Mobile Web services authentication using SAML and 3GPP generic bootstrapping architecture

Kalid Elmufti · Dasun Weerasinghe · M. Rajarajan · Veselin Rakocevic · Sanowar Khan · John A. MacDonald

Abstract In this paper we present a platform for the direct consumption of web services by a Mobile Station. We give an architectural solution where Mobile Operators play the role of Trusted Third Parties supplying service credentials that allow a co-located 3GPP Network Application Function and Liberty-enabled Identity Provider entity to implement a controlled Shopping Mall service to Mobile Stations from multiple trust domains. We consider both the protocol and the structure and syntax of the various tokens required to minimise service latency over the bandwidth and performance constrained mobile system, whilst providing adequate security services to protect against the perceived threat model. To validate our proposal we have developed code to create a Web Service test scenario using SAML authentication tokens utilising readily available J2ME, Java Card, J2SE and J2EE platforms, Web Services tools from Apache, the KToolBar emulator from Sun, and the JCOPS suite of tools for Java Card applet development.

Keywords Mobile Web Services · Authentication · GAA · SAML · 3GPP generic bootstrapping architecture · Mobile Authentication Protocol

This work was supported by sponsorship funding from City University, London. This work was supported by sponsorship funding from Telefonica Móviles, España.

K. Elmufti (✉) · D. Weerasinghe · M. Rajarajan · V. Rakocevic · S. Khan
Mobile Networks Research Group,
School of Engineering and Mathematical Sciences,
City University, Northampton Square, London EC1V 0HB, UK
e-mail: k.elmufti@city.ac.uk

J. A. MacDonald
Information Security Group, Royal Holloway,
University of London, Egham TW20 0EX, UK
e-mail: john@madgo.com

1 Introduction

This paper proposes a protocol for authentication and payment between a consumer and a Web Service Provider that builds upon the Mobile Operator relationship with the mobile subscriber. The proposed scheme enables the Mobile Operator to provide a trusted authentication service that allows a third party to implement an environment where Web Service Providers gain direct commercial access to the Mobile Operator's subscriber base for the consumption of digital and physical products.

But why should the Mobile Operator wish to encourage such access? It has long been noted [6] that distribution structures, and specifically the consumer facing retailing function, evolve as industries mature. Many consider traditional Mobile Operators to be at the early stages of their development as retailers of digital content. The current distribution structures typified by Vodafone Live! from Vodafone, T Zones from T-Mobile, and e-mocion from Telefonica are examples of "one stop shops". Vertically integrated, they source, market and advertise a range of goods to consumers who are encouraged to repeat purchase. They may be considered as analogous to a Department Store on the high street. The typical High Street has evolved, however, and in many cases is complemented (if not replaced) by the Shopping Mall. Comprising both Department Stores and specialist retailers the operator of the Shopping Mall benefits from a large number of customers (i.e. traffic volume) whilst remaining independent from the cost and management of the retailed stock. As the commercial benefit from provision of digital content to mobile consumers transitions from promotional to revenue generating, the "Shopping Mall" concept of digital content retailing may become an attractive model for the traditional Mobile Operator.

2 The web service requirement

Our proposal involves four main actors; the Consumer, the Mobile Operator, the Shopping Mall Operator and the Service Provider.

The consumer is assumed to access the scheme via a bandwidth-constrained Mobile Station, comprising mobile device and service-enabling SIM card connected to a GPRS or UMTS mobile network. Service latency should be minimal without the need to purchase new equipment, and the "purchase experience" should be consistent across all services, irrespective of the actual service provider. Payment for services should be through the normal on-phone and off-phone payment mechanisms. Anonymity is an optional consumer requirement. Service consumption is ad hoc, irregular and transitory in duration.

The Shopping Mall Operator is assumed to require the maximum number of consumers for the available services, and the maximum number of available services for the participating consumers. Service Providers and Consumers should be capable of dynamically and asynchronously entering and leaving the system. The service should be available to consumers from various and disparate trust domains and the service must be terminal vendor independent and capable of being set-up using Over The Air (OTA) techniques.

Finally, Mobile Operator and Service Provider entities will not want to develop new business processes solely for specific Shopping Mall Operators. These entities must interact with the system using standard, internationally agreed protocols.

We base our proposed scheme on the assumption that these requirements are met with a Web Services architecture, where:

- the consumer service endpoint is an OTA installed application running on a mobile device that uses the SIM card as its security element,
- the service content is provided by a Web Services Provider in accordance with internet standards,
- the Mobile operator provides the authentication service, and
- the Shopping mall operator implements a co-located 3GPP network application function and liberty-enabled Identity Provider entity.

3 The proposed scheme

In our proposal the Shopping Mall Operator acts as an Identity Provider between Web Service Providers and each of the Web Service consumers (Mobile Stations). The Mobile Operator owning the SIM deployed in the Mobile Station, acts as an Authentication Authority to the Shopping Mall Operator.

We utilise the combined Liberty & 3GPP GAA model, as defined in [3], to combine the Service Orientated Architecture of Web Services with a Mobile End User end point. We target the provision of identity-consuming services where knowledge of the user (principal) is important. In this way we address the highest value scenario; specifically:

- where the service is enhanced by knowledge of some data related to the identity of the principal (e.g. payment).
- where privacy, trust and authentication are highly relevant.

We consider a federated environment where it is in the principal's interest to re-use such assertions/validations/vouches for access to unrelated services. Our platform implements a permission based access control where the permission level is a function of the "quality" of the initial assertion. Further, we consider the general case where, although the consumer identity is provided by the Mobile Operator customer owning entity, they are given the freedom to attest for the identity of a particular consumer up to "a certain level". Therefore not all assertions are necessarily considered to be of equal quality.

Securing identity is fundamental for Web services security, and as the identity of valid users must move around when information moves from one trust domain to another, and the fact that Web services will be used to cross trust domains makes portable trust an important requirement for Web services security.

Authentication credentials are defined in SAML vocabularies. The Security Assertion Markup Language (SAML) is the XML based security standard created to enable portable identities and the assertion of these identities. SAML is used to exchange authentication and authorization credentials across different security domains [4].

With reference to Fig. 1, in the GSM/3GPP mobile architecture, security and trust reside in two locations. These are the network home location register (HLR) of the home subscriber system (HSS) and the Operator issued tamper resistant

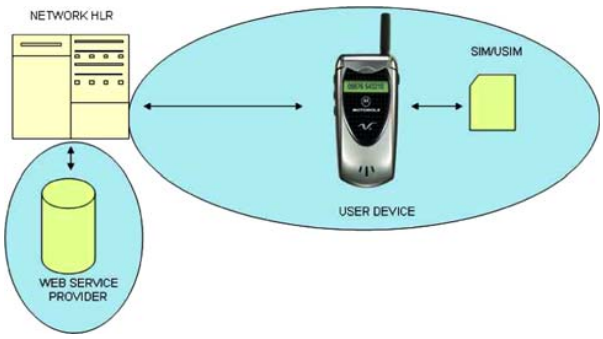
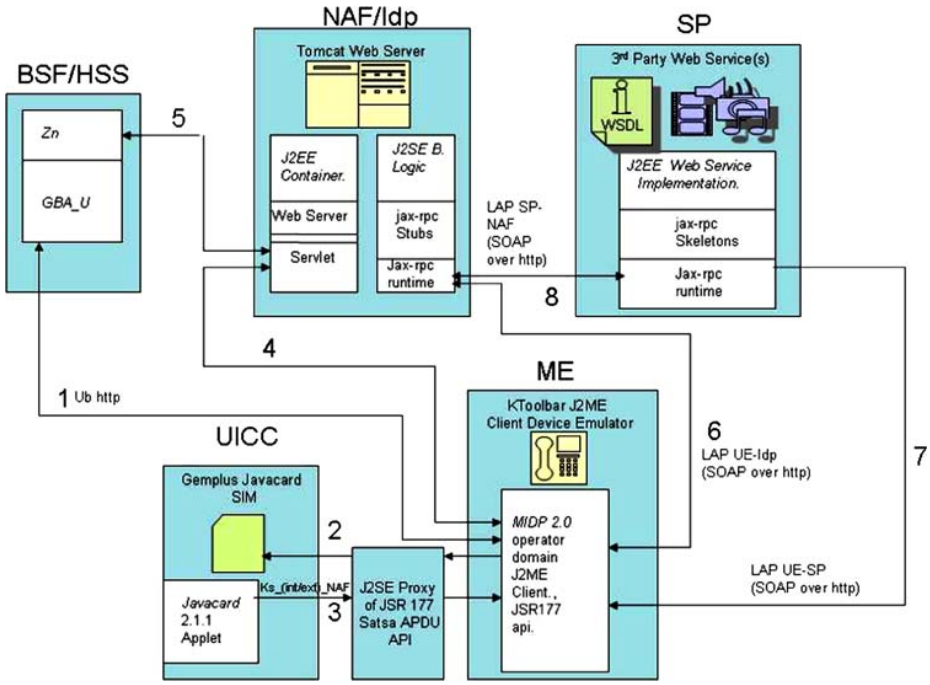


Fig. 1 Security model

Fig. 2 Scheme description



SIM card. We therefore consider the network HSS as the customer owning entity.

A client application needs to run within the user device in order to use the processing capabilities of the user device. However this user device is unlikely to be trusted by scheme entities to hold a valuable network level identity. [Note: This distrust is likely to increase as devices move from traditionally closed proprietary operating systems to more open operating systems capable of performing the file manipulation required by advanced 2.5G and 3G services].

The customer owning entity—the network HLR—attests the identity of a particular consumer up to "a certain level". Application layer credentials are bootstrapped from the (3G) cellular network mutual authentication process and provided to both the End User device and the Service Provider. This allows the Service Provider and End User to communicate securely as they now share the same secret.

We use the GBA or Generic Bootstrapping architecture of generic authentication architecture (GAA) as described in [3] to exploit the 3GPP Authentication and Key Agreement process to produce application credentials. The Mobile Station uses the Bootstrapping Server Function of the Mobile Operator's Home Subscriber System to create these application layer credentials, i.e. GBA, over the Ub interface. These are then shared with the Identity Provider (IdP or sometimes referred to as the Network Application Function) via the Zn interface. The Mobile Station client can then communicate directly with the Service Provider using these credentials.

The use case for a non-registered user begins when the User attempts to purchase a service from an (SP). The (SP) advises the (NAF/IdP) who determines if User is registered. If not registered, the (NAF/IdP) determines if User has the capability, i.e. the "User Agent", in the form of the MIDP2.0 and Javacard application code. If the User Agent is not present then this application code is over the air (OTA) downloaded to the User Equipment in accordance with [8]. The User Agent then registers with the (NAF/IdP) for single sign-on service.

The scheme for a registered user is summarised with reference to Fig. 2.

1. Once registered, the user agent of the (UE) performs GBA_U with (BSF) over Ub.
2. The user agent applet within the UICC is provided with Ub parameters.
3. The UICC component of the user agent calculates the K_s and provides the ME with the service layer credentials $(K_s\_(int/ext)\_NAF)$. The K_s always remains in the UICC.
4. The user agent makes contact with the (NAF/IdP) to obtain a "Shopping Mall" identity.
5. Service credentials appropriate to the User Agent are communicated via Zn to the (NAF/IdP).
6. A SAML authentication token for the "Shopping Mall" is provided to the User Agent from the (NAF/IdP).
7. (UE) communicates with (SP) using service credentials and requests a service.
8. (SP) confirms validity of (UE)'s service credentials.

The (UE) can now purchase from (SP) using On-Phone billing (i.e. via HSS as the payment gateway) or Off-Phone billing (i.e. via a second Service Provider who performs a payment gateway service). The use case continues with (UE) accessing multiple Service Providers until the session is actively terminated either by the User or the (SP).

This process allows the service provider to deliver an identity consuming web service direct to the End User, without having to resort to the use of end user certificates or setting up its own identification system.

The Mobile Station is assumed to implement a *Security Agent* function—an example of which is presented in [8]. The *Security Agent* comprises a device executed MIDlet application for I/O and computationally intensive operations, together with a tamper-resistant module (e.g. Trusted Programme Module (TPM) and/or SIM card) executed application for secure storage and cryptographic processing. The Shopping Mall Operator is assumed to implement a Token distribution centre.

We adopt a push-based model [7] to exchange authentication and payment SAML authorisation tokens between the scheme entities. Tokens are pushed from the Shopping Mall Operator to the Mobile Station, for local storage. This allows a shopping basket of services to be assembled before the tokens are subsequently pushed from the Mobile Station to the Web Service Providers in exchange for their services.

By storing the tokens on the Mobile Station we simulate a familiar shopping behaviour. We allow the consumer to pause (i.e. service interruption) between the phases of entering the Shopping Mall (i.e. authentication), browsing and selecting the goods (web service selection) and proceeding to the checkout (i.e. payment). It is considered good practice [5] to design mobile applications so that they can be interrupted by the user.

4 Implementation options

Web Services are defined [9] as software systems that support interoperable network interactions. They allow implementation of a service-orientated architecture incorporating the entities of Service Provider, Service Consumer and Service Registry. For information to be moved around the network it must be packaged in a format that is understood by these entities. The Simple Object Access Protocol (SOAP) [9] is the standardised packaging protocol currently utilised for Web Services. SOAP supports information exchanges by specifying a way to structure XML messages.

As in any open network environment, these exchanges are exposed to security threats of message leakage, tampering and vandalism. We propose protocol and token implementation options that are designed to resist masquerading, message tampering, replay, and denial of service attacks. Further, as the characteristic of a Web Service is a response to a message, perceived service quality is also dependent on latency between message and response. We therefore also consider the implementation options that affect this.

We present both specific protocol exchanges and the structure and syntax of the authentication and payment tokens.

4.1 Prerequisites for protocol

Our protocol uses both symmetric and asymmetric cryptographic techniques to provide the authentication and integrity services required.

The following requirements must be met prior to the use of the protocol.

- All actors have agreed on a specific signature algorithm. The signature on data X using private key K is written $s_K(X)$.
- All actors have agreed on an asymmetric encryption algorithm, for which the encryption of data X using public key P is written $e_P(X)$.
- All actors except the consumer have encryption key pairs for encryption scheme, and all the actors possess a trusted copy of the public key of the other actors.
- All actors except the consumer have asymmetric key pair for a signature scheme, and all the actors possess a trusted copy of the public key of the other actors.

4.2 Protocol

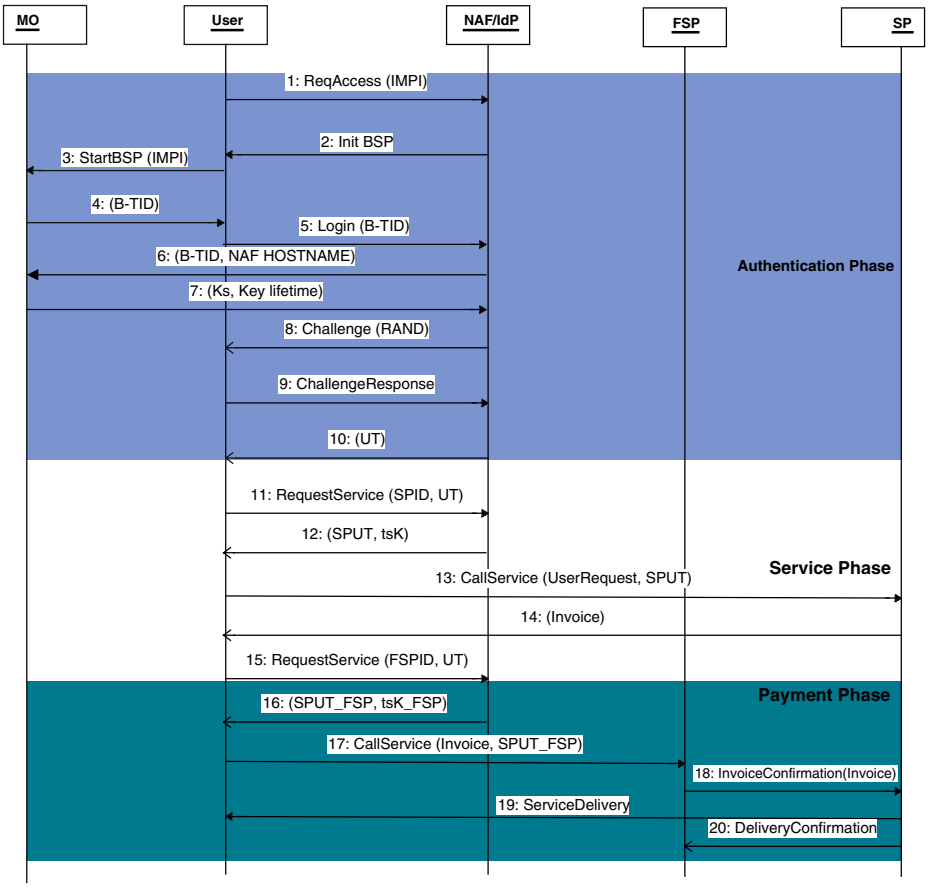
We describe the critical protocol exchanges to address the threat model by considering the authentication, service selection and payment phases of the protocol. Our description assumes that an authenticated key establishment process has taken place between the Mobile Operator and the *Security Agent* of a Mobile Station [8].

We adopt the following additional notation:

SP = Service Provider
IdP = Identity Provider
NAF = Network Application Function
BSF = Bootstrapping Server Function
$User$ = Mobile phone user
WS = Web Service
$IMPI$ = IP Multimedia Private Identity

We have divided the protocol into three sections; Authentication, Service Selection, and Payment, the following subsections will describe each section. Figure 3 provides an overview of the protocol message flow.

Fig. 3 Proposed protocol



4.2.1 Authentication:

1. User to NAF/IdP [ReqAccess(IMPI)]. The User sends a request to access the "Shopping Mall" attaching with the request the User IMPI number.
2. NAF/IdP to User [Init BSP]. Assuming the User has not been authenticated at this stage, the NAF/IdP will send a request to the User to initiate a new Bootstrapping Procedure (BSP).
3. User to BSF [StartBSP(IMPI)]. We assume at this stage that the User does not have a valid bootstrapping session or the freshness of the key material is not sufficient. The User will initiate the BSP with the BSF via the Ub interface, the details are defined in [1].
4. BSF to User [B-TID, Key lifetime]. The BSF will generate B-TID which is a string of base 64 random data and the BSF server domain name; it will also generate key material K_s which is the result of concatenating the Confidentiality Key (CK) and the Integrity Key (IK) resulting from the AKA protocol. The details of the generation of B-TID and the Ks are defined in [2]. The User will use the B-TID as the Username and the K_s as the password to access the NAF/IdP. B-TID will be sent to the User via the Ub interface along with the Key Lifetime, the password K_s will be generated by the user based on the AKA protocol and it will be stored in the UICC.
5. User to NAF/IdP [Login (B-TID)]. The User starts the login procedure by forwarding its 'Username', i.e. the B-TID to the NAF/IdP.
6. NAF/IdP to BSF [B-TID, NAF hostname]. The NAF/IdP needs to obtain the User's password, i.e. K_s that belongs to B-TID in order to be able to authenticate the User. This is done by the NAF/IdP sending the B-TID and its NAF hostname to the BSF via Zn interface, the details of this operation are defined in [2].
7. BSF to NAF/IdP [K_s, Key lifetime]. In response to step 6 the BSF will send to the NAF/IdP the User password i.e. K_s and the key lifetime (Note: other related data will be sent in this message, these data were omitted here for simplicity); the details of this operation are defined in [2] and the security of this message are defined in [1].
8. NAF/IdP to User [Challenge (RAND)]. The NAF/IdP will challenge the User the possession of the password,

i.e. K_s. This step is required to protect against re-play attack. The NAF/IdP generates a random number RAND and sends it to the User.

9. User to NAF/IdP [ChallengeResponse]. After the User receives the RAND, the User will generate the ChallengeResponse and sends it to the NAF/IdP to prove the possession of the password, i.e. K_s. The challengeResponse is a function of the RAND and K_s; ChallengeResponse = f(RAND,K_s); this operation will take place in UICC as K_s will never be reviled to the handset. It is assumed that both the User and the NAF/IdP uses the same function 'f' to generate the ChallengeResponse.
10. NAF/IdP to User [UserToken]. The NAF/IdP needs to verify the ChallengeResponse received in step 9, and if not successful it repeat step 8; if successful it will generate a UserToken (UT) and sends it to the User. The UT will be generated as follows: the IdP part of the NAF/IdP will generate a Temporary User ID (TUID), this will be used to access the SSO system in which the IdP acts as the Authentication Server. The TUID is derived by the IdP from the User ID (UID).
    Note: the NAF IdP mapping is done using a 'User map table', which maps the User's IMPI to the UID (or TUID).
    The UT will be built by concatenating the TUID to a date/time timestamp (TS), and signing the TUID||TS with the IdP digital signature private key $IdP_{ds:sk}$, and encrypting the result with the IdP encryption public key $IdP_{e:pk}$, such that the UT $= e_{IdP_{e:PK}}(s_{IdP_{ds:SK}}$ [TUID||TS]).
    This UT will be sent to the User encrypted using the password K_s received in step 7.

This concludes the authentication phase, all steps can happen at an earlier time before requesting access to any particular third party service provider, providing the lifetime of the keys have not been exceeded.

4.2.2 Service selection:

The proposed protocol for the Service Phase of Fig. 3 is described below:

1. User to NAF/IdP [RequestService (SPID, UT)]. Once the User receives the UT he/she can now request access to any service provider (SP) in the Shopping Mall, however to do that the User must first receive SP UserToken from the NAF/IdP. This is achieved by the RequestService message where the User sends the ID of the requested SP to the NAF/IdP concatenated with the UT.
    The RequestService message will be encrypted with K_s to protect the message confidentiality, the RequestService = e_{K_s}(SPID||UT).
2. NAF/IdP to User [SPUT, tsK]. The NAF/IdP now generates a specific UserToken for the User to be used only with the SP requested by the SPID from the RequestService message; this UserToken will be referred to as the SPUT. The SPUT is built as follows: first a Temporary Session Key (tsK) is generated by the IdP, this will be concatenating with the SPID and the TUID and a new timestamp TS; these data then will be signed with the IdP digital signature private key $IdP_{ds:SK}$, and encrypted with the SP encryption public key $SP_{e:PK}$, such that the SPUT $= e_{SP_{e:PK}}(s_{IdP_{ds:SK}}$[TUID||SPID||tsK||TS]). The SPU T will be sent along with tsK to the User in a message encrypted using K_s.
    It is the creation of the service provider specific SPUT, from the user token UT generated following successful authentication of the user by the NAF/IdP, that provides the user anonymity towards the SP. This is an important aspect of the proposed scheme.
3. User to SP [CallService(UserRequest, SPUT)]. The User can now talk directly with the SP requesting any services offered by this SP, the CallService message will contain the UserRequest and the SPUT. The UserRequest will be encrypted using the tsK to protect the User privacy.
    Note: it is assumed at this stage that when the User sends this message to the SP that the User is confirming his/her selection, which can be indicated in the UserRequest.
4. SP to User [Invoice]. Once the SP receives the CallService message it decrypts the SPUT using its encryption private key $SP_{e:SK}$, it then verifies the signature of the SPUT, this is done by validating the SPUT using the NAF/IdP signature public key $s_{IdP_{ds:PK}}$, to ensure the integrity of the content of SPUT; if the validation is successful the SP compares the TUID (or UID) to its registered Users database if it exist, this option allows the SP to give customized services to its customers. Then the SP gets the tsK from SPUT and use it to decrypt the UserRequest; the SP will reply with an 'Invoice', this Invoice will contain a confirmation of the UserRequest, Price, and a method of payment (e.g. Credit Cards only). The Invoice will be signed by the SP digital signature private key and encrypted with tsK. Invoice= $e_{SP_{tsK}}(s_{SP_{ds:SK}}$ ([UserRequest||Price||MethodOfPayment||TS]))

4.2.3 Payment:

The proposed protocol for the Payment Phase of Fig. 3 is described below:

1. User to NAF/IdP [RequestService (FSPID, UT)]. The User verifies the invoice by decrypting it using tsK, and verifies the content of it. If the Invoice verification process is successful, the User now starts the payment phase. It is assumed that the User has an account with a

Financial Service Provider (FSP), who will charge the User and pay the SP. However for the User to communicate with the FSP the User must obtain a SPUT for this FSP; this is done the same way as in steps 11, 12, and changing the SPID with the FSPID.

2. NAF/IdP to User [SPUT_FSP, tsK_FSP], SPUT_FSP = $e_{FSP_{e:PK}}(s_{IdP_{ds:SK}}$[TUID||FSPID||tsK_FSP||TS]).
3. User to FSP (CallService[Invoice, SPUT_FSP]). The User forward the Invoice and the SPUT_FSP to the FSP in a CallService message, to indicate the User confirmation for the FSP to charge the User and pay the SP indicated in the Invoice.
4. FSP to SP [InvoiceConfirmation(Invoice)]. Similar to protocol message 14 of Fig. 3, the FSP will decrypt and validate the signature of SPUT_FSP (received in step 17) to obtain the tsK_FSP which will be used to decrypt the Invoice, which if successful will indicate the User confirmation to process the Invoice.
   The FSP then charge the Users with the amount stated in the Invoice, and generate an InvoiceConfirmation, which is the Invoice concatenated with the FSPID and a status flag to indicate the statues of the charging, which can only be True (successful operation) or False (unsuccessful operation). The InvoiceConfirmation message will be signed with the FSP signature private key $s_{FSPd_{s:SK}}$ to protect the integrity of the message and to act as a proof of payment. InvoiceConfirmation= $s_{FSPd_{s:SK}}$ (Invoice||FSPID||StatusFlag)
5. SP to User [Service Delivery], once the SP receives the InvoiceConfirmation message from the FSP, it validate the message signature and then checks the StatusFlag, which if set to True, the SP will deliver the service to the User; an optional message can be sent to the FSP to confirm service delivery.

As mentioned above the scheme supports both On-Phone and Off-Phone payment mechanisms. The protocol depicted in Fig. 3 and described in detail above is for the Off-Phone payment mechanism. The On-Phone payment mechanism refers to the case when the user uses the Mobile Operator as a FSP by charging the user's phone bills.

The payment protocol will be exactly as in the Off-Phone case, with the main difference that the FSP will be the MO—the entity that contains the BSF.

4.3 Authentication and payment tokens

Our platform creates a collaborative commercial environment. Central to this is the notion of portable trust, i.e. identity credentials issued in one domain being accepted as proof of the subject's claimed identity (authentication) in another. There exists, therefore many parallel authentication processes. Section 3 and subsequent explanations describe the process adopted for one of them, namely GBA leveraging the 3GPP mobile cellular credentials. To cater for this generic requirement we implement the scheme tokens (e.g. UT) as SAML objects, whose quality rating is based on the value of the attestation that the authentication domain gave the subject. The SAML object, or token, UT is therefore an authentication assertion of the subject (single domain entity), that has been accepted by the NAF/IdP for use within the collaborative commercial environment of the controlled shopping mall (CSM). By issuing a UT to the subject, the subject is now considered a principal (Liberty terminology) within the CSM. The UT is, in essence, the portable identity authentication assertion of the subject. To provide the quality metric, the UT is signed by the NAF/IdP in a way that is appropriate for the attesting authority. This quality metric is a very important element of our platform as it allows many diverse SP's to decide how much to "trust" the principal.

The following is a list of the various tokens deployed in "The Proposed Scheme":

- UserToken (UT=$e_{IdP_{e:PK}}(s_{IdP_{ds:SK}}$[UID||TS])); used by the NAF/IdP only to identify the user in the "Shopping Mall".
- SP UserToken (SPUT=$e_{SP_{e:PK}}(s_{IdP_{ds:SK}}$[TUID||SPID||tsK||TS])); user identifier that is unique for every SP inside the "Shopping Mall".
- Invoice $e_{SP_{tsK}}(s_{SP_{ds:SK}}$ ([UserRequest||Price||MethodOfPayment||TS])); is the payment token.
- InvoiceConfirmation $s_{FSPd_{s:SK}}$(Invoice||FSPID||StatusFlag); used as proof of payment.

  These tokens are implemented as XML objects, as detailed below:
- UserToken

```
<UserToken>
    <UID>String</UID>
    <TimeStamp>Timestamp</TimeStamp>
</UserToken>
```

- SP UserToken

```
<SPUserToken>
    <TempUID>String</TempUID>
    <SPID>String</SPID>
    <TempSessionKey>Key</TempSessionKey>
    <TimeStamp>Timestamp</TimeStamp>
</SPUserToken>
```

- Invoice

```
<Invoice>
    <InvoiceNumber>String<InvoicecNumber>
    <UserRequest>
```

```
            <Item>String</Item>
            <Quantity>int</Quantity>
        </UserRequest>
        <Price>double</Price>
        <TimeStamp>Timestamp</TimeStamp>
    </Invoice>
```

– InvoiceConfirmation

```
<InvoiceConfirmation>
    <Invoice>
        <InvoiceNumber>String<InvoicecNumber>
        <UserRequest>
            <Item>String</Item>
            <Quantity>int</Quantity>
        </UserRequest>
        <Price>double</Price>
        <TimeStamp>String</TimeStamp>
    </Invoice>
    <FinacialSP>String<FinacialSP>
    <Status>boolean<Status>
</InvoiceConfirmation>
```

These XML objects are incorporated in the SOAP messages that exchange information between scheme actors. Example SOAP messages arising from the scheme are presented in Appendix A.

4.4 Proof of concept prototype

To validate our proposal we have constructed a Proof of Concept model, based on the readily available open source tools:

– BSF, NAF_IdP, SP and FSP are deployed as Web Services in Axis (Apache Extensible Interaction System). Axis is a SOAP processor that has been developed as an Apache open source project. Apache Axis 1.3 is deployed on top of Jakarta Tomcat application server and the above Services are deployed in the Apache Axis 1.3. Those services are implemented in J2EE environment.
– A J2ME Client performs the Mobile End User function and is emulated by the Wireless KToolbar [10] from Sun Microsystems, running our *Security Agent* MIDP 2.0 MIDlet on the reference J2ME implementation. The SIM card *Security Agent* function is provided by the JCOPS suite of tools for Java Card applet development.
– Communication between Web Services as well as Web Services and Mobile client has been developed using SOAP messages over http. For authentication SAML tokens were used and are added to the SOAP messages. Axis client is also included in some of the Web Services to invoke services in another Web Service. WSDL document for each web service is created by the Axis Engine.
– According to the protocol, communication between all the entities are secured using java.security and javax.crypto libraries. SOAP messages are signed by XML signature to ensure message integrity. VeriSign's Trust Services Integration Kit is used generate XML signatures.

The WSDL of the simulated controlled Shopping Mall service (NAF IdpService) is presented in appendix B.

The demonstration environment of our proof of concept model is implemented in J2ME and J2EE. J2ME provides the necessary Mobile device simulation and J2EE provide the web service implementation and deployment. The model is designed so that each phase of a specific use case is initiated manually and monitored by visual feedback through the use of J2ME mobile simulator.

5 Conclusion

In this paper we have introduced a scheme for the direct consumption of Web services by a Mobile Station; we described how the Liberty Alliance ID-FF model can make use of the extended authentication services offered by 3GPP GAA to provide a *Controlled Shopping Mall* environment to mobile phone users and Service Providers.

The protocol and the system structure described in this paper, in addition to the syntax of the various security tokens; were used in the development of a "proof of concept prototype" to validate our proposal. The prototype was developed using readily available J2ME, Java Card, J2SE and J2EE platforms.

Our contributions of system architecture, protocols, and enabling data structures could form the basis of a new business model for the changing telecommunication industry. To summarise, our proposal provides:

1. the user with a high level service discovery interface plus anonymity from Web Service Providers;
2. the Mobile Operator with a pivotal role and a revenue generating opportunity in the provision of a web services security and payment platform;
3. the Service Provider with a secure, scalable distribution channel.

In conclusion, our proposal allows developers and researchers to rethink current distribution structures and business models for the sourcing and delivery of digital services to mobile subscribers.

Appendix A: Scheme SOAP messages

SOAP message with the SAML payment token

```
<?xml version="1.0" encoding="utf-8"?> <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Header> <saml:Assertion ID="Asser007"
IssueInstant="2006-09-06T16:38:28.921Z" Version="2.0"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" soapenv:actor=""
soapenv:mustUnderstand="0"> <saml:Issuer NameQualifier="Kalid's
Computer"/> <saml:Subject> <saml:NameIDFormat="URI"
SPProviderID="Kalid's Server"/> <saml:SubjectConfirmation
Method="MethodURI"> <saml:NameID Format="URI" SPProviderID="Kalid's
Server"/> <saml:SubjectConfirmationData InResponseTo="SAML Assertion
Ref ID" NotBefore="2006-09-06T16:38:29.968Z"
NotOnOrAfter="2006-09-06T16:38:29.984Z" Recipient="RecipientURI"/>
</saml:SubjectConfirmation></saml:Subject> <saml:Conditions
NotBefore="2006-09-06T16:38:29.937Z"
NotOnOrAfter="2006-09-06T16:38:29.937Z">
<saml:AudienceRestriction><saml:Audience>Audience
URI</saml:Audience></saml:AudienceRestriction> </saml:Conditions>
<saml:Advice> <saml:AssertionIDRef>Assertion ID
Reference</saml:AssertionIDRef> <saml:AssertionURIRef>Assertion URI
Ref</saml:AssertionURIRef> <saml:Assertion ID="Assertion ID
Reference" IssueInstant="2006-09-06T16:38:29.953Z" Version="2.0"/>
</saml:Advice><saml:AttributeStatement><saml:Attribute
FriendlyName="Attribute Name" Name="AtTrIbUtEnAmE"
NameFormat="AttributeNameURI"/></saml:AttributeStatement>
<saml:AuthzDecisionStatement Decision="Permit"
Resource="WebServiceURI">
<saml:ActionNamespace="ActionURI">AuthorizedAction</saml:Action>
<saml:Evidence> <saml:AssertionIDRef>Assertion ID
Reference</saml:AssertionIDRef> <saml:Assertion ID="Assertion ID
Reference" IssueInstant="2006-09-06T16:38:29.984Z" Version="2.0"/>
<saml:AssertionURIRef>Assertion URI Ref</saml:AssertionURIRef>
</saml:Evidence></saml:AuthzDecisionStatement> </saml:Assertion>
</soapenv:Header>

<soapenv:Body> <ns1:StartBSPResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://soapinterop.org/"> <CallPaymentServiceReturn
href="#id0"/> </ns1:CallPaymentService> <multiRef id="id0"
soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
#xsi:type="UserInformation"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"> <Invoice
xsi:type="soapenc:invoice"> <InvoiceNumber
xsi:type="soapenc:string">String<InvoicecNumber> <UserRequest
si:type="soapenc:userRequest"> <Item xsi:type="soapenc:string">Live
FootBall Clip</Item> <Quantity xsi:type="soapenc:int">2</Quantity>
</UserRequest> <Price xsi:type="soapenc:double">1200.00</Price>
<TimeStamp xsi:type="soapenc:string">04-03-2006 23:12</TimeStamp>
</Invoice> </multiRef> </soapenv:Body> </soapenv:Envelope>
```

SOAP message with encrypted payment token

```
<?xml version="1.0" encoding="utf-8"?> <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body> <xenc:EncryptedData
Type="http://www.w3.org/2001/04/xmlenc#Element"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
<xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
<xenc:CipherData>
<xenc:CipherValue>5PLpoZblWWuCxemqLECvsITG/915oLIOHGQ7CUbsLkb
IjXdDs6Ld/L+/7pRSYbfo9za9L6TUZLdquVNiC7Uw1nrLCrh6sttsCNAfdOsO
CdP0TY9RFucyPPdRAe+KWlZEvMUAr3m33BL6GhdsS67xksgv40B1vgLOublYn
MRXG+kdm9uqJlTad.....wEFYnbbyRB0rWfBtFaw==</xenc:CipherValue>
</xenc:CipherData> </xenc:EncryptedData> </soapenv:Body>
</soapenv:Envelope>
```

Signed and Encrypted SOAP envelope with contents payment token

```
<?xml version="1.0" encoding="utf-8"?> <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Header> <ds:Signature
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"> <ds:SignedInfo>
<ds:CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
<ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
<ds:Reference URI=""> <ds:Transforms> <ds:Transform}
```

```
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<ds:TransformAlgorithm=
"http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments"/>
</ds:Transforms> <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>zPJxl6q7d2qvsVHH4tFgliLlKNM=</ds:DigestValue>
</ds:Reference> </ds:SignedInfo>
<ds:SignatureValue>FMHb70eO5DmN6fH2f4AD7+dGAVhtcgzpQJxtvJsy+SbeItv
5dhETBA==</ds:SignatureValue> <ds:KeyInfo> <ds:X509Data>
<ds:X509Certificate>
MIIC2TCCApcCBENeHwowCwYHKoZIzjgEAwUAMFIxCzAJBgNVBAYTAlVTMRIwE
AYDVQQKEwlXcm94IEJhbmsxGjAYBgNVBAsTEVdyb3ggQmFuayBPbi1saW5lMR
MwEQYDVQQDEwpXZWIg.....ZfcJIkVVRE0rGCxwBczDG <ds:DSAKeyValue> <ds:P>
/X9TgR11EilS30qcLuzk5/YRt1I870QAwx4/gLZRJmlFXUAiUftZPY1Y+r/F9
HTRv8mZgt2uZUKWkn5/oBHsQIsJPu6nX/rfGG/g7V+fGqKYVDwT7g/bTxR7DA
K2HXKu/yIgMZndFIAcc= </ds:P>
<ds:Q>l2BQjxUjC8yykrmCouuEC/BYHPU=</ds:Q>
<ds:G>9+GghdabPd7LvKtcNrhXuXmUr7v6OuqC+VdMCz0HgmdRWVeOutRZT+Z
zwkyjMim4TwWeotUfI0o4KOuHiuzpnWRbqN/C/ohNWLx+2J6ASQ7zKTxvqhRk
Zl6Ae1UlZAFMO/7PSSo= </ds:G> <ds:Y>
l49/FUaZcrInNuQgiNJdPhNhwjlxhifgbh1i56CJ5MWGgsHcEicR2IBPTOO+W
bxCUwELLhWr1oPOYyfhPHOpudyKFN5vLFgLuDCPZyaeBqdFHbqcDaDZ8PSCq6
924DjUt0MAJ2fXhhFmw= </ds:Y> </ds:DSAKeyValue> </ds:KeyValue>
</ds:KeyInfo> </ds:Signature> </soapenv:Header> <soapenv:Body>
<xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
<xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
<xenc:CipherData>
<xenc:CipherValue>5PLpoZblWWuCxITG/915oLIOHGQ7CUbsLkbIjXdDs6L
d/L+/7pRSYbfo9za9L6TULdquVNiC7Uh6sttsCNAfdOsOCdP0TY9RFucyPPdRA
e+KWlZEvMUAr3m33BL6Ghds........yRB0rWfBtFaw==</xenc:CipherValue>
</xenc:CipherData> </xenc:EncryptedData> </soapenv:Body>
</soapenv:Envelope>
```

Appendix B: Service WSDL

A collapsed version of the Web Service Description Language (WSDL) for the proof of concept (NAF IdpService) implementation is presented below:

```
 <?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
targetNamespace="http://localhost:8080/axis/services/NAFService"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://localhost:8080/axis/services/NAFService"
xmlns:intf="http://localhost:8080/axis/services/NAFService"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <wsdl:message name="ReqServiceResponse">
      <wsdl:part name="ReqServiceReturn" type="soapenc:string"/>
   </wsdl:message>
   <wsdl:message name="ReqServiceRequest">
      <wsdl:part name="spID" type="soapenc:string"/>
      <wsdl:part name="userToken" type="soapenc:string"/>
   </wsdl:message>
   <wsdl:message name="ReqAccessRequest">
      <wsdl:part name="impi" type="soapenc:string"/>
   </wsdl:message>
   <wsdl:message name="ReqAccessResponse">
   </wsdl:message>
   <wsdl:message name="LoginResponse">
      <wsdl:part name="LoginReturn" type="soapenc:string"/>
   </wsdl:message>
   <wsdl:message name="ReqLoginRequest">
      <wsdl:part name="tempID" type="soapenc:string"/>
   </wsdl:message>
   <wsdl:message name="ReqLoginResponse">
      <wsdl:part name="ReqLoginReturn" type="soapenc:string"/>
   </wsdl:message>
   <wsdl:message name="LoginRequest">
      <wsdl:part name="loginChallenge" type="soapenc:string"/>
      <wsdl:part name="tempID" type="soapenc:string"/>
   </wsdl:message>
   <wsdl:portType name="NAF_IdPService">
      <wsdl:operation name="ReqAccess" parameterOrder="impi">
         <wsdl:input message=
           "impl:ReqAccessRequest" name="ReqAccessRequest"/>
```

```
        <wsdl:output message=
          "impl:ReqAccessResponse" name="ReqAccessResponse"/>
      </wsdl:operation>
      <wsdl:operation name=
        "Login" parameterOrder="loginChallenge tempID">
        <wsdl:input message=
          "impl:LoginRequest" name="LoginRequest"/>
        <wsdl:output message=
          "impl:LoginResponse" name="LoginResponse"/>
      </wsdl:operation>
      <wsdl:operation name="ReqLogin" parameterOrder="tempID">
        <wsdl:input message=
          "impl:ReqLoginRequest" name="ReqLoginRequest"/>
        <wsdl:output message=
          "impl:ReqLoginResponse" name="ReqLoginResponse"/>
      </wsdl:operation>
      <wsdl:operation name="ReqService" parameterOrder="spID userToken">
        <wsdl:input message=
          "impl:ReqServiceRequest" name="ReqServiceRequest"/>
        <wsdl:output message=
          "impl:ReqServiceResponse" name="ReqServiceResponse"/>
      </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="NAFServiceSoapBinding" type="impl:NAF_IdPService">
      <wsdlsoap:binding style=
        "rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="ReqAccess">
        <wsdlsoap:operation soapAction=""/>
          +<wsdl:input name="ReqAccessRequest">
          +<wsdl:output name="ReqAccessResponse">
        </wsdl:operation>
      <wsdl:operation name="Login">
        <wsdlsoap:operation soapAction=""/>
          +<wsdl:input name="LoginRequest">
          +<wsdl:output name="LoginResponse">
        </wsdl:operation>
      <wsdl:operation name="ReqLogin">
        <wsdlsoap:operation soapAction=""/>
          +<wsdl:input name="ReqLoginRequest">
          +<wsdl:output name="ReqLoginResponse">
        </wsdl:operation>
      <wsdl:operation name="ReqService">
        <wsdlsoap:operation soapAction=""/>
          +<wsdl:input name="ReqServiceRequest">
          +<wsdl:output name="ReqServiceResponse">
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="NAF_IdPServiceService">
      +<wsdl:port binding="impl:NAFServiceSoapBinding" name="NAFService">
    </wsdl:service>
</wsdl:definitions>
```

References

1. Access to network application functions using hypertext transfer protocol over transport layer security. Technical report, ETSI European Telecommunications Standards Institution, June 2005. UMTS, Generic Authentication Architecture (2005)
2. Generic bootstrapping architecture. Technical report, ETSI European Telecommunications Standards Institution, June 2005. UMTS, Generic Authentication Architecture (2005)
3. Interworking of Liberty Alliance ID-FF, ID-WSF and Generic Authentication Architecture. Technical report, 3GPP 3rd Generation Partnership Project, July 2005. 3GPP TR 33.980; Technical Specification Group Services and System Aspect, Release 4 (2005)
4. SAML V2.0 Executive Overview. Technical report, OASIS, April 2005. OASIS Standard (2005)
5. Block, C., Wagner, A.C.: MIDP 2.0 Style Guide. Addison-Wesley, London (2003)
6. Ford, R.: Managing retail service businesses for the 1990s: Marketing aspects. Eur. Manage. J. **8**, 58–66 (1990)
7. Krishna, S.: Web Services Framework and Assertion exchange using SAML. W3C, http://www.w3.org (2001)
8. MacDonald, J.A., Sirett, W.G., Mitchell, C.J.: Overcoming channel bandwidth constraints in secure SIM applications. In: Security and Privacy in the Age of Ubiquitous Computing. Springer Science and Business Media (2005)
9. Snell, J., Tidwell, D., Kulchenko, P.: Programming Web Services with SOAP. O'Reilly, Cambridge (2002)
10. Sun Microsystems, http://java.sun.com/products. Wireless Toolkit, Version 2.1 (2003)