Generic Authentication Architecture

Timo Olkkonen
Helsinki University of Technology
Timo.Olkkonen@hut.fi

Abstract

Nowadays, the owners of modern mobile phones are able to use a vast amount of different services, most of them requiring some kind of authentication. Until now, the user had to manage with different credentials manually, which can become a considerable nuisance, when the amount of used services grows. Also, this hassle with the credentials can also create security problems, for example, as the users use the same, weak passwords over and over again for different services. Also, if the service providers should provide all the necessary credentials to the users, it would be very expensive.

The Generic Authentication Architecture (GAA) is 3GPP's solution to the aforementioned problems. It provides fresh key material for clients and servers that require shared secret based authentication, and signs certificates for those applications which require asymmetric authentication. The users' equipments authenticate themselves to the operator's GAA service by existing 3G or 2G authentication protocols, and in the process receive new keys. Also the services, which the users want to use, are able to fetch them from GAA. This way the clients and servers are able to share secrets.

In addition to other services, the method described above can also be used to authenticate clients to a public key infrastructure, which can then be asked to sign certificates for the client's public key(s).

GAA obviously eases the use of different services for users, but also creates new business possibilities for the operators and service providers. However, there are also problems: the operator might not be trustworthy to authenticate everything, for example electronic voting or banking. Also, the operator is able to bill from authenticating to other providers' services, which increases clients' costs.

KEYWORDS: AKA, authentication, 3G, Generic Authentication Architecture, Mobile authentication

1 Introduction

As digital convergence slowly becomes a reality, the users of modern mobile phones use more and more services on the Internet, such as different WWW sites. Also, the cellular operators have started to offer an increasing number of new services. The user might, for example, read his mail by his school webmail service on the Internet and chat with his friends by using an Instant Messaging and Presence Service[6] offered by his cellular operator. Common to many of these new services is that they require authentication. It is clear that in the aforementioned cases the user does not wish for anyone to be able to read his e-mail or chat under his name. Also, the service provider needs to know who is using the service, at least for billing purposes.

Until now, the user had to have credentials for each service he was going to use. Either he had to use them by hand, e.g. by writing his username and password when challenged or configuring them into a client application, or they were encased in a smartcard, such as the subscriber identity module (SIM) cards in mobile phones. The existence of several credentials is a problem not only because it is inconvenient for the user to manage them, but also because provisioning these credentials to the users is expensive for the operators and other service providers, and it requires effort from the users as well.[1] For example, a third-party WWW-based service provider might send credentials via e-mail to the user and ask him to change the password as soon as possible. This, of course, is not very secure, and requires the user to choose a good password and also to remember it. An example about the costs of provisioning for the operators is the distribution of SIM cards: they have to be manufactured and sent to the clients somehow. They also have to be replaced when new services emerge and require new kind of functionality.

The Generic Authentication Architecture (GAA) is a solution to the growing need for authentication and key agreement between the client and the services on the Internet or in the cellular operators network. GAA tackles the problem stated above by using the already deployed and widely used GSM authentication system as a basis for providing new credentials for both clients and servers. The main idea is that GAA is an authentication service provided by the cellular network operator which allows the client and the service to authenticate each other.[1] However, GAA does not provide a Single Sign On service, just the shared secrets for the both parties.

As one of GAA's main advantages being the ability to use an existing authentication system, the 3G Authentication and Key Agreement (AKA)[12], it is only natural that this advantage is extended to support also the 2G SIM cards and 2G AKA.[3]

The rest of the paper is organized as follows. First, section 2 describers the overall GAA system. Next, section 3 explains the generic bootstrapping architecture (GBA) in detail. Then, section 4 gives a detailed description of the support for subscriber certificates. After this, section 5 explains the differences between the 3G and 2G bootstrapping, and finally, section 6 offers the conclusion.

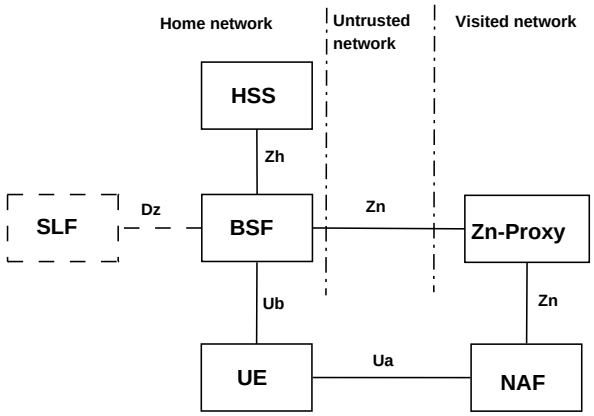

Figure 1: The GAA network entities [5]

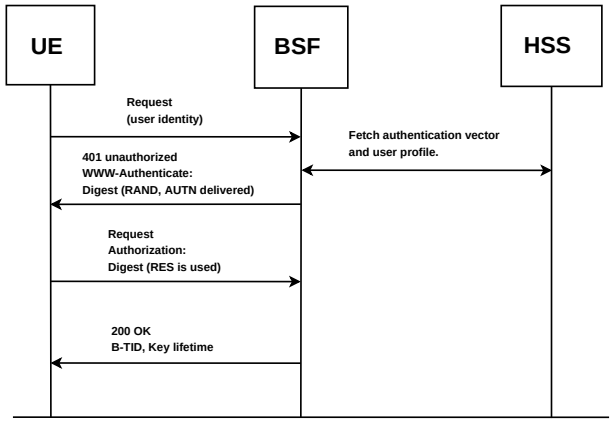

Figure 2: The bootstrapping authentication procedure[5]

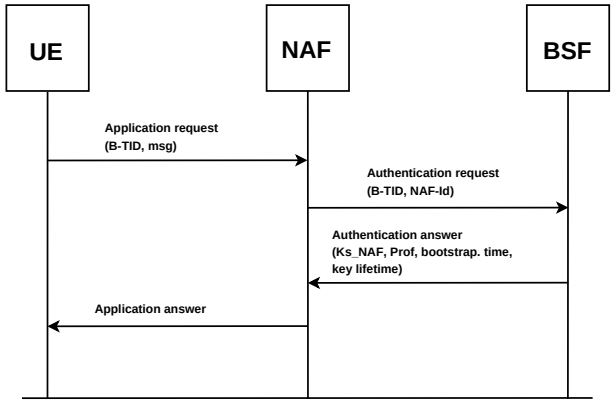

Figure 3: The bootstrapping usage procedure[5]

2 System overview

There are two main ways of using GAA. The first is based on a shared secret between the client and the server, and the second on public and private key pairs and digital certificates. In the shared secret case, the client and the operator are first mutually authenticated by means of the 3G authentication and key agreement (AKA) and they agree on session keys that can be later used between the client and the services which the client wants to use. This is called bootstrapping. After that the services can fetch the session keys from the operator, and they can be used in some application specific protocol between the client and the services.[3]

In the second case, GAA is used to authenticate a certificate enrolment request by the client. First the bootstrapping procedure is carried out as in the previous case. After this the client can request certificates from the operator's PKI infrastructure, where the authentication is done by the session keys obtained by accomplishing the bootstrapping procedure. These certificates and the corresponding key pairs can then be used to produce digital signatures or to authenticate to a server instead of using the session keys.[3]

Fig. 1 shows the GAA network entities and the interfaces between them. Optional entities are drawn with dashed lines and network borders with dotted dash. The user equipment (UE) is, for example, the user's mobile phone. The UE and the Bootstrapping Server Function (BSF) mutually authenticate themselves over the Ub interface, by using the HTTP Digest AKA[8] protocol. The UE also communicates with the Network Application Functions (NAF), which are the application servers, over the Ua interface, which can use any application specific protocol necessary.[5]

BSF retrieves the subscriber's data from the Home Subscriber Server (HSS) over the Zh interface, which uses the Diameter Base Protocol[10]. If there are several HSSs in the network, BSF must first figure out which one to use. This can be done by either configuring a pre-defined HSS to BSF, or by querying the Subscriber Locator Function (SLF) over the Dz interface.[5]

NAFs retrieve the session keys from BSF over the Zn interface, which also uses the Diameter Base Protocol[10]. If NFA is not located in the home network, it shall use a Zn-Proxy to communicate with BSF.[5]

3 Generic Bootstrapping Architecture

The Generic Bootstrapping Architecture (GBA) is described in detail in a 3GPP technical specification[5]. The GBA usage can be divided into two procedures: the bootstrapping authentication procedure and the bootstrapping usage procedure. The bootstrapping authentication procedure consist of authenticating the client to the home network and deriving the key material. In the usage procedure UE tells NAF what key to use, and NAF then fetches this key from BSF. There are also two different mechanisms of using GBA: GBA_ME and GBA_U. The latter is a more secure one which stores keys to the 3G USIM application of the universal integrated circuit card (UICC), i.e the SIM card, instead of the 2G SIM application. However, this requires modifications to UICC. The former mechanism is explained first, and then the differences between it and the GBA_U.

Before UE starts to communicate with NAF, it must figure out whether the GBA is used or not. If UE does not know this in advance, it starts the communication over the Ua interface without any GBA parameters. If NAF requires GBA, it will respond that the bootstrapping initiation is required. The protocol used in the Ua interface is application specific.

When UE knows that GBA is used, it performs a bootstrapping authentication to BSF. Fig. 2 shows the mes-

sage flow chart of the procedure. The details of the used AKA authentication can be found in a 3GPP technical specification[12], and the protocol used is the HTTP Digest AKA[8].

The authentication procedure consists of two request-response pairs between UE and BSF. First, UE sends a request with its username. The BSF then uses this username to fetch the corresponding GBA user security settings (GUSS) and an authentication vector from the HSS. The authentication vector consists of random (RAND), authentication token (AUTN), expected response (XRES), cipher key (CK) and integrity key (IK) values. Of these values BSF sends RAND and AUTN with the authentication challenge response to UE. Then UE runs the AKA algorithms on its SIM, authenticates BSF by verifying the AUTN, and derives the RES value and the session keys CK and IK. After this UE sends the second request with the derived RES value. Then BSF authenticates the user by comparing RES from UE to XRES in the authentication vector. If they match, UE is authenticated, and BSF creates a Bootstrapping Transaction Identifier (B-TID) from the RAND value and the BFS's name. Then B-TID is included in the 200 OK response message to UE. Also included in the 200 OK response is the lifetime of the key material Ks.

Both UE and BSF are now able to create the Ks by concatenating the aforementioned keys CK and IK. However, the actual key material Ks_NAF is computed from the Ks on demand, i.e. in UE when it starts communicating with NAF, and in BSF when Ks_NAF is queried by NAF. The Ks_NAF is created by using a key derivation function.

Fig. 3 shows the message flow chart of the bootstrapping usage procedure. First, UE sends an application request with B-TID to NFA. Then NFA sends an authentication request to the BSF, to get the key material corresponding to the given B-TID. The authentication request also includes NAF-Id, which includes NFA's public hostname used by UE and the Ua security protocol identifier. BSF then verifies that NAF is authorized to use the given hostname. If the verification is successful, and there is a key found with the given B-TID, BSF sends Ks_NAF with its bootstrapping time and lifetime to NAF. In addition to Ks_NAF, NAF may also request some application specific information from BSF. However, if no key is found with B-TID, BSF tells this to NAF, which then sends a bootstrapping renegotiation request to UE. UE must then perform the bootstrapping authentication procedure again.

In the GBA_U case, the idea is that the keys CK and IK never leave the UICC. In the GBA_U bootstrapping procedure, AUTN send by BSF in the 401 authentication challenge message is different than in the GBA_ME case. When UE receives the challenge, the mobile equipment (ME) part of it sends RAND and AUTN to UICC, which then computes CK, IK and RES. Then UICC stores the CK and IK values and gives RES for ME for sending to BSF. After this, BSF and UICC are able to create Ks_NAF like in the GBA_ME case. However, now two keys are created, Ks_ext_NAF and Ks_int_NAF. Both are created with the key derivation function: the former with the same parameters as the single key in the GBA_ME case, and the latter with slightly different ones.

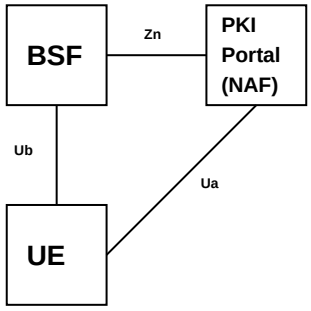

Figure 4: The network elements in the certificate procedures[4]

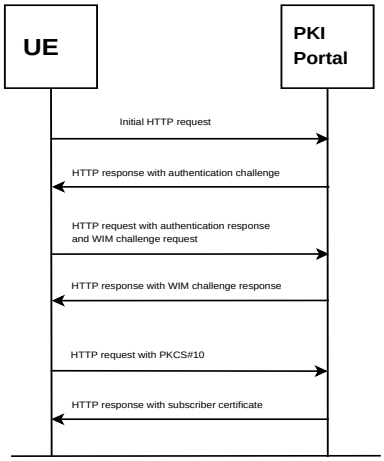

Figure 5: The certificate request using PKCS#10 with HTTP digest authentication[4]

The usage procedure between UE and NAF is basically the same as in the GBA_ME case. UE and NAF agree which type of keys to use, Ks_ext_NAF, Ks_int_NAF or both. The default key is always Ks_ext_NAF, to support UEs and NAFs which do not support GBA_U. Also, NAF tells BSF if it supports GBA_U or not, to get the suitable key(s).

4 Support for Subscriber Certificates

The certificate support in GAA is described in detail in a 3GPP technical specification[4]. It consist of issuing subscriber certificates for UEs and delivering operator CA certificates. A subscriber certificate is a certificate issued to a mobile subscriber. It includes the subscriber's public key and optionally other information, such as the subscriber's identity. A CA certificate is a certificate, which includes the public key of CA. CA uses the corresponding private key to sign the subscriber certificates.

Fig. 4 shows the elements involved in the certificate procedures. The PKI Portal issues the certificates for the UE and delivers the operator CA certificate. The PKI Portal is a PKI Registration Authority, as it authenticates the certification requests from the UE. In addition, it can also function as a PKI Certification Authority and issue certificates. However, there might be an existing PKI infrastructure, which

performs the CA. BSF provides the key material for UE and the PKI Portal as described in the previous chapter, and also PKI Portal specific security data, such as information of what kind of certificates is UE allowed to enrol. UE might have a WAP Identity Module (WIM)[9] in which the private key is stored. WIM is a tamper resistant device, which is capable of providing a proof of that the key is actually securely stored in it. WIM could be, for example, a tamper resistant SIM card.

Fig. 5 shows the message flow chart of the certificate request. First, UE sends an empty HTTP request to the PKI portal, to which the PKI portal responds with a digest challenge. Then UE uses B-TID as an username and Ks-NAF as the password and generates a proper Authorization-header for the next request message. Next, if the certificate is requested for a WIM application in UE, UE creates a WIM challenge request which contains parameters needed for a key proof-of-origin generation. After receiving the request, the PKI portal fetches the corresponding Ks-NAF from BSF with B-TID, thus being able to perform the normal HTTP digest authentication procedure. If extra assurance for the WIM application is needed, the PKI portal may use specific user security settings also fetched from BSF to generate a WIM challenge response. After this UE generates a PKCS#10 request[7]. If a WIM application is used, its identifier and a proof that the key is stored in it is added to the request. If the PKI portal is not a CA, it forwards the request to a one by using some available protocol. After the request has been processed, the new certificate is delivered to the PKI portal. The PKI portal will then send a response to UE which includes either the certificate or a pointer to a one, or a full certificate chain.

Fig. 6 shows the message flow chart of the CA certificate delivery. First, UE starts with an empty HTTP request just like in the subscriber certificate request procedure, and BSF responds with a challenge. After receiving the challenge response, UE creates a new request message with the CA issuer name in the request URL, as specified in[11]. The digest authentication is performed as in the subscriber certificate request procedure. When the PKI portal receives the new request, it authenticates UE and generates a response with the CA certificate. KS_NAF is used to authenticate and integrity protect the response by using the Authentication-info header. After receiving the response, UE must first validate it. If the validation is ok, UE stores the certificate as a trusted CA certificate.

5 2G support

The support for 2G UEs is relevant because, for example, many cellular operators have not delivered 3G SIM cards to their subscribers. The 2G support is described in detail in annex I of the 3GPP GBA technical specification[5]. It is invisible to 3G subscribers, and the GAA network entities are the same as in the 3G case. However, the bootstrapping authentication procedure is different, because the UE is obviously not capable of performing the 3G AKA authentication. Because of this, BSF needs to also be aware of 2G AKA. The usage of the derived key with NAF is just as in the 3G case.

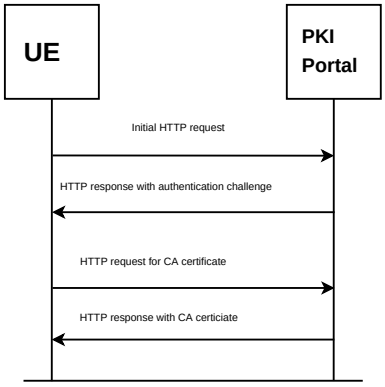

Figure 6: The CA certificate delivery with digest authentication[4]

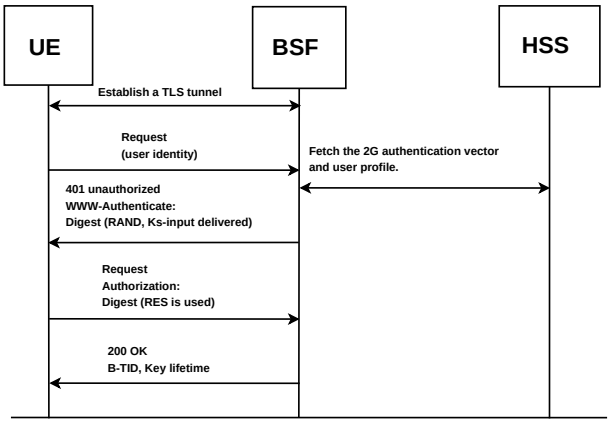

Figure 7: The 2G bootstrapping procedure[5]

Fig. 7 shows the message flow chart of the 2G GBA bootstrapping procedure. First, UE sets up a confidentiality-protected TLS tunnel with BSF. After the tunnel is set, UE sends an initial HTTPS request. After receiving the initial request, BSF fetches GUSS and an authentication vector of the user from the HSS. Then BSF creates a HTTP digest AKA challenge, and sets a Ks-input value to the aka-nonce of the WWW-Authenticate header. After receiving the challenge response, UE performs the 2G AKA procedures and responds with an authorization message. After BSF has verified the authorization, it generates the key material Ks by using the Ks-input, and also B-TID as in the 3G GBA case. Then BSF sends an 200 OK response message including B-TID and the lifetime of Ks. After this the both are able to compute Ks_NAF.

6 Conclusion

Laitinen et al[1] describes the many desirable advantages offered by GAA to the end users: they do not have to create and remember new credentials every time they sing up to a new service, and they can avoid the management of several credentials to different services. They can also easily change UE they are using by, for example, switching the SIM card from a cellular phone to another. Also, the service

providers benefit from GAA as they do not have to provision a huge number of credentials to their users. They also get a vast amount of potential customers from the mobile subscribers, if they start supporting GAA. For cellular operators, the GAA makes new business models possible, for example, they can bill the subscribers and maybe even the service providers for authentication.

However, the ability of the operator to bill everyone can also be seen as a negative thing. Also, because the operator provides keys and certificates, it must be trusted. For example, one might not trust a cellular operator to authenticate users to an electronic voting system, as mentioned by N. Asokan and Lauri Tarkkala[2].

I believe that there is a need for GAA and it does offer solutions to the problems stated in the introduction. Also, the trust issues really don't matter that much in most of the services. Moreover, on-line banking with one's cellular phone would be a killer application for the 3G phones. Unfortunately, at least I do not trust my operator that much.

References

[1] Pekka Laitinen et al; Extending cellular authentication as a service. In *Proc. 3rd Annual Secure Mobile Communications Forum* September 2005.

[2] N. Asokan, Lauri Tarkkala; Issues in Initializing Security. In *IEEE International Symposium on Signal Processing and Information Technology* 2005.

[3] 3GPP TR 33.919; Generic Authentication Architecture (GAA); System description Available: `http://www.3gpp.org/ftp/Specs/html-info/33919.htm`, March 2006.

[4] 3GPP TS 33.221; Generic Authentication ARchitecture (GAA); Support for subscriber certificates Available: `http://www.3gpp.org/ftp/Specs/html-info/33221.htm`, March 2006.

[5] 3GPP TS 33.220; Generic Authentication Architecture (GAA); Generic bootstrapping architecture Available: `http://www.3gpp.org/ftp/Specs/html-info/33220.htm`, June 2006.

[6] Open Mobile Alliance; WV-040 System Architecture Model; Available: `http://www.openmobilealliance.org/release\_program/imps\_archive.html`, January 2005.

[7] PKCS #10; Certification Request Syntax Standard Available: `http://www.rsasecurity.com/rsalabs/node.asp?id=2132` May 2000

[8] A. Niemi et al. RFC 3310 - Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA) Available: `http://www.faqs.org/rfcs/rfc3310.html`, September 2002

[9] Wireless Identity Module Version 1.1; Available: `http://www.openmobilealliance.org/release\_program/wpki\_v10.html`, June 2004

[10] P. Calhoun et al. RFC 3588 - Diameter Base Protocol Available: `http://www.faqs.org/rfcs/rfc3588.html` September 2003

[11] Open Mobile Alliance Wireless Application Protocol Public Key Infrastructure Definition Available: `http://www.openmobilealliance.org/release\_program/wpki\_v10.html` June 2004

[12] 3GPP TS 33.102; Technical Specification Group Services and System Aspectes; 3G security; Security architecture Available: `http://www.3gpp.org/ftp/Specs/html-info/33102.htm` Januray 2006