An overview of Web Services security

P Kearney, J Chapman, N Edwards, M Gifford and L He

Security and Web Services are consistently reported among the top technologies of interest to businesses. Concerns about security are a major deterrent to companies considering use of the technology. This paper attempts to give an overview of the current state of Web Services security. The main body of the paper is a tour through key concepts used in Web Services security. Examples based on software demonstrators built by the authors are used to explain how the ideas are used in combination to achieve particular aims. The state of play as regards standards is also reviewed. The concluding section gives some pointers as to active research topics.

1. Introduction

Security and Web Services are consistently reported among the top technologies of interest to businesses[1], for very good reasons. Companies are looking to Web Services and other XML-based standards and technologies as a means of automating the interaction of their customers and business partners with their business processes. For established businesses[1], attractions include the prospect of reducing costs and delays, and the ability to reach new customers and partners. In addition, the new technology and standards make viable novel business models, particularly by lowering the barriers to market entry for small enterprises. On the other hand, security, or rather the lack of it — real or perceived — is seen as a major deterrent to companies seeking these benefits. Similarly, fear of fraud and the lack of confidence in the ability of companies to protect personal information is a major barrier to acceptance of electronic modes of transaction.

Making applications accessible to other parties over the Internet raises significant security issues, especially when these applications are business critical. The Internet is already perceived as a dangerous place, and the very attributes of Web Services that make them attractive also necessarily introduce new potential vulnerabilities. By making access easier for legitimate users and systems from outside the enterprise boundary, new possibilities for malicious and felonious entities to exploit are also brought about. For example, it is accepted good practice to install firewalls at the enterprise boundary to filter out dangerous types of traffic. Sending text packets over HTTP (e.g. for standard browsing of Web pages) is regarded as safe, and firewalls are generally configured to let such traffic through — after all, what harm can text do? This is the normal channel via which Web Services messages are sent, at least partly because firewalls are configured to let it through. The fact that firewalls do not get in the way of Web Services is often billed as an advantage in sales literature[2]. However, whereas a human-readable Web page poses few dangers, a SOAP message[3] is designed to trigger some activity in the system receiving it, and this is open to abuse as well as legitimate use[4].

For a general introduction to Web Services, the reader is referred to Cerami [1], and to other papers in the current volume. This paper attempts to give an overview of the current state of Web Services security. Readers who have experience of security in distributed systems will find here some familiar ideas (encryption, digital signatures, tokens, trusted third parties, etc), as many of the concepts used in securing Web Services have been arrived at by abstracting and generalising from tried-and-tested approaches.

The coming challenge is to apply them in appropriate combinations to provide security for open heterogeneous distributed applications, the various elements of which are owned and managed by independent entities.

[1] A survey in Computing (9 January 2003) reported that 95% of respondents rated security as being very or fairly important to their organisations over the next 12 months. The figure for Web Services was 81%. These were the top two categories.

[2] The problems of letting information and control pass through a firewall have hindered the adoption of previous integration technologies such as CORBA.

[3] SOAP is the specification defining the message model underpinning Web Services.

[4] The same is, of course, true for dynamically generated Web pages. In fact, one can consider Web Services as a more elegant and more general way of achieving what has been previously done via CGI scripts, etc, and commands disguised as URLs.

The theme running through this paper is the need to think of Web Services applications in terms of three levels:

- hosts/connections,
- messages/Web Services/operations,
- people and organisations/message content/ business transactions.

This model will be introduced in the next section. The main body of the paper is a tour through key concepts used in Web Services security. Examples based on software demonstrators built by the authors are used to explain how the ideas are used in combination to achieve particular aims. The state of play as regards standards is then reviewed. The concluding section gives some pointers as to active research topics.

2. Web Services as middleware

Figure 1 shows a schematic representation of a typical business-to-business (B2B) application of Web Services. At various points in their respective business processes, the business application software of company 1 and company 2 need to exchange messages. These messages can be thought of as requests that some operation be performed, or as replies to such requests. For example, company 1 might place an order for Widget X. In this case the operation requested might be called `placeOrder`; the `placeOrderRequest` message would need to pass a purchase order detailing the type of widget being ordered; and the `placeOrderResponse` message would carry a confirmation that the order had been accepted.

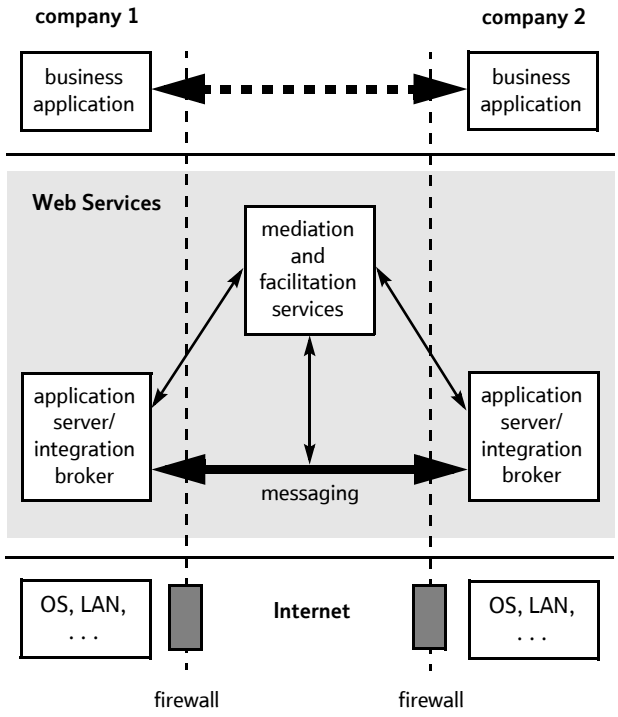

Fig 1 A three-level view of Web Services.

In a realistic application, such an exchange would be embedded in a more complex extended dialogue between the companies. This might start with a query stating general requirements, to which company 2 responds with a list of relevant widgets. This would be followed by message exchanges concerned with selecting the specific widget and agreeing terms. The order is then placed as above, and subsequently fulfilled. Company 2 is then billed and payment collected.

To enable this interaction between the business applications, each company uses an application server to deploy an interface to its application as a Web Service. A Web Service is basically a collection of related operations, with each operation (e.g. `placeOrder`) being associated with a message or a pair of messages (`placeOrderRequest`, `placeOrderResponse`). We can thus talk of a Web Service, WS1, executing on server S1 owned by company C1, performing an operation `placeOrderRequest` in response to a request on behalf of company C2 sent by WS2 executing on S2.

The application servers (and analogous platforms and/or APIs) together with means of conveying SOAP messages between them, and various mediation and facilitation services (directories, etc), constitute a middleware layer that insulates the applications from the need to know specific details of the underlying network topology. The triangular relationship in the centre of Fig 1 is characteristic of so-called service-oriented architectures (SOAs). The SOA triangle is often drawn with client, server and mediation/facilitation service nodes at the apices, but it should be noted that whether a Web Service plays a client or server role at a given time depends on the message exchange that is being considered[5].

It is important to remember that business Web Services operations (such as shown in Fig 1) are executed to enact transactions between companies (or other legal entities). Messages could be interpreted as being a legally binding commitment on behalf of company 1 to buy the items described in the purchase authority passed as an argument. Thus, as well as the network of Web Services-enabled applications linked by operations invoked by messages, we must also consider the network of 'real-world' entities linked by business and personal relationships, between whom the business transactions actually take place. Triangular relation-

[5] The terminology 'Web Service' is misleading in this respect.

ships are useful here too. For example, two parties who have not done business with each other before might be unwilling to engage in a transaction without the participation of a mutually trusted third party.

The distinctions and interactions between these layers are discussed below, with software demonstrators produced in the course of the 'Secure Web Services' project providing an illustrative context.

3. The entity with the identity

The concept of identity is central to Web Services security. An identity is basically a collection of attributes relating to some 'real world' entity, linked to a unique symbol (the identifier). It is possible for an entity to have many identities, each of which may be thought of as the *persona* the entity adopts in different contexts. An account with an on-line retailer is an example of an identity.

It is useful in the context of this paper to distinguish at least three types of entity, which will be referred to as:

- agent — organisation (e.g. company 1, 2), person (e.g. company 1, 2's employee), and software agent acting with their authority,
- Web application — Web Services node acting as client, server or intermediate node (Web browsers and Web sites can also be grouped in this category),
- host/platform providing the execution environment for the Web applications.

It is the first category of entity that really matters: it is people and organisations that engage in electronic transactions. However, they must interact electronically via the Web applications, which in turn make use of services provided by the hosts.

Much of security is concerned with:

- confirming a claim that two identities (or references to identities) refer to the same entity (authentication), for example, confirming the truth of a claim that 'the sender of this message' is 'Fred Bloggs',
- confirming a claimed entitlement of one entity to perform some action on another (authorisation), for example that Fred Bloggs is entitled to receive the service requested in a given message,
- enforcing the entitlements in both positive and negative senses, i.e. making sure entities can get what they are entitled to, and that they cannot get what they are not entitled to — the latter includes ensuring that entities cannot fool the system by pretending to be someone else, or find loopholes around enforcement mechanisms, while the former includes preventing one entity doing something that denies another access to something it is entitled to,
- reasoning about, communicating about and recording the above.

Later sections of the paper will return to these topics, but first we need to give an overview of some basic building blocks.

4. Encryption, signatures and certificates

Cryptographic techniques may be used in a variety of ways to provide security. This section gives a brief summary for the lay reader. For a more detailed exposition, see Selkirk [2, 3] and Bosworth and Tedeschi [4].

To encrypt a document[6], one applies a reversible mathematical function to render it unreadable to anyone who does not know the inverse function used to decrypt it. Both functions require a parameter known as a key to be specified. The parameterised functions are designed so that an encryption or decryption operation cannot be replicated without knowing the function and the key. Normally the parameterised functions are public knowledge, but the keys are only revealed to those permitted to perform the operation.

Cryptography comes in two main flavours, depending on whether the keys used for encryption and decryption are the same (symmetric) or different (asymmetric). The asymmetric flavour is known as public key cryptography, as one key is normally kept secret, and the other freely distributed. In the most common public key systems, the parameterised function used for both encryption and decryption is the same; the result of applying it twice using complementary keys is equivalent to applying no operation at all. This is analogous to rotating a shape twice, with the two angles adding up to 360°. This means that a public key can be used to encrypt something to be decrypted using the corresponding private key, or to decrypt something that has been encrypted using the corresponding private key.

Digital cryptography is used for three main purposes:

[6] In this paper, the term 'document' is used very generally to refer to any chunk of information regardless of media type.

- to maintain confidentiality by preventing those not in possession of a secret/private key from gaining access to information,
- to provide a means of authentication through proof of possession of a secret/private key, e.g. a known string could be encrypted by A using its private key and sent to B who decrypts it using A's public key, successfully recovering the string — B now knows it really is communicating with A, assuming A has not revealed its private key to anyone,
- to provide a means of verifying the integrity of a piece of information (i.e. confirming it has not been modified accidentally or by some unauthorised agency) — it is usually the case with cryptographic functions, that a small change to the plain text results in a large change in the encrypted text, and vice versa, and it is this property that means any modification of an encrypted document is apparent when the document is decrypted.

The latter two points are combined in a digital signature. A signature is an encrypted digest of a document. A digest is a string generated from the document using a public 'one-way function', such that it is easy to recreate the digest if you know the message, but very difficult to regenerate the message from the digest. It provides a means of detecting whether the document has been altered. Suppose the signature (encrypted with the sender's private key) is sent along with the (unencrypted) document to a recipient. The receiving party can check the signature by decrypting it using the signer's public key, and comparing the result with the digest generated directly from the message. A match between the two confirms that the message has not been changed in transit, and also authenticates the signer (otherwise the signer's public key would not have decrypted the digest correctly).

This discussion begs the question of how the recipient of a signed document knows reliably it is using the correct key to check the signature. The usual answer is that the public key has been obtained from an X.509 certificate. A certificate is basically a digitally signed document, containing the public key and information about the identity with which it is associated. It is interpreted as an assertion by the signatory in question that the key is the public half of a pair owned by the entity identified. The certificate can be 'self-signed', in which case it offers little assurance about the 'true' identity of its owner, but can still be useful in checking that one is still 'talking to the same person as before'. Alternatively it can be signed by an 'authority'. In this case, the confidence in the identity information given in the certificate depends on the trust that can be placed in the authority and the processes that authority uses to validate information before issuing a certificate. A commercial example of this is the Certification Authority services offered by VeriSign attesting to the identities of thousands of eBusiness servers.

5. Communications security

Communications security is concerned with protecting a communications channel between two end-points, and the information flowing down it. It is convenient (and conventional) to consider this separately from what happens at the end-points (access control, etc).

Once more, it is useful to think in terms of three layers (see Fig 2). To clarify what these are, let us consider a human-oriented metaphor:

- a manager dictates a letter to a secretary,
- the secretary seals the letter in an envelope, and takes it to the post room,
- a chain of messengers delivers the sealed envelope to the post room of the destination company, where the mail is sorted and the letter delivered to an office post point,
- a secretary or clerk opens the mail, and passes the letter to the relevant manager.

What really matters is that the communication is delivered faithfully end-to-end from manager to manager. In some applications it is also important that unauthorised people are unable to discover the content of the letter, or even to know that the letter was sent. What measures are taken, depends on the importance of the communication and the degree of trust that can be placed in the various intermediaries (secretaries, messengers, etc) involved.

The parallel with B2B Web Services communications is quite close. A business communication (e.g. a purchase order) originating from a person or automated process is expressed in an XML-based language, enclosed in a SOAP message, and sent via a transport mechanism to a destination server. There, the SOAP message is routed to the appropriate Web Service, and the content passed 'up the stack' to a business software component for interpretation and action.

The cryptographic techniques outlined earlier can be applied within each of these layers to provide confidentiality, integrity and confirm the identity of the relevant communication end-points. It should be noted, however, that what is regarded as end-to-end security in a lower layer, may cover only segments of a communication path at a higher layer. Similarly, measures taken at a lower level tend to be less discriminating, protecting flows of packets rather than

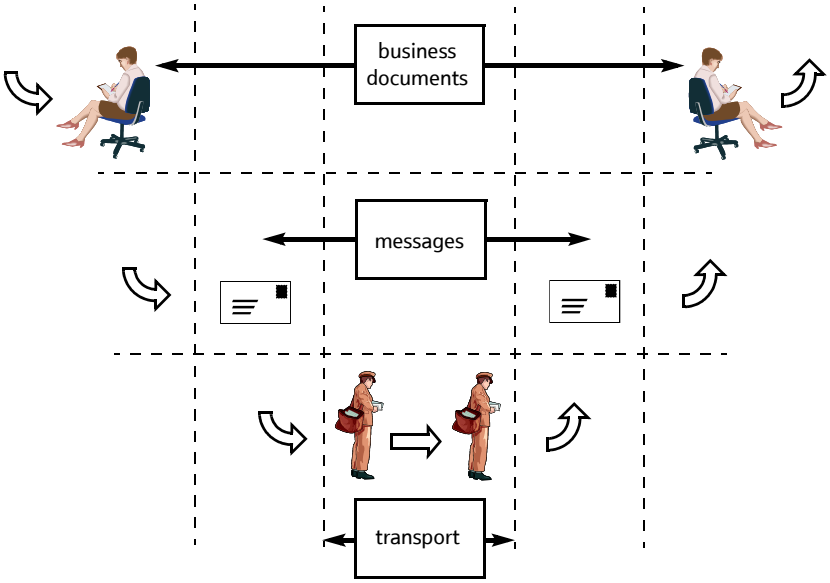

Fig 2 Three layers of communication.

messages, for example. Thus, while traditional communications security measures, such as packet-based virtual private networks [5] and transport level security, are valuable in the context of Web Services, they do not provide a complete answer. This is explored in more detail below.

5.1 Transport layer security

A common protective measure is to send messages over a secure connection, e.g. using SSL or TLS. These protocols use public key techniques to authenticate one or both end-points and agree a (symmetric) secret key, which is then used to encrypt packets flowing over the connection, hiding the communication contents from third parties and stopping them interfering with communications without detection. If the connection is used to provide a secure tunnel between two domains that trust each other this provides a good level of protection.

However, SSL is a point-to-point technology operating on the lowest of the three levels discussed in this paper (host-to-host). It provides the originating (client) host with assurances about the identity of the destination (server) host[7], and of confidentiality and integrity during transmission between them. However, it conveys nothing about any processing done at either end of the connection. If, say, the server does not trust the client, then being sure of the client's identity provides little comfort. Furthermore, if any processing is required at the message envelope or content level during communication, the SSL connection must be terminated. Of course, a new onward connection can be established, but then the two end-point hosts have lost sight of each other.

[7]And *vice versa* if client-side certificates are used.

This also applies to processing by firewalls. Since SSL packets are encrypted, an application-level firewall cannot examine their contents to make decisions on the safety of allowing them through. Consequently, either SSL connections must be terminated at (or before) a firewall, or the firewall must be configured to allow the connection through with the content unchecked.

It is important to note that the SOAP specifications clearly envisage the possibility of SOAP networks rather than just point-to-point connections. In addition to client and server nodes, the specifications define intermediate nodes, which can, for example, perform router-like functions. Applications that can understand SOAP messages in the sense defined in the specification are termed SOAP nodes. In the context of a particular message, a node may act in three main roles — SOAP sender, SOAP receiver, or SOAP intermediary (which combines sender and receiver roles). Thus, it is quite possible that the applications at either end of the SSL connection are acting as forwarding nodes for messages. While transport layer security measures such as SSL may play a useful role in protecting point-to-point connections within a SOAP network, they do not in the general case, provide an end-to-end solution.

Returning to the metaphor, use of transport layer security is like employing a security company to deliver letters between company sites in armoured vans. Protection of the letters while in the van is strong enough for most purposes, but the letters must be taken out of the van to sort, forward and deliver them. If all you are concerned with is connecting two trusted sites, there is no problem. However, the larger each of those sites is, and the greater the number of possible onward connections, the less information you have

about the origin and ultimate destination of a given message. Thus, while SSL may well be sufficient to secure today's limited Web Services applications, the more complex the network of interconnected Web Services becomes, the greater the need for a network (as opposed to point-to-point) solution at the message (as opposed to transport) level.

5.2 *Message level security*

Except in relatively simple situations, then, security also needs to be addressed at the message level. The basic building blocks for this are already established. Digital signatures and encryption can be used to protect the message (or parts of it) while in transit from source node to destination node.

Encrypting a message with the public key of the intended recipient means that the message can only be decrypted by an entity with access to the recipient's private key, providing confidentiality and confirming the identity of the destination. Digital signing of the message using the sender's private key confirms the identity of the sending node and protects message integrity, as explained in section 4.

The public key could already be known to the recipient, but an alternative approach is to send a digital certificate as a security token in the message header (see section 8 for more details). The certificate could also be obtained from the trusted authority that issued it or from another third party. The signature ties the message body to a public key, and the certificate ties the public key to an identity. Thus, the receiver of the message has a chain of evidence linking the message body to the individual or company described in the certificate.

The XML signature and XML encryption specifications [6] provide standard methods for digitally signing and encrypting XML documents (including SOAP messages). Not only can whole documents be signed/encrypted, but also individual parts. The WS-Security specification [7] 'leverages' these standards, providing a SOAP message header section where security-related information may be held. This includes information about the signed and encrypted parts of the message as defined by XML signature and XML encryption specifications, digital certificates and other 'tokens'. A more general overview of the situation regarding Web Services security standards is given in section 10.

Digitally signing a SOAP message is like sealing a letter in a transparent envelope using a distinctive, forgery-proof sticky label — it does not stop anyone reading the letter, but the recipient can be sure that no-one has tampered with it in transit. Furthermore, it provides evidence of who sealed the envelope/where it was sealed. Encrypting the message using the recipient's public key makes the envelope opaque, and also ensures that it can only be opened by the intended recipient (or rather, an entity in possession of the recipient's key). It was noted above that SOAP is based on a network model of communications, with intermediate nodes, e.g. performing message routing actions[8]. The intermediate nodes may need to see parts of the header and/or body of the message in order to perform their roles, and may also need to modify the message (e.g. by adding or amending header fields). Thus, the sending node must take care not to interfere with legitimate processing by intermediate nodes by encrypting or signing parts of the message that must be read or changed in transit.

Provided this is respected, encryption and signatures can be used to supply end-to-end message security between source and destination nodes.

5.3 *Message content*

The same techniques and standards can be used to provide confidentiality, integrity and attribution assurances regarding the message content. Care must be taken, though, in interpreting the semantic significance of a digital signature. All one can be sure of is that the signed content was in the possession of the holder of the given private key, and that it has not been changed since. Any implication that the certified owner of the key, originated, approves of, or is bound by commitments expressed in the signed content would depend on application-specific conventions[9].

In a similar vein, digital signatures and encryption protect 'bits' not information. In many Web Services applications, intermediate nodes will process message content, i.e. the information content will be modified, e.g. by a 'prime contractor' Web Service combining and adding value to input from other Web Services.

Assuring end-to-end integrity of the information content may well involve use of signatures and encryption, but this must be combined with process information and trust in the participants involved in providing the service. Such matters are still very much topics for research.

[8] SOAP does not define roles played by intermediate nodes, but provides hooks for such roles to be defined by extension specifications. One such extension is WS-Routing, which defines header fields concerned with constraining/recording the path taken by a message.

[9] In 1999, the European Commission published a directive to provide a Community framework for electronic signatures, focusing on their legal validity. In response, the European ICT Standards Board, launched an initiative bringing together industry and public authorities, experts and other market players — the European Electronic Signature Standardisation Initiative (EESSI) [8].

6. Examples applying digital signatures

Figure 3 shows a typical present-day Web Services application scenario that we have used for one of our demonstrators. Here, MusicVision is an on-line retailer making use of three 'back-end' Web Services from different suppliers — a payment service, a wholesaler, and MusicComment (an on-line repository of reviews). The end user interacts with MusicVision via a browser-based thin client. In our demonstration, subscribers identify and authenticate themselves to MusicVision via user name and password. Messages from MusicVision to MusicComment (sending new comments entered by users) are digitally signed using MusicVision's private key. The user providing the comments is identified in the message via a unique identifier.

The three layers are quite clear here:

- agents — the user (John), the MusicVision company, and the various supplier companies,
- Web applications — the pages displayed by John's browser, the MusicVision Web site/service, and the supplier Web Services,
- hosts — John's browser and PC and the servers/platforms on which the Web sites and services are running.

Correspondingly, there are three layers of communication:

- John asserts his opinion about a piece of music to be shared with the MusicVision/MusicComment user community,
- John enters some text on a comment submission form and sends the form to the MusicVision Web site — the MusicVision Web Service then sends an 'addReview' request to the MusicComment Web Service, enclosing the text and associating it with John's identifier,
- the above layers make use of communications channels (sockets, or at a more abstract level, sessions), which are established between the browser and the MusicVision server, and between the MusicVision server and MusicComment server.

The user name and password effectively associate John with a session connecting his browser with the

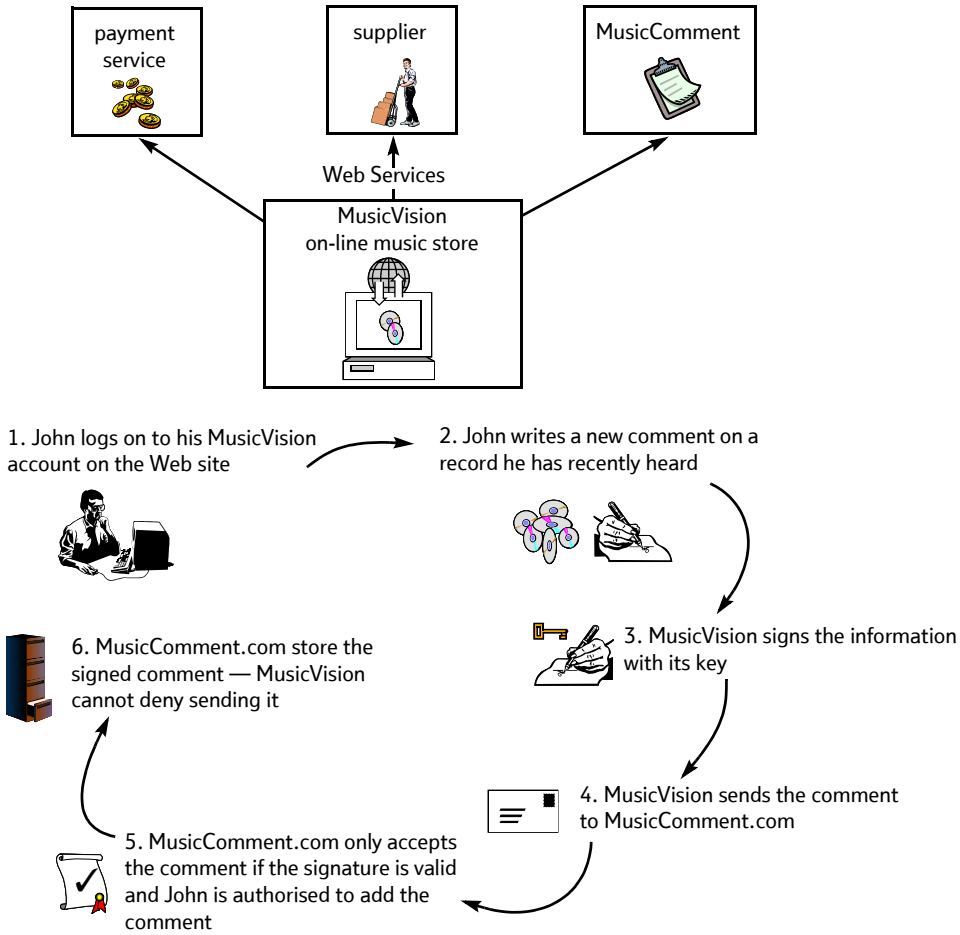

Fig 3 An example broker Web Service.

Web site. Although a user name plus password is a weak identity credential, the password provides at least some evidence the person at the browser really is John if measures are taken to prevent caching of passwords. Using secure HTTP with client-side authentication (also available in this demonstrator) provides much stronger assurance to the Web server that it is connected to John's browser. However, since the private cryptographic key used is installed in the browser/PC file system, it does not provide any extra assurance that John is physically present[10].

The MusicVision Web Service makes the assumption that John's association with the session implies that he is the one posting the review, and that it is his comments that the review expresses. MusicVision has issued the identity used. This links John with the username, password and an identifier. The identifier is unique within the scope of MusicVision's identity domain, and so can be considered globally unique when combined with a unique identifier for the domain. MusicVision takes responsibility for confirming that John is the entity 'owning' the identity claimed via the user name entered (authentication). In this demonstrator, the identifier is 'opaque', i.e. John's true identity cannot be inferred from the identifier alone. Thus, to MusicComment, reviews are anonymous. The identifier gives some assurance that a real individual originated the review, enables different reviews by the same identity to be connected, and potentially provides an audit trail, e.g. in the event of legal action regarding defamatory statements. MusicVision's signature of the messages to MusicComment implies (in the context of this application) its confirmation of the association of the identity with the review.

The above analysis contains many implied connections and application/context-specific assumptions. This is quite typical of the state of the art: the standard building blocks for providing trust and security are available (or becoming so), but the semantic interpretation that defines their significance is decided on an *ad hoc* basis. This is bound to lead to problems as the scale, openness and 'dynamicness'[11] of Web Services networks grow. Let us consider another demonstrator developed within the Secure Web Services project, which adopts a different approach but uses similar ingredients.

The Secure Document Storage Web Services demonstrator implements a shared document repository whereby authors may make their work available to a reader community (see Fig 4). In principle, these documents could be of any media type. Digital signatures are used to associate electronic documents with their authors reliably. The repository has a Web Services wrapper so that a variety of client applications (automated and human-driven) may interact with it. Prospective authors register with the Web Service and are issued with public-private key pairs. The private keys are issued to authors by some secure means. The public keys are stored along with digital certificates (which relate the public key to the identity of its owner). Authors must digitally sign the documents before submitting them via their chosen client application.

When the repository Web Service receives a request to store a document, it checks the digital signature on the document using the stored public key of the claimed author. If these match, the Web Service countersigns the document using its own private key and adds it to the repository.

Readers retrieving documents from the store can be assured of their integrity and authenticity by checking either the server's or author's signature on the document against the relevant public key.

There are a number of contrasts to the approach taken in the MusicVision example.

- Authenticity

  The authenticated association is that between the document and its author. The document is communicated as part of the message 'payload', and there is a degree of implication that the sender of the message is the author. The identity of the message sender is not critical to the application, though, as it is the authenticity of the document that matters. This is the opposite way round to MusicVision where the authenticated association is with the session or the message as a whole. It would be possible to combine approaches, e.g. by having the sender sign the message and the author sign the document. Even so, going from factual evidence (that a document/message was signed by an entity in possession of a private key certified as belonging to a particular identity) to conclusions about authorship and requesting identity involves making assumptions or *ad hoc* conventions.

- Identity

  The identity provider is now the 'back-end' Web Service rather than the user-facing client. Note that creation of the key pair is logically distinct from managing the identity *per se*. The pair is generated then the public key is bound to the identity via a certificate. The author could have used an

[10] Of course, biometrics, smartcard/USB token-based keystores, or SecureID tokens would provide stronger evidence than passwords.

[11] In the future we can expect it to be normal for chains of Web Services to be built on the fly to provide personalised and contextualised solutions to a user's requirements.

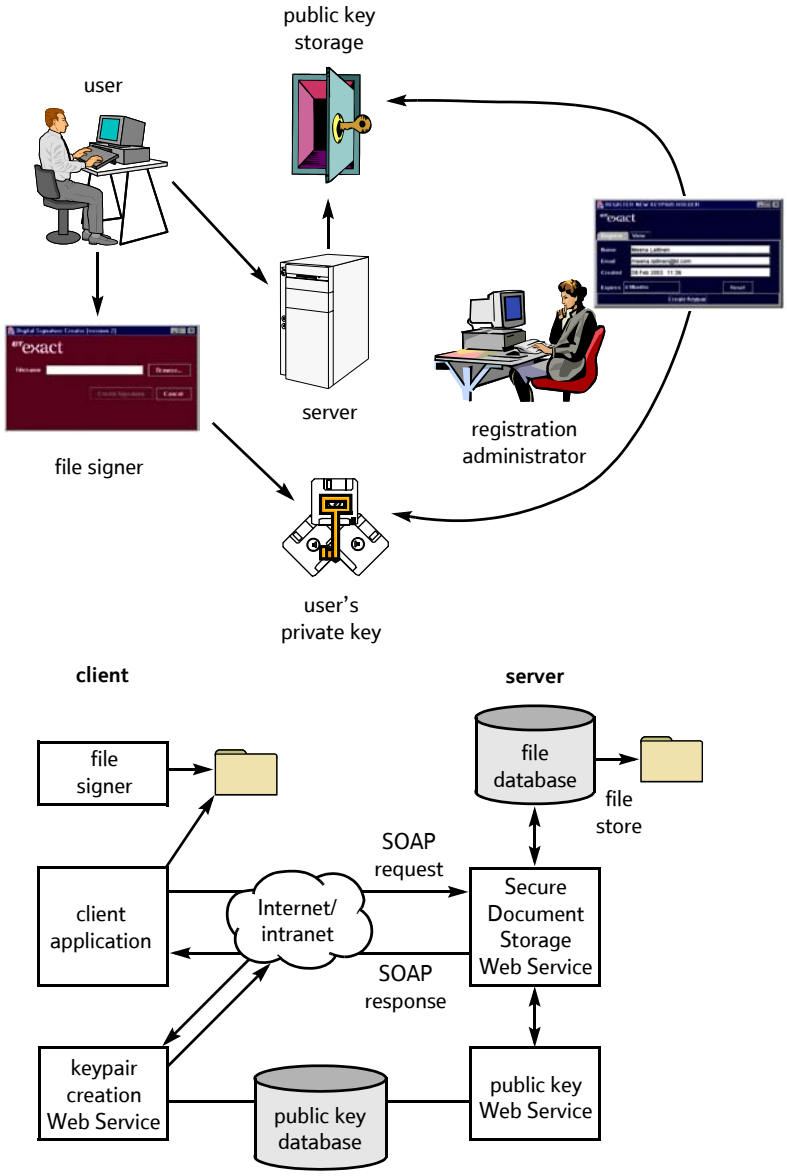

Fig 4 Secure Document Storage Web Service (SDSWS).

application on his/her PC to generate the keys then registered the public key with the identity provider as an attribute of the identity. This process would itself need to be secure and involve verification of the identity of the person updating the public key. There is also no reason why multiple identity providers should not be active with a Web Services network, issuing multiple identities to the same entities. The Liberty Alliance specification, for example [9], describes ways by which identity providers can know they are talking about the same entity without giving away information.

7. Policy-based access control

So far, we have dealt with protection of messages in transit. This section considers the security measures that are taken when the message arrives at its destination site, in terms of a series of tests that must be passed before the message is allowed through to the destination Web Service for processing, and the complementary processes at the sender's end.

In an ideal world, configuration, operation and management of security functionality should not require specialist skills. Such tasks should be performable by staff involved in business operations rather than technical experts. For this to happen, the person responsible must be able to lay down 'rules' that are meaningful both to a human being and to the Web Services management software. Such rules are known as policies. Loosely speaking, in this context a policy is a statement about rights or responsibilities. For example a policy might lay down circumstances in which a request might be granted to access a resource. The

familiar file access rules implemented by most operating systems giving users or groups read and write access to particular files can be seen as elementary examples of policies. A more sophisticated example might be along the lines of:

> *Grant X's request to do A if X is an employee of organisation Y and Y has an agreement B with us that lists A as a right applicable to all employees.*

Policies are not restricted to file access permissions. We might for example have a policy that lays down under what circumstances the identity of the sender of a message must be verified using a digital signature. A language for expressing policies needs to have a mathematical formality so that the set of policies can be processed automatically, but also be close enough to natural language that a non-expert can both write and understand the rules. This is not a trivial combination to achieve.

Policies are interpreted in a given context by so-called policy decision points (PDPs). The resulting decisions are applied at policy enforcement points (PEPs). Separation of decision and enforcement is important from a security point of view. A PEP is like a stereotypical doorman at a nightclub — strong, but not paid to think. It applies very simple rules — does the clubber have a membership card? — any real decisions are referred to the PDP.

Two important classes of check are authentication checks and authorisation checks. Authentication checks verify identity of the originator of a request as laid down by the policies for this type of message and circumstance (*Doorman 1 checks that the partygoer has a valid photo-id*). Authorisation is the process of determining whether the requester is entitled to make that request (*Doorman 2 checks that the partygoer's name is on the guest list*). Note that authorisation at this point does not necessarily mean that the request contained in the message should be granted, but only that the message is allowed onward for processing. Further policy-based authorisation decisions may be made as part of the internal processing of the request. Authentication and authorisation are not the only types of operation that can usefully be performed. Additional steps that could be added to the chain include: logging for non-repudiation purposes, and checking for syntactic compliance (e.g. confirming the message and its parts are valid XML and match XML document types that are known to and understood by the relevant Web Service).

The checks and other operations may apply to all levels of abstraction. The following are types of authentication/authorisation check applicable to successively higher levels:

- confirm that the message was delivered via a trusted channel, e.g. by checking that the message arrived via an SSL connection from a trusted node,
- confirm the message was from a trusted source, e.g. by checking that the message body was digitally signed using the private key of a trusted Web Service,
- confirm the identity of the requester/responder and that it is allowed to send a message of this type.

As a general rule it is sensible to perform the check on the lower levels first.

To help visualise this process, imagine a messenger arriving at the offices of KafkaCo, a rather bureaucratic company. The receptionist must decide whether to accept the letter from the messenger and pass it to the first in a series of clerks for processing. This clerk applies a first test then either rejects the letter, or initials it and passes it on to a send clerk, and so on. The receptionist and clerks are PEPs. They apply simple specialised mechanistic rules given them by their managers. If something outside the scope of their rules arises they can refer to their managers (the PDPs) for a judgement. In turn, the managers can refer to the great book of policy on their shelves.

In order for a message to pass many of these checks at the receiving end, it will need to contain certain information, be formatted in a certain way, etc. Consequently, a complementary processing chain is required at the transmitting end to ensure the requirements are met. It is wise to keep the modules applying the transformations, checks, etc, separate from that implementing the application functionality so that the two can be used independently — a Web Services consumer, for example, may obtain similar services from different providers, complying with different security requirements in each case. This leads to the adoption of an architecture such as that shown in Fig 5. Note that in general, each company must be allowed to set its own policies. However, to ensure compatibility and (more generally) interoperability, the processing of inbound and outbound request and response messages by each company must be complementary. This is indicated schematically by the double-headed arrow in Fig 5. Consequently there needs to be a degree of alignment of policies, either by convention/standardisation, prior agreement, or negotiation 'on the fly'.

8. Tokens and security token services

In the previous section, reference was made to security-related information being added to a message as it leaves the sender, and 'consumed' at the server end as

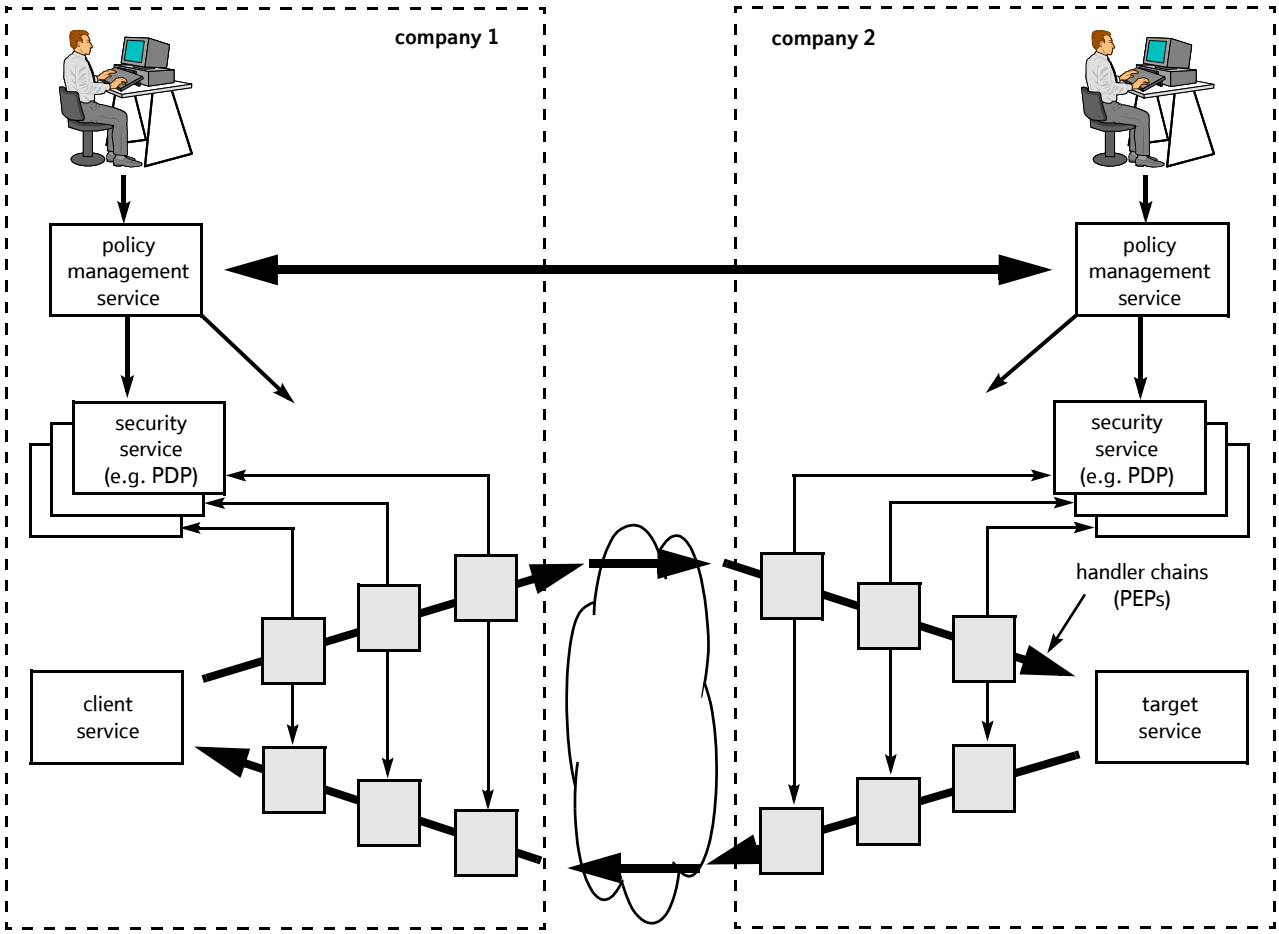

Fig 5 An architecture for policy-based access control.

part of the access control process. One can think of this in terms of the sender making assertions or 'claims' about itself to persuade the receiver to accept the message, or to perform the action that the message requests. A piece of XML expressing one or more claims and accompanying a message (usually as part of the SOAP header), is generally known as a token, and is an important concept in Web Services security. The WS-Security specification, mentioned earlier, defines the notion of a security token and provides a SOAP header element within which to attach tokens to messages.

Tokens may express claims the sender makes about itself, in which case (to be convincing) they need to contain evidence of the veracity of the claim (or be linked in some way to it). An example is a token containing a username and password, which makes a claim as to the identity of the sender and provides the evidence to support it. One weakness (among many) in this example is that sender and receiver must share a 'secret', the password. If the username/password combination has general validity across a number of service providers, this means that an unscrupulous provider could obtain the password and use it to masquerade as the user to other service providers.

An alternative approach is for the claimant to present evidence to a third party trusted by both principals, and for this third party to issue a token expressing its support for the claim. An additional advantage is that the token may be reused subject to validity conditions, e.g. an expiry date.

SAML [10], a specification with increasing support in the market-place, identifies three types of security assertion[12]:

- authentication assertions describe checks that have been done to confirm the identity of the subject of the assertion,
- attribute assertions attest to the values of one or more attributes of the subject,
- authorisation assertions attest to which services the subject is or is not entitled.

As well as XML document types for expressing these assertions, SAML also defines simple request-response

[12] This does not mean these are the only three types, they are merely the ones defined by SAML.

message pairs for obtaining these assertions from authorities.

Note that the step whereby identification credentials are presented to the authentication authority is outside the scope of SAML. Let us suppose a requester has already done this. It can then ask the authentication service for a token stating the checks that had been performed to verify its identity, and attach it to the request for a service. The service provider can compare the checks and the identity of the service issuing the assertions against its policy to see if the token is acceptable as proof of identity. Alternatively, the request could have been sent with a token identifying the authentication service that had been used, and the service provider could have requested the assertion from the authentication service. Similar possibilities apply to authorisation.

9. An example using authentication tokens

This section describes a third software demonstrator produced within the project. In this scenario, the user is represented by a 'user agent' — a trusted personal proxy mediating the user's interactions with Web Services. The other entities in the scenario are an identity provider Web Service and a number of ordinary Web Services owned by different providers. The user agent trusts the identity provider not to misuse potentially sensitive credentials, and the service providers trust the tokens it issues. However, the user agent does not trust the service providers, and so will not provide them with tokens or other credentials they could then use to masquerade as the user.

The solution adopted here is reminiscent of Kerberos, in that two types of token are issued authenticating the user (see Fig 6). The first is issued to the user agent by the identity provider in return for presentation of credentials. This token should only be used in communication to the identity provider. The second type is specific to a particular service provider. It can safely be sent to that service provider, which cannot then use it to misrepresent itself to other providers.

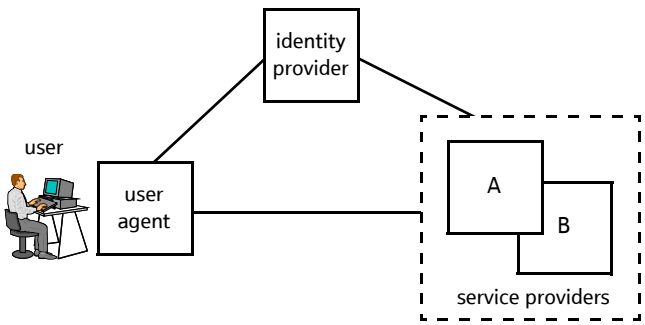

Fig 6 Architecture of the authentication token demonstrator.

The general sequence of events is as follows:

- the user agent obtains an identity token from the identity provider (this step does not need to be repeated until the token expires),
- the user agent obtains a token from the service provider, which will identify the service provider uniquely to the identity provider,
- the user agent requests and receives a service-specific identity token from the identity provider — it includes the above two tokens in the request,
- the user agent requests the service from the service provider, enclosing the service-specific token as proof of identity.

Public key cryptography is used to protect the contents of the tokens.

It should be noted that Fig 6 includes a line between the service providers and the identity provider, which was not used in the above sequence. It allows for the possibility, for example, that the service provider could inform the identity provider in advance of its policy regarding required means of confirming identity, or that it could enquire about the checks that had been performed to issue the token.

10. Standards and interoperability

There are several standards bodies active in this space, notably W3C (see section 10.2) and OASIS (see section 10.3), as well as various organisations formed, for example, to establish interoperability guidelines in particular areas, and consortia and individual companies formulating and proposing specifications. Typically, the process is driven 'bottom up' by companies/consortia publishing specifications. These may be submitted to standards bodies for development and ratification, in which case a technical committee is formed to handle this process. 'Bindings' are often published to accompany specifications to show how the abstract specifications can be applied to achieve a particular purpose or implemented using a particular technology. These are often illustrative rather than prescriptive, however. As a specification becomes more popular and is taken up by vendors and end users, interoperability guidelines are needed to ensure the specifications are interpreted in a consistent way in given application and technical contexts. Ultimately, whether a specification merits the label 'standard' depends on its acceptance and take up by software vendors and end-user organisations.

The situation regarding standards for Web Services, including those relevant to security, is very complex and fluid. This is partly as a result of the rather 'Darwinian'

process by which a consensus emerges around a specification. Another factor is the need to combine short-term expediency — enabling vendors and end users to implement simple, application/platform-specific solutions now, while providing a stable foundation for more complex and generic interoperability solutions in the future. This section attempts to map out some of the more significant initiatives.

10.1 The 'WS-X' series of specifications

The label 'WS-X' is used here to refer to the series of specifications (with names beginning 'WS-') being developed by Microsoft, IBM and others as part of the Global XML Web Services Architecture (GXA [11]) platform. IBM and Microsoft have published a white paper [12] outlining their vision for development of the security elements of the WS-X series of specifications. It lists seven specifications organised in three levels. WS-Security (mentioned earlier in this paper) provides the foundation layer, and is now reasonably well established. Despite its name, WS-Security does not claim to provide a complete solution to securing Web Services. It is a building block that is intended to be used in conjunction with other Web Services- and application-specific protocols to accommodate a wide variety of security models. OASIS has established the Web Services Security technical committee to continue its development as a standard.

A batch of additional specifications [13] on the roadmap (WS-Trust, WS-SecurityPolicy and WS-SecureConversation) were published in December 2002, with WS-Federation added in July 2003. Also listed in the roadmap, but as yet unpublished, are WS-Privacy and WS-Authorisation. These specifications seem to present a coherent set of basic concept definitions abstracted from the emerging view of best practice in distributed systems security, and proposals for representing these concepts in XML. They are still relatively immature, and it remains to be seen to what extent they are taken up in practice.

Even when further fleshed out these specifications will be important building blocks rather than giving a complete picture of Web Services security. Some important 'research frontiers' are mentioned later.

10.2 W3C (World-Wide Web Consortium) activities

W3C [14] tends to provide a home for several of the fundamental Web Services standards, including SOAP and WSDL, whereas OASIS tends to sponsor more application/business-oriented standards building upon these. This general tendency is reflected in the division of security-related standards. Important W3C standards and associated working groups include:

- XML Encryption and XML Signature — these were mentioned above as being 'leveraged' by WS-Security,
- XML Key Management (XKM) — this working group will develop a W3C specification based on the XKM specification (XKMS) previously published by Verisign, Microsoft and webMethods, which specifies protocols for distributing and registering public keys suitable for use with XML Encryption and XML Signature.

W3C also hosts important initiatives to do with defining and representing the meaning of electronic information. The umbrella activity is the Semantic Web, but, more specifically, there is also a Web Ontology Working Group (defining the OWL language). There are close links between the W3C Semantic Web activity and the US DARPA DAML initiative, which includes DAML-S. This is a DAML-based Web Services ontology, which supplies Web Services providers with a core set of mark-up language constructs for describing the properties and capabilities of their Web Services in computer-interpretable form. Although defining and interpreting semantics has received little attention in current Web Services applications, it will become vital as Web Services networks become larger and more complex.

10.3 OASIS

Two important OASIS [15] technical committees (TCs) are the Web Services Security TC (WSSTC) (developing a specification based on WS-Security) and the Security Services TC (SSTC) (developing a specification based on SAML). Other relevant TCs are: Digital Signature Services TC, Extensible Access Control Language TC (XACML, dealing with authorisation policies), Rights Language TC (developing a digital rights language based on XrML, extensible rights mark-up language), Web Application Security TC (vulnerabilities and threat analysis), Web Services Reliable Messaging TC, and XML Common Biometric Format TC.

OASIS is also responsible, jointly with UN/CEFACT, for ebXML (electronic business using extensible mark-up language). ebXML is a suite of specifications for business-to-business interaction using XML messages. Being based on SOAP, ebXML applications arguably are Web Services. However, the ebXML model is not based on WSDL. ebXML is more vertically integrated, but less flexible than the 'regular' Web Services stack. This applies to message security issues, too. Although the same basic techniques are used, ebXML is much more prescriptive, laying down how they are applied. This is good for interoperability, but not so good for generality and flexibility. ebXML also pays more attention to the business significance of the message content and to definition and communication of collaborative

processes. It remains to be seen whether an accommodation is reached whereby e.g. ebXML standardises at the business level and endorses lower level services based on, say, the WS-X standards rather than specifying its own messaging service standards as it does at the moment.

10.4 Liberty Alliance

The Liberty Alliance Project [16] was formed in September 2001 to develop open standards for federated network identity management and identity-based services. In part, at least, it was originally a response to Microsoft's identity management technology, Passport. The Liberty Architecture seems well thought-out and has many attractive features. It is highly compatible with and complementary to SAML (by design). It is noticeable that the list of contributors to SAML and the OASIS SSTC membership have much in common with the Liberty management board and sponsor members. There are areas of overlap and 'competition' between the WS-X security specifications and the Liberty/SAML combination. However, several of the key Liberty/SAML protagonists (notably Verisign) are also collaborators with IBM and Microsoft on WS-X.

10.5 Web Services Interoperability organisation (WS-I)

WS-I [17] is an organisation formed to promote interoperability among Web Services. Its main deliverables are profiles illustrating the application of Web Services standards, and testing tools. A profile is described as '... a named group of Web Services specifications at specific version levels, along with conventions about how they work together'. The initial profile, WS-I Basic, only covers XML Schema 1.0, SOAP 1.1, WSDL 1.1, and UDDI 2.0. However, a Basic Security working group has been established, whose scope includes use of HTTPS, SOAP attachment security and the OASIS WSSTC specifications (i.e. WS-Security).

11. Future directions

Returning to the three-layer model, we have seen that transport layer security (TLS/SSL) can be used to provide point-to-point confidentiality and integrity protection at the lowest layer. When the correspondences between entities on the three levels (real world entities, Web Services and host applications) are clear-cut, information obtained by authenticating the connection end-points may also serve to identify the higher level entities. For simple applications, this may provide sufficient security. However, in general, message-level security (at least) is required.

Most of the components are in place to implement security at the message level covering simple exchanges of messages between pairs of Web Services (public key cryptography and digital certificates, security tokens and token services, etc). There is plenty of scope for variation as regards selection and application of options. The WS-I Basic Security profile may help in this respect once it is published; however, flexibility will remain so that a pair of Web Services will need to communicate/negotiate over which security measures should be used.

Attention in the Web Services world is shifting from simple pairwise request-response exchanges to more complex patterns of messages extended over time (conversations or dialogues) and 'space' (so-called choreography or orchestration of processes involving several Web Services). These patterns can be seen as compositions of the 'atomic' exchanges, and it seems likely that the security measures can also be composed in an analogous fashion. How this should be done remains to be decided, however. The recently issued WS-SecureConversation specification only scratches the surface of the problem.

The same basic components alluded to earlier can also be applied to elements of the message content. For example, parts of the message body or attached documents may be digitally signed. It is clearly important to know the semantic significance of such a signature. Does it commit the individual identified in the associated certificate to, say, a legally binding agreement, or does it simply protect the integrity of item or identify authorship? Such interpretations can be established by convention in closed communities of, for example, supply chain partners. However, to achieve the grand Web Services vision of enabling seamless end-to-end automated processing, a way needs to be found of communicating and interpreting the intended semantics of digital signatures and other 'security statements'. Research on semantic annotations for digital documents is going on in the Semantic Web [14] and related programmes. It is important that a strand within this should address security, and the related topic of trust.

Establishing a clear semantic interpretation (of, for example, security tokens) is important at the message level as well. As systems become larger and more complex one can imagine that by the time a request reaches its destination it will have accumulated a large number of tokens asserting who has done what to it and why. The receiving agent will have to treat these tokens as evidence, and reason about whether the message can be trusted.

Another important issue is manageability. Security cannot be treated as a black box add-on. Measures need to be matched to requirements. In an ideal world one

should be able to define confidentiality, integrity, availability, privacy and non-repudiation policy requirements expressed in terms of the business application along with the business processes, services, partners, etc. The Web Services platform should configure itself to realise these requirements, providing high-level facilities to manage, monitor and record. It should also negotiate with the platforms of partners to agree the parameters necessary to ensure meaningful interoperation. The current situation is a long way from this ideal, and clearer semantics relating to security and application issues would greatly facilitate progress.

12. Conclusions

Security is fundamental to the successful adoption of Web Services for business applications. It is wise to be aware of the limitations of the current state of knowledge, but this should not deter organisations too strongly from fielding Web Services applications. The security concerns surrounding conducting business over Web Services are real, but so is the potential value to be gained. It is possible to introduce Web Services slowly and gently, starting with simple applications. While providing real benefit in the short term, this will limit risk and build the confidence and experience required to roll out more complex services in the future as standards, infrastructure and services mature.

The most basic security measure is to use transport layer security, for example, an SSL connection between two points, and this may be sufficient for simple applications. For more complex environments, e.g. more than two parties, or multiple Web Services, complete messages or individual parts of messages may be encrypted and signed to protect the confidentiality and integrity of Web Services messages. Tokens may also be added to messages to assert claims, e.g. about checks that have been carried out by a trusted authority to confirm identities.

These mechanisms are quite usable, but the way they are applied has to be decided on a case by case basis. For example, the significance of a signature applied to a particular part of a message needs to be agreed by sender and recipient. The next few years should see steady progress in research and standardisation communities towards generality and interoperability. A major challenge will be to combine this generality with ease of management of security measures via business-oriented policies.

Acknowledgements

Other students and staff of the secure systems and security technologies research groups contributing to the work described in this paper are Adam Clarke, Tim Davy, Robert Glassford, Meena Laitinen, Steven McLellan and Luann Rragami.

References

1 Cerami E: 'Web Services essentials', O'Reilly and Associates Inc (2002).

2 Selkirk A: 'XML and Security', BT Technol J, 19, No 3, pp 23—34 (July 2001).

3 Selkirk A: 'Using XML Security Mechanisms', BT Technol J, 19, No 3, pp 35—43 (July 2001).

4 Bosworth K P and Tedeschi N: 'Public key infrastructures — the next generation', BT Technol J, 19, No 3, pp 44—59 (July 2001).

5 Gooch D et al: 'Firewalls — evolve or die', BT Technol J, 19, No 3, pp 89—98 (July 2001).

6 McTaggert M: 'An introduction to XML encryption and XML signature', — http://www-106.ibm.com/developerworks/library/s-xmlsec.html/

7 WS-Security Specification V1.0, (April 2002) — http://www-106.ibm.com/developerworks/library/ws-secure/

8 EESSI — http://www.ictsb.org/EESSI_home.htm

9 Liberty Alliance Project Specification Archive V.1.1 — http://www.projectliberty.org/specs/archive/v1_1/index.html

10 SAML Specification V1.1 — http://www.oasis-open.org/committees/download.php/2791/sstc-saml-1.1-cs-02.zip

11 Box D: 'Understanding GXA', — http://msdn.microsoft.com/library/

12 'Security in a Web Services World: A Proposed Architecture and Roadmap', IBM/Microsoft White Paper — http://www-106.ibm.com/developerworks/security/library/ws-secmap/?dwzone=security

13 WS-Security specifications index page — http://msdn.microsoft.com/library/en-us/dnglobspec/html/wssecurspecindex.asp

14 World Wide Web Consortium — http://www.w3.org/

15 Organisation for the Advancement of Structured Information Systems — http://www.oasis-open.org/

16 Liberty Alliance — http://www.projectliberty.org/

17 Web Services Interoperability — http://www.ws-i.org/

Paul Kearney is leader of the Secure Systems Research group in BT's Business Systems Research Lab at Adastral Park. He has a BSc (1st class) in Mathematical Physics (Liverpool University) and PhD in theoretical particle physics (Durham University). He spent nearly 10 years working for British Aerospace in a variety of technical roles. At the end of this time he was leading an R&D group in IKBS and Advanced Software Engineering. Joining Sharp Laboratories of Europe in 1990, he led a research group looking at personal agents and multi-agent systems (MAS) until 1997. He then joined BT, initially as a Senior Research Fellow working on MAS applied to business problems, self-organisation and emergence of collective behaviour, and software engineering methods for MAS. In 1997, he established a security research group and is now leading the Secure Web Services project. His major research interest is in how to achieve trust and security in a context of dynamic relationships, and shared governance, processes and resources.

James Chapman is an experienced consultant, developer, team leader and project manager in security and advanced network and mobility applications. He joined BT in 1989 and, during his career in BT, he has worked on many aspects of communications such as unified messaging, automated speech recognition, and open network interfaces for next-generation networks. He has been responsible for developing concepts, trials and live services for customers within and outside BT. In 2002, he led technical aspects for BT in the PUMA project (Personalised User Interfaces for Information Management and Authorisation), developing a biometric-enabled user interface exploiting aspects of acoustic, phonetics and context for speech for personalisation and fraud detection. He delivered a joint trial with the University of Birmingham involving 20 000 recordings for the PUMA interface. Currently he is working on the security of high-speed networks.

Nicholas Edwards graduated in 1990 with first class Honours in Natural Sciences from Jesus College Cambridge. After completing a DPhil and Postdoctoral Research Fellowship in Laser Physics at Oxford University, he joined BT in 1995. He initially worked in a team creating new applications for communications in healthcare such as a Telecare system for supporting older people in the community. From 1999 until 2002, he worked on intelligent networks, and the development of open network application programming interfaces such as Parlay. He now leads the Security Technologies Research group in BT Exact at Adastral Park, and is responsible for security research projects covering biometrics, smart cards, network security and mobility security.

He is a member of the Institute of Physics and the Institute of Electrical Engineers, and a Chartered Physicist and Engineer.

Maurice Gifford was awarded a BSc honours degree in Electronic Physics at the University of London in 1983. He joined BT in October 1989 intially working on the development of a leading edge digital switch. He is currently working in the Security Technologies Research Group of BT Exact at Adastral Park. Work to date includes biometric systems research, developing software demonstrators for biometric technologies, XML security and aspects of network security. Previous work for BT has included ***evoca*** product development, network and network management systems design and development.

Liwen He graduated with a top class BEng degree in China in 1990 and PhD degree in 2002. He first worked at a software company as a research engineer for the Chinese National 863 high-tech research projects. He then became an R&D department manager for a Sino-HongKong joint venture of electronics manuafacturing. In 1996, he studied for the PhD degree at the Automatic Contol and Systems Engineering department, University of Sheffield, majoring in evolutionary computation methods for manufacturing scheduling optimisation and network routing, and fundamental theories of genetic algorithms. In 1999, he joined the Advanced Telecommunications Research department at BT Exact as a research engineer, and is now a senior research engineer in the Secure Systems Research group at Adastral Park. His major research interests are network design optimisation, management, capacity planning of optical/mobile/IP using AI technologies, network cost modelling, reliability analysis, information system security (especially Web Services security) and communications network security (especially routing protocols security using encryption-based techniques), intelligent intrusion detection and intrusion tolerace techniques. He has two international patents and a number of publications in these areas.