Formal Security Analysis of OpenID with GBA Protocol

Abu Shohel Ahmed[1] and Peeter Laud[2]

[1] Ericsson Research
[2] Cybernetica AS and Tartu University

Abstract. The paper presents the formal security analysis of 3GPP standardized OpenID with Generic Bootstrapping Architecture protocol which allows phone users to use OpenID services based on SIM credentials. We have used an automatic protocol analyzer to prove key security properties of the protocol. Additionally, we have analyzed robustness of the protocol under several network attacks and different threat models (e.g., compromised OP, user entity). The result shows the protocol is secure against key security properties under specific security settings and trust assumptions.

1 Introduction

In today's world, digital identity [26] of a person is important for accessing online services. At present, most online (i.e., application) service providers maintain their own identity management systems. This approach requires the user to register and authenticate separately for each online services. OpenID is one such solution which can overcome these problems. OpenID (OpenID 2.0 [22]) allows the user to control his identity information and it also allows access to several online services (i.e., web sites) with a single identifier [23].

Although OpenID provides an easy way for services to identify the user, the usability of OpenID is questionable [9]. The user is still required to type username and password (most common authentication mechanism) at OpenID provider to prove his identity. Moreover, from the security perspective, humans are not good at choosing and remembering strong passwords [18]. All these problems suggest that current OpenID implementations are not user friendly and secure for smart phone platforms.

For a long time, telecommunication companies have been successfully dealing with large subscriber identity systems. Telecommunication companies can easily assert identity attributes to a real person or against a subscriber identity module (SIM). Using this unique property, 3rd Generation Partnership Project (3GPP) has defined an internetworking combining OpenID with generic bootstrapping architecture (GBA). GBA is a mechanism for generating shared keys between a smartphone and a mobile network operator (MNO) [1]. The generated shared key is used to authenticate the user to the OpenID provider. This process poses a number of interesting security questions. We have addressed those questions

R. Prasad et al. (Eds.): MOBISEC 2011, LNICST 94, pp. 113–124, 2012.

by performing a formal security analysis of the protocol. Our analysis concludes that the protocol is secure within the specific trust relationships of participating parties.

To date, a number of security analyses of OpenID have been performed. Existing analyses concentrate mostly on the (lack of) protection of the HTTP traffic between the user, the relying party, and the OpenID provider. Sovis et al. [24] have found that if the HTTP 302 redirection performed by the relying party and the OpenID provider is onto HTTP (not HTTPS) endpoints, then these redirections can be changed later by the attacker. Urueña et al. [25] discuss the information leaks to third parties through HTTP headers. Both Sovis et al. and Feld and Pohlmann [12] discuss attacks based on lack of authentication of the OpenID provider. Same kinds of attacks are also discussed by Lindholm [17] who has performed, by our knowledge, the only tool-supported analysis of OpenID so far, using the AVISPA protocol analyser [5]. AVISPA is also used to analyze security properties of EAP AKA [27], which similar to GBA is based on UMTS authentication and key agreement protocol. In this paper, we are interested in the interference of OpenID and GBA, hence we do not consider these kinds of issues and model the protocols in a way that in the OpenID protocol, the necessary authentication of parties has been performed. In this paper, section 2 describes the inter-networking of OpenID with GBA and base security assumptions (i.e., honesty of parties and security of channels). Section 3 presents the formalization of the protocol using Proverif [8]. Section 4 presents verification results from our analyses. Finally, section 5 presents several scenarios based on modified base security model.

2 OpenID with GBA Protocol

This section describes the inter-networking of OpenID with GBA according to TR 33.924 [2]. The OpenID with GBA protocol consists of the following parties.

- The user U wants to authenticate to a Relying party (RP) using OpenID with GBA protocol. Our analysis assumes implicit users.
- The user entity (UE) (e.g., a client application combined with GBA modules in a Smartphone) is actually used to access the RP. Although, the original protocol [2] contains a separate GBA client module and a UICC[1] module, we consider them as a single user entity. Initially, both the UICC and the HSS[2] share a master secret key K.
- The RP contains resources to which the user wants to connect.
- NAF-OP is a combination of OpenID Provider and Network Application Function. The NAF-OP has a secret key sk_O which has a public counterpart pk_O that is bound to the NAF-OP by cert_O. The NAF-OP acts as a trusted party for the RP.

[1] Universal Integrated Circuit Card.

[2] Home Subscriber System.

- The Bootstrapping server (BSF) is a central party of the protocol. It acts as a trusted party and mediates the application specific user session key between the user entity and the NAF-OP. The BSF has a secret key sk_B which has a public counterpart pk_B that is bound to the NAF-OP by cert_B.
- The Home Subscriber Server (HSS) shares a master secret with the UICC. The HSS has a secret key sk_H which has a public counterpart pk_H that is bound to the HSS by cert_H. The BSF and the HSS maintain a both way secure channel.

Besides, a Certificate Authority (CA) exists to issue certificates for NAF-OP, BSF, and HSS. There are also means to verify the validity of the certificate.

2.1 Protocol Description

The protocol is depicted in Fig. 1. The protocol session is initiated by the user (not shown) deciding to access resources from the Relying party through the user entity. The user entity provides the user-supplied identifier O_u to the Relying party. The RP normalizes and discovers [22] an identity provider based on the O_u. After the discovery, the RP and the NAF-OP establish a shared key using Diffie-Hellman (DH) association. The DH is performed over a server authenticated TLS tunnel [10].

The RP sends a redirection message (includes claimed id O_i, relying party identity O_{rt}, provider identity OP_{End}, DH association key H_{DH} to the NAF-OP through the user entity. The redirection message between the UE and NAF-OP is sent through a server authenticated TLS tunnel. The NAF-OP uses HTTP digest authentication [13] for user authentication. According to HTTP authentication, the NAF-OP sends a HTTP digest challenge to the UE containing a nonce N_0 and NAF_{realm}. To reply this challenge, the UE requires a valid user name and password. This leads the UE to start the bootstrapping process.

Bootstrapping is a four step process based on HTTP digest AKA [20]. At first, the UE starts bootstrapping by sending the $IMPI$[3] to the BSF. The BSF, in turn, creates a two way secure channel with the HSS. The HSS contains the shared master secret key K and a running sequence number SQN for each $IMPI$. The HSS generates $RAND$: a random number; $XRES$: a number generated from one way hash function [6] using K, $RAND$, and SQN; CK: a generated session key for confidentiality; IK: a generated session key for integrity; $AUTN$: contains encrypted sequence SQN, authentication management field AMF, and a message authentication code (MAC) [21]. The generated $RAND$, $XRES$, CK, IK, and $AUTN$ are sent to the BSF using the secure channel. Second,the BSF sends a challenge to the UE containing $RAND$, and $AUTN$. The UE generates RES, CK, IK, and MAC based on its own shared master key K and sequence number SQN'. At this stage UE performs two equivalence tests: 1) it checks the UE sequence number is big enough from the HSS sequence number. 2) it also checks the equivalence of MAC received from the BSF with its own MAC. The UE authenticates the BSF based on the results of these two tests. Third,

[3] IP Multimedia Private Identity.

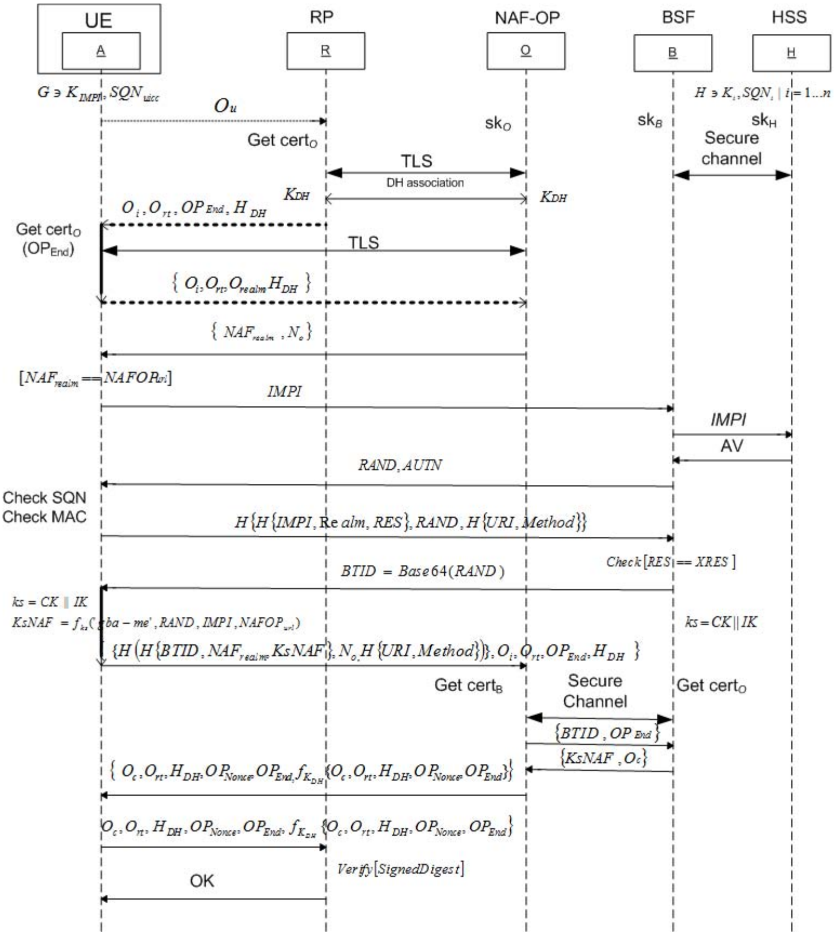
UE
A
RP
R
NAF-OP
O
BSF
B
HSS
H
$G \ni K_{IMPI}, SQN_{uicc}$
O_u
Get cert$_O$
sk$_O$
sk$_B$
sk$_H$
$H \ni K_i, SQN_i \mid i = 1 \ldots n$
Secure channel
TLS
DH association
K_{DH}
K_{DH}
$O_i, O_{rt}, OP_{End}, H_{DH}$
Get cert$_O$ (OP$_{End}$)
TLS
$\{ O_i, O_{rt}, O_{realm} H_{DH} \}$
$\{ NAF_{realm}, N_o \}$
$[NAF_{realm} == NAFOP_{uri}]$
$IMPI$
$IMPI$
AV
$RAND, AUTN$
Check SQN
Check MAC
$H\{H\{IMPI, Realm, RES\}, RAND, H\{URI, Method\}\}$
$BTID = Base64(RAND)$
$Check[RES == XRES]$
$ks = CK \| IK$
$KsNAF = f_{ks}('gba-me', RAND, IMPI, NAFOP_{uri})$
$\{H(H\{BTID, NAF_{realm}, KsNAF\}, N_o, H\{URI, Method\}\}, O_i, O_{rt}, OP_{End}, H_{DH}\}$
$ks = CK \| IK$
Get cert$_B$
Secure Channel
Get cert$_O$
$\{BTID, OP_{End}\}$
$\{KsNAF, O_c\}$
$\{ O_c, O_{rt}, H_{DH}, OP_{Nonce}, OP_{End}, f_{K_{DH}}\{O_c, O_{rt}, H_{DH}, OP_{Nonce}, OP_{End}\}\}$
$O_c, O_{rt}, H_{DH}, OP_{Nonce}, OP_{End}, f_{K_{DH}}\{O_c, O_{rt}, H_{DH}, OP_{Nonce}, OP_{End}\}$
$Verify[SignedDigest]$
OK

Fig. 1. OpenID with GBA protocol

the UE sends a HTTP digest response (with RES as password) to the BSF. The BSF validates the response against the previously stored $XRES$ value. The BSF authenticates the UE if only if the received RES and generated $XRES$ are equal. Upon successful UE authentication, the BSF generates a bootstrap key identifier ($BTID$) and a new temporary shared key Ks (Ks is a combination of CK and IK). Finally, the BSF sends $BTID$ to the UE. The integrity of $BTID$ is protected using HTTP digest parameter $qop = auth - int$. The UE stores the $BTID$, temporary shared secret key Ks (i.e., generated from CK and IK). At this stage, both the BSF and the UE are mutually authenticated and share a new master session key Ks.

To complete the HTTP authentication with the NAF-OP, the UE generates an application specific key $KsNAF$ using Ks and NAF-OP identity. The UE generates a HTTP digest response for the NAF-OP based on $BTID$ as a user name and $KsNAF$ as a password. The NAF-OP verifies this reponse in consultation with the BSF. Finally, the NAF-OP redirects the UE to the RP with a signed (i.e., using DH key) assertion. The UE is allowed to access RP resources based on the assertion (e.g., signed digest) from the NAF-OP.

2.2 Constraints

OpenID with GBA can be deployed in several security contexts (e.g., messages can communicate within server authenticated tunnel or in non-tunneled mode). Our analysis is based on common deployment scenarios: the UE and the RP communicates in the non-tunneled mode while the communication between the UE and the OP, and the communication between the RP and the OP are within tunneled mode. The communication between the UE and the BSF is in non-tunneled mode while OP to BSF and BSF to HSS use two way secure channels. In addition, the analysis is applicable for GBA_ME [15, p. 45] which is the most common variant of GBA. The model only focuses on the authentication steps of GBA_ME. This implies the synchronization of SQN in GBA_ME has been performed beforehand. Additionally, our model is valid for stateful OpenID where an association (i.e. D-H) is performed between the RP and the OP to verify the subsequent protocol messages.

2.3 Base Security Model

We have made the following assumptions about the security of various channels and entities as our base security model.

1. There will be several users (i.e., user entity) and RP sites, some of the RP sites are under adversarial control.
2. The UICC and the GBA modules reside within the user entity that is under the control of the respective user. All these modules are trusted in our base security model.
3. The channel between the user and his user entity is secure. We assume the client is trustworthy and only trustworthy client can access the GBA and the UICC module.
4. There are multiple honest NAF-OP servers.
5. There is only one honest BSF, and one HSS server controlled by the mobile network operator. The channel between them is secure.

We are interested in the secrecy properties of master key K shared between the UICC and the HSS, temporary session key Ks shared between the BSF and the UE, and application session key $KsNAF$ shared between the UE and the NAF-OP. We are also interested in the TLS keys agreed between honest users and servers. Additionally, we are interested in the correspondence properties e.g., if a server thinks that it has communicated with a client controlled by a user using a key k and the user is honest, then the user also thinks that it has communicated with the server in a session where the client application is using the key k *(integrity of servers)* [16]. The same principle must hold if we swap the role of the user and the server *(integrity of clients)*.

3 Formalization in ProVerif

ProVerif [7] is a mature protocol analyzer based on the Dolev-Yao [11] (or: perfect cryptography) model. This model, which has been extremely fruitful for cryptographic protocol construction and analysis, abstracts from the actual cryptographic algorithms by stating that only an enumeration of operations on messages — the actual invocations of algorithms — make sense, both for the honest parties and the adversary.

The analyzer takes as input a protocol specified in a dialect of the applied pi-calculus [3] and a specification of a security property. It verifies whether the protocol satisfies this property, answering either correctly or inconclusively (because of undecidability of protocol verification). The process language contains primitives for sending and receiving messages over channels, generating new names as keys, nonces, channel names etc., constructing and destructing messages, branching, sequential and parallel composition, and replication. The security property can either be a secrecy property (the attacker is unable to deduce a certain value), a correspondence property (certain parts of the protocol can be executed only after other parts have been executed), or a certain process equivalence. The verification proceeds by translating the protocol into a set of Horn clauses (possibly abstracting from the actual behaviour of the protocol) and then applying a specialized inference engine on it. Below we describe our model of OpenID with GBA protocol.

Secure Channels. In the OpenID with GBA protocol, certain parties have secure channels between them. To model those, we need a channel name that only these parties know. We obtain such name for two parties as $\mathsf{tochannel}(sk_1, pk_2) = \mathsf{tochannel}(sk_2, pk_1)$ where sk_i is the secret key of the i-th party, $pk_i = \mathsf{pk}(sk_i)$ is the public key corresponding to it, tochannel and pk are term constructors, and the equation is stated to hold (ProVerif can model certain equations, including this one). The public keys are bound to names by means of *certificates.*

TLS Handshake. Certain parties exchange messages over confidentiality-providing TLS tunnels where one end (the server) is authenticated. We do not want to analyze the TLS protocol in this work, because its security properties have been thoroughly studied [14,19]. To model the tunnel, we again construct a new channel name. It is done by the client C sending the server S a nonce N encrypted under the public key of S, after which the channel is defined as $\mathsf{tlschan}(N, pk_S)$, where tlschan is a public constructor. The server is authenticated by its ability to decrypt and find N. The honest parties will construct secure channels and TLS tunnels only after checking the *certificate* of the other party, thereby excluding man-in-the-middle attacks.

Certificates and Identification. The names of the parties of the protocol are bound to their public keys by means of certificates. Certificates are public messages, signed by a trusted party — the certificate authority (CA). While detailed modeling of certificates and CAs is possible in ProVerif, we want to avoid it, and thus abstract a certificate as a message $\mathsf{cert}(X, pk_X, role)$, where

X is the name of the party, pk_X its public key, and the third argument shows in which role (NAF-OP, or BSF, or HSS) this public key is intended to be used. The issuing and publication of certificates of honest parties are modeled by constructing that certificate inside the process of that party and outputting it on a public channel. The constructor cert is *private* — the attacker cannot cannot construct such messages itself. On the other hand, the destructors, giving access to the components of a certificate, are public.

To model dishonest parties inside the system, the attacker needs a means to obtain certificates to public keys where it knows the corresponding private key. Hence we include a simple "CA" in our analyzed process. This CA receives a public key from the attacker and issues certificates for it. The name appearing in the certificate is chosen by the CA, not the attacker, hence the attacker cannot masquerade as an honest party.

Symmetric master keys shared by the BSF and each of the user devices are modeled similarly. There is a private constructor impi2key that transforms IMPI into the master key. Similarly to certificates, the adversary can get (IMPI,key)-pairs from the simple CA.

The Whole System. After the abstractions described above, the modeling of the system is quite straightforward. We have processes for the user equipment (client application, GBA and UICC are all modeled in a single process), NAF-OP, RP, BSF, and HSS running in parallel, and replicated (except BSF and HSS) to model several parties of the same kind. All processes, except the user equipment and the RP, start by generating a name for themselves and creating their public and private key pair. They will publish the certificate binding their name, their role and their public key, and will then continue with the protocol sessions [4].

Note that in our model, RPs do not have certificates. The users will thus communicate with them over public channels (i.e. using HTTP, not HTTPS). The lack of RP certificates also means that they may be under adversarial control. It is noteworthy that the results we report in the next section are valid in this setting. On the other hand, the lack of RP certificates means that the users can never be sure about the identities of RPs and the security properties mentioning those cannot be verified. We do not consider this the weakness of our model — using HTTPS (or: TLS handshake) would be the standard way for the user to verify the identity of RP and the properties of TLS imply that in this case, the user can be sure who he has connected with.

The users also do not have certificates. However, the users have IMPIs and, through the private mapping impi2key, the user shares symmetric keys with the BSF.

4 Verification Results

In this section, we report the security properties we are interested in, the way we are modeling them in ProVerif, and the verification result. We perform the formal analysis in two phases: first only for bootstrapping process, second, the combined OpenID with GBA protocol (which, according to our knowledge, has not been subject of such formal analysis before).

4.1 BootStrapping

Secrecy. We are interested in the secrecy of long-term keys, as well as the short-term keys *Ks* and *KsNAF*. In the Dolev-Yao model, we model this property as the attacker's inability to obtain the term corresponding to this key. In ProVerif, such queries are straightforward to state. We obtain that all keys initially shared or constructed during a protocol session between an honest user and an honest BSF remain secret.

Authentication. We want to make sure that if the BSF thinks it has completed a session with the user U (and associated the value *BTID* with the IMPI of U), then the user U actually has participated in a protocol session with the BSF. This is a typical correspondence property and it can be easily specified in ProVerif. The protocol code may contain statements "**event** M", where M is a message. The execution of this statement has no effect on the protocol run, but the occurrence of the event M can be recorded. ProVerif can answer queries regarding the order in which the events occur.

To model this property, we add events *BSFEnd(impi)* in the end of the process modeling a session for BSF, and *UICCBegin(impi)* in the beginning of the process modeling a session for UICC. We ask ProVerif whether each *BSFEnd(i)* must be preceded by a *UICCBegin(i)* for any term i. The answer is negative because the attacker may also be known as a user of the system to the BSF, and the attacker does not emit the event *UICCBegin(...)* in the beginning of its session. So we check whether *impi* is the IMPI of an honest participant before emitting *BSFEnd(impi)*. We do this by the common technique of emitting the honest IMPIs on a private channel (this is added to the user process) and trying to receive *impi* from this channel before emitting the event. Now ProVerif states that the correspondence property holds — each *BSFEnd(impi)* for an honest *impi* is preceded by *UICCBegin(impi)* in all traces. The correspondence is even injective — different *BSFEnd*-s are preceded by different *UICCBegin*-s.

We also want to make sure that if the UICC thinks it has completed an authentication session, then the BSF also thinks the same and they also agree on the sequence numbers (and other parameters) they used. To model this property, we add the event *BSFBegin(impi,sqn)* before the last message sent by the BSF to the UICC, and add the event *UICCEnd(impi,sqn)* after the UICC has received this message. We want each *UICCEnd*-event to be preceded by a *BSFBegin*-event with the same parameters. ProVerif can verify that the correspondence property holds, but it cannot show injective correspondence. This is caused by our inability to model that a sequence number can be used only in a single session, and cannot be repeated. From the non-injective correspondence, and from the non-repeatability of sequence numbers we can deduce that in reality, there is an injective correspondence between those events.

4.2 OpenID with GBA

Secrecy. The critical resource is application specific key *KsNAF*. ProVerif verifies that application specific key is still not revealed to the attacker.

Authentication. The main property that we are interested in, is that when a protocol session is finished by RP with a user identified by UID, then there is a user that also participated in this session and this user has a legitimate claim of being identified as UID. Because of the multitude of identities and authorities, the precise property is not trivial to state.

First we note that OpenID with GBA protocol, as we have specified it here, is a *pure authentication protocol* — as the user and the RP are not establishing any common secrets, the only information that the RP gets from the protocol is that a user with a legitimate claim to the identifier UID was alive during the protocol session and used the same OpenID provider that RP thinks it used. Hence we put the event *RPEnd*(*RPID*, *UID*, *OPID*) to the end of RP's process for a protocol session.

The user is using the identifier UID with RP and OP, and the identifier *impi* with the BSF. The identifier *impi* will be bound to the transient value BTID by the BSF. The OP will receive BTID from both the BSF and the user. The OP also received UID from the user and can bind it with BTID. Hence we come up with the requirement that if the RP accepts the UID then a user has started a session as UID and these bindings have been done. The property can be modeled by adding event *UserEnd*(*RPID*, *impi*, *UID*, *OPID*) to the end of the user process (before sending the last message to the RP), adding event *OPEnd*(*UID*, *BTID*, *OPID*) to the end of the NAF-OP process (before sending the last message to the user) and adding event *BSFEnd*(*BTID*, *impi*) to the BSF process (before sending the last message to the OP). Our check with ProVerif shows that the event *RPEnd* must be preceded by all of these three events with matching parameters; the correspondence is even injective. As before, we emit the *RPEnd*-event only if the user and the OP are honest.

Arguably, the property described in previous paragraph does not give strong guarantees to the RP. It ensures that whenever RP accepts a user as UID, there has been a user that has used UID and is correctly identified to the BSF. Two different occurrences of the same UID may in principle come from different users.

We thus also consider the scenario where the RP learns user's real identity (derived from *impi*) from the OP at the end of the protocol [22]. In this case the authentication property is straightforward to model — each event *RPEnd*(*RPID*, *UID*, *OPID*) must be preceded by *UserEnd*(*RPID*, *UID*, *OPID*), where UID is the real identity of the user. ProVerif states that this correspondence property does indeed hold.

Anonymity. We may be interested in the adversary not learning the real identities of the users (i.e. their *impi*-s) participating in the protocol. Such anonymity is a type of process equivalence where the process with a random *impi* is indistinguishable (to the adversary) from the process where *impi* has been selected by the adversary. Such *noninterference* queries are supported by ProVerif, although the treatment is incomplete — ProVerif checks the stronger property of the two processes evolving in lock-step.

For the case where RP does not learn the real identities of users (as the RP may also be adversarial), we have checked whether the process is noninterferent with respect to *impi*. Quite clearly, it is not — *impi* is sent in clear text from the user to the BSF. But when we move this transmission to a private channel, ProVerif succeeds in proving the noninterference.

5 Security against Malicious Participants

The inter-networking of OpenID with GBA, as modeled in section 2.1 and based on the base security model of section 2.3 is proved secure using ProVerif. However, the base security model only reflects an ideal situation. If we modify the security model certain things can go wrong.

5.1 BSF under Adversarial Control

BSF acts as an identity provider both for the user and the NAF-OP (i.e., $BTID$ acts as a transient user identity for both). This means the protocol runs with an honest BSF. Quite clearly, an adversarial BSF can break the protocol. We have modeled this situation by distributing fake certificates of BSF (allowing the adversary to obtain a certificate for the role BSF). In this case, both the user and the OP lack knowledge of the honest BSF. As a result, any rogue client can convince the OP and subsequently the RP that everything is fine.

5.2 NAF-OP under Adversarial Control

The NAF-OP provides an assertion to the RP that a user controls a particular identity. Therefore, an honest NAF-OP is required to prove the validity of the user claim. We modeled a NAF-OP under adversarial control by distributing fake certificates of the NAF-OP. Under such circumstances, the malicious NAF-OP can bypass the user authentication process with the BSF. In addition, it can trick the RP to accept invalid users while rejecting the valid user from gaining access to the service. Another form attack could be, the user is confused about the identity of NAF-OP. This can lead the user to pass user credentials to a malicious NAF-OP. The malicious NAF-OP, in turn, tries to use the user credential to login to a different service. This type of phishing attack is not severe against OpenID with GBA because the user only provides site specific key to the malicious NAF-OP. This meas the attacker cannot use the same user credential for login to a different service.

5.3 Attacker Partially Controls the User Entity

The user entity consists of three modules: UICC, GBA client and an application. The severity of the attack depends on the level of control attacker has over these modules [15]. First, the UICC module can be considered secure because the

UICC card securely contains the the shared master key K. Second, the GBA client contains a temporary master session key Ks. Exposure of this key enables the attacker to login any GBA supported NAF-OP within a particular session. Third, if the attacker controls the application (e.g., browser) then the application specific key $KsNAF$ will be exposed. Exposure of $KsNAF$ enables the attacker to authenticate in a particular NAF-OP within a particular session. We see that the control of any of these three modules directly leads to a successful attack, justifying our coarse-grained model of the user entity.

6 Conclusion

The paper reports the formal specification and security properties of a non-trivial, practical protocol — OpenID with GBA. The security analysis has been performed based on specific security settings and trust relationships among participating entities. The security analysis suggests that the inter-networking of OpenID with GBA is secure in the ProVerif model. The protocol maintains both the secrecy and the correspondence properties. However, the protocol breaks under strong adversarial models. In our security model, one of the main concerns is the identity of the RP due lack of RP certificates. In addition, we have not considered the weaknesses of cryptographic algorithms.

References

1. 3GPP TS 33.220 - Technical Specification Group Services and System Aspects; Generic Authentication Architecture (GAA); Generic bootstrapping architecture (2007), http://www.3gpp.org/ftp/Specs/html-info/33220.htm
2. 3GPP TR 33.924 - Identity management and 3GPP security interworking; Identity management and Generic Authentication Architecture (GAA) interworking (2009), http://www.3gpp.org/ftp/Specs/html-info/33924.htm
3. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: POPL, pp. 104–115 (2001)
4. Ahmed, A.S., Laud, P.: ProVerif model files for the OpenID with GBA protocol (2011), http://research.cyber.ee (last accessed March 30, 2011)
5. Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuellar, J., Drielsma, P.H., Heám, P.C., Kouchnarenko, O., Mantovani, J., Mödersheim, S., von Oheimb, D., Rusinowitch, M., Santiago, J., Turuani, M., Viganò, L., Vigneron, L.: The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 281–285. Springer, Heidelberg (2005)
6. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
7. Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: Proceedings of the 14th IEEE Workshop on Computer Security Foundations, CSFW 2001, Washington, DC, USA, pp. 82–96. IEEE Computer Society (2001)
8. Blanchet, B.: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In: 14th IEEE Computer Security Foundations Workshop (CSFW-14), Cape Breton, Nova Scotia, Canada (2001)

9. Dhamija, R., Dusseault, L.: 7 Flaws of Identity Management: Usability and Security Challenges. In: IEEE Security & Privacy, vol. 6, p. 24 (March 2008)
10. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (August 2008)
11. Dolev, D., Yao, A.C.: On the security of public key protocols. Tech. rep., Stanford, CA, USA (1981)
12. Feld, S., Pohlmann, N.: Security analysis of OpenID, followed by a reference implementation of an nPA-based OpenID provider. In: Information Security Solutions Europe (ISSE) Conference, Madrid, Spain (2008)
13. Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., Stewart, L.: Http authentication: Basic and digest access authentication (1999)
14. Gajek, S., Manulis, M., Pereira, O., Sadeghi, A.-R., Schwenk, J.: Universally Composable Security Analysis of TLS. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) ProvSec 2008. LNCS, vol. 5324, pp. 313–327. Springer, Heidelberg (2008)
15. Holtmanns, S., Niemi, V., Ginzboorg, P., Laitinen, P., Asokan, N.: Cellular Authentication for Mobile and Internet Services., 1st edn. Wiley Publishing Inc. (2008)
16. Laud, P., Roos, M.: Formal Analysis of the Estonian Mobile-ID Protocol. In: Jøsang, A., Maseng, T., Knapskog, S.J. (eds.) NordSec 2009. LNCS, vol. 5838, pp. 271–286. Springer, Heidelberg (2009)
17. Lindholm, A.: Security Evaluation of the OpenID Protocol. Master's thesis, Royal Institute of Technology (KTH), Stockholm, Sweden (2009)
18. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography, 5th edn., vol. 1, ch.10. CRC Press (2001)
19. Morrissey, P., Smart, N.P., Warinschi, B.: A Modular Security Analysis of the TLS Handshake Protocol. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 55–73. Springer, Heidelberg (2008)
20. Niemi, A., Arkko, J., Torvinen, V.: Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA). RFC 3310 (September 2002)
21. Niemi, V., Nyberg, K.: UMTS Security, 1st edn., ch.8. Wiley Publishing Inc. (2003)
22. OpenID Authentication 2.0 - Final (2010), http://openid.net/specs/openid-authentication-2_0.html (last accessed March 30, 2011)
23. Recordon, D., Reed, D.: OpenID 2.0: a platform for user-centric identity management. In: Proceedings of the Second ACM Workshop on Digital Identity Management, pp. 11–16. ACM (2006)
24. Sovis, P., Kohlar, F., Schwenk, J.: Security Analysis of OpenID. In: Freiling, F.C. (ed.) Sicherheit, GI. LNI, vol. 170, pp. 329–340 (2010)
25. Urueña, M., Busquiel, C.: Analysis of a Privacy Vulnerability in the OpenID Authentication Protocol. In: IEEE Multimedia Communications, Services and Security (MCSS 2010), Krakow, Poland (2010)
26. Windley, P.J.: Digital Identity - Ebook edition. O'Reilly Media (2008)
27. Zhang, J., Warkentin, P., Sankhla, V.: AVISPA model for EAP: Extensible Authentication Protocol, http://www.avispa-project.org/library/EAP_AKA.html (last accessed March 30, 2011)