Tailoring the Dolev-Yao Abstraction to Web Services Realities
– A Comprehensive Wish List –

Michael Backes and Thomas Groß
IBM Research Division
Rüschlikon, Switzerland
mbc@zurich.ibm.com, tgr@zurich.ibm.com

ABSTRACT

Web Services are an important series of standards for adding semantics to web-based and XML-based communication. For analyzing the security of Web Services protocols composed of these standards, it is tempting to exploit their similarity to traditional security protocols by first transforming them into the Dolev-Yao abstraction, where cryptographic operators are treated symbolically as constructors of a free algebra, and as a second step by applying existing symbolic techniques for machine-assisted or even fully automated protocol verification within this abstraction.

We show in this paper that this approach tends to ignore intrinsic aspects of Web Services standards and protocols and to hence be too coarse-grained for capturing Web Services security in all its facets. We identify a series of such aspects both on the conceptual level and on the level of concrete Web Services protocols: service requestors and providers have additional properties independent of the protocol under consideration and hence offer additional attack possibilities, protocol behaviors can be defined by explicit Web Services policies and complex message parsings which do not necessarily follow the common Dolev-Yao-style parsing conventions, etc. We sketch in a series of examples how to exploit these aspects for mounting successful attacks against Web Services protocols, and we discuss possibilities to circumvent these attacks. In particular, this exemplifies the need for tailoring Dolev-Yao abstractions specifically to Web Services idiosyncrasies, which go beyond the standard Dolev-Yao assumptions.

Categories and Subject Descriptors: C.2.0 [Computer-Communication Networks]: General–Security and protection

General Terms: Security, Theory, Standardization

Keywords: Web Services security, Dolev Yao, federated identity management, protocol model, security analysis, security proof of protocols, formal method, tool support

1. INTRODUCTION

Web Services are a series of standards that add higher-layer semantics and quality of service to web-based communication. They use XML as the basic format for all exchanged content and SOAP as the basis for message exchanges [15]. In principle, Web Services are independent of the underlying transport protocol; in practice, as the name suggests, typical web protocols are commonly used. Security features in the Web Services area are addressed by a set of standards and pre-standard proposals that can, at least syntactically, be combined in a highly flexible way. It is well-known, however, that combinations of security elements have to be treated with care in the sense that many combinations may not yield the properties that a naive user might expect.

Security analyses of Web Services protocols resemble analyses of traditional security protocols in many respects. In particular, owing to the distributed-system aspects of multiple interleaved protocol runs, both Web Services and traditional security protocols analyses are known to be error-prone and awkward to make for humans. Security analyses of traditional security protocols have been extensively studied during the last twenty years which in particular resulted in symbolic techniques for machine-assisted or even fully automated protocol verification. From the start, the actual cryptographic operations in such proofs were idealized into so-called Dolev-Yao models, following [17] with extensions in [20, 41]. These models replace cryptography by term algebras, e.g., encrypting a message m twice does not yield a different message from the basic message space but the term $\mathsf{E}(\mathsf{E}(m))$. A typical cancellation rule is $\mathsf{D}(\mathsf{E}(m)) = m$ for all m. It is assumed that even an adversary can only operate on terms by the given operators and by exploiting the given cancellation rules. Today, the Dolev-Yao abstraction has asserted its position as an appropriate model to formally analyze security protocols which is substantiated by a comprehensive body of literature (a very partial list includes [42, 39, 32, 16, 40, 33, 37, 47, 51, 1]).

It is hence tempting to exploit the similarities of Web Services and traditional protocols by first transforming Web Services protocols into corresponding Dolev-Yao-style representations, and as a second step by applying existing symbolic techniques for machine-assisted or even fully automated protocol verification within this abstraction. Recent work has demonstrated the feasibility of this approach for Web Services protocols [12, 10, 11, 13, 34, 28, 4]; some of these works additionally discovered previously unknown attacks on the protocol under consideration which underlines the detection of structural attacks against security protocols as the premium capability of Dolev-Yao-style analyses. The natural next step already postulated in some of these works is to move from finding attacks against Web Services protocols to showing the absence of all attacks against such protocols. While this has proven promising for the analysis of traditional security protocols based on a vari-

SWS'05, November 11, 2005, Fairfax, Virginia, USA.

ety of different techniques, and while this is clearly a worth-while goal to strive for, we point out in this paper that solely relying on the Dolev-Yao model tends to ignore intrinsic aspects of Web Services standards and protocols and to hence be too coarse-grained for capturing Web Services security in all its facets. We deliberately speak of a tendency here since several of these aspects can potentially be faithfully reflected in the Dolev-Yao abstraction yet require explicit, non-standard extensions of the abstraction in order to capture Web Services realities.

We identify several of such aspects both on the conceptual level and on the level of concrete Web Services protocols. Overall, the collection of these aspects serves as a comprehensive wish list of what has to be reflected for achieving proofs of security in Dolev-Yao-style analyses of Web Services protocols, or which aspects might have to be investigated by complementary techniques. The identified aspects range from features of the actual Web Services servers and clients, e.g., that service requestors and providers have additional properties independent of the protocol under consideration and hence offer additional attack possibilities, to specific scheduling aspects, e.g., that protocol behaviors can be defined by explicit Web Services policies and complex message parsings which do not necessarily follow the common Dolev-Yao-style parsing conventions, to aspects related to unexpected information flows and XML-based message representations. We sketch in a series of examples how to exploit these aspects for mounting successful attacks against Web Services protocols, and we discuss possibilities to circumvent these attacks. The feasibility and especially the simplicity of these attacks further exemplify the need for tailoring Dolev-Yao abstractions specifically to Web Services idiosyncrasies, which go beyond the standard Dolev-Yao assumptions. Finally, let us stress that we do not claim novelty for all of these aspects. Some of them have already been investigated in existing analyses conducted in specific Dolev-Yao models, but these aspects are often tacitly abstracted away in other analyses as well, yielding "provably secure" protocols that are susceptible to simple attacks. Some of the identified aspects however appear more difficult to reflect in the Dolev-Yao model and might hence require additional proof effort based on complementary techniques.

Overview We start with a brief summary of the Dolev-Yao model and its intrinsic assumptions in Section 2. We then investigate different aspects whose Dolev-Yao-style representations typically differ from Web Services realities. We start with aspects of Web Services servers and clients in Section 3, move on to aspects of actual Web Services profiles and their respective message flows in Section 4, and finally investigate aspects concerning the XML-based message representations and the relationship to actual cryptography in Section 5.

2. THE DOLEV-YAO ABSTRACTION AND ITS USAGE IN SYMBOLIC ANALYSES

The Dolev-Yao abstraction has established itself as an appropriate model to formally analyze security protocols by idealizing the behavior and properties of cryptographic operations. Technically, the Dolev-Yao abstraction abstracts from the details of real cryptography by considering a term-algebra of messages, where concrete data like participants' names, nonces, or keys are represented by constants, and cryptographic operations like encryptions and digital signatures are represented by function symbols, i.e., they constitute constructors over the algebra. Furthermore, the algebra is equipped with certain cancellation rules. For instance, public-key encryption is represented by operators E for encryption and D for decryption with one cancellation rule, $\mathsf{D}(\mathsf{E}(m)) = m$ for all m. Encrypting a message m twice in this model does not yield another message from the basic message space but the term $\mathsf{E}(\mathsf{E}(m))$. This abstraction allows for exploiting symbolic reasoning techniques for analyzing security protocols and hence for leaving the tedious parts of proofs to machines, i.e., model checkers or automatic theorem provers. This means to code the cryptographic protocols into the language of such tools, which may need more or less start-up work depending on whether the tool already supports distributed systems or whether interaction models have to be encoded first.

Inherent in the Dolev-Yao models considered in the literature, there are several simplifying assumptions. First and foremost, the adversary is restricted to the operations of the algebra. Since cryptographic objects are represented by monolithic terms that cannot be further decomposed, e.g., on the bitstring level, this in particular implies that the adversary is not able to mount attacks that require explicit bitstring manipulations. Moreover, there is an assumption that syntactically different terms always represent different bitstring messages. Another assumption is that it is not defined a-priori what happens when someone attempts to decrypt an encrypted message using a wrong key, or how messages are parsed if their format does not match the expected format of the respective protocol message. Usually, Dolev-Yao models resolve such situations by following commonly accepted parsing conventions as described in [30] and hence would throw an error or drop the respective message. Another assumption commonly taken for analyzing concrete protocols in the Dolev-Yao model is that honest parties only run the considered protocol stand-alone, or at least that no other concurrently running protocol relies on the same secrets, e.g., signature keys.

3. MODELING WS-SECURITY ENVIRONMENTS

An intrinsic aspect of real WS-Security deployments is that WS-Security protocols are not executed by protocol machines that only run the protocol under consideration, but the protocols are considered and explicitly designed as components of larger systems and built upon underlying transport protocols that may impact the behavior of the WS-Security protocol. More precisely, WS-Security protocols are executed in a client/server environment and mostly executed over HTTP. In this section, we identify essential properties of WS-clients and WS-server engines and point out in which respects the behaviors of protocol principals in reality, i.e., in actual Web Services deployments, differ from their corresponding behaviors in the Dolev-Yao abstraction. We exemplify the relevance of these differences by pointing out possible attacks that are not necessarily visible in the Dolev-Yao abstraction yet that successfully exploit these properties for attacking actual Web Services deployments. We furthermore discuss possibilities to circumvent these attacks. We start with properties of the server environment and move on to properties of the client.

3.1 Properties of Server Environments

WS-Security servers do not constitute homogeneous entities that each only host a single service. Instead, they usually consist of a complex stack built of an underlying web server, an application server, a servlet engine, and a Web Services engine. Such a so-called compound server usually hosts many different services that are executed concurrently. Moreover, these services may have different security levels and purposes. Thus, the common assumption of the Dolev-Yao abstraction that protocols are executed stand-alone, or at least that they do not share long-term secrets with other protocols, cf. Section 2, is usually not justified in real WS-

Security deployments. This gives rise to classes of attacks that can be mounted against real WS-Security deployments yet that remain undetected in analyses in the Dolev-Yao abstraction. In this section, we show that real WS-Security deployments can be susceptible to very simple attacks exploiting these issues. We start by considering issues arising from different service identities, proceed with issues arising from key ownership, and finally investigate issues related to multiple services and policies.

Service Identities Web Services protocols usually rely on a diversity of different identities associated with a WS-Security server and its services. The relation between these identities and their ownership is moreover often non-trivial. The following identities are core in the context of a WS-Security server and its services:

(i) the *hostnames* of the server, used as an identity for normal channels. A server may control multiple hostnames. Due to insecure DNS, bindings of hostnames to servers may be manipulated by an adversary. (We stress that manipulation of hostnames is not a problem that is specific to Web Services, but that applies similarly to traditional security protocols.)

(ii) the identity of the public-key certificate, used for the *server-side authentication* of secure channels. The corresponding secret key is usually under the control of the web server or the application server.

(iii) the actual *URI of the service* under consideration.

(iv) the *signature and encryption keys*, used for creating WS-Security messages or tokens.

(v) the *identifier URI*, used in Issuer and Audience fields of Web Services tokens.

The hostname, the certificate, and the URI maintain a strong relation among each other and are required to be in the same hierarchy. However, usually multiple services are covered by the same hostname and the same server-side authentication of secure channels. Also, the identifier URI used in the Issuer and Audience fields may be unrelated to the other identities and shared by multiple Web Services protocols. Hence explicit security checks are required for ensuring that different identities are correctly related to each other in a protocol run in order to detect man-in-the-middle attacks. In the Dolev-Yao abstractions, some or all of these identities are usually consolidated into a single identity [28, 4]. This simplification however bears the risk of incomplete security checks and hence of undetected, successful attacks. An example pointing out an attack not covered by such abstractions is as follows.

EXAMPLE 1 (CONSOLIDATION OF IDENTITIES). There exists a vulnerability of the SAML V1.1 Single Sign-on Browser/Artifact profile [45] discovered in [22]. This profile uses an SAML artifact, i.e, a pseudorandom number bound to the identity supplier's URI_{S}, used to reference a credential. The identity supplier also stores the URI_{C} from which the identity consumer receives such artifacts. The request to the identity consumer contains a third identity $target$ that references the service to be accessed.

Consider a scenario where the identity consumer hosts a low-level security service (e.g., a stock exchange information service) and a high-level security service (e.g., an e-banking application). As the artifact is only bound to the identity supplier by URI_{S} and to the identity consumers artifact receiver URI_{C}, the artifact can be used to authenticate at any service hosted by the identity consumer. Thus, the low-level security service can use its information to impersonate a user to a high-level security service. In SAML V1.1, the relation of the actual $target$ address, i.e., the service on the identity consumer, and the artifact receiver URI is not properly taken into account. SAML V2.0 [46] repairs this problem by distinguishing service addresses of the same identity consumer. In general, it is suitable to differentiate entity identifiers on the service level.

This illustrates the need for either carefully modeling all identities and their relationships within the Dolev-Yao abstraction, or for proving that the actual security checks related to identities are comprehensive enough to justify a consolidation into one identity. To retain the full automation of such proofs, modeling all identities and their relations explicitly seems to be the preferred solution here.

Key Ownership Web Services protocols usually rely on cryptographic signature and encryption keys for setting up secure or authenticated channels, for establishing security tokens etc. An intrinsic aspect of actual WS-Security deployments is that some of the corresponding private keys are not under control of the actual principals for the WS-Security protocol under consideration, but they are controlled by an entity hosting several other services as well. For instance, a secure channel authenticated by means of a server-certificate is established with the web server or application server, not with the protocol principal for the concrete WS-Security protocol analyzed.

However, these aspects are not easy to synchronize with the assumption commonly taken in Dolev-Yao models that protocols are analyzed stand-alone, that secret keys are under control of the principal, and that their usage is restricted to the protocol under consideration. For instance, the Dolev-Yao assumption of a stand-alone protocol execution and key separation would bind the keys for the establishment of a secure channel to the principal, and not to the hosting service. The keys for signing and encrypting WS-Security tokens or messages are directly affected by the key separation assumption, but in real deployments these keys are often controlled by a protocol handler responsible for multiple protocols of the same class; in fact, it is not even unusual to delegate these signing and encryption operations to a separate service in the Web Services engine, cf. the mechanisms offered by the WS-Trust standard [26]. Such assumptions occur in Dolev-Yao models of WS-Security protocols, e.g., [4, 10], as well as in security proofs building upon other models, e.g., [23]. The exclusive binding of keys to the service under consideration might however result in missed protocol interference attacks.

EXAMPLE 2 (SHARED KEYS FOR MULTIPLE PROTOCOLS). Let a server S run multiple Web Services protocols, where security tokens, e.g., SAML assertions, are generated by a dedicated service. Also, let the dedicated token service set issuer and audience fields of its SAML assertions similar to the WS-Federation Passive Requestor Interop (WSFPI) [31, 29] as domain name of the server URI_{S}. We consider a scenario in which the server runs the WSFPI protocol, which was proven secure in [23], and another insecure POST protocol INSEC that also uses SAML assertions and the same server identifier.

Consider a protocol run of INSEC in which a honest user wants to single sign-on on service C. The adversary may rewrite the address of the service targeted to an address on C^*. The server will have its token service issue a SAML token for the INSEC protocol run. The dedicated server sets the issuer to URI_{S} and the audience to URI_{C^*}. The adversary may observe this token in the subsequent communication of INSEC and insert the token in a protocol run of the secure WSFPI protocol.

It is important to carefully choose protocols that share the same signature key. To prevent protocol interference attacks, it is suitable to use a signature key exclusively for one specific Web Services protocol. Moreover, there exists prior work on the secure composition of protocols in the Dolev-Yao model, pointing out specific conditions under which a stand-alone analysis of such protocols is justified, cf. [27].

Finally, there are profiles that explicitly prescribe the use of the same keys for the establishment of secure channels and for message-based authentication and encryption. These profiles may successfully pass an analysis conducted in the Dolev-Yao model because of the inherent key separation assumption, but they might be vulnerable when used in a real Web Services deployment because of the possibility of interference of different protocol layers (e.g., secure channel and WS-Security). [23, 24, 25] treat secure channels in a modular manner and impose the requirement on the WS-Security deployments not to use the channel keys for other purposes.

EXAMPLE 3 (SHARED KEYS ON DIFFERENT LAYERS). Let us consider a hypothetic secure channel protocol that contains a message exchange, in which the client sends a nonce to the server and the server answers with a signature of this nonce.

$$\mathsf{C} \rightarrow \mathsf{S} : N; \mathsf{S} \rightarrow \mathsf{C} : \mathsf{sig}_{\mathsf{S}}(N);$$

Also, consider a higher-level Web Services protocol that uses signed WS-Security tokens for authentication. An adversary A can construct a unsigned WS-Security token m^* that states an identity of a valid user account. The adversary generates a list of digests of message elements to be signed: $((id_1, \mathsf{hash}(m_1)), \ldots, (id_n, \mathsf{hash}(m_n))$. The adversary uses the secure channel protocol as a signature oracle as follows: A acts in client role and initiates a secure channel establishment with S. During the channel establishment, A sets

$$N := ((id_1, \mathsf{hash}(m_1)), \ldots, (id_n, \mathsf{hash}(m_n));$$

and sends N to the server. $\mathsf{A} \rightarrow \mathsf{S} : N; \mathsf{S} \rightarrow \mathsf{A} : \mathsf{sig}_{\mathsf{S}}(N);$ Thus, adversary A could obtain the signature value corresponding to a WS-security token A forged.

It is suitable to use different keys for message and channel security.

Adversary in Server Role It is a reasonable assumption that an adversary may also act in a server role. This assumption is supported by actual Web Services environments as well as by actual Dolev-Yao models. An adversary acting in a server role especially comprises the so-called bogus merchant described in [35] as a special case. The Web Services environment permits such adversaries, since the passive requestors usually trust a large set of Root Certification Authorities (CAs) and hence may fall prey to weak certification policies of some of the CAs in the trust trees. Thus, an adversary may obtain a key and certificate for an address of its own. Dolev-Yao models often allow static corruptions of certain principals by which an adversary may corrupt an honest principal and use that participant's private keys.

Dolev-Yao abstractions that take the design decision not to let an honest party connect to the adversary or a corrupted party such as [28] exclude a whole class of adversaries and vulnerabilities. We propose to let the adversary at least act in the role of the common bogus merchant. In such a scenario one can prove that the adversary still cannot impersonate an honest user to an honest server.

Multiple Services and Policies Recent research already started to weaken the assumption that honest parties run the protocol under consideration stand-alone. Thus, the models of the Web Services protocols take into account other services that may be executed on the same host. The following small example demonstrates that such services may indeed be used for successful cross-protocol attacks.

EXAMPLE 4 (MULTIPLE SERVICES). In the WS-Federation Passive Requestor profile [31], the service at the identity consumer is specified by two parameters: the $wtrealm$, which specifies the security domain of the identity consumer, and the $wreply$ address, which names the exact URI to which the identity supplier redirects the passive requestor. The identity supplier checks that $wreply$ is in the hierarchy of $wtrealm$.

Assume that the application server runs a test service that simply outputs all messages it receives. An adversary can manipulate the $wreply$ field of the initial messages of the WS-Federation Passive Requestor profile as they are allowed to be transmitted through an insecure channel. The adversary is bound to the hierarchy constraint $wreply \subseteq wtrealm$, however, the adversary can redirect the passive requestor to arbitrary services within the security domain $wtrealm$, e.g., the test service assumed.

It is suitable to enforce that all services of a security domain follow a suitable security policy.

To discover such attacks in a Dolev-Yao abstraction, the assumption of other services on the same host needs to be modeled. We can identify two major approaches in current research for dealing with this problem. On the one hand, [23, 24, 25] took the approach of devising top-down assumptions about the security policies of other services in the same security domain, i.e., covered by the same certificate for a secure channel. On the other hand, we have the approach of modeling scenarios according to concrete security policies [11]. This approach currently concentrates on message-based security measures, but it clearly has the potential to cover more complex properties of Web Services protocols. While the first approach may fall prey to over-abstraction and miss attacks where concrete services on a server do not fulfill the security policy assumed, the second approach is very concrete and may thus yield analysis results that are no longer easily applicable if the protocol is composed with services in other policies.

A possible way to resolve this situation would be that the standard bodies devise a general policy document that is applicable to all profiles and services of the corresponding message standard. Such a document could serve as an agreed common ground of security assumptions on arbitrary services of the same server. Consequently, approaches with top-down assumptions about the service behavior can refine their assumption according to the general policy document whereas policy-driven approaches can use this document as a wildcard for arbitrary yet compliant services. We believe that such a policy document is inherently different from the security considerations already taken in the Web Services message standards and profiles. The security considerations given are profile-centric, i.e., they propose methods that should guarantee that the profile under consideration be not compromised. The common policy, however, ensures that a profile does not compromise *other* profiles.

3.2 Client Properties

The landscape of clients for WS-Security protocols is diverse: there are active requestors, enhanced browsers and fully passive requestors, e.g., standard browsers. Those client types differ in their cryptographic capabilities and their flexibility in protocol runs.

Levels of Authentication WS clients may be authenticated on different levels. First, there is the possibility to achieve client-authentication on the channel level, i.e., authentication by means of a client-certificate for the underlying mutually authenticated secure channel. Secondly, authentication can be achieved based on a preceding key exchange conducted by the mechanisms offered by WS-SecureConversation. Thirdly, a client may authenticate itself through a server-side authenticated secure channel by sending specific information through the channel, e.g., a username-password token or a SAML assertion [45, 46].

Whereas secure channels with mutual authentication are often modeled and well understood, semi-anonymous secure channels that model SSL without client-certificate have only rarely been considered yet. Several works aimed at modeling such channels, e.g., [49, 24]; however, these works do not rely on the Dolev-Yao abstraction and are hence difficult to incorporate in Dolev-Yao style security analyses. A major step in this direction was the analysis and partially formal treatment of SSL and TLS [52, 43, 48, 36]. A direct modeling of an SSL channel used for analyzing SAML with tool-support was recently given in [28].

Modeling Password-Based Authentication A second challenge is the proper modeling of username/password authentication. A common approach to capture passwords in the Dolev-Yao abstraction is to consider passwords as fresh nonces, i.e., as sufficiently long, fully random strings [13, 10, 11]. The approach taken in [13, 10, 11] additionally allows the adversary to provoke the generation of new user accounts. The approach currently does not model passwords as low entropy secrets, i.e., the adversary is not granted the possibility to guess passwords. As a consequence, this approach requires further refinement for coming up with a fine-grained analysis of password-based protocols, aimed at distinguishing between security against online attacks and security against offline attacks. Technically, such a refinement could be based on additional calculi that formally reflect low-entropy secrets and that hence can be used to reason about security against offline attacks, e.g. [14, 18]. Another approach for reasoning about passwords is to consider the user authentication as a separate sub-protocol and to subsequently prove properties about the protocol itself [23, 24].

EXAMPLE 5 (PASSWORD-DIGEST AUTHENTICATION). Consider a password-authenticated Web Services that uses the Digest Authentication [44] over an insecure channel. Simplified, the digest has the form

$$N, id_U, \mathsf{Hash}(N, pwd),$$

where N is a nonce, id_U is the authenticated user identity, and pwd is the corresponding password. This construction should be considered insecure as elements except for the password are public, hence an adversary can launch an offline attack on the low-entropy secret, the password. However, Dolev-Yao abstractions that model the password as a fresh nonce will not recognize this attack since there is not formal deduction of pwd from the accumulated terms.

A suitable way to circumvent this problem is to never transmit a password or password digest without appropriate secure channel or message security.

Adversary in Client Role An adversary may act as a client in a Web Services protocol. We can think of two common scenarios where this is important. On the one hand, there are client types that do not hold an identity on their own and act anonymously. We have such situations with passive requestors, e.g., standard web browsers. On the other hand, one can assume that the adversary is able to obtain valid user accounts which it exploit for attacking the accounts of honest users.

Both scenarios have in common that the adversary may obtain additional information by direct interaction with servers instead of only by observing and manipulating the network traffic. Although it is not unusual in Dolev-Yao models to let the adversary act in any role, there are examples of Web Services analyses that took design decisions against this [12]. A small example how an adversary could use this additional knowledge is the following:

EXAMPLE 6 (REFERENCE GUESSING). The WS-Federation Passive Requestor profile allows to reference security tokens issued by means of a URI. This method is similar to the SAML artifacts, however, the structure of these resource pointer URIs is not as thoroughly defined. Consider a WS-Federation protocol that consistently uses secure channels but does not use cryptographically strong pseudo-random numbers as resource pointers. If the adversary only observes the network traffic, he does not gain knowledge about the resource pointers because of the secure channels used. However, if the adversary is allowed to interact with the server directly, it can learn a set of valid resource pointers and gain an advantage on guessing a valid resource pointer for an honest user.

To prevent such guessing attacks, all references to credentials, security tokens and other valuable data should be generated as cryptographically secure pseudo-random number of appropriate length.

4. MODELING PROFILES AND MESSAGE FLOWS

While the previous section specifically considered properties concerning the environment of Web Services protocols in general, this section investigates the modeling of concrete profiles and message flows within the Dolev-Yao abstraction. This section in particular shows that reflecting all relevant aspects of Web Services protocol flows within the Dolev-Yao abstraction is a difficult task, but it also exemplifies the need for tailoring the abstraction towards Web Services idiosyncrasies in order to allow for faithful and comprehensive proofs of security.

Non-Dolev-Yao Constraints WS-Security profiles are usually specified by means of explicit constraints, i.e., restrictions of the principals' behavior, yet without prescribing concrete instantiations of these constraints. This degree of freedom gives rise to whole classes of such constraints for which we do not see an easy, canonical transformation into the Dolev-Yao abstraction. Simple examples are non-trivial relations between meta-data of a party, e.g., that one URI must be within the hierarchy of another URI; another example is that a certain URI must be owned by a certain principal. Not thoroughly reflecting these aspects might make a protocol susceptible to malicious attacks. A second class of constraints prevalent in Web Services protocols are timing constraints. Such constraints might claim that an artifact be stored on a blacklist after it was received or that the clocks of two principals in a protocol run be synchronized up to a few minutes. Since the Dolev-Yao abstraction typically relies on a fully asynchronous, sequential scheduling model without offering a quantitative notion of time, it does not easily allow for reasoning about such constraints. As a consequence, this might leave timing attacks undetected.

EXAMPLE 7 (TIMING BLACKLISTS). The SAML V2.0 Web SSO Browser/Artifact profile [46] provides an artifact blacklist at the identity consumer serving as a protection mechanism for artifacts potentially leaked to a malicious party. The identity consumer stores all received artifacts for a specific period of time. However, since artifacts may be removed from this blacklist when they are still valid in the view of the identity supplier that issued the artifact, a tight clock synchronization between both principals is an important requirement.

Timing attacks of Web Services protocols can be avoided by enforcing synchronization between principals as good as possible and by adding significant overlap times to anticipate a possible desynchronization.

A third class of constraints consider explicit, sometimes non-standard manipulations of the state of a principal. While protocol analyses based on the Dolev-Yao abstraction let the principals accumulate their knowledge during a protocol run, and while the general parsing conventions typically check if, e.g., a received nonces matches a nonce that was previously sent, Web Services protocols are often explicitly defined to act differently. On the one hand, Web Services protocol engines partially act statelessly by only reacting upon incoming messages without considering the history of the current protocol run. In particular, they might discard some or all of their knowledge about previously received messages. On the other hand, Web Services protocols might explicitly prescribe a principal to delete pieces of information from their state as security measures. An important example of such security measures is the invalidation of SAML artifacts which enforces that a SAML artifact is valid only once and will hence be considered unknown in case it will be subsequently received.

EXAMPLE 8 (STATE MANIPULATION). Consider Example 7. If the Dolev-Yao abstraction does not properly reflect that the identity consumer removes artifacts from its blacklist again, then the consumer does never accept an artifact twice in the Dolev-Yao abstraction. In real deployments however, the consumer accepts the artifact again after the removal from the blacklist.

Information Flow Since WS-clients and WS-servers constitute parts of a diverse environment, information flow to different communication partners as well as to underlying components such as operating systems have to be investigated. While the Dolev-Yao abstraction is suitable for reasoning about information flow that occurs on the actual network, Web Services additionally require the modeling of other kinds of information flow. Two different kinds of such flows of information can be distinguished: First, there are information flows via channels different from the network; secondly, there are information flows induced by the behavior of an entity that is not specified or investigated on the layer of the Web Services protocol under consideration.

For WS-clients, these problems mostly occur for passive or enhanced requestors since those have a predefined behavior and are partially unaware of the Web Services protocol itself. Their predefined behavior might result in an information flow to their communication partners as well as to the underlying operating system. Although this information flow is neither specified nor investigated in the Web Services protocol under consideration, it might easily compromise desired security properties of the protocol. Information flow to the operating system constitutes another channel different from the usually investigated network. Storing history, cache and password storage on the local hard disk is a problem in kiosk scenarios and may crucially affect protocols that transfer confidential information through the URL or inappropriate cache directives. Information flow introduced by a client behavior that was not specified on the Web Services layer was illustrated in [22] which described a vulnerability in SAML using information flow resulting from a SAML artifact.

EXAMPLE 9 (ARTIFACT INFORMATION FLOW). In a protocol flow of the SAML V1.0 Single Sign-on Browser/Artifact profile [45], an adversary can provoke the information flow of a SAML artifact by interrupting the secure channel between the identity consumer and the identity supplier. Given such a communication failure, the identity consumer will send an error message to the browser, potentially containing a link to an unprotected URI. Upon clicking such a URI, an insecure browser sends the URI previously visited in the so-called Referer tag and therefore leaks the SAML artifact over an insecure channel.

Security measures to such information flow attacks were proposed by [22] and subsequently reflected in [46].

For WS-servers, log entries, header data stored, or passing communication through different layers can serve as additional channels for passing over information to an adversary. Furthermore, an adversary may explicitly trigger information flow by provoking specific server behaviors that are not specified on the Web Services layer, e.g., by provoking error messages of transport protocol layers such as SOAP and HTTP.

Underlying Transport Protocols Web Services protocols may be executed based on various transport protocols. The structure of the underlying layer however has immediate impact on the message flow of the Web Services protocol itself. Consider a Web Services protocol that is executed with an HTTP over an SSL binding. In this case, Web Services protocols inherit the client/server respectively request/response structure of this transport protocol. They are hence bound to the communication structure of the underlying protocol. In this scenario, the underlying transport binding has the inherent property that the client connects to the server for establishing a channel. Then the client sends a request which in turn causes the server to send its response through the same channel. This property is inherited by a Web service protocol on top of this transport binding. Thus, it is highly unlikely that a server —upon having received a request from a passive requestor— will open a new communication channel to the passive requestor and send its response through this new channel. If a Dolev-Yao abstraction neglects the properties of the transport binding, it will at least produce false positives, as we see in the following example:

EXAMPLE 10 (TRANSPORT PROTOCOL PROPERTIES). Consider a SAML V1.0 Single Sign-on Browser/Artifact [45] protocol run with HTTP over SSL binding. According to HTTP 1.1 [21], a server uses the channel opened by the client to send a response to an HTTP request. Thus, both the request and the response are sent over the same channel and hence naturally have the same security properties.

A recent SAML V1.1 analysis [28] pointed out a vulnerability in SAML because SAML does not specify security constraints for the response from the identity consumer to the browser. This however is a false positive since it relies on the assumption that the server may send its response by other means than the channel established, and moreover with weaker security properties.

5. MODELING THE MESSAGE LAYER

Parsing Conventions Incoming messages in the Dolev-Yao abstraction are typically processed according to commonly accepted parsing convention, cf. Section 2. In particular, this kind of parsing follows a stringent pattern matching, causing incoming messages that do not fit the expected form in terms of type or number of parameters to be discarded. Web Services protocols with their strong focus on extensibility allow parsing of additional elements of unforeseen structure and type, which might be potentially unknown to the principal. This yields a situation where parsing messages according to the Dolev-Yao conventions might drop messages that Web Services parsers would consider valid. Again this gives rise to attack possibilities if the parsing of messages is not carefully adapted to Web Services realities. A promising approach in this direction has been put forward in [12, 13] which represents messages by specific predicates that reflect the structure of actual Web Services messages, and by defining the parsing accordingly. In its present state, this method however does not yet allow arbitrary ordering of terms and does not yet capture the extensibility of Web Services protocols in its full generality. Substantiated by the positive results in [12, 13], reducing the ambiguities between commonly used Dolev-Yao parsing and the deviated parsing conventions in the Web Services area by tailoring the abstraction to the Web Services extensibility can be seen as one of the crucial yet manageable points for coming up with comprehensive security proofs of Web Services protocols.

Message Structure In general, we have two dimensions in the structure of Web Services messages. On the one hand, there is the nested structure of elements that may contain attributes and additional elements. On the other hand, elements of Web Services messages can reference other elements of the message or other resources. While the term algebra of the Dolev-Yao abstraction is in principle capable of modeling the tree-structure of Web Services messages, e.g., by introducing a constructor for each type of tag recognized, actual models of Web service protocols in Dolev-Yao often take the design decision to abstract from all these elements and restrict their attention to cryptographically relevant terms. There are proposals to do this automatically, e.g., [12, 11, 34], which partially directly parses XML schema files to construct a Dolev-Yao-style representation. Other proposals derive the Dolev-Yao abstraction systematically but still by hand [23, 4, 28]. As far as the references to other elements are concerned, most proposals abstract from the references by applying the Dolev-Yao constructors/functions directly to the terms referenced [23, 4]. Some recent proposals treat references in a more fine-grained manner by explicitly adding a notion of references to their Dolev-Yao abstraction [12, 13, 34].

Context-Awareness and XML Signatures Web Services protocols use the XML signature standard [19] to ensure data integrity of messages and message parts. Although the name suggests that the standard uses asymmetric digital signatures, the name in most cases refers to symmetric message authentication codes (MACs). If a message m should be authenticated with respect to a key sk, this is however not achieved in the expected manner by computing $\mathsf{mac}(sk(m)$, but by the following construction:

EXAMPLE 11 (STRUCTURE OF XML SIGNATURES). Let $m_1, \dots m_n$ be message parts to be signed with identifiers $id_1, \dots id_n$. Then the corresponding XML signature essentially has the following structure:

$$\mathsf{mac}_{sk1}((id_1, \mathsf{hash}(m_1)), \dots, (id_n, \mathsf{hash}(m_n)).$$

The so-called digest of a message part m_i is generated by lookup of the corresponding identifier id_i and hashing of the message part referenced. The order of the message parts in the signature and the message does not necessarily correspond.

We stress that such an XML signature may reference different message parts that should be included into the signature. If only certain parts of a message are signed, this construction can be problematic: The semantic of an element of a Web Services message strongly depends on its position in the nested element hierarchy, i.e., the element's context. The signature, however, is not aware of this context, since it is a context-independent statement of the form "There exists an element in the message that has the identifier id referenced and matches hash value h". The following example illustrates this vulnerability, which is similar to the one presented by Fournet at the DIMACS Workshop on Security of Web Services and E-Commerce in 2005 and the more general attacks in [38]:

EXAMPLE 12 (XML SIGNATURE OF THE BODY TAG). Consider an honest principal that produces a message containing a signature on the body element b. An adversary can rewrite this message such that the old body element b is enclosed by arbitrary other tags, whereas the adversary adds another body element b^* to the message. The signature stays still valid as the body b still exists in the message and is validated by the signature. However, the parser of the message will take b^* as body element.

More generally, one could move arbitrary elements with generic names, such as the Timestamp tag of WS-Security, to different positions of the message without rendering the signature invalid. However, this changing might affect the context-dependent semantics of the overall message.

EXAMPLE 13 (XML SIGNATURE OF GENERIC TAGS). Consider the scenario that a Web Services token contains two WS-Security tokens, each of them equipped with a Timestamp tag to state the token's expiration time. We further assume that there is one signature over the message that specifically references both timestamp elements. An adversary can rewrite the message and exchange the timestamps from both WS-Security tokens without rendering the XML signature invalid.

Attacks of this type and suitable best practices are presented in more details in [38] in the very same proceedings.

Both examples have in common that in the Dolev-Yao abstraction, signatures are typically built and analyzed directly on the corresponding elements, i.e., without taking the referencing into account, and only implicitly maintain context on the order of the elements signed.

Order of Cryptographic Primitives A long raging discussion in the Web Services community is the proper order of integrity and encryption measures. This discussion is flanked by two side issues that makes a final conclusion more difficult. XML DSIG signatures treat asymmetric signatures and symmetric message authentication codes (MACs) in the same way although they have different cryptographic properties.

We start by considering the order of symmetric encryption and message authentication codes. In a Dolev-Yao abstraction the order of MAC and encryption typically does not cause a major difference. For actual cryptography however, there exist results by Krawczyk [36] showing that first encrypting the message and subsequently authenticating the ciphertext yields a secure, authenticated channel; reversing the order however does not automatically guarantee authentication of the message and hence requires additional, protocol-specific reasoning.[1] Hence encrypt-then-authenticate should be the preferred choice here. The formal structure recommended by [36] is:

$$c := \mathsf{enc}_{sk_1}(m);\, a := \mathsf{mac}_{sk_2}(c); \mathsf{Send}(c, a)$$

Note that actual XML DSIG are still different from this construction due to the indirection by introducing the prior hashing of all data under the MAC.

For the composition of asymmetric encryption and digital signatures the situation is slightly different. Here, both encrypt-then-sign and sign-then-encrypt can be used to set up a secure channel

[1] However, it was also shown in [36] however that this authenticate-then-encrypt approach serves as a secure channel if it is implemented with a secure block cipher in CBC mode or a secure stream cipher.

[3]. For the case that the adversary may be an insider of the system under consideration—which we believe is the appropriate scenario for Web Services protocols—non-repudiation, i.e., a signing principal should not be able to deny that it has signed the considered message, has not been taken into account in [3]. Web Services protocols, however, may benefit from this property, especially since digital signatures are often used for authenticating security tokens and not for channel establishment. A signature is often regarded as a proof of ownership of the actual message. When a message is encrypted first and then signed, this property is lost for the original message:

EXAMPLE 14 (LOSS OF NON-REPUDIATION). Assume that Alice sends a message m to Bob in which Alice specifies her novel invention to make Web Services bullet-proof secure. Alice has the message encrypted and then signed. The adversary intercepts message m, strips off Alice' signature and signs it itself. Now, the adversary sends this message m' to Bob without knowing its content. Bob will assume that the adversary is owner of the invention described.

We stress that this tampering will already be detected in analyses based on existing Dolev-Yao abstractions. Because of the non-repudiation of the message m, early robustness principles for cryptographic protocols recommend to sign first and then encrypt [2]. For asymmetric signatures in Web Services tokens, we consider this method as good rule of thumb.

$$s = \mathsf{sig}_{sks}(m);\, c := \mathsf{enc}_{sk_1}(m, s);\, \mathsf{Send}(c)$$

However, one needs to be aware that non-repudiation is not always needed or wanted. One may deliberately give up this property in favor of, e.g., anonymity or blinding of data as firstly suggested by [50]. Thus, this is a design decision that needs to be taken after careful deliberation.

The Perfect Cryptography Assumption Finally, the Dolev-Yao abstraction crucially relies on the assumption that cryptography is perfect, i.e., cryptographic operators are treated as symbolic constructors in a free algebra, and they can only manipulated according to a limited number of well-defined rules. It is known that, even if the symbolic analysis is careful in distinguishing primitives like symmetric encryption and authentication, and even if one assumes that an implementation is made with primitives secure according to the strictest usual cryptographic definitions, the results of such a symbolic analysis may not carry over to the real implementation. The most prominent example is that it cannot be avoided in general that the length of encrypted payload data leaks. Other problems that may occur in general scenarios are due to the probabilism of secure public-key encryption, key-stealing attacks, and manipulations of symmetric encryptions unless authenticated encryption [9, 8] is used in the implementation of the cryptographically sound Dolev-Yao-style library of [6, 5]. Consequently, in a Dolev-Yao-style abstraction designed to be implemented based on arbitrary cryptographically secure primitives and to be usable in a secure way within arbitrary protocols with arbitrary security properties, both the symbolic version and the real version must have certain idiosyncrasies. Hence we consider it interesting future work to augment formal tools for analyzing Web Services protocol by the idiosyncrasies of the cryptographically sound, symbolic Dolev-Yao style model of [6, 7, 5]. While implementing the primitives of WS-Security with the extended realizations from those papers (e.g., some additional tagging and randomization) might realize the goal of Web Services security to offer completely composable primitives also in a semantic sense, neither has been done yet. Other work on bridging the gap between symbolic and cryptographic security concentrated more on keeping very close to standard symbolic and real versions at the cost of generality. However, at present none of them covers broad protocol classes as are needed for Web Services protocols, nor the security properties required, but it may be interesting future work to extend such results on restricted usage of cryptographic libraries to the typical usage in WS-Security protocols. First important steps in this direction can be found in [4].

6. REFERENCES

[1] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 1999.

[2] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, 1996.

[3] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In L. Knudsen, editor, *Advances in Cryptology – EUROCRYPT '2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107, Amsterdam, The Netherlands, Apr. 2002. Springer-Verlag, Berlin Germany.

[4] M. Backes, S. Mödersheim, B. Pfitzmann, and L. Viganò. Symbolic and cryptographic analysis of the Secure WS-ReliableMessaging scenario. Technical Report IBM Research Report RZ 3619, IBM Research Division, Aug. 2005.

[5] M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *Proc. 17th IEEE Computer Security Foundations Workshop (CSFW)*, 2004. Full version in IACR Cryptology ePrint Archive 2004/059, Feb. 2004, `http://eprint.iacr.org/`.

[6] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations (extended abstract). In *Proc. 10th ACM Conference on Computer and Communications Security*, pages 220–230, 2003. Full version in IACR Cryptology ePrint Archive 2003/015, Jan. 2003, `http://eprint.iacr.org/`.

[7] M. Backes, B. Pfitzmann, and M. Waidner. Symmetric authentication within a simulatable cryptographic library. In *Proc. 8th European Symposium on Research in Computer Security (ESORICS)*, volume 2808 of *Lecture Notes in Computer Science*, pages 271–290. Springer, 2003. Extended version in IACR Cryptology ePrint Archive 2003/145, Jul. 2003, `http://eprint.iacr.org/`.

[8] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology: ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2000.

[9] M. Bellare and P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient constructions. In *Advances in Cryptology: ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 317–330. Springer, 2000.

[10] K. Bhargavan, R. Corin, C. Fournet, and A. Gordon. Secure sessions for web services. In *ACM Workshop on Secure Web Services (SWS)*. ACM Press, to appear, 2004.

[11] K. Bhargavan, C. Fournet, and A. Gordon. Verifying policy-based security for web services. In *Proc. 11th ACM Conference on Computer and Communications Security*, pages 268–277, 2004.

[12] K. Bhargavan, C. Fournet, A. Gordon, and R. Pucella.

TulaFale: A security tool for web servics. In *Proc. 2nd International Symposium on Formal Methods for Components and Objects (FMCO)*, 2003. To appear in Springer LNCS, Revised Lectures, 2004.

[13] 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.

[14] B. Blanchet. Automatic proof of strong secrecy for security protocols. In *Proc. 25th IEEE Symposium on Security & Privacy*, pages 86–100, 2004.

[15] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. Simple object access protocol (SOAP) 1.1, May 2000.

[16] M. Burrows, M. Abadi, and R. Needham. A logic for authentication. Technical Report 39, SRC DIGITAL, 1990.

[17] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[18] P. H. Drielsma, S. Mödersheim, and L. Viganò. A formalization of off-line guessing for security protocol analysis. In A. V. Franz Baader, editor, *LPAR*, volume 3452 of *LNAI*, pages 363–379. ETH Zürich, Computer Science, Springer-Verlag, Berlin Germany, Mar. 2005.

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

[20] S. Even and O. Goldreich. On the security of multi-party ping-pong protocols. In *Proc. 24th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 34–39, 1983.

[21] R. T. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616: Hypertext transfer protocol – HTTP/1.1, June 1999. Status: Standards Track.

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

[23] 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.

[24] 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. To appear; preliminary version IBM Research Report RZ 3600, April 2005.

[25] T. Groß, B. Pfitzmann, and A.-R. Sadeghi. Proving a WS-Federation Passive Requestor profile with a browser model. In *2005 ACM Workshop on Secure Web Services (SWS)*, Fairfax, Virginia, USA., Nov. 2005. ACM Press. To appear.

[26] M. Gudgin and A. N. (ed.). Web Services Trust Language (WS-Trust), Feb. 2005. Available at `http://www-106.ibm.com/developerworks/library/specification/ws-trust/`.

[27] J. Guttman and J. Thayer Fabrega. Protocol independence with disjoint encryption. In *Proc. 13th IEEE Computer Security Foundations Workshop (CSFW)*, pages 24–34, 2000.

[28] 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.

[29] M. Hur, R. D. Johnson, A. Medvinsky, Y. Rouskov, J. Spellman, S. Weeden, and A. Nadalin. Passive Requestor Federation Interop Scenario, Version 0.4, Feb. 2004. `ftp://www6.software.ibm.com/software/developer/library/ws-fpscenario2.d%oc`.

[30] F. Jacquemard, M. Rusinowitch, and L. Vigneron. Compiling and verifying security protocols. In *Proc. 7th International Conference on Logic for Programming and Automated Reasoning (LPAR)*, volume 1955 of *Lecture Notes in Computer Science*, pages 131–160. Springer, 2000.

[31] C. Kaler and A. N. (ed.). WS-Federation: Passive Requestor Profile, Version 1.0, July 2003. BEA and IBM and Microsoft and RSA Security and VeriSign, `http://www-106.ibm.com/developerworks/library/ws-fedpass/`.

[32] R. Kemmerer. Analyzing encryption protocols using formal verification techniques. *IEEE Journal on Selected Areas in Communications*, 7(4):448–457, 1989.

[33] R. Kemmerer, C. Meadows, and J. Millen. Three systems for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, 1994.

[34] E. Kleiner and A. Roscoe. On the relationship of traditional and web services security protocols (extended abstract). Unpublished manusscript, available from `http://web.comlab.ox.ac.uk/oucl/work/eldar.kleiner/`, 2005.

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

[36] H. Krawczyk. The order of encryption and authentication for protecting communications (or: how secure is SSL?). In *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 2001.

[37] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proc. 2nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer, 1996.

[38] M. McIntosh and P. Austel. Xml signature element wrapping attacks and countermeasures. In *2005 ACM Workshop on Secure Web Services (SWS)*, Fairfax, Virginia, USA., Nov. 2005. ACM Press. To appear.

[39] C. Meadows. Using narrowing in the analysis of key management protocols. In *Proc. 10th IEEE Symposium on Security & Privacy*, pages 138–147, 1989.

[40] C. Meadows. Formal verification of cryptographic protocols: A survey. In *Proc. ASIACRYPT '94*, volume 917 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 1994.

[41] M. Merritt. *Cryptographic Protocols*. PhD thesis, Georgia Institute of Technology, 1983.

[42] J. K. Millen. The interrogator: A tool for cryptographic protocol security. In *Proc. 5th IEEE Symposium on Security & Privacy*, pages 134–141, 1984.

[43] J. C. Mitchell, V. Shmatikov, and U. Stern. Finite-state analysis of SSL 3.0 and related protocols. In *DIMACS*

Workshop on Design and Formal Verification of Security Protocols, Sept. 1997. `http://dimacs.rutgers.edu/Workshops/Security/`.

[44] A. Nadalin, P. Griffin, C. Kaler, P. Hallam-Baker, and R. Monzillo. Web Services Security UsernameToken profile 1.0, Mar. 2004.

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

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

[47] L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Cryptology*, 6(1):85–128, 1998.

[48] L. C. Paulson. Inductive analysis of the internet protocol TLS. *ACM Transactions on Information and System Security*, 2(3):332–351, 1999.

[49] V. Shoup. On formal models for secure key exchange. Research Report RZ 3120 (#93166), IBM Research, Apr. 1999. Version 4, November 1999, available from `http://www.shoup.net/papers/`.

[50] P. Syverson. Limitations on design principles for public key protocols. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 62–72, Oakland, CA, May 1996. IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Society Press.

[51] F. J. Thayer Fabrega, J. C. Herzog, and J. D. Guttman. Strand spaces: Why is a security protocol correct? In *Proc. 19th IEEE Symposium on Security & Privacy*, pages 160–171, 1998.

[52] D. Wagner and B. Schneier. Analysis of the SSL 3.0 protocol. In *Proc. 2nd USENIX Workshop on Electronic Commerce*, pages 29–40, 1996.