, and access: Shibboleth and globus integration to facilitate grid collaboration. In *4th Annual PKI R&D Workshop*, April 2005.

Security Mechanisms for Data Intensive Systems

Periorellis P., Wu J., and Watson P.

panayiotis.periorellis@ncl.ac.uk

University of Newcastle Upon Tyne
School of Computer Science, e-Science Centre
Devonshire Building, NE1 7RU, Newcastle, UK

January 2006

Abstract. *GOLD (Grid-based Information Models to Support the Rapid Innovation of New High Value-Added Chemicals) is concerned with the dynamic formation and management of virtual organizations in order to exploit market opportunities. The project aims to deliver the enabling technology to support the creation, operation and successful dissolution of such virtual organizations. A set of middleware technologies has been designed and being implemented to address issues such as trust, security, contract management and monitoring, information management, etc. for virtual collaboration between companies. In this paper we discuss some of the more general requirements for authentication and authorization aspects in GOLD virtual organizations and we use these requirements as benchmark to assess some of the more popular tools that are currently available in dealing with these issues, together with our own approach that addresses these problems.*

1 Introduction

Authentication and authorization mechanisms are integral to the operation of any virtual organization. The GOLD project spent a considerable amount of time and effort gathering a full set of requirements [Periorellis P. et al 2004] that address the needs of virtual organizations in terms of such mechanisms. We have identified a need for flexible, interoperable solutions that are capable of dealing with the cross-organisational nature of VOs as well as its dynamics in terms of composition. The paper is structured as follows. In the next section we discuss general requirements for authentication and authorization in VOs. We move on to present the mechanisms we have developed for authorization and authentication based on web service standards and specifications.

2 Requirements

Virtual organizations bring together a number of autonomic organizations to assess a market opportunity. The duration of this collaboration can be brief or require a larger life span. It is certain that during the lifetime of a VO, the parties that form it will be required to share resources. Hence access of those resources will require crossing of organizational boundaries. Participants of a VO are expected to have implemented their own security mechanisms to protect resources within their boundaries by some authentication mechanisms. A participant may require access to several resources scattered across several organizational boundaries. Hence the problem that is created is that of multiple logins. Obviously expecting a party to login several times in order to carry out a task that is part of the same activity is not productive. The way we solve this problem is by providing 3rd trusted parties that provide security assertions as and when needed. These 3rd trusted parties authenticate VO participants and whenever a party requires remote access they vouch for that party's identity usually by sending a signed credential token. Examples of these are the Microsoft passport which offers a central ID provision and management.

Another solution is that of federation, which is supported by the Liberty Alliance Group. In this case parties agree on pre-determined trust relations between services providers and identity providers. Blum D. [2005] defines federation as a community of organizations or domains that use collective terms describing agreements, standards and technologies that make identity portable across autonomous domains. Federation although offers a conceptual solution to the issue of multiple logins, is not always possible to adopt. The main problem that arises within a commercial context is that of parties disagreeing on the authorities they trust. Federation requires that assertions made by entities within a trust circle are accepted, i.e. they should be trusted. In the domain that we have been looking at [Conlin et al. 2005] this is not always the case. We therefore need to allow parties that form the VO to have the option of validating an assertion by entities that may not directly be linked with a VO. As such we cannot impose authorities that everyone within a VO must trust. Instead we need to provide the underlying protocols that allow participants of a VO to validate security related assertions by authorities which individual entities trust.

Regarding authorization, given the dynamicity of VOs, both the topology of the VO (parties that form it and links between them) and its workflow are subject to change. Therefore static rights assignment prior to a workflow does not capture the eventuality of a party leaving the VO or being added. In addition given the sensitivity of the information that may be shared in some VOs (which raises concerns regarding competitive advantage) parties are not expected to be handed permission that last throughout the duration of a VO. It is more likely that companies will agree limited access or gradual access to their resources depending on progress. In GOLD we want to be able to restrict role access on resources depending on the execution context. In a virtual organization the execution context can be regarded as the particular project or the goal that all participants have come together to achieve. In a virtual organization there can be many projects with various participants resulting to complicated interrelationships between them. For example, some of them may play the same role in various projects, carrying out the exact same tasks, or have different roles within the same project depending on the performed task. The question we raise is 'Should a role have the same permission and access rights throughout the set of similar or even identical projects and the tasks within those projects?' Our view is that in dynamic access control systems we should separate roles from role instances. Different role instances may require different permissions and indeed additional levels of authorization depending on the project and task in which they are active.

The main GOLD case study involves the collaboration of a number of companies to enable the development of chemicals. The domain itself requires the sharing of sensitive information between participants who may have conflicting interests. In order to raise the levels of trust within such a VO we need to make sure that we have developed fine grained access control mechanisms. RBAC or other traditional techniques do not provide this level of granularity. The main issues regarding access control relate to the degree of granularity embedded in the controlling mechanism itself. By granularity we refer to the level of detail for which we are prepared to define access rights. In GOLD, the simple subject-object permissions model on which RBAC is based is not sufficient. We need fine grained permissions for instances of roles as well as instances of objects. For example, a chemist role may be granted access to chemical documents but we do not however wish to grant access to all chemical documents produced by the system. Instead, we want any access permissions granted to the chemist role to be project-specific (e.g., the instance of a particular collaboration) as well as task-specific (e.g., the instance of a particular pre-defined activity). The management of roles and access permissions in GOLD is integrated with the management and monitoring of dynamic service level agreement or contracts between the participating services. The contracts can capture the expectations from specific tasks, using pre- and post-conditions. Permissions for roles can be activated and deactivated based on the progress of the monitored contracts.

Given the plethora of policies for access control within a VO in conjunction with the fact that there is not a single authority that governs these policies (the majority will stem from participants' requirements on how they want to protect their resource) validation will be needed to make sure that there are no logical inconsistencies prior to the workflow enactment. Finally policies would also need to be verified against industrial regulations that govern a particular sector such that of chemical development. Therefore policies should comply with the higher more abstract industry policies on how chemical development should be conducted.

We need an authorization-authentication framework that allows single sign on to be achieved by various methods without imposing any decisions regarding who the trusted parties should be to the VO participants. It should support federations but at the same time it should provide flexible protocols that allow participants to validate security assertions using authorities that those parties trust.The next paragraph describes the mechanisms developed by GOLD to deal with both of these issues.

3 Authentication

Authentication describes the process of securely establishing and verifying identities of a network subject which may take the form of users, agents and registered service components. The objectives of the authentication mechanism of the GOLD infrastructure are twofold; Firstly, to make sure that only the right participants enter and operate within the VO and secondly to allow the participants to move freely (within the range of the access control policies see next section) within the various security domains in GOLD. The GOLD infrastructure supports both federated and brokered trusted relationships via implementations of standardised token retrieval, token exchange, token signing and validation.

In the first instance all VO participants must login to the VO. This is done by providing some details (usually typical username and password) which are validated by the VO. Upon validation of these details the VO infrastructure issues a federated identity. For reasons such as data protection and privacy the infrastructure issues a federated identity valid only within the VO. The participants can therefore retain their privacy (the federated identity does not identify the real identity of the participant) without however any lack of accountability which is the real concern. This implies that the infrastructure maintains a traceable link between the federated identity and the real identity of the participant. Therefore both accountability and privacy are supported. Communications between the VO participants is achieved by exchanging security tokens between participants. WS-Security [WS-Security 2002] is used to carry security tokens as part of a SOAP message header. The specification allows various token structures such as X.509, Username, and XML based tokens which we customized to support SAML [SAML 2005] assertions. The structure of a message carrying a token and the life cycle of a token from request to validation is shown in the next figures. Note here that the figure implies the existence of a federation.

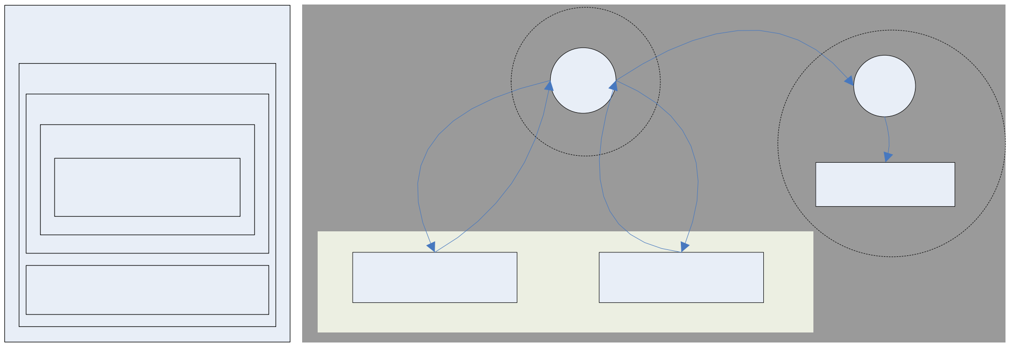


Figure 1. Secure Message Structure & Token Life Cycle

(dotted line denotes organisational boundary

According to the token life cycle, signed tokens are issued to participants from the federation. These tokens can be in the form of X.509 or SAML or Username tokens depending on the requirements expressed by the participants. As it is mentioned earlier the infrastructure should not impose any constraints regarding token usage. As such the infrastructure retrieves such requirements from service interface descriptions if those have been expressed using WS-Policy [Ws-Policy 2004] and WS-Policy Attachment specifications [WS-Policy-Attachment] that specify ways of expressing such requirements and attaching them on WSDL interface descriptions. Alternatively customised forms are offered for participants to express such requirements during the formation of the VO. The token is then sent by the requestor to the resource owner for validation. Upon validation access to the resource is granted. Note here that prior to requesting a signed token from the federation requests are authorised by additional services which are discussed in section 5.8. The above description implies that a federation has been established and that there is a direct trust relationship between the federation and the participants. If this direct relationship is not present the above model would fail. To alleviate this problem the GOLD infrastructure makes use of the WS-Trust [WS-Trust 2005] specification that offers mechanism for managing brokered trust relations. This manifests itself as a set of protocols for managing tokens and validating them in the presence of broker trust relations. The additional functionality offered is demonstrated by the following figure.

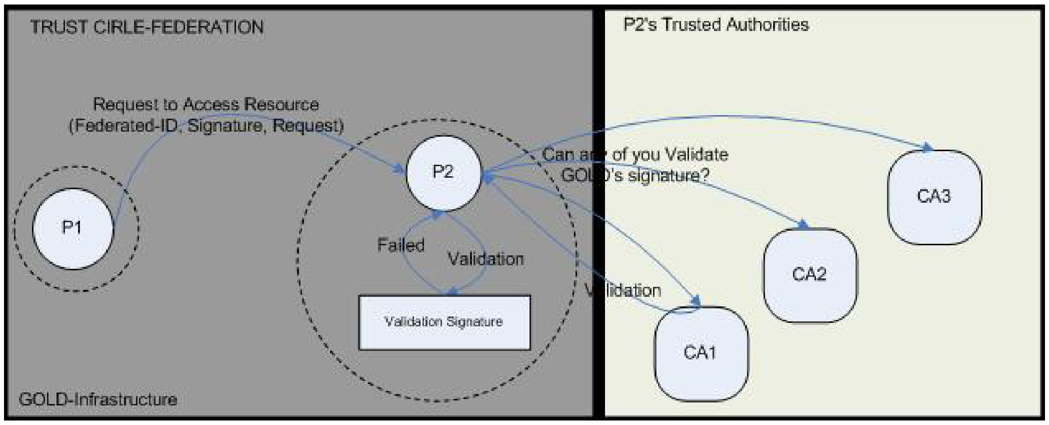


Figure 2. Token Validations via Broker Trust Relationship

WS-Trust provides the means for validating tokens in the absence of direct trust relations. Figure 5 illustrates this point. P2 fails to validate the security token issued by the GOLD infrastructure which would imply lack of direct trust between them. Using the protocols specified by WS-Trust specification [WS-Trust 2005] the token can be send for validation to certificate authorities outside the scope of the VO. As figure 5 suggests CA1 trusts the GOLD token authority and as such validates its token. P2 as a response grants access to P1 to access its resource. Having looked at the protocols we have implemented for authentication the following section discusses the authorisation process.

4 Authorisation

In earlier publications [Periorellis et al. 2004] we discussed the advantages and disadvantages of access control models, ranging from active control lists to role and task based systems. We concluded that the dynamic nature of virtual organisations makes it necessary for any VO infrastructure to support a mechanism that deals with dynamic rights activation and de-activation. In dynamic systems such VOs, both the topology of the VO (parties that form it and links between them) and its workflow are subject to change. Therefore static rights assignment prior to a workflow does not capture the eventuality of a party leaving, added or any alterations to the workflow itself. Several authors have elaborated on this issue [Coulouris, G., et al. 1994, Roshan, K., T., et al. 1997, Sandu, R., S., et al. 1996, Thomas R., K., 1997]. In addition, given the sensitivity of the information that may be shared in some VOs, (which raises concerns regarding competitive advantage) parties are not expected to be assigned a single set of rights that would last throughout the duration of a VO. It is more likely that VO participants would agree limited or gradual access to their resources depending on workflow progress. In GOLD we want to be able to restrict role access on resources depending on the execution context. In a virtual organization the execution context can be regarded as the particular project or the goal that all participants have come together to achieve. In a virtual organization there can be many projects with various participants resulting to complicated interrelationships between them. For example, some of them may play the same role in various projects, carrying out the exact same tasks, or have different roles within the same project depending on the performed task. The question we raise is 'Should a role have the same permission and access rights throughout the set of similar or even identical projects and the tasks within those projects?'

Our view is that in dynamic access control systems we should separate roles from role instances. Different role instances may require different permissions and indeed additional levels of authorization depending on the project and task in which they are active. In order to raise the levels of trust in those cases, one needs to make sure that adequately fine grained access control mechanisms are supported. Granularity refers to the level of detail for which one can define access rights. Fine grained permissions are needed for instances of roles as well as instances of objects. For example, a chemist role may be granted access to chemical documents but we do not however wish to grant access to all chemical documents produced by the system. Instead, we want any access permissions granted to the chemist role to be project-specific (e.g., the instance of a particular collaboration) as well as task-specific (e.g., the instance of a particular pre-defined set of activities). So the management of roles and access permissions in GOLD needs to be integrated with the management and monitoring of dynamic service level agreements or contracts between the participating services. The contracts can capture the expectations from specific tasks, using pre- and post-conditions. Permissions for roles can be activated and de-activated based on the progress of the monitored contracts. Since part of the infrastructure requires us to de-centralise control of the resources there are several functionalities which the infrastructure has to support including,

a) a common language for expressing authorisation policies that is understood by all participants
b) a protocol for expressing policies and rules that is understood by all participants
c) a protocol for transferring/communicating these policies across the wire
d) a centralised policy repository
e) a verification component prior to the execution of the workflow.

4.1 XACML

We tackled the some of the problems mentioned above using the eXtensible Access Control Mark up Language (XACML) which is a n OASIS WS-standard [OASIS 2003]. XACML which is an xml based language for expressing and communicating access control policies between services. It provides standard XML schema for expressing policies, rules, obligations and conditions. It also specifies a response/request protocol for sending a request and having it approved. The request/response language expresses queries about whether a particular access should be allowed (requests) and describes answers to those queries (responses). Policies are defined in terms of subjects, (i.e. users, machines, services etc) and resources (documents, machines etc). Both subjects and resources are identified using URIs. The total number of subjects and resources together, define what XACML specification terms as the target space. Earlier publications provide very detailed information about the standard [Wu et al. 2005]. The figure below describes the architecture of our service implementation.

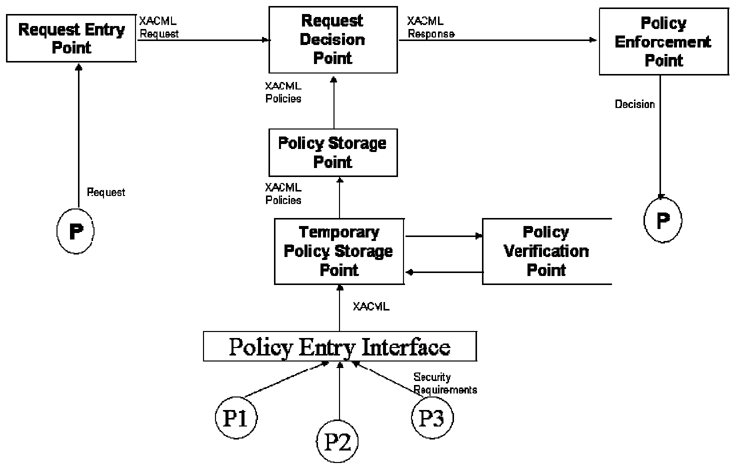


Figure 3. Access Control Service Architecture

In the above figure P represents a participant and the rest of the boxes the services that we have implemented. As the figure suggests VO participants (i.e. P1, P2 and P3) can express their security requirements using the service interface. These requirements form a set of policies that describe how individual participants want their resources to be protected. The entire set (assuming that is part of the same project or workflow) are placed in temporal storage (implemented as mySQL database) where a verification service will validate those against logical inconsistencies. The verification service guarantees that the workflow will not throw any exceptions as a result of policy mismatching. These are raised and negotiated prior to workflow enactment. The following section provides a few more technical details on this issue. Assuming that a participant P requests access to a resource (resources are expressed as URIs) several services coordinate the process of expressing a request and providing a response. As figure 4 suggests the Request Entry Point service provides an interface that accepts requests in the form of (subject or credential, resource-URI, action, project, VO) and translates these to XML documents which complies with the XACML schema for requests. The subject or credential refer to the identity of the requestor, the resource-URI is the actual resource identifier, the action corresponds to the reason the resource is requested, the project is an identifier which corresponds to a specific project and the VO is an identifier which corresponds to the VO. It is assumed that the semantic information for the action element is established during the formation stage of the VO. The Request Decision Point is the service that actually makes the decision whether a request should be granted. Upon receipt of an XACML request it retrieves the relevant policies documents from the database. At this point the evaluation takes place. This is done through a collection of Combining Algorithms. Each algorithm represents a different way of combining multiple decisions into a single decision. There are Policy Combining Algorithms and Rule Combining Algorithms. An example of these is the Deny Overrides Algorithm, which says that no matter what, if any evaluation returns Deny, or no evaluation permits, then the final result is also Deny. These Combining Algorithms are used to build up increasingly complex policies, and while there are seven standard algorithms, you can build your own to suit your needs. When a decision is reached a XACML response document is created (containing the decision) and forwarded to the Policy Enforcement Point. This is the service that parses the XACML response, retrieves the response and forces the decision on participant *P*. We have used this tool to build both role based access control as well as task based access control policies without having to conform to the federation protocol that Shibboleth [Morgan et al. 2004] requires, or the non-standard way of policy expression that Permis [Chadwick et al. 2003] provides. Additional details on comparative studies of access control approaches, is provided by [Wu et al. 2004]. The language of XACML is quite powerful in terms of expressiveness. Combining this with the verification service it allows us to dynamically alter, delete or add depending on the progress of the workflow. The entire implementation allows us to move towards a more dynamic environment of rights activation and de-activation as opposed to a static one where roles and rights have to be predetermined.

4.2 Verification

Given the plethora of policies for access control within a VO in conjunction with the fact that there is not a single authority that governs these policies (the majority will stem from participants' requirements on how they want to protect their resource) validation will be needed to make sure that there are no logical inconsistencies prior to workflow enactment. Policies would also need to be verified against industrial regulations that govern a particular sector such that of chemical development. Therefore policies should comply with the higher more abstract industry policies on how chemical development should be conducted. We are using Margrave, [Fishler et al. 2005, Greenburg et al. 2005] a tool developed at Brown University for verifying XACML policies. The tool wrapped as a Web service exposes a WSDL interface which implements several methods for the validation of XACML policies. The tool as part of our overall authorisation architecture is used to verify a set of XACML policies against logical inconsistencies that would otherwise inhibit the progress of a workflow enactment. The tool is used whenever a policy is added or altered in the database.

5 Conclusion

We attempted to describe in this paper some of the issues regarding authentication and authorization for distributed systems based on the web service architecture. We elaborated on the use of standards discussing their pros and cons before describing the mechanisms developed by the GOLD project. As part of the description of our solutions we elaborated on certain standards regarding their usage and applicability.

6 References

Chadwick D., Otenko S. 2003_1, A comparison of the Akenti and Permis authorization infrastructures, Proceedings of the ITI First International Conference on Information and Communications Technology (ICICT 2003) Cairo University*, pages 5-26,*

Chadwick D., Otenko A. 2003_2, The PERMIS X.509 role based privilege management infrastructure, Future Generation Computer Systems, Vol. 19, Issue 2, 277-289

OASIS, 2003, An Open, Role-based, Access Control Architecture for Secure Interworking Services, Cambridge University EPSRC project, http://www.cl.cam.ac.uk/Research/SRG/opera/projects/

Morgan R. L., Cantor S., Carmody S., Hoehn W., & Klingenstein K. 2004, Federated Security: The Shibboleth Approach, Educause Quarterly, Vol. 27, No. 4,

Liberty Alliance Project 2003, Introduction to the Liberty Alliance Identity Architecture, Revision 1.0.

Liberty Alliance Project 2004, Digital Identity Defined, http://projectliberty.org/

OASIS Security Services (SAML) TC, 2005 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

Periorellis, P., Townson, C. and English, P., 2004 CS-TR: 854 Structural Concepts for Trust, Contract and Security Management for a Virtual Chemical Engineering, School of Computing Science, University of Newcastle

Conlin, A.K., English, P.J., Hiden, H.G., Morris, A.J, Smith, R. and Wright, A.R. 2005, "A Computer Architecture to Support the Operation of Virtual Organisations for the Chemical Development Lifecycle", in Proc. European Symposium on Computer Aided Process Engineering, (ESCAPE 15), pp 1597-1602

Blum D., 2005, Concepts and Definitions, Identity and Privacy Strategies In-Depth Research Overview, Version 1.0, Burton Group

Sandhu R., Coyne E., Feinstein H. & Youman C., 1996, Role-Based Access Control Models, IEEE Computer, Volume 29, Number 2

SAML Specifaction OASIS, "Security Assertion Markup Language (SAML) v2.0." http://www.oasis-open.org/committees/security, 2004.

WS-Policy specification 2004, http://specs.xmlsoap.org/ws/2004/09/policy/ws-policy.pdf

WS-Policy Attachments specification 2004, http://specs.xmlsoap.org/ws/2004/09/policy/ws-policyattachment.pdf

WS-Secutity specification 2002, http://schemas.xmlsoap.org/specs/ws-security/ws-security.htm

Roshan, K. T., and Sandhu, R. S., "ask-Based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-Oriented Authorization Management, presented at IFIP TC11 WG11.3 Eleventh International Conference on Database Securty XI: Status and Prospects, 1997.

Thomas, R. K., Team-based Access Control (TMAC): A Primitive for Applying Role-based Access Controls in Collaborative Environments, presented at Second ACM Workshop on Role-based Access Control, Fairfax, Virginia, United States, 1997.

Fisler, K., Krishnamurthi, S., Meyerovich, L., A., Tschantz, M., C., Verification and Change-Impact Analysis of Access-Control Policies**,** Proceedings of the 27th international conference on Software engineering, pp196-205 2005, St. Louis, MO, USA May 15 - 21, 2005

Coulouris, G., and Dollimore, J., Security Requirements for Cooperative Work: A Model and its System Implications, presented at 6th Workshop on ACM SIGOPS European Workshop: Matching Operating Systems to Application Needs, Wadern, Germany, 1994.

Wu, J., Periorellis, P., Authorization-Authentication Using XACML and SAML, technical reports series CS-TR: 907, School of Computing Science, University of Newcastle, May 2005

Wu, J., Periorellis, P., Review of Access Control systems for Virtual Organizations, technical reports series CS-TR: 917, School of Computing Science, University of Newcastle, May 2005

WS-Trust Specification 2004, Web Services Trust Language (WS-Trust)." http://msdn.microsoft.com/ws/2004/04/ws-trust/

SAML Artifact Information Flow Revisited

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

Birgit Pfitzmann
IBM Zurich Research Lab
Rüschlikon, Switzerland
bpf@zurich.ibm.com

Abstract

The standardized OASIS Security Assertion Markup Language (SAML) has become one of the most deployed frameworks in federated identity management even though it focuses only on single sign-on. Answering industry's pursuit of the reduction of user-management costs and enabling cost-efficient deployment because of its browser-based profiles, SAML is believed to become widely used soon. With the revision to Version 2.0, especially SAML's browser/artifact profile has gained new security measures defeating old vulnerabilities. We analyze this profile and focus on the problem of artifact information flow. We devise a concrete exploit to demonstrate the impact of this problem. We address this problem by a new browser/artifact profile called Janus. The innovation is to split the artifact into two independent shares that have different information flow in a standard web browser. This new method defeats artifact information flow efficiently without relying on assumptions on the artifact lifetime.

1 Introduction

One of the recent advances of access control and user management products was the introduction of federated identity management proposals such as the Security Assertion Markup Language (SAML) [17]. Industry expects a dramatic reduction of user management costs from federated identity management by savings in password helpdesks, user management, and user deletion.

SAML features browser-based profiles that only rely on a standard web browser to carry out identity federation, e.g., by means of single sign-on. These protocols complement the general advantages of federated identity management solutions with the property of being zero-footprint. This means that do not require the installation of additional client software and are, therefore, cost-efficient to deploy. However, designing secure protocols with a standard web browser as the client is not trivial. One of the major challenges of this protocol class is that the browser is not aware of the protocol it participates in. The browser has a predefined behavior, reacts to predefined messages and generates information flow both to the underlying operating system and to communication partners. Each of these properties may put a protocol's security at stake. In particular, protocols that transfer confidential information through a browser's URL are endangered by this protocol-unaware behavior of a standard web browser. In this paper, we analyze the browser/artifact profile of SAML, a prominent example of this protocol class. It leverages the browser redirect URL to transfer a random reference to a user's credential, the so called artifact, to a service provider.

The SAML V1.1 Web SSO Browser/Artifact Profile [17] was already analyzed by [7] and suffered from some problems introduced by a standard web browser as client. In the meantime, SAML has advanced to Version 2.0 [18] and revised also this profile, repairing most of the problems discussed in [7] and discussing the improvements in an SSTC response [15]. The structure and naming in the standards has also slightly changed, hence the corresponding protocol (in the terminology of security protocol research) is now the SAML V2.0 Web Browser SSO/Response/Artifact Feature.

In the meantime, research on federated identity management protocols and in particular browser-based protocols has also advanced. Microsoft Passport [16] was the first protocol to be analyzed. Its detailed analysis by [13] also discussed inherent problems of browser-based protocols. Inherent problems of browser-based client authentication were also discussed in [5] independent of the federation aspect. Liberty [14] is an identity federation protocol that is based upon SAML V1.1, however, has also influenced the development of SAML V2.0. Weaknesses in the original version of the Liberty enabled-client profile were found by [19] and repaired in subsequent Liberty versions. Shibboleth [4] is a federated identity management solution for universities also based upon SAML. Research also started to provide positive security statements for federated identity management protocols; a profile of WS-Federation [12] was analyzed first by [8] based upon top-down assumptions and without a detailed browser model. Very recently a generic model for the analysis and security proofs of browser-based protocols was proposed [9] that is to be used for more in-depth proofs such as [10]. However, there is no positive statement for a browser/artifact profile such as the SAML V2.0 Web Browser SSO/Response/Artifact Feature yet. Research in this area is complemented by the tool-supported analysis of standards such as in the Web Services area starting with [6, 3, 2] and the original SAML V1.1 [11]. The inherent problems of modelling Web Services and identity federation protocols with Dolev Yao-alike abstractions were pointed out in [1]. However, protocols involving a standard web browser were not considered by the tool-support approach.

Our Contribution. We analyze the security measures newly introduced by version 2.0 into the browser/artifact profile of SAML [18, 15]. We discuss the generic security goals for the profile and their impact on the SAML specification. We point out the still existing problem of artifact information flow and construct a concrete exploit for a specific scenario to demonstrate its impact.

Furthermore, we turn to the general problem of information flow of SAML artifacts through a standard web browser. This problem is inherent to the whole protocol class with artifacts and may compromise the protocol security: if a valid artifact flows to an adversary, the adversary may impersonate the corresponding honest user. For instance, [7] proposed an attack based upon an adversary provoking information flow of an artifact through the browser's Referer tag. Currently, there is no solution that fully solves this problem. Though the SAML V2.0 Web Browser SSO/Response/Artifact Feature strengthened the protection against such attacks by introducing a one-time request constraint at the service provider, it can still be compromised by information flow of the SAML artifact. The potential resulting damage is only reduced by recommendations about the artifact's lifetime, assumptions about the clock skew between protocol principals, and the reasoning that an adversary cannot cause too much harm within that timeframe.

We discuss solutions to this problem that do not rely on timing assumptions. Furthermore, we devise a new variant of the SAML V2.0 Web Browser SSO/Response/Artifact Feature that renders a potential artifact information flow via the Referer tag unusable by an adversary. We do this by splitting a SAML artifact into two independent shares that produce different information flow in a standard web browser. Thus, we introduce a completely new approach into the options for solving the artifact information flow problem that does not rely on timing. Its advantages are that it uses neither additional messages in most cases, and thus does not introduce new latency into the profile, nor storage-expensive measures like artifact blacklists.

Outline. We begin our discussion of the SAML V2.0 Web Browser SSO/Response/Artifact Feature with a protocol overview in Section 2. This section introduces our notation for the elements of the SAML message standard as well as for the corresponding protocol meta-data. We complement this section with an tabular overview over SAML as multipart standard in Appendix Section A.1. In Section 3, we define authenticity as our main security goal for identity federation protocols and SAML in particular. Given this basis, we analyze concrete security properties of SAML V2.0 Web Browser SSO/Response/Artifact Feature in Section 4, where we focus of the artifact information flow as main source of vunerabilities. This analysis motivates the introduction of a new SAML profile called Janus in Section 5, which leverages the abstract idea of using two artifacts that produce a different information flow in a standard web browser and is complemented by a security and an efficiency analysis.

2 Protocol Overview

In this section, we give an overview of the SAML V2.0 Web Browser SSO/Response/Artifact Feature. SAML is a multi-part standard for which we denote the most important parts for our analysis in Table 2 of Appendix Section A.1.

According to the SAML conformance specification, the combination of profile, message exchange and a selected binding is termed a SAML V2.0 feature. This corresponds to a security protocol in the classical sense of security research. However, for a security analysis one has to remember that an adversary might try to replay message parts from one protocol in another. Furthermore, one has to remember that several steps in the protocol are not concrete algorithms, but only described by certain values to be generated and constraints on them. Similarly, the messages do not have one specific format, but a range of possible formats with constraints.

2.1 Notation

We assign identifiers to variables like identities and different URIs. We denote SAML's unique identifiers for entities as defined in [18, Metadata, Section 2.2.1] by idi for identity providers and idp for service providers. Likewise, we denote unique user identities as registered with identity providers by idu. If a variable occurs at several participants, we prefix it with the participant whose view we discuss. E.g., the assertion consumer service URI ACS of a service provider P in the view of this service provider is $\mathsf{P}.ACS$ while the view of the identity provider I of this URI in a SAML protocol run is $\mathsf{I}.ACS$. We also use the dot notation for elements of structured messages, e.g., $art.handle$ for the handle field in an artifact art. By ϵ we denote that such a parameter is not present. We use the notation $U(Params)$ to denote a URI U augmented by a list of parameters $Params$ encoded in its querystring. The predicate $\mathsf{controls}(id, URI)$ states that the participant with identity id controls the URI URI. This is a meaningful (although not fully specified) statement for SAML by the assumption that all participants in SAML have unique identifiers and that one can evaluate whether an address belongs to a participant.

2.2 Protocol Steps

Figure 1 summarizes the SAML V2.0 Web Browser SSO/Response/Artifact Feature. As indicated in the figure, we show the simpler redirect binding for the request, and the entire request phase is optional. We only show those parameters and constraints that will be most important later.

In Step 2, the service provider P requests a single sign-on authentication from the identity provider I. This is done by means of a SAML AuthnReq message that P includes in a redirect response. The service provider sends $AuthnReq$ via the browser to a single sign-on service address SSO of a desired identity provider with identity idi. Its most important parameters are $Issuer$, the request issuer, and a request identifier ID. The only constraint is that the service provider has to set the issuer parameter to its own identity idp.

In Step 4, the identity provider identifies the user by some means. We call the resulting user identity idu.

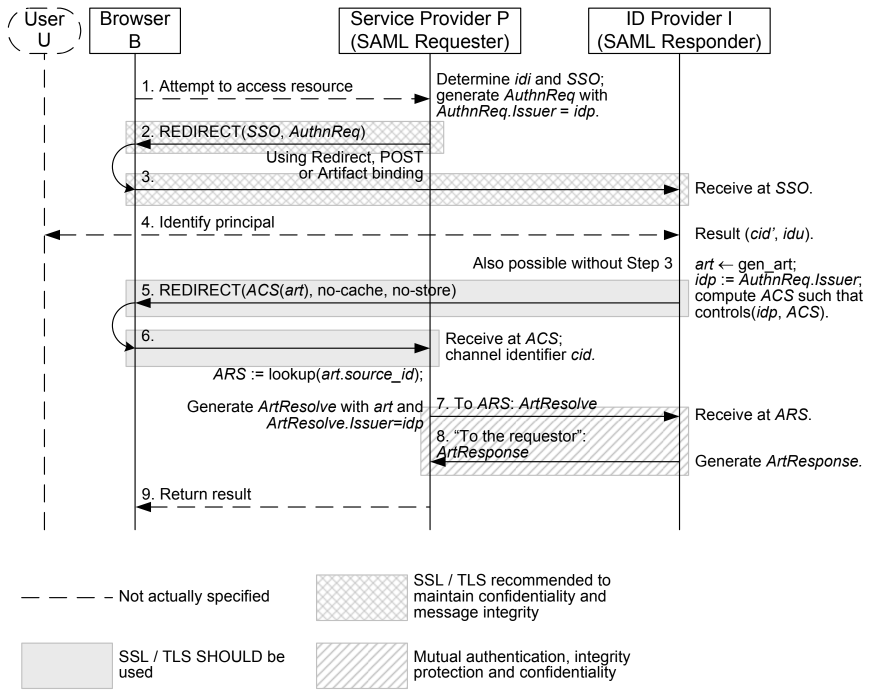

User U
Browser B
Service Provider P (SAML Requester)
ID Provider I (SAML Responder)
1. Attempt to access resource
Determine *idi* and *SSO*; generate *AuthnReq* with *AuthnReq.Issuer* = *idp*.
2. REDIRECT(*SSO*, *AuthnReq*)
Using Redirect, POST or Artifact binding
3.
Receive at *SSO*.
4. Identify principal
Result (*cid'*, *idu*).
Also possible without Step 3
art ← gen_art; *idp* := *AuthnReq.Issuer*; compute *ACS* such that controls(*idp*, *ACS*).
5. REDIRECT(*ACS*(*art*), no-cache, no-store)
6.
Receive at *ACS*; channel identifier *cid*.
ARS := lookup(*art.source_id*);
Generate *ArtResolve* with *art* and *ArtResolve.Issuer*=*idp*
7. To *ARS*: *ArtResolve*
Receive at *ARS*.
8. "To the requestor": *ArtResponse*
Generate *ArtResponse*.
9. Return result
Not actually specified
SSL / TLS recommended to maintain confidentiality and message integrity
SSL / TLS SHOULD be used
Mutual authentication, integrity protection and confidentiality

Figure 1: SAML V2.0 Web Browser SSO/Response/Artifact Feature

Then, for Step 5, the identity provider generates an artifact art and redirects the browser back to the service provider. In our case, the artifact is placed in a query string parameter named $SAMLart$, other profile variants place it in a hidden form control. The identity provider addresses this redirect to the assertion consumer service URI ACS of the service provider indicated in the request; recall that $ACS(art)$ denotes ACS plus the artifact art encoded in the querystring. The values no-cache and no-store in this message refer to Cache-Control and Pragma header fields. [18, Profiles, Section 4.1.3.5] stresses that the identity provider MUST have some means to establish that [the assertion consumer service] is in fact controlled by the service provider, which we model by the predicate $\mathsf{controls}(idp, ACS)$.

In Step 7, the service provider P opens a secure back-channel to the identity provider and asks I for the assertion corresponding to the artifact; this is done in an ArtifactResolve message $ArtResolve$. The function lookup models the lookup of an artifact resolution service address ARS under the $source\_id$ of an artifact. It is supposed to guarantee $\mathsf{controls}(idi, ARS)$, but does not directly specify how to fulfill this constraint. The profile and the artifact binding prescribe mutual authentication, integrity protection, and confidentiality for Steps 7 and 8, and both state that this can be done by signing or by binding-specific measures (here the lower-level binding for these steps is meant, e.g., using SOAP). We will assume that this authentication refers to the identities idi and idp; certainly I has a local view $\mathsf{I}.idp$ of the service provider's identity associated with the artifact art from before Step 5. P can use $art.source\_id$ to look up idi.

SAML poses several requirements on the Response Res element within $ArtResponse$ in Step 8. It is required that the issuer of the response and all assertions included match the identity provider, that is $Res.Issuer \in \{idi, \epsilon\}$ and $assert.Issuer = idi$ for every assertion $assert$ in Res. Furthermore, there must be at least one assertion, say $assert^*$, that contains an authentication statement, and that has subject confirmation method bearer and the assertion consumer URL as recipient, $assert^*.recipient = ACS$. Furthermore, if there was Step 3, then the Step 8 message must match the identifier of the Step 3 message: $Res.InResponseTo = AuthnReq.ID$.[1]

There are lifetime and one-time use properties associated with artifacts and assertions which we will describe and analyze in Section 4.

3 Security Goal: Authenticity

In this section, we define authenticity as the main security goal of the SAML V2.0 Web Browser SSO/Response/Artifact Feature.

> Authenticity means that if a service provider P has finished a SAML protocol run successfully, then it can be sure that its communication partner is a user with a certain received user identity idu and certain attributes att at identity provider I.

In order to model this security goal, we clarify the meaning of three statements: (i) Who is the service provider's communication partner considered that SSL/TLS does not provide client authentication? (ii) How is the user identity idu derived? (iii) What does a successful protocol run imply?

As the service provider P has no means to determine "its communication partner" before the SAML protocol run, we refer to the secure channel that P has with that partner. A unilaterally authenticated channel like SSL or TLS only guarantees that there is one fixed communication partner, but not the client's identity. We refer to the channel by an identifier cid.

The user identity idu is derived from the assertion $assert^*$ retrieved in Step 8 in Section 2.2. For simplicity, we assume that P derives idu from the subject identifier in this assertion: $assert^*.Subject$.

Thus we model the successful termination of a SAML protocol run by an output $(\mathsf{accepted}, cid, idu, att)$, which binds the user identity idu to the secure channel identified by cid.

We make the authenticity definition for one given service provider and one given identity provider. Though not explicitly specified in SAML, we can assume

[1]This constraint, however, cannot be verified in practice; if P receives a response Res with $Res.InResponseTo \neq \epsilon$, then P should check that there exists a valid $AuthnReq$ of P issued to idi with $Res.InResponseTo = AuthnReq.ID$.

that the different principals have means to enforce the SAML constraints about identifier and URI uniqueness and addressing, which we call setup. Thus we refine the informal authenticity statement of the beginning of this section as follows:

Honest principals. P is an honest service provider and I its honest identity provider. U is an honest user with correct browser B who has the identity idu at the identity provider I.

Setup execution. U, P, and I have executed setup according to SAML 2.0.

Condition for authenticity. *IF* the service provider P obtains an output $(\mathsf{accepted}, cid, idu, att)$ from the SAML V2.0 Web Browser SSO/Response/Artifact Feature, ...

Authenticity claim. ...*THEN* the secure channel with channel identifier cid is indeed a channel with the honest user U with identity idu (unless the user authentication and tracking of U at I is compromised by other means).

4 Analysis of Selected Security Measures

In this section, we analyze selected security measures added to SAML V2.0 [18, Bindings, Security Consideration] and described in the SSTC response [15]. We focus on the information flow of SAML artifacts. SAML has designed this part of the artifact binding carefully and taken several precautions against this problem. These precautions provide good protection against information flow attacks on the SAML artifact known in prior literature. However, this protection is (as in all browser/artifact profiles known) not perfect. Spotting a weakness in the well-elaborated harness of security measures is not trivial. Therefore, we first shed light on different aspects of the problem and corresponding security measures, in order to allow developers of future SAML or other profiles to understand the advantages and disadvantages of the different security measures. Then we construct one attack exploiting different aspects of SAML to circumvent the security measures in a specific scenario. We do this for the purpose of demonstration that the protection against artifact information flow is still not perfect and as motivation that one needs to look at further measures to solve this problem once and for all.

4.1 One Time Request Property

An important security measure for the SAML artifacts is the so-called one-time request property of the SAML artifact: the identity provider I enforces that an artifact may only be used once to obtain an assertion. If the identity provider sees the artifact a second time it will behave as if it does not know the artifact. This property provides protection from replay attacks. However, [7] proposed to interrupt the channel between service provider P and identity provider I in order to prevent an artifact from being invalidated. A counter-measure proposed by [7] and adopted by SAML V2.0 is the provision of checks of the one-time request property by the service provider P. SAML V2.0 uses a variant of this proposal, in which the service provider P puts an artifact on a blacklist only if “an attempt to resolve an artifact does not complete successfully” [18, Bindings, Section 3.6.5]. This measure defeats the so-called Referer attack of [7]. However, it does not cope with arbitrary information flow of valid artifacts to an adversary A. We consider a specific scenario that defeats this security measure in Section 4.3.

Recommendation. Step 8 of the SAML V2.0 Web Browser SSO/Response/Artifact Feature must have the postcondition that *all* artifacts that the identity provider sent out in Step 5 are invalidated either in the view of I or the view of P. One way to reach this goal is that the service provider P puts all artifacts seen on a blacklist; however, this is costly in terms of state space to hold at P. Alternatively, I can explicitly confirm the invalidation of SAML artifacts in its Step 8 response to P. Then P puts all artifacts seen and not confirmed to be invalidated on its own blacklist; consequently all artifacts are put on the blacklist if communication fails or the artifact resolution was unsuccessful.

4.2 Artifact Lifetimes

The SAML V2.0 Web Browser SSO/Response/Artifact Feature uses the short lifetime of the SAML artifacts as an additional security measure. The lifetime is specified as a few minutes, where identity providers and service providers should have clock skews of at most a few minutes [15, Section 1.3.4], [18, Security Consideration, Section 6.5.1]. We believe (and so, we think, do the SAML designers) that timestamps as freshness measure are a suitable heuristic to prevent accidental disclosure of a still valid artifact, however, if a determined adversary tries to break a SAML protocol run of a specific user, several minutes are enough time to do so. Therefore, we see timing as a complement for other security measures, yet, timing does not guarantee security. Instead, we prefer to base security on active prevention of information flow of the artifact beyond the sphere of influence of the protocol.

Recommendation. Do not rely on artifact lifetime as a primary security argument of a browser/artifact profile. Render information flow of the artifact beyond the protocol run itself impossible.

4.3 Accumulating Artifacts

The SAML security analysis [7] noted the possibility of an adversary accumulating multiple artifacts in a Step 6 redirect to the service provider. How may this possibility affect the protocol’s security? The SAML specification only prescribes service provider P to send the (one) artifact to the identity provider I [18, Core, Section 3.5.1], [18, Bindings, Section 3.6.5]; SAML does not contain provisions for handling URLs with multiple artifacts. Thus, accumulating multiple artifacts in a request may leave valid artifacts around. This leaves the adversary the option to get hold of those artifacts that were not invalidated. However, no concrete exploit based on this idea was contained in [7]; in particular it mentioned only the possibility that a malicious service provider can accumulate valid artifacts in the URL by executing Steps 3-6 repeatedly. However, these artifacts were issued for the malicious service provider P^* and will only lead to assertions accepted by P^*, and thus do not threaten authenticity by themselves.

Instead, we will now use these artifacts as a disguise for a valid artifact for an honest service provider. We devise a concrete exploit as follows. Let us assume a scenario where an adversary A wants to get access to an honest service provider P impersonating an honest user U. First A makes up some artifacts art_i with the parameters used by the identity provider I (or collects them by repeatedly executing Steps 3 to 5 with I). In addition, A contacts P in the role of a user in order to make A issue an AuthnRequest $AuthnReq$. Now A redirects the browser B of user U to the identity provider I; for this A must either intercept an unprotected Step 1 message or be contacted by U in the role of a service provider. In this redirect, it includes $AuthnReq$ from P, and all the accumulated artifacts art_i in the querystring.

I will issue an artifact art valid for P and redirect the browser B to P. The postcondition of this flow that defines the scenario to be one where the attack works is that the redirect target URL ACS is augmented by the accumulated artifacts art_i as well as art.

In Step 6, P now chooses one artifact from the URI, where SAML does not define which one. Moreover, the artifact format does not allow P to distinguish which artifact was indeed issued for it. Therefore, we can assume that there exists a combination of identity provider I and service provider P where the probability that I puts the artifact art issued for P at a different position than that one from which P takes the artifact is not negligible. Whenever this happens, i.e., P picks an artifact art_i, then P and I both invalidate art_i and not art. The valid artifact art can flow to A by means of, for instance, a browser Referer tag. Then A can impersonate U at P. For completeness, the scenario up to the leakage of art is shown in Figure 3 (Appendix A.2).

Discussion. Is such an exploit realistic? To answer this question we need to check on the one hand how browser and servers react upon having multiple parameters with the same name in a URL's query string. We checked by experiment that (i) both entities accept such input and that (ii) different servers have different heuristics which element to choose from the querystring (see Appendix, Section A.3).

On the other hand, the postcondition defined above is only fulfilled if the implementation of the identity provider's single sign-on service copies the parameters in the querystring of SSO in Step 3 into the redirect target ACS. In SAML, the exact format of those URLs is not specified, thus, an implementation doing so is behaving according to the specification. Still, the question arises whether any reasonable implementation might behave this way. Firstly, we note that identity federation systems are mostly not stand-alone solutions, but embedded in a larger access control and identity management environment. Therefore identity federation systems have a natural selection of solutions that are compliant with the access control environment. Secondly, we observe that there are access control systems that use querystring parameters internally. One example is the dynamic URL addressing of resources, which dispatches requests depending on values of querystring parameters. Another example is the enrichment of the URL querystring with a session id and user attribute parameters. An identity federation system not forwarding querystring parameters when redirecting a browser may hamper other functionality of its environment, which may lead architects of those systems to come to a design decision to copy querystring parameters where allowed.

Recommendation. A SAML deployment must be capable of handling all sorts of message formats, especially messages that contain multiple artifacts. We propose that either identity providers control that no artifact is already included in requests issued to them in Step 3, or service providers are extended by rules how to handle and invalidate multiple artifacts in Step 6.

5 The SAML Web Browser SSO/Response/Janus Artifact Feature

It is crucial to prevent information flows of a valid SAML artifact through a standard web browser. Even more so, we would like to devise an option that renders an artifact misdelivered useless for an adversary by construction. In this section, we construct a profile and binding of SAML, i.e., a feature in SAML terms, that can tolerate certain misdelivery of SAML artifacts. We call it the Janus profile according to the homonymous god of the Roman mythology, the gate-keeper, or, as full feature name, the SAML Web Browser SSO/Response/Janus Artifact Feature.

> *Janus* is the Roman god of gates and doors (ianua), beginnings and endings, and hence represented with a double-faced head, each looking in opposite directions.

Our profile is based on the idea to issue two artifacts instead of one, where each artifact produces a different information flow within a standard web browser. Following the Janus metaphor, we want the adversary to be able to observe one face of Janus, yet not both. Thus we include one artifact in the URI at P to which the browser is redirected by I as the standard SAML V2.0 Web Browser SSO/Response/Artifact Feature does. However, we include a second artifact in the last user authentication URI of I. The browser will potentially include this second artifact in the Referer tag of the Step 6 request to the service provider P. It is crucial to note that now two artifacts arrive at service provider P in Step 6, but that they have different information flow in subsequent steps.

5.1 Profile Description

As SAML does with its profiles and bindings, we specify the Janus profile by means of constraints. Actually, regarding real constraints, Janus is a sub-feature of the SAML V2.0 Web Browser SSO/Response/Artifact Feature. Therefore Janus inherits the constraints of its parent feature and extends them by using a so-called Janus artifact with additional constraints.

The profile relies on one assumption about the consistency of browser behavior:

> A browser B shows consistent Referer tag behavior if it either sets Referer tags in the communication with all servers or with none if the preconditions for Referer tags from HTTP are fulfilled.

For such a browser, the predicate $\mathsf{SetsReferer}$ is TRUE if the browser does set Referer tags. A *Janus artifact* is a SAML artifact which is chosen from two artifacts transported in different ways in the protocol, depending on $\mathsf{SetsReferer}$:

$$\textbf{if } \mathsf{SetsReferer}(\mathsf{B}) \textbf{ then } art = art_1 \textbf{ else } art_2$$

where $\in_{\mathcal{R}}$ denotes uniformly random or pseudorandom and independent choice of a value from a domain, and $l = 160$. (In general l could be a security parameter.) The two artifacts are transmitted in different ways to ensure that an adversary can obtain at most one.

Notation As in Section 2, I and P have unique entity identifiers idi and idp, respectively. The identity provider I controls two URLs: it performs user authentication and issues SAML artifacts at SSO and has its artifact resolution service at ARS. The service provider P controls the assertion consumer service URI ACS. By $\mathsf{SAMLart}$ we denote the domain of SAML artifacts; we only need to know here that a correctly chosen SAML artifact has a sufficiently large (pseudo-)random part to be guessable by an adversary only with negligible probability.

Step by Step. An overview of the Janus profile is shown in Figure 2. The general flow is as in the SAML V2.0 Web Browser SSO/Response/Artifact Feature surveyed in Section 2. We start the step-by-step description with the principal identification at I. SAML defines Step 4 as out-of-scope; in Janus we require that it ends in Step 4.z at the identity provider's address SSO and that Step 4 is done through a server-authenticated secure channel. The querystring in Step 4.z is augmented by a SAML artifact art_1 generated by the function $\mathsf{gen\_art}$. How this is done depends on the principal identification solution used by I. If the principal identification is POST-based, the HTML form querying the user for authentication may hold $SSO(art_1)$ as the submission address. Solutions based upon a GET request may issue an explicit redirect to $SSO(art_1)$ in the preceding Step 4.y. However, if the overall solution takes more than one request-response pair, the identity provider may find a way to execute Step 4.y at URI $SSO(art_1)$ without any additional round-trip.[2] Given a Step 4.z request, I derives a user identity $idu \neq \epsilon$

[2]In implementations of I where the principal identification in Step 4 is based upon a GET request

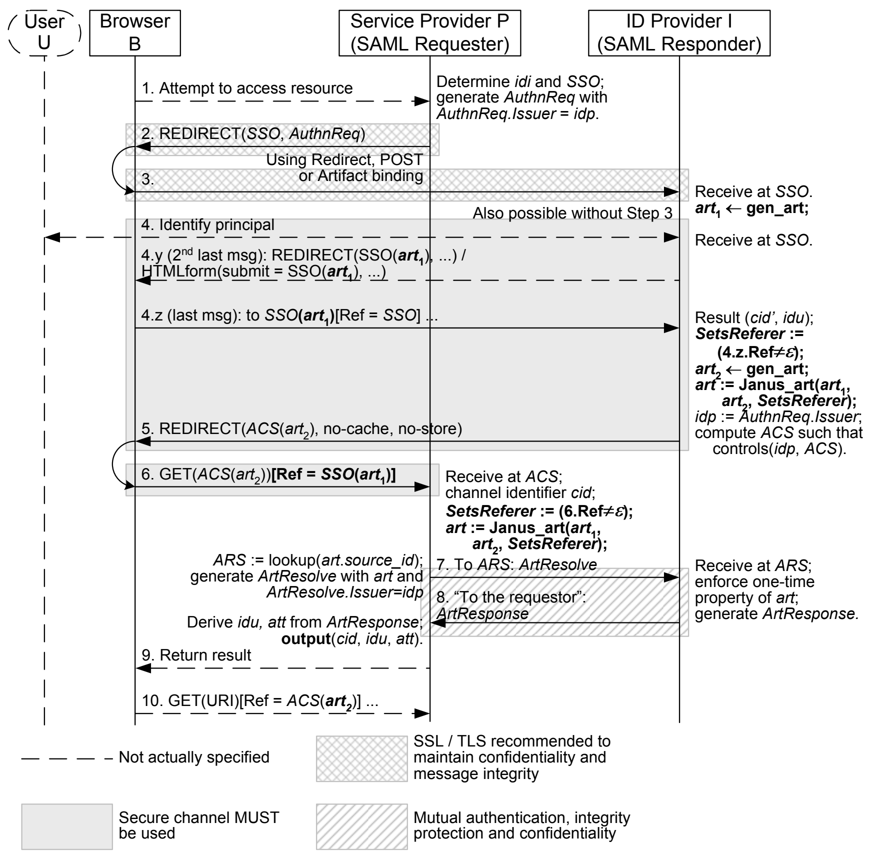

User U
Browser B
Service Provider P (SAML Requester)
ID Provider I (SAML Responder)
1. Attempt to access resource
Determine idi and SSO; generate AuthnReq with AuthnReq.Issuer = idp.
2. REDIRECT(SSO, AuthnReq)
Using Redirect, POST or Artifact binding
3.
Receive at SSO. art1 ← gen_art;
Also possible without Step 3
4. Identify principal
Receive at SSO.
4.y (2nd last msg): REDIRECT(SSO(art1), ...) / HTMLform(submit = SSO(art1), ...)
4.z (last msg): to SSO(art1)[Ref = SSO] ...
Result (cid', idu); SetsReferer := (4.z.Ref≠ε); art2 ← gen_art; art := Janus_art(art1, art2, SetsReferer); idp := AuthnReq.Issuer; compute ACS such that controls(idp, ACS).
5. REDIRECT(ACS(art2), no-cache, no-store)
6. GET(ACS(art2))[Ref = SSO(art1)]
Receive at ACS; channel identifier cid; SetsReferer := (6.Ref≠ε); art := Janus_art(art1, art2, SetsReferer);
ARS := lookup(art.source_id); generate ArtResolve with art and ArtResolve.Issuer=idp
7. To ARS: ArtResolve
Receive at ARS; enforce one-time property of art; generate ArtResponse.
8. "To the requestor": ArtResponse
Derive idu, att from ArtResponse; output(cid, idu, att).
9. Return result
10. GET(URI)[Ref = ACS(art2)] ...
Not actually specified
SSL / TLS recommended to maintain confidentiality and message integrity
Secure channel MUST be used
Mutual authentication, integrity protection and confidentiality

Figure 2: Janus profile, or SAML Web Browser SSO/Response/Janus Artifact Feature.

corresponding to the principal identification; this is bound to the channel identifier cid'. Furthermore, I tests whether the browser B is setting Referer tags by checking whether the Step 4.z request contains such a tag. Potential Referer tags are indicated in the figure by elements Ref in brackets. Here we assume that the authentication method used gets the address SSO of Step 4.z from a source with its own URI, so that the precondition for setting the Referer Tag is fulfilled.

In Step 5, I generates a second SAML artifact art_2 independently from art_1 using gen_art, and computes a Janus artifact art from art_1 and art_2. Next, I derives the service provider's entity identifier idp from $AuthnRequest.Issuer$ and computes ACS such that controls (idp, ACS) holds. Finally, it redirects the browser to $ACS(art_2)$ by sending a REDIRECT response to the channel with identifier cid'.

Upon the browser's GET request at the end of the redirect (Step 6), service provider P checks whether the browser has set a Referer tag. Then P can also apply the function Janus_art. Note that if SetsReferer(B) is false, then P does not

and only takes one request-response pair, or where identification is retained from a previous protocol run, I needs to issue one additional redirect message to direct the browser to $SSO(art_1)$ and therefore loses the round-trip-time advantage of Janus. However, a typical case is an initial internal redirect to an authentication service.

have art_1, but also does not need it in this function.

Now P looks up the identity provider's artifact resolution service URI ARS by means of the artifact's $source\_id$.

In Step 7, P sends the artifact art in an $ArtResolve$ message to ARS to resolve the SAML artifact. The identity provider looks up the SAML artifact and enforces the one-time request property. If the artifact can be resolved to an assertion, then I sends this assertion enclosed in an $ArtResponse$ message to service provider P in Step 8. Recall that Janus inherits the security checks and constraints prescribed by the SAML V2.0 Web Browser SSO/Response/Artifact Feature for these steps, in particular mutual authentication and confidentiality for Steps 7 and 8.

5.2 Security

The core security property achieved by Janus artifacts is the *general prevention of artifact information flow*. This property is crucial for the security of all SAML artifact profiles and renders the SAML 2.0 WebSSO Browser/Janus Artifact Feature superior to other proposals. We claim the following information flow property:

> If the trust and setup preconditions of authenticity of Section 3 are true, and the browser has consistent Referer tag behavior (Section 5.1, then the Janus profile defined in Section 5.1 produces no information flow from the Janus artifact art to other parties than P, I and B.

We have two cases depending on the Referer tag behavior of the browser.

Case 1: B *sets Referer tags.* If the identity provider I observes in Step 4.z that browser B sets Referer tags, it uses art_1 as the Janus artifact art. Here art_1 is issued in the redirect location and Step 4.y and used in the URI in Step 4.z. Thus the browser B puts art_1 in the Referer tag of the subsequent HTTP request, which is Step 6. The usages in Step 4 are over a secure channel and thus unobservable except for I and B, and Step 6 is over a secure channel to an address controlled by P so that only P learns art_1 here. P sees art_1 in the Referer tag and therefore computes the same Janus artifact art as I. P uses the artifact art in Step 7, but this step provides confidentiality, so that there is no information flow of art here except back to I. There is no further use of art or art_1 in the profile. Hence an adversary has no advantage in guessing art.

Case 2: B *does not set Referer tags.* In this case, the protocol flow is identical to the normal SAML V2.0 Web Browser SSO/Response/Artifact Feature; however, information flow by means of the Referer tag is now prevented by the precondition of the case.

5.3 Efficiency

How does the Janus profile behave compared to other measures that address the artifact information flow problem? We consider three dimensions of efficiency measures: message complexity, storage complexity at the service provider P, and assumptions about the environment. We compare our proposal to the original SAML 2.0 WebSSO Browser/Artifact Feature and discuss two prominent alternatives to enhance its security apart from Janus. We give an overview of these methods and their efficiency in Table 1. In this table, we use the variable n for the total number of artifacts I has sent/P has seen; f denotes the number of artifacts that failed to be send to the identity provider I.

We start with the alternative that the service provider employs a *cleaning self-redirect* after Step 8 to make the browser strip off a potentially valid SAML artifact in the Referer tag (see Table 1, column "Redirect"). Such solutions are widely used by e-mail providers preventing a user's session identifier to be disclosed when redirecting the user's browser to other servers. However, these solutions have the disadvantage of additional round-trip times.

Another proposal is to enforce a *full one-time request property* at the service provider P (see Table 1, column "Full Blacklist"). Instead of storing only a blacklist of artifacts where the resolution failed, the service provider stores *all* artifacts seen. In Section 4.1 we proposed a light-weight alternative to store all artifacts that were not confirmed by the identity provider I to be invalidated. Still, both solutions burden service providers with storing a potentially high number of artifacts. Additionally, an adversary may attack this solution by sending large numbers of artifacts in Step 6 messages to the service provider and have the service provider exhaust its storage bounds for the artifacts. Moreover, any cleanup measures on the artifacts that are based on expiration times would again be based on timing assumptions.

Table 1: Efficiency measures for SAML 2.0 WebSSO Browser/Artifact proposals.

	SAML 2.0	**Redirect**	**Full Blacklist**	**Janus**
Msg POST Auth	0	+2	0	0
Msg GET Auth	0	+2	0	(+2)
Storage P	$O(f)$	$O(f)$	$O(n)$	$O(0)$
Storage I	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Assumptions	timing P,I ∞ cache	timing P,I ∞ cache	timing P,I ∞ cache	consistent B

With the SAML Web Browser SSO/Response/Janus Artifact Feature, we pursue a solution that does not have these drawbacks (see Table 1, column "Janus"). This new solution does not need an additional self-redirect in most cases, nor does it rely on P storing lots of artifacts or assumptions about the artifact's lifetime. Actually, the artifact in the Step 6 ACS URL may indeed flow to the adversary. The point of the Janus profile is that this artifact is completely worthless for an adversary. We recall that the profile relies on the assumption that a standard web browser behaves consistently in communication with other servers: either the browser sends Referer tags to all servers, or does not send them to anyone. A browser that does not send Referer tags to P, yet, does send Referer tags to another server reach-able from Step 9 of the profile, may break the profile. If one does not trust this assumption, we recommend to complement the Janus profile with the light-weight blacklist measure with additional cleanup after times significantly beyond the artifacts' lifetimes, and thus using only a very weak timing assumption.

6 Conclusion

We have analyzed the SAML V2.0 Web Browser SSO/Response/Artifact Feature and focused on the problem of information flow of the SAML artifact, which is inherent to all browser/artifact profiles. For a specific scenario, we have devised a concrete exploit that circumvents the current security measures in SAML and demonstrates such an information flow.

With the Janus profile, or SAML Web Browser SSO/Response/Janus Artifact Feature, we have devised a novel efficient solution to this problem. This solution neither relies on timing assumptions about the artifact's lifetime, nor does it need additional messages in most cases, nor does the service provider have to hold state in the form of blacklists or similar space-consuming measures. Only leveraging the information-theoretical or computational independence of two artifact shares and an assumption about the consistency of a standard browser's behavior, it presents a new approach to the problem of artifact information flow.

References

[1] M. Backes and T. Groß. Tailoring the Dolev-Yao abstraction to web services realities. In *Proceedings of the 2005 ACM Workshop on Secure Web Services (SWS)*, pages 65–74. ACM Press, Nov. 2005.

[2] M. Backes, S. Mödersheim, B. Pfitzmann, and L. Viganò. Symbolic and cryptographic analysis of the Secure WS-ReliableMessaging scenario. In *In Foundations of Software Science and Computation Structures (FOSSACS)*, volume 3921 of *Lecture Notes in Computer Science*, pages 428–445. Springer-Verlag, Berlin Germany, 2006.

[3] K. Bhargavan, C. Fournet, and A. D. Gordon. A semantics for web services authentication. In *31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 198–209. ACM Press, 2004.

[4] S. Cantor and M. Erdos. Shibboleth-architecture draft v05, May 2002. `http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-arch-v05.pdf`.

[5] K. Fu, E. Sit, K. Smith, and N. Feamster. Dos and don'ts of client authentication on the web. In *Proceedings of the 10th USENIX Security Symposium*, Washington, D.C., Aug. 2001. USENIX. An extended version is available as MIT-LCS-TR-818.

[6] A. D. Gordon and R. Pucella. Validating a web service security abstraction by typing. In *Proc. 2002 ACM Workshop on XML Security*, pages 18–29, Fairfax VA, USA, Nov. 2002.

[7] T. Groß. Security analysis of the SAML Single Sign-on Browser/Artifact profile. In *Proc. 19th Annual Computer Security Applications Conference*. IEEE, Dec. 2003.

[8] 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.

[9] T. Groß, B. Pfitzmann, and A.-R. Sadeghi. Browser model for security analysis of browser-based protocols. In *ESORICS: 10th European Symposium on Research in Computer Security*, volume 3679 of *Lecture Notes in Computer Science*, pages 489–508. Springer-Verlag, Berlin Germany, 2005.

[10] T. Groß, B. Pfitzmann, and A.-R. Sadeghi. Proving a WS-Federation Passive Requestor profile with a browser model. In *Proceedings of the 2005 ACM Workshop on Secure Web Services (SWS)*, pages 54–64. ACM Press, Nov. 2005.

[11] S. M. Hansen, J. Skriver, and H. R. Nielson. Using static analysis to validate the SAML single sign-on protocol. In *Proceedings of the 2005 workshop on Issues in the theory of security (WITS '05)*, pages 27–40, New York, NY, USA, 2005. ACM Press.

[12] 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/`.

[13] D. P. Kormann and A. D. Rubin. Risks of the Passport single signon protocol. *Computer Networks*, 33(1–6):51–58, June 2000.

[14] Liberty Alliance Project. Liberty Phase 2 final specifications, Nov. 2003. `http://www.projectliberty.org/`.

[15] J. Linn and P. Mishra. SSTC response to "security analysis of the SAML Single Sign-on Browser/Artifact", working draft 01, Jan. 2005. `http://www.oasis-open.org/committees/documents.php?wg_abbrev=security`.

[16] Microsoft Corporation. .NET Passport documentation, in particular Technical Overview, and SDK 2.1 Documentation (started 1999), Sept. 2001.

[17] OASIS Standard. Security assertion markup language (SAML) V1.1, Nov. 2002.

[18] OASIS Standard. Security assertion markup language (SAML) V2.0, Mar. 2005.

[19] B. Pfitzmann and M. Waidner. Analysis of Liberty single-signon with enabled clients. *IEEE Internet Computing*, 7(6):38–44, 2003.

A Appendix

A.1 SAML 2.0 as a Multipart Standard

Table 2: SAML 2.0 as multi-part standard

Document	**Element**	**Description**
Core		Basic message formats.
	Assertion	Corresponds to security tokens or credentials in other terminologies. An assertion has an issuer, typically a subject about whom the issuer asserts something, and optional elements like signatures and conditions.
	Request/Response	Message pairs that can transport assertions.
	AuthnRequest/ Response	Transport a request for authentication and the resulting one or more assertions.
	ArtifactResolve	Asks for the real response referred to by an artifact.
	ArtifactResponse	Delivers this response.
Bindings		Bind messages to concrete transport protocols (e.g., HTTP)
	Artifact Binding	Describes how an artifact is transported through a browser. The corresponding actual SAML message is retrieved directly, using the artifact as a reference.
Profiles		Define entire sequences of message exchanges, for instance for single sign-on. Due to reference to the bindings, these profiles are rather modular.

A.2 Details of the Artifact Accumulation

In Figure 3 we depict the artifact accumulation attack.

A.3 On Multiple Querystring Parts

As an example that there are sites that accept multiple querystring elements with the same name, and that different sites interpret these querystrings differently, one can check out the ACM and IEEE digital libraries:

For ACM, a given URL is `http://portal.acm.org/browse_dl.cfm?linked=1&part=transaction&coll=portal`. We now add another element named “part”. This URL gives the same page: `http://portal.acm.org/browse_dl.cfm?linked=1&part=transaction2&part=transaction&coll=portal&dl=ACM`. In contrast, the following one looks up non-existing transactions “transaction2”: `http://portal.acm.org/browse_dl.cfm?linked=1&part=transaction&part=transaction2&coll=portal&dl=ACM`. The conclusion seems to be that the last version counts.

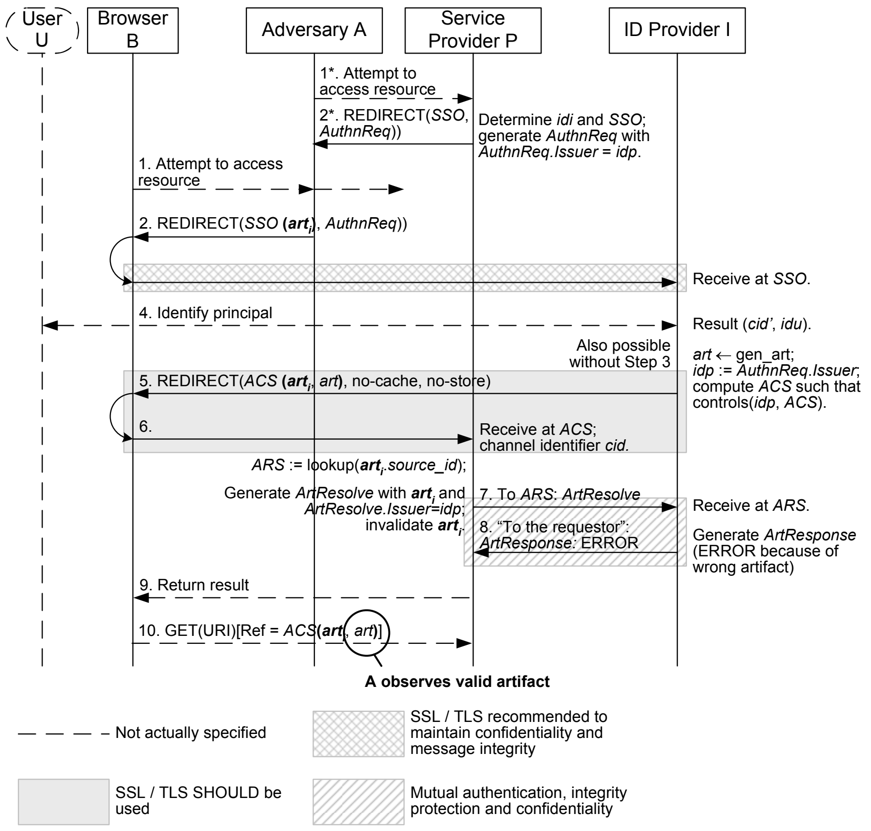

User U
Browser B
Adversary A
Service Provider P
ID Provider I
1*. Attempt to access resource
2*. REDIRECT(SSO, AuthnReq))
Determine idi and SSO; generate AuthnReq with AuthnReq.Issuer = idp.
1. Attempt to access resource
2. REDIRECT(SSO (art_i), AuthnReq))
Receive at SSO.
4. Identify principal
Result (cid', idu).
Also possible without Step 3
art ← gen_art; idp := AuthnReq.Issuer; compute ACS such that controls(idp, ACS).
5. REDIRECT(ACS (art_i, art), no-cache, no-store)
6.
Receive at ACS; channel identifier cid.
ARS := lookup(art_i.source_id);
Generate ArtResolve with art_i and ArtResolve.Issuer=idp; invalidate art_i
7. To ARS: ArtResolve
Receive at ARS.
8. "To the requestor": ArtResponse: ERROR
Generate ArtResponse (ERROR because of wrong artifact)
9. Return result
10. GET(URI)[Ref = ACS(art_i, art)]
A observes valid artifact
Not actually specified
SSL / TLS recommended to maintain confidentiality and message integrity
SSL / TLS SHOULD be used
Mutual authentication, integrity protection and confidentiality

Figure 3: Artifact accumulation attack up to artifact leakage

In contrast, for the correct URL `http://www.computer.org/portal/site/transactions/index.jsp?&pName=transactions_level1&path=transactions/tc/mc&file=author.xml&xsl=article.xsl&` adding an element `file=author2.xml` after the original one keeps the page, while adding it before the original gives an error. So here the first version seems to count.