Ubiquitous One-Time Password Service using the Generic Authentication Architecture

Chunhua Chen[1*] Chris J. Mitchell[2] Shaohua Tang[3]
[1,3] School of Computer Science and Engineering
South China University of Technology
Guangzhou 510641, China
[1] chen.chunhua@mail.scut.edu.cn, [3] csshtang@scut.edu.cn
[2] Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
c.mitchell@rhul.ac.uk

June 29, 2011

Abstract

The Generic Authentication Architecture (GAA) is a standardised extension to the mobile authentication infrastructure that enables the provision of security services, such as key establishment, to network applications. In this paper we first show how Trusted Computing can be extended in a GAA-like framework to offer new security services. We then propose a general scheme that converts a simple static password authentication mechanism into a one-time password (OTP) system using the GAA key establishment service. The scheme employs a GAA-enabled user device and a GAA-aware server. Most importantly, unlike most OTP systems using a dedicated key-bearing token, the user device does not need to be user or server specific, and can be used in the protocol with no registration or configuration (except for the installation of the necessary application software). We also give two practical instantiations of the general scheme, building firstly on the mobile authentication infrastructure and secondly on Trusted Computing. The practical systems are secure, scalable, fit well to the multi-institution scenario, and enable the provision of ubiquitous and on-demand OTP services.

Keywords: One-time password, Generic Authentication Architecture, mobile security, Trusted Computing

*The author is a PhD student at the South China University of Technology. This work was performed during a visit to the Information Security Group at Royal Holloway, University of London, sponsored by the Chinese Scholarship Council and the Natural Science Foundation of Guangdong Province, China (No. 9351064101000003).

1 Introduction and Motivation

A one-time password (OTP) is a means of proving the identity of a user that is only valid for a single authentication session or for a short time period. On-demand OTP systems (e.g. RFC 4226 [16]) typically involve the use of a server-specific security token that shares a secret key with the server and that requires an initialisation process. Such a process costs time and money. Also, if the token is compromised, lost or stolen, then an adversary might be able to use it to impersonate the user to the server, at least until it is revoked. This is likely to increase system support costs, including the possible need to manage a token revocation mechanism. In addition, such schemes do not fit well to a multi-institution scenario, since a user would be required to possess a token for each server with which he or she interacts. Of course, a multi-institution scheme in which a token is equipped with a separate secret key for each of a number of servers is technically feasible; however, such a scheme may be difficult to deploy and manage in practice. These disadvantages increase the management burden and cost, and hence limit the use of OTP systems to security-critical applications.

A major challenge in the management of authentication in networks such as the Internet is the need to establish the necessary credentials. Building an authentication system using a pre-existing security infrastructure is thus a potentially cost-effective approach. One such security infrastructure is the widely used static password authentication infrastructure, in which weak passwords are shared by users and application servers. However, this security infrastructure is vulnerable to a number of attacks. Another pre-existing security infrastructure is the mobile authentication infrastructure, in which a long-term secret subscriber key is shared by a mobile device and its home mobile network. Widely deployed mobile networks include the Global System for Mobile Communications (GSM) and Universal Mobile Telecommunications System (UMTS) networks. The Generic Authentication Architecture (GAA) [5], standardised by 3GPP[1] and 3GPP 2[2], is a general framework that builds on the mobile authentication infrastructure to enable the provision of security services, such as key establishment, to network applications. A third pre-existing security infrastructure of interest here is the Trusted Computing [17] security infrastructure, in which a Trusted Platform Module in a PC is equipped with a certified RSA key pair.

In this paper we first investigate a possible means of extending the Trusted Computing security infrastructure via a GAA-like framework to provide new security services. We then propose a general scheme that converts a simple static password authentication mechanism into a OTP system using the GAA key establishment service. The user 'token' that generates

[1]The 3rd Generation Partnership Project (3GPP).
[2]The 3rd Generation Partnership Project 2 (3GPP2).

the OTP is a GAA-enabled device that is not user or server specific, and can be used in the protocol with no registration or configuration (except for the installation of the necessary application software). We describe two practical instantiations of the general scheme, building firstly on the mobile authentication infrastructure and secondly on Trusted Computing. Our analysis show that the schemes overcome the disadvantages discussed above and enable the provision of ubiquitous and on-demand OTP services.

The rest of this paper is organised as follows. In section 2 we briefly introduce the standard version of GAA that relies on the mobile authentication infrastructure; we then describe the new GAA scheme that builds on the Trusted Computing security infrastructure. In section 3 we review related work. We describe the GAA-based OTP systems in section 4, and discuss their advantages and limitations in section 5. In section 6 we provide an informal security analysis, and in section 7 we draw conclusions.

2 Generic Authentication Architecture

We start by describing the GAA architecture, introducing the main roles in the system and the two main procedures. We follow this by describing two separate implementations of GAA, i.e. as supported by the mobile authentication infrastructure (the standard version of GAA), and our new version of GAA as supported by trusted computing.

2.1 Overview of GAA

Within the GAA framework, the following entities play a role:

- the *Bootstrapping Server Function (BSF)* provides authenticated key establishment services to GAA-enabled devices and application servers, using a pre-existing security infrastructure;
- a *GAA-enabled user device* accesses the security services provided by the BSF using a pre-existing security context;
- a *GAA-aware application server* accesses the security services provided by the BSF, and has the means to establish a mutually authenticated secure channel (e.g. as provided by SSL/TLS) with the BSF.

GAA involves two procedures, *GAA bootstrapping* and *Use of bootstrapped keys*.

- *GAA bootstrapping* involves use of an authenticated key agreement protocol to set up a shared master key *MK* between a GAA-enabled user device and the BSF. Also established is a Bootstrapping Transaction Identifier *B-TID* for *MK* and the lifetime of this key. *B-TID* must consist of a (statistically) unique value which can identify an

instance of *GAA bootstrapping* and the BSF's network domain name. The authenticated key agreement protocol builds on an existing security infrastructure.

- *Use of bootstrapped keys* is a procedure which establishes a server- and application-specific shared secret session key *SK* between a GAA-enabled user device and a GAA-aware application server, using the master key *MK* shared by the user device and the BSF. The procedure operates in the following way. The device first derives a session key *SK* as follows:

$$SK = \text{KDF}(MK, NAF\text{-}Id, \text{other values})$$

where KDF is a key derivation function, and *NAF-Id*[3] is an application-specific value consisting of the Fully Qualified Domain Name (FQDN) of an application server and the identifier of the underlying application protocol. Other values may be included in the key derivation computation depending on the nature of the underlying security infrastructure.

The device then starts the application protocol by sending a request containing *B-TID* to the application server. The application server submits the received *B-TID* and its own identifier *NAF-Id* to the BSF to request the session key *SK*. Note that *B-TID* contains the network domain name of the BSF, so the application server knows where to send the request. As stated above, the application server and the BSF have the means to establish a mutually authenticated secure channel, and hence the BSF can verify the server against its FQDN. If the server authentication succeeds, the BSF derives *SK* from the *MK* identified by *B-TID*, and sends *SK*, its lifetime, and other relevant information to the application server via a secure channel. The user device and the application server now share *SK*, which they can use to secure application-specific messages.

2.2 UMTS GAA

The standard versions of GAA [5] build on the mobile authentication infrastructures (including the GSM or the UMTS infrastructures). In this paper we focus on GAA as supported by the UMTS authentication infrastructure, which we refer to as UMTS GAA.

As shown in Figure 1, the following UMTS-specific entities play a role in UMTS GAA:

[3]In the GAA specifications [5], the functionality of a GAA-aware application server is referred to as the Network Application Function (NAF).

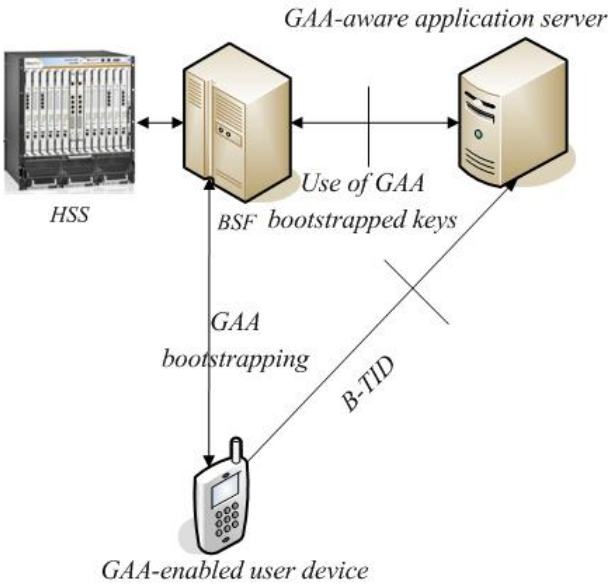

Figure 1: UMTS GAA framework

- the Home Subscriber Server (HSS) is provided by a network operator, and has access to the long-term key for each of the operators's subscribers;
- the BSF connects to the HSS and acts as an intermediary between the HSS and a GAA-enabled mobile device;
- the GAA-enabled user device is a UMTS-enabled mobile device, which shares a long-term subscriber key with its HSS.

The UMTS Authentication and Key Agreement (UMTS AKA) protocol [1] provides authentication and key establishment using a long-term secret subscriber key K shared by a mobile device (strictly its UMTS Subscriber Identity Module (USIM)) and its HSS. As a result of a successful UMTS AKA procedure, a pair of secret session keys is shared by the mobile device and its HSS. These keys are CK, used for confidentiality protection, and IK, used for integrity protection. Note that, in the UMTS AKA procedure, a random challenge ($RAND$) is sent by the HSS to the mobile device.

UMTS GAA bootstrapping, as defined in the Generic Bootstrapping Architecture (GBA) [5], uses the UMTS AKA protocol to set up a GAA master session key (MK) between a GAA-enabled mobile device and the BSF, where MK is the concatenation of IK and CK. The BSF also sends an identifer $B\text{-}TID$ for MK and the MK lifetime to the mobile device. $B\text{-}TID$ consists of a combination of the $RAND$ value and the BSF's network domain name.

The server- and application-specific session key SK is derived as follows:

$$SK = \text{KDF}(MK, \text{GBA\_variant}, RAND, \text{IMPI}, NAF\text{-}Id)$$

where GBA_variant indicates the bootstrapping variant (such as GBA_ME or GBA_U), and the IP Multimedia Private Identifier (IMPI) is derived from the International Mobile Subscriber Identity (IMSI) [4], which is unique to each mobile device (strictly its USIM).

2.3 TC GAA

A Trusted Platform (TP) compliant with the Trusted Computing Group (TCG) specifications is a computing platform with a built-in Trusted Platform Module (TPM) [19, 20, 21]. A TPM is a self-contained processing module with protected capabilities, including random number generation, asymmetric key generation, digital signing and encryption. A TPM is equipped with a unique Endorsement Key (EK) pair, an RSA key pair that is used only for encryption/decryption. The private decryption EK is known only to the TPM and never disclosed, while the public encryption EK may be revealed externally. A variety of other categories of keys can be generated and used by a TPM, and a range of types of public key certificate (and associated certification authorities) are defined in the trusted computing specifications. We use the term Trusted Computing (TC) security infrastructure to refer to the set of deployed TPMs, the associated keys, and the supporting public key infrastructures.

We next outline a possible means of using this security infrastructure to support a GAA security framework, which we refer to as TC GAA. Note that we give only a high level description here; a detailed description of TC GAA is provided in a separate document [8].

As shown in Figure 2, the following Trusted Computing specific entities play a role in TC GAA:

- the supporting public key infrastructures;
- the BSF, which is equipped with a certified signature key pair used for entity authentication;
- the GAA-enabled user device, a TCG complaint TP whose TPM has been equipped with a certified signature key pair (e.g. an Attestation Identity Key (AIK) generated by the TPM) for entity authentication. The TP is also capable of implementing the authenticated key agreement protocol which forms part of the GAA bootstrapping procedure.

The TC GAA scheme operates as follows.

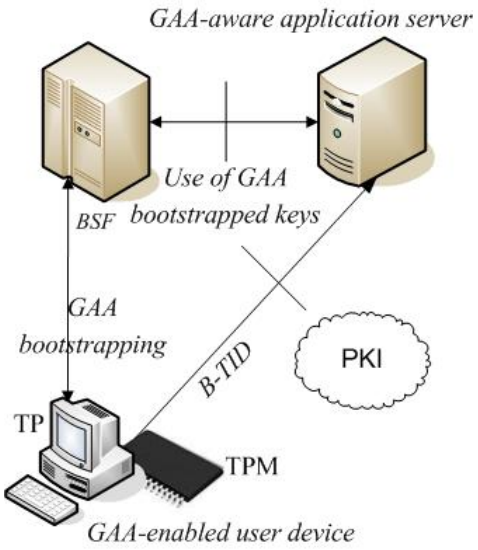

Figure 2: TC GAA framework

- The TC GAA bootstrapping procedure involves use of an authenticated key agreement protocol employing the signature key pairs possessed by the BSF and the TP. The protocol conforms to the two-pass unilateral authentication protocol specified in clause 5.1.2 of ISO/IEC 9798-3:1998 [12]. In every protocol run a new temporary asymmetric key pair is generated by the TPM. The public key of this pair is cryptographically bound to a TP identity Id_{TP} using the TPM's private signing key; the resulting certificate is sent to the BSF. The BSF generates a secret session key MK, encrypts it using the temporary public encryption key bound to Id_{TP}, and sends it to the TP together with a key identifier $B\text{-}TID$ and the key lifetime. After successful completion of the protocol, during which a random challenge R is sent by the BSF to the TP, the session key MK is shared by the TP and the BSF. Note that the key identifier $B\text{-}TID$ consists of a combination of R and the BSF network domain name.

- In the TC GAA Use of bootstrapped keys procedure, the TP and the GAA-aware application server follow the procedure defined in section 2.1 to establish a server- and application-specific session key SK using the master key MK shared by the TP and the BSF. The session key SK is derived as follows:

$$SK = \mathrm{KDF}(MK, R, Id_{TP}, NAF\text{-}Id).$$

We observe that TC GAA provides a way of exploiting the now very widespread trusted computing infrastructure for the provision of fundamen-

tally important generic security services. Of course, application-specific security protocols building on the infrastructure can be devised independently of any generic service and, indeed, there is a large and growing literature on such schemes. However, the definition of a standard GAA-based security service enables the trusted computing infrastructure to be exploited in a simple and uniform way, and it also provides an opportunity for trusted computing aware third parties to provide novel security services. This may help with providing the business case necessary for the emergence of the wide range of third party security services necessary to fully realise the goals of trusted computing.

3 Related Work

An application of GAA for HTTP digest authentication [9] has been described by 3GPP [2, 3]. In this scheme, the username and password in normal HTTP Digest authentication are replaced by the GAA's *B-TID* and an application-specific key, respectively. At the application server, a user is identified through the identity of his or her GAA-enabled mobile device. This requires that the GAA-enabled device involved is user-specific (which is not true for the scheme we describe below).

Alzomai and Josang [6] proposed a multi-institution OTP scheme using Trusted Computing, in which a mobile phone equipped with a Mobile Local-Owner Trusted Module (MLTM) [18, 22] acts as an OTP generator. A configuration procedure is needed to generate and maintain a user-specific secret key shared by the MLTM and the application server, which is used by the MLTM to generate a counter-based OTP sequence. In this scheme, as is the case for the 3GPP scheme described above, an adversary in possession of the user-specific device could misuse it to impersonate the legitimate user.

Molva and Tsudik [15] were the first to propose the use of a non-user-specific security token (card) for user authentication. A token shares a strong cryptographic key with an authentication server, and is used solely to provide a secure channel between a human user and the server. Since the token is not associated with any particular user, the scheme is resistant to token compromise and avoids the need for a token-specific user registration procedure. However, it requires every server to securely distribute a key-bearing token to every user, which is likely to be a significant burden in practice, as discussed in section 1.

Holtmanns et al. [11] (section 4.2.1) proposed a OTP scheme designed for use in authenticating users to a corporate network, and which has similarities to the scheme described in section 4.3. Both schemes generate OTPs from GAA bootstrapped keys and a long-term user password. However, the OTPs (actually secret keys) in the Holtmanns et al. scheme are used for mutual authentication, whereas in the scheme of section 4.3 they are employed solely

for user authentication.

The GAA OTP scheme is designed to be integrated into existing authentication infrastructures in a simple and cost-effective way. It converts static password authentication into an OTP scheme using the GAA service, and requires only small changes to the client and the server, as described in sections 4.3 and 4.4.

4 GAA-based OTP Systems

We first describe a general scheme, GAA OTP, that uses the GAA architecture described in section 2.1 to convert a static password authentication system into an OTP system. We then specify two practical instantiations of this general scheme, building on UMTS GAA and TC GAA.

4.1 The General GAA OTP Scheme

In the general scheme, the following entities play a role:

- a user U with identifier *username*;
- a GAA-aware application server S, which shares a user-specific secret password pw with every user U, and whose clock is synchronised with that of the BSF;
- a client application C, used by U to access S;
- a BSF that provides the GAA service;
- a GAA-enabled user device T that can access the GAA service provided by the BSF.

Figure 3 summarises the general GAA OTP protocol. We next give a full description, referring to the step numbers given in the figure.

When U wishes to access S, U directs its client C to S. U then causes T to engage in a GAA bootstrapping procedure with the BSF (step 1). After successful execution of this process, the values *B-TID*, *MK*, *MK* lifetime, and other relevant values are shared and cached by T and the BSF. T then derives a session key SK, as described in section 2.1 (step 2). Note that SK is not specific to U, and cannot be used to authenticate U to S.

After derivation of SK, T computes an authenticator otp as a function of SK and pw (step 3), i.e.

$$otp = f(SK, pw).$$

The function f can be implemented in many ways. One possibility is to instantiate f using HMAC [14] based on a suitable cryptographic hash function. That is, otp could be computed as:

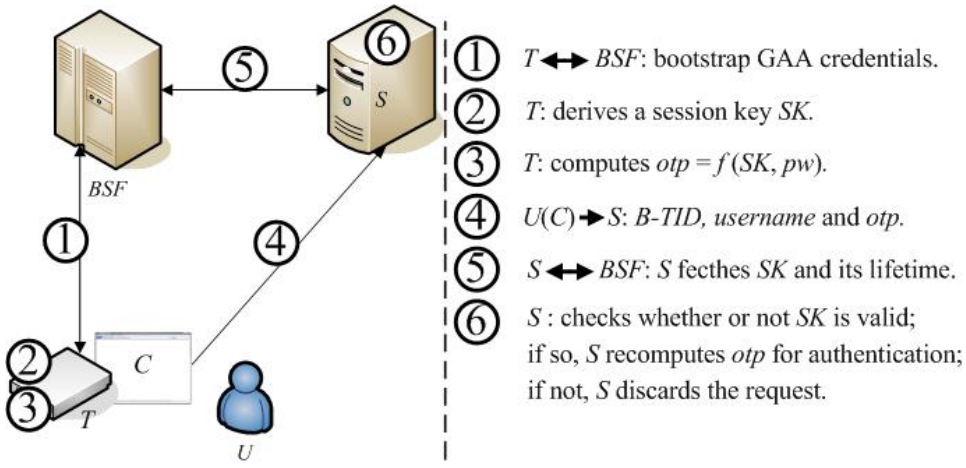

Figure 3: General GAA OTP protocol

$$otp = \mathrm{HMAC}_{SK}(pw).$$

The user password pw should not be cached by T, and the keys MK and SK and other related information should be deleted after computation of otp.

The server authenticates the user U by requiring the values *B-TID*, *username*, and otp to be submitted via the communication channel between C and S (step 4). To verify the received otp, S engages in the Use of bootstrapped keys procedure by sending the received *B-TID* and its own identifier *NAF-Id* to the BSF; it receives in return the same SK as available to T together with its lifetime and other relevant information (step 5). The SK lifetime must be set to be the same as that of MK. Before recomputing otp, S must check whether or not the current system time of S is within the SK lifetime. If not, SK is invalid and U is rejected; otherwise, S now uses the received SK to recompute otp. If the recomputed otp and the otp submitted by U match, U is granted access (step 6).

If the lifetime of SK is chosen to be sufficiently short then the scheme has the one-time property, since otp is only valid for a short time. In practice the lifetime of MK and SK will depend on the application's specific security requirements.

4.2 Possible variants

In the scheme described above, we can arrange for an otp to be usable only once simply by changing the checking performed by S. This can be achieved by deeming otp to be valid only if it is computed using a fresh SK. To ensure this S must check whether or not the received *B-TID* has been used previously. Note that this requires S to cache all the *B-TID* values it

receives until their corresponding SK expires.

Note that the timeliness of an authenticator otp is controlled by the lifetime of SK. To prevent an otp from being accepted at a much later time, the lifetime of SK must be sufficiently short. However, SK could be used as a long-term authentication secret key, with the necessary management mechanisms in the user device. In such a case we could modify the protocol to achieve the same security objectives by computing otp as a function of SK, pw and a time variant parameter var, i.e.

$$otp = f(SK, pw, var)$$

where the time variant parameter is either a time stamp or a sequence number. The system would then work as follows.

The user U submits the values *B-TID*, *username*, *var*, and *otp* to S. In addition to verifing the validity of SK, the application server S must verify the freshness of the time variant parameter, i.e. it must verify the validity of the timestamp or sequence number. Whilst such a change would reduce the use of GAA bootstrapping procedure, it would also incur additional management overheads, namely the requirement for U and S to either have synchronised clocks or maintain bilateral sequence numbers.

4.3 The UMTS GAA OTP System

We next describe a practical instantiation of GAA OTP building on UMTS GAA. In this system, referred to as UMTS GAA OTP, C and T have the following UMTS-specific properties.

- The client application C (e.g. a web browser) resides in a personal computer. We assume that an application supporting the system has been installed in this PC (e.g. as a Java applet/browser plug-in). The scheme-specific application must be aware of the FQDN of the GAA-aware application server S and the identifier of the application protocol.

- The GAA-enabled user device T is a UMTS-enabled mobile phone. We assume that an application supporting the system has been installed in T. We also require T to possess a means of communication with the scheme-specific application in the user PC in order to exchange the necessary information, e.g. as provided by a USB cable or a Bluetooth link. The application is assumed to be capable of accessing the GAA session key SK in order to compute the authenticator otp, as described in section 4.1.

Figure 4 summarises the UMTS GAA OTP protocol. We next give a more detailed description, referring to the step numbers given in the figure.

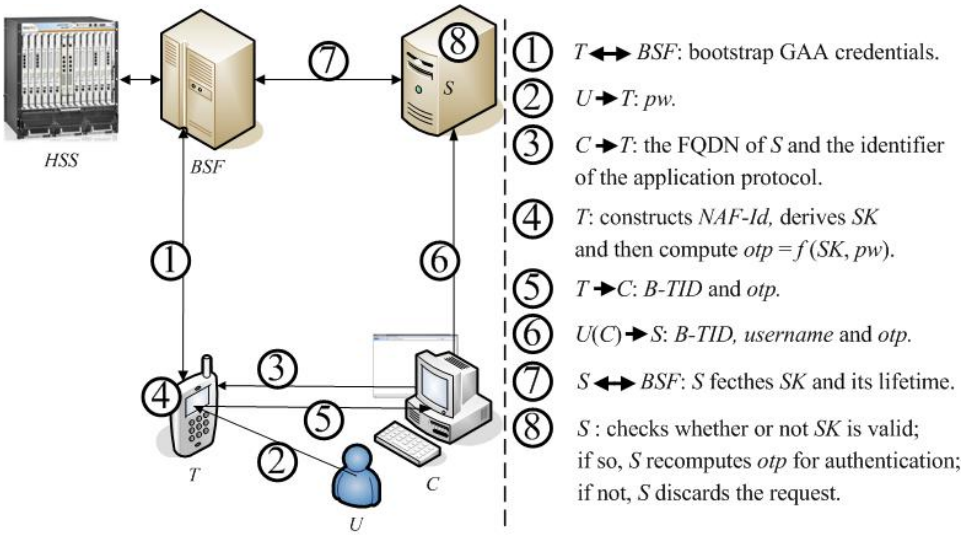

Figure 4: UMTS GAA OTP protocol

When U wishes to access S, U directs the browser C to S. U then causes T to engage in a UMTS GAA bootstrapping procedure with the BSF (step 1). After successful execution of this process, the values *B-TID*, *RAND*, *MK*, and the *MK* lifetime are shared and cached by T and the BSF.

T now prompts the user for the password *pw* (step 2), and requests from C the FQDN of S and the identifier of the application protocol (step 3). T next constructs *NAF-Id* and derives a session key *SK*, as described in section 2.2; T now uses *SK* and *pw* to compute an authenticator *otp* (step 4). T then sends to C the values *B-TID* and *otp* (step 5). U submits the values *B-TID*, *username* (which must be input by U at some point), and *otp* to S (step 6). The remaining steps of the protocol involve S fetching GAA credentials (step 7) and S verifying the request (step 8), which are the same as steps 5 and 6 of the GAA OTP, as described in section 4.1.

4.4 The TC GAA OTP System

Finally, we describe a practical instantiation of GAA OTP building on TC GAA. In this system, referred to as TC GAA OTP, C and T have the following TC-specific properties.

- The GAA-enabled user device T is a personal computer with a TCG compliant TPM.
- The client application C (e.g. a browser) resides in the GAA-enabled user device. We assume that an application supporting the system has been installed in C (e.g. as a Java applet/browser plug-in). The

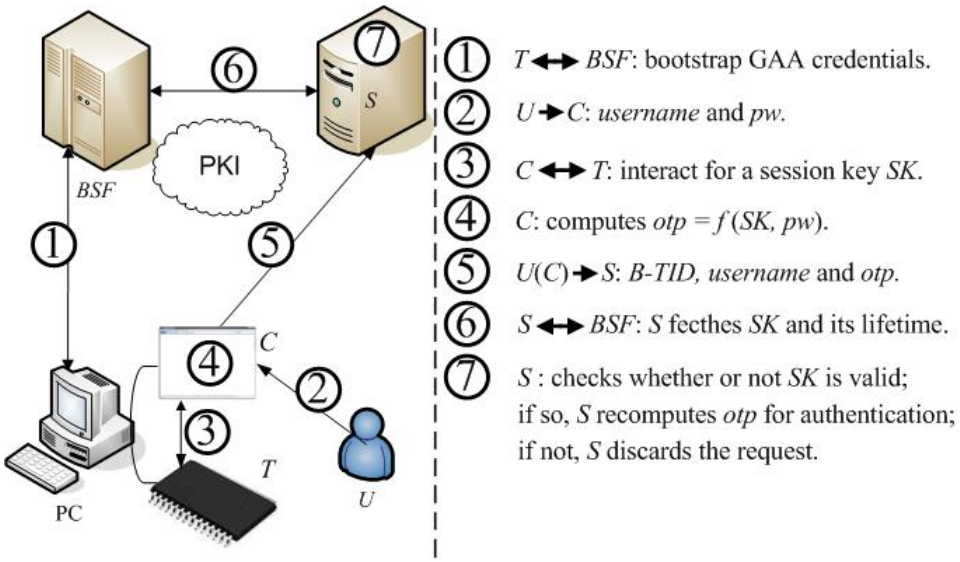

Figure 5: TC GAA OTP protocol

scheme-specific application must be aware of the FQDN of the GAA-aware application server S and the identifier of the application protocol. The application is assumed to be capable of accessing the GAA session key *SK* in order to compute the authenticator *otp*, as described in section 4.1.

Figure 5 summarises the TC GAA OTP protocol. We next give a more detailed description, referring to the step numbers given in the figure.

When U wishes to access S, U directs the browser C to S. C (as instructed by U) now causes T to engage in a TC GAA bootstrapping procedure with the BSF (step 1). After successful execution of this process, the values *B-TID*, *R*, *MK*, and the *MK* lifetime are shared and cached by T and the BSF.

C now prompts the user for the *username* and the password *pw* (step 2). C asks T for a session key *SK* by providing the FQDN of S and the identifier of the application protocol; in response T constructs *NAF-Id* and derives a *SK*, as described in section 2.3. C receives back from T the *SK* and the corresponding *B-TID* (step 3). C now uses the *SK* and *pw* to compute an authenticator *otp* (step 4). U (or C acting on behalf of U) submits the values *B-TID*, *username*, and *otp* to S (step 5). The remaining steps of the protocol involve S fetching GAA credentials (step 6) and S verifying the request (step 7), which are the same as steps 5 and 6 in the GAA OTP scheme, as described in section 4.1.

5 Advantages and Limitations

Unlike most OTP systems using a dedicated key-bearing token, the system does not require an initialisation process. That is, the GAA-enabled device (i.e. the TP or the mobile device) is neither user nor server specific, and can be used in the protocol with no registration or configuration (except for the installation of the necessary application software). This approach fits well to the multi-institution scenario. The system enables server and application-specific session keys to be generated using a GAA-enabled device, where each such key can be used to help authenticate a user to the appropriate GAA-aware server. The GAA-enabled device thus acts as a multi-institution OTP token. Moreover, since a GAA bootstrapped session key is used in the computation of the authenticator, there is no need to generate and securely distribute a key-bearing token to every user, which is a potentially major management overhead for dedicated OTP token schemes.

To employ UMTS GAA OTP, a user needs only a GAA-enabled mobile phone with a valid subscription, and there are a very large number of subscription holders across the world. In the case of TC GAA OTP, a GAA-enabled TCG compliant TP is needed. According to a report published by TCG [10], in 2008 over 200 million TPMs were shipped in original equipment manufacturer (OEM) platforms. The TCG further report that it is expected that, by the end of 2011, TPM penetration in OEM platforms will be more than 80 percent. Both systems thus have good scalability. We observe that the use of a third party server (the BSF server) might give rise to scalability issues, and could act as a single point of failure. In practice, the BSF service could be deployed in a distributed manner; that is, a set of BSF servers (with distinct network domain names) could be used to provide the GAA service. This commonly used approach gives a degree of fault tolerance and removes the single point of failure.

The most commonly employed user authentication mechanism for browser access to a website is almost certainly the static password, possibly sent via an SSL/TLS-protected channel. The two systems we have described can be deployed to seamlessly convert a static password scheme into a OTP system with only small changes required to the server and client.

The UMTS GAA OTP scheme requires a connection (e.g. as provided by a USB cable or Bluetooth link) between the browser and the mobile phone. This poses a challenge to the scheme's usability and portability. We are currently working on prototyping the UMTS GAA OTP scheme and certain variants of it. One such variant is a so-called multi-channel scheme, which has been developed specifically to address this usability challenge. This issue is discussed further in section 7.

We observe that the server and user must both trust the party which operates the BSF not to compromise long-term user passwords by intercepting client-server communications and performing a brute force search (see also

section 6). Such a trust relationship could be supported by a contractual agreement.

Finally, use of the scheme will incur the cost of using the GAA service. However, the variant schemes described in section 4.2 could be employed to reduce use of the GAA bootstrapping process, and hence reduce the cost overhead.

6 Informal Security Analysis

The security of the schemes we have described relies on the security of the underlying UMTS GAA and TC GAA procedures. In turn, the security of UMTS GAA is built on the assumption that learning the subscriber key and/or MK by attacking UMTS AKA is not possible [11]. Similarly, by using a secure authenticated key agreement protocol in TC GAA (and there are a number of provably secure protocols [7]), we can ensure that it is not possible to learn MK by attacking the operational protocol [8].

A further requirement for use of the scheme is that the application server and the BSF have the means to set up a mutually authenticated secure channel, which implies that the BSF has the means to authenticate the requester against the FQDN. As a result, an adversary can neither obtain SK by monitoring the link between S and the BSF, nor request the SK intended for S from the BSF, since the adversary cannot claim ownership of the FQDN for this SK.

In both schemes, the GAA-enabled device (i.e. the TP or the mobile device) must be trusted by the user, since it has access to the user's long-term password and GAA bootstrapped keys. Similarly, the entity operating the BSF must be trusted not to use the GAA bootstrapped keys to perform a dictionary attack on an intercepted otp to recover pw. However, the PC hosting the client C in the UMTS GAA OTP scheme does not need to be trusted since it will not have access to the GAA bootstrapped key or pw; that is, the scheme provides protection of the long-term password even when used with untrusted or compromised PCs. In this latter case we need to assume that the UMTS mobile cannot be compromised by communicating with a malicious PC.

Of course, if an untrustworthy PC is used with UMTS GAA OTP, then otp will be compromised, as would be the case in the event of a phishing attack or an attack on the communications link between client and server. However, given that the GAA bootstrapped key SK remains secure, and that a secure means is used to compute otp from SK and pw, compromise of otp will only give an intruder access to the genuine server for a short period of time.

The only remaining strategy for an attacker trying to learn pw is to mount an on-line dictionary attack. In such an attack, an adversary guesses

pw and uses a GAA-enabled device to engage in the protocol (masquerading as the user) to test whether or not this single guess is correct. To defeat such an attack, S must lock out a user after a fixed number of failed authentication attempts.

Note that, in order to reduce the risk of compromise, the GAA bootstrapping keys and the user's password should be deleted from the GAA-enabled device after the computation of *otp*. Moreover, one-time passwords are consumed on demand, and thus the only secret that needs to be securely managed is the user's long-term password, which is known only by the user and the server.

7 Conclusions and Future Work

GAA is a general framework that enables pre-existing security infrastructures to be used to provide general purpose security services, such as key establishment. We have shown how GAA services can be built on the Trusted Computing security infrastructure, complementing the previously standardised GAA schemes built on mobile phone infrastructures. We then proposed a general scheme that converts a simple static password authentication mechanism into a one-time password (OTP) system using the GAA key establishment service. In this scheme the GAA-enabled device (the OTP token) is neither user nor server specific, and can be used in the protocol with no registration or configuration. The system also fits well to the multi-institution scenario. We have also described two practical instantiations of the general scheme, building firstly on UMTS GAA and secondly on TC GAA.

In these two instantiations, the GAA-enabled device is either a GAA-enabled 3G mobile phone or a personal computer with a TCG compliant TPM, both of which are very widely used. Thus both schemes have good scalability. Moreover, the systems can be used to enhance the most widely used authentication mechanism on the Internet with only small changes in the client and server.

We are currently working on prototype implementations of the UMTS GAA OTP system and certain variants of it in order to investigate security, usability, and performance in practice. In the UMTS GAA OTP system described in section 4.3, the web browser needs to be enhanced to be able to fetch the credentials from the mobile phone and insert them in the HTML forms. One way of achieving this is to install a supporting application (e.g. as a browser plug-in) in the browser. Another approach would become possible if Javascript was enhanced to support communications with a mobile phone. If such a facility was in place, a supporting application could be provided as a Javascript program that is downloaded when required.

As noted above, UMTS GAA OTP requires a connection (e.g. as pro-

vided by a USB cable or Bluetooth link) between the browser and the mobile phone. This requirement potentially reduces the scheme's usability. However, we have developed a multi-channel variant of the UMTS GAA OTP system designed to address this issue. In this scheme, the mobile device communicates with a application server directly to generate a counter- or time-based one-time password *otp*. The user reads the *otp* displayed on the mobile device and enters it, along with his or her *username*, into the PC web browser. Note that, in this multi-channel system, no supporting application needs to be installed in the web browser, and the need for a phone-browser connection is avoided.

The schemes we have described, together with certain variants of them currently under development, provide a simple and cost-effective means of upgrading static password authentication to a one-time system, addressing the serious threat posed by phishing [13].

Acknowledgments

The authors would like to thank Liqun Chen, Zheng Gong and Qiang Tang for their invaluable encouragement and advice.

References

[1] 3rd Generation Partnership Project (3GPP). *3G Security: Access Secure for IP-based Services, Technical Specification TS 33.203, Version 9.3.0*, 2009.

[2] 3rd Generation Partnership Project (3GPP). *Bootstrapping interface (Ub) and network application function interface (Ua); Protocol details, Technical Specification TS 24.109, Version 9.1.0*, 2009.

[3] 3rd Generation Partnership Project (3GPP). *Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS), Technical Specification TS 33.222, Version 9.1.0*, 2009.

[4] 3rd Generation Partnership Project (3GPP). *Numbering, Addressing and Identification, Technical Specification TS 23.003, Version 9.2.0*, 2009.

[5] 3rd Generation Partnership Project (3GPP). *Technical Specification Group Services and Systems Aspects, Generic Authentication Architecture (GAA), Generic Bootstrapping Architecture, Technical Specification TS 33.220, Version 9.2.0*, 2009.

[6] M. Alzomai and A. Josang. The mobile phone as a multi OTP device using Trusted Computing. In *Proceedings of the 4th International Conference on Network and System Security*, pages 75–82, Melbourne, Australia, 2010. IEEE Computer Society.

[7] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003.

[8] C. Chen, C. J. Mitchell, and S. Tang. Extending Trusted Computing as a security service. In preparation. A poster of this paper is available at *http://www.isg.rhul.ac.uk/cjm/Papers/etcaas.pdf*.

[9] J. Franks, P. M. Hallam-Baker, J. L. Hostetler, S. D. Lawrence, P. J. Leach, A. Luotonen, and L. C. Stewart. HTTP authentication: Basic and digest access authentication. Internet Engineering Task Force, RFC 2617, June 1999.

[10] T. Hardjono and G. Kazmierczak. Overview of the TPM key management standard. TCG Presentations at the 1st IEEE Key Management Summit, Baltimore MD, 23-24 September 2008. Available at *http://www.trustedcomputinggroup.org/resources/*.

[11] S. Holtmanns, V. Niemi, P. Ginzboorg, P. Laitinen, and N. Asokan. *Cellular Authentication for Mobile and Internet Services*. John Wiley and Sons, December 2008.

[12] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 9798-3:1998/Amd 1:2010, Information technology—Security techniques—Entity authentication—Part 3: Mechanisms using Digital Signature Techniques*, 1998.

[13] L. James. *Phishing Exposed*. Syngress, January 2006.

[14] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-hashing for message authentication. Internet Engineering Task Force, RFC 2104 (Informational), February 1997.

[15] R. Molva and G. Tsudik. Authentication method with impersonal token cards. In *Proceedings of the 1993 IEEE Symposium on Security and Privacy*, pages 56–65, Oakland, California, USA, 1993. IEEE Computer Society.

[16] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen. HOTP: An HMAC-based one-time password algorithm. Internet Engineering Task Force, RFC 4226 (Informational), December 2005.

[17] S. Pearson. Trusted Computing Platforms, the next security solution. Technical Report HPL-2002-221, Hewlett-Packard Laboratories, November 2002. Available at *http://www.hpl.hp.com/techreports/2002/HPL-2002-221.pdf*.

[18] Trusted Computing Group. *TCG Mobile Reference Architecture, TCG Specification, Version 1.0, Revision 1*, 2007.

[19] Trusted Computing Group. *TPM Main, Part 1 Design Principles, TCG Specification, Version 1.2, Revision 103*, 2007.

[20] Trusted Computing Group. *TPM Main, Part 2 TPM Data Structures, TCG Specification, Version 1.2, Revision 103*, 2007.

[21] Trusted Computing Group. *TPM Main, Part 3 Commands, TCG Specification, Version 1.2, Revision 103*, 2007.

[22] Trusted Computing Group. *TCG Mobile Trusted Module Specification, TCG Specification, Version 1.0, Revision 7.02*, 2010.