Securing SOAP Web Services for Mobile Devices on Different Platforms

Carsten Kleiner, Thole Schneider
University of Applied Sciences and Arts
Department of Computer Science
Postfach 920261
30441 Hannover
Germany
ckleiner@acm.org, thole.schneider@googlemail.com

Abstract: Enterprise applications are often arranged in service-oriented architectures (SOA) nowadays. Many times services in a SOA are implemented by SOAP web services often including application-level security. With their increased computing power mobile devices such as PDA and smart-phone become promising clients for such enterprise applications. This paper contains an analysis of built-in support for secure SOAP web services on different recent mobile device platforms. As it turns out, the support for SOAP in general is not comparable to stationary devices. This is even more true for security extensions to such web services. Therefore we also evaluate and implement feasible add-ons to the platforms providing some support. The platforms are finally compared according to several different dimensions regarding secure SOAP web services and individual strengths and weaknesses are presented.

1 Introduction and Motivation

The total number of mobile devices[1] sold globally is continuously increasing. At the same time the computing power of these devices is growing at an unbelievable pace and has already reached the power of a desktop PC from a couple of years ago. Since many of the restrictions of earlier generations of mobile devices such as limited main memory and persistent storage capabilities, limited CPU power as well as very limited and intermittent internet connection and bandwidth are not prevalent anymore these devices may nowadays be used for advanced client applications.

Recent trends in the consumer market are reflecting this as mobile device applications currently move away from browser-based applications to full strength client-applications often called *apps* in this context. There are several different reasons for this trend but a detailed analysis is out of the scope of this paper. We just recall that full client applications on mobile devices are frequently used lately.

[1] In this paper we use the term *mobile device* to describe mobile telephones or smart-phones; some of the findings may be transferred to other types of mobile devices as long as they are based on one of the operating system platforms considered, but this has not been the focus of our work.

Several different operating system platforms such as Android, Symbian OS or iOS are currently used by the device vendors on their mobile devices. All of these platforms require a specific programming language or dialect for implementation of applications (similar to the situation on the desktop and server computers about 10 years ago). Basically the only platform-independent alternative is Java ME for which there exist virtual machines on most recent platforms. Unfortunately Java ME is already rather old and many of the previously mentioned then existing restrictions of the devices have been used in its design. This makes Java ME somewhat outdated today; it is nevertheless the only platform-independent technology to date.

As many of the commercial applications are not pure-mobile applications but rather use mobile clients in a complex and distributed software system there is an urgent need for (at least) platform-independent server implementation of distributed applications. Web services are frequently used to provide such an implementation. While many of the consumer mobile applications require rather thin interactions and are thus often implemented as REST web services, the semantically rich interfaces of commercial applications often use SOAP based web services. Thus in order to be able to use recent mobile devices in business processes the support for SOAP web services on mobile device platforms is important. The support for SOAP web services is still not perfect on the aforementioned platforms nowadays but there are (rather straightforward) ways to use such services as will be explained in section 4.

Since commercial applications often deal with information of high and varying sensitivity it is also essential to be able to secure such web services. Different applications have different requirements (cf. section 3) regarding security and the full encryption on the transport layer (https in most cases) may often not be the desired solution. There exist quasi-standards for securing SOAP web services on application level (e.g. [NKMHB04] or [NGGG07]) but support of such standards on mobile device platforms has to be examined.

In our work which is summarized in this paper we have evaluated the support for securing SOAP web services on the application level on different mobile device platforms. In particular we have used sample geo web services based on the OpenGIS standard (cf. [Mab08]) on the server side. We did that in order to obtain sufficiently general results using a standard on one hand while still being able to base the findings on a concrete application domain. This is required to define realistic security requirements for the services. Regarding security support our results are easily transferable to other service for mobile devices as there is no specific spatial component in the security architecture. On the client side we have evaluated three different platforms, namely Symbian OS, Android and Windows Mobile 6. We have chosen these due to their wide market penetration and free availability of development tools.

After a review of related work in section 2 we introduce security requirements for different scenarios in section 3. In section 4 we analyze the support for SOAP web services on the different mobile device platforms in general and for relevant security extensions in particular. We applied these features to scenarios from section 3 in prototypical implementations one of which is presented as an example in section 4.1.4 for illustration purposes. We also assess the amount of manual work required to obtain these solutions. Finally we conclude with a summary of the results in section 5. In this part we also present an

overview of supported security configurations as well as a classification of the different platforms with regard to implementation of secure SOAP web services. Ideas for future possible extensions are briefly discussed in section 6.

2 Related Work

The Open Mobile Alliance (OMA) defines two architecture models for web service usage on mobile devices ([Ope05]): direct and indirect. The first one requires the mobile device as service requester to have a full web service stack available whereas the second one uses a proxy for a protocol translation between mobile device and service provider. If security aspects play a major role for services (as in our case), the proxy model should not be used, because end-to-end security is impossible. In addition the proxy model had been motivated by service clients that lack the computing power to implement a full web service stack. Nowadays this assumption is not true for mobile devices (anymore); thus we focus on the direct interaction between service requester (mobile device) and provider.

Earlier work in mobile web services focused on dealing with the resource problems on the constrained mobile device (e. g. [WN07]). Such restrictions are non existing anymore as even sensors using SOAP web services are suggested today (cf. [GPF09]). Therefore similar to [NKS08] we propose the usage of full strength SOAP web services on mobile devices for enterprise applications.

Some work on using SOAP on mobile device platforms, which as explained in section 4 is seldom natively supported, has been published. In [SKH08] the authors describe a SOAP extension to Java ME. Even though being the only portable programming language on mobile devices Java ME is somewhat outdated today which is why we do not use it. A migration of the described extension to e. g. Android might be successful and might be considered in future work. Similarly [KS07] also uses Java ME technology and kSOAP (as we will do for Android) but they do not consider security for messages. The work presented in [SJP07] is closely related to our work, even though it focuses on mobile devices as web service providers. Therefore that work is constrained to the platform used for their mobile hosts; in contrast we compare security aspects on different client platforms.

A discussion of XML security specifications and reviews that provide the foundation for our work can be found in [EFG+08].

3 Requirements for Securing Advanced Web Services

The specific requirements for security of a web service depend on the particular service and its usage scenario. Whereas some, typically rather simple, consumer-oriented services might be just fine without any security features, commercial business services will have more sophisticated security requirements. Our work focuses on semantically rich services which typically use complex messages and data types such as OGC compatible

geo services.

The general security goals of confidentiality, integrity, authenticity, availability and non-repudiability are also valid for mobile web services. In fact due to the different type of underlying network they are typically even more important than for stationary devices. Specifically the first three of these goals may be achieved by appropriate technology such as encryption and signing. In the context of SOAP web services there are well-known standards for achieving these goals such as web service security, web service policy, XML encryption and XML signature.

While encryption and signature may also be achieved on the transport level by using SOAP over https, this does typically not solve the problem for semantically rich services. These services often require individual encryption and/or signing of single SOAP elements in order to satisfy the aforementioned requirements as well as guarantee for a maximum level of user privacy. Furthermore the complete message encryption of https does not allow for complex processes on the provider side where one provider aggregates results from different other providers to generate a response to the client. This is very frequent in commercial applications using business processes and might even be required for consumer services such as a routing service (cf. figure 1). In consumer services the privacy aspect is often more important.

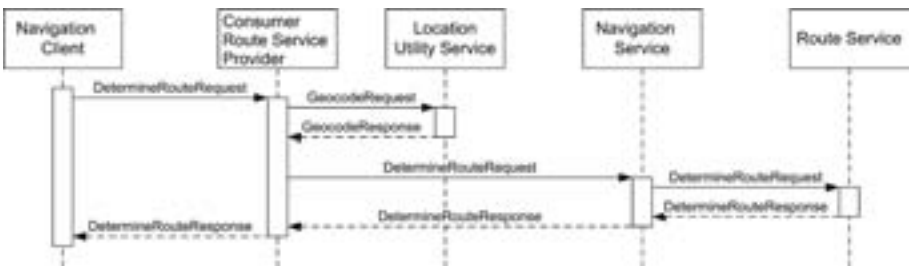

Figure 1: Scenario with multiple service providers

In this scenario the client sends its current position and a target address to the consumer route service provider. The current position is encrypted for the navigation service whereas the address is encrypted for the consumer route service provider. This provider uses a location utility service for geocoding the provided target address in order to obtain coordinates. Then it sends the encrypted position and computed target coordinates to the navigation service. This service in turn is able to decrypt the position and then uses a route service to compute the route. The route is then encrypted for the client and send back via the consumer route service provider to the requester.

To observe location privacy of the client, note that neither the consumer route service provider is able to obtain the current position of the client nor does the navigation or route service get access to the provided address as it may reveal more information than just the coordinates. Also the consumer rote service provider does not get access to the computed route. With only transport level security such privacy could not be achieved as either the consumer route service provider or the navigation service would get access to

all information request and response. Also as the consumer route service provider may be compromised it is also important for the service requester to sign the submitted information in order to obtain a reliable result. If e. g. automated processing is present on the client side it may otherwise be difficult to detect unreasonable responses.

This rather simple consumer-oriented example already shows that application level security is very important in semantically rich services. Therefore being able to encrypt and/or sign individual SOAP elements for specific actors is very important. Also the full message encryption prevents other important features from being used such as less functional intermediaries or specific routing protocols/strategies. Therefore it is important to provide security features beyond transport level.

4 Support for Web Services and Security on Mobile Device Platforms

In our previous work ([BKZ10]) we have already evaluated the extent of native support for implementing SOAP web service clients on some mobile device platforms. In this section we extend this overview to other more recent types of device platforms and also include an analysis how the native support can be extended by using available extension libraries. An overview of supported security configurations and other findings of this section will be presented in table 1 in section 5.

4.1 Windows Mobile 6

At the time of writing Windows Mobile 6 has still been the most recent mobile device platform by Microsoft for which devices have been available. The next generation, Windows Phone 7, is already announced and supposed to be completely redesigned. The findings in this section will have to be re-evaluated once devices are available. But since focus is more on the user interface there is a good probability that the results will be similar.

4.1.1 SOAP Web Services

Devices operating on Windows Mobile 6 which additionally have the .NET compact framework 3.5 or higher can use the native SOAP web service support offered. There are two different bindings available: `BasicHttpBinding` or `CustomBinding`. The former makes several well-known web service standards in previous versions available but has the benefit of being compatible with other web service frameworks on server side such as Apache Axis or Glassfish Metro. The latter offers support for newer versions of e.g. SOAP at the expense of being less inter operable. Support for advanced web service standards such as WS-Transaction is not present in either binding. There are other configurations which are specific for the Windows Communication Foundation and thus by design are not inter operable in heterogeneous systems. We did not analyze these any further as we consider missing interoperability to be a knock-out criterion in a web service based system.

4.1.2 Security for SOAP Web Services

Transport layer security is provided by the platform through SSL/TLS. Note though that the authentication of a service requester is only possible on the HTTP level which is not supported by some of the widely used web service frameworks on server side.

Security on the application level is only supported in a single configuration implementing a subset of the WS-Security 1.0 standard. This configuration is based on an asymmetric security binding based on certificates. More specifically X.509 certificates are used for mutually signing and encrypting messages between requester and provider and in addition for authentication. All these security measures may be used for both message directions, i.e. request and response. This configuration which is also depicted in figure 2 does only support full encryption of the SOAP body of a message and in addition parts of the header. Fine grained securing of individual SOAP elements or parts of the body is not supported. Thus the security configuration for the sample service explained in figure 1 would not be possible. There is still an advantage over transport level security as the authentication issues there would be solved and also additional information in the SOAP header could be used by intermediaries e.g. for routing wihtout being able to access the message body.

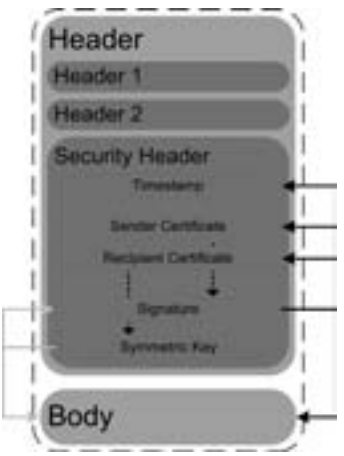

Figure 2: Securing SOAP messages in WCF

4.1.3 Development Complexity

In order to implement a SOAP client for Windows Mobile 6 a client stub generator may be used. It generates client stubs that may be included into the client application thus making the web service access transparent. The possibility to automatically generate client stubs is essential for a future semi-automated service composition. Additionally the stub generation observes WS-Policy specifications.

The client application for a secured service differs only marginally from the unsecured

version. The corresponding client and server certificates for the service have to be loaded in the application code. Also these certificates have to be uploaded to a certificate store on the mobile device previously to calling the service by the device user. The import of the server certificate is also required for transport layer security in case the provider certificate is not signed by a trustworthy root certification authority.

4.1.4 Prototypical Implementation

In order to illustrate supported security configurations we implemented several different variants prototypically. To have a sufficiently general applicability of the results we based the implementation on an OGC-compliant Directory Service (`OGCTopologyService`) which returns cities from a database together with their geographical location and a thematic attribute specified in the request (e. g. population). Only the Windows Mobile based implementation is explained in detail due to space constraints. Similar implementations for the other platforms with their specific security configurations have also been completed.

As explained in section 4.1.2 both message confidentiality and integrity are supported based on an asymmetric binding. On the server side an OGC-compliant version of the service has been implemented which supports mutual authentication between client and server based on certificates. The key in the certificate is also used to encrypt the whole SOAP body as well as security relevant header elements. This has been implemented in Java based on the Metro web service stack. A matching client application in C# for Windows Mobile 6 has already been implemented. In addition to the previously mentioned application-level security it also supports transport-level security.

On client side the implementation uses a specifically designed class `CertStore` which encapsulates the access to the certificate and key store by the application. On one hand it imports keys and certificates from the Windows Mobile native store and simplifies access to them for the client application. On the other hand it is also possible to add specific keys and/or certificates dedicated for this application. Choosing the security configuration to use is dynamic and may be chosen by the end user for illustration purposes at runtime. The corresponding screen is shown in figure 3. In practical applications this will not be left to an end user. But the option for a dynamic decision at runtime facilitates an automatic choice of the best available security option by the application in more complex scenarios.

4.2 Symbian OS with gSOAP

Symbian OS is provided by the Symbian Foundation and has been supported by several important mobile device vendors in recent years, e. g. Nokia, Samsung, Motorola and Sony Ericsson. Even though the number of supporters is decreasing and the platform might be deceasing in some years, it is still a very frequently used device platform nowadays and probably for some years to come.

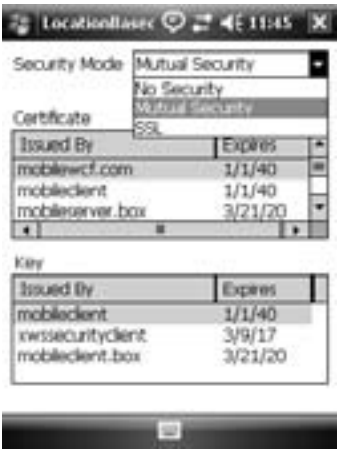

Figure 3: Security Configuration Dialogue in Windows Mobile 6

4.2.1 SOAP Web Services

Natively the Symbian platform does not support SOAP web service clients on mobile devices. But there are several extension options which facilitate a flexible implementation of SOAP web service clients as applications are implemented in C/C++. Thus any C/C++ library for SOAP would be an option. Some extensions have been compared in [BKZ10]. The only of these options that will support security extensions to SOAP later is the gSOAP library. Therefore we only analyze this in further detail.

The library gSOAP is an open-source implementation of a web service framework for C/C++. Thus a client implementation based on gSOAP would also be usable for any other client platform where applications are implemented in C/C++ such as stationary devices or iOS. Our findings should thus also apply for such other platforms. An important aspect making gSOAP the first option on mobile devices is that it operates very resource efficient both in terms of main memory as well as computing power. The gSOAP framework used in our study operates on the WS-I Basic Profiles 1.0a, 1.1 or 1.2. Consequently all HTTP, SOAP and WSDL in version 1.1 have been used.

4.2.2 Security for SOAP Web Services

Security on the transport layer similar to WCF as described in section 4.1.2 is supported by SSL/TLS tunnels provided by a plug-in (`wsse`) based on WS-Security 1.0. In addition to provider authentication as in the WCF, gSOAP also supports requester authentication. Thus for simple scenarios (no fine-grained security, no intermediaries or routing nodes) transport layer security can support message integrity and confidentiality.

More complex services require application layer security. In contrast to WCF there are no fixed configurations for securing SOAP web services in gSOAP. The library supports

both an asymmetric binding based on certificates as well as a symmetric binding based on a pre-shared key. Authentication based on certificates is only available in the asymmetric binding whereas authentication in the symmetric binding is based on username tokens. Message security is available based on XML-Signature by using a gSOAP plug-in (`smdevp`). This plug-in currently only supports message signing (i.e. integrity); encryption of messages (based on XML-Encryption) would have to be implemented manually or might be available in a future version of the plug-in. The signature features are available both for the whole SOAP body as well as for individual elements. Other advanced features which are supported by this platform are timestamps, nonces as well as different types of wsse username tokens.

In summary, the promising approach to support both symmetric and asymmetric binding as well as individual SOAP element security will only become beneficial in the future when XML-Encryption is supported in addition to XML-Signature.

4.2.3 Development Complexity

Similar to the WCF there are tools provided by gSOAP for automated generation of header files and client stubs for secured web services. These may be used inside the client application for web service access. Several security libraries (e. g. `libssl`, `libcrypt`) have to be linked to the application. Unfortunately due to the missing fixed security configurations all required security elements for a service call have to be loaded by the client application manually and then added to the service invocation call. Even though the API to be used is rather straightforward the disadvantage is that the developer of the client application has to be aware of the particular security configuration used and implement it correctly. This significantly increases the development complexity.

Certificate management on a Symbian device is supported by different libraries and APIs, but no off-the-shelf management tool feasible for an end user is provided. Thus either a generic management tool has to be delivered with the client application or additional implementation within the application is required.

4.3 Android

Android is a rather new mobile device platform based on Linux which is recently supported by several major players in the market, e. g. Google, HTC and Samsung. It is developing very fast and several new versions are released every year. Our study is based on version 2.2, but as development focus is on the user interface as well the results should be transferable to more recent versions.

4.3.1 SOAP Web Services

In line with the consumer oriented focus of recent developments for mobile devices there is no native support for SOAP web services in the Android platform. Most consumer oriented

services use rather simple interfaces which can be implemented with more lightweight REST based web services. As we focus on semantically rich services an extension to the platform has to be employed for SOAP services. Since the platform has only been around for a very short time, there is no SOAP extension specifically for Android that we are aware of. This coupled with the Java-like programming on Android led to the idea of using a library originally developed for J2ME SOAP web services to Android; namely we used the Android port of kSOAP2.

The major drawback of kSOAP2 compared to the technologies in the previous sections is that it does not provide automatic client stub generation. This is of particular interest as the services we consider in our study are semantically rich and thus require many different data types on client side which now have to be implemented manually. Besides an increased effort for client implementation this approach does not transfer to a future (semi-)automated service composition.

4.3.2 Security for SOAP Web Services

The library kSOAP2 does not support any security features in SOAP web services. This is true for both transport as well as application layer security. As Android itself does not support SOAP web services at all, there is also no support for secure web services. Consequently in order to have any kind of secure SOAP web services on Android one needs to implement an extension himself. We decided to build an extension of the kSOAP2 library in order to obtain a general purpose security extension.

4.3.3 Security Extension of kSOAP2

In our work we built a layered extension that is added below the classical kSOAP2 extension. I. e. it intercepts the SOAP messages and adds or removes and validates the security information. The layers of our extension are shown in figure 4. On the lowest level important general purpose security features are made available by means of the well-known bouncy-castle library. On top of that these security technologies are applied to messages by using implementations of the XML-Security and XML-Signature specifications. A future extension to use XML-Encryption as well is possible but has been omitted in our prototype due to time constraints. An implementation of a corresponding subset of the WS-Security specification is added in order to include or remove the secured elements in the SOAP messages. Finally there is a component interacting with the classical non-secured kSOAP2 library.

A great advantage of the Android platform in conjunction with the bouncycastle library is the flexibility in using different keystores for storing private keys or public certificates. It is also possible to use the central Android keystore for this purpose. This has the advantage of being able to use the Android standard applications for key and certificate management which is rather easy for the end user. In addition a developer has the choice of either defining a fixed keystore for his application by providing the keystore as resource with the application or to dynamically integrate a keystore in the memory of the mobile device or a storage card.

Figure 4: Extension of kSOAP2 for Security Features

The implementation of the XML-Security layer is based on several Apache libraries which are easily usable on the Android platform as well due to their limited size and compatibility with the Android dialect of Java. On the SOAP security level on the other hand the well-known standard implementations WSS4J and XWSS could not be used on the Android platform. They require too many dependent libraries so that resources become an issue on the mobile device. Also, certain modifications to the dependent libraries are necessary in order to operate properly on Android which would lead to a huge maintenance effort for every update of these libraries. Thus we decided to implement this layer prototypically on our own in order to show the feasibility of a general purpose implementation. Due to time constraints the functionality is limited but extended functionality could be provided in the same manner by mere additional implementation work.

The implementation is able to control the integrity of SOAP messages and also validate the authenticity of the sender of a message. We use an asymmetric binding for this purpose similar to the one explained in section 4.1.2 for Windows Mobile 6. Currently certificates can be used to sign parts of the SOAP message header (i. e. a time-stamp in order to prevent replay attacks) or the full message body similarly to the situation on Windows Mobile 6. Support for message confidentiality could be easily added to the current implementation whereas application of security features for individual elements in the body would be more difficult to add.

5 Conclusion

In this paper we have analyzed the potential of different mobile device platforms to act as clients for secured SOAP web services. The iOS platform has been excluded from our analysis as we consider the lock-in to use the Apple AppStore as the only potential source for client applications to be a knock-out criterion for any business relevant application. We have analyzed the built-in capabilities of the different platforms and also considered and implemented extensions required to support secured web services to a reasonable degree. Table 1 summarizes the supported security configurations of the different platforms and the effort required for the corresponding implementation. Configurations that have

OGCTopologyService variant	**Windows Mobile 6**		**Symbian (gSOAP)**		**Android (kSOAP)**	
		complexity		complexity		complexity
No security	$\surd$	low	$\surd$	low	$\surd$	high
Asymmetric binding (mut. integrity & authenticity)	–		$\surd$	medium	$\surd$	high
Asymmetric binding (mutual integrity, authenticity & encryption)	$\surd$	low	–		◯[a]	
Symmetric binding (integrity & authenticity)	–		–		–	
Symmetric binding (pre-shared key)	–		◯		–	
Transport Layer (SSL and UserToken)	◯		◯		–	

[a] Extension of security layer possible with low additional effort

Table 1: Overview of supported security configurations of the different platforms

been implemented in our work are marked $\surd$ whereas configurations that have not been implemented but should be supported based on documentation are marked ◯.

Table 1 shows that the support for symmetric bindings which would be beneficial due to the reduced resource usage are not supported on all platforms. The only exception is gSOAP with pre-shared keys which might have disadvantages on the deployment side as the keys would have to be distributed off-line. The best solution both functionally as well as effort-wise using an asymmetric binding is provided by the Windows Mobile platform. But it also has its disadvantages as it does not allow for selective encryption of body elements. Any implementation based on kSOAP (Android) requires a very heigh development effort.

From a less technical point of view we have further classified the platforms according to different aspects of implementing secured SOAP clients. These findings are summarized in figure 5 where the scale for each aspect is 0 to 5 with 5 being the best. The scale is qualitative and determined by our analysis, a quantitative assessment is not possible for most dimensions.

A general result to be concluded from the figure is that Windows Mobile 6 has pretty high ratings in most categories because it offers the most advanced features there. As long as the drawbacks (which might nevertheless be very important and significant in practice), namely missing portability and flexibility in the application layer security, are acceptable this platform offers the most features at a rather low effort. On the other hand if specific security configurations on the application level are important and/or portable client application are required, Symbian OS with gSOAP (or other platforms where gSOAP runs) are a better choice at the expense of an increased development effort. At this time the Android platform has a stronger focus on consumer applications which typically do require fewer security features on the application level; thus for the applications we used as foundation

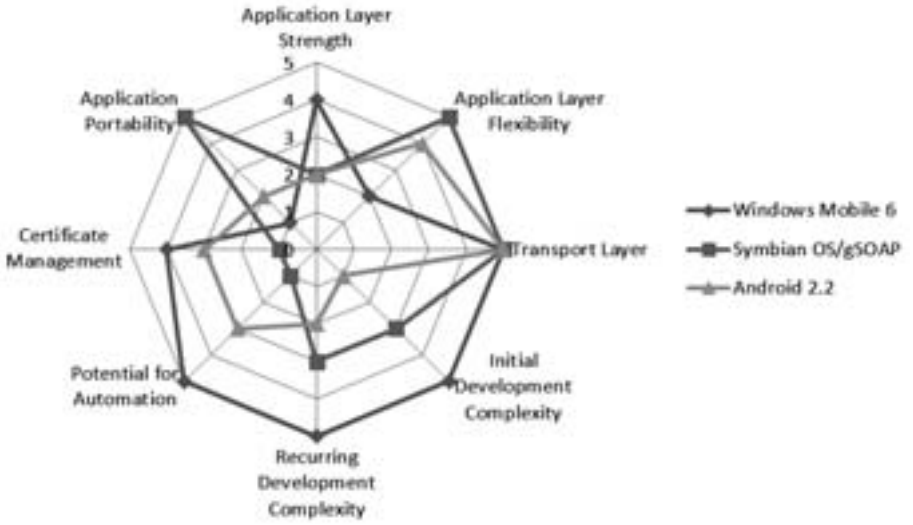

Figure 5: Comparison of Different Platforms for Secure Mobile SOAP Clients

it is not as well suited as the other two. Its strength is the simplicity of building very nice graphical user interfaces which in conjunction with transport-layer security also requires only a small development effort.

6 Future Work

An important result to be obtained from the previous sections is that there is still a lot of room for improvement for secure SOAP web services on mobile device platforms. Improvements could be made either in available functionality and security options (specifically Android and Symbian OS) or flexibility to use the available options as desired (Windows Mobile 6). Such improvements are required to achieve a status that client applications for stationary devices have already reached, be it Java-based clients or other platforms. An important part of the problem seems to be the absence of a general-purpose and portable programming language for mobile device platforms as Java provides for stationary devices. Such advanced features would be required for mobile devices before they can be integrated into enterprise-style services. The potential for using mobile devices as clients in enterprise applications and service-oriented architectures is huge so we expect the development to continue at a very fast pace.

Already today there are new versions of some of the platforms available (Windows Phone 7, Android 3) which would have to be analyzed in a similar way in order to update our results. Also, there are other platforms arising which should be included into out analysis

because they also seem to have a good potential. Finally it would also be interesting to move the study from standard-conformant OGC services to some other type of services which even more closely reflect typical enterprise services.

References

[BKZ10] Jens Bertram, Carsten Kleiner, and David Zhang. Implementation of Generic OpenLS-compliant Web Service Clients for Mobile Devices. In *Proceedings of the 6.GI/ITG KuVS Fachgesprch Ortsbezogene Anwendungen und Dienste*, Schriftenreihe des Geographischen Instituts der Universitt Heidelberg, pages 91–100, 2010.

[EFG+08] A. Ekelhart, S. Fenz, G. Goluch, M. Steinkellner, and E. Weippl. XML security - A comparative literature review. *J. Syst. Softw.*, 81:1715–1724, October 2008.

[GPF09] Nils Glombitza, Dennis Pfisterer, and Stefan Fischer. Integrating wireless sensor networks into web service-based business processes. In *Proceedings of the 4th International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks*, MidSens '09, pages 25–30, New York, NY, USA, 2009. ACM.

[KS07] Tomas Kozel and Antonin Slaby. Mobile devices and web services. In *Proceedings of the 7th Conference on 7th WSEAS International Conference on Applied Computer Science - Volume 7*, pages 322–326, Stevens Point, Wisconsin, USA, 2007. World Scientific and Engineering Academy and Society (WSEAS).

[Mab08] Marwa Mabrouk. OpenGIS Location Services (OpenLS): Core Services. Opengis interface standard, Open Geospatial Consortium Inc., September 2008.

[NGGG07] Anthony Nadalin, Marc Goodner, Martin Gudgin, and Hans Granqvist. WS-SecureConversation 1.3. Oasis standard specification, OASIS, Mrz 2007. http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.html.

[NKMHB04] Anthony Nadalin, Chris Kaler, Ronald Monzillo, and Phillip Hallam-Baker. Web Services Security: SOAP Message Security 1.1 (WS-Security 2004). Oasis standard specification, OASIS, Februar 2004. http://docs.oasis-open.org/wss/v1.1/.

[NKS08] Yuri Natchetoi, Viktor Kaufman, and Albina Shapiro. Service-oriented architecture for mobile applications. In *Proceedings of the 1st international workshop on Software architectures and mobility*, SAM '08, pages 27–32, New York, NY, USA, 2008.

[Ope05] Open Mobile Alliance. OMA Web Services Enabler (OWSER): Overview. Technical report, Open Mobile Alliance, Dezember 2005.

[SJP07] S. N. Srirama, M. Jarke, and W. Prinz. Security analysis of mobile web service provisioning. *Int. J. Internet Technol. Secur. Syst.*, 1:151–171, August 2007.

[SKH08] Holger Schmidt, Andreas Köhrer, and Franz J. Hauck. SoapME: a lightweight Java ME web service container. In *Proceedings of the 3rd workshop on Middleware for service oriented computing*, MW4SOC '08, pages 13–18, New York, NY, USA, 2008.

[WN07] Huaigu Wu and Yuri Natchetoi. Mobile shopping assistant: integration of mobile applications and web services. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 1259–1260, New York, NY, USA, 2007.