

Security Analysis of the Consumer Remote SIM Provisioning Protocol

ABU SHOHEL AHMED*, Aalto University, Finland

ALEKSI PELTONEN, Aalto University, Finland

MOHIT SETHI, Aalto University, Finland

TUOMAS AURA, Aalto University, Finland

Remote SIM provisioning (RSP) for consumer devices is the protocol specified by the GSM Association for downloading SIM profiles into a secure element in a mobile device. The process is commonly known as eSIM, and it is expected to replace removable SIM cards. The security of the protocol is critical because the profile includes the credentials with which the mobile device will authenticate to the mobile network. In this paper, we present a formal security analysis of the consumer RSP protocol. We model the multi-party protocol in applied pi calculus, define formal security goals, and verify them in ProVerif. The analysis shows that the consumer RSP protocol protects against a network adversary when all the intended participants are honest. However, we also model the protocol in realistic partial compromise scenarios where the adversary controls a legitimate participant or communication channel. The security failures in the partial compromise scenarios reveal weaknesses in the protocol design. The most important observation is that the security of RSP depends unnecessarily on it being encapsulated in a TLS tunnel. Also, the lack of pre-established identifiers means that a compromised download server anywhere in the world or a compromised secure element can be used for attacks against RSP between honest participants. Additionally, the lack of reliable methods for verifying user intent can lead to serious security failures. Based on the findings, we recommend practical improvements to RSP implementations, to future versions of the specification, and to mobile operator processes to increase the robustness of eSIM security.

CCS Concepts: • **Security and privacy** → **Formal security models; Mobile and wireless security; Security protocols; Authentication.**

Additional Key Words and Phrases: mobile communication, eSIM, formal modeling and verification

1 INTRODUCTION

Subscriber identity module (SIM) is the secure element in a mobile phone or device that contains the identifiers and cryptographic credentials with which the *user equipment* (UE) authenticates the mobile subscriber to the *mobile network operator* (MNO). For a long time, the SIM has been a miniature smart card that is inserted into the mobile device, and by changing the SIM, it has been possible to use the same device for different mobile subscribers and networks. However, in new mobile devices, the removable SIM is being replaced by the *embedded SIM* (eSIM). It is a secure element integrated into the circuit board of the mobile device, and it can be programmed with *SIM profiles* that contain the identifiers and credentials.

Remote SIM provisioning (RSP) [31] is the protocol specified by the GSM Association (GSMA) for the purpose of downloading and installing SIM profiles into the secure element. Remote SIM provisioning reduces the logistical and production costs, and it gives subscribers the flexibility of changing operators online [55]. There are two variants of the RSP protocol: the machine-to-machine (M2M) version [32, 33] is used in remotely controllable devices, and the consumer version [34, 35] is used in consumer devices such as smartphones, tablets, and wearables. These two protocol variants differ in how much participation is expected from the end users. The M2M variant does not need interaction with the end user for SIM profile management. In contrast, the consumer RSP protocol requires the end user to trigger

*Corresponding Author

Authors' addresses: Abu Shohel Ahmed, abu.ahmed@aalto.fi, Aalto University, Espoo, Finland; Aleksi Peltonen, aleksi.peltonen@aalto.fi, Aalto University, Espoo, Finland; Mohit Sethi, mohit.sethi@aalto.fi, Aalto University, Espoo, Finland; Tuomas Aura, tuomas.aura@aalto.fi, Aalto University, Espoo, Finland.

the management operations such as remote provisioning, enabling, disabling, and deletion of SIM profiles on the mobile device.

The SIM profile contains security-critical information such as the international mobile subscriber identity (IMSI) and subscriber key K_i , which is a shared secret between the UE and the MNO. The UE uses these credentials from the SIM profile in the authentication and key agreement (AKA) procedure to enable the subscriber's access to the mobile network [2]. Thus, the secure delivery of the SIM profile to the mobile device is of the utmost importance. Unwanted exposure or tampering of the SIM profile or the credentials within it could lead to identity theft, billing fraud, eavesdropping, and various privacy violations against the mobile subscriber. In one documented security failure [50], the attackers captured credentials of SIM profiles by penetrating the internal networks of a SIM manufacturer. In another case [44], the attackers were able to carry out memory-exhaustion attacks against the secure element using flaws in the M2M RSP protocol. These attacks highlight the importance of careful design and analysis for the RSP protocols.

This paper investigates the security of the *consumer RSP protocol*. We develop a formal protocol model and requirements using applied pi calculus and the ProVerif verification tool [16] for security-protocol modeling and verification. Our formal verification shows that the consumer RSP protocol meets the expected security goals in the presence of a network attacker. However, we also analyze the protocol under partial compromise scenarios where one of the system components is not completely trustworthy. This analysis provides insights into the protocol design and also reveals several weaknesses that could lead to security failures in realistic scenarios.

Specifically, our contributions are the following:

- (1) We extract the core message flows and content from the consumer RSP protocol specification [34, 35]. Based on this information, we develop a comprehensive formal model that covers the key protocol variants. To our knowledge, this is the first formal model of the consumer RSP protocol.
- (2) From the specification, we extract both explicit and implicit security requirements that the RSP protocol must fulfill and use them to define formal security goals. Furthermore, we define partial compromise scenarios that will provide interesting new information about the weaknesses of the protocol.
- (3) We present the verification results, which include several unexpected security failures under the partial compromise scenarios. When a formal security goal fails, we analyze and explain the attacks and their root causes in a way that is easy to relate to the protocol specification.
- (4) Finally, we provide practical recommendations for future development of the RSP specification, as well as, implementation guidance to increase its robustness against attacks. The defensive effect of these recommendations is verified in the formal model.

The rest of the article is organized as follows. Section 2 provides background information on protocol modeling, ProVerif, and SIM provisioning security. Section 3 explains the RSP architecture and message flows and the different ways in which the user can initiate profile download. Section 4 describes the formal model and the modeled partial compromise scenarios, and Section 5 defines the formal security goals. Section 6 presents the verification results, as well as, recommendations for improving the specification and its secure implementation. Section 7 discusses the implications of the results and other aspects of the modeling process. Finally, Section 8 concludes the article.

2 BACKGROUND

This section provides an overview of related work on protocol verification, the formal modeling and verification tool ProVerif, and previous research on SIM provisioning security.

2.1 Related work on security protocol analysis

Security protocols are prone to subtle design errors. The academic research in this area started from the publication of the Needham and Schroeder protocol [45] in 1978. Since then, researchers have discovered protocol flaws as well as proposed improved designs and security requirements [5, 14, 23, 38, 40, 56, 57]. Nevertheless, logical flaws continue to appear in security protocols, at least in the early stages of their development. These flaws may be simple mistakes, but sometimes they reflect our improved understanding of the security assumptions and requirements for a specific type of protocols. Because of this history, it is generally agreed that security protocols should undergo a thorough scrutiny by the security research community.

One approach to the quality assurance is formal analysis [17, 39, 41, 42], which can verify that the protocol meets well-defined security goals or uncover weaknesses. A common approach to formal analysis is symbolic model checking [8]. Model checking tools such as ProVerif [16], Tamarin [43, 51], and DeepSec [18] perform automated verification of protocol models against formalized security goals. For example, Bhargavan et al. [12] used ProVerif to verify the Transport Layer Security (TLS) 1.3 [49] standard candidates, leading to modifications in the protocol. Similarly, Cremers et al. [22] used the Tamarin model checker and found possible attacks on certificate-based and pre-shared key authentication in TLS. They also modeled the full protocol [21]. Basin et al. [9–11] formalized and verified the security goals for the fifth-generation (5G) mobile network authentication and key agreement (AKA) [1] protocol and the EMV payment card protocol. Their work also supported the standardization by formalizing the security requirements and by proposing improvements. These examples show the benefits of formal modeling and verification. However, written protocols specifications are not always easy to convert into precise formal models, and the adversary model and security goals for the protocols are not always fully understood. Moreover, the number of protocol variants that needs to be modeled can be high; for example, Cremers [20] analyzed over forty variants of the Internet key exchange (IKE).

2.2 Applied pi calculus and the ProVerif tool

ProVerif [16] is a tool for modeling and automatic verification of cryptographic protocols. The protocol participants are modeled as communicating processes in applied pi calculus [4]. The security goals are formalized as queries that will be evaluated against the protocol model. In particular, authentication can be formalized as a correspondence between local events in the concurrent processes. ProVerif has built-in support for the network adversary, i.e., the Dolev-Yao type adversary [25], which is in full control of the network and can read, modify, delete, and inject messages and perform cryptographic computations on them. When the verification of a security goal fails, the tool tries to construct an attack trace that shows how the adversary can violate it. The models are symbolic, and they treat cryptographic primitives, such as signatures and encryption, as abstract functions that are assumed to be secure. This can be contrasted with computational models [15], which consider the probability of breaking cryptographic primitives. The symbolic approach is good for finding logical flaws in the protocol design.

2.3 SIM provisioning security

In standard terminology, the older SIM is a universal integrated circuit card (UICC), and the embedded version is an *embedded UICC* (eUICC). Traditionally, a SIM profile is provisioned to the UICC in the smart-card factory. One provisioned UICC is inserted into the user equipment (UE) for each mobile-network subscription. This approach requires physical delivery of the SIM card, and it also limits the user's flexibility in changing subscriptions. An earlier attempt to overcome these shortcomings was the virtualized *soft SIM* [47]. It provides similar functionality to the removable SIM card but without hardware-based secure credential storage. The soft-SIM approach was not adopted widely due to its perceived lack of security [36]. RSP aims to provide the same level of security as the removable SIM card by storing the SIM profile in the eUICC, which is an embedded secure element in the mobile device.

So far, security analysis of RSP has focused on the eUICC component and the M2M variant of the protocol. Vahidian [54] analyzed the security of the eUICC. The threat analysis by GSMA identifies several risks for the M2M RSP [32]. Meyer et al. [44] discovered weaknesses in the M2M RSP [33] related to denial of service by exhausting the capacity of the eUICC. Ding et al. [24] model M2M RSP with SPIN [37], which is, however, a general-purpose model checker and not especially suitable for verifying security properties [46]. Park et al. [48] proposed an alternative to the GlobalPlatform secure channel protocol (SCP) [28, 29], which is also used in RSP, with potential scalability and performance gains.

3 CONSUMER REMOTE SIM PROVISIONING PROTOCOL

This section gives an overview of the consumer RSP protocol. We explain the protocol participants, trust relations among them, and message flows between the participants. The information has been collected from the GSMA architecture and technical specifications [34, 35] and the GlobalPlatform secure channel protocol specification [29], totaling 446 pages. The M2M RSP protocol specification [32, 33] was also used as background information.

3.1 Consumer RSP participants and phases

Figure 1 shows the participants of the RSP protocol, their trust relations, user interactions, and the message flow for the profile download. The active participants are:

- (1) *Mobile network operator* (MNO) provides mobile network access to users.
- (2) *User equipment* (UE) connects to the mobile network.
- (3) *Local profile assistant* (LPA) is a software application that runs on the UE.
- (4) *User* owns the UE and has a subscription with the MNO.
- (5) *eUICC* is an embedded secure hardware module inside the UE. It stores the *SIM profiles*.
- (6) *Subscription manager data preparation* (SM-DP+) *server* (or just *server*) is a new entity in the RSP protocol.

To connect to the MNO's mobile network, the UE must have in its eUICC a SIM profile with the necessary credentials. To obtain the SIM profile, the user first contacts the MNO; this can take place on the web, with the help of the LPA application, or offline. The MNO initiates the RSP protocol by sending the profile order to the SM-DP+ server. The server prepares the SIM profile and makes it available for download. The user then asks the LPA application to download the SIM profile from the server to the eUICC.

We describe the protocol in two parts: (1) the profile ordering and download initialization phases, and (2) the common handshake and profile download phases. Variants of the former are shown in Figures 2 and 3, and the latter is in Figure 4. Tables 1–3 summarize the notation used in the protocol description. (In the figures, black text represents the messages

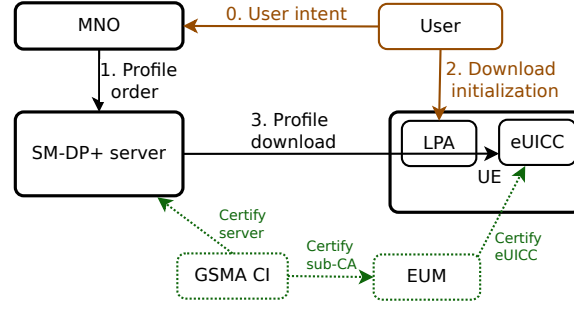


Fig. 1. Participants and their interaction in the consumer remote SIM provisioning protocol

and processing steps, brown text represents user actions, and blue text represents the events in the ProVerif model, which will be introduced later.)

3.2 Trust relations and adversary model in the RSP specification

In RSP, both the SM-DP+ server and eUICC trust a common GSMA *certificate issuer* (CI). The CI issues certificates to the SM-DP+ servers and to *eUICC manufacturers* (EUM). The EUM acts as a subordinate certificate authority and issues certificates to eUICCs. Consequently, the SM-DP+ server and the eUICC can establish mutual trust using certificate chains that start from a common root authority.

The RSP specification assumes that the CI, SM-DP+ server, EUM, eUICC, and MNO are all trusted participants. In other words, the adversary has no access to their secret keys and cannot control their operations. In particular, the specification assumes that the adversary cannot obtain a certificate for an arbitrary eUICC identifier and key. The LPA running in the UE is a trusted application that is assumed not to leak any information to the adversary.

The RSP specification mainly considers a *network adversary*, that is, an attacker who can intercept and alter network communications between the RSP participants. The adversary can also trigger a profile download for a small number of eUICCs in mobile devices which the adversary owns.

3.3 Profile ordering and download initialization phases

In the profile ordering phase, an MNO requests a server to prepare a SIM profile. The server then prepares the profile and returns download initialization pointers (e.g., an activation code) to the MNO. In the download initialization phase, the MNO delivers the download initialization pointers to a user and their LPA software to trigger the profile download. The RSP specification supports three different approaches to profile ordering and download initialization: (a) default server, (b) activation code, and (c) subscriber manager discovery server (SM-DS) assisted. For brevity, we model the first two approaches and defer discussion of the third one to Section 7.4.

Default server approach. Figure 2 shows the default-server approach. The UE manufacturer or reseller, which can be the MNO, pre-provisions the default SM-DP+ server domain name S to the eUICC. When the user intends to get a mobile subscription, the user contacts the MNO. The MNO orders a profile for the specific eUICC identifier U to the default server. Consequently, the server prepares a new SIM profile P for the eUICC. The MNO notifies the user, who starts the LPA software in the UE. The LPA retrieves the default SM-DP+ address S from the eUICC. The LPA is now

Table 1. Certificates in RSP

| Id, certificate, key | Subject | Purpose | GSMA reference |
|-----------------------------|---------|---------------------------|------------------------------------|
| $CI, Cert_{CI}, SK_{CI}$ | GSMA CI | Root CA | CERT.CIECDSA, SK.CIECDSA |
| $S, Cert_S, SK_S$ | SM-DP+ | TLS authentication | CERT.DP.TLS, SK.DP.TLS |
| $Sa, Cert_{Sa}, SK_{Sa}$ | SM-DP+ | Server authentication | CERT.DPauth.ECDSA, SK.DPauth.ECDSA |
| $Sp, Cert_{Sp}, SK_{Sp}$ | SM-DP+ | Profile binding | CERT.DPpb.ECDSA, SK.DPpb.ECDSA |
| $U, Cert_U, SK_U$ | eUICC | eUICC authentication | CERT.U, SK.U |
| $EUM, Cert_{EUM}, SK_{EUM}$ | EUM | eUICC Manufacturer sub-CA | CERT.EUM, SK.EUM |

Table 2. Notations and symbols

| Symbol | Description | GSMA reference |
|-------------------------------|--|----------------|
| U | eUICC identifier | EID |
| S | Server domain name | |
| $mnold$ | MNO identifier | |
| $userId$ | User identity authenticated by MNO | |
| P | SIM profile including the secret key K_i for AKA | |
| $iccid$ | Unique identifier of the SIM profile | ICCID |
| I_{ac} | Matching identifier in activation code | |
| SKI_{CI} | SubjectKeyIdentifier of $Cert_{CI}$ | |
| Seq | Sequence counter for profile install notifications | |
| $N_U \sim R, N_S \sim R$ | Random numbers generated by eUICC and server | |
| $I_t \sim R$ | Random session identifier generated by server | |
| $d_U, Q_U = d_U \cdot G$ | eUICC ECDHE key components | |
| $d_S, Q_S = d_S \cdot G$ | Server ECDHE key components | |
| Z_{US} | ECDHE shared secret between eUICC and server | |
| KDF | Key derivation function | |
| k, k' | Encryption and integrity keys for server and eUICC | S-ENC, S-MAC |
| $E(M; k)$ | Encryption of message M with symmetric key k | |
| $MAC(M; k')$ | MAC of message M with symmetric key k' | |
| $Sign(M; SK)$ | Signature of message M with private key SK | |
| $serverOID$ from $Cert_x$ | Server OID in $Cert_{Sa}$ and $Cert_{Sp}$ | |
| subject of $Cert$ | Subject identifier to which $Cert$ was issued | |
| $Cert_a \triangleleft Cert_b$ | $Cert_a$ is signed by the subject key of $Cert_b$ | |
| $:=, =$ | Assignment operator, equality comparison | |

Table 3. ProVerif events

| Event and their parameters | Description |
|---|--|
| AUTHORIZE(Sp) | GSMA authorizes the server to issue profiles |
| OWNER($userId, U$) | User is the owner of an eUICC |
| INTENT($userId, mnold, U, I_{ac}$) | User requests a profile from an MNO |
| ORDER($userId, mnold, S, U, P, I_{ac}$) | Server receives an order for a profile from MNO |
| U0(U, S) | eUICC begins authentication of the server |
| U1(U, Sa, I_t, S) | eUICC completes authentication of the server |
| U2(U, Sa, Sp, I_t) | eUICC accepts the server as a profile download entity |
| U3($U, Sa, Sp, I_t, k, P, mnold, I_{ac}$) | eUICC accepts session keys and accepts a profile from the server |
| S0($Sa, I_t, S, mnold, I_{ac}$) | Server begins authentication of the eUICC |
| S1($U, Sa, Sp, I_t, mnold, I_{ac}$) | Server completes authentication of the eUICC |
| S2($U, Sa, Sp, I_t, k, P, mnold, I_{ac}$) | Server starts key agreement and profile delivery |
| S3($U, Sa, Sp, I_t, P, S, mnold$) | Server accepts profile download notification |

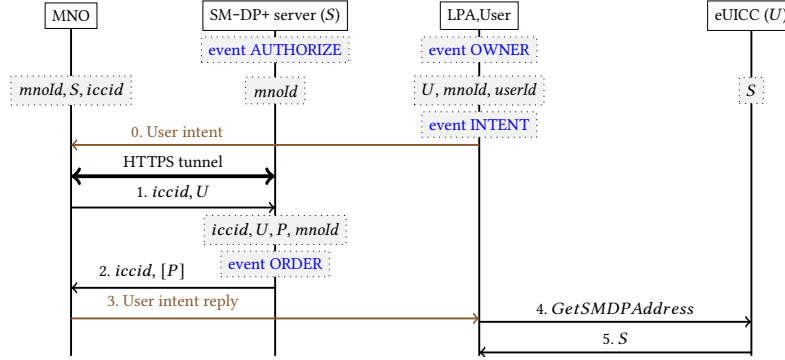


Fig. 2. Default-server approach to profile ordering and download initialization: the server stores the eUICC identifier U , and the LPA reads the default server domain name S from the eUICC

ready to move to the common handshake and profile download phases. The profile will be available for download at the address S only for the eUICC with the certified identity U .

Activation code approach. Figure 3 shows the activation-code approach. The MNO orders SIM profiles to a server. The server prepares the profiles and makes each profile available with an activation code, which it returns to the MNO. The activation code includes the server domain name S , a matching identifier I_{ac} that is unique to the profile, and optionally, a server OID, which is a unique identifier for the SM-DP+ server. In this approach, the profile preparation can be done either before or after the user contacts the MNO for a new subscription. When the user intends to get a mobile subscription, the MNO delivers an activation code to the user. The user inputs the code into the LPA software, for example, by scanning a two-dimensional bar code. The profile can then be downloaded by anyone who knows the activation code. The matching identifier in the activation code has to be secret and sufficiently random to be unguessable.

In a variant of the activation-code approach, the user or LPA communicates the eUICC identifier U to the MNO, which passes it on to the server. In this case, the profile can be downloaded only to the specific eUICC, which must have both the certified identity U and the matching identifier I_{ac} . Pre-establishing the eUICC identifier is practical when the user requests the profile from the MNO with the LPA software and the MNO and SM-DP+ server are working closely together. It is not convenient to ask the user to communicate the eUICC identifier to the MNO.

In the security analysis, we first assume that the activation code does not include the server OID and that the server does not know U . We then explain how these optional features would improve the protocol security.

TLS and out-of-band channels. In all these approaches, the communication between the MNO and the SM-DP+ server is protected with mutually authenticated TLS [49]. The MNO is the TLS client, and the SM-DP+ server is the TLS server. Both have certificates [19]. Trust for this channel is established earlier, so that both sides know the other's certified name.

The user's communication with the MNO may be protected in several different ways: (1) the user goes in person to a shop that represents the MNO and requests a profile; (2) the user logs into an MNO portal online and request a profile; or (3) the user requests a profile with the help of the LPA or another software application. In the first case, the communication occurs out-of-band, and in the other two cases, over a TLS connection where the MNO is

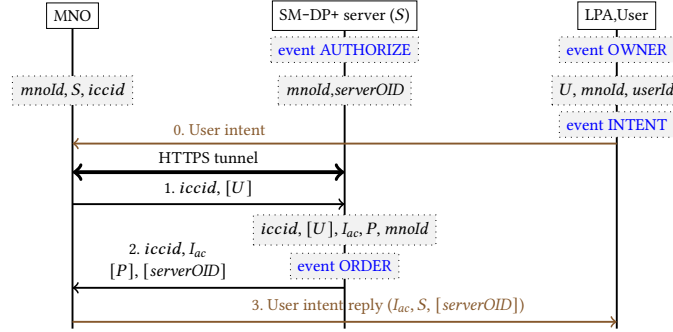


Fig. 3. Activation-code approach to profile ordering and download initialization: LPA stores the matching identifier I_{ac} , the server address S , and optionally the server OID; the server optionally stores the eUICC identifier U

authenticated with a web server certificate. The user must be authenticated as well. The method and strength of the user authentication depend on the local regulation and business decisions. In the activation-code approach, the activation code ($I_{ac}, S, [serverOID]$) is communicated to the user and LPA as part of the same protected interaction.

3.4 Common handshake and profile download phases

When the profile has been ordered and the download initialized, it is time for the common handshake and profile download phases [35, Section 3.1.2]. These are shown in Figure 4.

There are three main entities involved in these phases: the eUICC, the LPA and user, and the SM-DP+ server. The certificates used in the handshake are summarized in Table 1. The server has three different certificates: $Cert_{St}$ is the TLS server certificate for the domain name in the server address S . $Cert_{Sa}$ is used to authenticate the server to the eUICC, and $Cert_{Sp}$ authorizes the server to issue SIM profiles. The last two certificates have the same subject name, and they contain the same server OID; however, they contain different certificate policy identifiers. $Cert_{St}$ may also contain a server OID, but it is not compared with the other two OID values because the certificate is verified by the TLS layer in the client and not by the RSP protocol implementation. The GSMA CI directly issues all the server certificates. The eUICC has a certificate chain consisting of two certificates $Cert_U$ and $Cert_{EUM}$. The latter is an equipment manufacturer certificate issued by the GSMA CI, and the former is the eUICC certificate issued by the manufacturer to the eUICC identifier U .

At the beginning of the profile download phase, the server always has the profile P and the associated MNO identifier $mnold$. The server also has the receiving eUICC identifier U or a matching identifier of an activation code I_{ac} , or both. The LPA software knows the server address S , and it may have a matching identifier I_{ac} and the server OID.

We will now explain the common handshake and profile download phases in Figure 4 step by step.

TLS tunnel. The protocol takes place inside a TLS tunnel from the LPA to the SM-DP+ server, which is authenticated with the server certificate $Cert_{St}$. The TLS client is not authenticated. The ways in which the LPA can learn the server domain name S were discussed above in Section 3.3.

Mutual authentication (messages 1–7). The eUICC and the server authenticate each other inside the TLS tunnel. In the base case, these endpoints do not have prior knowledge of each other's identities U and Sa . The eUICC indicates in message 2 its preferred CI root key with SKI_{CI} . This key identifier also determines the elliptic curve that will be

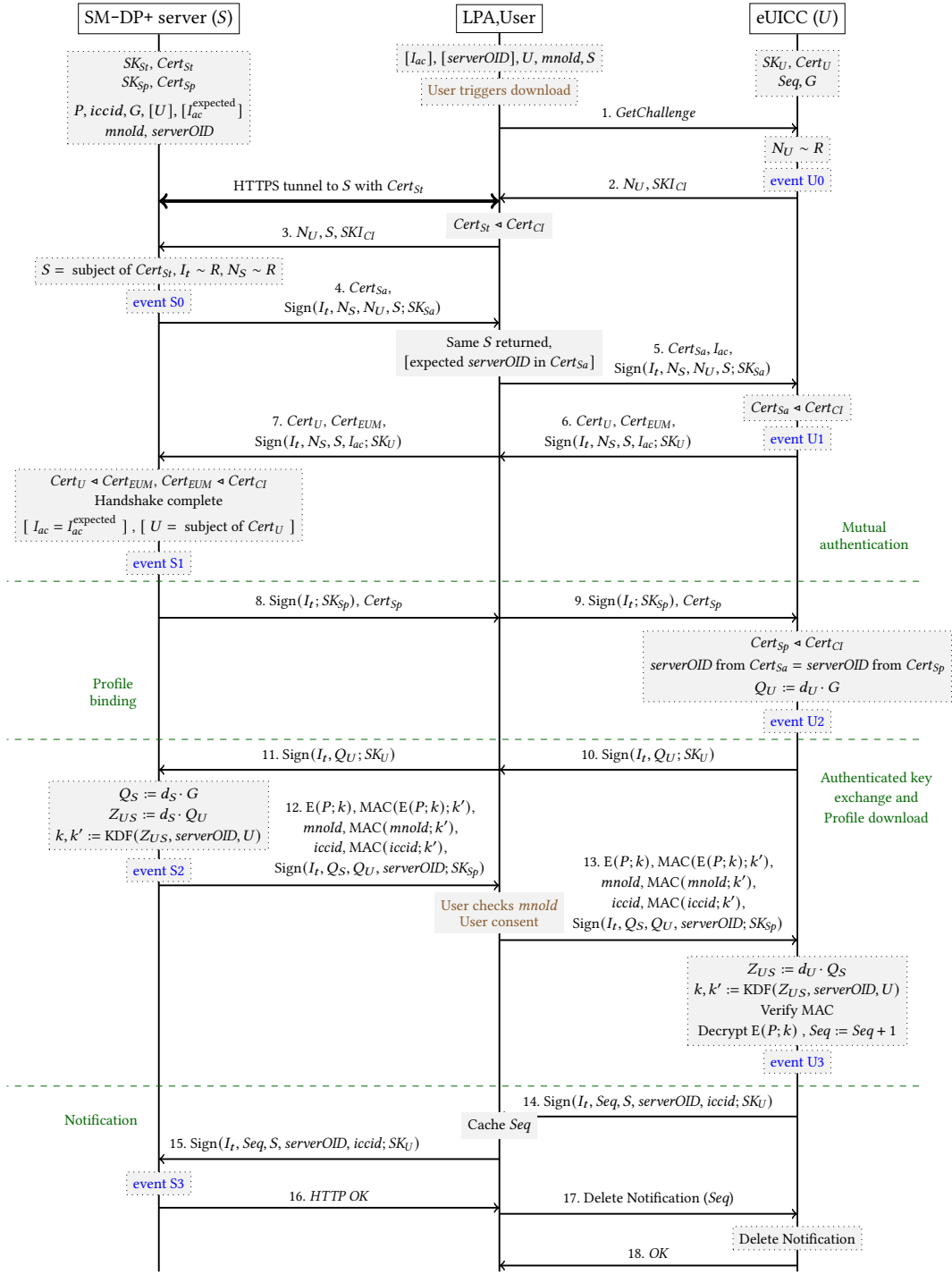


Fig. 4. Common handshake and profile download in the RSP protocol

used for all the certificates, signatures, and ECDH key exchange throughout the protocol. The server accepts any valid eUICC certificate chain $Cert_U$, $Cert_{EUM}$ that originates from the root key, and the eUICC accepts any valid server certificate $Cert_{Sa}$ issued by the same root key. The nonces N_S and N_U ensure the freshness of the mutual authentication. During the authentication, the server selects a fresh session identifier I_t , which will be used to identify the ongoing session throughout the rest of the protocol.

The server includes the server domain name S from its TLS certificate in the signed message 4. Before passing the message to the eUICC, the LPA verifies that S matches the server name to which it created the TLS tunnel. This provides a level of channel binding between the TLS tunnel and the inner authentication. If the LPA knows the server OID beforehand, it verifies that the server certificate $Cert_{Sa}$ contains the correct *serverOID*.

In the activation-code approach, the LPA gives the matching identifier I_{ac} to the eUICC in message 5, and the eUICC includes it in the signed message 6–7 to the server. The server then selects the correct profile for the eUICC based on the matching identifier. In the default-server approach, the LPA sets the matching identifier to a null value and the server selects the profile based on the eUICC identifier U in $Cert_U$.

Profile binding (messages 8–9). After the mutual authentication, the server checks that it has a profile available for download to the eUICC. If it does, it signs the session identifier I_t with the profile binding key and sends the signature together with the profile-binding certificate $Cert_{Sp}$ to the eUICC. The eUICC verifies the signature and certificate, and it also checks that the two server certificates $Cert_{Sa}$ and $Cert_{Sp}$ contain the same server OID. This step assures the eUICC that the previously authenticated server is an authorized SM-DP+ server.

Authenticated key exchange (messages 10–13). The key exchange is based on elliptic-curve key agreement (ECKA) [26, 29]. The eUICC and server generate ephemeral elliptic-curve Diffie-Hellman (ECDHE) key components d_U , $Q_U = d_U \cdot G$ and d_S , $Q_S = d_S \cdot G$, respectively. The eUICC and server exchange the public components in messages 10–13 and then compute the shared secret $Z_{US} = d_S \cdot Q_U = d_U \cdot Q_S$. From this shared secret, they generate session keys k and k' with the key derivation function (KDF) [26, Section 4.3.3]. Other inputs to KDF are the server OID and the eUICC identifier U . The key exchange is authenticated by the signatures and certificates in messages 10–11 and 12–13. In the following download phase, k will be used for data encryption and k' for data authentication.

Profile download (messages 12–13). This phase overlaps with the last message of the authenticated key exchange. The server performs authenticated encryption [33, Section 4.1.3] of the profile P . That is, it first encrypts the profile with the key k and then appends a message authentication code (MAC) computed with the key k' . The server sends the encrypted profile $E(P; k)$ and the MAC to the LPA. It also sends the *mnoid* (and other metadata that is unimportant for the current discussion) protected with a separate MAC.

The *mnoid* identifies the mobile network operator that asked the server to prepare the profile. The LPA displays *mnoid* to the user, who should check that it refers to the intended operator. Once the user has approved the operator, the LPA forwards the information to the eUICC. Consequently, the eUICC computes the session keys k , k' , checks the MACs, and decrypts the received profile P . The eUICC can now take the profile P into use.

Notification (messages 14–17). After accepting the downloaded profile, the eUICC deletes the session keys k , k' and notifies the LPA with a signed notification message. The LPA may deliver the notification to the server immediately or after a delay. The signed message contains the server address and OID as well as a sequence number Seq . For this, the eUICC maintains a sequence counter, which it stores in non-volatile memory and increments by one for each new notification. The server checks that the received sequence number is greater than the previous one received from the

same eUICC. The server then responds with *HTTP OK* to the LPA. The LPA asks the eUICC to delete the notification from its queue, and the eUICC responds with *OK* status to the LPA.

4 FORMAL MODEL OF RSP SECURITY

This section gives an overview of the ProVerif model¹, defines the partial compromise scenarios, and discusses some of the abstractions and security assumptions made in the model. Figure 5 shows the participants and communication channels in the model.

4.1 Model overview

Participants. We explicitly model four participants of the RSP protocol as ProVerif processes: (1) MNO, (2) SM-DP+ server (3) user and UE with LPA software, and (4) eUICC. The roles of the user, UE and LPA have been merged into a single process because their integrity is mutually dependent: on one hand, the user could choose a hacked phone as the UE or install modified LPA software, and on the other hand, the faithful execution of the user’s intent depends on both the LPA software and UE hardware. In comparison, even though the eUICC is in the user’s possession, it is a hardware module with a security perimeter that protects its local integrity and secrets.

Communication channels. We model the communication channels between the participants as follows. The *User/LPA-to-MNO* and *MNO-to-server* channels are modelled as private channels to which the adversary has no access; their implementation is out of the scope of the analysis. However, some of the partial compromise scenarios represent specific types of failures of this channel, such as identity fraud where the adversary is able to impersonate the user on the *User/LPA-to-MNO* channel. The *User/LPA-to-server* channel is the main target of our investigation. We model it as a public channel that is vulnerable to the network adversary. The server-authenticated TLS tunnel on the *User/LPA-to-server* channel is modelled explicitly. Finally, the *User/LPA-to-eUICC* channel is a pairwise secure channel inside the UE.

Events. Table 3 lists the events emitted by the protocol participants that will be used for specifying the security goals. The same events occur in Figures 2–4. The event *AUTHORIZE* indicates the legitimacy of the SM-DP+ server, and *OWNER* indicates the user’s ownership of the device that contains the eUICC. *INTENT* models the user’s decision to obtain a SIM profile from the MNO, which triggers the RSP execution. The events *OWNER* and *INTENT* are linked together by a parameter *userId*, which is an abstract representation of the authenticated user identity and does not appear in the RSP messages. The server emits the event *ORDER* when it receives the profile order from the MNO. The remaining events *U0–U3* and *S0–S3* signal the lockstep progress of the SM-DP+ server and the eUICC through the common handshake and profile download phases. *S2* and *U3* respectively indicate the completion of the critical parts of the protocol at the server and eUICC.

4.2 Assets that require protection

The focus of the model is on the confidentiality and integrity of the SIM profile. The profile is initially known only to the SM-DP+ server and MNO. After the eUICC downloads the profile, it is assumed that the server forgets the data and the eUICC stores it securely. With these assumptions, the primary goal of the RSP protocol is to establish the profile as an authenticated shared secret between the MNO and the intended eUICC.

¹The model is available at https://github.com/peltona/rsp_model

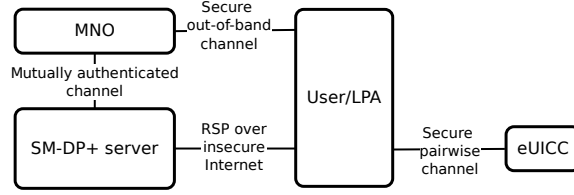


Fig. 5. Participants and channels in the protocol model

The profile is modeled as an opaque block of data P . That is, the contents of the profile are not modeled explicitly. For simplicity, the reader may want to think of the profile as the shared key K_i that will be used for the AKA procedure [3, 6] between the UE and the MNO to authorize mobile network access. The profile also contains other data, such as the subscriber and MNO identifiers, which require integrity protection.

4.3 Adversary model and partial compromise scenarios

The Dolev-Yao adversary model, which is built into ProVerif, assumes that the endpoints are trusted while the underlying network is untrusted. That is, the communication network is assumed to be under the adversary's control.

Given the use of TLS tunnels for network communication and out-of-band authentication between the user and the MNO, we do not expect major security failures against the base-case network adversary. Nevertheless, it is necessary to verify that the TLS tunnels and the protocols executed inside these tunnels combine to provide the desired security properties.

More importantly, we extend the network adversary in the following scenarios by modeling the compromise of selected endpoints and channels:

1. **base-case:** All participants are honest. The network between the LPA and SM-DP+ server is untrusted.
2. **server:** The adversary has all the private keys of the SM-DP+ server.
3. **eUICC:** The adversary has the private key SK_U of the honest user's eUICC.
4. **LPA:** The adversary is the user who controls the LPA, or compromised LPA software misleads the user.
5. **2nd server:** The intended SM-DP+ server is honest, but there exists another compromised server.
6. **2nd eUICC:** The adversary has compromised the private key of the eUICC in its own device.
7. **2nd MNO:** The intended MNO is honest, but there exists another compromised MNO.
8. **order as user:** The adversary impersonates the honest users when ordering a profile.
9. **order for eUICC:** The adversary requests a profile for the honest user's eUICC.
10. **AC leak:** The adversary learns the activation code associated with an ordered profile.
11. **AC spoof:** The adversary tricks the LPA or user to use the wrong activation code.

In scenarios 2–3, one of the main communication endpoints, either the SM-DP+ server or eUICC, is compromised. This is modelled by leaking the endpoint's private keys to the adversary. In practice, the server may have been hacked, or the eUICC keys may have been compromised in the supply chain. A serious security failure is expected in these scenarios. They are included in the analysis because we can learn about the failure modes — and also as sanity checks for the model.

In scenario 4, the adversary is the mobile device user. They can install modified LPA software and even UE firmware, but unlike in the previous scenario, they have not compromised the integrity of the eUICC hardware inside the UE. This scenario also covers the case where an honest user performs all their actions, including the profile ordering, with compromised LPA software.

Scenarios 5–7 represent situations where the adversary has compromised one SM-DP+ server, eUICC, or MNO and uses that to attack RSP between honest participants. The existence of the compromised MNO should not influence the security of other MNOs and their customers — not even if they use the same SM-DP+ server. Likewise, the existence of a compromised server should not influence the security of those who use a different server. Also, an advanced adversary could compromise the integrity of a small number of eUICCs and extract their private keys in the laboratory, but that should have no effect on the security of other eUICCs.

Scenarios 8–11 model four different types of fraud during the profile ordering. In scenario 8, the adversary can impersonate the honest user in the ordering phase, and in scenario 9, the adversary knows the identifier of the honest user’s eUICC, which enables it to order profiles for the eUICC. In scenario 10, the adversary spies the activation code from a careless user, and in scenario 11, the adversary tampers with the activation code that is given to the user or LPA. While the adversary in these scenarios is relatively weak, these scenarios are likely to occur in the real world.

In all the scenarios including the base case (scenario 1), the adversary has its own user identity and eUICC, for which it can order profiles with the RSP protocol. This represents the adversary’s ability to purchase a mobile phone and a mobile network subscription for it.

We model all these scenarios *with and without TLS tunnel* between the LPA and SM-DP+ server. The primary reason is to investigate how important the TLS layer is for the security of RSP. Another reason is that the TLS server is exposed directly to Internet and its the private TLS key SK_S is thus susceptible to hacking. This is more likely than the full compromise of the server (scenario 2) because the application keys SK_{Sa} and SK_{Sp} are typically stored in a secure hardware module or in a backend server.

When an entity’s private key leaks in scenarios 5-7, the model emits a marker event, e.g. $event(CompromiseCert(U))$ in the case of a compromised eUICC. The formal verification goals will use these events to differentiate between the compromised entity and honest ones of the same type. Scenarios 8-9 emit similar marker events to differentiate between the dishonest and honest orders.

We generate separate models for each of the two profile ordering and download initialization approaches in the relevant partial compromise scenarios. In order to understand the significance of the TLS tunnel between the UE and server, we generate separate model variants with TLS and without. The results is 40 different ProVerif models.

4.4 Abstractions made in the model

In the symbolic protocol model, cryptographic primitives including signature, key agreement and encryption algorithms are modelled as simple rewriting rules without considering their cryptographic implementation. Some of the message fields are abstracted away: we omit eUICC capabilities, cryptographic algorithm suites, message lengths, profile contents, and error codes. The activation code never expires in our model. Also, the eUICC certificate in the model is issued directly by the CI (and not by the EUM), so that we avoid modeling the two-certificate chain.

ProVerif supports unlimited process replication for verifying protocols with unlimited number of participants and unlimited number of sessions between them. All the positive verification results reported in this paper were verified with *unlimited participants* and *unlimited sessions*. To complete some verification tasks, we had to simplify the model by combining the User/LPA and eUICC into one process. This does not limit the generality of the results because the

optimization was necessary only in cases where ProVerif found an attack but needed help in reproducing the attack trace.

5 FORMALIZATION OF THE SECURITY GOALS

This section formalizes the security goals of the RSP protocol. The RSP specification [35, sec. 2.6.2] defines some explicit security requirements, and implicit requirements exist throughout the specification. We derive precise formal security goals from both types of requirements. We then use ProVerif to verify the goals against the formal protocol model described above. In the following, the message numbers and events refer to Figures 2–4.

There are both authentication and secrecy goals. The *authentication goals* are expressed as correspondences between local events in the participating processes. The event parameters represent local variable values in the process where the event occurs. The *secrecy goals* in sections 5.3 and 5.5 state that a session key or profile accepted by one honest participant must not be known to the adversary. The events and their full parameters are listed in Table 3. In the security goals below, we use the underscore $_$ to denote event parameters that are unused or unimportant for the security goal in question. In the formal queries, each $_$ on the left side of the \Rightarrow can be read as a universally quantified wildcard variable and, on the right side, as an existentially quantified one.

The security goals can also be divided into server-side and client-side ones. The *server-side* goals formalize the assurance that the server gets from the protocol execution, including authentication of the client; they can be recognized from having a server event to the left of \Rightarrow . The *client-side* goals express the assurance that the client gets, including authentication of the server; they have a client event to the left of \Rightarrow .

5.1 Mutual authentication

The first two authentication goals are based on explicit requirements from the RSP specification [35]:

“The communication SHALL be origin authenticated, as well as integrity and confidentiality protected.”

“The Server (the entity providing the function, e.g. SM-DP+) SHALL be authenticated first by the Client. Authentication SHALL include the verification of a valid Server Certificate signed by a GSMA CI.”

“The Client SHALL be authenticated by the Server in a second step. In case the Client is the eUICC, authentication SHALL include the verification of a valid eUICC and EUM Certificate signed by a GSMA CI.”

We focus here on the authentication and integrity requirements. The mutual entity authentication takes place in messages 1–7 (see Figure 4). It is formalized as the following correspondence properties:

$$\text{event}(U1(U, Sa, I_t, S)) \Rightarrow \text{inj-event}(U0(U, S)) \ \&\& \ \text{inj-event}(S0(Sa, I_t, S, _)) \quad (\text{Auth A})$$

$$\text{event}(S1(U, Sa, _, I_t, mnold, I_{ac})) \Rightarrow \text{inj-event}(S0(Sa, I_t, _, mnold, I_{ac})) \ \&\& \ \text{inj-event}(U1(U, Sa, I_t, _)) \quad (\text{Auth B})$$

Auth A and Auth B above are *stepwise* goals in the sense that they require a correspondence over single steps of the protocol execution. Transitivity, these properties say that, if the client or the server accepts the mutual authentication (event U1 or S1, respectively), then both sides must have started the authentication (both events U0 and S0).

In addition to the stepwise goals, it is possible to specify transitive correspondence between the latest event and initial events that triggered the protocol execution. We have chosen to include here one such transitive goal, Auth B',

because it provides some interesting verification results.

$$\begin{aligned}
 \text{event}(S1(U, Sa, _, I_t, mnold, I_{ac})) &\Rightarrow \text{event}(\text{OWNER}(userId, U)) \\
 \&\& \text{event}(\text{INTENT}(userId, mnold, U, I_{ac})) && (\text{Auth B}') \\
 \&\& \text{inj-event}(\text{ORDER}(userId, mnold, _, [U], _, I_{ac}))
 \end{aligned}$$

Auth B' specifies a correspondence between the end of the authentication step and the initial user intent and profile order. The correspondence with ORDER is injective, meaning that event S1 should occur at most once for each profile order received by the SM-DP+ server. The user identity *userId* and the concept of ownership do not occur in the RSP specification; they are needed in the formal model to express the user's intent to download a profile for a specific eUICC. In the activation-code approach, the ORDER event in the server is associated with an eUICC identifier ([*U*] in Auth B' denotes *U*), while in the default-server approach, the order is not made for a specific eUICC ([*U*] in Auth B' denotes *null*). In the activation-code approach, the server must accept only the *I_{ac}* value registered in the profile ordering phase. In the default-server approach, there is no such value (*I_{ac}* = *null*).

The advantage of formal verification is that it forces us to be precise about the security goals. Above, we had to precisely state what knowledge each participant has at the events, which leads us to several important observations.

First, the formalization makes it explicit that the initial client event U0 does not depend on the server OID or certificate *Sa*, and that the initial server event S0 does not depend on the client identity *U*. This draws our focus onto the fact that, in most deployments, neither side knows the identity of the other at the start of the common handshake phase. Instead, the client is willing to accept any server OID that is certified by the GSMA CI, and the server is willing to accept any client identity *U* that has a certificate chain with GSMA CI as the root. If we included the identifiers as parameters in the initial events in Auth A and Auth B, we would not be able to verify these goals against the protocol model — or more accurately, we could verify them only against the variant of the model where the identities are communicated through a secure channel in the profile download and initialization phases.

Second, the formalization reveals weaknesses in the channel binding to TLS. The LPA in the client knows the HTTPS address *S* of the server, but there is no logical binding between *S* and the server identifier *Sa* or OID authenticated by the eUICC. We could have included *S* as parameter in the events in Auth B and then presented the attacks that violate this security goal. For example, by tampering with the LPA in its own mobile device, the adversary can get the eUICC in that device to use a different SM-DP+ server from the default one expected by the eUICC. Also, the intended SM-DP+ server could forward the protocol messages to another server through a second HTTPS tunnel without being detected. We have chosen to present a weaker version of Auth B above to pinpoint what that protocol actually achieves.

After specifying a correspondence property, we sometimes find that the verification produces an attack scenario that is uninteresting. For example, the attacker can technically violate Auth B by swapping the authentication codes between two of its own mobile devices, but there is no adverse real-world impact from this. We can ignore such uninteresting attacks scenarios by adding disjunctive terms to the right side of the correspondence. There is no one right way to specify these exclusions, and experimentation is needed. Excluding a minor attack may lead to the successful verification of the security goal, but sometimes it leads to the discovery of more serious failure. To preserve the readability of the security goals, we have omitted the disjunctive terms from the formulas presented here and refer the reader to the formal model.

5.2 Profile binding

After the mutual entity authentication, the server proves to the eUICC that it is authorized to deliver a profile by signing the session identifier I_t with its GSMA-issued profile binding certificate $Cert_{Sp}$. We formalize the binding as the correspondence Auth C, which links the subject identity Sp in the profile binding certificate to the previously authenticated server identity Sa . The formal goal says that, if the eUICC accepts the two certificates together, Sp must be an authorized server, and the server must have intended the two certificates to be used together in this session. The injective correspondence indicates freshness of the binding.

$$\begin{aligned} event(U2(U, Sa, Sp, I_t)) \Rightarrow \\ sevent(AUTHORIZE(Sp)) \ \&\& \ inj-event(U1(U, Sa, I_t, _)) \ \&\& \ inj-event(S1(U, Sa, Sp, I_t, _, _)) \quad (\text{Auth C}) \end{aligned}$$

In the protocol, the profile binding is implemented so that the eUICC compares the server OID between the profile binding certificate in message 9 and the previously-authenticated server certificate. The freshness depends on rather intricate reasoning. Messages 8–9 do not contain the client nonce N_U or any other fresh value that originates from the client. Instead, the client reasons that the server always chooses a fresh I_t in the mutual authentication step (messages 4–5), which then proves the freshness of messages 8–9 to the client. This kind of indirect reasoning complicates the security analysis and formal verification. Nevertheless, ProVerif is able to verify Auth C without being given any hint about the reasoning.

The process of formalizing the security goals often leads us to question the protocol design details and suggests potential optimizations. While RSP includes the fresh profile binding step (messages 8–9) in each session, an alternative would be to view the ability to provision SIM profiles as a persistent property of the server. By issuing the profile binding certificate $Cert_{Sp}$ with the same server as the server certificate $Cert_{Sa}$, GSMA attests that the server is authorized to provision SIM profiles as long as both certificates remain valid. Seen this way, the profile binding messages do not need to be signed or associated with a specific session. Thus, messages 8–9 could be omitted, and the profile binding certificate could be attached to one of the other messages.

5.3 Authenticated key exchange and profile download

The eUICC and SM-DP+ server perform an ephemeral ECDH key exchange in messages 10–13. To authenticate the key exchange, they sign the messages with the certificates that were previously exchanged. Note that the server has now switched into using the profile binding certificate $Cert_{Sp}$ and the identity Sp in it. The following formal security goals state that, if either side accepts the key exchange results, both sides must have started the key exchange.

$$\begin{aligned} event(S2(U, Sa, Sp, I_t, _, P, mnold, I_{ac})) \Rightarrow \\ inj-event(S1(U, Sa, Sp, I_t, mnold, I_{ac})) \ \&\& \ inj-event(U2(U, Sa, Sp, I_t)) \quad (\text{Auth D}) \end{aligned}$$

$$\begin{aligned} event(U3(U, Sa, Sp, I_t, k, _, mnold, I_{ac})) \Rightarrow \\ inj-event(U2(U, Sa, Sp, I_t)) \ \&\& \ inj-event(S2(U, Sa, Sp, I_t, k, _, mnold, I_{ac})) \quad (\text{Auth E}) \end{aligned}$$

The goal of the key exchange is to create a fresh session key for the profile download. The key exchange neither has nor needs many of the properties that a more generic key exchange protocol would have. As is apparent from Auth D, there is no key confirmation [27] for the server (k is not a parameter in the events S2 and U2). Auth E, on the other hand, requires key confirmation for the client. In the protocol, it is implemented by computing two message

authentication codes for messages 12-13 with the related session key k and k' . Because k and k' are closely related, it is sufficient to mention k in the formalized security goals.

The secrecy of the session key is expressed with the following two queries.

$$attacker(k) \ \&\& \ event(S2(U, Sa, _, _, k, _, mnold, _)) \Rightarrow false \quad (\text{Sec W})$$

$$attacker(k) \ \&\& \ event(U3(U, Sa, _, _, k, _, mnold, _)) \Rightarrow false \quad (\text{Sec X})$$

Sec W and Sec X state that the adversary must not know a session key that is accepted by the server or eUICC. Critically, this secrecy property must hold even when the eUICC is in an adversary-owned device, i.e., when the LPA and user are untrusted.

In messages 12–13, the eUICC finally receives the SIM profile P and the associated metadata $mnold$. The formal security goal below refines Auth E by binding the profile P to the events.

$$\begin{aligned} event(U3(U, Sa, Sp, I_t, k, P, mnold, I_{ac})) \Rightarrow \\ inj-event(U2(U, Sa, Sp, I_t)) \ \&\& \ inj-event(S2(U, Sa, Sp, I_t, k, P, mnold, I_{ac})) \end{aligned} \quad (\text{Auth F})$$

After the events S2 and U3, the protocol has reached its main objective of SIM profile download. At this point, the mobile user can start accessing the mobile network services. Clearly, we should look at the transitive correspondences from S2 and U3 to the initial events U0 and S0 and to the preceding the INTENT and ORDER. Before doing that, we will briefly discuss the final messages of the protocol.

5.4 Profile install notification

The client acknowledges the successful download and installation of the SIM profile by sending a profile install notification to the server (messages 14-15). The formal security goal below states that the server should only accept the notification if it previously sent the profile to the eUICC and if the eUICC has accepted the profile.

$$\begin{aligned} event(S3(U, Sa, Sp, I_t, P, _, mnold)) \Rightarrow & event(S2(U, Sa, Sp, I_t, _, P, mnold, _)) \\ & \&\& \ event(U3(U, Sa, Sp, I_t, _, P, mnold, _)) \\ & \&\& \ event(OWNER(userId, U)) \\ & \&\& \ event(INTENT(userId, mnold, U, _)) \\ & \&\& \ event(ORDER(userId, mnold, _, [U], _, _)) \end{aligned} \quad (\text{Auth G})$$

The RSP specification states that “Seq protects against replay attacks” [35]. This can be verified with an injective version of Auth G. The non-injective version included above highlight the observation that the replay protection is unnecessary. The eUICC includes *Seq* in the notification it sends after successfully installing, enabling, disabling, or deleting a profile. The replay protection is needed when enabling and disabling profiles, so that the server knows the current profile state. Installing and deleting, on the other hand, can occur only once for each profile, and replays of the success notification would not cause any confusion.

The notification response (message 16) contains only the *OK* status. The LPA then tells the eUICC to delete the notification from its message queue. These messages are not critical for the security of RSP, and they have no cryptographic end-to-end protection. We omit the formalization but note that the security depends entirely on the TLS tunnel and on the integrity of the LPA.

5.5 Security goals for the full protocol

After the stepwise security goals, it is now time to look at the goals for the RSP protocol as a whole. To summarize the outcome of the common handshake and profile download (messages 1–13), we verify an injective correspondence between the eUICC accepting the profile in event U3 and the initial events U0 and S0.

$$event(U3(U, Sa, _, I_t, k, P, mnold, _)) \Rightarrow inj-event(U0(U, _)) \ \&\& \ inj-event(S0(Sa, I_t, _, mnold, _)) \quad (\text{Auth I})$$

This formula states that, if the eUICC accepts a profile, it must have started a matching common handshake. It can be verified with any subset of the MNO identifier *mnold*, server domain name *S* (in the default-server approach), and activation code *I_{ac}* (in the activation-code approach) as parameters in the events. We included only *mnold* above because it must match the expected value while the other parameters no longer have significance after the protocol has completed. We will verify Auth I directly with ProVerif. Interestingly, the same result can be obtained as the transitive closure of the stepwise formulas (Auth F, Auth D, and Auth B).

Above, we have focused on the common handshake and profile download phases. The following two goals summarize the overall objectives of RSP also including the profile ordering and download initialization phases. The events S2 and U3 respectively indicate the server sending the profile and the eUICC accepting it. The correspondences are from these events all the way back to initial user intent and profile order.

$$\begin{aligned} event(U3(U, Sa, _, I_t, _, P, mnold, I_{ac})) &\Rightarrow event(OWNER(userId, U)) \\ &\&\& event(INTENT(userId, mnold, U, I_{ac})) \\ &\&\& inj-event(ORDER(userId, mnold, _, [U], P, I_{ac})) \end{aligned} \quad (\text{Auth J})$$

$$\begin{aligned} event(S2(U, Sa, _, I_t, _, P, mnold, I_{ac})) &\Rightarrow event(OWNER(userId, U)) \\ &\&\& event(INTENT(userId, mnold, U, I_{ac})) \\ &\&\& inj-event(ORDER(userId, mnold, _, [U], P, I_{ac})) \end{aligned} \quad (\text{Auth K})$$

Auth K builds on the earlier goal Auth B', and we will find that these two goals fail in the same situations.

The ultimate secret that requires protection in RSP is the SIM profile *P*. This is because the profile includes the secret shared key that will be used for authentication between the mobile device and the MNO. The following formal security goals state that a profile which has been sent by the server or accepted by the eUICC must not be known to the adversary.

$$attacker(P) \ \&\& \ event(S2(U, Sa, _, _, k, P, mnold, I_{ac})) \Rightarrow false \quad (\text{Sec Y})$$

$$attacker(P) \ \&\& \ event(U3(U, Sa, _, _, k, P, mnold, I_{ac})) \Rightarrow false \quad (\text{Sec Z})$$

6 VERIFICATION RESULTS

This section describes the results from verifying the security goals formalized in Section 5 against the ProVerif protocol model described in Section 4. The 15 authentication and secrecy goals were each verified against the 40 models explained in Section 4.4, resulting in 570 verification targets. We will mostly discuss the authentication goals that failed in the partial compromise scenarios. While some of the failures are expected in the given scenarios, others provide new insights into the protocol and lead to recommendations for both implementations and for future versions of the standard.

Tables 4–5 summarize the verification results for the two profile ordering and download initialization approaches explained in Section 3.3: the default-server approach and the activation-code approach. Each row in the tables corresponds to one of the compromise scenarios defined in Section 4.3. The columns correspond to the security goals

Table 4. Results for the default-server approach

| Partial compromise scenario | Authentication goals | | | | | | | | | | | Secrecy goals | | | |
|-----------------------------|----------------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| | A | B | B' | C | D | E | F | G | I | J | K | W | X | Y | Z |
| 1: — | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 2: server | X ² | X ^c | ✓ | X ² | X ^c | X ² | X ² | (✓) | X ² | X ² | ✓ | ✓ | X ² | ✓ | X ² |
| 3: eUICC | ✓ | X ⁴ | ✓ | O ^d | X ⁴ | ✓ | ✓ | X ⁴ | ✓ | ✓ | ✓ | X ⁴ | ✓ | X ⁴ | (✓) |
| 4: LPA | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 5: 2nd server | O ³ | O ^c | ✓ | O ³ | O ^c | O ³ | O ³ | ✓ | O ³ | O ³ | ✓ | ✓ | O ³ | ✓ | O ³ |
| 6: 2nd eUICC | ✓ | ✓ | ✓ | O ^d | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 7: 2nd MNO | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8: order as user | ✓ | ✓ | X ⁷ | ✓ | ✓ | ✓ | ✓ | X ⁷ | ✓ | ✓ | X ⁷ | ✓ | ✓ | ✓ | ✓ |
| 9: order for eUICC | ✓ | ✓ | X ^a | ✓ | ✓ | ✓ | ✓ | X ^a | ✓ | X ^a | X ^a | ✓ | ✓ | ✓ | ✓ |

Attacker owns some eUICCs in all the scenarios 1–9. Client-side goals are gray. No security is expected in Scenarios 2-3.

Table 5. Results for the activation-code approach

| Partial compromise scenario | Authentication goals | | | | | | | | | | | Secrecy goals | | | |
|-----------------------------|----------------------|----------------|------------------|----------------|----------------|----------------|----------------|--------------------|----------------|----------------|------------------|----------------|----------------|----------------|----------------|
| | A | B | B' | C | D | E | F | G | I | J | K | W | X | Y | Z |
| 1: — | ✓ | ✓ | O ¹ | ✓ | ✓ | ✓ | ✓ | O ¹ | ✓ | ✓ | O ¹ | ✓ | ✓ | ✓ | ✓ |
| 2: server | X ² | X ^c | X ^{1,f} | X ² | X ^c | X ² | X ² | X ^{1,f} | X ² | X ² | X ^{1,f} | ✓ | X ² | ✓ | X ² |
| 3: eUICC | ✓ | X ⁴ | X ^{1,6} | O ^d | X ⁴ | O ^e | O ^e | X ^{1,4,6} | O ^e | O ^e | X ^{1,6} | X ⁴ | ✓ | X ⁴ | (✓) |
| 4: LPA | ✓ | ✓ | X ^{1,9} | ✓ | ✓ | (✓) | (✓) | X ^{1,9} | ✓ | X ⁹ | X ^{1,9} | ✓ | ✓ | ✓ | ✓ |
| 5: 2nd server | O ³ | O ^c | O ¹ | O ³ | O ^c | O ³ | O ³ | O ¹ | O ³ | O ³ | O ¹ | ✓ | O ³ | ✓ | O ³ |
| 6: 2nd eUICC | ✓ | O ⁵ | O ¹ | O ^d | O ⁵ | ✓ | ✓ | O ^{1,5} | ✓ | ✓ | O ¹ | O ⁵ | ✓ | O ⁵ | (✓) |
| 7: 2nd MNO | ✓ | ✓ | O ¹ | ✓ | ✓ | ✓ | ✓ | O ¹ | ✓ | ✓ | O ¹ | ✓ | ✓ | ✓ | ✓ |
| 8: order as user | ✓ | ✓ | X ^{1,7} | ✓ | ✓ | ✓ | ✓ | X ^{1,7} | ✓ | ✓ | X ^{1,7} | ✓ | ✓ | ✓ | ✓ |
| 10: code leaks | ✓ | ✓ | X ^{1,8} | ✓ | ✓ | ✓ | ✓ | X ^{1,8} | ✓ | ✓ | X ^{1,8} | ✓ | ✓ | ✓ | ✓ |
| 11: code spoofed | ✓ | ✓ | X ^{1,b} | ✓ | ✓ | ✓ | ✓ | X ^{1,b} | ✓ | X ^b | X ^{1,b} | ✓ | ✓ | ✓ | ✓ |

Attacker owns some eUICCs in all the scenarios 1–11. Client-side goals are gray. No security is expected in Scenarios 2-3.

Table 6. Summary of major vulnerabilities in RSP

| Vulnerability | Attack references | Recommendations |
|--|-------------------|-------------------------------|
| Security of RSP depends partly on the TLS tunnel. Compromise of TLS causes leak of the activation code and general failure in scenarios 5 and 6. | 1,3,5,c,d,e | R9 (incl. R1, R2, R3, R7, R8) |
| The client (LPA and eUICC) may lack a-priori knowledge of the server OID. Communicating the OID out-of-band is optional in the activation-code approaches and not supported in the default-server approach. | 3 | R1,R2,R7 |
| The SM-DP+ server may lack a-priori knowledge of the eUICC identifier U . In that case, the security depends entirely on the secrecy of the activation code, which is prone to leaking and injection. | 1,5,8,9,b,f | R3 |
| Misbinding vulnerabilities are created because several signed messages lack identity information on which the client and server should agree. Messages 4–7 should include the server OID. Messages 8–9 should include the eUICC identifier U . | c,d | R7,R8 |
| No reliable method for verifying user intent, including both the mobile subscriber identity and eUICC ownership. Many MNOs do not authenticate the user or verify the user's possession of the eUICC before ordering a SIM profile. Thus, the adversary could order a profile on behalf of another user or make an unauthorized order for their eUICC. | 7,8,a,b | R4,R5,R6 |

from Section 5. The columns for client-side goals are on gray background, while server-side goals are on white. The successfully verified security goals are marked with \checkmark or O, and the failed security goals are marked with X. O denotes goals that fail if TLS is disabled between the LPA and server. (\checkmark) denotes cases where ProVerif terminates without a conclusive result (see Section 7.3). Red is used for highlighting the interesting failures.

As expected, *all the security goals were successfully verified for the case where all the participants are honest and only the network is untrusted*, i.e., in scenario 1. A few goals in scenario 1 depend on TLS (¹ in results), and we will return to them in Section 6.6. Otherwise, the rest of the discussion is about the partial compromise scenarios 2–11.

6.1 Failures of the SM-DP+ server authentication

As can be expected, there is a catastrophic security failure in scenario 2, in which the server private keys are compromised. In the attack, the adversary plays the role of the server towards the eUICC, and the victim server does not need to be involved at all. This violates all the client-side correspondence properties (² in results). The profile secrecy goal (Sec Z) also fails because the client accepts a fake profile created by the adversary.

The first somewhat surprising result is that the security of server authentication (Auth A) in scenario 5 depends on the TLS tunnel. Recall that the adversary in scenario 5 has compromised one SM-DP+ server, but not the one chosen by the MNO for the victim user. If TLS is not used, the adversary can redirect connections from the client to the compromised server. The consequence is that the client-side security properties fail (³ in results), exactly as in scenario 2 (² in results). The client will even accept a fake profile from the compromised server. The relevant rows in Tables 4–5 show the similarity between scenarios 2 and 5 when TLS is not used. The failure in scenario 5 is more serious, however, because *an adversary that has compromised an SM-DP+ server anywhere in the world can issue fake SIM profiles to any subscriber of any MNO* – even if they have no business relation with the compromised server.

The root cause for the problem in scenario 5 is that, as we observed in Section 5.1, the client typically lacks a-priori knowledge of the server OID. This allows the adversary to sign message 4–5 with any GSMA-issued server certificate. The RSP protocol has the option of delivering the server OID to the client together with the activation code, so that the LPA can check it after receiving message 4. Unfortunately, there is currently no similar option for the default-server approach, where the eUICC stores only the server domain name. We make the following recommendations:

- R1** *In the activation-code approach, always provide the server OID to the LPA during the profile download initialization.*
- R2** *Amend the RSP specification for the default-server approach so that, in addition to the default server domain name S , either the LPA or the eUICC stores the default server OID and compares it with the value in the server certificate $Cert_{Sa}$ during the server authentication.*

The choice between implementing **R2** in the LPA or in the eUICC depends on whether there is a secure path for configuring the correct server OID value into the eUICC. If the value can be configured securely in the device supply chain (together with the server domain name S), the eUICC should perform the check. On the other hand, if the correct server OID is input into the eUICC via the LPA, we might just as well trust the LPA to make the comparison with $Cert_{Sa}$.

6.2 Failures of client authentication and profile secrecy

Auth B is a technical client authentication goal that covers only the authentication step. As we might expect, the stepwise goal fails in scenario 3, where the eUICC’s private key is compromised (⁴ in results). This is because the eUICC authentication relies on the certificate and signature in message 6-7. The adversary in scenario 3 has the private key,

and it can sign a spoofed message 7. In the activation-code approach, the adversary needs a valid activation code for the spoofed message; it can obtain one by ordering a profile for itself.

In scenario 6, the security of Auth B depends on TLS, but only in the activation-code approach (⁵ in results). Recall that, in scenario 6, the adversary has compromised the private key of one eUICC, e.g., by reverse-engineering its own mobile device. Without TLS, the adversary can capture the activation code I_{ac} from an honest user. It then uses the captured code in a spoofed message 7, which it signs with the compromised eUICC's private key. This attack is possible in the activation-code approach because the server does not know the correct eUICC identifier U and accepts any eUICC certificate $Cert_U$ in message 7.

While these failures of the client authentication (Auth B) are rather technical, they have serious consequences later in the protocol. Failure of the initial client Auth B leads to the failure of the client authentication in the key exchange (Auth D), which in turn breaks the secrecy of the session key (Sec W) and secrecy of the profile (Sec Y). In other words, the Diffie-Hellman session key is compromised and, therefore, *the SIM profile leaks to the adversary*.

In order to avoid the failure in scenario 6, we make the following recommendation:

R3 *In the activation-code approach, always register the eUICC identifier U to the server during the profile ordering.*

6.3 Identity fraud and leaked activation code

The high-level client-authentication goal Auth B' compares the authentication results with the initial user intent and the order received by the server. The most obvious failure occurs again in scenario 3, where the client private key has been compromised (⁶ in results). The other failures of Auth B', on the other hand, are quite different from the stepwise goal Auth B.

The other failures of Auth B' are relatively easy to understand if we start the discussion from scenario 8, which models identity fraud in the profile ordering. In the attack (⁷ in results), the adversary spoofs the victim user's identity when requesting a profile from the MNO. The adversary then downloads the profile to an adversary-owned eUICC. There is a slight technical difference between the default-server and activation-code approaches: in the former, the adversary makes the order explicitly for the adversary-owned eUICC, while in the latter, the adversary receives an activation code and uses it to download the profile into the adversary-owned eUICC. The end result of the attack is the same in both approaches.

Identity fraud in the profile ordering is not the only way in which the adversary can obtain an activation code that is associated with a victim user's account at the MNO. Scenario 10 models explicitly the leaks that may happen due to careless handling of the activation code by the victim user, MNO employees, or other parts of the supply chain. The consequent security failures (⁸ in results) are exactly the same as in the identity fraud case (⁷ in results). Furthermore, in scenario 4, where the LPA software is compromised, the LPA can leak the code because delivering the activation code to the eUICC is one of the main features of the LPA. Again, the security failures are the same (⁹ in results).

The practical consequences of the leaked activation code are serious, and we will highlight them here. With the activation code, the adversary can download the honest user's SIM profile into the adversary's mobile device. The adversary's mobile network access will then be charged to the victim user's account. Even more seriously, the adversary may be able to hijack the victim's mobile telephone number, answer and make calls at that number, and impersonate the victim in applications that use the telephone number for authentication. This type of attack is called *SIM swapping* [7]. The general observation here is that identity fraud in the profile ordering phase can enable *identity fraud* in other contexts where the mobile device is used for authentication. Moreover, the attack is fairly realistic. When the order is

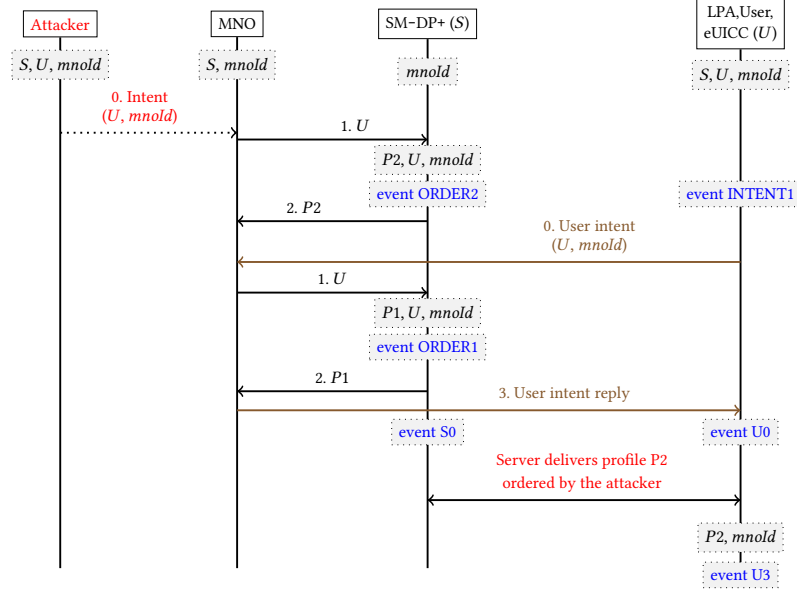


Fig. 6. The adversary orders a second profile for the target eUICC. When the user initiates download, the server may deliver the wrong profile to the eUICC. The attack works against the default-server approach to profile download initialization.

made offline in a shop or over the phone, the primary objective of the MNO staff is to sell subscriptions and not to act as security officers. When the order is made online or with the LPA software, the strength of the user authentication depends on the supported identity verification methods, which depend on the MNO practices and on national regulation.

We summarize the need for due diligence as a recommendation:

R4 *When user requests a profile for their eUICC, the MNO should verify the user identity carefully. In the default-server approach, it should do this before ordering the profile from the SM-DP+ server, and in the activation-code approach, before giving the activation code to the user.*

The RSP specification does not explicitly state the above requirement because it is an MNO function and, thus, outside the technical scope of the specification. Nevertheless, it is essential for the security of the protocol. Additionally, *the user, MNO, and the entire supply chain should treat the activation code as a critical secret*. This secrecy requirement for the activation code is already stated in the RSP specification.

6.4 Unauthorized order for an eUICC

When a malicious user contacts the MNO to request a SIM profile for an eUICC, the user identity is not the only thing they can misrepresent. In the default-server approach, they may also lie about their eUICC identifier U . Scenario 9 models explicitly situations where the adversary requests a profile with its own user identity but for someone else's eUICC identifier. The client authentication goal (Auth B') fails in this scenario (^a in results), and so does the dependent goal for the full protocol (Auth K).

The attack for the default-server approach is shown in Figure 6. The victim user orders a profile for their own eUICC. Meanwhile, the attacker has ordered a profile from the same MNO for its own user identity but for the victim's eUICC. There are now two profiles for the same eUICC in the same server, and it is unclear which profile the victim will

download. Depending on the timing of the requests and on the server implementation, the victim may end up installing the profile ordered by the attacker. The attack is possible if (i) the adversary can request a profile from an MNO without proving its possession of the eUICC, and (ii) the adversary knows the eUICC identifier U of the target user. It seems likely that these conditions will be met in practical situations.

The practical consequences of the attack are quite troubling. The installed profile is associated with the adversary's mobile subscription, and thus the adversary gains some control over the victim's mobile access. It may be able to redirect the user's calls and text messages. It may also be able to access sensitive information about the victim's mobile access including call logs or location. Moreover, we should consider an extended scenario 9 where the user is careless about checking the *mnoId* during the profile download (after message 12 in Figure 4). In that case, the attacker can order its profile from a different MNO that uses the same server. This second MNO could be a compromised one that leaks session keys to enable call interception, or it could pay the adversary for bringing in new subscribers, albeit unwilling ones. These attacks violate some of the most fundamental security requirements of mobile networks: call confidentiality and billing integrity.

In the activation-code approach, similar failures happen in scenario 11, where the adversary can inject an activation code to the victim user (b^6 in results). Moreover, the compromised LPA in scenario 4 could not only leak an activation code (see Section 6.3) but also inject one (9 in results). Again, the practical consequence is that the user may install a SIM profile that is associated with someone else's — usually the adversary's — mobile subscription.

To prevent the attacks, the MNO should verify that the user who requests the profile actually owns the eUICC with the specific identifier. Unfortunately, this verification is not easy to implement. In our formal model, the user is both the person who holds the mobile device containing the eUICC, and the mobile subscriber who pays the bills. In the real world, on the other hand, they can be two different entities. The intent to obtain a new SIM profile should, ideally, be authorized by both the subscriber and the person with the mobile device. There is currently no support for such dual authorization in RSP, and adding it would require major extensions to the protocol concepts, user interface, and messages.

While there is no strong solution, it is possible to mitigate the attacks. As the most important mitigation, *users have to be careful not to use activation codes from untrusted sources*. Obviously, they should also not install LPA software from untrusted sources. Additionally, we recommend the following changes to the RSP specification:

- R5** *Amend the RSP specification with design guidelines for a uniform user-experience on how the user confirms the MNO before the profile download or installation.*
- R6** *Amend the RSP specification for the default-server approach so that, if there are multiple profiles on the SM-DP+ server, the user is warned and asked to choose the one to download.*

The RSP specification already supports the inclusion of the MNO icon and name in the profile metadata. The LPA or the UE may display the profile metadata to the user for confirmation. However, there is currently no guidance on how the information should be displayed, what the user should compare, or what the MNO icons and names look like.

From Tables 4–5, we observe that scenarios 8, 9, 10 and 11 as well as scenario 4 in the activation-code approach have nearly the same failures. The explanation is that the server-side authentication goals require a correspondence between initial user intent and the server's later session state for several parameters including the MNO, server, and eUICC identifiers. The attacks in the different scenarios cause a mismatch of different parameters, but the end result is always a mismatch between the user intent and the later server state, causing failure of the server-side goals Auth B' and Auth K.

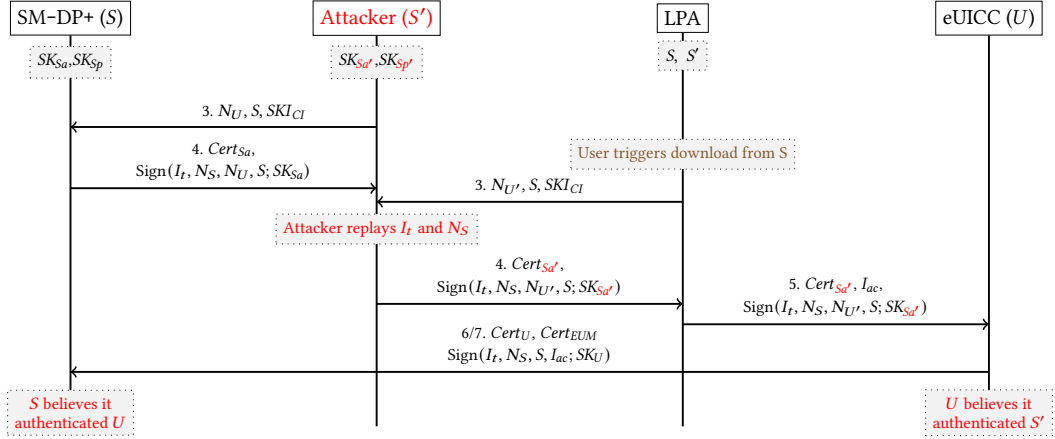


Fig. 7. Misbinding attack in scenario 5 when TLS is not used. Adversary has compromised a second server S' . The eUICC and honest server end up with unmatched beliefs.

6.5 Misbinding without TLS

In scenario 2, as well as in scenario 5 when TLS is not used, there is an additional failure of the stepwise client authentication (Auth B) due to a so-called *misbinding* attack [14, 38, 52] (^c in results). In the attack, the eUICC is willing to perform mutual authentication with the authorized SM-DP+ servers S and S' . The latter server is compromised, i.e., the adversary controls its private keys. The details of the misbinding attack are shown in Figure 7. The result is that the eUICC thinks the session identified by I_t is with S' while S thinks the session is with the eUICC. This violates the correspondence property Auth B. Note that this is not normal server impersonation or man in the middle; in a sense, the adversary gets the honest server S to impersonate the compromised server S' .

The misbinding attack causes inconsistent states between the legitimate protocol participants. The practical consequence could be confusion among the users and MNO service staff. In this case, the failure of Auth B leads to the further failure of Auth D. The consequences are mitigated by the fact that misbinding attacks, in general, do not cause leakage of secrets.

To prevent the misbinding, the protocol messages should be more explicit about the server identity. We recommend the following:

- R7** Amend the RSP specification so that the signed messages in the initial authentication (messages 4–5 and 6–7) include the server OID, which the message recipient compares with the expected value. This requires the adoption of recommendations **R1** and **R2**.

There is also a misbinding vulnerability in the reverse direction. Without the TLS tunnel, Auth C fails in scenarios 3 and 6 (^d in results), in which the adversary has compromised the private key of at least one eUICC. We first explain the attack in the activation-code approach, as shown in Figure 8. The adversary intercepts message 7 from the honest eUICC U . It replaces the signature and certificate on message 7 with those of a compromised eUICC U' and forwards the modified message to the server. When the server receives message 7, it believes the identity information in the eUICC certificate. The server responds with message 8, which only contains the session identifier I_t , from which U cannot tell that the message was intended for U' . The result is that U thinks the session identified by I_t is with S , while S thinks

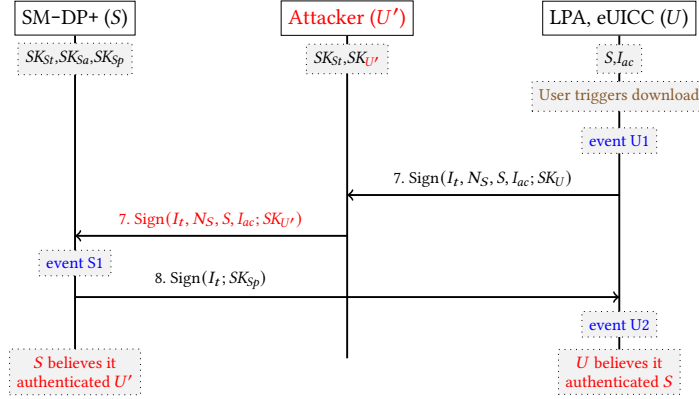


Fig. 8. Misbinding attack in scenario 6 when TLS is not used. The adversary replaces the eUICC signature on message 7 with one by a compromised second eUICC. The eUICC and server end up with unmatched beliefs. Similar attacks also present in scenario 3

the session is with the U' , which violates the correspondence property Auth C. In the default-server approach, the adversary will first prepare for the attack by ordering a profile for the compromised eUICC U' to the same server S .

The consequences of the second misbinding are limited because the attack will be detected during the following key exchange, where the server checks that message 11 is signed by the same eUICC as the earlier message 7. Nevertheless, we recommend increasing the robustness of the profile binding step:

- R8** Amend the RSP specification so that the server includes the eUICC identifier U from the initial authentication (message 7) in the signed profile-binding (message 8–9), and that the eUICC checks the identifier upon receipt of the message.

6.6 Dependence on the TLS tunnel

We have already seen some failures that occur if the TLS tunnel is disabled (^{3,5,c,d} in results). Let us now return to the pure network adversary (scenario 1). In the default-server approach, the security goals do not depend on the TLS tunnel. In the activation-code approach, on the other hand, *TLS is needed to protect the activation code*. If TLS is not used, the activation code leaks, which results in the previously mentioned failures (¹ in results) with the same consequences as in scenario 10 (⁸ in results). Naturally, any failure of scenario 1 occurs in all the other scenarios as well; this can be seen in Table 5 columns B', G and K where the security always fails without TLS. The root causes of these failures are (i) that the server does not initially know the eUICC identity U and relies on the secret activation code to authorize the eUICC, and (ii) that without TLS, the adversary can capture the activation code and use it to download the profile to a different eUICC.

In addition to stealing the activation code, the adversary can replace the code with the wrong one. In scenario 3, where the adversary has the private key of the honest user's eUICC, this causes some additional failures of client-side security properties (^e in results). In the attack, the adversary intercepts message 7, replaces the activation code I_{ac} in it, signs the modified message, and forwards it to the server.

It is important to realize that the model without the TLS tunnel is also a model for situations where the security of the TLS tunnel is compromised. As explained in Section 4.3, the TLS server and its private key are relatively vulnerable to hacking. The effect of the compromised tunnel can be seen in scenarios 2 and 4 as well, where the server and client

end of the tunnel are compromised, respectively. The failures in these scenarios with TLS (^{f,9} in results) are similar to the failures in scenario 1 without TLS (¹ in results).

RSP is an unusual cryptographic protocol in that its security depends partly on the encapsulation of the messages into a TLS tunnel. Looking at the symbols O in Tables 4-5, we see that quite a few vulnerabilities arise in the absence of the TLS tunnel between the LPA and SM-DP+ server. The protocol would be more robust if it was designed to be independent of the TLS transport. We recommend the following:

R9 *Redesign the RSP protocol so that it does not depend on the TLS channel for any critical security requirements. This can be achieved by implementing recommendations R1, R2, R3, R7 and R8.*

More precisely, in the default-server approach, implementing recommendations **R2**, **R7** and **R8** removes the dependence on TLS. In other words, these recommendations change the verification results for the default-server approach so that there is no difference between using or not using TLS (i.e., O becomes ✓ in Table 4).

In the activation-code approach, implementing recommendations **R1**, **R3**, **R7** and **R8** similarly removes the dependence on TLS, with the exception of scenario 3. The reader might not expect such a big improvement in the activation-code approach because TLS protects the confidentiality of the activation code. The explanation is that recommendation **R3** removes the client-authentication role of the activation code and thus the need to keep it secret; instead, the client authentication will depend on the pre-registered eUICC identifier U at the server and on the eUICC certificate $Cert_U$ in messages 6-7. (In scenario 3, **R3** is ineffective due to the compromised eUICC private key. This should have no impact on the protocol design decisions because scenario 3 is irreparably insecure due to the other attacks covered in Section 6.2.)

Notwithstanding the argument made above, the TLS provides an additional layer of encryption that protects the user privacy. Without the tunnel, RSP would leak the eUICC identifier and certificate to the communication network. TLS fulfills the requirement for confidentiality protection that was quoted in Section 5.1. Therefore, we suggest keeping the tunnel but relegating it to only a privacy protection layer.

6.7 Failures of install notification

The install notification takes place after the activation of the SIM profile, and any security failures at this stage are not catastrophic. If the server accepts a spoofed notification, it and the MNO may wrongly believe that the profile was installed but not in use. The result could be a discrepancy between the MNO's business information systems and the mobile network state. It is unlikely that the user's access to the network would be affected. We summarize the attacks for completeness.

Authentication of the profile install notification (Auth G) can fail for several different reasons. First, it depends on the TLS tunnel. Second, attacks against the client authentication goals Auth B or B' can be continued against Auth G (^{1,4,5,6,7,8,9,a,b,f} in results). The only exception is that the misbinding attack against Auth B (^c in results) does not propagate to Auth G because the profile install notification message 15, unlike message 7 (Figure 4), contains the server OID of the intended recipient.

7 DISCUSSION

This section provides a summary of main vulnerabilities identified in our research. It also discusses the importance of partial compromise scenarios, the limitations of the modeling and verification methods, and the possibility of simplifying the RSP protocol.

7.1 Summary of the main vulnerabilities

We will now discuss the significance the most important vulnerabilities discovered in Section 6. They are also summarized in Table 6.

Dependence on TLS. Our most important observation is that the security of RSP depends on the TLS tunnel from the mobile device to the SM-DP+ server, and that this dependence is mostly unnecessary. In Section 6.6, we recommend a clear path to removing this dependence. In the default-server approach, there is hardly any reason not to implement the recommended changes (**R2**, **R7** and **R8**). In the activation-code approach, the recommended changes (**R1**, **R3**, **R7** and **R8**) would similarly make RSP independent of TLS, but the decision to implement these changes is not as straightforward as in the default-server approach. The convenience of activation codes depends on the fact that they are distributed to potential customers without knowing in advance who will install the profile and on which device. This advantage would be lost with **R3**, which requires pre-registration of the eUICC identifier for each activation code. An alternative way to remove the dependence on TLS is to encrypt the activation code (message 7 in Figure 4) on the RSP level. This is possible but will require restructuring of the cryptographic protocol, so that the Diffie-Hellman key exchange and server authentication take place before the client sends the activation code to the server.

No a-priori knowledge of identifiers. The next two vulnerabilities in Table 6 are related to the lack of a-priori knowledge about the peer identity. There is some overlap with the TLS dependence discussed above, but the design consideration is nevertheless different. The client and server in RSP may not initially know each other's identifiers. When that is the case, they have no way of knowing whether they are talking to the intended entity. The authentication with certificates does assure the server that it is talking with some eUICC, and it does assure the client that it is talking with some SM-DP+ server in the world. The weakness is that they cannot differentiate between the intended entity and other entities of the same class that may be operated by the adversary.

A general solution to this problem is pre-established identities, which can then be authenticated with the certificates. The server OID certainly should be pre-distributed to the LPA or eUICC (recommendations **R1** and **R2**). As already discussed above, pre-registering the eUICC identifier (**R3**) is not as obvious because it would require major changes to the business processes in the activation-code approach. Without **R3**, the entire path of the activation code in RSP must protect its confidentiality. This path includes the SM-DP+ server (^f in results), the MNO sales and support staff or online channels and the mobile user (^{8,b} in results), the LPA (⁹ in results), the eUICC (⁵ in results), as well as the TLS tunnel from LPA back to the server (¹ in results) – or redesigning the protocol to protect the confidentiality without TLS, as already suggested above.

Misbinding vulnerability. The next vulnerability in Table 6 is a simple protocol design flaw that enables the misbinding attacks. The solution is to include the server and eUICC identities in the messages (recommendations **R7** and **R8**).

Difficulty of verifying user intent. The final vulnerability in Table 6 is more fundamental. RSP relies on the user deciding which profile they want to install. For the protocol designer, this creates multiple difficult problems which were discussed in Sections 6.3 and 6.4.

The first problem is that the MNO needs to verify the identity and authorization of the user who requests a profile from it. The communication with the user may take place offline, or online without strong authentication. Moreover, the MNO and its employees are incentivized to make the profile ordering process as easy as possible, and they may prefer getting new business to performing identity checks.

A further problem is that the user intent should be authorized both by the mobile subscriber and by the person holding the mobile device with the eUICC. These are not always the same person or entity. The difficulty of verifying both the subscriber and eUICC owner intent reliably may be one of the reasons why the default-server approach is currently not in wide use. *An advantage of the activation-code approach is that the MNO gives the code to the subscriber and it is entered into the LPA by the person holding the mobile device, which implicitly ensures the authorization from both user roles.*

Finally, the users themselves could make mistakes, such as leaking the activation code or accepting the wrong code. There is a step in the protocol where the user should check the MNO identifier before the profile download. This gives the user another opportunity to stop a download that does not match the user's intent, but a typical mobile user may not be able to check the MNO identifier reliably.

7.2 Do failures in the partial compromise scenarios matter

In scenarios 2-3, the private keys of one of the protocol endpoints are compromised. The serious failures of server authentication (⁶ in results) as well as client authentication and profile secrecy (^{2,4} in results) are quite expected, and it would be unrealistic to defend against such strong attackers.

In scenario 3, we can compare RSP with the analogous attacks against the old, removable SIM cards. In scenario 3, the adversary has compromised the physical integrity or supply chain of the target eUICC and extracted its private key. As a consequence, the adversary is able to capture the SIM profile including the shared secret in the profile. This is comparable to the adversary compromising the integrity of an old-fashioned SIM card or its supply chain and extracting the pre-shared secret in it. Thus, we can argue that RSP does not create a new vulnerability in scenario 3.

The other partial compromise scenarios 4-10, on the other hand, are more realistic, and the security failures in them should be taken seriously.

Scenario 5 assumes that there is one compromised SM-DP+ server somewhere in the world, which seems quite possible. This leads to failure of the server authentication (^{3,c} in results). Scenario 6, where the adversary has compromised the private keys of one or more eUICCs somewhere in the world, is almost guaranteed to occur. The resulting attacks break not only the client authentication but also the secrecy of the authentic SIM profile (^{5,d} in results). The security failures in these scenarios are prevented by running RSP inside the TLS tunnel, but as we have argued, that is a relatively weak protection, and the protocol should not depend on TLS for security.

In scenario 6, we can again compare RSP with removable SIM cards. In scenario 6, *the adversary has extracted the private key of one eUICC anywhere in the world, and this enables it to capture the SIM profile of any target eUICC with a network attack.* There is no similar weakness in the old removable SIM cards. The attack is possible against MNOs that distribute activation codes without associating the code with a specific eUICC, which is a common practice.

One historical concern in mobile networks has been SIM cloning [30, 53, 58]. Modern removable SIM cards are resistant to the published cloning methods [53]. The closest equivalent attack against RSP is the leaked profile in scenario 3 or 6 (^{4,5} in results). The attacker could then continue by emulating an eUICC with the stolen profile or by injecting the profile into another eUICC in scenario 2 or 5 (^{2,3} in results).

Scenarios 8-10 seem quite likely to occur in practice because they model mistakes by the human participants including the mobile user and MNO employees. Comparing with the removable SIM card, the attacks (^{7,8,a,b} in results) are similar to stealing the victim's SIM card and replacing the victim's SIM with the wrong one. Even when the SIM profile is successfully protected by the secure hardware module, which can be either the eUICC or the removable SIM card, the module may end up in the wrong user's mobile device. The new issue in RSP is that the SIM profile is a digital rather

than physical object, which means that the current methods for identifying the object and authorizing actions need to be replaced with digital ones.

7.3 Limitations of the modeling and verification

A formal protocol model is necessarily an abstraction of the full specification and leaves out many details. It is essential to identify the protocol participants and the important messages between them.

The RSP protocol has various participants: the eUICC, LPA, user, MNO, and server. On one hand, we can model these entities as separate processes and can consider separately the compromise of each entity and each communication channel between them. On the other hand, by combining multiple such entities into one process, the model becomes easier to understand and more manageable for the verification tools, but some subtle attacks might go undetected. We have modeled the LPA and user as one combined process.

The protocol messages in the model must be similarly optimized. The first challenge was to identify the critical messages in RSP and their parameters from the information scattered all over the specification. Even then, it was necessary to optimize the model so that only the essential messages are included. As one major optimization, we do not model the negotiation of the cryptographic algorithms (and thus might miss downgrade attacks [13]) or error handling.

Sometimes, the specification does not fully define the participant behavior, and we had to make assumptions about the implementation. For example, the LPA software acts as an intermediary between the server and eUICC. As shown in Figure 4, it checks the value of S in message 4 and $mnold$ in message 12 before passing them on to the eUICC. The RSP specification does not say whether LPA should perform any other sanity checks on the messages, such as verifying the signatures and comparing other message fields with their expected values. We modeled both a relaxed LPA that checks only these two values and otherwise passes through any messages without looking at their contents, and a strict LPA that checks all the values it can. The verification results did not indicate any difference in the security of the two LPA models, but the strict model produced more verification results.

The TLS channel between the LPA and the server was particularly difficult to model accurately. First, only the server is authenticated while the LPA is anonymous. Second, we might expect the tunnel to protect the integrity of the session inside, but since the communication is implemented as an HTTP API, TLS will actually protect only the individual request-response pairs. Any continuity of state between the requests has to be implemented by the application-layer protocol, in this case RSP. We experimented with different models of the TLS tunnel. Fortunately, we did not find cases where subtle differences in the TLS model would impact the verification results.

Furthermore, the RSP protocol has alternative modes and optional features that cause variation to the message flows and content. We solve this problem by modeling a large number of protocol variants (the default-server and activation-code approaches and the optional features relevant to the recommendations). Additional model variants were created by the partial compromise scenarios. To maintain consistency between the model variants, we generate all the models from one combined source. A theoretical limitation that arises from this way of modeling the protocol variants is that we will not detect potential adverse interactions between different protocol variants.

In addition to the protocol participants and messages, the security goals were also formalized. The queries were optimized iteratively, on one hand, to detect any attacks, and on the other hand, to verify the security properties that can be verified. The interesting queries are often ones that fail in one scenario and succeed in another. One difficulty is that the first security failure found by the verification tool may be a trivial one. In that case, it is necessary to experiment with queries that ignore the first discovered attack, so that more serious or more profound failures can be uncovered. The queries shown in this paper have many possible variations, e.g., on which parameters to include in the injective

correspondences. The queries included in Section 5 are the ones that best demonstrate the analysis results of Section 6; many further variants were evaluated during our research work.

In general, the modeling process involves iterative inclusion and exclusion of details from the protocol model and from security goals as well as optimization of their representation.

Nevertheless, there were some security goals that ProVerif was unable to verify or disprove conclusively. These are denoted with (\checkmark) in Tables 4–5. In these cases, the verification returns a tentative attack trace where the honest eUICC and adversary receive the same SIM profile. The attack is not really possible because the server never resends an already downloaded profile. The inconclusive result is due to ProVerif’s limited time model which cannot easily express such a rule. Nevertheless, the difficulty in the verification points to a potential weakness of the protocol. In RSP, the profile is prepared at the profile order time and downloaded later, and the two actions are not strongly linked to each other. If the server generated the profile at the download time, we could be certain that every downloaded profile is unique. After experimentally making this change to the model, ProVerif was able to verify the previously inconclusive goals. Generating the profile (and keys within it) at download time is not a practical solution, however, and we do not recommend it to implementors.

7.4 SM-DS assisted profile ordering and download initialization

As mentioned in Section 3.3, the RSP specification includes a third, *SM-DS assisted* approach to the profile ordering and download initialization. We do not model this approach because it is has not been deployed as an open system. Rather, individual vendors have deployed proprietary variants of this approach.

The SM-DS assisted approach relies on an additional hierarchical entity, the subscriber manager discovery servers (SM-DS), to assist the mobile device in finding the profiles that are waiting for it in the SM-DP+ servers. The LPA receives one or more *event identifiers* from the SM-DS servers. The rest of the protocol is similar to the activation-code approach with a pre-registered eUICC identifier (recommendation **R3**). If the SM-DS and SM-DP+ server are closely coupled as a proprietary solution, the security would correspond to this combination. In a more open model, we would have to model also the partial compromise scenario in which some of the SM-DS servers are compromised.

7.5 Simplifying the RSP protocol

The common handshake and profile download, as seen in Figure 4, comprise more credentials, messages and roundtrips than a typical authentication protocol.

The multiple different server certificates (see Table 1) in the consumer RSP protocol have their origin in the machine-to-machine RSP specification [33]. In M2M RSP, two technically and commercially separate entities, the subscription manager secure routing (SM-SR) and the subscription manager data preparation (SM-DP), perform the entity authentication and session key agreement, respectively. In contrast, consumer RSP combines these roles to a single commercial entity, the SM-DP+. Thus, the separate key pairs and steps for entity authentication and profile binding are not strictly necessary, and the profile binding would not need to be a separate step in the protocol.

The key agreement and authenticated encryption (messages 10-13) in the RSP protocols borrow from the GlobalPlatform defined SCP11a [29] specification. This explains why they are a separate phase rather than merged with the entity authentication.

Thus, the consumer RSP protocol could be simplified. For example, we could combine the profile binding (messages 8-9) with the server authentication (messages 4-5) and the client key-exchange messages (10-11) with the client authentication (messages 6-7). The dependence on TLS could be removed completely by encrypting the handshake

messages similar to TLS 1.3 [49]. However, such major changes to the protocol design will require a thorough consultation with the stakeholders in the standards process. We believe that the recommendations made in this paper provide a much shorter path to increasing the robustness of RSP security.

8 CONCLUSION

We have modelled and analyzed the GSMA consumer remote SIM provisioning protocol that allows remote download of SIM profiles to eUICCs. Our work includes constructing a formal model of the RSP protocol based on its specifications, identifying assumptions and implicit security goals in the specification, formalizing the security goals, and defining realistic partial compromise scenarios where the protocol should be tested. We performed verification of 15 selected security goals with the ProVerif tool against two protocol variants in 11 scenarios, all with and without a TLS tunnel. In addition to presenting the verification results and describing potential attacks against the protocol, we discussed the root causes of the failures and made practical recommendations for improving the robustness of the RSP protocol specification and its implementations. The results show the power of formal modeling and verification in clarifying the goals of security protocols and in understanding the assumptions made in their design.

REFERENCES

- [1] 3GPP. 2018. TS 33.501 Security architecture and procedures for 5G System (Release 15). http://www.3gpp.org/ftp/Specs/archive/33_series/33.501/33501-f40.zip.
- [2] 3GPP. 2021. TS 33.401 technical specification Group services and system aspects 3GPP system architecture evolution (SAE) Security architecture (Release 17). https://www.3gpp.org/ftp/Specs/archive/33_series/33.401/33401-h00.zip.
- [3] 3GPP. 2022. TS 33.102 3G security; Security architecture (Release 17). https://www.3gpp.org/ftp/Specs/archive/33_series/33.102/33102-h00.zip.
- [4] Martín Abadi, Bruno Blanchet, and Cédric Fournet. 2018. The applied pi calculus: Mobile values, new names, and secure communication. *Journal of the ACM (JACM)* 65, 1 (2018), 1.
- [5] Martín Abadi and Roger Needham. 1996. Prudent engineering practice for cryptographic protocols. *IEEE transactions on Software Engineering* 22, 1 (1996), 6–15.
- [6] J Arkko, V Lehtovirta, and P Eronen. 2006. *Extensible Authentication Protocol method for 3rd generation authentication and key agreement (EAP-AKA)*. RFC 4187. IETF.
- [7] Georgia Bafoutsou, Slawomir Bryska, and Marnix Dekker. 2021. ENISA Countering SIM-Swapping. <https://www.enisa.europa.eu/publications/countering-sim-swapping>.
- [8] David Basin, Cas Cremers, and Catherine Meadows. 2018. Model checking security protocols. In *Handbook of Model Checking*. Springer, 727–762.
- [9] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. 2018. A formal analysis of 5G authentication. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM Press, New York, NY, 1383–1396.
- [10] David Basin, Ralf Sasse, and Jorge Toro-Pozo. 2021. Card brand mixup attack: Bypassing the PIN in non-Visa cards by using them for Visa transactions. In *30th USENIX Security Symposium (USENIX Security 21)*. 179–194.
- [11] David Basin, Ralf Sasse, and Jorge Toro-Pozo. 2021. The EMV standard: Break, fix, verify. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1766–1781.
- [12] Karthikeyan Bhargavan, Bruno Blanchet, and Nadim Kobeissi. 2017. Verified models and reference implementations for the TLS 1.3 standard candidate. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Washington, DC, USA, 483–502.
- [13] Karthikeyan Bhargavan, Christina Brzuska, Cédric Fournet, Matthew Green, Markulf Kohlweiss, and Santiago Zanella-Béguelin. 2016. Downgrade resilience in key-exchange protocols. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Washington, DC, USA, 506–525.
- [14] Simon Blake-Wilson and Alfred Menezes. 1999. Unknown key-share attacks on the station-to-station (STS) protocol. In *International Workshop on Public Key Cryptography*. Springer-Verlag, Berlin, Germany, 154–170.
- [15] Bruno Blanchet. 2007. Computationally sound mechanized proofs of correspondence assertions. In *20th IEEE Computer Security Foundations Symposium (CSF'07)*. IEEE Computer Society, Washington, DC, USA, 97–111.
- [16] Bruno Blanchet, Ben Smyth, Vincent Cheval, and Marc Sylvestre. 2021. ProVerif automatic cryptographic protocol verifier user manual and tutorial, version 2.04.
- [17] Michael Burrows, Martin Abadi, and Roger Needham. 1990. A logic of authentication. *ACM Transactions on Computer Systems (TOCS)* 8, 1 (1990), 18–36.
- [18] Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. 2018. DEEPSEC: Deciding equivalence properties in security protocols theory and practice. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Washington, DC, USA, 529–546.

- [19] David Cooper, Stefan Santesson, S Farrell, Sharon Boeyen, Russell Housley, and W Polk. 2008. *Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile*. RFC 5280. IETF.
- [20] Cas Cremers. 2011. Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2. In *European Symposium on Research in Computer Security*. Springer, 315–334.
- [21] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. 2017. A comprehensive symbolic analysis of TLS 1.3. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM Press, New York, NY, 1773–1788.
- [22] Cas Cremers, Marko Horvat, Sam Scott, and Thyla van der Merwe. 2016. Automated analysis and verification of TLS 1.3: 0-RTT, resumption and delayed authentication. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Washington, DC, USA, 470–485.
- [23] Dorothy E Denning and Giovanni Maria Sacco. 1981. Timestamps in key distribution protocols. *Commun. ACM* 24, 8 (1981), 533–536.
- [24] Zhonglin Ding, Yang Hu, Wei Luo, Zhongming Huang, Jintao Xue, and Zhongyuan Qin. 2021. Formal analysis and verification of embedded SIM session key agreement protocol. In *Proceedings of the 2021 5th International Conference on Electronic Information Technology and Computer Engineering*. ACM Press, New York, NY, 882–888.
- [25] Danny Dolev and Andrew Yao. 1983. On the security of public key protocols. *IEEE Transactions on information theory* 29, 2 (1983), 198–208.
- [26] Federal Office For Information Security. 2018. Technical Guideline BSI TR-03111, Elliptic Curve Cryptography, version 2.10. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03111/BSI-TR-03111_V-2-1.pdf.pdf.
- [27] Marc Fischlin, Felix Günther, Benedikt Schmidt, and Bogdan Warinschi. 2016. Key confirmation in key exchange: A formal treatment and implications for TLS 1.3. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Washington, DC, USA, 452–469.
- [28] GlobalPlatform. 2018. Card specifications, version 2.3.1. https://globalplatform.org/wp-content/uploads/2018/05/GPC_CardSpecification_v2.3.1_PublicRelease_CC.pdf.
- [29] GlobalPlatform. 2021. Secure Channel Protocol 11 Card Specification, Amendment F version 1.3. <https://globalplatform.org/specs-library/secure-channel-protocol-11-amendment-f/>.
- [30] Ian Goldberg. 1999. GSM cloning. <http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html>. Accessed at: 09-01-2019.
- [31] GSMA. 2018. eSIM WhitePaper. <https://www.gsma.com/esim/wp-content/uploads/2018/12/esim-whitepaper.pdf>.
- [32] GSMA. 2020. SGP.01 Embedded SIM remote provisioning architecture, version 4.1. <https://www.gsma.com/esim/wp-content/uploads/2020/06/SGP.01-v4.1.pdf>.
- [33] GSMA. 2020. SGP.02 Remote provisioning architecture for embedded UICC, version 4.2. <https://www.gsma.com/esim/wp-content/uploads/2020/07/SGP.02-v4.2.pdf>.
- [34] GSMA. 2021. SGP.21 Architecture Specification, version 2.3. <https://www.gsma.com/esim/wp-content/uploads/2021/07/SGP.21-2.3.pdf>.
- [35] GSMA. 2021. SGP.22 Technical Specification, version 2.3. <https://www.gsma.com/esim/wp-content/uploads/2021/07/SGP.22-v2.3.pdf>.
- [36] GSMA Intelligence. 2015. Understanding SIM evolution. <https://www.gsmainelligence.com/research/?file=81d866ecda8b80aa4642e06b877ec265>.
- [37] Gerard J. Holzmann. 1997. The model checker SPIN. *IEEE Transactions on software engineering* 23, 5 (1997), 279–295.
- [38] Hugo Krawczyk. 2003. SIGMA: The ‘SIGn-and-MAC’ approach to authenticated Diffie-Hellman and its use in the IKE protocols. In *Annual International Cryptology Conference*. Springer, 400–425.
- [39] Dong Ling and Kefei Chen. 2012. *Cryptographic protocol: security analysis based on trusted freshness*. Higher Education Press, Beijing.
- [40] Gavin Lowe. 1996. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*. Springer-Verlag, Heidelberg, Germany, 147–166.
- [41] Wenbo Mao. 2003. *Modern cryptography: theory and practice*. Pearson Education India.
- [42] Catherine A Meadows. 1994. Formal verification of cryptographic protocols: A survey. In *International Conference on the Theory and Application of Cryptology*. Springer, 133–150.
- [43] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. 2013. The TAMARIN prover for the symbolic analysis of security protocols. In *International Conference on Computer Aided Verification*. Springer-Verlag, Berlin, Germany, 696–701.
- [44] Maxime Meyer, Elizabeth A Quaglia, and Benjamin Smyth. 2018. Attacks against GSMA’s M2M remote provisioning. In *FC’18: 22nd International Conference on Financial Cryptography and Data Security*. Springer-Verlag, Heidelberg, Germany.
- [45] Roger M Needham and Michael D Schroeder. 1978. Using encryption for authentication in large networks of computers. *Commun. ACM* 21, 12 (1978), 993–999.
- [46] Tristan Ninet, Axel Legay, Romaric Maillard, Louis-Marie Traonouez, and Olivier Zendra. 2019. Model checking the IKEv2 protocol using SPIN. In *2019 17th International Conference on Privacy, Security and Trust (PST)*. IEEE Computer Society, Washington, DC, USA, 1–7.
- [47] Ofcom. 2012. Reprogrammable SIMs: Technology, Evolution and Implications, Final Report. https://www.ofcom.org.uk/__data/assets/pdf_file/0024/44754/reprogrammable-sims.pdf.
- [48] Jaemin Park, Kiyoungh Baek, and Cheoloh Kang. 2013. Secure profile provisioning architecture for embedded UICC. In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*. IEEE Computer Society, Washington, DC, USA, 297–303.
- [49] Eric Rescorla. 2018. *The transport layer security (TLS) protocol version 1.3*. RFC 8446. IETF.
- [50] J. Scahill and J. Begley. 2015. The great SIM heist. <https://firstlook.org/theintercept/2015/02/19/great-sim-heist/>.
- [51] Benedikt Schmidt, Simon Meier, Cas Cremers, and David Basin. 2012. Automated analysis of Diffie-Hellman protocols and advanced security properties. In *2012 IEEE 25th Computer Security Foundations Symposium*. IEEE, 78–94.

- [52] Mohit Sethi, Aleksi Peltonen, and Tuomas Aura. 2019. Misbinding Attacks on Secure Device Pairing and Bootstrapping. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. ACM Press, New York, NY, 453–464.
- [53] Jaspreet Singh, Ron Ruhl, and Dale Lindskog. 2013. GSM OTA SIM cloning attack and cloning resistance in EAP-SIM and USIM. In *2013 International Conference on Social Computing*. IEEE, 1005–1010.
- [54] Elaheh Vahidian. 2013. *Evolution of the SIM to eSIM*. Master's thesis. Institutt for Telematikk.
- [55] Alexandr Vesselkov, Heikki Hämmäinen, and Pertti Ikalainen. 2015. Value networks of embedded SIM-based remote subscription management. In *2015 Conference of Telecommunication, Media and Internet Techno-Economics (CTTE)*. IEEE Computer Society, Washington, DC, USA, 1–7.
- [56] Thomas YC Woo and Simon S Lam. 1992. Authentication for distributed systems. *Computer* 25, 1 (1992), 39–52. IEEE.
- [57] Thomas YC Woo and Simon S Lam. 1997. Authentication for distributed systems. In *Internet besieged: countering cyberspace scofflaws*. ACM Press/Addison-Wesley Publishing Co., New York, NY.
- [58] Yuanyuan Zhou, Yu Yu, François-Xavier Standaert, and Jean-Jacques Quisquater. 2013. On the need of physical security for small embedded devices: a case study with COMP128-1 implementations in SIM cards. In *International Conference on Financial Cryptography and Data Security*. Springer, 230–238.