How to authenticate mobile devices in a web environment - The SIM-ID approach

Florian Feldmann, Jörg Schwenk

Horst Görtz Institute for IT-Security
Ruhr-University Bochum*
Universitätsstr. 150
44801 Bochum
florian.feldmann@rub.de
joerg.schwenk@rub.de

Abstract: With the advent of the iPhone AppStore and Google Play, the 'walled garden' approach of telecommunication companies to supply content to their customers using standard GSM/UMTS/LTE authentication has failed: Neither Google nor Apple, nor any other content provider on the mobile internet, uses the SIM card for authentication. This is mainly due to the fact that mobile telecommunication and internet architectures differ substantially.

In this paper, we propose several bridging technologies to fill this gap. We exemplarily show how to use SIM authentication for web-based Single-Sign-On protocols. Starting from simple password replacement in the authentication between User Agent (UA) and Identity Provider (IdP), we show how we can achieve strong channel bindings between all TLS channels and SIM based authentication.

1 Introduction

In many ways, today's smartphones can be regarded as fully operational computer systems, packing most features of desktop PCs from a few years ago in addition to extended communication functionality. This makes it possible to run applications similar to those of desktop PCs on these devices.

Many of these applications require communication with one or several internet servers providing a certain service (hence called "Service Provider", or *SP*). In most cases, these services require some form of authentication or authorization because certain information connected to these services may either be privacy restricted (e.g. personal data, the user does not wish to make publicly available) or legally restricted (e.g. certain company data which only employees of the corresponding company should have access to).

The most prominent type of authentication method nowadays is the username/password combination. This method, however, has some significant drawbacks: Passwords can be

*This project has been partially funded by the German Federal Ministry of Education and Research spyed out by an attacker during transmission and weak passwords could even be guessed easily. Though password policies can be used to force users to choose strong passwords and data encryption can be used to protect passwords from being spyed upon, there is still the risk of computer viruses and trojans reading the authentication data directly from the user's computer (e.g. by monitoring user input) or mounting Man-in-the-Middle attacks on security critical connections.

On desktop PCs, antivirus software is relatively common these days. On mobile devices, however, antivirus software is not yet as common. Also several shortcomings of mobile devices (e.g. restricted display size or restricted input possibilities) make it even harder for users to detect malicious behaviour on their devices. Thus, on these devices it is not advisable to use username/password combinations for login procedures. Instead, we propose a novel variant of Single Sign-On (SSO) services, where a user authenticates himself to a trusted entity which in turn authenticates the user towards the desired service. The proposed procedure makes use of the authentication features already available in every mobile device, i.e. the authentication features of GSM/UMTS using the SIM card plugged into mobile devices.

Previous Works The WebSIM approach [GKP00], proposed in 2000 by Guthery, Kehr and Posegga, shows how to implement web server functionality into a SIM card, thus rendering the SIM card accessible by internet applications. In [KPS$^+$01], Kehr et al. enhance this work and define an internet authentication protocol using features of the SIM Application Toolkit [3GP07a] and WebSIM. However, their approach significantly differs from the Single Sign-On protocols currently used, thus, it cannot easily be applied to current SSO scenarios. Also, it does not provide any information on secure TLS bindings nor any other security against Man-in-the-Middle attacks. Further, their implementation of WebSIM only took into account regular mobile phones, e.g. mobile phones providing only basic telephony functionality like phone calls or SMS, and not the nowadays commonly used smartphones, which are much more powerful and also customizable by the user, e.g. by installing additional software applications.

The Generic Authentication Architecture (GAA) [3GP13] specified by 3GPP for UMTS shares some similarities to our approach, but does not take into account current web architectures and developments. Instead, it forces service providers to support the interfaces defined in GAA. Secure TLS bindings are mentioned in this specification, but it is not explicitely defined how to use them in this context.

In [3GP12b], a framework is defined by 3GPP which allows for the adaption of the aforementioned GAA into actual standardized SSO scenarios. This very closely resembles our approach, but also does not provide any further information on secure TLS bindings. This is exactly the gap our work tries to close.

2 Related Work

For our approach, we mainly combine three existing mechanisms to form a new and secure authentication method for mobile devices: We make use of the mobile device's inherent authentication feature using the SIM card, utilize this feature to enhance a Single Sign-On login procedure and secure this procedure by applying certain cryptographic bindings.

2.1 SIM card authentication

Each "Subscriber Identity Module", or SIM for short, features a unique and secret key K_{SIM} (chosen by the mobile service operator during SIM card creation) which it shares with the corresponding mobile service operator. Also, several algorithms (e.g. for authentication, key derivation or encryption) are available on the SIM card which make use of the secret key K_{SIM} and generate corresponding cryptographic responses when triggered by a certain input. Detailed information on SIM cards can be found in [3GP07b]. For our purpose, only one algorithm is of interest, namely the authentication algorithm specified in the UMTS standard [3GP12a].

The standard does not demand a specific cryptographic algorithm to be used as authentication algorithm, but rather only states its functionality leaving the actual implementation to the design decisions of the mobile service operators. In short, the algorithm takes as input a 128 bit random value $nonce$ and uses the secret key K_{SIM} to compute the distinct corresponding "signed response" $SRES$, which is also 128 bits in length. Note, that "signed" in this case is rather a descriptive name and does not pertain to an actual digital cryptographic signature. The authentication algorithm functions more like a Message Authentication Code (MAC). As both SIM card and mobile service operator know the secret key K_{SIM}, the algorithm can be used for a challenge/response protocol where the challenger checks the other party's response - it computes the expected result $XRES$ (by computing $auth(K_{SIM}, nonce)$) and compares it to the received value $SRES$.

2.2 Single Sign-On

A Single Sign-On scenario is comprised of three parties - a user agent *UA* wishing to authenticate himself to a service provider *SP* and a trusted third party called identity provider *IdP*. *UA* and *SP* do not share any secret information which could be used by *UA* to authenticate himself to *SP*. This especially means that *UA* does not have a username/password combination for *SP*. Both parties *UA* and *SP*, however, have some sort of trust established in *IdP*, i.e. *UA* has some option to authenticate himself to *IdP* and *SP* trusts certain security assertions issued by *IdP*.

Figure 1 shows a typical Single Sign-On scenario. First, the user agent *UA* tries to access some restricted information on *SP* (denoted by the "GET" command). Because *SP* does not initially know and trust *UA*, it issues a so called "Authentication Request" $Auth\_Req$

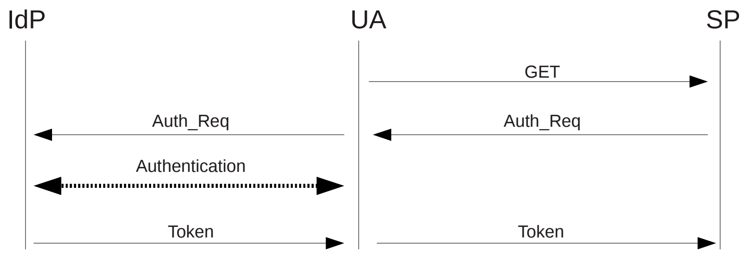

Figure 1: Single-Sign On

to *UA* containing a redirect to the corresponding identity provider *IdP*. This authentication request holds information about the issuer *SP* and possibly about the authentication methods accepted by *SP* to grant access to the requested ressource. *UA* forwards $Auth\_Req$ to *IdP*, which starts an “Authentication” procedure with *UA*.

This authentication could be done by a simple username/password combination. Due to the sensitive nature of an *IdP* (usually, *UA* can use the same login information to get access to multiple service providers) this, however, should be avoided. Some form of strong authentication should be used between *UA* and *IdP*. Examples for strong authentication include smartcards, one-time passwords or biometric data.

After *UA* has authenticated himself to *IdP*, *IdP* issues a “Token” to *UA* with a redirect message to the service provider who originally issued the authentication request. This Token must be integrity protected (most times this is done by *IdP* creating a digital signature over it) and should include information about the identity of *UA*.

Once *UA* has forwarded the Token to *SP*, *SP* can validate the signature and then use the information given in the Token about the identity of *UA* to grant access to corresponding resources.

Several frameworks allowing SSO systems to be built currently exist and are already widely in use (e.g. OpenID [RR06], OAuth [HL10], Facebook Connect [MU11] or the Security Assertion Markup Language (SAML) [CKPM05]). Our approach is very well suited to be used in conjunction with SAML, but can easily be adopted for any other SSO framework.

2.3 Secure Bindings

Several attacks are known which allow an adversary to gain Man-in-the-middle access to certain secure connections between parties or steal authentication Tokens from the user agent to use them to authenticate himself as *UA* to *SP* (examples can be found in [Kam08], [SSA$^+$09], [Mar09]).

To counter these threats, secure bindings have been proposed, which can be used to cryptographicaly bind certain identity information to specific TLS/SSL connections or specific communication partners.

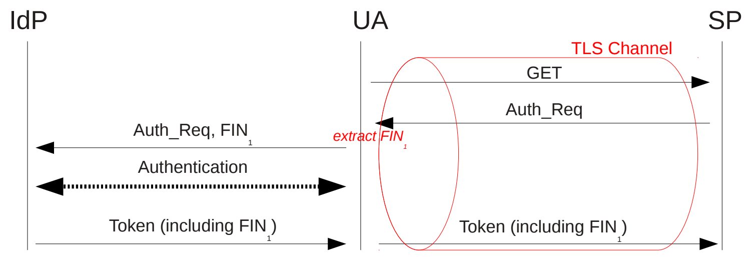

Figure 2: SSO login procedure with tls-unique binding

2.3.1 TLS-unique

For our purpose the *tls-unique* binding proposed in RFC 5929 [AWZ10] is of specific interest. The idea of this binding is to take some information uniquely identifying a certain TLS/SSL connection and cryptographically bind it to the authentication information. The *Finished* messages are the last two messages sent from user agent to server, resp. server to user agent, during the TLS/SSL connection establishment and the first messages which are actually encrypted with the key material derived in the connection establishment. They contain a MAC over all messages previously sent in this connection establishment, thus uniquely identifying this specific TLS/SSL session (any other TLS/SSL session would be established with at least differing user agent and/or server nonces, resulting in different Finished messages). By default, the first Finished message of a connection is used to uniquely identify it for the purpose of *tls-unique* bindings.

Figure 2 shows a Single Sign-On login procedure as described in Section 2.2. Before forwarding the authentication request to *IdP*, *UA* extracts the first Finished message from the TLS/SSL session established between *UA* and *SP* (shown in red) and appends it to the authentication request. If the subsequent authentication procedure between *UA* and *IdP* results in a positive outcome, *IdP* will include the Finished message it received from *UA* into the authentication token (along with all other information for this token as described above). When *SP* receives the token and successfully validates its signature, *SP* also extracts the first Finished message from the TLS/SSL session established with *UA* (note that this must still be the same TLS/SSL session, so the session must be kept alive throughout the authentication procedure and no renegotiation is allowed). It then compares the extracted Finished message to the one included in the authentication token. Because an attacker mounting a Man-in-the-Middle attack on the TLS/SSL session between *UA* and *SP* will have different Finished messages in its connections to *UA* and *SP* respectively, *SP* will detect any MitM attacker at this point.

2.3.2 Strong Locked Same Origin Policy

The Same Origin Policy (SOP) implemented in most current web browsers protects user data by allowing e.g. cookies only to be sent to the same server which has stored the cookie in the first place. The "same server" is hereby denoted by the triplet of (protocol, domain name, port). The Strong Locked Same Origin Policy (SLSOP) [KSTW07] enhances this concept to cryptographically bind an SSO Token to the public/private key pair of the intended *SP*. A detailed analysis of SSO using SLSOP is given in [SKA11].

3 The SIM-ID Protocol

In order to enhance the security of SSO login procedures on mobile devices we include SIM card functionality into it. As shown in Section 2.2, Figure 1 shows a typical SSO login procedure between a Client *UA* (in this case a mobile device) and a Service Provider *SP*, utilizing a trusted Identity Provider *IdP* to establish the authentication between *UA* and *SP*. In our scenario, *SP* is most likely a web server requiring user authentication to provide a certain service. *SP* is assumed to have a public/private key pair pk_{SP}/sk_{SP} along with a corresponding certificate to check the validity of its public key. The user agent *UA* in this particular setup is a mobile web browser running on a mobile device. The web browser does not own any cryptographic keys, but has access to a SIM card sharing a symmetric secret key K_{SIM} with the corresponding mobile service provider. This mobile service provider also acts as Identity Provider *IdP* possessing its own public/private key pair pk_{IdP}/sk_{IdP} together with the shared key K_{SIM}. We assume that this *IdP* is trusted by *UA* and the Service Provider(s) *SP* associated with it. We also assume that *IdP* knows the correct public keys pk_{SP_i} of its associated Service Providers.

3.1 SIM-ID Authentication Towards Mobile Network Provider

In our proposal the mobile network operator will serve as Identity Provider in the SSO scenario. Thus, the mobile device requires a means to authenticate itself to the mobile network operator. The GSM/UMTS standards already provide authentication of a mobile device towards a base station, i.e. mobile network operator. In case of UMTS, this authentication is even mutual. This approach was originally intended for securing communications within GSM/UMTS networks, but can easily be adapted for use in internet (e.g. WLAN) connection scenarios.

As described in Section 2.1, UMTS authentication between mobile device and base station is performed by mutually sending a nonce to which the other partner replies with the value resulting from the authentication algorithm using the symmetric secret key K_{SIM}. As K_{SIM} is known only to the SIM card and the mobile network operator, this functions as an implicit authentication between the two parties.

We use exactly this authentication algorithm in conjunction with the tls-unique binding described in Section 2.3.

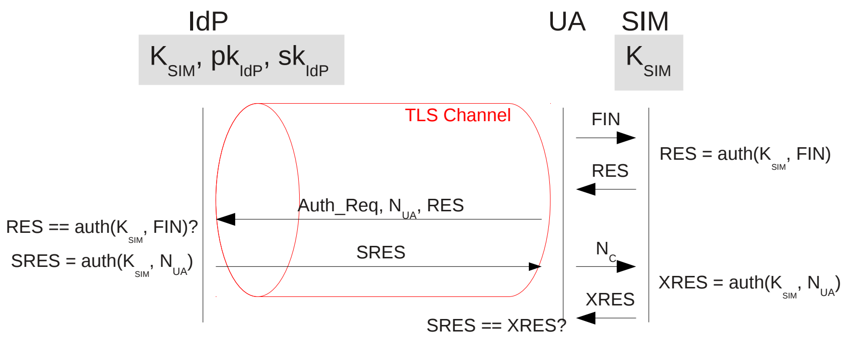

Figure 3: SIM-ID protocol - Authentication Towards Mobile Network Provider

Figure 3 shows the resulting protocol. First, *UA* establishes a TLS/SSL connection with *IdP* and extracts the first Finished message FIN (according to the tls-unique binding). *UA* forwards FIN to the authentication algorithm on its SIM card, which computes and returns the signed response value $RES = auth(K_{SIM}, FIN)$. *UA* then sends an authentication request $Auth\_Req$, a nonce N_{UA} and the authentication value RES to *IdP* via the established TLS/SSL connection.

IdP now verifies the authentication value by also extracting the first Finished message FIN from the TLS/SSL channel, performing the same computation of $auth(K_{SIM}, FIN)$ and comparing this value with the received RES. To protect against an attacker impersonating *IdP*, *IdP* is also required to perform an authentication by computing the signed response $SRES = auth(K_{SIM}, N_{UA})$ and sending the result to *UA*. *UA* can now use the SIM card to check whether the expected response to his nonce N_{UA} equals the one received from *IdP* by computing $XRES = auth(K_{SIM}, N_{UA})$ and comparing $SRES == XRES$.

3.2 SIM-ID Authentication Towards Service Provider

With authentication between mobile network provider and mobile device already described in Section 3.1, we now concentrate on how to extend this authentication to a three-party-scenario.

Figure 4 shows the setup and the enhanced SSO protocol. Note that according to Section 3.1 we already assume a mutually authenticated TLS/SSL connection between *UA* and *IdP*, even though this is not explicitly shown in the figure.

We denote by $[m]sk_i$ a signature over message m created with the secret key sk_i of party i which is sent along with the message (thus, $[m]sk_i$ means we send the message m and its correponding signature in the same communication phase). Likewise, $\{m\}pk_i$ denotes

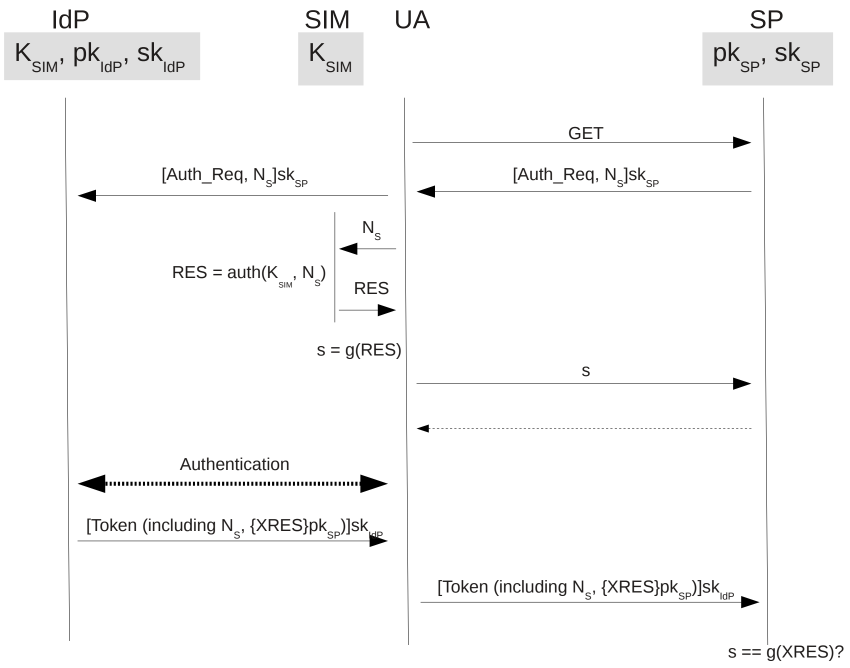
IdP
K_{SIM}, pk_{IdP}, sk_{IdP}
SIM
K_{SIM}
UA
SP
pk_{SP}, sk_{SP}
GET
[Auth_Req, N_S]sk_{SP}
[Auth_Req, N_S]sk_{SP}
N_S
RES = auth(K_{SIM}, N_S)
RES
s = g(RES)
s
Authentication
[Token (including N_S, {XRES}pk_{SP})]sk_{IdP}
[Token (including N_S, {XRES}pk_{SP})]sk_{IdP}
s == g(XRES)?

Figure 4: SIM-ID protocol

an encryption of message m with public key pk_i for intended recipient i (of course, in this case only the encrypted message $\{m\}pk_i$ is sent, not the plaintext m).

With the authentication request $Auth\_Req$ corresponding to the selected SSO scheme, *SP* sends a nonce N_S (the request, or at least the nonce, should be signed with the private key sk_{SP} of *SP* to protect integrity and authenticity of the nonce). When forwarding the authentication request to *IdP*, *UA* extracts the nonce N_S and sends it to its integrated SIM module. The SIM module then takes the nonce N_S as input and uses the authentication algorithm to calculate the corresponding response $RES = auth(K_{SIM}, N_S)$.

The response $s = RES$ can now be sent back to *SP* where it can be validated. If for some reason the value RES should be further used by both parties, e.g., by deriving a symmetric key from it, the value should obviously not be sent in the clear. Instead, it can be blinded using a cryptographic one way function g (e.g. a cryptographic hash function), thus sending $s = g(RES)$.

In the next step *UA* has to authenticate himself to *IdP* in order to receive an authentication token. In Section 3.1 we already described an authentication protocol between a mobile network operator and a mobile device. Technically, though, every mutual authentication protocol between *UA* and *IdP* could be used here.

After authentication between *UA* and *IdP* has been performed, *IdP* calculates an expected result $XRES = auth(K_{SIM}, N_S)$ using the symmetric key K_{SIM} corresponding to the authenticated user. $XRES$ is encrypted with the public key of *SP*, pk_{SP}, and the resulting cipher text $\{XRES\}pk_{SP}$ is included into the authentication token (the format of this token is dependent on the actual SSO scheme selected, but like the nonce N_S in the authentication request before, this token should at least be signed using the private key of *IdP*, sk_{IdP}, to ensure integrity and authenticity). *IdP* sends the token to *UA*, who forwards it to *SP* (the Token and the authentication value s can also be sent together in one single message).

When *SP* receives the token, it is validated according to the chosen SSO scheme. Also, *SP* decrypts the encrypted value $\{XRES\}pk_{SP}$ and checks if $s == XRES$ (or checks if $s == g(XRES)$ instead, if the blinded version is used).

3.2.1 Including Bindings

As we have not yet assumed any TLS/SSL session between *UA* and *SP* - apart from the encryption of $XRES$ with the public key of *SP* (and of course the authentication procedure between *UA* and *IdP*, described in Section 3.1) no encryption is used - the protocol described thus far is prone to a simple Man-in-the-middle attack between *UA* and *SP*: An attacker could simply read all data sent from *UA* to *SP* and then steal the result value s as well as the token and use both to authenticate himself as *UA* against *SP*.

The viability of the proposed protocol becomes visible when SSL/TLS is used for the communication between *UA* and *SP* as well. After all, the connection between a mobile web browser and a corresponding web server providing access to restricted resources should most likely be encrypted to provide a secure transport of these resources.

Utilizing the idea described in Section 2.3 we can bind the authentication info of *UA* to the current TLS/SSL session, again by using the first Finished message FIN from the TLS/SSL session establishment. Only two minor adjustments have to be done to the protocol to include this form of session binding:

- When creating the blinded response s, the user agent also includes the first Finished message FIN by calculating $s = g(FIN \text{ XOR } RES)$
- Similarily, *SP* also takes the Finished message into account by checking if $s == g(FIN \text{ XOR } XRES)$

In Figure 5, the enhanced protocol is shown.

Note that the actual TLS-unique binding has the Finished message sent from user agent to *IdP*, where it is included into the token. The corresponding equivalent in our case would be to input FIN into the authentication function of the SIM card, providing $RES = auth(K_{SIM}, FIN)$ and having the *IdP* also compute this value as $XRES = F_{K_{SIM}}(FIN)$ (compare Section 3.1). However, the *SP* is not able to compute this value himself, and also there is no possibility for him to extract FIN from $XRES$. The *SP* does not gain any knowledge about the Finished message sent from user agent to *IdP* and thus it is not possible for *SP* to check whether or not the token is really bound to a specific TLS channel.

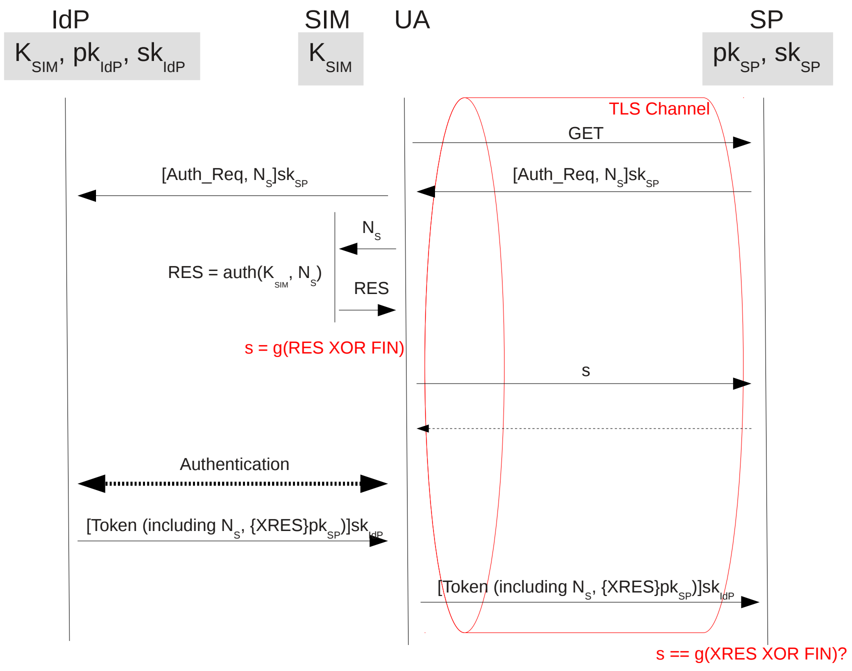

Figure 5: SIM-ID protocol with included TLS-unique binding

By using $s = g(FIN \text{ XOR } RES)$ on *UA* side and checking if $s == g(FIN \text{ XOR } XRES)$ on *SP* side, *SP* can be sure that the user agent really is the mobile device which it claims to be (because $XRES$ is confirmed by the trusted *IdP*) and also that it shares the current TLS channel with *SP* (by providing the correct Finished message).

Also note that by encrypting the value $XRES$ specifically for *SP*, we effectively included a server end-point binding of SLSOP type (compare Section 2.3.2).

4 Conclusions and Outlook

We have shown how to adapt the UMTS authentication procedure for use within internet scenarios. Our proposed solution makes use of secure cryptographic bindings to strengthen the mutual authentication between a user agent *UA* and a mobile service provider acting as identity provider *IdP*. By using these secure bindings, we can effectively prevent an attacker from being able to mount a Man-in-the-Middle attack on the communication between *UA* and *IdP*.

We have then extended our protocol to also apply to a three party Single Sign-On scenario, where *UA* wants to authenticate to a service provider *SP* without any prior shared secret between *UA* and *SP*. For this purpose we use a mutually trusted third party *IdP* to build mutual trust between *UA* and *SP*. Our protocol is also robust against impersonation attacks and Man-in-the-Middle attacks between *UA* and *SP*.

Our protocol can be used with any existing Single Sign-On scheme. Service providers which already support Single Sign-On with the tls-unique channel binding, only need minor adjustments to their authentication procedures. Users can continue to use their current mobile devices and do not need any additional secure hardware.

The limitations of our protocol become evident when considering mobile malware present on the user's device: Though a simple keylogger would no longer be sufficient to spy out the user's username/password combination (this type of espionage in our protocol is, in fact, impossible, as we do not use username/password combinations), a powerful malware which has gained full control of the mobile device could use the SIM card to authenticate against arbitrary service providers. Once the user has unlocked use of his SIM card, there is nothing stopping a malware from obtaining the required authentication values from it, as the SIM card cannot distinguish between 'legal' (i.e. invoked by the user) and 'illegal' (i.e. triggered by the malware) authentication requests. In short, with our protocol, *IdP* and *SP* will be convinced that they both are communicating with a specific mobile device (or rather, a mobile device using a specific SIM card), but they cannot be sure whether the communication was conducted by the regular user or a mobile malware.

References

[3GP07a] 3GPP. 3GPP TS 11.1; Specification of the SIM Application Toolkit (SAT) for the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface. Technical Report 11.11, 3rd Generation Partnership Project (3GPP), http://www.3gpp.org/ftp/Specs/html-info/1114.htm, June 2007.

[3GP07b] 3GPP. 3GPP TS 11.11; Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface. Technical Report 11.11, 3rd Generation Partnership Project (3GPP), June 2007.

[3GP12a] 3GPP. 3GPP TS 33.102; 3G Security; Security architecture. Technical Report 33.102, 3rd Generation Partnership Project (3GPP), December 2012.

[3GP12b] 3GPP. 3GPP TS 33.980; Interworking of Liberty Alliance Identity Federation Framework (ID-FF), Identity Web Services Framework (ID-WSF) and Generic Authentication Architecture (GAA). Technical Report 33.980, 3rd Generation Partnership Project (3GPP), September 2012.

[3GP13] 3GPP. 3GPP TS 33.222; Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS). Technical Report 33.222, 3rd Generation Partnership Project (3GPP), March 2013.

[AWZ10] J. Altman, N. Williams, and L. Zhu. Channel Bindings for TLS. RFC 5929 (Proposed Standard), July 2010.

[CKPM05] Scott Cantor, John Kemp, Rob Philpott, and Eve Maler. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. Technical report, March 2005.

[GKP00] Scott Guthery, Roger Kehr, and Joachim Posegga. How to turn a GSM SIM into a web server. In *Smart Card Research and Advanced Applications*, pages 209–222. Springer, 2000.

[HL10] Eran Hammer-Lahav. The oauth 1.0 protocol. 2010.

[Kam08] Dan Kaminsky. Black ops 2008: It's the end of the cache as we know it. *Black Hat USA*, 2008.

[KPS$^+$01] Roger Kehr, Joachim Posegga, Roland Schmitz, Peter Windirsch, et al. Mobile security for Internet applications. *Proceedings of Kommunikationssicherheit KSI'2001*, pages 27–28, 2001.

[KSTW07] Chris K. Karlof, Umesh Shankar, Doug Tygar, and David Wagner. Dynamic pharming attacks and the locked same-origin policies for web browsers. Technical Report UCB/EECS-2007-52, EECS Department, University of California, Berkeley, May 2007.

[Mar09] Moxie Marlinspike. More Tricks For Defeating SSL In Practice. *Black Hat USA*, 2009.

[MU11] Marino Miculan and Caterina Urban. Formal analysis of Facebook Connect single sign-on authentication protocol. In *SOFSEM*, volume 11, pages 22–28, 2011.

[RR06] David Recordon and Drummond Reed. OpenID 2.0: a platform for user-centric identity management. In *Proceedings of the second ACM workshop on Digital identity management*, DIM '06, pages 11–16, New York, NY, USA, 2006. ACM.

[SKA11] Jörg Schwenk, Florian Kohlar, and Marcus Amon. The power of recognition: secure single sign-on using TLS channel bindings. In *Proceedings of the 7th ACM workshop on Digital identity management*, DIM '11, pages 63–72, New York, NY, USA, 2011. ACM.

[SSA$^+$09] Marc Stevens, Alexander Sotirov, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, and Benne De Weger. Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate. In *Advances in Cryptology-CRYPTO 2009*, pages 55–69. Springer, 2009.