ABSTRACT
Authentication is the base of securing modern Web Services. Four different authentication techniques are discussed and compared to each other to find the best authentication method available. A final solution proposes a combination of the several techniques, combining the strong points of these methods.

Keywords
Web Services, security, authentication, multiparty authentication, WS-ABA, OWL-S and Lifecycles.

1. INTRODUCTION
As the Internet is growing all over the world and the need to exchange data is becoming more important, the need of Web Services to support the data exchange is growing as well. Every year, increasing numbers of companies are using Web Services to connect their applications and provide themselves and their customers of more, better and more actual information.

A Web Service is a term to describe the way services in a network communicate. As Wikipedia [WIK1] describes a web service: “A web service is a collection of protocols and standards used for exchanging data between applications”

Web Services have several explicit characteristics, as explained by Muschamp in his paper about Web Services [MUS04]:
- “It can describe itself – so that other components can understand the functionality it offers and how to access that functionality”
- “It can allow other components to locate it – so it can be used when required”
- “It can be readily invoked whenever another component wishes to use its functions”

Main issue with web services is keeping the data with the persons and other services which are authorized to view it and keeping it secure from unwanted access.

The first researches on Web Services primarily focused on the exchange of data and making it available for as much other users and services as possible. With the growth of the Web Services more data is passing the Services. This is not only public data anymore: an increasing part of the data is private data which needs to be secured. The problem of securing Web Services is a well-known issue in the development of the Web Services. Damiani et al. have identified this issue and found out authentication is one of the main security issues in modern Web Services [Dam02]. The security of Web Services can be split into several parts. One of the main issues is the authentication, the detection and verification of the authenticity of a connecting partner.

Traditional authentication was performed by sending a username and password. This method is still widely used, but lacks functionality and flexibility required for Web Services. A difference has to be made to the subparts of security systems. Authentication is, as mentioned before, the detection of the authenticity of a party. Authentication is different from authorization. Authorization is the subsystem controlling the permission of a party. In other words, which parts of the system are accessible by the already authenticated client or authenticated server. Authentication is the first step required for a security system, detecting the true identity of the party. After the authentication has taken place, the second step, authorization can start giving access to the subsystems. The lack of sufficient authentication methods for large Web Service systems has stalled the development of the Services. Recently, authentication methods have been developed by several groups of researchers to solve this problem. However, there is no overview of these methods and they have not been compared to each other yet. All methods seem to have some weaker points in their implementation, but a real comparison has not been made.

This paper will address the problem of authentication in Web Services by comparing four available types of authentication methods. It has to be noted these are not all available authentication algorithms, but just a selection of the most used ones. These algorithms are selected because of the differences in approach of the authentication issue in Web Services. Goal of this research it to determine the best authentication method for Web Services available at this moment. This could be one of the most used ones researched in this paper or a combination of the strong points of them, minimizing the weaker parts of a certain algorithm.

2. PROTOCOL OUTLINE
This chapter will give an overview of four of the most used authentication methods. The methods will be explained and the strong and weak points will be shown. There is no comparison made in this chapter; the methods are compared to each other in chapter 4.

2.1 Multiparty authentication
Using multiple services to retrieve information from is one of the advantages of the Web Services framework. If one service fails or is busy, another service can take over the workload of the first server. This advantage has one major issue to solve: it requires a special authentication mechanism to distribute the authentication details among the services serving the data to one client. To solve this issue the multi-party authentication protocol has been developed [Zha04]. It consists of several parts which will be described in the next paragraphs.

A request for information starts at a client. This primary client requests a new session at the Session Authority. The Session Authority supplies a unique session identifier and a session secret to the client.

The client connects to the Web Service using the identifier given by the Session Authority. The Web Service then creates a
new server instance for that client only and passes the session identifier to this instance. The instance sends a special message to the Session Authority which requests joining the session of the client. The Session Authority responds with the same session secret given to the client. The session between the client and the service provider can now be authenticated by the session identifier combined with the session key.

When a service provider wishes to delegate the work to an other provider, it passes the session identifier to the new service. This service connects to the Session Authority to obtain the session secret of the session as well. The client now sends its requests to the new service provider which proves to be a legal user of the session by using the correct session secret.

The same steps can be done at the client side if the client wants to delegate its work to another client. This process can be repeated any number of times as required. Figure 1 illustrates the behavior of the Multiparty Authentication protocol. Client 1 starts the communication in step 1 with requesting a new session at the SA. The SA responds in step 2, sending the session secret to the client. The client connects to the Web Service in step 3. The service spawns a new service instance (Step 4), which registers at the SA in step 5 using the session identifier. The SA responds to the Service provider with the session secret given to the client. The Session Authority responds in step 6, after with the communication between client and server can start, authenticated by the session secret.

Delegation to another service is displayed in steps 7 till 11. The old service sends a message to the new service provider. The new service sends a ‘Join’-message to the SA and receives the session secret. The new server can now take over the workload of the original Web Service and communicate in a secure way with the connecting client.

Delegation to another client requires the same steps as the delegation to another server. The old client notifies the new client to join the session. This new client sends the request to the SA and receives the session secret to start the communication with the already existing Web Service.

2.2 OWL-S

OWL-S is the extension of the Web Ontology Language [W3C-1] for Security purposes. This security extension is a language used to develop ontologies for the security measures in Web Services.

Ontologies are explicit specifications of conceptualizations [Gru93]. In more understandable language these ontologies are a representation of user properties in a tree structure. An example ontology is shown in Figure 2. The data model of an ontology consists out of classes, attributes and relations. The connecting client is regarded as an instance of a class in the representing ontology.

The attributes of a class are divided in two parts. First part is the attributes on which a user is compared and linked to the class. The second part is the information about the authentication details the user will get when authorized. This authorization is a different security method as authentication and to keep this research on-topic, it will be neglected.

![Figure 2 - Example ontology](image)

In the example ontology in Figure 2 a relation of fruit and vegetables is modeled. Both are subclasses of the main class Food. Both fruit and vegetables have subtypes themselves, as shown in the figure. All classes in the ontology tree have properties, like the color and seedless properties of the Strawberry class. In the example only the lowest classes in the tree have properties, but also high-level classes can have these properties. An example in Figure 2 could be the tag poisonous in the Food class. All classes below inherit this property, resulting in having the poisonous property in all types of fruit and vegetables.

OWL-S provides a solution for grouping users with equal access permissions into easy manageable groups. Because of the ontologies, clients connecting to a Web Service can be placed into one of the groups and based on the properties of this group the authentication can take place. Depending on the size of the ontology, many user types and groups can authenticate on the Web Service and been given the right credentials. The OWL-S language is primary designed for the use of user authentication, but it can be perfectly used for authorization as well. The language is mainly used to model the rights of users and groups, thus describing the user authentication. However, the ontology structure can be used for authorization by adding the required authentication details into the classes. Requirements like a password or required client properties can be inserted into the ontology tree, resulting in excellent properties to use for authentication.

An example of a tree containing user access might contain a main Client class. This user class contains the user credentials such as password. For an airline Web Service, which provides access to the flight information and is able to add reservation for flights, the user can be extended by several groups which can connect to the Service, such as ServiceDesk, TravelAgency or Website. These subclasses all have properties of their own, like the number of a service desk, a name of the travel agency.
or the URL of a website. A TravelAgency instance can have several classes underneath it, for example the FrontOffice, the BackOffice and an InformationDesk. All these departments can have several users connected to it, which will inherit all authorization rights and requirements from the parent department. A visual conception of this example authentication ontology is displayed in Figure 3. It shows the ontology tree used in the authorization process, with the authentication details added to the root node. This means the authentication requirements are inherited into all child nodes and if required the authentication data can be overwritten in the child node.

Figure 3 - Airline Web Service Ontology

2.3 WS-ABA

The protocol WS-ABA is one of the parts of the WS-Security framework. WS-Security is one of the most used frameworks used for Web Services security. It has various implementations in for example SOAP [W3C-2] and many other Web Service languages used on the Internet. WS-ABA is the Ws-Attribute Based Access control mechanism, controlling the access, authentication and authorization to Web services using the WS-Security framework.

WS-ABA is a system designed for the control of access to a Web Service [Ber04]. The protocol is designed to be a flexible solution for controlling the access to Web Services in general. It mainly consists of functions controlling the access to the server and the WS-ABA model also includes an authentication method which will be used as the base for the policy-based authentication algorithms.

WS-ABA depends on a Mutual Authentication protocol, which means both sides of the communication need to authenticate to each other and the assumption is made the other side is not required to be honest. The authentication of the other party need to be thoroughly checked using a solid authentication mechanism.

The authentication algorithm used in WS-ABA is based on policies to determine if a client can access the service. Policies are sets of rules. These rules can be detailed and difficult, but most of the times these rule sets are fairly simple. The policies are mostly defined in XML and can be changed on a running server fairly easy.

Figure 4 - Example WS-ABA policy

An example of a WS-ABA policy file is showed in Figure 4. This is a part of a XML file, defining an identity, the group of the user, the role and the clearance of this user. A lot of policies can be defined using these policies, creating large structures of XML. Many different policies can be defined in a server, creating a thorough way of authenticating a client. These policies all need to be written by the system maintainer or on system setup. For large authorization systems, these policy XML definitions can get quite large and harder to maintain, as there is no easy way to manage them yet.

2.4 LIFECYCLE MANAGEMENT

Lifecycles are a solution for general security solutions introduced by Skogsrud et al. [Sko04]. In their research they propose an algorithm for a lifecycle mechanism primary built on Trust management. It however also includes a way of authentication and this authentication is handled a different way as conventional user authentication.

In the Trust-Serv algorithm [Sko04-2], authentication is negotiated between the connecting client and the Web Service. This negotiating mechanism authenticates the user depending on several negotiable properties. If a new client connects to a Web Service, the two parties start to negotiate about the access rights the client should get.

After this negotiation the main characteristic of this algorithm shows up: the lifecycle. These lifecycles control the behavior of the Web Service in time. The impact of these lifecycles on the authorization and trust management are clearly worked out in the papers of Skogsrud. For the authentication, the lifecycles add a specific length of time to the validity of the credentials of a logged in user. After a certain period of time, the credentials are no longer a valid authentication on the Web Service. The client needs to reauthenticate to be able to use the Web Service again. To reauthenticate, the client is required to send a new authentication request to the Web Service, containing a complete new login request.

The functioning of the lifecycles in the authorization part of the Lifecycle Management protocol does not have any influence on the authentication, except the fact clients need to reauthenticate every cycle. To keep this paper on the topic of authentication only, the detailed functioning of the Lifecycle Management protocol will be disregarded. For detailed information of the system I would like to refer to the paper of Skogsrud et al.

3. PROTOCOL EVALUATION

3.1 Multiparty authentication

The most important advantage of Multiparty Authentication is clearly the support of multiple clients and Services. Multiparty Authentication is the only frequently used protocol that supports the delegation of both the client as the service role. This results in a flexible Web Service solution.
There is a problem with the Multiparty Authentication. As every delegation to another client or server is done with the Session Key, a big security issue rises. If the key is lost to an untrusted third party, that party can take over the complete session, either the client or the Service. Such an intruder can act as a new server or a new client and take over the complete handling of that side. This way the other side is communicating directly with an untrusted party, without any possibility to detect this session intruder. Unaware of this unwanted user, the uncompromised party continues to exchange data and possible private data can be lost to this intruder. The other way around, if the Web Service is compromised, the client might receive information that is not correct. The compromised server can misinform the connecting client without the client knowing the data is invalid.

Another big disadvantage of the Multiparty Authentication algorithm is the required Session Authority. These systems will not function without the Session Authority handing out the session secrets. The central Session Authority needs to be trusted at any time. If the Session Authority is compromised, illegal session keys can be handed out. Or even worse, the Session Authority can leak the session secret to a third party, which could take over control of the complete session.

The requirement of the Session Authority also includes an availability issue. If due to some reason, which might be unscheduled or scheduled maintenance, an outage or routing problems so the Session Authority cannot be reached by the client or the Web Service provider, the complete Web Service will not function. Both sides need to have the same session secret supplied by the Session Authority. If one of the sides fails to receive the session secret, no communication can occur at all.

### 3.2 OWL-S

As OWL-S is based on ontologies, clients connecting to a Service provider can be easily placed into a group corresponding section of the ontology. This method is easy and straight-forward, making it easy to implement. However, the ontologies need to be detailed to accommodate all possible users or they need to be vague. The vagueness introduces a new security issue, because non-authorized clients can fit into the properties of a vague class and get authorized wrongly.

To be fully efficient, OWL-S ontologies need to consist out of well designed structures of classes and relations. If the ontology is not well designed, problems as described before can occur and the system might not function as expected.

### 3.3 WS-ABA

Policies are the main characteristic of authentication using the WS-ABA algorithm [Ber04]. WS-ABA heavily depends on these policies and is flexible to use. The main advantage of using policies is the possibility to grant access to former unknown clients. The policies can identify a client or if a client is not known, the policy can classify the connecting client. Using correct, accurate policies connecting clients can be put in the right categories according to the rule set.

The main security issue pops up at this point as well. Just as the ontologies in OWL-S the policies need to be well designed to assure no false positives can occur. False positives might give untrusted clients access to the Web Service. The security hole might be worked around by using better and more advanced policies, but as long as ‘new’ clients are permitted, there is a risk an intruder can access the data supplied by the Web Service.

The problem is even larger, if an intruder knows the way the policies works, the system can be compromised and clients that should not have access can connect and authorize easily.

The other way around there is no way to authenticate the server on the client. The client connects to a Web Service, or requests an answer to one of his messages in general, but has no ability to check the authenticity of the Web Service answering his message and supplying the required data. There is no guarantee at all the response to the request is given by a trusted Web Service. A request from a client could end up at an untrusted Web Service which still accepts the client. The untrusted Web service can send anything to the client without this client being able to authenticate the source of the data. This might be solved by adding a unique ID to every Web Service, but this ID can be copied or stolen as well.

### 3.4 Lifecycle Management

The Lifecycle Management protocol uses negotiation to authenticate a connecting client. This has the advantage, just like the policy based algorithms like WS-ABA, formerly unknown clients are able to connect to the Web Service and authenticate as a fully authorized client.

The main advantage of Lifecycle Management above ‘normal’ policy-based is the limited time span such an authorization is valid. Due to the lifecycles, a client needs to log in again after a certain span of time. This causes a slightly increased overhead, but the security increases significantly. In all other algorithms, if the security system is broken, the intruder can stay infiltrated in the system. This intruder is either spreading incorrect data or receiving private data without the other side knowing.

The Lifecycle Management protocol overcomes this problem by the required re-authentication. Authorized clients need to reauthenticate every time span intruders are excluded from the system and these intruders need to break the security system as well. This additional security feature adds an extra layer of safety to any of primary authentication methods used, mainly time negotiating authentication.

### 4. COMPARISON

The four authorization methods that are worked out in this paper are the most used authorization mechanisms in Web Services at the moment. This chapter will compare these methods from three different angles: flexibility, usability and security. For the flexibility I will only compare the way the protocols handle formerly unknown clients, for usability the easiness in which the rules or policies internally can be changed and finally the main security issues of these algorithms will be compared. For each of these aspects the strong and weak points of the algorithms will be discussed.

#### 4.1 Flexibility

As classic password-based authorization is inflexible and has no support for new clients. Two of the algorithms, the Multiparty Authentication and OWL-S authentication do not have the ability to grant access to formerly unknown clients that should have access.

Multiparty Authentication does not support new clients because it is primarily built on the classic password-based authorization. The session secret is supplied using password authentication or no authentication at all. This means clients connecting to the Session Authority need to be known at the SA to receive the session secret. Formerly unknown clients cannot receive a key due to security restrictions.
The ontologies of OWL-S are quite strict and because of this they do not support the login of clients that have never been specified directly in the ontology. A slight difference in the properties of a connecting client might cause the representation not to fit on the model, causing the authorization to fail. Even if the ontologies are well-designed, this problem can occur. This leads to the conclusion of a low flexibility of the OWL-S authentication protocol.

Both other algorithms mentioned in this paper do have the flexibility to support formerly unknown client. The WS-ABA protocol is using the policies best: with well designed policies there is just a small change of false positive authentication and a lot of clients that should have access will be allowed to connect to the Web Service and receive data. The policies allow anyone that looks like another client that already has access to access the Web Service system and use the services.

The Lifecycle Management authorization protocol is usually a negotiation-based algorithm, which is flexible and will allow many previous unknown clients to connect. It is possible to hook another primary authentication mechanism into the Lifecycle scheme, resulting in a slightly different type of authentication. Because negotiation is one of the most flexible schemes for authorization it is chosen to be evaluated in this paper. Other primary authentication schemes are possible to be used with the Lifecycle Management system, but these will perform less regarding flexibility and equal or less regarding security. This makes the combination Lifecycle Management with the negotiating authentication method the most flexible combination.

Negotiating about the rights the client will receive is the best solution to have a flexible authorization system, closely followed by the policy-based algorithm WS-ABA. These systems combine the access of new clients with a closely regulated system which controls the authentication of these new clients.

### 4.2 Usability

The usability of the four used algorithms is compared by the amount of work required to change the underlying policy, ontology or rule set.

The Multiparty Authentication is using a simple password-based primary authentication protocol. This protocol enables a simple addition, deletion and modification of clients to the system. The management of larger groups of these users will however cause problems. Managing large groups or single users without a supporting system is hard and is not user friendly. This hard-to-manage user system makes the Multiparty Authentication protocol score a low mark for the usability section.

OWL-S however requires quite a hard ontology, but after it is designed it required almost no maintenance. There are tools available to design the ontologies and debug them, making the design of an ontology a lot easier. More information about debugging these ontologies can be found in the research of Parsia et al. [Parf05]. Using these tools OWL-S is one of the most user-friendly protocols available.

Only the rule sets of the policy-based WS-ABA authentication protocol are more user friendly as the ontologies of OWL-S are. There are tools to define and design policies for the WS-ABA authorization protocol, resulting in a simple way to design the policies and manage them while the system is in use. These tools are widely available on the internet making the WS-ABA, and the policy based solution of WS-Security in general the most user-friendly method in this paper.

The Lifecycle Management authorization protocol, and more specific the negotiating authentication method is hard to setup. There are almost no tools available for automatic configuration of the negotiating rules, mainly caused by the fact the rules need to be adapted to the specific situation the Web Service is functioning. As the negotiation rules have to be strict and need to be custom designed for every single Web Service, this method consumes a lot of time setting up and maintaining.

The most user friendly methods for authentication in Web Services are the policy and ontology based algorithms. Having lots of tools available both authentication systems are easy to set up. The policy based protocol WS-ABA is easier to maintain compared to the harder to change ontologies of OWL-S.

### 4.3 Security

Last but not least the security of the authorization methods should be taken into account. As more and more private data is spread using Web Services, the authentication methods should assure unauthorized access is prohibited.

Main issue for the security in Web Services is the choice system developers have to make when setting up the Web Service. If a flexible system is required, providing a lot of options to connect new clients to the Web Service, security issues arise almost instantly. Bhargavan even states flexibility is merely a threat to security: “Flexibility can be the enemy of security, and we may hope that standard policies and practices can be agreed” [BHA01]

These issues are caused by the option to connect new clients to the Web Service. If a new client can connect, there is a possibility the connecting client is not trusted, but due to a security hole in the policies judging the authorization request the illegal client could receive access permission wrongfully. These security issues arise mainly at the policy-based protocols like WS-ABA. Having a huge flexibility authorizing new clients, the possibility one of the connecting clients should not have given permission to access the system dramatically increases.

Illegal usage of Web Services is not only caused by the possibility to grant permission to these new clients, but also the ontology based OWL-S authentication system suffers from serious security issues. The properties of a connecting client are compared to the properties of the representation of a class in the ontology. If due to some error the comparison gives a false positive, an illegal connecting client could be authorized being a fully legal connecting client. The illegal client can access the Web Service like being a fully authorized, while it never should have been granted access.

Proper designed policies and ontologies minimize the possibility of such an intrusion of an untrusted client. However, it is not possible to completely exclude this security issue.

A large improvement to the security system is the limited time a granted authorization is valid in the Lifecycle Management authorization protocol. This protocol requires every client connected to the Web Service to reauthenticate once in a while. This happens when a complete lifecycle of the authorization mechanism has been completed.

Multiparty Authentication does not suffer from this security issue as much as the policy-based algorithms do because it does not support the connection of new clients. It however suffers from another security issue. A session is identified by a public
visible Session Identifier and authorized using the Session Secret. Communication between the client and the Web Service is also authorized using the Session Secret.

The main security flaw in Multiparty Authentication is this Session Secret. As long as it is private and only known to the client and the Web Service, the system is safe. But if the Session Secret is lost by one of both parties, stolen or sniffed by an intruder, this intruder can take over the communication between client and Web Service. It can actually act as one of both parties, taking over control of one side of the system. The other side, which is still communicating using the private Session Secret, has no possibility at all of detecting this intrusion orswap of communicating partner. It is even possible to take over both sides of the communication, creating a man-in-the-middle situation where all communication passes through the infiltrated instance. This issue creates a huge security leak if the Session Secret has been compromised and unexpected behavior of authentication can start. Due to the handing over of the session secret, this method is

5. COMBINING PROTOCOLS

All in this paper mentioned authentication methods have their own security issues, most of them caused by the intrusion of an untrusted third party.

To solve this problem, some of the solutions can be combined to profit from the advantages of multiple methods and decrease security risks.

Most of all the concept of limited duration of the Lifecycle Management can be combined with any other authentication method. The implementation should not be dependant on the lifecycles of the authorization protocol. All previously mentioned algorithms, especially the Multiparty Authentication algorithm, can be expanded with a mechanism that requires a reauthentication once every time span. The exact length of the time span should be determined by executing tests to get the optimal length. As only reauthentication is required, there are no problems combining a timed login with any of the protocols. A re-login message needs to be created which signals the client to authenticate again. This way, safety can be assured and intruders are filtered out of the Web Service automatically.

The most optimal solution is the combination of the flexible policies for primary login, Multiparty Authentication for the delegation of the workload and a maximum time span an authentication is valid. This will result in the most flexible and safe solution possible.

6. CONCLUSION

After a small research on the functioning of the authentication protocols most used in current Web Services, we’ve compared several characteristics of the Services.

First of all, flexibility of the protocols is compared. As described in paragraph 4.1, policy based algorithms like WS-ABA are the most flexible authentication mechanisms available, allowing new clients to be authorized even if they are not know in the system.

Usability is one of the main problems with all Web Service authentication solutions. The better a solution work, the more detailed the setup of that solution needs to be. This results in large setup and maintenance costs. Due to the many tools available for designing Web Services policies, this should be addressed as the most user friendly solution available. However, a well-protected Web Service still needs a lot of work setting up the design for the initial policies to be fail-safe and these policies need to be adjusted continuously to prevent unauthorized access.

The most optimal solution for authentication is the combination of a policy-based primary authentication protocol like WS-ABA, combined with the flexibility to delegate the work of the Multiparty Authentication protocol and the safety of a limited time span used in the Lifecycle Management protocol.

If flexibility is not required, the security of the Web Service can be improved significantly by removing the access policies so unauthorized access is minimized. Removal of the multiparty protocol will improve the safety slightly, but the risk of losing the Session Secret is minimized by the implementation of the maximum length of validity of the authentication and the Session Secret.

If security is the most important thing a Web Service requires, the Lifecycle Management algorithm is the best solution to use, but the negotiating primary authentication method should be replaced by a traditional fail-safe password-based authentication system or by well designed ontology-based system. Only with these combinations maximum security can be assured, even if the authentication system is compromised the maximum time span of the authentication will require the intruder to reauthenticate and the system will be self-repairing.

7. FUTURE WORK

Future work might build and test a fully functional Web Services authentication protocol, using policies, multiparty features and a fixed maximum duration of an authentication. As there are theoretical no problems for designing such a system, there needs to be a real-life test to assure no new issues arise combining these three concepts into one complete Web Services authentication solution. The system should be tested thoroughly and the security could be compared to the existing protocols.

ACKNOWLEDGMENTS

I would like to thank the University of Twente and in special the Faculty of Electrical Engineering, Mathematics and Computer Science and its teachers for their support during the Referaat course.

REFERENCES

[Ber04] Bertino and Squicciarini, A Flexible Access Control Model for web services, FQAS 2004 pag 13-16, 2004


