Università degli Studi di Roma Tor Vergata

DIPARTIMENTO DI INGEGNERIA CIVILE E INGEGNERIA DELL'INFORMAZIONE
Corso di Laurea Magistrale in Ingegneria Informatica

Sicurezza delle eSIM: analisi e sperimentazione mediante sviluppo di user agent e server SM-DP+

Candidato:
Matteo Fanfarillo
Matricola 0316179

Relatore:
Prof. Giuseppe Bianchi

Correlatori:
Prof. Francesco Gringoli
Dott. Lorenzo Valeriani

Anno Accademico 2023/2024

"I'll walk through fire to save my life".

Abstract

L'argomento principale della presente trattazione è la sicurezza dell'eSIM, una tecnologia entrata in vigore negli ultimi anni che sta diventando sempre di più un tema di attualità. Il lavoro proposto consiste nell'analizzare il protocollo definito per questa tecnologia e le relative implementazioni, in modo tale da individuare e trattare eventuali vulnerabilità, ipotizzare degli scenari di attacco e, se possibile, definire dei meccanismi di difesa.
All'interno del presente documento, anzitutto verrà introdotto il problema e verrà svolta una trattazione dettagliata sull'architettura dell'eSIM e sul protocollo di comunicazione tra eUICC, LPA e server SM-DP+, con lo scopo di fornire al lettore gli strumenti necessari per comprendere appieno le tematiche centrali del lavoro. Seguirà una breve trattazione sullo stato dell'arte per chiarire quali sono gli aspetti di sicurezza delle eSIM che sono stati già trattati. Verrà poi illustrato l'intero procedimento che ha portato all'implementazione dei simulatori delle entità (eUICC + LPA da una parte, server SM-DP+ dall'altra), che fungono da strumenti di base per effettuare le analisi e le considerazioni successive. Verranno mostrati in particolar modo i risultati ottenuti dall'analisi della sicurezza dell'eSIM a boot-time (i.e. durante la fase di configurazione). Infine, verranno tratte delle conclusioni sul lavoro svolto e verrà fornita una panoramica sui possibili progetti futuri che potranno essere intrapresi a partire dai risultati ottenuti attraverso questo lavoro.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Panoramica sull'eSIM

L'eSIM (embedded-SIM) non è altro che una SIM virtuale: grazie a lei, quando l'utente vuole cambiare operatore, non deve più acquistare fisicamente una nuova SIM card presso un negozio del nuovo operatore, bensì gli è sufficiente ricevere via e-mail un profilo, ossia una "SIM digitale" che può essere caricata subito sul telefono mediante la scansione di un QR code. Si tratta di una soluzione molto più pratica rispetto a recarsi fisicamente presso il negozio dell'operatore, tant'è vero che negli ultimi anni si sta diffondendo sempre di più: uno studio di Juniper Research del 2023 stima che il numero di telefoni che utilizzano la connettività eSIM aumenterà dai 986 milioni del 2023 ai 3.5 miliardi entro il 2027 [1]. Per questi motivi, e poiché le informazioni associate alla comunicazione tra eSIM sono sensibili, è fondamentale garantire un livello di sicurezza sufficientemente elevato per il funzionamento dell'eSIM sia a run-time che a boot-time.

1.2 Obiettivo del lavoro

La presente trattazione si propone di:

1. introdurre una panoramica sul mondo dell'eSIM, descrivendo tutti i dettagli architetturali e protocollari necessari;
2. definire il punto a cui sono arrivate le ricerche precedenti, in modo tale da riprendere il lavoro a partire da quel punto ed evitare così di ripetere le analisi e gli esperimenti che sono stati già resi di dominio pubblico;
3. effettuare un'analisi di sicurezza e delle vulnerabilità dell'eSIM e del suo funzionamento, dopo aver orchestrato un ambiente di test dal quale avviare gli esperimenti;
4. se possibile, tentare di sfruttare, anche con delle attività di laboratorio, le eventuali vulnerabilità trovate.

1.3 Definizioni preliminari

- **eUICC (embedded Universal Integrated Circuit Card)**: è un chip utilizzato nei telefoni all'interno del quale è embeddato il software dell'eSIM. È integrato direttamente nei dispositivi (i.e. non è removibile) ed è progettato per essere programmato a distanza. Può contenere uno o più profili eSIM.
- **LPA (Local Profile Assistant)**: è un'applicazione che vive nel telefono dell'utente ed è responsabile della gestione dei profili all'interno della rete mobile, inclusi la creazione, l'aggiornamento e la cancellazione. Essa funge anche da intermediario in qualunque comunicazione tra l'eUICC e il server.

- **SM-DP+ (Subscription Manager Data Preparation plus)**: è un protocollo che rappresenta una tecnica di provisioning usata per configurare le eSIM in modo automatico e remoto. Rispetto alla versione base SM-DP, offre delle funzionalità aggiuntive come un sistema di crittografia più avanzato e un'architettura di rete più flessibile. La parte server del protocollo è appunto detta **server SM-DP+**.

Capitolo 2

Interfacce e funzionamento dell'eSIM

2.1 Architettura di RSP

Per comprendere appieno come funziona e come si interfaccia l'eSIM all'interno dei dispositivi mobili, è necessario introdurre il protocollo **RSP**, anche perché l'eSIM si colloca proprio all'interno dell'architettura di RSP.
RSP (Remote SIM Provisioning) è un protocollo utilizzato dal protocollo SM-DP+ per gestire la comunicazione tra il server SM-DP+ e la scheda eSIM del dispositivo mobile (i.e. l'eUICC). In particolare, definisce le operazioni di provisioning specifiche per la comunicazione dell'eUICC. Quest'ultimo comprende i dati sia dell'operatore che dell'utente che, nel caso delle SIM tradizionali, verrebbero memorizzati su una SIM card fisica. L'end user che vuole ottenere un profilo eSIM offerto da un particolare operatore (nel quale viene definito un piano tariffario) deve pagare l'operatore affinché esso gli fornisca un codice QR. Dopodiché, deve effettuare la scansione del codice QR per avviare lo scaricamento (operazione di Download) e l'installazione (operazione di Install) del profilo eSIM: a questo punto, la connessione tra end user (col relativo profilo eSIM) e operatore è completata. Se in un secondo momento l'end user ha la necessità di ottenere un secondo profilo eSIM, gli è sufficiente ripetere i medesimi passaggi appena descritti, e questo secondo profilo può essere installato all'interno del medesimo eUICC che ospita già il primo profilo. Tale meccanismo è illustrato nella figura 2.1 tratta da [2].
Per quanto riguarda l'architettura interna di RSP nello specifico, esistono due soluzioni diverse [3].

1. **LPA embeddato nel dispositivo mobile ma non all'interno dell'eUICC (LPAd)**: oltre alla comunicazione tra l'applicazione LPA e SM-DP+, si utilizzano delle apposite interfacce anche per la comunicazione tra l'eUICC e l'applicazione LPA, come mostrato nella figura 2.2 tratta da [3]. Di seguito è riportato un breve glossario che chiarisce il significato di alcuni componenti appartenenti all'architettura di RSP raffigurata in 2.2.

    - **CI** = Certificate Issuer: nota anche come eSIM CA RootCA, è un'entità autorizzata a rilasciare certificati digitali.
    - **Device App** = una qualunque applicazione installata nel dispositivo mobile.
    - **Enterprise** = impresa (i.e. azienda, organizzazione o entità governativa) che si iscrive ai servizi mobili che devono essere utilizzati dai dipendenti a supporto dell'impresa stessa.
    - **EUM** = eUICC Manufacturer: è il fornitore delle eUICC e del software residente (e.g. firmware, sistema operativo); svolge anche il ruolo di certificate authority subordinata al CI e rilascia certificati all'eUICC [3][4].
    - **HRI Server** = server che fornisce le High Resolution Icon, che sono icone che vengono create per essere visualizzate in alta risoluzione.
    - **LDSd** = Local Discovery Service (quando l'LPA non è nell'eUICC).
    - **LPDd** = Local Profile Download (quando l'LPA non è nell'eUICC).
    - **LUId** = Local User Interface (quando l'LPA non è nell'eUICC).
    - **SM-DS** = Subscription Manager Discovery Server: è il componente che consente a SM-DP+ di raggiungere l'eUICC senza dover sapere a quale rete il dispositivo è connesso.

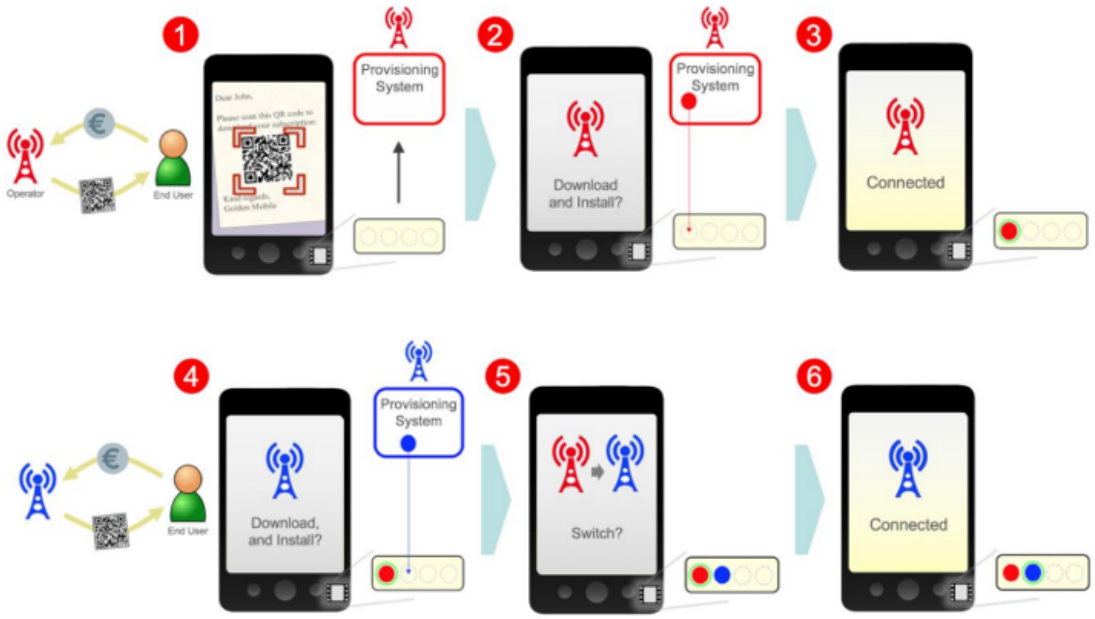


Figura 2.1: Comunicazione tra end user e operatore nel contesto del Remote SIM Provisioning.

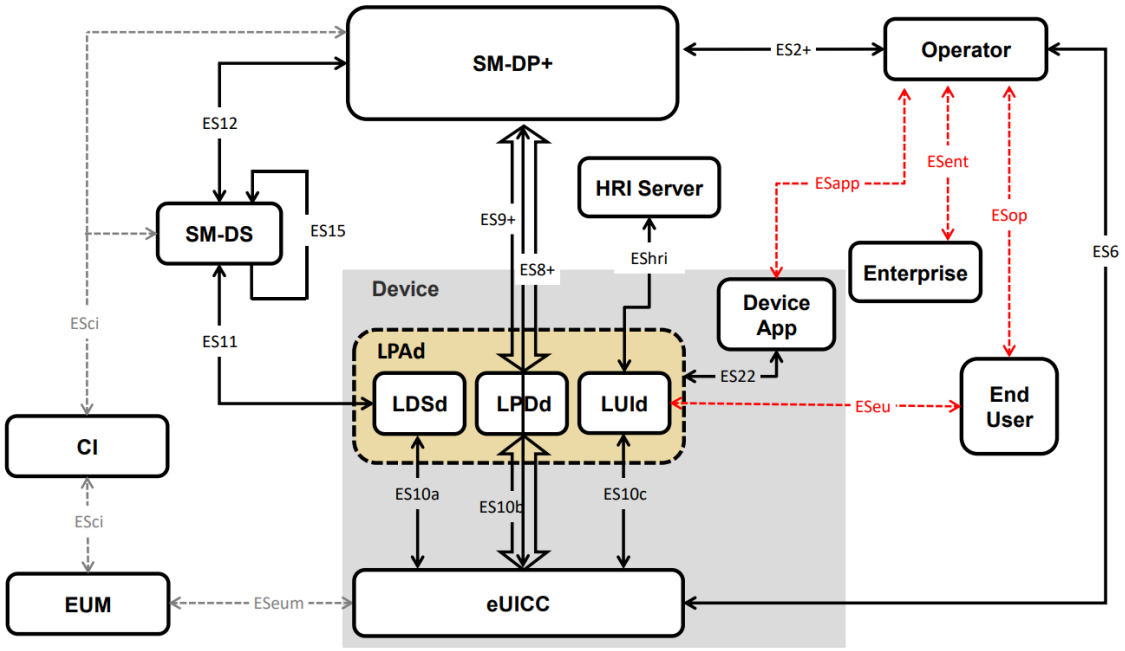


Figura 2.2: Architettura di RSP nel caso di LPA non embeddato nell'eUICC.

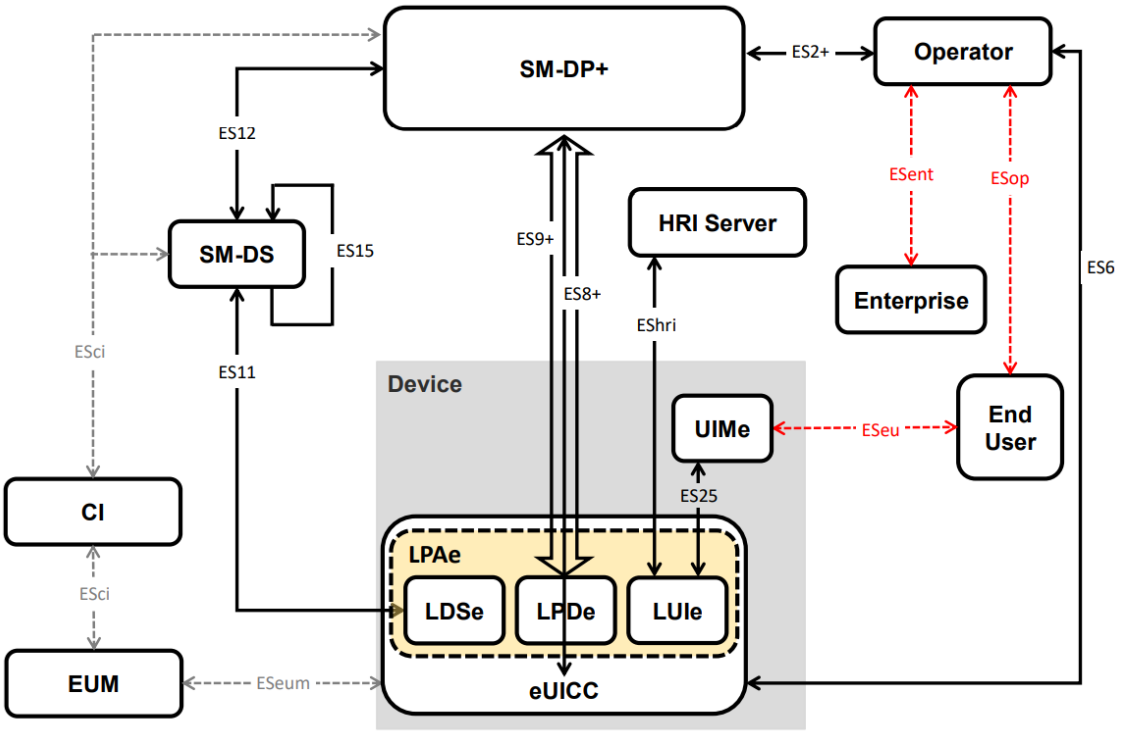


Figura 2.3: Architettura di RSP nel caso di LPA embeddato nell'eUICC.

2. **LPA embeddato all'interno dell'eUICC (LPAe)**: sono necessarie solo delle interfacce tra l'eUICC e SM-DP+, come mostrato nella figura 2.3 tratta da [3]. Successivamente è riportato un breve glossario che chiarisce il significato di alcuni componenti appartenenti all'architettura di RSP raffigurata in 2.3.

   - **LDSe** = Local Discovery Service (quando LPA è nell'eUICC).
   - **LPDe** = Local Profile Download (quando LPA è nell'eUICC).
   - **LUIe** = Local User Interface (quando LPA è nell'eUICC).
   - **UIMe** = User Inteface Module.

2.1.1 Interfacce presenti nell'architettura di RSP

Sono illustrate nella tabella 2.1, costruita a partire da informazioni tratte da [3].

**Un Profile Package è un pacchetto di dati associato a un profilo che contiene le informazioni di configurazione necessarie per attivare e utilizzare quel profilo all'interno di una scheda eSIM. Esistono diversi tipi di Profile Package: l'Unprotected Profile Package (UPP) è un pacchetto di dati non protetto da alcun meccanismo di sicurezza; il Protected Profile Package (PPP) è un pacchetto di dati protetto da alcuni meccanismi di sicurezza, come l'autenticazione e/o la crittografia; il Bound Profile Package (BPP) è un pacchetto di dati legato a un particolare dispositivo o a una piattaforma di servizi; il Segmented Bound Profile Package (SBPP), infine, non è altro che un BPP suddiviso in molteplici segmenti che possono essere utilizzati in modo indipendente e separato.*

***ISD-P (Issuer Security Domain Profile) è un contenitore sicuro che ospita un unico profilo.*

****Root SM-DS è il server primario utilizzato da un operatore di rete mobile per gestire le attivazioni e le disattivazioni delle sottoscrizioni e per gestire funzionalità come l'autenticazione e l'autorizzazione degli utenti. D'altra parte, si hanno gli Alternative SM-DS, che sono server di backup a cui si ricorre quando il Root SM-DS non è disponibile.*

Tabella 2.1: Interfacce in RSP.

Interfaccia	Componente 1	Componente 2	Descrizione
ES2+	Operatore	SM-DP+	Viene usata dall'operatore per invocare la preparazione del Profile Package*.
ES6	Operatore	eUICC	Viene usata dall'operatore per gestire il contenuto dei profili.
ES8+	SM-DP+	eUICC	Fornisce un canale end-to-end sicuro tra SM-DP+ e l'eUICC per l'amministrazione dell'ISD-P** e del relativo profilo durante il download e l'installazione.
ES9+	SM-DP+	LPD	Viene usata per fornire trasporto sicuro tra SM-DP+ e LPD per la consegna del Profile Package.
ES10a	LDSd	eUICC	Viene usata da LPAd per ottenere gli indirizzi configurati dall'eUICC per Root SM-DS*** (gestione di una Discovery Request).
ES10b	LPDd	eUICC	Viene usata da LPAd per trasferire un Profile Package all'eUICC.
ES10c	LUId	eUICC	Viene usata da LPAd per la gestione locale dei profili installati sull'eUICC da parte dell'end user (e.g. Enable, Disable, Delete).
ES11	LDS	SM-DS	Viene usata per l'ottenimento di eventi.
ES12	SM-DP+	SM-DS	Viene usata per la gestione degli eventi.
ES15	SM-DS	SM-DS	Viene usata per connettere gli SM-DS tra loro nel caso in cui ce ne sia più di uno.
ES22	LPAd	Device App	Viene usata da un'applicazione del dispositivo mobile per interoperare con l'LPA.
ES25	UIMe	LUIe	Viene usata per trasferire verso l'LPA le interazioni dell'end user.
ESop	Operatore	End user	È specifica per le relazioni di business tra l'operatore e l'end user.
ESeu	End user	LUI	È specifica per le relazioni di business tra l'end user e la LUI.
ESeum	eUICC	EUM	È specifica per le relazioni di business tra l'eUICC e l'EUM.
ESci	CI	SM-DP+, SM-DS, EUM	Viene usata per richiedere certificati.
EShri	LUI	HRI Server	Viene usata per recuperare le High Resolution Icon.
ESent	Operatore	Enterprise	È un'interfaccia che prescinde dagli scopi del presente documento.
ESapp	Operatore	Device App	È un'interfaccia che prescinde dagli scopi del presente documento.

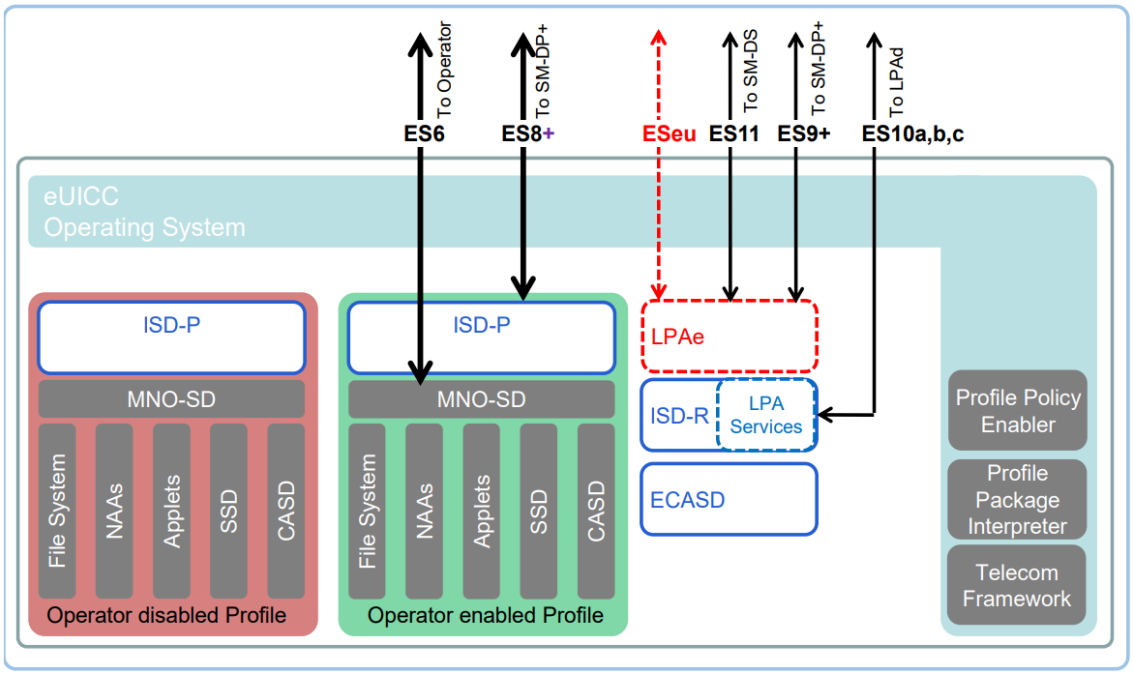


Figura 2.4: Architettura dell'eUICC.

2.2 Architettura dell'eUICC

Nella figura 2.4 tratta da [3] è schematizzata l'architettura interna del chip eUICC, dove i riquadri e le frecce in rosso sono relativi rispettivamente ai componenti e alle interfacce che, nell'ambito dell'eUICC, sono presenti esclusivamente nel caso in cui l'applicazione LPA sia effettivamente embeddata all'interno dell'eUICC (LPAe).
Di seguito, invece, è riportato un breve glossario che chiarisce il significato di alcuni componenti appartenenti all'architettura dell'eUICC raffigurata in 2.4.

- **CASD** = Controller Authority Security Domain: è un'area di storage sicura all'interno dell'ISD-P in cui vengono memorizzate le credenziali richieste (i.e. chiavi, certificati) per supportare le funzionalità di sicurezza sensibili.
- **ECASD** = Embedded Controller Authority Security Domain: è il componente CASD direttamente incapsulato all'interno dell'eUICC.
- **ISD-R** = Issuer Security Domain Root: è il componente responsabile della creazione di nuovi ISD-P e della gestione del loro ciclo di vita.
- **LPA Services** = i seguenti quattro servizi: trasferimento del Bound Profile Package da LPAd all'ISD-P; ottenimento della lista dei profili installati; recupero dell'EID (eUICC ID); ottenimento delle operazioni di gestione del profilo locale (Local Profile Management Operations).
- **MNO-SD** = Mobile Network Operator Security Domain: è la parte del profilo posseduta dall'operatore che fornisce all'operatore Over The Air (OTA) un canale di comunicazione sicuro; viene usato per gestire il contenuto di un profilo una volta che è stato abilitato.
- **NAAs** = Network Access Applications: sono le applicazioni che consentono l'accesso alla rete.
- **Profile Package Interpreter** = servizio del sistema operativo dell'eUICC che traduce i dati del Profile Package in un profilo installato all'interno dell'ISD-P usando il formato interno dell'eUICC.
- **Profile Policy Enabler** = componente che verifica che il profilo eSIM possa essere installato sull'eUICC.

- **SSD** = Supplementary Security Domain: è un'area di memoria protetta all'interno dell'ISD-P che viene utilizzata per l'esecuzione di funzioni di sicurezza come le operazioni crittografiche. Di fatto, il suo scopo principale è quello di proteggere le informazioni riservate dell'utente (i.e. chiavi, password) da accessi non autorizzati e attacchi esterni.
- **Telecom Framework** = servizio del sistema operativo dell'eUICC che fornisce algoritmi di autenticazione di rete standardizzati alle applicazioni NAAs ospitate nei rispettivi ISD-P.

2.2.1 Caratteristiche hardware e software dell'eUICC

1. Deve essere resistente al tampering (manomissione) dei componenti hardware.
2. Contiene un unico ECASD (eUICC Controlling Authority Security Domain).
3. Supporta SHA-1.
4. Supporta Milenage, che è un set di funzioni di autenticazione e di generazione di chiavi.
5. Tutte le funzioni crittografiche devono essere resistenti al tampering e agli attacchi side-channel.

2.3 Chiavi crittografiche e certificati

2.3.1 Chiavi crittografiche

I principali attori che interagiscono nel protocollo RSP sono l'eUICC, l'LPA e il server SM-DP+. Le chiavi utilizzate da loro hanno tutte un nome di tipo <XX>.<YY>.<ZZ> [3], dove:

- <**XX**>: indica la natura della chiave (i.e. chiave pubblica PK, chiave privata SK, chiave pubblica one-time otPK, chiave privata one-time otSK).
- <**YY**>: indica il proprietario della chiave.
- <**ZZ**>: indica l'utilizzo della chiave (i.e. digital signature SIG, key agreement KA, TLS).

Le chiavi di maggiore rilievo, comunque sia, sono riportate nella tabella 2.2 tratta da [5].

2.3.2 Certificati

I certificati propri dei principali componenti che partecipano all'interazione data dal protocollo RSP sono riportati nella tabella 2.3 tratta da [3].

La figura 2.5 tratta da [5] definisce uno schema riassuntivo della struttura originale della catena di certificati definita dalla Public Key Infrastructure (PKI) di RSP.
Attualmente, invece, la struttura della certificate chain è più complessa e introduce molteplici varianti differenti (i.e. presenza o meno di GSMA CI subordinati, presenza o meno di EUM subordinati, presenza o meno di SM-DP+ intermediari, presenza o meno di SM-DS intermediari). Di seguito, per semplicità, più porzioni distinte della catena verranno analizzate separatamente.

- Porzione della catena comprendente il CI, l'EUM e l'eUICC: è riportata nella figura 2.6 tratta da [3] e prevede quattro varianti differenti.
  - **Variante O (originale)**: il CI root rilascia certificati per l'EUM root, il quale rilascia a sua volta certificati per l'eUICC.
  - **Variante A**: il CI root rilascia certificati per l'EUM root; l'EUM root rilascia certificati per l'EUM subordinato; l'EUM subordinato, infine, rilascia certificati per l'eUICC.
  - **Variante B**: il CI root rilascia certificati per il CI subordinato; il CI subordinato rilascia certificati per l'EUM root; l'EUM root, infine, rilascia certificati per l'eUICC.
  - **Variante C**: il CI root rilascia certificati per il CI subordinato; il CI subordinato rilascia certificati per l'EUM root; l'EUM root rilascia certificati per l'EUM subordinato; l'EUM subordinato, infine, rilascia certificati per l'eUICC.

Tabella 2.2: Chiavi crittografiche in RSP.

Nome	Descrizione
PK.EUICC.SIG	Chiave pubblica dell'eUICC usata per verificare le signature dell'eUICC. È inclusa nel certificato CERT.EUICC.SIG.
SK.EUICC.SIG	Chiave privata dell'eUICC usata per generare le signature.
PK.DPauth.SIG	Chiave pubblica del server SM-DP+ usata per verificare le signature del server in fase di autenticazione. È inclusa nel certificato CERT.DPauth.SIG.
SK.DPauth.SIG	Chiave privata del server SM-DP+ usata per generare le signature per autenticarsi all'eUICC.
PK.DPpb.SIG	Chiave pubblica del server SM-DP+ usata per verificare le signature del server comprese nel BPP. È inclusa nel certificato CERT.DPpb.SIG.
SK.DPpb.SIG	Chiave privata del server SM-DP+ usata per generare le signature per il binding dei profili.
PK.DSauth.SIG	Chiave pubblica del server SM-DS usata per verificare le signature di SM-DS in fase di autenticazione. È inclusa nel certificato CERT.DSauth.SIG.
SK.DSauth.SIG	Chiave privata del server SM-DS usata per generare le signature per autenticarsi all'eUICC.
PK.EUM.SIG	Chiave pubblica dell'EUM usata per verificare i certificati degli eUICC. È inclusa nel certificato CERT.EUM.SIG.
SK.EUM.SIG	Chiave privata dell'EUM usata per firmare i certificati degli eUICC.
PK.CI.SIG	Chiave pubblica del CI usata per verificare i certificati dell'EUM, dei server SM-DS e del server SM-DP+.
SK.CI.SIG	Chiave privata del CI usata per firmare i certificati dell'EUM, dei server SM-DS e del server SM-DP+.
otPK.EUICC.KA	Chiave pubblica one-time dell'eUICC usata per il key agreement.
otSK.EUICC.KA	Chiave privata one-time dell'eUICC usata per il key agreement.
otPK.DP.KA	Chiave pubblica one-time del server SM-DP+ usata per il key agreement.
otSK.DP.KA	Chiave privata one-time del server SM-DP+ usata per il key agreement.
PK.DP.TLS	Chiave pubblica del server SM-DP+ usata per verificare le signature TLS del server. È inclusa nel certificato CERT.DP.TLS.
SK.DP.TLS	Chiave privata del server SM-DP+ usata per generare le signature TLS per autenticarsi all'LPA.
PK.DS.TLS	Chiave pubblica del server SM-DS usata per verificare le signature TLS di SM-DS. È inclusa nel certificato CERT.DS.TLS.
SK.DS.TLS	Chiave privata del server SM-DS usata per generare le signature TLS per autenticarsi all'LPA.

- Porzione della catena comprendente il CI e i certificati di tipo SIG di SM-DP+ e SM-DS: è riportata nella figura 2.7 tratta da [3] e prevede quattro varianti differenti.
  - **Variante O (originale)**: il CI root rilascia direttamente i certificati CERT.DPauth.SIG, CERT.DPpb.SIG, CERT.DSauth.SIG.
  - **Variante A**: il CI root rilascia certificati per l'SM-DP+ intermediario e l'SM-DS intermediario; l'SM-DP+ intermediario rilascia a sua volta i certificati CERT.DPauth.SIG, CERT.DPpb.SIG; l'SM-DS intermediario rilascia il certificato CERT.DSauth.SIG.
  - **Variante B**: il CI root rilascia certificati per il CI subordinato, il quale rilascia a sua volta i certificati CERT.DPauth.SIG, CERT.DPpb.SIG, CERT.DSauth.SIG.
  - **Variante C**: il CI root rilascia certificati per il CI subordinato; il CI subordinato rilascia certificati per l'SM-DP+ intermediario e l'SM-DS intermediario; l'SM-DP+ intermediario rilascia i certificati CERT.DPauth.SIG, CERT.DPpb.SIG; l'SM-DS intermediario rilascia il certificato CERT.DSauth.SIG.
- Porzione della catena comprendente il CI e i certificati TLS di SM-DP+ e SM-DS: è riportata nella figura 2.8 tratta da [3] e prevede quattro varianti differenti.

Tabella 2.3: Certificati in RSP.

Nome	Descrizione	Note
CERT.CI.SIG	Certificato GSMA CI	Viene firmato e rilasciato da se stesso.
CERT.CISubCA.SIG	Certificato GSMA CI subordinato	Se esiste, viene firmato e rilasciato dal CI root.
CERT.EUM.SIG	Certificato EUM	Viene firmato e rilasciato dal CI (root o subordinato).
CERT.EUMSubCA.SIG	Certificato EUM subordinato	Se esiste, viene firmato e rilasciato dall'EUM root.
CERT.DPSubCA.SIG	Certificato SM-DP+ intermediario	Viene firmato e rilasciato dal CI (root o subordinato).
CERT.DPauth.SIG	Certificato SM-DP+ per autenticarsi all'eUICC	Viene firmato e rilasciato dal SM-DP+ intermediario o dal CI (root o subordinato).
CERT.DPpb.SIG	Certificato SM-DP+ per rilasciare e firmare i profili eSIM	Viene firmato e rilasciato dal SM-DP+ intermediario o dal CI (root o subordinato).
CERT.DP.TLS	Certificato TLS di SM-DP+	Viene firmato e rilasciato dal SM-DP+ intermediario o dal CI (root o subordinato).
CERT.DSSubCA.SIG	Certificato SM-DS intermediario	Viene firmato e rilasciato dal CI (root o subordinato).
CERT.DSauth.SIG	Certificato SM-DS	Viene firmato e rilasciato dal SM-DS intermediario o dal CI (root o subordinato).
CERT.DS.TLS	Certificato TLS di SM-DS	Viene firmato e rilasciato dal SM-DS intermediario o dal CI (root o subordinato).
CERT.EUICC.SIG	Certificato eUICC	Viene firmato e rilasciato dall'EUM (root o subordinato).
CERT.CA.SIG	Certificato di una qualunque CA pubblica	Può firmare e rilasciare certificati TLS.

  - **Variante O (originale)**: il CI root rilascia direttamente i certificati CERT.DP.TLS, CERT.DS.TLS.
  - **Variante A**: il CI root rilascia i certificati per l'SM-DP+ intermediario e l'SM-DS intermediario; l'SM-DP+ intermediario rilascia il certificato CERT.DP.TLS; l'SM-DS intermediario rilascia il certificato CERT.DS.TLS.
  - **Variante B**: il CI root rilascia certificati per il CI subordinato, il quale rilascia a sua volta i certificati CERT.DP.TLS, CERT.DS.TLS.
  - **Variante C**: il CI root rilascia certificati per il CI subordinato; il CI subordinato rilascia certificati per l'SM-DP+ intermediario e l'SM-DS intermediario; l'SM-DP+ intermediario rilascia il certificato CERT.DP.TLS; l'SM-DS intermediario rilascia il certificato CERT.DS.TLS.

Per quanto riguarda i certificati TLS, al posto del CI, può esserci come root qualunque CA pubblica, come mostrato nella figura 2.9 tratta da [3]. Anche con questa soluzione esistono diverse varianti.

- **Prima variante OO (originale)**: la CA pubblica root rilascia direttamente i certificati CERT.DP.TLS, CERT.DS.TLS.
- **Seconda variante OO**: la CA pubblica root rilascia certificati per una catena di una o più CA pubbliche subordinate, l'ultima delle quali rilascia CERT.DP.TLS, CERT.DS.TLS.
- **Prima variante AA**: la CA pubblica rilascia certificati per l'SM-DP+ intermediario e l'SM-DS intermediario; l'SM-DP+ intermediario rilascia il certificato CERT.DP.TLS; l'SM-DS intermediario rilascia il certificato CERT.DS.TLS.

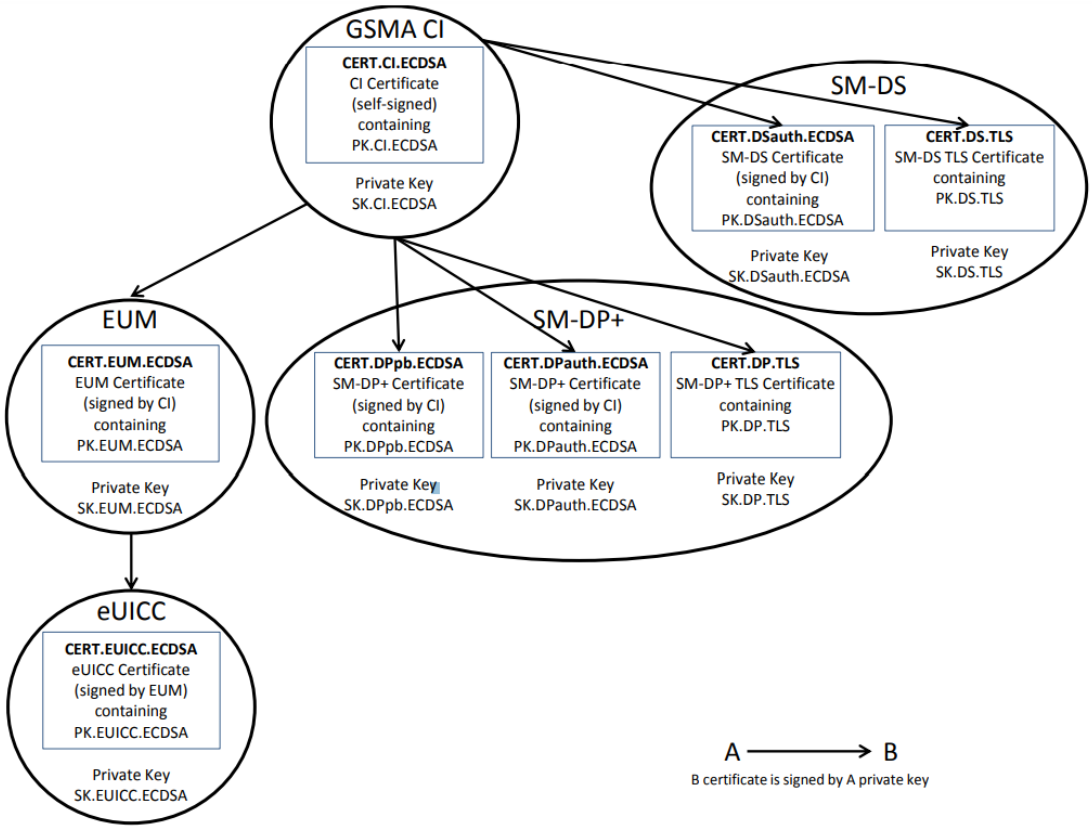


Figura 2.5: Catena di certificati definita originariamente dalla PKI di RSP.

- **Seconda variante AA**: la CA pubblica rilascia certificati per una catena di una o più CA pubbliche subordinate; l'ultima CA pubblica subordinata rilascia certificati per l'SM-DP+ intermediario e l'SM-DS intermediario; l'SM-DP+ intermediario rilascia il certificato CERT.DP.TLS; l'SM-DS intermediario rilascia il certificato CERT.DS.TLS.

All'interno della PKI, il Certificate Issuer di GSMA (CI) è la Root Certification Authority del servizio RSP e, di conseguenza, rappresenta il nodo radice della catena. Inoltre, tutti i certificati possono essere rilasciati direttamente dal CI (root o subordinato), fatta eccezione di CERT.EUICC.SIG che, invece, viene rilasciato dall'EUM (root o subordinato). Tutti i certificati che possono essere rilasciati direttamente dal CI hanno la possibilità di essere revocati in qualunque momento, in particolar modo se le entità corrispondenti (CI, EUM, SM-DP+, SM-DS) vengono compromesse. D'altra parte, i certificati eUICC (CERT.EUICC.SIG) non vengono revocati in modo individuale: di fatto, è difficile che un singolo eUICC venga compromesso. Piuttosto, è più verosimile che un modello eUICC o un intero batch di produzione di eUICC venga dichiarato come compromesso; quando ciò avviene, quello che si fa è revocare direttamente il certificato EUM (CERT.EUM.SIG) associato a quel modello o batch di produzione di eUICC [3].
Il CI fornisce una Certificate Revocation List (CRL), che è la lista dei certificati revocati tra tutti i certificati non scaduti che erano stati rilasciati da quello stesso CI. Ciascun CI, per giunta, deve pubblicare la propria CRL sia periodicamente, sia ogni volta che viene revocato un particolare certificato [3].
In realtà, i certificati relativi alla PKI di RSP di cui si è discusso finora non sono gli unici certificati utilizzati per effettuare il deployment dei profili eSIM: esiste anche un certificato per firmare l'applicazione LPA e un certificato da inserire in ciascun profilo eSIM da distribuire all'end user. Dove entrano in gioco tali certificati? Con riferimento alla guida di Android per le API e per l'implementazione del deployment dei profili eSIM [6], l'interazione tra un'applicazione LPA e l'interfaccia all'eUICC (legata al componente EuiccManager in [6]) può avvenire solo se l'applicazione dispone dei privilegi dell'operatore. Di norma, tali privilegi sono conferiti all'applicazione se il certificato usato per firmarla coincide col certificato presente nel profilo fornito da SM-DP+.

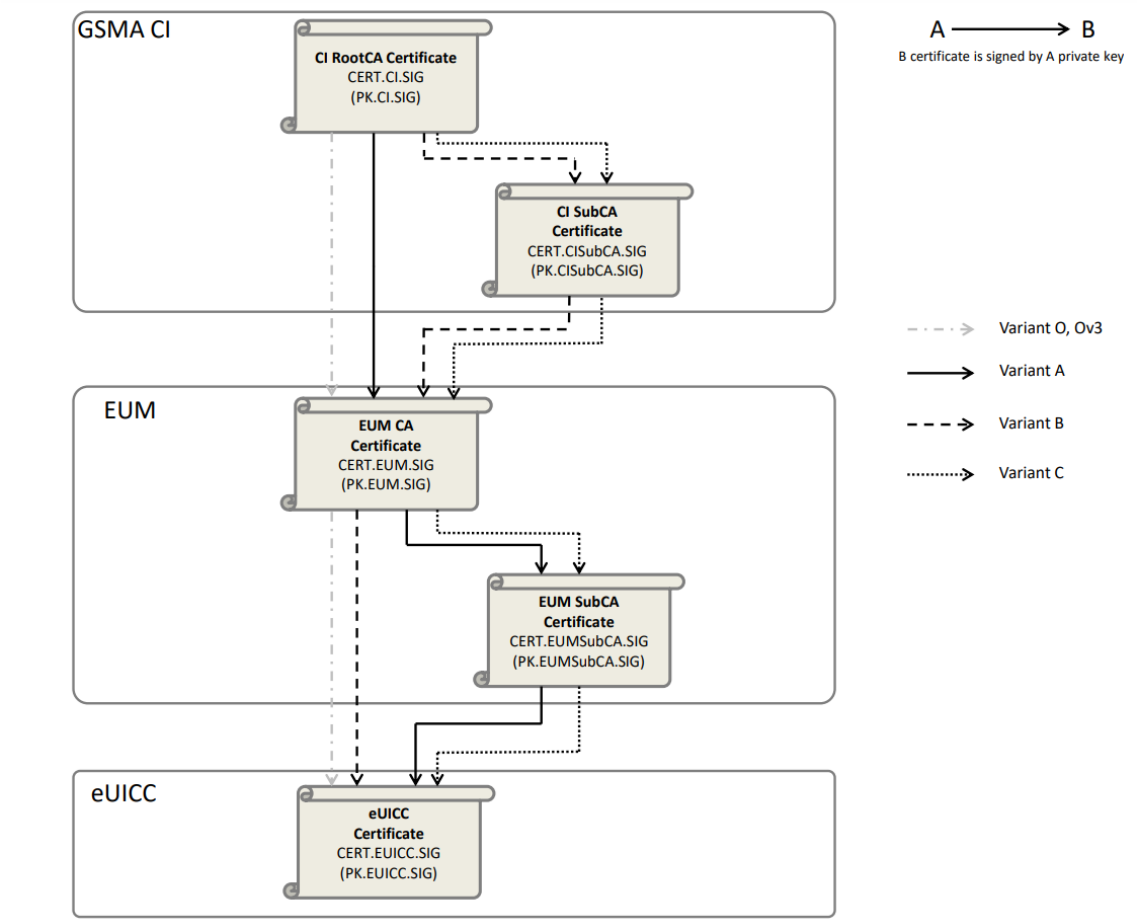


Figura 2.6: Prima porzione dell'attuale catena di certificati definita dalla PKI di RSP.

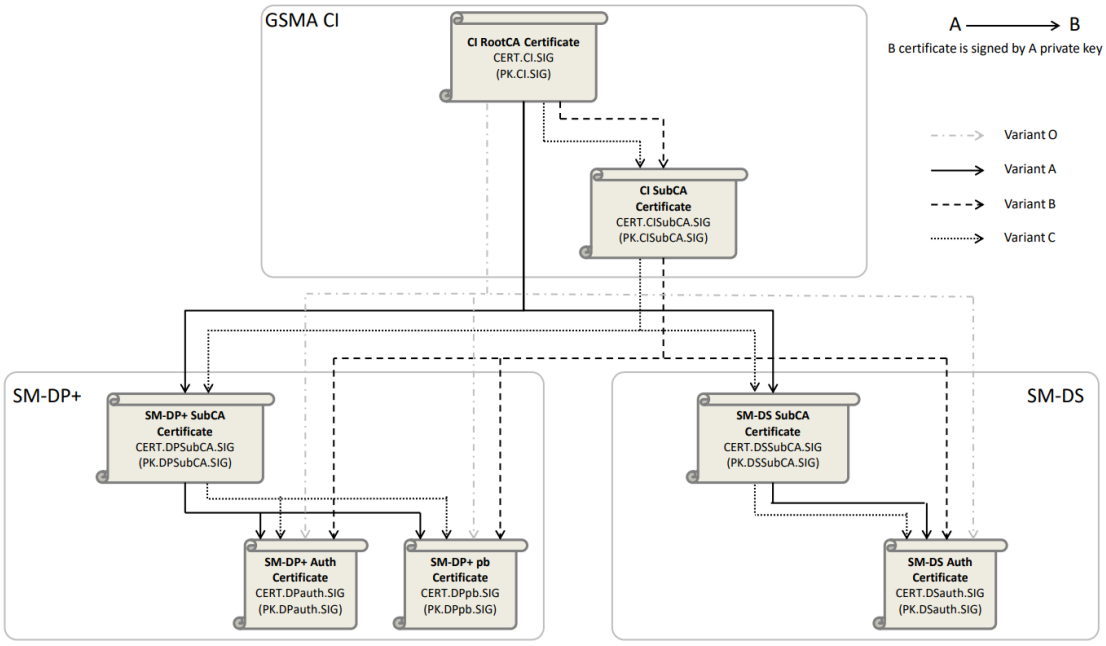


Figura 2.7: Seconda porzione dell'attuale catena di certificati definita dalla PKI di RSP.

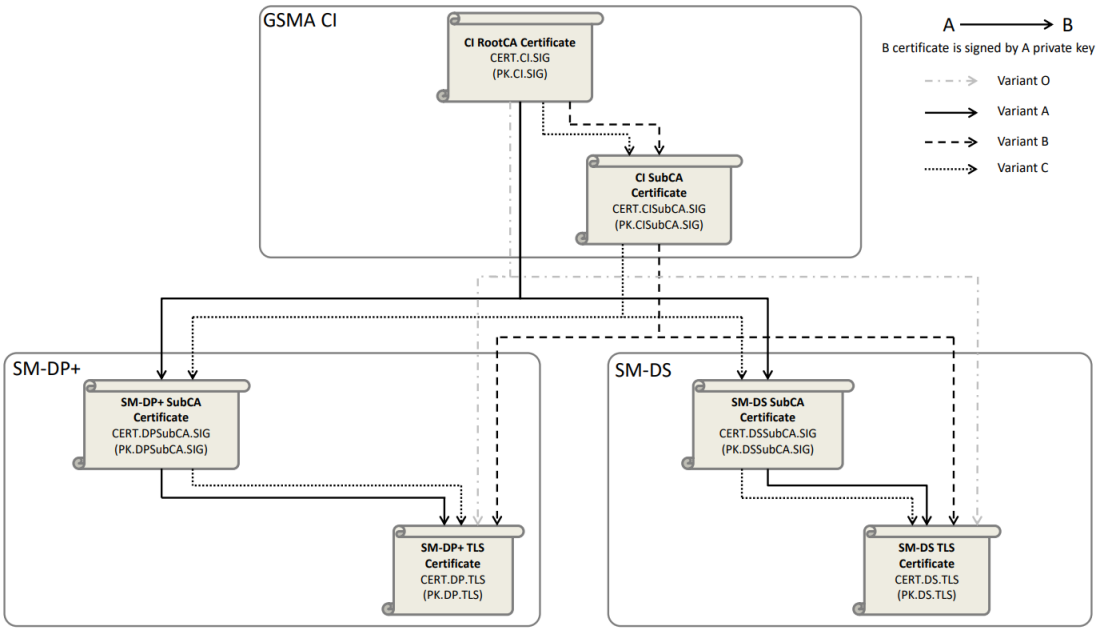


Figura 2.8: Terza porzione dell'attuale catena di certificati definita dalla PKI di RSP.

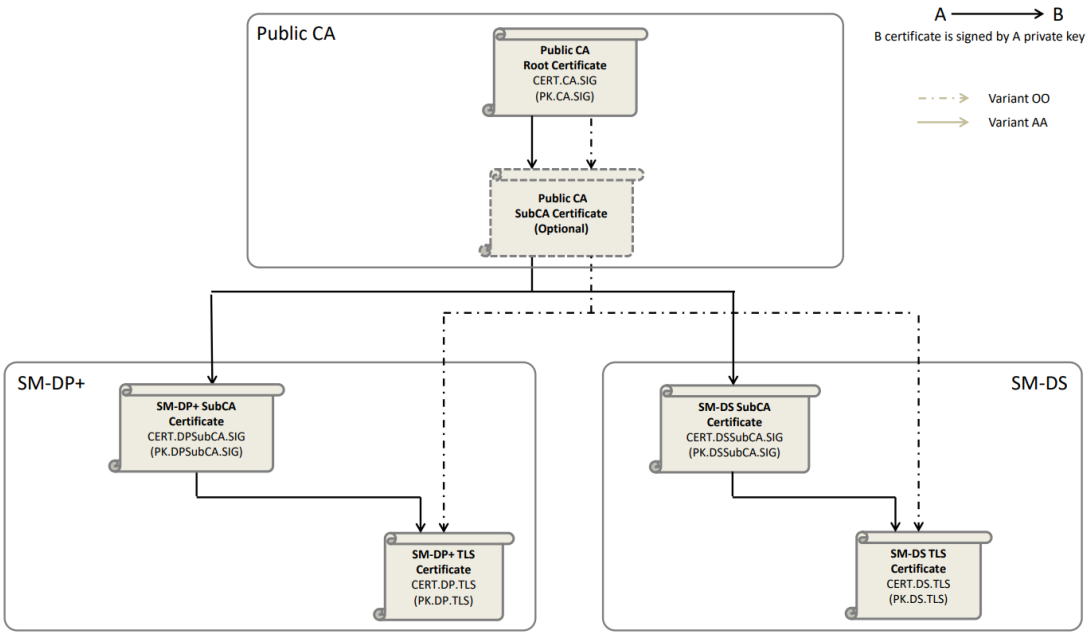


Figura 2.9: Quarta porzione dell'attuale catena di certificati definita dalla PKI di RSP.

2.3.3 Aggiornamento delle chiavi pubbliche nell'eUICC

L'eUICC può fornire un meccanismo per aggiornare il set di chiavi pubbliche memorizzate nell'ECASD dell'eUICC (che sono chiavi a lunga durata). L'implementazione di tale meccanismo è lasciata all'EUM o al produttore del dispositivo mobile e, stando alla guida ufficiale di GSMA [3], deve essere sicura. Tuttavia, nel momento in cui un'implementazione è affidata a terze parti, nessuno può garantire con certezza che la prescrizione sulla sicurezza venga rispettata, poiché non vengono seguite delle linee guida standard e consolidate.

Contenuto dell'ECASD

Tutti gli eUICC devono avere un ECASD che contenga [3]:

- la chiave privata dell'eUICC (SK.EUICC.SIG);
- il certificato dell'eUICC (CERT.EUICC.SIG) contenente la chiave PK.EUICC.SIG, che è la chiave pubblica dell'eUICC;
- la chiave pubblica del CI root (PK.CI.SIG);
- il certificato dell'EUM (CERT.EUM.SIG) e, opzionalmente, il certificato degli EUM subordinati (CERT.EUMSubCA.SIG);
- un keyset dell'EUM per il rinnovo di chiavi e certificati.

Quando avviene l'aggiornamento delle chiavi?

La necessità di rinnovare le chiavi contenute nell'ECASD può presentarsi in due casi [3].

- L'applicazione LPA ha determinato che l'eUICC non supporta più il Public Key identificator (i.e. la rappresentazione in esadecimale dell'identificatore della chiave pubblica) del CI a capo delle chiavi contenute nell'ECASD.
- Durante la procedura di Common Mutual Authentication, il server SM-DP+ ha restituito un certificato CERT.DPauth.SIG che ha come parent un Root Certificate non supportato dall'eUICC.

2.4 Interazione tra eUICC, LPA, SM-DP+ e operatore

2.4.1 Sicurezza TLS

Il protocollo TLS, la cui versione 1.2 è definita in RFC 5246 [7] e la cui versione 1.3 è definita in RFC 8446 [8], è utilizzato per proteggere il traffico sulle interfacce ES2+ (tra server SM-DP+ e operatore) e ES9+ (tra server SM-DP+ e LPA), dove è prevista la mutua autenticazione tra le parti. La documentazione di GSMA di riferimento [3] sottolinea l'obbligatorietà di fare uso di TLS v1.2 sia per gli algoritmi di autenticazione e autorizzazione, sia per l'integrità dei messaggi, sia per la confidenzialità. In realtà, introduce anche la possibilità (e suggerisce) di utilizzare TLS v1.3, che è la versione più recente di TLS e risolve le vulnerabilità che caratterizzano TLS v1.2, per cui, in linea di principio, dovrebbe risultare particolarmente difficoltoso da penetrare. Tuttavia, attualmente sembra essere solo un suggerimento, per cui in uno sviluppo futuro potrebbe essere interessante verificare quale sia la versione di TLS utilizzata per proteggere la comunicazione tra LPA e server SM-DP+.
Un discorso analogo vale per l'interazione che si ha nella mutua autenticazione tra l'eUICC e il server SM-DP+, dove le due parti interagiscono tra loro tramite un TLS tunnel [4]. Per quanto invece riguarda la comunicazione tra eUICC e applicazione LPA, è richiesto l'utilizzo di un pairwise secure channel (ovvero di un canale di comunicazione sicuro rispetto alla confidenzialità e all'autenticazione dei messaggi) che collega le due parti all'interno del dispositivo mobile [4]. Tuttavia, non esiste una specifica universale che imponga l'utilizzo di un particolare protocollo di crittografia per proteggere il pairwise secure channel, anche se il protocollo più utilizzato rimane TLS.

2.4.2 Regole per la comunicazione in RSP

Qualunque comunicazione remota definita per il protocollo RSP deve far fede alle regole riportate di seguito [3].

- **Mutua autenticazione tra eUICC e server SM-DP+**: il server deve essere autenticato per primo da parte dell'eUICC, dove il processo di autenticazione deve includere la verifica di una catena di certificati del server. D'altra parte, l'eUICC deve essere autenticato in un secondo momento da parte del server, dove il processo di autenticazione, di nuovo, deve includere la verifica di una catena di certificati dell'eUICC; l'autenticazione dell'eUICC non si applica all'LPA.
- **Privacy dei dati**: l'eUICC, in quanto client, non deve rivelare alcuna informazione privata a un server SM-DP+ non autenticato. Inoltre, non deve generare materiale firmato prima del completamento del processo di autenticazione del server.
- **Protezione della comunicazione**: quando possibile, la comunicazione tra eUICC e server SM-DP+, oltre a essere protetta dall'integrità dei messaggi, dalla cifratura e dall'autenticazione del mittente, dovrebbe essere caratterizzata dalla proprietà di Perfect Forward Secrecy. Secondo tale proprietà, se anche una chiave a lungo termine viene compromessa, le chiavi di sessione generate a partire da essa rimangono comunque riservate.
- **Autorizzazione**: il server SM-DP+ deve sempre verificare che il client che ha inviato una richiesta sia effettivamente autorizzato prima di far partire l'esecuzione della funzione desiderata.

2.4.3 Step dell'interazione in RSP

L'interazione tra le parti, nel contesto del protocollo RSP, avviene in quattro fasi distinte: profile ordering, download initialization, common handshake e profile download [4].

- **Profile ordering & download initialization**: nella prima fase, l'operatore richiede al server SM-DP+ di preparare un profilo eSIM, e il server gli restituisce dei download initialization pointer (che possono essere rappresentati ad esempio da un codice di attivazione). Nella seconda fase, l'operatore consegna i download initialization pointer all'LPA in modo tale che poi sia possibile effettuare il download vero e proprio del profilo. Per questi primi due step, si possono seguire tre tipi di approcci differenti:
  1. Default server approach: la figura 2.10 tratta da [4] mostra questo approccio. L'operatore (indicato con MNO - Mobile Network Operator) pre-condivide con l'eUICC l'indirizzo S del server. Quando vuole effettuare una nuova sottoscrizione e ottenere così un nuovo profilo, l'end user contatta l'operatore (messaggio 0), il quale ordina al server SM-DP+ il profilo per l'identificatore U dell'eUICC target (messaggio 1). Il server crea il nuovo profilo e lo invia all'operatore (messaggio 2), il quale notifica l'utente (messaggio 3). A tal punto, l'applicazione LPA viene avviata e, tramite un'operazione di get, recupera S dall'eUICC.
  2. Activation Code approach: la figura 2.11 tratta da [4] mostra questo secondo approccio. L'operatore ordina al server SM-DP+ dei profili (messaggio 1), e il server li rende disponibili con un codice di attivazione (tipicamente un codice QR) e li restituisce all'operatore (messaggio 2). Il codice di attivazione include l'indirizzo S del server, l'identificatore I_{ac} del profilo e, opzionalmente, l'OID, ovvero l'identificatore del server SM-DP+. Quando l'end user vuole effettuare una nuova sottoscrizione e ottenere così un nuovo profilo, contatta l'operatore (messaggio 0, che può essere inviato sia prima che dopo l'interazione tra operatore e server SM-DP+ appena descritta), il quale restituisce il codice di attivazione opportuno (messaggio 3).
  3. SM-DS assisted approach: è un approccio analogo all'Activation Code, con la differenza che SM-DP+ si appoggia sui server SM-DS per comunicare con l'eUICC.
- **Common handshake**: questa fase, nota anche come Common Mutual Authentication, coinvolge tre attori fondamentali: il server SM-DP+, l'applicazione LPA e l'eUICC. I relativi dettagli sono riportati nella sezione 2.4.5.

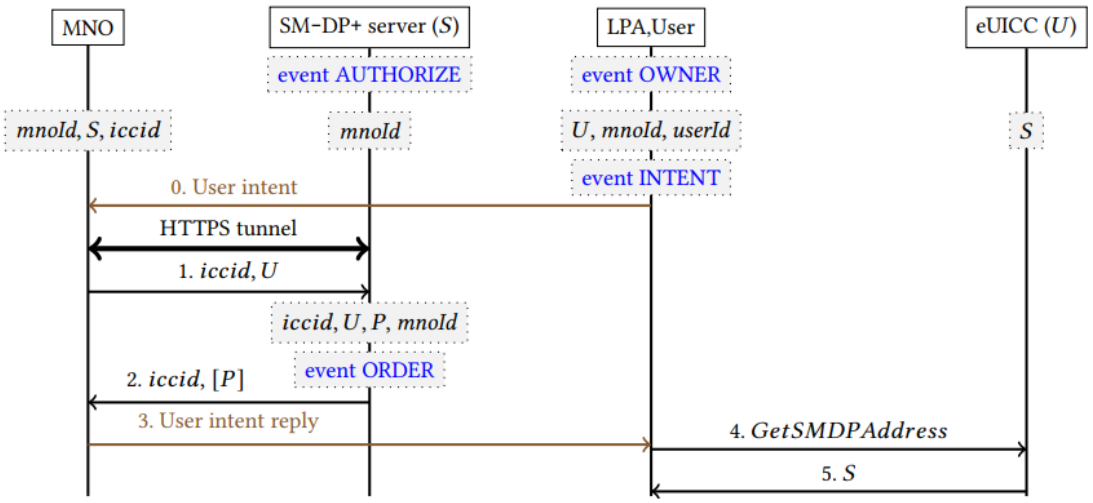


Figura 2.10: Approccio default server per le fasi di profile ordering e download initialization.

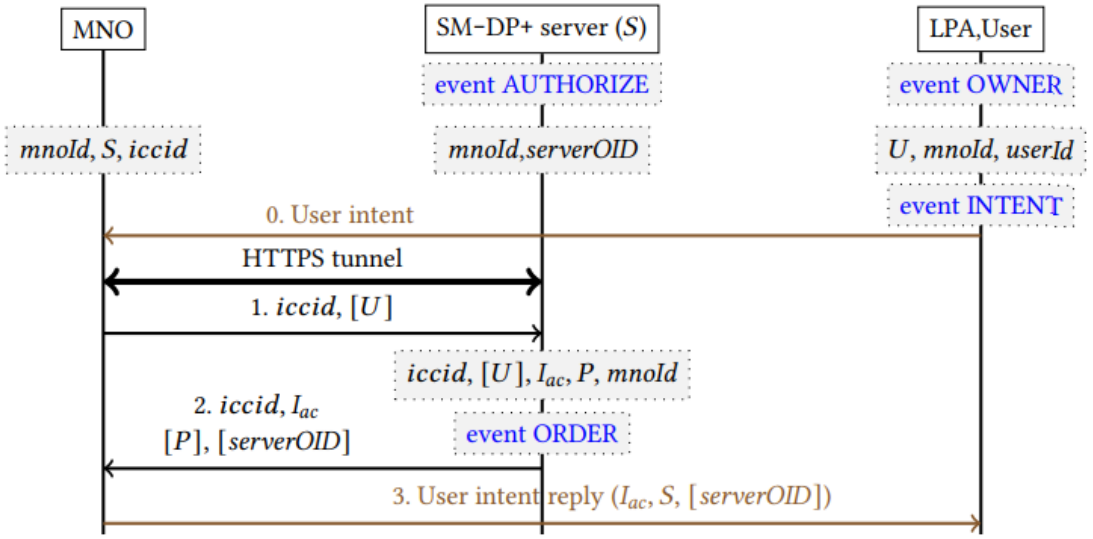


Figura 2.11: Approccio Activation Code per le fasi di profile ordering e download initialization.

- **Profile download**: in quest'ultima fase, i cui dettagli sono riportati nella sezione 2.4.6, viene calcolato uno shared secret da cui, mediante una key derivation function (KDF), saranno derivate le chiavi one-time di sessione k (per la cifratura) e k' (per l'integrità dei dati). Avviene poi il download del profilo eSIM, in cui il server SM-DP+ invia all'LPA il profilo cifrato con la chiave k e l'operatore di riferimento, dove entrambe le informazioni sono firmate singolarmente con la chiave k' mediante il meccanismo di message authentication code (MAC). Dopodiché, l'LPA mostra l'operatore all'utente, che dovrà stabilire se è corretto: se sì, l'LPA inoltra tutte le informazioni all'eUICC il quale, dopo aver derivato a sua volta le chiavi di sessione k, k', dovrà decriptare il profilo P, che risulterà finalmente essere utilizzabile.

2.4.4 Ciclo di vita dei profili in SM-DP+

Prima di approfondire più nel dettaglio l'interazione in RSP, è bene illustrare il ciclo di vita dei profili eSIM.
La tabella 2.4 tratta da [3][5] fornisce un elenco degli stati in cui ciascun profilo eSIM può trovarsi nell'arco della sua esistenza. Nelle figure 2.12, 2.13 tratte da [3][5], invece, sono mostrati due diagrammi a stati finiti che illustrano per bene il ciclo di vita dei profili in SM-DP+.

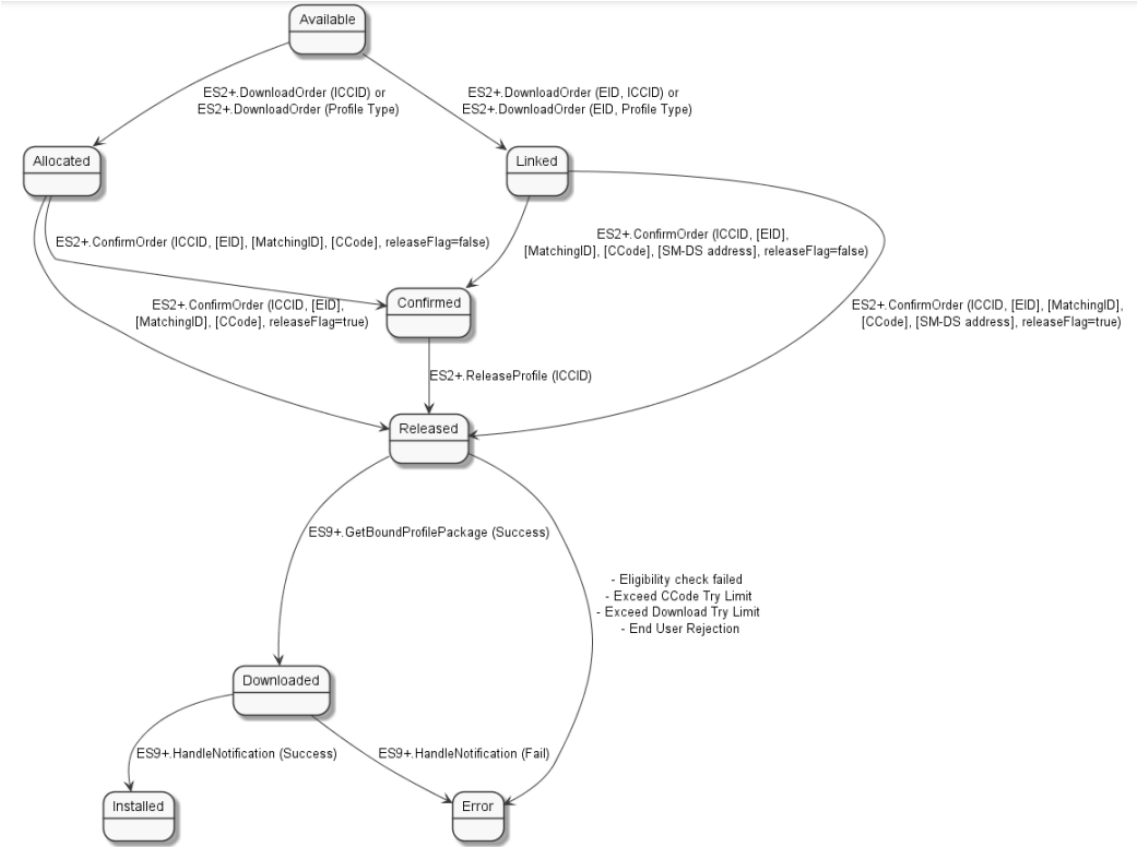


Figura 2.12: Primo diagramma a stati per i profili eSIM.

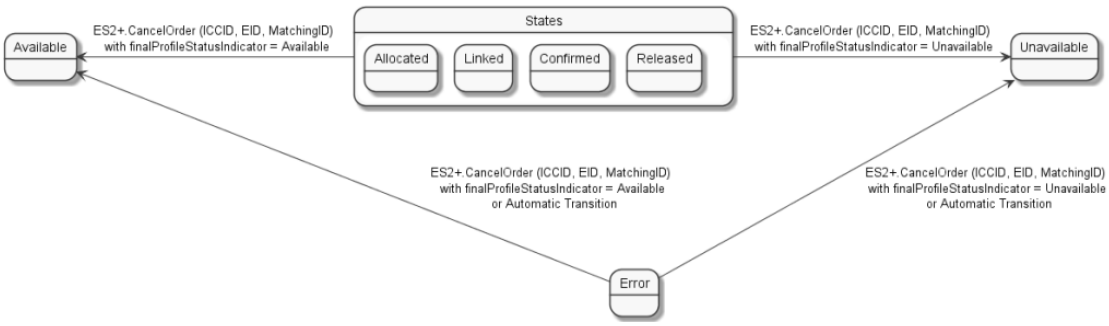


Figura 2.13: Secondo diagramma a stati per i profili eSIM.

Tabella 2.4: Stati dei profili eSIM.

Nome	**Descrizione**
Available	Il profilo è disponibile nell'inventory del server SM-DP+.
Allocated	Il profilo è riservato per il download senza essere linkato a un EID (eUICC ID).
Linked	Il profilo è riservato per il download ed è linkato a un EID.
Confirmed	Il profilo è riservato per il download (che sia esso linkato o non linkato a un EID) col Matching ID (i.e. il codice che identifica la transazione di download) e il codice di conferma (i.e. il codice che deve essere inserito dall'end user), se richiesti.
Released	Il profilo è pronto per il download e l'installazione dopo che l'operatore ha effettuato la configurazione di rete.
Downloaded	Il profilo è stato consegnato all'LPA (i.e. è stato scaricato).
Installed	Il profilo è stato installato sull'eUICC con successo.
Error	Il profilo non è stato installato a causa di una condizione di errore.
Unavailable	Il profilo non può essere più riutilizzato da SM-DP+.

Con riferimento alla figura 2.12, a partire dallo stato Available:

- Si passa allo stato Allocated se si effettua l'ordine di download senza specificare l'EID.
- Si passa allo stato Linked se si effettua l'ordine di download specificando l'EID.

A partire dallo stato Allocated:

- Si passa allo stato Confirmed se si conferma l'ordine di download con releaseFlag=false.
- Si passa allo stato Released se si conferma l'ordine di download con releaseFlag=true.

A partire dallo stato Linked:

- Si passa allo stato Confirmed se si conferma l'ordine di download con releaseFlag=false.
- Si passa allo stato Released se si conferma l'ordine di download con releaseFlag=true.

A partire dallo stato Confirmed:

- Si passa allo stato Released se si effettua il rilascio del profilo in modo tale che sia effettivamente pronto per il download e l'installazione.

A partire dallo stato Released:

- Si passa allo stato Downloaded se il profilo viene consegnato all'LPA con successo.
- Si passa allo stato Error se si ha un errore nel consegnare il profilo all'LPA.

A partire dallo stato Downloaded:

- Si passa allo stato Installed se il profilo viene installato sull'eUICC con successo.
- Si passa allo stato Error se si ha un errore nell'installare il profilo sull'eUICC.

Con riferimento alla figura 2.13, a partire dagli stati Allocated / Linked / Confirmed / Released:

- Si torna allo stato Available se l'ordine di download viene annullato con finalProfileStatusIndicator=Available.
- Si passa allo stato Unavailable se l'ordine di download viene annullato con finalProfileStatusIndicator=Unavailable.

A partire dallo stato Error:

- Si torna allo stato Available con una transizione automatica oppure se l'ordine di download viene annullato con finalProfileStatusIndicator=Available.
- Si passa allo stato Unavailable con una transizione automatica oppure se l'ordine di download viene annullato con finalProfileStatusIndicator=Unavailable.

2.4.5 Dettagli sulla Common Mutual Authentication

La figura 2.14 ripresa da [3] illustra tutti i messaggi che il server SM-DP+, l'LPA e l'eUICC si scambiano tra loro e le operazioni che queste tre entità svolgono durante la procedura di Common Mutual Authentication. Tutti i relativi dettagli [3] sono spiegati nelle sottosezioni successive. Si osservi che l'intero meccanismo resta valido e invariato se si ha un server SM-DS al posto del server SM-DP+.

Condizioni iniziali

- Il server SM-DP+ è dotato di:
  - certificato CERT.DPauth.SIG;
  - chiave privata SK.DPauth.SIG;
  - certificato del CI (CERT.CI.SIG);
  - certificato TLS CERT.DP.TLS;
  - chiave privata TLS SK.DP.TLS;
  - certificati di SM-DP+ intermediari, se esistenti (CERT.DPSubCA.SIG).
- L'eUICC, d'altra parte, è dotato di:
  - certificato CERT.EUICC.SIG;
  - chiave privata SK.EUICC.SIG;
  - certificato dell'EUM (CERT.EUM.SIG);
  - certificati di CI subordinati, se esistenti (CERT.CISubCA.SIG);
  - certificati di EUM subordinati, se esistenti (CERT.EUMSubCA.SIG);
  - chiave pubblica del CI (PK.CI.SIG).

Procedimento

1. Il primo step, che è opzionale, si articola in tre punti:
   a) Se non lo aveva già fatto in un momento precedente, l'LPA richiede le informazioni dell'eUICC (i.e. Specification Version Number e identificatori delle chiavi pubbliche del CI che possono essere utilizzate per autenticare il server e per firmare i propri dati) identificate dalla variabile euiccInfo1.
   b) L'eUICC restituisce euiccInfo1 all'LPA.
   c) Se esiste una restrizione sulle chiavi pubbliche consentite del CI, l'LPA crea una nuova istanza di euiccInfo1 senza le chiavi pubbliche non compatibili del CI. Se dopo questa operazione rimane una lista vuota di chiavi pubbliche, l'LPA informa l'end user che la procedura di mutua autenticazione deve terminare.
2. L'LPA richiede all'eUICC una challenge (euiccChallenge).
3. L'eUICC genera la challenge che successivamente dovrà essere firmata dal server SM-DP+ per l'autenticazione del server stesso.
4. L'eUICC restituisce la challenge all'LPA.
5. L'LPA stabilisce una nuova connessione HTTPS col server SM-DP+. Il setup della sessione TLS, poiché non può riutilizzare le chiavi da una sessione precedente, deve prevedere un nuovo key exchange, che avviene nella fase del TLS handshake. Qui il server fornisce all'LPA un certificato CERT.DP.TLS e l'LPA deve verificarne la validità; se l'LPA non riesce a effettuare la verifica, il server deve inviargli un certificato CERT.DP.TLS differente, e così via, fin tanto che l'LPA non sarà riuscito a validare un certificato oppure il numero di tentativi per ritrasmettere il certificato non avrà raggiunto il limite massimo. In questo secondo caso l'LPA interrompe la procedura di Common Mutual Authentication.
6. L'LPA invoca la funzione *InitiateAuthentication* passandovi come parametri le proprie capability, euiccChallenge, euiccInfo1 e l'indirizzo SM-DP+, il quale viene usato dall'LPA per accedere al server.

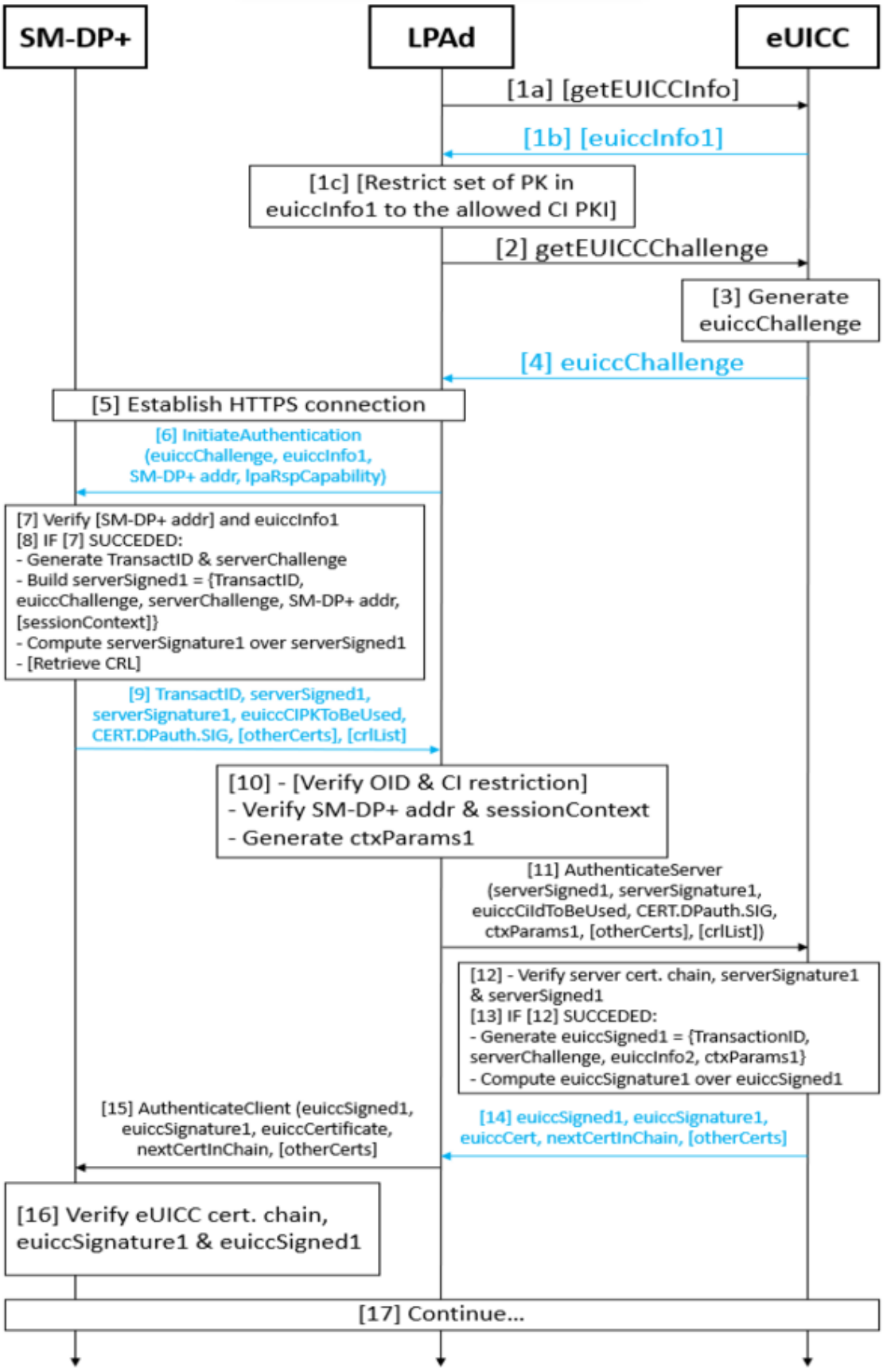


Figura 2.14: Sequence diagram che descrive la Common Mutual Authentication.

7. Il server SM-DP+ esegue le seguenti operazioni:

   - Verificare che l'indirizzo SM-DP+ inviato dall'LPA sia valido.
   - Verificare il contenuto della variabile euiccInfo1, incluse le chiavi pubbliche del CI, tra cui deve essercene almeno una che può essere accettata dal server stesso.

   Se anche solo uno di questi controlli non va a buon fine, il server restituisce una condizione di errore e l'LPA interrompe la procedura di Common Mutual Authentication.

8. Il server SM-DP+ esegue le seguenti altre operazioni:

   - Generare un ID di transazione che serve per identificare la sessione RSP.
   - Generare una challenge (serverChallenge) che dovrà essere firmata dall'eUICC per l'autenticazione dell'eUICC stesso.
   - Generare una struttura dati serverSigned1 a partire dall'ID di transazione, da euiccChallenge, da serverChallenge, dall'indirizzo SM-DP+ ed eventualmente dal contesto di sessione.
   - Calcolare la serverSignature1 a partire da serverSigned1, utilizzando la chiave privata SK.DPauth.SIG.
   - Se sia l'eUICC che l'LPA prevedono il crlStaplingV3Support, che è la funzionalità che permette al server di fornire una Certificate Revocation List (CRL), il server deve anche recuperare la CRL per ciascun certificato che abbia l'estensione CRLDistributionPoints (i.e. per ciascun certificato che dà informazioni su dove è possibile ottenere una CRL).

9. Il server SM-DP+ restituisce all'LPA alcune informazioni, tra cui l'ID della transazione, serverSigned1, serverSignature1, euiccCIPKToBeUsed (che dovrà corrispondere alla chiave pubblica del CI accettata tra quelle proposte dall'eUICC), il certificato CERT.DPauth.SIG, eventuali altri certificati ed eventualmente la CRL.

10. L'LPA esegue le seguenti operazioni:

    - Verificare l'OID, che è l'Object Identifier del server SM-DP+ (se fornito in precedenza).
    - Verificare che l'indirizzo SM-DP+ restituito dal server (incapsulato in serverSigned1) matchi con l'indirizzo SM-DP+ che l'LPA aveva inviato allo step (6).
    - Verificare che la chiave pubblica associata del Root della certificate chain associata al certificato CERT.DPauth.SIG sia inclusa in euiccInfo1 (se l'opzione euiccCiUpdateSupport dell'LPA è attiva).
    - Effettuare altre verifiche sui certificati che non verranno approfondite in questa sede.

    Se anche solo uno di questi controlli non va a buon fine, l'LPA interrompe la procedura di Common Mutual Authentication. In caso contrario, procede col generare la struttura dati ctxParams1, che dovrà essere inviata all'eUICC affinché venga poi inclusa tra i dati firmati.

11. L'LPA invoca la funzione *AuthenticateServer* passandovi come parametri serverSigned1, serverSignature1, euiccCiIdToBeUsed, il certificato CERT.DPauth.SIG, eventuali altri certificati, ctxParams1 ed eventualmente la CRL.

12. L'eUICC esegue le seguenti operazioni:

    - Verificare il certificato CERT.DPauth.SIG e altri eventuali certificati nella catena.
    - Verificare serverSignature1.
    - Verificare che l'euiccChallenge contenuta in serverSigned1 matchi con la challenge generata dall'eUICC stessa allo step (3).
    - Verificare che la chiave pubblica del CI sia effettivamente supportata.
    - Se il contesto di sessione prevede il crlStaplingV3Support, l'eUICC deve anche verificare la validità della CRL e assicurarsi che nessun certificato all'interno della certificate chain sia stato revocato.

    Se anche solo uno di questi controlli non va a buon fine, la procedura di Common Mutual Authentication deve essere interrotta. In caso contrario, il server SM-DP+ risulta autenticato all'eUICC.

13. L'eUICC esegue le seguenti altre operazioni:
    - Generare la struttura dati euiccSigned1 a partire dall'ID di transazione, dall'indirizzo del server SM-DP+, da serverChallenge, da euiccInfo2 e da ctxParams1, dove euiccInfo2 è un sovrainsieme di euiccInfo1 e comprende le informazioni complete dell'eUICC. Si noti che euiccChallenge non è incluso in euiccSigned1.
    - Calcolare la euiccSignature1 a partire da euiccSigned1, utilizzando la chiave privata SK.EUICC.SIG.
14. L'eUICC restituisce all'LPA alcune informazioni, tra cui euiccSigned1, euiccSignature1 e la catena di certificati dell'eUICC.
15. L'LPA invoca la funzione *AuthenticateClient* passandovi come parametri euiccSigned1, euiccSignature1 e la catena di certificati dell'eUICC.
16. Il server SM-DP+ esegue le seguenti operazioni:
    - Verificare che l'ID di transazione contenuto in euiccSigned1 corrisponda a quello comunicato dal server stesso allo step (9).
    - Verificare che il certificato root della certificate chain comunicata dall'eUICC corrisponda con quella selezionata dal server stesso (euiccCIPKToBeUsed) durante l'esecuzione della funzione *InitiateAuthentication*.
    - Verificare che la certificate chain dell'eUICC sia valida.
    - Verificare euiccSignature1.
    - Verificare che la serverChallenge contenuta in euiccSigned1 matchi con la challenge generata dal server stesso allo step (8).

    Se anche solo uno di questi controlli non va a buon fine, il server restituisce una condizione di errore e l'LPA interrompe la procedura di Common Mutual Authentication. In caso contrario, l'eUICC risulta autenticato al server SM-DP+.

In definitiva, in questa sezione è emerso come l'LPA svolga sì il ruolo di relay nell'interazione tra server SM-DP+ ed eUICC, ma svolge anche delle importanti funzioni di sicurezza. Infatti, com'è stato già menzionato, si occupa anzitutto di verificare che il certificato CERT.DP.TLS sia valido e, in un secondo momento, effettua altri controlli sull'OID, sull'indirizzo del server SM-DP+, sulla chiave pubblica associata al certificato CERT.DPauth.SIG e sulla certificate chain del server nello specifico. Tale caratteristica implica la necessità di considerare l'applicazione LPA come un'entità trusted.

2.4.6 Dettagli su download e installazione dei profili

La figura 2.15 ripresa da [3] illustra tutti i messaggi che l'operatore, il server SM-DP+, l'LPA e l'eUICC si scambiano tra loro e le operazioni che queste quattro entità svolgono durante la procedura di download e installazione dei profili. Tutti i relativi dettagli [3] sono spiegati nelle sottosezioni successive. Anche qui l'intero meccanismo resta valido se si ha un server SM-DS al posto del server SM-DP+, a meno di variazioni di poco conto.

Condizioni iniziali

- L'applicazione LPA potrebbe aver recuperato l'indirizzo del server SM-DP+ e l'identificativo della chiave pubblica del CI Root consentita; se tale identificativo viene effettivamente recuperato a partire dall'eUICC, allora l'LPA deve restringere l'insieme degli identificativi delle chiavi pubbliche dei CI Root compatibili a quel valore.
- Per ogni profilo nello stato Released il server SM-DP+ mantiene un contatore dei tentativi di download di quel profilo e un contatore dei tentativi di immissione del codice di conferma. Di fatto, il server deve limitare il valore di questi due contatori.
- Se è richiesto l'Activation Code per il download e l'installazione del profilo, l'end user deve averlo già immesso all'LPA.

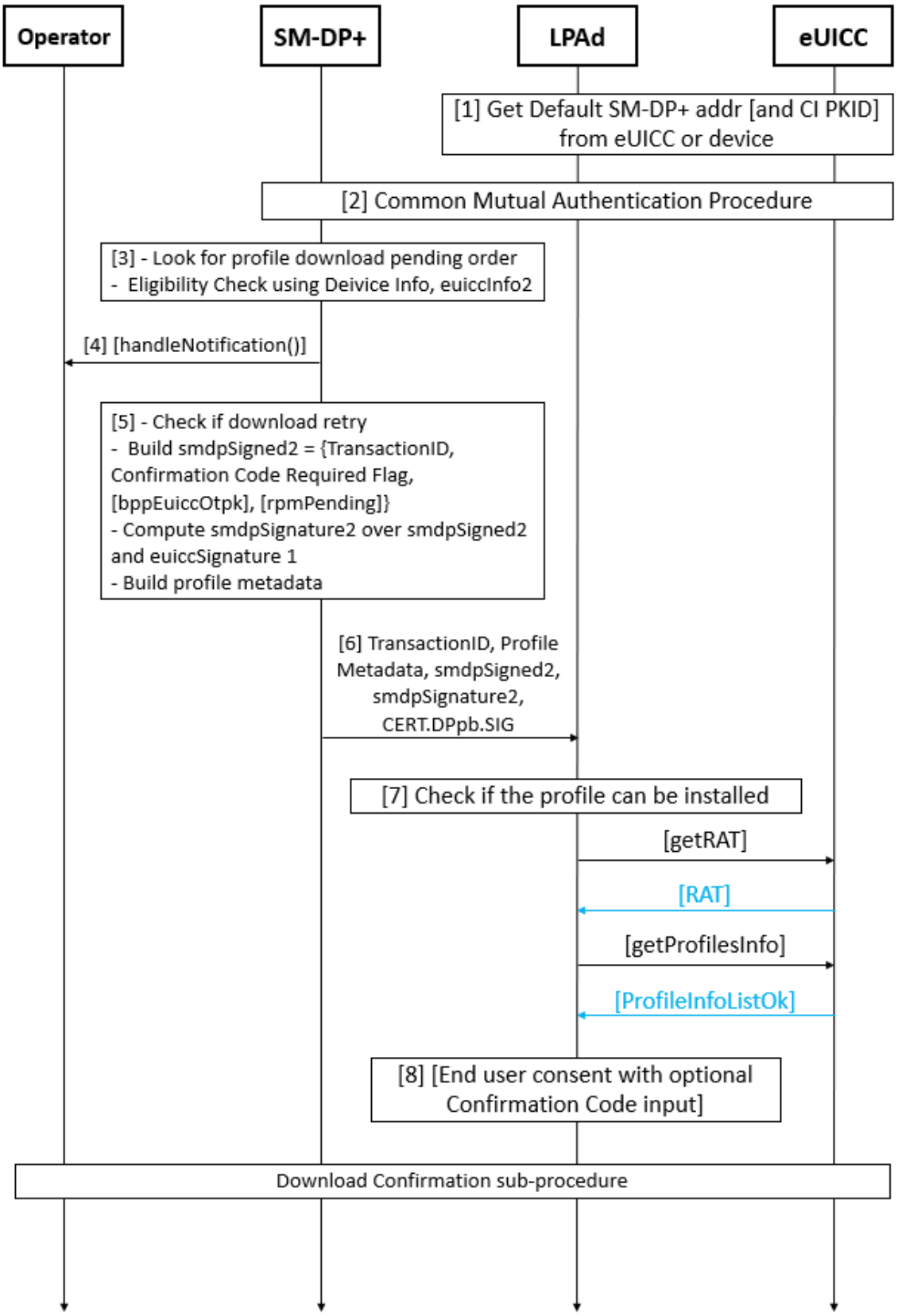


Figura 2.15: Sequence diagram che descrive il download e l'installazione dei profili.

Procedimento

1. Se non lo ha già fatto in precedenza e se è necessario, l'applicazione LPA effettua il parsing dell'Activation Code: così ottiene l'indirizzo del server SM-DP+ e, opzionalmente, l'OID del server SM-DP+ e l'identificativo della chiave pubblica del CI Root.

2. Viene eseguita la procedura di Common Mutual Authentication definita nella sezione 2.4.5. In particolare, nel messaggio *AuthenticateClient*, l'LPA deve specificare il Matching ID, che è l'identificatore della transazione di download corrente e, dunque, individua esattamente il profilo che si vuole installare.

3. Il server SM-DP+ esegue le seguenti operazioni:

   - Verificare che esista un profilo correlato al Matching ID ricevuto in *AuthenticateClient*.
   - Se l'ordinazione di download del profilo è già linkato a un EID, verificare che quest'ultimo corrisponda all'EID dell'eUICC appena autenticato.
   - Verificare che il profilo sia nello stato Released o, nel caso di retry a seguito del fallimento di un'installazione precedente, nello stato Downloaded.

   Se anche solo uno di questi controlli non va a buon fine, la procedura di download e installazione del profilo deve essere interrotta. In caso contrario, il server SM-DP+ deve procedere con le seguenti altre azioni:

   - Incrementare il contatore dei tentativi di download del profilo target. Se il numero massimo di tentativi viene sforato, il server deve notificare l'operatore del fallimento del download e l'intera procedura deve terminare.
   - Effettuare i check di idoneità opportuni.

4. Il server SM-DP+ notifica l'operatore con l'esito dei check di validità mediante la funzione *handleNotification* (step opzionale). La comunicazione tra server e operatore è protetta dall'uso di un pairwise secure channel, come nell'interazione tra eUICC e LPA; anche qui il protocollo più utilizzato è TLS ma non viene imposto dallo standard di GSMA. Di conseguenza, potrebbe essere nuovamente utile stabilire empiricamente se nel canale di comunicazione tra server SM-DP+ e operatore viene utilizzato TLS o meno.

5. Se i check di idoneità falliscono, allora il server SM-DP+ esegue le seguenti operazioni:

   - Portare il profilo target allo stato Error.
   - Restituire uno status di errore all'LPA in modo tale che l'intera procedura termini.

   In caso contrario, il server SM-DP+ esegue le seguenti altre operazioni:

   - Determinare se è richiesto un codice di conferma (Confirmation Code) per l'ordinazione pendente.
   - Generare una struttura dati smdpSigned2 che contiene le proprie informazioni.
   - Calcolare smdpSignature2, che è un fingerprint ottenuto da smdpSigned2 ed euiccSignature1, dove euiccSignature1 è un'informazione che è stata scambiata durante la fase di Common Mutual Authetication.
   - Generare i metadati del profilo target.

6. Il server SM-DP+ fornisce all'LPA la risposta ad *authenticateClient* (funzione invocata durante la procedura di Common Mutual Authentication), che comprende transactionId, i metadati del profilo, smdpSigned2, smdpSignature2 e CERT.DPpb.SIG.

7. L'LPA verifica se il profilo può essere effettivamente installato. Per far ciò, deve ricorrere a un'apposita struttura dati detta Rules Authorisation Table (RAT) e/o alla lista dei profili installati. Se non dispone già di queste informazioni, deve richiederle all'eUICC invocando le funzioni *getRAT* e/o *getProfilesInfo*. Informazioni più dettagliate sulla RAT e, più in generale, sulla gestione delle policy dei profili, sono riportate nella sezione 2.5.

8. Se necessario, l'LPA richiede all'end user di inserire il codice di conferma che l'operatore gli aveva fornito. In caso contrario, l'LPA richiede una conferma semplice (Simple Confirmation) sul download del profilo, che può prevedere poche semplici opzioni di risposta come 'Yes', 'No', 'Not now'. Se l'end user non inserisce correttamente il codice di conferma o non risponde positivamente alla conferma semplice, si procede con la cancellazione della sessione (Common Cancel Session Procedure, i cui dettagli sono riportati nella sezione 2.4.9); altrimenti, l'installazione del profilo può completarsi con successo mediante la sotto-procedura di Download Confirmation, che verrà illustrata qui di seguito.

2.4.7 Sotto-procedura di Download Confirmation

La figura 2.16 ripresa da [3] illustra tutti i messaggi che l'operatore, il server SM-DP+, l'LPA e l'eUICC si scambiano tra loro e le operazioni che queste quattro entità svolgono durante la sotto-procedura di Download Confirmation, che conclude la procedura di download e installazione dei profili introdotta poc'anzi. Tutti i relativi dettagli [3] sono spiegati nelle sottosezioni successive. Ancora una volta i passaggi rimangono gli stessi se si ha un server SM-DS al posto del server SM-DP+.

Condizioni iniziali

- L'end user ha consentito con successo il download del profilo target.

Procedimento

1. L'LPA invoca la funzione *prepareDownload* passandovi come parametri le informazioni precedentemente ricevute del server SM-DP+ (come smdpSigned2, smdpSignature2 e il certificato CERT.DPpb.SIG) e opzionalmente l'hash del codice di conferma (Confirmation Code).

2. L'eUICC esegue le seguenti operazioni:

   - Verificare che CERT.DPpb.SIG sia un certificato valido.
   - Verificare che CERT.DPauth.SIG e CERT.DPpb.SIG appartengano alla medesima entità e siano state rilasciate dalla medesima CA.
   - Verificare smdpSignature2.
   - Verificare che l'ID di transazione contenuto in smdpSigned2 corrisponda con l'ID di transazione che identifica la sessione RSP corrente.

   Se anche solo uno di questi controlli non va a buon fine, l'eUICC deve restituire uno stato di errore e la procedura di download e installazione del profilo deve essere interrotta.

3. L'eUICC prosegue con le seguenti altre azioni:

   - Generare una coppia di chiavi one-time (otPK.EUICC.KA, otSK.EUICC.KA).
   - Generare la struttura dati euiccSigned2, che comprende l'ID di transazione, la chiave pubblica one-time otPK.EUICC.KA ed eventualmente l'hash del Confirmation Code.
   - Calcolare euiccSignature2.

4. L'eUICC restituisce all'LPA la risposta a *prepareDownload*.

5. L'LPA invoca la funzione *getBoundProfilePackage* passandovi come parametri euiccSigned2 ed euiccSignature2.

6. Il server SM-DP+ verifica euiccSignature2 e stabilisce se è richiesto il codice di conferma: se sì, si procede con lo step (7), altrimenti si passa allo step (9).

7. Il server SM-DP+ esegue le seguenti operazioni:

   - Recuperare l'hash del Confirmation Code ottenuto dalla funzione *confirmOrder* relativa alle fasi di profile ordering & download initialization, e calcolarne il valore atteso come SHA256(SHA256(Confirmation Code) | transactionID).
   - Verificare che l'hash del Confirmation Code ricevuto all'interno di euiccSigned2 corrisponda col valore hash atteso.

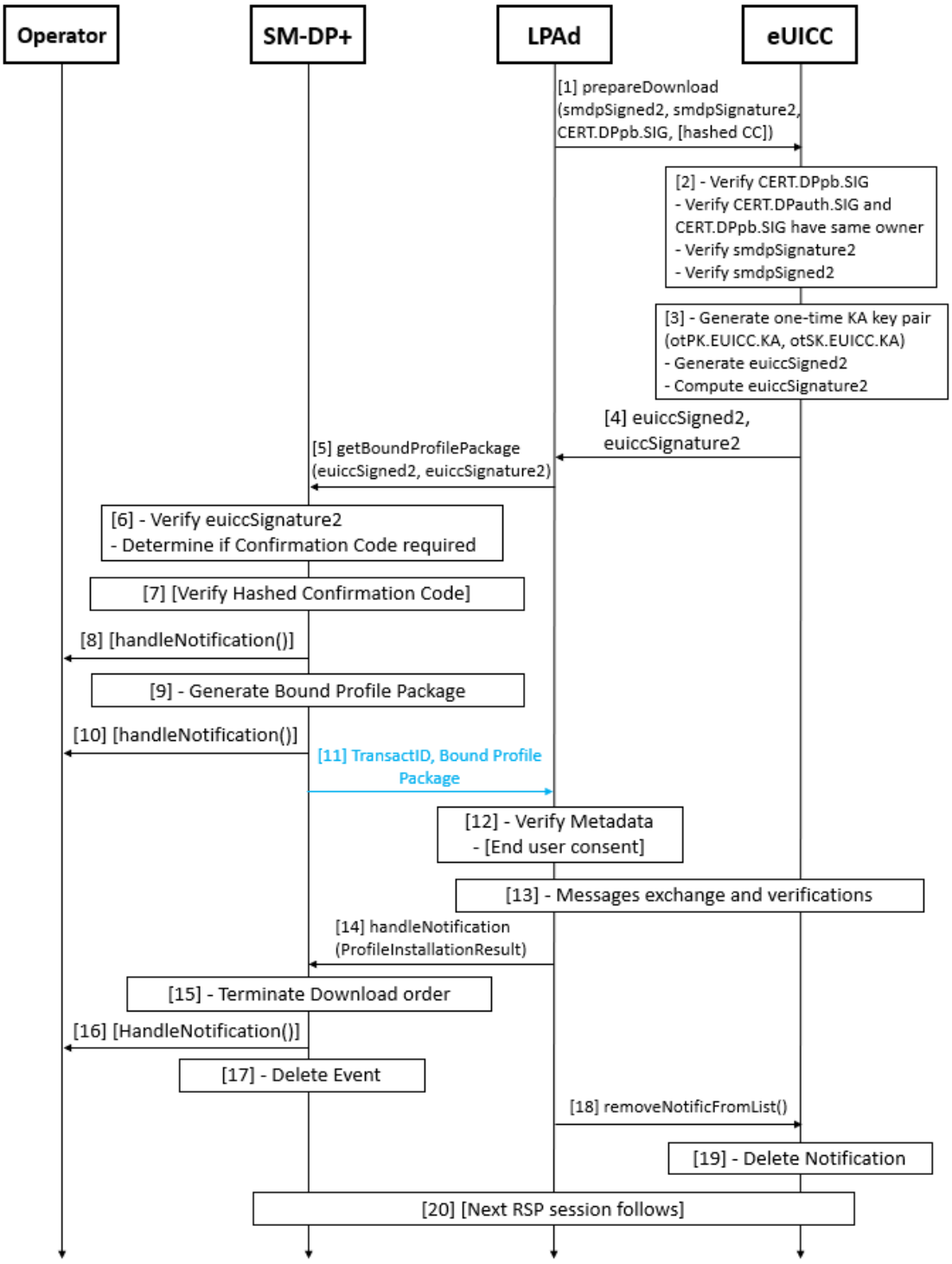


Figura 2.16: Sequence diagram che descrive la sotto-procedura di Download Confirmation.

- Se la verifica dell'hash del Confirmation Code fallisce, incrementare di un'unità il contatore dei tentativi di immissione del codice e verificare che tale contatore non sfori il numero massimo ammissibile.

Se anche solo uno di questi controlli non va a buon fine, il server SM-DP+ deve restituire uno stato di errore e la procedura di download e installazione del profilo deve essere interrotta. Più precisamente, se fallisce il controllo sul contatore dei tentativi di immissione del codice di attivazione, è necessario anche impostare il profilo target nello stato Error e passare allo step (8) prima di interrompere la procedura. Se invece i controlli vanno tutti a buon fine, si passa allo step (9).

8. Se il numero di tentativi di immissione del codice di attivazione supera il massimo stabilito, il server SM-DP+ informa l'operatore del fallimento mediante la funzione *handleNotification*; dopodiché, la procedura di download e installazione del profilo viene interrotta.

9. Il server SM-DP+ genera una coppia di chiavi one-time (otPK.DP.KA, otSK.DP.KA) da cui, assieme alla chiave otPK.EUICC.KA precedentemente ricevuta, ricava uno shared secret usato per cifrare e autenticare alcuni metadati del profilo da inviare al client; dopodiché, a partire da tali metadati e dal payload (cifrato) del profilo, genera il Bound Profile Package.

10. Il server SM-DP+ può opzionalmente notificare l'operatore del completamento del download del profilo mediante la funzione *handleNotification*.

11. Il server SM-DP+ invia all'LPA la risposta a *getBoundProfilePackage* e imposta lo stato del profilo target a Downloaded.

12. Se l'LPA ha già trattato la struttura dati Profile Metadata in precedenza, allora deve assicurarsi che le informazioni contenute nel Bound Profile Package (BPP) appena ricevuto le informazioni relative ai metadati del profilo siano rimaste coerenti; se non dovessero esserlo, deve avviare la Common Cancel Session Procedure (2.4.9) per cancellare la sessione corrente.

13. L'LPA e l'eUICC si scambiano una serie di messaggi per effettuare opportuni controlli e per portare a termine l'installazione del profilo all'interno dell'eUICC.

14. L'LPA invoca la funzione *handleNotification* con lo scopo di consegnare il risultato dell'installazione del profilo al server SM-DP+, il quale risponderà con un semplice acknowledgement.

15. Il server SM-DP+ esegue le seguenti operazioni:

    - Recuperare l'ordinazione di download del profilo target identificata dall'ID di transazione. Se viene ricevuto un ID di transazione ignoto, bisogna interrompere la procedura.
    - Chiudere l'ordinazione di download del profilo target e impostare il relativo stato a Installed o Error in base a quanto indicato dal risultato dell'installazione del profilo.

16. Eventualmente il server SM-DP+ invoca ancora una volta la funzione *handleNotification* specificando l'esito dell'installazione del profilo sull'eUICC.

17. Se oltre al server SM-DP+ è coinvolto anche un SM-DS, allora il server SM-DP+ esegue la procedura di eliminazione dell'evento dall'SM-DS.

18. Alla ricezione dell'acknowledgement da parte del server SM-DP+, l'LPA invoca la funzione *removeNotificationFromList* sull'eUICC.

19. L'eUICC cancella il risultato dell'installazione del profilo dalla propria memoria non volatile.

20. Se l'LPA aveva ricevuto rpmPending nel messaggio di risposta ad *authenticateClient*, allora dovrebbe iniziare a una sessione RSP aggiuntiva col server SM-DP+, settando l'operationType a RPM (Remote Profile Management).

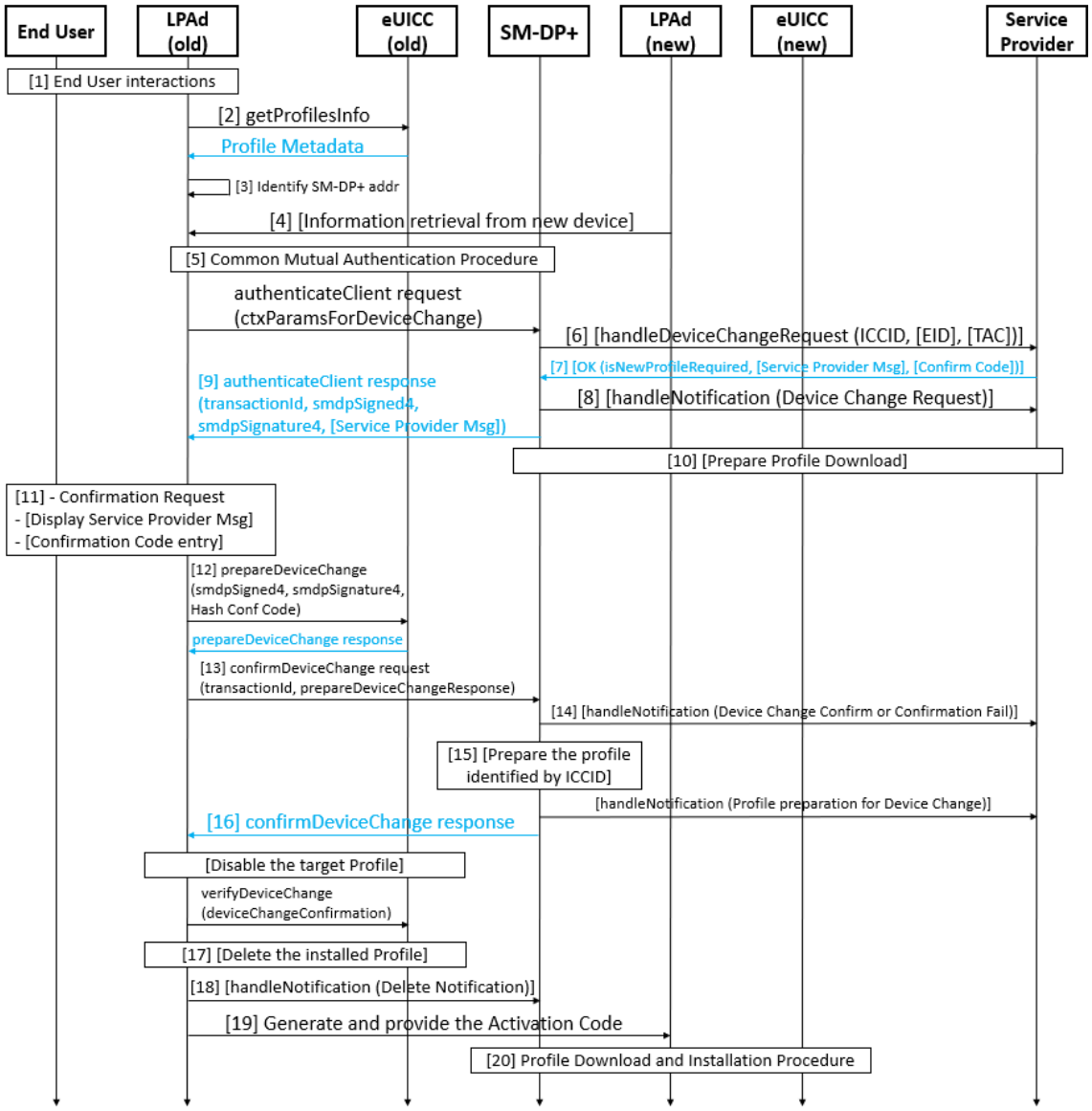


Figura 2.17: Sequence diagram che descrive il trasferimento di un profilo.

2.4.8 Cambio di dispositivo

La figura 2.17 ripresa da [3] illustra tutti i messaggi che l'end user, l'LPA installata nel vecchio dispositivo, l'eUICC presente nel vecchio dispositivo, il server SM-DP+, l'LPA installata nel nuovo dispositivo, l'eUICC presente nel nuovo dispositivo e l'operatore (qui indicato come Service Provider) si scambiano tra loro e le operazioni che queste sette entità svolgono durante la procedura di trasferimento di un profilo dovuto al cambio di dispositivo. Tutti i relativi dettagli [3] sono spiegati nelle sottosezioni successive.

Condizioni iniziali

- Il Service Provider ha fornito al server SM-DP+ la configurazione e altre informazioni rilevanti per il cambio di dispositivo. Tutti questi dati devono essere contenuti anche nel profilo all'interno del vecchio dispositivo.
- L'end user possiede sia un vecchio dispositivo contenente un profilo, sia un nuovo dispositivo.
- L'eUICC e l'LPA del vecchio dispositivo supportano il cambio di dispositivo.

Procedimento

1. L'end user dà inizio all'operazione di cambio di dispositivo a partire dall'LPA del vecchio dispositivo e seleziona il profilo da installare nel nuovo dispositivo.

2. L'LPA del vecchio dispositivo recupera DeviceChangeConfiguration dai metadati del profilo. Se DeviceChangeConfiguration indica requestToDp, allora si passa allo step (3); se invece DeviceChangeConfiguration indica usingStoredAc, allora si passa allo step (17).

3. L'LPA del vecchio dispositivo determina l'indirizzo del server SM-DP+ a partire da DeviceChangeConfiguration.

4. Se DeviceChangeConfiguration indica che è richiesto l'EID e/o la TAC (Type Allocation Code, che è un codice a 8 cifre univoco per il dispositivo) del nuovo dispositivo, allora l'LPA del vecchio dispositivo recupera l'EID e/o la TAC dal nuovo dispositivo.

5. L'LPA del vecchio dispositivo dà inizio alla procedura di Common Mutual Authentication definita nella sezione 2.4.5.

6. Se la funzione *handleDeviceChangeRequest* è configurata nel Service Provider, allora il server SM-DP+ la invoca passandovi come parametri l'ICCID (Integrated Circuit Card ID, che è l'identificativo del profilo) ed eventualmente l'EID e la TAC del nuovo dispositivo. D'altro canto, però, se il server SM-DP+ non supporta il cambio di dispositivo o il cambio di dispositivo non è consentito per il profilo, allora il server risponde con un messaggio di errore e la procedura termina.

7. Il Service Provider risponde al server SM-DP+ col booleano isNewProfileRequired e, opzionalmente, un messaggio di Service Provider per il cambio di dispositivo e un codice di conferma.

8. Se la funzione *handleNotification* è configurata nel Service Provider, il server SM-DP+ la invoca per notificare il Service Provider riguardo la richiesta del cambio di dispositivo.

9. Il server SM-DP+ restituisce all'LPA del vecchio dispositivo la risposta ad *authenticateClient* che comprende transactionId (l'identificativo della transazione), smdpSigned4 (informazioni del server SM-DP+), smdpSignature4 (signature di smdpSigned4) ed eventualmente il messaggio di Service Provider per il cambio di dispositivo.

10. Se il booleano isNewProfileRequired fornito dal Service Provider nello step (7) vale TRUE, allora il Service Provider esegue il Download Preparation Process (processo di preparazione del download) e opzionalmente il Subscription Activation Process (processo di attivazione della sottoscrizione al profilo).

11. L'LPA del vecchio dispositivo effettua una richiesta di conferma per il cambio di dispositivo. In particolare, se il Service Provider nello step (7) ha fornito il messaggio di Service Provider per il cambio di dispositivo, quest'ultimo viene presentato all'end user. Inoltre, se smdpSigned4 ha il campo ccRequiredFlag pari a TRUE, allora l'LPA del vecchio dispositivo chiede all'end user di inserire il codice di conferma fornito dal Service Provider nello step (7). Se l'end user non inserisce correttamente il codice di conferma entro un timeout prestabilito, si procede con la cancellazione della sessione (Common Cancel Session Procedure, 2.4.9).

12. L'LPA del vecchio dispositivo invoca la funzione *prepareDeviceChange* passandovi come parametri smdpSigned4, smdpSignature4 ed eventualmente l'hash del codice di conferma. Quest'ultimo viene calcolato come SHA256(SHA256(Confirmation Code) | transactionID).

13. L'LPA del vecchio dispositivo invoca la funzione *confirmDeviceChange* passandovi come parametri transactionId e prepareDeviceChangeResponse.

14. Se il booleano isNewProfileRequired fornito dal Service Provider nello step (7) vale TRUE, allora il server SM-DP+ invoca la funzione *handleNotification* per notificare il Service Provider riguardo l'esito della conferma del cambio di dispositivo da parte dell'end user. Se l'end user ha accettato il cambio di dispositivo, allora si passa allo step (15); altrimenti la procedura termina.

15. Se il booleano isNewProfileRequired vale FALSE, allora il server SM-DP+ prepara il profilo per il download e il Matching ID associato al profilo. Inoltre, se nello step (5) è stato fornito un EID, allora il server lo collega al download del profilo preparato. Infine, il server invoca la funzione *handleNotification* per notificare il Service Provider riguardo l'esito della preparazione del profilo.

16. Il server SM-DP+ restituisce all'LPA del vecchio dispositivo la risposta a *confirmDeviceChange* che comprende l'esito del cambio di dispositivo. Se tale risposta contiene encryptedDeviceChangeData, l'LPA del vecchio dispositivo disabilita il profilo target. Dopodiché l'LPA verifica la signature del server SM-DP+ invocando la funzione *verifyDeviceChange*.

17. Se previsto, l'LPA del vecchio dispositivo elimina il profilo target dall'eUICC del vecchio dispositivo. Inoltre, se DeviceChangeConfiguration indica requestToDp e il server SM-DP+ nello step (16) ha indicato di supportare la recovery dei profili eliminati, allora l'LPA del vecchio dispositivo dovrebbe memorizzare alcune informazioni del profilo eliminato, come l'ICCID. Se invece l'eliminazione del profilo non è richiesta, si passa direttamente allo step (19).

18. L'LPA del vecchio dispositivo invia al server SM-DP+ la notifica di eliminazione del profilo. Se l'invio della notifica fallisce, la procedura termina.

19. L'LPA del vecchio dispositivo genera il codice di attivazione (Activation Code) e lo fornisce all'LPA del nuovo dispositivo. Inoltre, dovrebbe fornire al nuovo dispositivo lo stato attuale del profilo in modo tale da consentire all'LPA del nuovo dispositivo di ripristinare correttamente tale stato.

20. Il profilo viene scaricato nel nuovo dispositivo a partire dal server SM-DP+ mediante la procedura di download e installazione del profilo, che è stata definita nella sezione 2.4.6.

2.4.9 Dettagli sulla Common Cancel Session Procedure

La figura 2.18 ripresa da [3] illustra tutti i messaggi che l'operatore, il server SM-DP+, l'LPA e l'eUICC si scambiano tra loro e le operazioni che queste quattro entità svolgono durante la Common Cancel Session Procedure, che è una particolare procedura avviata dall'LPA con lo scopo di interrompere la sessione RSP corrente a seguito di una particolare condizione di errore riscontrata dall'LPA stessa. Tutti i dettagli sul procedimento [3] sono spiegati all'interno della sottosezione successiva.

Procedimento

1. L'LPA invoca la funzione *cancelSession* sull'eUICC, passandovi come parametri l'ID di transazione e la reason, ovvero il motivo per cui è stata avviata la Common Cancel Session Procedure.

2. L'eUICC esegue le seguenti operazioni:

   - Generare la struttura dati euiccCancelSessionSigned che comprende l'ID di transazione e la reason, entrambi forniti dall'LPA.
   - Calcolare euiccCancelSessionSignature a partire da euiccCancelSessionSigned utilizzando la chiave privata SK.EUICC.SIG associata all'euiccCiPKIdToBeUsed ricevuto all'inizio della procedura di Common Mutual Authentication.

3. L'eUICC restituisce euiccCancelSessionSigned ed euiccCancelSessionSignature all'LPA. Se la reason è sessionAborted, allora l'LPA dovrà ignorare tale messaggio di risposta e interrompere la procedura.

4. Se la connessione HTTPS tra LPA e server SM-DP+ non è più attiva, allora l'LPA stabilisce una nuova connessione HTTPS col server come descritto nella procedura di Common Mutual Authentication.

5. L'LPA invoca la funzione *cancelSession* anche sul server SM-DP+, passandovi come parametri l'ID della transazione, euiccCancelSessionSigned ed euiccCancelSessionSignature.

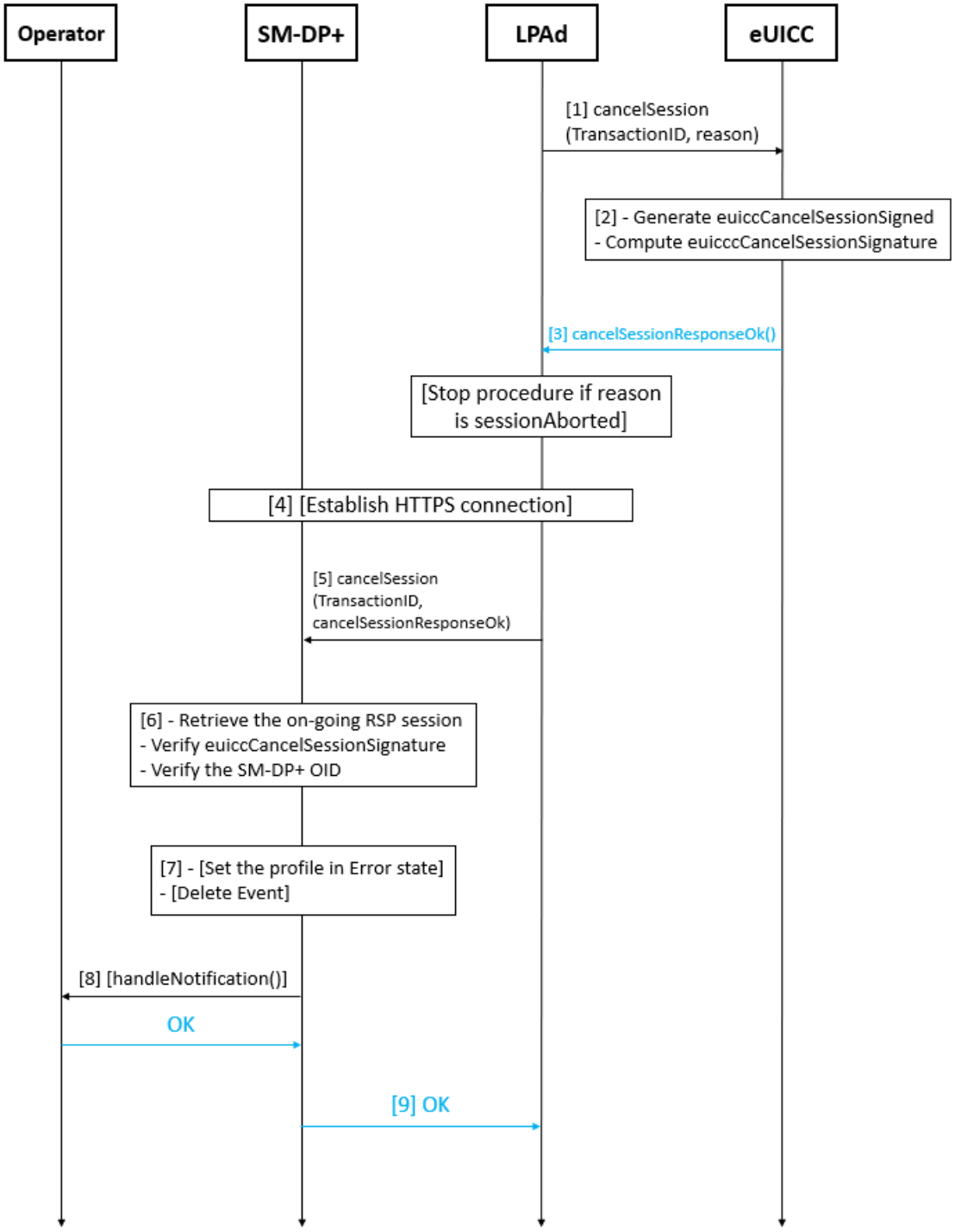


Figura 2.18: Sequence diagram che descrive la Common Cancel Session Procedure.

6. Il server SM-DP+ esegue le seguenti operazioni:

   - Recuperare la sessione RSP corrente identificata dall'ID di transazione appena ricevuto.
   - Verificare euiccCancelSessionSignature.
   - Verificare che l'OID ricevuto sia uguale a quello contenuto in CERT.DPauth.SIG durante la procedura di Common Mutual Authentication.

   Se anche solo uno di questi controlli non va a buon fine, o se l'ID di transazione ricevuto è ignoto, il server SM-DP+ deve restituire uno stato di errore e la procedura di Common Cancel Session deve essere interrotta. Inoltre, se la reason è postponed o timeout, allora il server deve mantenere l'ordinazione di download del profilo disponibile per ulteriori tentativi e passare allo step (9); in caso contrario, si procede con lo step successivo.

7. Il server SM-DP+ imposta il profilo allo stato Error. Inoltre, se un SM-DS è coinvolto nella sessione RSP, allora il server deve anche eliminare l'evento corrispondente dall'SM-DS.

8. Il server SM-DP+ può opzionalmente notificare l'operatore del fallimento del download del profilo mediante la funzione *handleNotification*.

9. Il server SM-DP+ restituisce un acknowledgement all'LPA. A questo punto, la procedura termina.

2.5 Policy dei profili

Come accennato nella sotto-sezione 2.4.6, durante la procedura di download e installazione del profilo, è necessario anche verificare le policy del profilo per stabilire se quest'ultimo può essere installato all'interno dell'eUICC. Di conseguenza, può tornare utile illustrare nel dettaglio il meccanismo di gestione delle policy dei profili previsto in GSMA [3].
Le policy dei profili possono essere descritte dalle cosiddette **Profile Policy Rules** (PPR), che sono delle regole che descrivono cosa è possibile e cosa non è possibile fare con ciascun profilo. Attualmente possono essere definite due PPR in particolare:

- PPR1 - 'Disabling of this Profile is not allowed'
- PPR2 - 'Deletion of this Profile is not allowed'

2.5.1 Rules Authorisation Table

La Rules Authorisation Table (RAT) è una struttura dati che tiene traccia dell'insieme delle PPR accettabili per ciascun profilo [3]. Da un punto di vista strutturale, la RAT mantiene una o più entry, ciascuna delle quali è relativa a una **Profile Policy Authorisation Rule** (PPAR). Una PPAR a sua volta è composta dalle seguenti informazioni (che costituiscono le colonne della tabella RAT):

- **Profile Policy Rule Identifier**: identifica una o più PPR a cui questa PPAR fa riferimento.
- **Allowed Operators**: è una lista di operatori che possono utilizzare le PPR identificate dalla colonna Profile Policy Rule Identifier.
- **End User Consent Required**: indica se, a seguito delle PPR identificate dalla colonna Profile Policy Rule Identifier, il profilo ha bisogno di un consenso esplicito dell'end user per essere installato.

In definitiva, la possibilità per un profilo di essere installato o meno dipende dalle eventuali PPR definite per quel profilo e dalle PPAR appartenenti alla RAT. In particolare:

- Se il profilo non contiene alcuna PPR, allora può essere installato senza problemi.
- In caso contrario, è necessario verificare se nella RAT esistono delle PPAR che facciano riferimento sia alle PPR contenute nel profilo, sia all'operatore che ha rilasciato il profilo:

- Se tutte le PPR vengono coperte (da una o più PPAR) nell'ambito dell'operatore che ha rilasciato il profilo, allora si controlla la colonna End User Consent Required della RAT: i PPR per cui End User Consent Required = true hanno bisogno del consenso dell'end user affinché il relativo profilo possa essere installato nell'eUICC, e viceversa.
- Se invece non tutte le PPR vengono coperte, non c'è la possibilità di installare il profilo nell'eUICC.

Si noti che il placeholder '*' per Allowed Operators indica "tutti gli operatori". Se esistono più PPAR valide per una medesima PPR e un medesimo operatore, fa fede quella che compare per prima all'interno della RAT.
Il componente dell'eUICC che si occupa di verificare se un profilo contenente una o più PPR è autorizzato dalla RAT o meno è detto **Profile Policy Enabler** (PPE) [3].

Primo esempio di RAT

La tabella 2.5, tratta da [3], raffigura un primo esempio di RAT.

Tabella 2.5: Primo esempio di RAT.

Profile Policy Rule Identifier	Allowed Operators	End User Consent Required
PPR1	OP-A	false
PPR2	OP-B	false
PPR1, PPR2	*	true

La RAT rappresentata in 2.5 descrive la seguente situazione:

- L'operatore OP-A può usare PPR1 senza il consenso dell'end user e PPR2 col consenso dell'end user.
- L'operatore OP-B può usare PPR1 col consenso dell'end user e PPR2 senza il consenso dell'end user.
- Tutti gli altri operatori possono usare sia PPR1 che PPR2 col consenso dell'end user.

Secondo esempio di RAT

La tabella 2.6, tratta da [3], raffigura un secondo esempio di RAT.

Tabella 2.6: Secondo esempio di RAT.

Profile Policy Rule Identifier	Allowed Operators	End User Consent Required
PPR1, PPR2	*	true

Secondo la RAT rappresentata in 2.6, tutti gli operatori possono usare PPR1 e PPR2 col consenso dell'end user.

Terzo esempio di RAT

Un terzo esempio classico è dato dalla tabella RAT vuota, secondo cui nessun operatore può usare alcuna PPR.

Capitolo 3

Stato dell'arte

3.1 Uno studio condotto nel 2022

Per quanto concerne la sicurezza dell'eSIM e del protocollo RSP, non sono ancora stati condotti numerosi studi e, al momento, risulta difficoltoso trovare degli articoli a riguardo in rete. Ciò è dovuto al fatto che questo mondo è del tutto nuovo e i telefoni che supportano le eSIM sono tuttora in fase di diffusione. Di conseguenza, in questo capitolo ci si limita a illustrare un particolare studio che, al momento, è di gran lunga il più importante a essere stato condotto nell'ambito dell'eSIM: si tratta di un'analisi portata a termine nel 2022 da quattro ricercatori dell'università Aalto in Finlandia (Abu Shohel Ahmed, Aleksi Peltonen, Mohit Sethi e Tuomas Aura) [4].
Il fine ultimo dello studio è quello di effettuare un'analisi di sicurezza del protocollo, a partire dalla procedura di Common Mutual Authentication fino al download e l'installazione del profilo, definendo 15 obiettivi di sicurezza, considerando due varianti del protocollo (default server approach e Activation Code approach) e lavorando su 11 scenari differenti, ciascuno dei quali viene analizzato sia con l'uso del tunnel TLS tra l'applicazione LPA e il server SM-DP+, sia senza l'uso del tunnel TLS. L'analisi è stata svolta con l'aiuto di ProVerif, un verificatore automatico di protocolli crittografici che lavora nel modello formale; in particolare, questo tool permette di dimostrare la validità di proprietà crittografiche come la confidenzialità, l'autenticazione, l'indistinguibilità tra più segreti e l'equivalenza di più processi client che differiscono esclusivamente nella sessione in cui comunicano col server [9].
La tabella 3.1, tratta da [4], elenca gli 11 scenari su cui si è lavorato. È chiaro che gli ultimi due sono applicabili soltanto nella variante del protocollo in cui è previsto l'uso dell'Activation Code (Activation Code approach).

Tabella 3.1: Gli 11 scenari considerati nello studio.

N°	**Scenario**	**Descrizione**
1	Base-case	Tutti i partecipanti sono onesti e solo la rete tra l'LPA e il server SM-DP+ è untrusted.
2	Server	L'attaccante possiede tutte le chiavi private del server SM-DP+.
3	eUICC	L'attaccante possiede la chiave privata dell'eUICC del telefono della vittima.
4	LPA	L'attaccante è l'utente che controlla l'LPA.
5	2nd server	Il server SM-DP+ target è onesto ma esiste un secondo server SM-DP+ compromesso.
6	2nd eUICC	L'attaccante possiede la chiave privata dell'eUICC del proprio telefono.
7	2nd MNO	L'operatore target è onesto ma esiste un secondo operatore compromesso.
8	Order-user	L'attaccante impersona la vittima quando esegue l'ordine di un profilo.
9	Order-eUICC	L'attaccante richiede un profilo per l'eUICC della vittima (senza impersonarla).
10	AC leak	L'attaccante ottiene l'Activation Code associato a un profilo da scaricare.
11	AC spoof	L'attaccante manipola l'LPA affinché utilizzi un Activation Code errato.

D'altra parte, i 15 obiettivi di sicurezza, il cui conseguimento è stato testato negli scenari di cui sopra (con e senza tunnel TLS) sono schematizzati all'interno della tabella 3.2 tratta da [4].

Tabella 3.2: I 15 obiettivi di sicurezza considerati nello studio.

ID	Tipo	Descrizione
A	Autenticazione	L'eUICC può completare l'autenticazione del server SM-DP+ solo dopo che entrambi gli end-point hanno avviato la procedura.
B	Autenticazione	Il server SM-DP+ può completare l'autenticazione dell'eUICC solo dopo aver avviato la procedura e dopo essere stato autenticato dall'eUICC.
B'	Autenticazione	Vi è una corrispondenza tra il completamento dell'autenticazione dell'eUICC, il fatto che l'end user sia l'effettivo proprietario dell'eUICC, la richiesta di un profilo all'operatore da parte dell'end user e la richiesta di ordinazione di quel profilo al server SM-DP+ da parte dell'operatore.
C	Autenticazione	L'eUICC può accettare il server SM-DP+ come entità da cui effettuare il download del profilo solo se la procedura di mutua autenticazione è stata completata con successo e se la GSMA ha autorizzato il server a rilasciare profili.
D	Autenticazione	Il server SM-DP+ può avviare il key agreement e la consegna del profilo solo se l'eUICC è stato autenticato e lo ha accettato come entità da cui effettuare il download del profilo.
E	Autenticazione	L'eUICC può accettare le chiavi di sessione solo se quest'ultimo ha avviato il key agreement.
F	Autenticazione	L'eUICC può scaricare il profilo dal server SM-DP+ solo se quest'ultimo ha avviato la consegna del profilo.
G	Autenticazione	Vi è una corrispondenza tra l'accettazione della notifica di download del profilo da parte del server SM-DP+, il fatto che l'end user sia l'effettivo proprietario dell'eUICC, la richiesta di un profilo all'operatore da parte dell'end user e la richiesta di ordinazione di quel profilo al server SM-DP+ da parte dell'operatore.
I	Autenticazione	L'eUICC può scaricare il profilo dal server SM-DP+ solo dopo che entrambi gli end-point hanno avviato la procedura di Common Mutual Authentication.
J	Autenticazione	Vi è una corrispondenza tra il download del profilo all'interno dell'eUICC, il fatto che l'end user sia l'effettivo proprietario dell'eUICC, la richiesta di un profilo all'operatore da parte dell'end user e la richiesta di ordinazione di quel profilo al server SM-DP+ da parte dell'operatore.
K	Autenticazione	Vi è una corrispondenza tra il rilascio del profilo da parte del server SM-DP+, il fatto che l'end user sia l'effettivo proprietario dell'eUICC, la richiesta di un profilo all'operatore da parte dell'end user e la richiesta di ordinazione di quel profilo al server SM-DP+ da parte dell'operatore.
W	Confidenzialità	L'attaccante non può accedere alle chiavi di sessione una volta che sono state accettate dal server SM-DP+.
X	Confidenzialità	L'attaccante non può accedere alle chiavi di sessione una volta che sono state accettate dall'eUICC.
Y	Confidenzialità	L'attaccante non può accedere al profilo decrittato una volta che è stato rilasciato dal server SM-DP+.
Z	Confidenzialità	L'attaccante non può accedere al profilo decrittato una volta che è stato scaricato nell'eUICC.

I risultati dei test eseguiti dai quattro ricercatori dell'università Aalto sono illustrati in figura 3.1 tratta da [4]. La spunta indica che il controllo di sicurezza è passato, il cerchio indica che il controllo di sicurezza è passato solo nel caso in cui è presente il tunnel TLS, la spunta tra parentesi indica che il tool ProVerif ha terminato senza un risultato conclusivo, mentre la X indica che il controllo di sicurezza non è passato. Tutti i simboli colorati di rosso nelle due tabelle sono relativi a risultati inaspettati o comunque degni di nota: di fatto, i casi in cui l'attaccante possiede le chiavi private del server SM-DP+ o dell'eUICC sono gli unici in cui ci si aspettava con un elevato livello di confidenza che l'intero protocollo fosse compromesso.

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.

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.

Figura 3.1: Risultati dell'approccio default server (in alto) e Activation Code (in basso).

Sulla base delle considerazioni riportate in [4], è possibile riassumere le vulnerabilità emerse nel seguente elenco.

1. Se si usa un server SM-DP+ S' compromesso a fianco del server SM-DP+ S coinvolto nella comunicazione RSP e non vi è un tunnel TLS tra l'LPA e il server S, allora l'attaccante può deviare la comunicazione verso il server S'. In tal modo, l'eUICC accetta anche i profili falsi generati dal server compromesso (S'). La vulnerabilità potrebbe essere dovuta al fatto che non compare alcun identificatore del server SM-DP+ (OID) all'interno dei messaggi che contengono i profili.

2. Se l'attaccante possiede l'eUICC U, viene compromesso un secondo eUICC U', non vi è un tunnel TLS tra l'LPA e il server SM-DP+ e si ricorre all'approccio Activation Code, allora l'attaccante può rubare l'Activation Code dalla vittima che possiede U, firmarlo con la chiave privata dell'eUICC compromesso (U') e inviare il messaggio così generato al server SM-DP+. In tal modo, il server risponde a U' (anziché a U) con il nuovo profilo. La vulnerabilità potrebbe essere dovuta al fatto che all'interno del server SM-DP+ non viene registrato l'identificatore dell'eUICC target (EID) nel momento in cui viene ordinato un nuovo profilo.

3. L'attaccante può scaricare un profilo in modo illegittimo sul proprio eUICC in uno dei seguenti due modi: il primo modo consiste nell'impersonare un utente legittimo deviando il download sul proprio eUICC; il secondo modo consiste nel ricostruire l'Activation Code a partire da leak di informazioni lasciati dall'utente vittima, dall'operatore oppure da una LPA modificata. La vulnerabilità potrebbe essere dovuta al fatto che, quando un utente richiede il download di un profilo, l'operatore non verifica in modo soddisfacente la sua identità: di fatto, non vi è alcun legame crittografico tra il profilo e l'end user che ne richiede il download, bensì è sufficiente esibire il codice di attivazione per dimostrare di essere il proprietario legittimo del profilo target.

4. L'attaccante può scaricare un profilo nell'eUICC vittima (U) con la propria sottoscrizione (e quindi in modo illegittimo) in uno dei seguenti due modi: il primo modo consiste nel conoscere l'identificatore (l'EID) dell'eUICC U e richiedere il download del profilo per conto

di U; il secondo modo consiste nell'iniettare un Activation Code per conto di U, ad esempio tramite una LPA modificata.

5. Se si hanno due server SM-DP+ (S, S'), tra cui S' è compromesso, e non vi è un tunnel TLS tra l'LPA e il server S, allora l'attaccante può far sì che il server S impersoni il server S': così, durante la procedura di Common Mutual Authentication, S crederà di aver autenticato l'eUICC e l'eUICC crederà di aver autenticato S'. La vulnerabilità potrebbe essere dovuta al fatto che l'OID non viene incluso nei messaggi iniziali della Common Mutual Authentication.

6. Se si hanno due eUICC (U, U'), tra cui U' è compromesso, e non vi è un tunnel TLS tra l'LPA e il server SM-DP+, allora l'attaccante può intercettare l'autenticazione tra l'eUICC U e il server: così, alla fine della procedura di Common Mutual Authentication, U crederà di aver autenticato il server e il server crederà di aver autenticato U'. La vulnerabilità potrebbe essere dovuta al fatto che l'EID non viene incluso nei messaggi iniziali della Common Mutual Authentication.

7. Se non vi è un tunnel TLS tra l'LPA e il server SM-DP+ e si ricorre all'approccio Activation Code, allora, anche senza componenti compromessi, potrebbero rimanere dei leak di informazioni riguardanti l'Activation Code che un eventuale attaccante potrebbe sfruttare a proprio vantaggio.

Tutti i punti sopra elencati, eccetto il terzo e il quarto, fanno riferimento a vulnerabilità che occorrerebbero nel caso in cui non si avesse un tunnel TLS tra l'applicazione LPA in utilizzo e il server SM-DP+ contattato dall'LPA. Tuttavia, si noti anche che le vulnerabilità illustrate possono essere sfruttate solo in situazioni molto particolari dove si hanno malfunzionamenti, componenti compromessi o requisiti imposti dalle specifiche di GSMA non rispettati (come l'assenza del tunnel TLS tra l'LPA e il server SM-DP+). Di conseguenza, rimane impossibile assumere che il protocollo RSP così definito non sia formalmente corretto.

3.2 Altri studi in corso

Tuttora vengono portati avanti degli studi sulla sicurezza dell'eSIM e si tengono delle conferenze a rigurardo. Un esempio è l'OffensiveCon, il cui talk sull'eSIM tenutosi nel maggio 2023 è disponibile su YouTube: https://www.youtube.com/watch?v=5oecn43xsDg.
Qui è stato discusso su come deployare un profilo custom definito manualmente, come installare applicazioni custom nell'eUICC e come sfruttare qualche vulnerabilià dell'eUICC (e.g. utilizzo di un toolkit per catturare informazioni, esecuzione di comandi sulla macchina vittima mediante una connessione custom). Comunque sia, di nuovo, non sono stati riscontrati problemi o vulnerabilità nella definizione formale del protocollo RSP.

Capitolo 4

Implementazione dei simulatori

4.1 Obiettivo prefissato

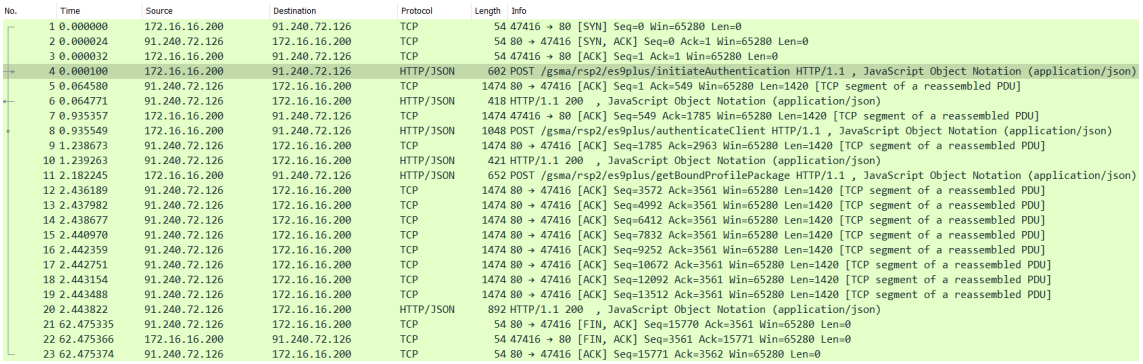
Lo scopo che è stato inizialmente definito per la fase operativa del presente lavoro è quello di costruire un simulatore sia per il client (composto da eUICC e LPA) sia per il server SM-DP+, in modo tale da:

1. creare un ecosistema simulato che funzioni secondo le specifiche di GSMA;
2. far comunicare ciascun simulatore rispettivamente con server SM-DP+ reali e con eUICC + LPA reali, con l'idea di verificare come questi rispondono se vengono sollecitati in modo diverso. In tal modo, con test opportuni, è possibile mettere alla prova le entità reali per stabilire se:
   - è possibile ricavare qualche informazione in più o qualche vulnerabilità;
   - implementano ogni controllo e ogni messaggio secondo le specifiche di GSMA oppure accettano un qualche pacchetto non conforme (e.g. contenente una signature o un certificato non valido).

Per aiutarsi con l'implementazione dei simulatori, è utile avere a disposizione una traccia che rappresenti un reale scambio di messaggi tra un'applicazione LPA e un server SM-DP+. Questa, assieme alla documentazione, può rappresentare il punto di riferimento a cui affidarsi per implementare nel modo più preciso e fedele possibile lo scambio di messaggi e i check effettuati dalle due controparti.

4.2 Ottenimento delle catture su Wireshark

È stato effettuato il download di un nuovo profilo (i.e. piano tariffario) reale di Very Mobile su un apposito dispositivo mobile che supporta le eSIM. Lo scambio di messaggi che ha coinvolto l'LPA del dispositivo mobile e il server SM-DP+ di Very Mobile è stato salvato su una traccia PCAP con l'ausilio di un'apposita applicazione di cattura di pacchetti di rete. Com'è stato già detto, il traffico tra LPA e server è traffico HTTPS per cui, di base, la traccia che si ottiene mostra dei pacchetti protetti dal protocollo TLS, offuscando il contenuto dei messaggi previsti dalle specifiche di GSMA di cui si è parlato nel capitolo precedente.
Per questo motivo, durante la fase di cattura, è stato sfruttato un proxy relay che ha il compito di interporsi nella comunicazione tra l'LPA e il server SM-DP+. Ciò ha permesso di creare due sessioni TLS distinte: una tra il telefono e il proxy relay e l'altra tra il proxy relay e il server SM-DP+. Nel mezzo, il tool di proxy è stato in grado di decrittare il traffico ricevuto da un end-point per poi cifrarlo nuovamente verso l'altro end-point, salvando il traffico decrittato su una traccia (più precisamente tre tracce, ma ne è stata presa in considerazione una in particolare) che, quindi, contiene uno scambio di pacchetti dei quali è possibile accedere al payload, ovvero al contenuto dei messaggi dati dal protocollo RSP. Per far ciò, è stato necessario predisporre l'utilizzo del dispositivo mobile coi privilegi di root, per cui è buona norma non coinvolgere il proprio telefono personale nell'esperimento.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.200	91.240.72.126	TCP	54	47416 → 80 [SYN] Seq=0 Win=65280 Len=0
2	0.000024	91.240.72.126	172.16.16.200	TCP	54	80 → 47416 [SYN, ACK] Seq=0 Ack=1 Win=65280 Len=0
3	0.000032	172.16.16.200	91.240.72.126	TCP	54	47416 → 80 [ACK] Seq=1 Ack=1 Win=65280 Len=0
4	0.000100	172.16.16.200	91.240.72.126	HTTP/JSON	602	POST /gsma/rsp2/es9plus/initiateAuthentication HTTP/1.1 , JavaScript Object Notation (application/json)
5	0.064580	91.240.72.126	172.16.16.200	TCP	1474	80 → 47416 [ACK] Seq=1 Ack=549 Win=65280 Len=1420 [TCP segment of a reassembled PDU]
6	0.064771	91.240.72.126	172.16.16.200	HTTP/JSON	418	HTTP/1.1 200 , JavaScript Object Notation (application/json)
7	0.935357	172.16.16.200	91.240.72.126	TCP	1474	47416 → 80 [ACK] Seq=549 Ack=1785 Win=65280 Len=1420 [TCP segment of a reassembled PDU]
8	0.935549	172.16.16.200	91.240.72.126	HTTP/JSON	1048	POST /gsma/rsp2/es9plus/authenticateClient HTTP/1.1 , JavaScript Object Notation (application/json)
9	1.238673	91.240.72.126	172.16.16.200	TCP	1474	80 → 47416 [ACK] Seq=1785 Ack=2963 Win=65280 Len=1420 [TCP segment of a reassembled PDU]
10	1.239263	91.240.72.126	172.16.16.200	HTTP/JSON	421	HTTP/1.1 200 , JavaScript Object Notation (application/json)
11	2.182245	172.16.16.200	91.240.72.126	HTTP/JSON	652	POST /gsma/rsp2/es9plus/getBoundProfilePackage HTTP/1.1 , JavaScript Object Notation (application/json)
12	2.436189	91.240.72.126	172.16.16.200	TCP	1474	80 → 47416 [ACK] Seq=3572 Ack=3561 Win=65280 Len=1420 [TCP segment of a reassembled PDU]
13	2.437982	91.240.72.126	172.16.16.200	TCP	1474	80 → 47416 [ACK] Seq=4992 Ack=3561 Win=65280 Len=1420 [TCP segment of a reassembled PDU]
14	2.438677	91.240.72.126	172.16.16.200	TCP	1474	80 → 47416 [ACK] Seq=6412 Ack=3561 Win=65280 Len=1420 [TCP segment of a reassembled PDU]
15	2.440970	91.240.72.126	172.16.16.200	TCP	1474	80 → 47416 [ACK] Seq=7832 Ack=3561 Win=65280 Len=1420 [TCP segment of a reassembled PDU]
16	2.442359	91.240.72.126	172.16.16.200	TCP	1474	80 → 47416 [ACK] Seq=9252 Ack=3561 Win=65280 Len=1420 [TCP segment of a reassembled PDU]
17	2.442751	91.240.72.126	172.16.16.200	TCP	1474	80 → 47416 [ACK] Seq=10672 Ack=3561 Win=65280 Len=1420 [TCP segment of a reassembled PDU]
18	2.443154	91.240.72.126	172.16.16.200	TCP	1474	80 → 47416 [ACK] Seq=12092 Ack=3561 Win=65280 Len=1420 [TCP segment of a reassembled PDU]
19	2.443488	91.240.72.126	172.16.16.200	TCP	1474	80 → 47416 [ACK] Seq=13512 Ack=3561 Win=65280 Len=1420 [TCP segment of a reassembled PDU]
20	2.443822	91.240.72.126	172.16.16.200	HTTP/JSON	892	HTTP/1.1 200 , JavaScript Object Notation (application/json)
21	62.475335	91.240.72.126	172.16.16.200	TCP	54	80 → 47416 [FIN, ACK] Seq=15770 Ack=3561 Win=65280 Len=0
22	62.475366	172.16.16.200	91.240.72.126	TCP	54	47416 → 80 [FIN, ACK] Seq=3561 Ack=15771 Win=65280 Len=0
23	62.475374	91.240.72.126	172.16.16.200	TCP	54	80 → 47416 [ACK] Seq=15771 Ack=3562 Win=65280 Len=0

Figura 4.1: Lista di pacchetti catturati.

È dunque possibile osservare che sia l'applicazione LPA utilizzata sia il server SM-DP+ abbiano accettato di comunicare direttamente con un'entità (il proxy relay) il cui certificato TLS non è firmato da alcuna Certification Authority riconducibile a GSMA. Ciò è in linea con la trattazione già svolta sulle catene di certificati all'interno della sezione 2.3.2.
Il file PCAP contenente la traccia così ottenuta è stato analizzato tramite Wireshark, che è un analizzatore di rete, e consente sia di mostrare tutti i dettagli relativi a uno scambio di pacchetti già salvato, sia di catturare a sua volta una traccia, almeno nel caso in cui il traffico di interesse coinvolga il sistema su cui il tool è installato.

4.3 Analisi dei messaggi catturati

Wireshark risulta fondamentale per individuare il formato esatto di ciascun pacchetto catturato all'interno della traccia. Innanzitutto è necessario stabilire quali sono i messaggi presenti all'interno del file PCAP per stabilire quale corrispondenza ci sia tra la comunicazione di riferimento tra l'LPA del dispositivo mobile e il server SM-DP+ di Very Mobile. In figura 4.1 è rappresentata la schermata di Wireshark che mostra la successione dei messaggi che sono stati scambiati durante la cattura.

Com'è possibile vedere, in mezzo a dei normalissimi pacchetti TCP, spiccano alcuni pacchetti contrassegnati dal protocollo HTTP/JSON, che sono relativi proprio ai messaggi che compongono la comunicazione tra l'LPA e il server SM-DP+. In particolare, ve ne sono tre inviati al server da parte del client, mentre gli altri tre non sono altro che le rispettive risposte del server. I tre messaggi inviati dal client sono elencati qui di seguito:

1. **initiateAuthentication**, che corrisponde al primo messaggio inviato al server SM-DP+ da parte dell'LPA allo step (6) della procedura di Common Mutual Authentication (sezione 2.4.5).

2. **authenticateClient**, che corrisponde al secondo messaggio inviato al server SM-DP+ da parte dell'LPA allo step (15) della procedura di Common Mutual Authentication (sezione 2.4.5).

3. **getBoundProfilePackage**, che corrisponde al primo messaggio inviato al server SM-DP+ da parte dell'LPA allo step (5) della sotto-procedura di Download Confirmation (sezione 2.4.7).

Le risposte del server all'interno del file PCAP, nell'ordine, sono relative ai seguenti altri messaggi:

1. **initiateAuthenticationResponse**: corrisponde al messaggio inviato all'LPA da parte del server SM-DP+ allo step (9) della procedura di Common Mutual Authentication (sezione 2.4.5).

2. **authenticateClientResponse**, che corrisponde all'unico messaggio inviato all'LPA da parte del server SM-DP+ allo step (6) della procedura di download e installazione dei profili (sezione 2.4.6).

3. **getBoundProfilePackageResponse**, che corrisponde al messaggio inviato all'LPA da parte del server SM-DP+ allo step (11) della sotto-procedura di Download Confirmation (sezione 2.4.7).

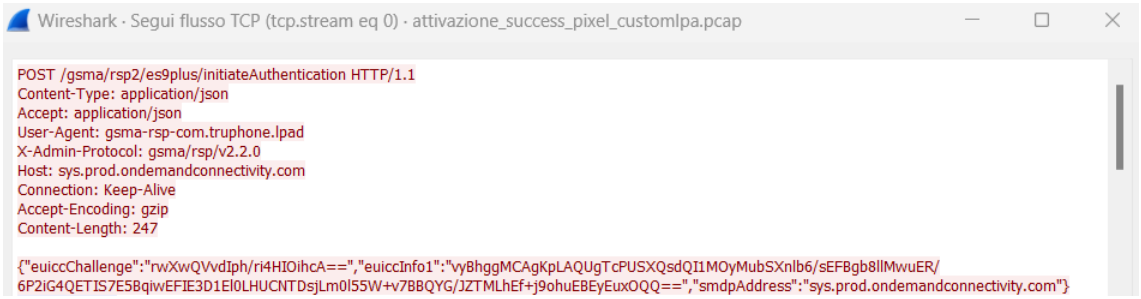


Figura 4.2: Messaggio di initiateAuthentication visualizzato su Wireshark.

In realtà, le altre due tracce che sono state prese, oltre a questi sei pacchetti HTTP/JSON, riportano ulteriori messaggi contrassegnati dal medesimo protocollo, e sembrano corrispondere tutti a un **handleNotification** inviato al server SM-DP+ da parte dell'LPA in modo periodico: tale handleNotification è relativo all'ultimo messaggio ricevuto dal server allo step (14) della sotto-procedura di Download Confirmation (sezione 2.4.7).
In definitiva, è possibile apprezzare un matching perfetto tra i pacchetti catturati nelle tracce PCAP e i messaggi previsti dalla documentazione di GSMA [3] nel momento in cui si considerano in sequenza la procedura di Common Mutual Authentication e la procedura di download e installazione dei profili (inclusa la sotto-procedura di Download Confirmation).
Per comprendere appieno la struttura e il formato dei messaggi scambiati tra LPA e server SM-DP+, è certamente sufficiente effettuare un'analisi dettagliata del primo messaggio inviato dal client durante la procedura di Common Mutual Authentication (initiateAuthentication) e della relativa risposta del server (initiateAuthenticationResponse). Tuttavia, nella presente trattazione si intende effettuare un'analisi completa di tutti i pacchetti scambiati dalle controparti con lo scopo di acquisire più informazioni possibili sui messaggi ed, eventualmente, sul server SM-DP+ di Very Mobile e sul profilo installato all'interno dell'eUICC del dispositivo mobile.

4.3.1 Analisi di initiateAuthentication

Da Wireshark è possibile osservare il contenuto del messaggio initiateAuthentication all'interno della comunicazione reale tra LPA del dispositivo mobile e server SM-DP+ di Very Mobile. La figura 4.2 ne mostra un estratto.
L'header HTTP riporta alcune informazioni interessanti:

- il metodo HTTP utilizzato dall'LPA è POST;
- la versione HTTP utilizzata è la 1.1;
- il protocollo seguito per l'interazione tra LPA e server è la versione 2.2.0 di GSMA-RSP;
- l'hostname del server SM-DP+ di Very Mobile è sys.prod.ondemandconnectivity.com;
- l'LPA in questione accetta anche messaggi di risposta il cui payload è stato compresso tramite gzip.

Per quanto invece concerne il payload del messaggio, è possibile osservare che ha una struttura JSON con tre coppie chiave-valore. In particolare, i primi due campi, che sono euiccChallenge ed euiccInfo1, assumono dei valori rappresentati nella codifica **base64**, mentre il terzo e ultimo campo, ovvero smdpAddress, riporta in chiaro l'hostname del server SM-DP+ contattato.

Codifica base64

È un sistema di codifica che permette di convertire una sequenza di byte in una stringa in formato ASCII, dove i possibili caratteri diversi che compongono la stringa sono 64 [10]. Lo scopo della codifica è principalmente quello di rendere stampabile qualunque tipo di informazione.
L'idea alla base è molto semplice: si prende la sequenza di byte di partenza, la si suddivide in gruppi di 6 bit, e ciascun gruppo di 6 bit viene tradotto nel corrispondente carattere ASCII secondo la convenzione riportata nella figura 4.3 tratta da [10].

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Figura 4.3: Tabella di conversione della codifica base64.

La tabella di conversione della codifica base64 riporta anche un sessantacinquesimo carattere ('='), che svolge la funzione di padding. Infatti, è possibile assumere che ogni sequenza di byte data in input all'algoritmo di codifica base64 abbia una lunghezza multipla di 8 bit, ma non è detto che abbia una lunghezza multipla anche di 6 bit; di fatto, avere una linghezza multipla di 6 bit significherebbe avere una lunghezza multipla di 24 bit. Si hanno dunque tre casistiche [10]:

1. La sequenza di byte di input ha una lunghezza multipla di 24 bit: in tal caso, non è necessario avere particolari accortezze e, in particolare, non è previsto alcun padding.

2. La sequenza di byte di input ha una lunghezza tale per cui alla fine avanzano 8 bit: in tal caso, vengono aggiunti 4 bit di padding pari a 0, vengono convertiti i 12 bit così ottenuti nei corrispettivi 2 caratteri ASCII e, in coda alla stringa di output, vengono aggiunti 2 caratteri di uguale ('==').

3. La sequenza di byte di input ha una lunghezza tale per cui alla fine avanzano 16 bit: in tal caso, vengono aggiunti 2 bit di padding pari a 0, vengono convertiti i 18 bit così ottenuti nei corrispettivi 3 caratteri ASCII e, in coda alla stringa di output, viene aggiunto un carattere di uguale ('=').

Sulla base dell'algoritmo di conversione appena esposto, è possibile concludere che il campo euiccChallenge del messaggio è uguale al valore intero 232645733703218863597835671721642336624, mentre il campo euiccInfo1 risulta pari alla sequenza di byte mostrata in figura 4.4.

Le specifiche di GSMA [3] suggeriscono che i tipi di dato complessi come euiccInfo1 in RSP vengono rappresentati secondo lo standard **ASN.1** sfruttando la codifica **DER**.

Codifica DER

DER (Distinguished Encoding Rules) è una variante restrittiva di **BER** (Basic Encoding Rules) che, di fatto, elimina alcune flessibilità offerte da BER [11].
BER a sua volta rappresenta un insieme di regole per la conversione di dati eterogenei in flussi di byte. Per la codifica, fa uso del formato **TLV** (Tag-Length-Value) e, per questa ragione, è particolarmente legato allo standard ASN.1 [12]. Di conseguenza, per comprendere al meglio il significato dei byte che compongono euiccInfo1, può risultare utile introdurre una panoramica generale sul funzionamento dello standard ASN.1.

```
BF 20 61 82 03 02 02 02  A9 2C 04 14 81 37 0F 51
25 D0 B1 D4 08 D4 C3 B2  32 E6 D2 5E 79 5B EB FB
04 14 18 1B F2 59 4C C2  E1 11 FF A3 F6 88 6E 10
11 32 12 EC 4E 41 AA 2C  04 14 81 37 0F 51 25 D0
B1 D4 08 D4 C3 B2 32 E6  D2 5E 79 5B EB FB 04 14
18 1B F2 59 4C C2 E1 11  FF A3 F6 88 6E 10 11 32
12 EC 4E 41
```

Figura 4.4: Sequenza di byte che descrive euiccInfo1 in initiateAuthentication.

Standard ASN.1

ASN.1 (Abstract Syntax Notation One) è uno standard di codifica che rappresenta le informazioni sfruttando il formato TLV, proprio come previsto dalle specifiche della codifica BER [13][14]. Ciò significa che ciascuna unità di informazione è suddivisa in tre parti:

- **Tag**: detto anche **type**, è composto tipicamente da un byte e indica il tipo di dato che si sta rappresentando. Il tipo può essere primitivo (e.g. Boolean, Integer, UTF8String [15]), composto (e.g. Sequence, Set [15]), un metadato (e.g. Object Identifier [15]) oppure un tipo di dato custom; in quest'ultimo caso, spesso il tag è composto da due byte anziché da uno.
- **Length**: è un campo che può essere composto da uno o più byte. Più precisamente, si hanno tre casi [14]:
  - Campo length composto da un solo byte compreso tra 0x00 e 0x7F (i.e. tra 0 e 127): in tal caso l'unico byte rappresenta la lunghezza effettiva del campo value.
  - Campo length composto da più di un byte: in tal caso il primo byte deve essere compreso tra 0x81 e 0xFF (i.e. tra 129 e 255) ed è tale per cui il suo valore diminuito di 0x80 (i.e. 128) rappresenta la lunghezza della restante porzione del campo length. La restante porzione del campo length, invece, indica la lunghezza effettiva del campo value. Questa convenzione viene applicata tutte e sole le volte in cui il campo value è più lungo di 127 byte. Per fare un esempio, per indicare una lunghezza di value pari a 257 byte, il campo length deve essere uguale a 0x820101: il primo byte (0x82) indica che, oltre a lui stesso, si hanno ulteriori due byte (i.e. 0x82-0x80) a comporre il campo length; gli ultimi due byte (0x0101), invece, stanno a indicare esattamente la lunghezza del campo value, che è proprio pari a 257.
  - Campo length il cui primo byte è pari a 0x80: in tal caso, si ha una length "indefinita", per cui è verosimilmente sintomo di una formattazione errata.
- **Value**: è il campo che contiene il payload vero e proprio dell'informazione che si sta rappresentando. Nel caso particolare in cui il relativo tipo di dato è composto, tale campo è costituito a sua volta da altri elementi TLV, creando così un'annidazione che può essere ricorsiva a piacere [14].

Comunque sia, conoscere lo standard ASN.1 non è sufficiente per interpretare in modo agevole le sequenze di byte come quella in figura 4.4 che è relativa a euiccInfo1. Infatti, a primo impatto si intuisce solo che euiccInfo1 è composto a sua volta da tre elementi in formato TLV (che iniziano rispettivamente coi byte 0x8203, 0xA92C, 0xAA2C), ma il loro significato non è chiaro. Di seguito è riportata la definizione precisa della struttura dati euiccInfo1 tratta da [16].

```
EUICCInfo1 ::= [32] SEQUENCE { -- Tag 'BF20'
    svn [2] VersionType, -- GSMA SGP.22 version supported (SVN)
    euiccCiPKIdListForVerification [9] SEQUENCE OF SubjectKeyIdentifier,
    euiccCiPKIdListForSigning [10] SEQUENCE OF SubjectKeyIdentifier
}
```

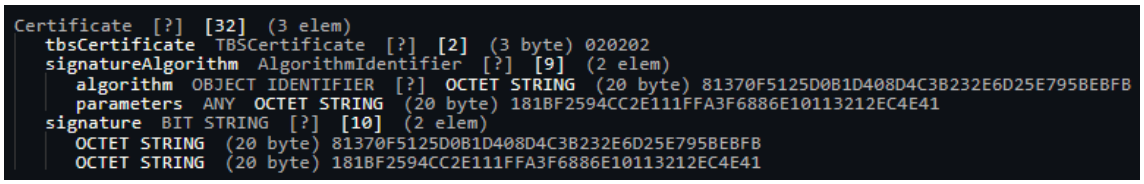


Figura 4.5: Decodifica del campo euiccInfo1 di initiateAuthentication.

Figura 4.6: Messaggio di initiateAuthenticationResponse visualizzato su Wireshark.

Come si può evincere, euiccInfo1 è una struttura dati composta da alcune informazioni che caratterizzano l'eUICC del dispositivo mobile coinvolto nella comunicazione. Tali informazioni sono schematizzate di seguito.

- lo Specification Version Number (SVN), che sarebbe la versione di GSMA SGP.22 supportata dall'eUICC;
- una lista (Sequence) di identificatori delle chiavi pubbliche del CI che possono essere utilizzate dall'eUICC per autenticare il server;
- una lista (Sequence) di identificatori delle chiavi pubbliche del CI che possono essere utilizzate dall'eUICC per firmare i propri dati.

Si noti che questa definizione di euiccInfo1 combacia perfettamente con le specifiche di GSMA [3] relative alla procedura di Common Mutual Authentication 2.4.5.
Per determinare nel modo più agevole possibile i valori assunti dai tre campi di euiccInfo1 sopra elencati, si può far riferimento al sito https://lapo.it/asn1js/ (*ASN.1 JavaScript decoder*) il quale, data una stringa base64, restituisce in chiaro le informazioni codificate con lo standard ASN.1. In figura 4.5 è riportato l'output del sito *ASN.1 JavaScript decoder* nel momento in cui gli viene dato in pasto la stringa base64 associata a euiccInfo1 in figura 4.2. In definitiva:

- SVN è pari alla sequenza di byte 0x020202, che verosimilmente indica la versione 2.2.2;
- euiccCiPKIdListForVerification ed euiccCiPKIdListForSigning sono due liste identiche ciascuna delle quali contiene due identificatori di chiavi pubbliche del CI.

4.3.2 Analisi di initiateAuthenticationResponse

La figura 4.6 mostra l'intero contenuto del primo messaggio di risposta del server SM-DP+.
Nell'header HTTP si ritrovano alcuni punti in comune col messaggio inviato dall'LPA: ad esempio, anche qui la versione HTTP utilizzata è la 1.1 e il protocollo usato per l'interazione tra LPA e server è la versione 2.2.0 di GSMA-RSP.

```
PKIMessage  SEQUENCE  (4 elem)
   header  PKIHeader  [?]  [0]  (16 byte) AEBC509ACAAA423690F473AD9F4A41B3
   body  PKIBody  [1]  (16 byte) AF05F0415BDD22987FAE2E0720E8A170
   [3]  (33 byte) sys.prod.ondemandconnectivity.com
   [4]  (16 byte) FF9B14F7C64615FA4F8E3012A2A75595
```

Figura 4.7: Decodifica del campo serverSigned1 di initiateAuthenticationResponse.

Il payload del messaggio ha una struttura JSON del tutto analoga a quella di initiateAuthentication. In particolare, si hanno i seguenti campi:

- **header**: è un campo in chiaro che indica l'esito del processamento del messaggio dell'LPA al livello del protocollo RSP. La presenza di { *"status":"Executed-Success"*} indica che tutti i check sulle informazioni ricevute dal client sono andati a buon fine e la comunicazione può proseguire secondo le specifiche di GSMA;
- **transactionId**: è un campo in chiaro che indica l'ID della transazione;
- **serverSigned1**: è un campo codificato in base64;
- **serverSignature1**: è un altro campo codificato in base64 che, come enunciato nella sezione 2.4.5, rappresenta la signature di serverSigned1;
- **euiccCiPKIdToBeUsed**: è il terzo campo codificato in base64;
- **serverCertificate**: è l'ultimo campo codificato in base64. La sua lunghezza è giustificata dal fatto che, con riferimento alla sezione 2.4.5, rappresenta il certificato CERT.DPauth.SIG, che è verosimilmente una struttura dati molto complessa.

Per comprendere appieno le informazioni celate nei campi di initiateAuthenticationResponse codificati in base64, si fa nuovamente riferimento a [16] e al sito *ASN.1 JavaScript decoder*.

La struttura dati serverSigned1 è molto semplice: dalla sua definizione riportata di seguito e dalla figura 4.7 [16], si capisce che è composta dall'ID di transazione, dalla challenge (euiccChallenge) che il server aveva ricevuto in initiateAuthentication, dall'hostname del server e da una nuova challenge generata dal server (serverChallenge).

```
ServerSigned1 ::= [32] SEQUENCE {
        transactionId [0] TransactionId, -- The Transaction ID generated by the RSP Server
        euiccChallenge [1] Octet16, -- The eUICC Challenge
        serverAddress [3] UTF8String, -- The RSP Server address
        serverChallenge [4] Octet16, -- The RSP Server Challenge
}
```

Il campo euiccCiPKIdToBeUsed cela anch'esso una struttura TLV dello standard ASN.1, com'è possibile vedere dalle figure 4.8, 4.9. Inoltre, il relativo campo value assume un valore che è comparso sia in euiccCiPKIdListForVerification, sia in euiccCiPKIdListForSigning, il che vuol dire che il server SM-DP+ accetta l'utilizzo di una delle due chiavi pubbliche di CA (i.e. una delle due certificate chain) proposte dal client nel messaggio initiateAuthentication.

Per quanto invece riguarda il campo serverCertificate di initiateAuthenticationResponse, il sito *ASN.1 JavaScript decoder* aiuta a comprendere molti dettagli sul certificato CERT.DPauth.SIG e sulla relativa certificate chain utilizzata. Le figure 4.10, 4.11 mostrano la decodifica di tale certificato. Inizialmente vi sono gli attributi fondamentali, come:

- **signature AlgorithmIdentifier**: indica con quale algoritmo vengono calcolate le signature mediante la chiave privata associata al certificato ed è pari a *ecdsaWithSHA256* (ANSI X.962 ECDSA algorithm with SHA256);

```
04 14 81 37 0F 51 25 D0  B1 D4 08 D4 C3 B2 32 E6
D2 5E 79 5B EB FB
```

Figura 4.8: Sequenza di byte che descrive euiccCiPKIdToBeUsed in initiateAuthenticationResp.

```
Certificate [?] OCTET STRING (20 byte) 81370F5125D0B1D408D4C3B232E6D25E795BEBFB
```

Figura 4.9: Decodifica del campo euiccCiPKIdToBeUsed di initiateAuthenticationResponse.

- **issuer Name**: è un campo relativo alla CA che ha rilasciato il presente certificato ed è a sua volta articolato in due sotto-campi, che sono l'organizationName (pari a "GSM Association") e il commonName (pari a "GSM Association - RSP2 Root CI1");
- **validity Validity**: dichiara che il certificato è valido dal 30 marzo 2023 al 29 marzo 2026, il che corrisponde a tre anni;
- **subject Name**: è un campo che fornisce dettagli informativi sul presente certificato ed è a sua volta composto in diversi sotto-campi, tra cui countryName (pari a "FR", i.e. Francia), localityName (pari a "Tours"), organizationName (pari a "Thales DIS France SA"), organizationalUnitName (pari a "ODC") e commonName (pari a "SMDP Plus Tours Platform").
- **subjectPublicKeyInfo SubjectPublicKeyInfo**: fornisce maggiori informazioni sulle chiavi associate al certificato, tra cui la curva ellittica utilizzata per generare le signature, che è identificata con *prime256v1*.

Più in basso nel certificato si trovano le cosiddette estensioni, ovvero informazioni aggiuntive opzionali che possono essere inserite a piacimento all'interno del certificato. Tra queste è possibile individuare:

- **subjectKeyIdentifier**: è l'identificatore della chiave pubblica associata al presente certificato;
- **authorityKeyIdentifier**: è l'identificatore della chiave pubblica dell'issuer del certificato e, per coerenza, deve corrispondere o contenere il valore associato al campo euiccCiPKIdToBeUsed del messaggio initiateAuthenticationResponse;
- **keyUsage**: rappresenta i possibili utilizzi delle chiavi associate al certificato;
- **subjectAltName**: rappresenta il nome alternativo dell'entità che possiede il certificato;
- altre informazioni che non verranno esaminate a fondo in questa trattazione, come certificatePolicies e cRLDistributionPoints.

Infine, in fondo al certificato, si hanno ulteriori informazioni riguardanti la signature e l'algoritmo di signature.

4.3.3 Analisi di authenticateClient

La figura 4.12 mostra il contenuto del secondo messaggio inviato dal client (authenticateClient). Il JSON del payload è composto da due campi:

- **authenticateServerResponse**: è una struttura dati che verrà approfondita tra poco;
- **transactionId**: corrisponde all'ID di transazione appena ricevuto nel messaggio initiateAuthenticationResponse.

Per quanto riguarda authenticateServerResponse, è una struttura composta da più elementi ASN.1 di tipo Sequence anche annidati tra loro, ed è descritta nella definizione riportata nelle pagine successive[16].

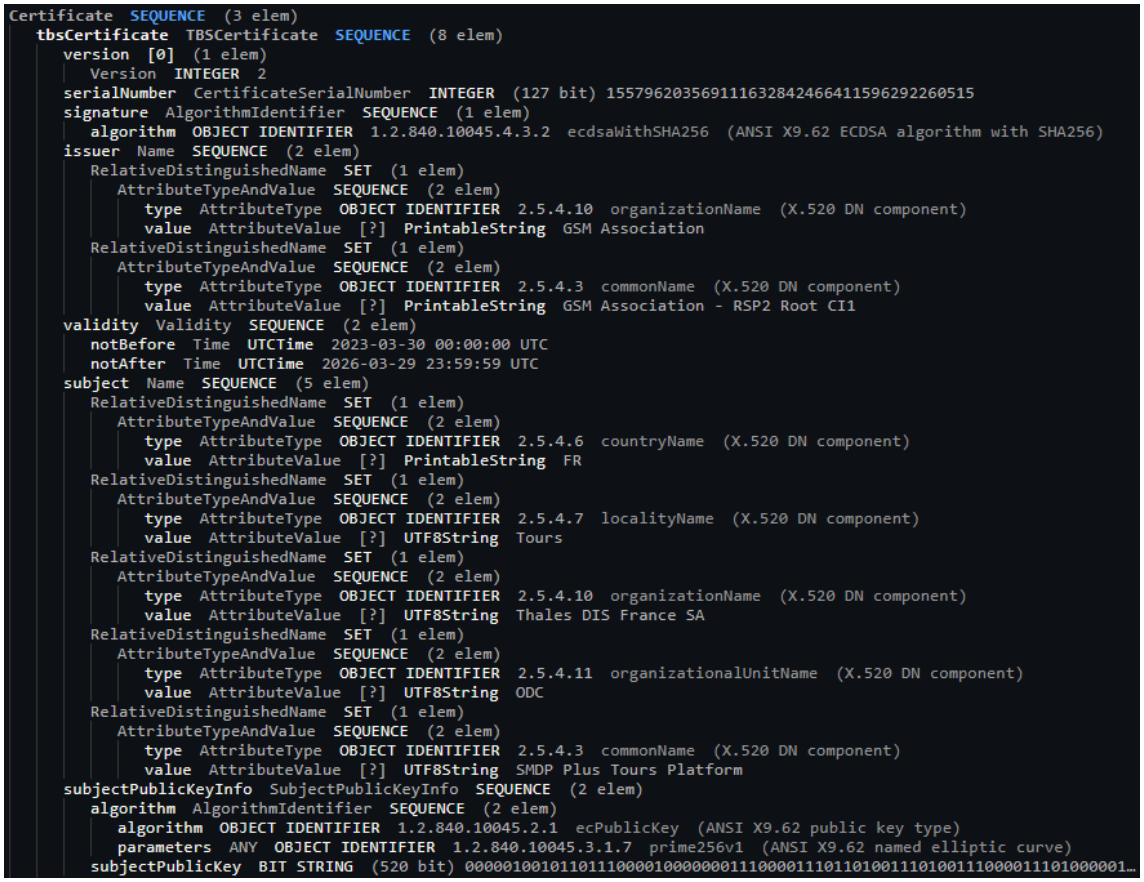


Figura 4.10: Decodifica del campo serverCertificate di initiateAuthenticationResponse (parte 1/2).

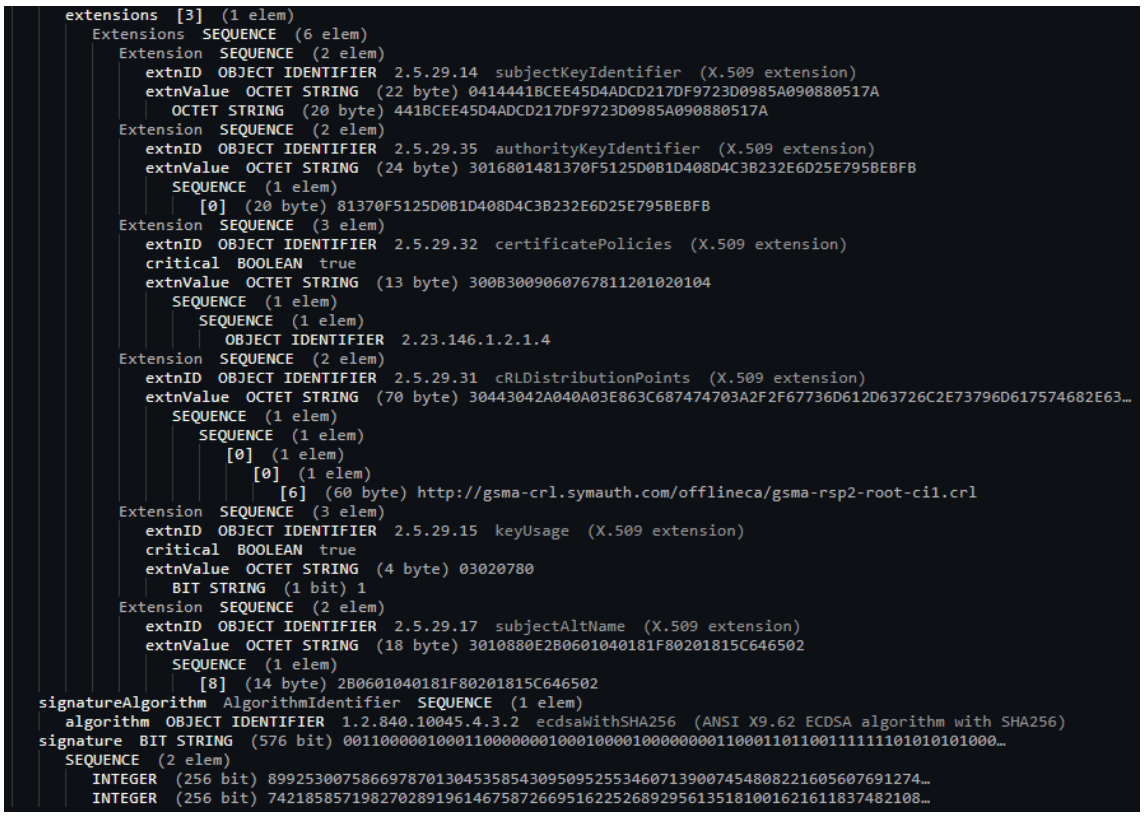


Figura 4.11: Decodifica del campo serverCertificate di initiateAuthenticationResponse (parte 2/2).

Figura 4.12: Messaggio di authenticateClient visualizzato su Wireshark.

```
AuthenticateServerResponse ::= [56] CHOICE { -- Tag 'BF38'
    authenticateResponseOk AuthenticateResponseOk,
    authenticateResponseError AuthenticateResponseError
}

AuthenticateResponseOk ::= SEQUENCE {
    euiccSigned1 EuiccSigned1, -- Signed information
    euiccSignature1 [APPLICAION 55] OCTET STRING, -- EUICC_Sign1, tag '5F37'
    euiccCertificate Certificate, -- CERT.EUICC.ECDSA signed by the EUM
    eumCertificate Certificate -- CERT.EUM.ECDSA signed by the CI
}

EuiccSigned1 ::= SEQUENCE {
    transactionId [0] TransactionId,
    serverAddress [3] UTF8String,
    serverChallenge [4] Octet16, -- The RSP Server Challenge
    euiccInfo2 [34] EUICCInfo2,
    ctxParams1 CtxParams1
}

EUICCInfo2 ::= [34] SEQUENCE { -- Tag 'BF22'
    profileVersion [1] VersionType, -- SIMAlliance Profile package version supported
    svn [2] VersionType, -- GSMA SGP.22 version supported (SVN)
    euiccFirmwareVer [3] VersionType, -- eUICC Firmware version
    extCardResource [4] OCTET STRING, -- Extended Card Resource Information
    uiccCapability [5] UICCCapability,
    ts102241Version [6] VersionType OPTIONAL,
    globalplatformVersion [7] VersionType OPTIONAL,
    rspCapability [8] RspCapability,
    euiccCiPKIdListForVerification [9] SEQUENCE OF SubjectKeyIdentifier,
    euiccCiPKIdListForSigning [10] SEQUENCE OF SubjectKeyIdentifier,
    euiccCategory [11] INTEGER {
        other(0),
        basicEuicc(1),
        mediumEuicc(2),
        contactlessEuicc(3)
    } OPTIONAL,
```

```
        forbiddenProfilePolicyRules [25] PprIds OPTIONAL, -- Tag '99'
        ppVersion VersionType, -- Protection Profile version
        sasAcreditationNumber UTF8String (SIZE(0..64)),
        certificationDataObject [12] CertificationDataObject OPTIONAL
}

CtxParams1 ::= CHOICE {
        ctxParamsForCommonAuthentication CtxParamsForCommonAuthentication
}

CtxParamsForCommonAuthentication ::= SEQUENCE {
        matchingId UTF8String OPTIONAL,
        deviceInfo DeviceInfo -- The Device information
}
```

Com'è possibile osservare, authenticateServerResponse si articola in due casi: authenticateResponseOk nel caso in cui l'autenticazione del server SM-DP+ sia andata a buon fine, e authenticateResponseError in caso contrario. La struttura authenticateResponseError è composta semplicemente dall'ID di transazione e da un codice di errore [16], mentre la struttura authenticateResponseOk è molto più articolata.
Anzitutto, authenticateResponseOk è composto da quattro campi: euiccSigned1 che è a sua volta un'altra struttura dati complessa, euiccSignature1 che, come ormai suggerisce il nome, rappresenta la firma digitale di euiccSigned1, il certificato dell'eUICC (CERT.EUICC.SIG) e il certificato dell'EUM (CERT.EUM.SIG). La struttura euiccSigned1 si articola in:

- **transactionId**: è l'ID della transazione;
- **serverAddress**: è l'hostname del server SM-DP+;
- **serverChallenge**: è la challenge generata precedentemente dal server e comunicata al client mediante il messaggio initiateAuthenticationResponse;
- **euiccInfo2**: è un'ulteriore struttura dati che comprende tutte le informazioni sull'eUICC che si sta per autenticare e su cui avverrà l'installazione del nuovo profilo. È un sovrainsieme del campo euiccInfo1 di initiateAuthentication;
- **ctxParams1**: come suggerito dal nome, rappresenta i parametri di contesto e, in particolar modo, le informazioni del dispositivo e il Matching ID. Quest'ultimo è tipicamente un token associato all'Activation Code (codice di attivazione) [16], il quale viene negoziato da server SM-DP+ e operatore durante la fase di *profile ordering* e viene rilasciato dall'operatore mediante un'apposita email durante la fase di *download initialization* (sezione 2.4.3). Un estratto dell'email di Very Mobile relativa al profilo che si sta per installare sull'eUICC del dispositivo mobile di riferimento è riportato in figura 4.13.

La figura 4.14 mostra come il sito *ASN.1 JavaScript decoder* decodifica il campo euiccSigned1 a partire dalla stringa base64 relativa all'ASN.1 del campo authenticateServerResponse del messaggio authenticateClient. A partire dalla sestultima riga dell'immagine si ha la traduzione della struttura ctxParams1: qui si nota la corrispondenza tra il sotto-campo matchingId e l'Activation Code presente nell'email di cui sopra.

4.3.4 Analisi di authenticateClientResponse

La figura 4.15 mostra il contenuto della risposta del server al secondo messaggio inviato dal client (authenticateClientResponse).
Il JSON del payload è composto dai campi elencati qui di seguito:

- **header**: contiene di nuovo { *"status":"Executed-Success"*}, il che vuol dire che anche in questa fase tutti i check effettuati dal server SM-DP+ hanno avuto successo;
- **transactionId**: è l'ID di transazione e assume sempre il medesimo valore;

Figura 4.13: Estratto dell'email di Very Mobile.

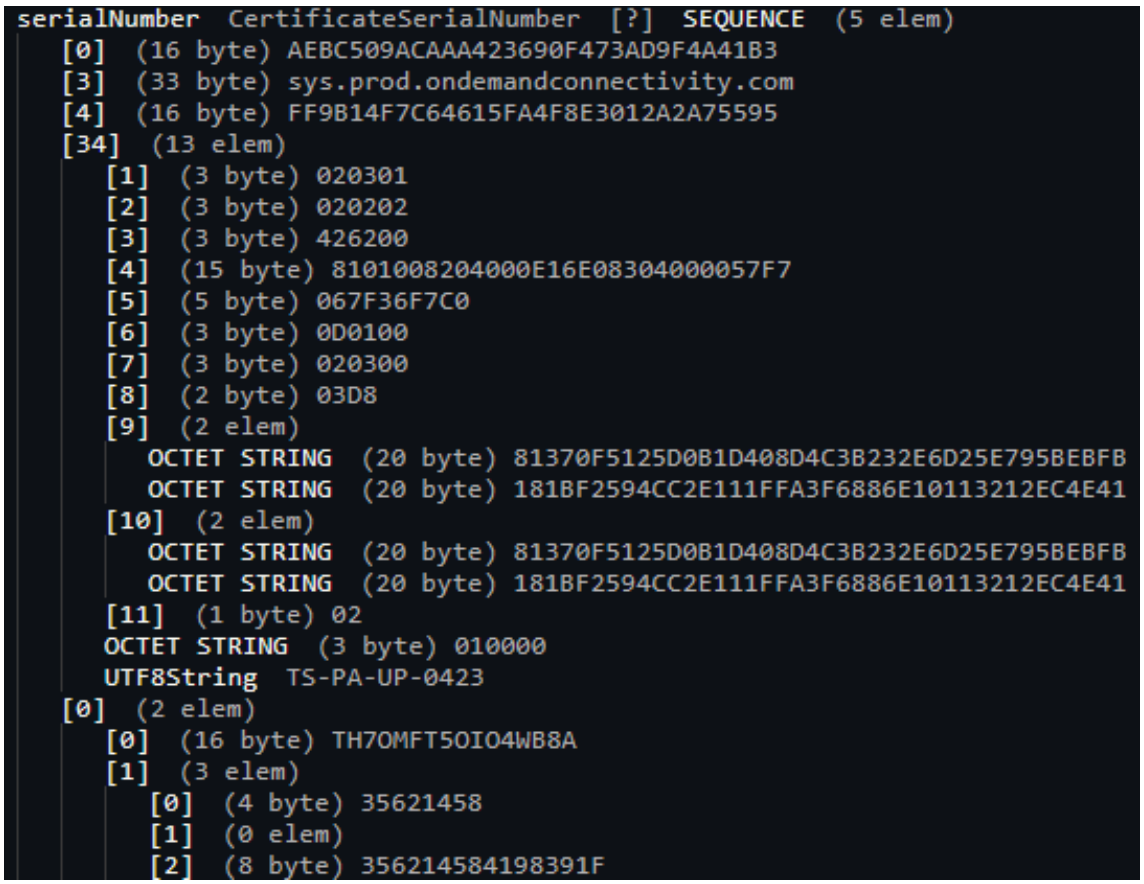


Figura 4.14: Decodifica del campo euiccSigned1 di authenticateServerResponse.

}HTTP/1.1 200
Date: Mon, 17 Jul 2023 16:24:20 GMT
Server: WAF
Content-Type: application/json
X-Admin-Protocol: gsma/rsp/v2.2.0
X-Kong-Upstream-Latency: 141
X-Kong-Proxy-Latency: 1
Via: kong/2.7.0
Vnd-Gemalto-Com-Log-CorrelationId: 5800a94a19a39b5657ff666ed8386040
Content-Length: 1326
Set-Cookie: GWAFSESSION=s.sys-es9plus-in1;path=/;httpOnly
Set-Cookie: GWAFSESSION=s.g401;path=/;httpOnly
Keep-Alive: timeout=60, max=299
Connection: Keep-Alive

{"header":{"functionExecutionStatus":{"status":"Executed-Success"}},"transactionId":"AEBC509ACAAA423690F473AD9F4A41B3","profileMetadata":"vyV
iWgqYk4iAVEBmZEj0kQRWZXJ5kgtWRVJZIG1vYmlsZZUBArYpMCeAAgRwgSFzeXMucHJvZC5vbmRlbWFuZGNvbm5lY3Rpdml0eS5jb223E4ADIvKIgQUAAAAAmYIFAAAAAAA=","smdpS
igned2":"MBWAEK68UJrKqkI2kPRzrZ9KQbMBAQA=","smdpSignature2":"XzdAvZ6AOIBoRkYKsIGC1JloJDZH00O0DIzrZaLrK67Op//a4ctfRxxKIs4KEfUSKZccjtbtquGIui/+
zn/uUDdPgg==","smdpCertificate":"MIIChzCCAiygAwIBAgIQFq05Sl1uEduZcpS3ahAGlTAKBggqhkjOPQQDAjBEMRgwFgYDVQQKEw9HU00gQXNzb2NpYXRpb24xKDAmBgNVBAMT
H0dTTSBBc3NvY2lhdGlvbiAtIFJTUDIgUm9vdCBDSTEwHhcNMjMwMzMwMDAwMDAwWhcNMjYwMzI5MjM1OTU5WjBtMQswCQYDVQQGEwJGUjEOMAwGA1UEBwwFVG91cnMxHTAbBgNVBAoMF
FRoYWxlcyBESVMgRnJhbmNlIFNBMQwwCgYDVQQLDANPREMxITAfBgNVBAMMGFNNRFAgUGx1cyBUb3VycyBQbGF0Zm9ybTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABOn45vtZu0n0LS
TDZgQbf9o6VHPxK3hXbf1yCLJ2Lo4TDQIGYnoC6nVB1Saz9exWla3Grk9rfskWCfMLTR/mYQGjgdYwgdMwFwYDVR0gAQH/BA0wCzAJBgdngRIBAgEFME0GA1UdHwRGMEQwQqBAoD6GPGh
0dHA6Ly9nc21hLWNybC5zeW1hdXRoLmNvbS9vZmZsaW5lY2EvZ3NtYS1yc3AyLXJvb3QtY2kxLmNybDAOBgNVHQ8BAf8EBAMCB4AwGQYDVR0RBBIwEIgOKwYBBAGB+AIBgVxkZQIwHQYD
VR0OBBYEFBLxY2T7tPYTLF1YK4l1LjsT5FOFMB8GA1UdIwQYMBaAFIE3D1El0LHUCNTDsjLm0l55W+v7MAoGCCqGSM49BAMCA0kAMEYCIQCj/5av1pJZ5vUQbOCnexOsb0v8x5F0IcVo2
/QN79MDTwIhAP61QJhmYEM2J+fyblCFpyivvIxNl2prk3cfGyddH9oj"}POST /gsma/rsp2/es9plus/getBoundProfilePackage HTTP/1.1

Figura 4.15: Messaggio di authenticateClientResponse visualizzato su Wireshark.

```
BF 25 62 5A 0A 98 93 88  80 54 40 66 64 48 F4 91
04 56 65 72 79 92 0B 56  45 52 59 20 6D 6F 62 69
6C 65 95 01 02 B6 29 30  27 80 02 04 70 81 21 73
79 73 2E 70 72 6F 64 2E  6F 6E 64 65 6D 61 6E 64
63 6F 6E 6E 65 63 74 69  76 69 74 79 2E 63 6F 6D
B7 13 80 03 22 F2 88 81  05 00 00 00 00 99 82 05
00 00 00 00 00
```

Figura 4.16: Sequenza di byte che descrive profileMetadata in authenticateClientResponse.

- **profileMetadata**: è un campo codificato in base64 che, come suggerisce il nome, dovrebbe comprendere i metadati del profilo di cui si sta per effettuare il download;
- **smdpSigned2**: è un campo codificato in base64 che, come suggerisce il nome, dovrebbe essere strutturalmente simile a serverSigned1 e a euiccSigned1;
- **smdpSignature2**: è un campo codificato in base64 relativo alla signature di smdpSigned2;
- **smdpCertificate**: è un ultimo campo codificato in base64 che, con riferimento alla sezione 2.4.6, rappresenta il certificato CERT.DPpb.SIG, che è potenzialmente diverso dal certificato CERT.DPauth.SIG inviato all'LPA precedentemente e viene utilizzato per firmare il profilo che si sta per scaricare.

I campi che possono essere approfonditi con l'aiuto di [16] e del sito *ASN.1 JavaScript decoder* sono profileMetadata (la cui sequenza di byte in ASN.1 è mostrata in figura 4.16) e smdpSigned2.

La struttura dati profileMetadata, come riportato di seguito, comprende informazioni sul profilo target come l'ICCID (che, come spiegato nella sezione 2.4.8, è l'identificativo del profilo), il nome dell'operatore, il nome del profilo, la classe del profilo ed eventuali Profile Policy Rules.

```
StoreMetadataRequest ::= [37] SEQUENCE { -- Tag 'BF25'
    iccid Iccid,
    serviceProviderName [17] UTF8String (SIZE(0..32)), -- Tag '91'
    profileName [18] UTF8String (SIZE(0..64)), -- Tag '92'
    iconType [19] IconType OPTIONAL, -- Tag '93' (JPG or PNG)
    icon [20] OCTET STRING (SIZE(0..1024)), -- Tag '94' (Data of the icon)
    profileClass [21] ProfileClass DEFAULT operational, -- Tag '95'
    notifConfig [22] SEQUENCE OF NotificationConfigurationInformation OPTIONAL,
    profileOwner [23] OperatorId OPTIONAL, -- Tag 'B7'
    profilePolicyRules [25] PprIds OPTIONAL -- Tag '99'
}
```

La struttura dati smdpSigned2, come mostrato di seguito, contiene l'ID di transazione e un booleano che indica se è richiesto il Confirmation Code immesso dall'utente per completare il download e l'installazione del profilo. La necessità o meno del Confirmation Code e l'eventuale relativo valore vengono negoziati da server SM-DP+ e operatore durante la fase di *profile ordering* (sezione 2.4.3). Nel caso specifico del profilo scaricato con l'operatore Very Mobile, il Confirmation Code non è richiesto, come testimoniato dalla figura 4.17 tratta dal sito *ASN.1 JavaScript decoder*.

```
SmdpSigned2 ::= SEQUENCE {
    transactionId [0] TransactionId, -- The TransactionID generated by the SM-DP+
    ccRequiredFlag BOOLEAN, -- Indicates if the Confirmation Code is required
    bppEuiccOtpk [APPLICATION 73] OCTET STRING OPTIONAL -- otPK.EUICC.ECKA
}
```

```
Certificate  SEQUENCE  (2 elem)
   tbsCertificate  TBSCertificate  [?]  [0]  (16 byte) AEBC509ACAAA423690F473AD9F4A41B3
   signatureAlgorithm  AlgorithmIdentifier  [?]  BOOLEAN  false
```

Figura 4.17: Decodifica del campo smdpSigned2 di authenticateClientResponse.

```
POST /gsma/rsp2/es9plus/getBoundProfilePackage HTTP/1.1
Content-Type: application/json
Accept: application/json
User-Agent: gsma-rsp-com.truphone.lpad
X-Admin-Protocol: gsma/rsp/v2.2.0
Host: sys.prod.ondemandconnectivity.com
Connection: Keep-Alive
Accept-Encoding: gzip
Content-Length: 297

{"prepareDownloadResponse":"vyGBnqCBmzBWgBCuvFCayqpCNpD0c62fSkGzX0lBBA8Vn1pAS+PWaygMoZEU3TDN7O2F7TXoV8
x082MJ3EKh4R3wu+I08UKvxOOGWEQ1ShpXHKkSPJaT+UQYYgb9sYRfN0Bv+E9IlcngRB9UcJk9NgX7ehgvQfwiPvf8LY9I5gCZWhWI
D5Zxj+m3xACv7inuufhuF6qCgU7kY/97OfLwiwvP","transactionId":"AEBC509ACAAA423690F473AD9F4A41B3"}HTTP/1.1
```

Figura 4.18: Messaggio di getBoundProfilePackage visualizzato su Wireshark.

4.3.5 Analisi di getBoundProfilePackage

La figura 4.18 mostra il contenuto del terzo messaggio inviato dal client (getBoundProfilePackage). Il payload è composto da due campi:

- **prepareDownloadResponse**: è una struttura dati in base64 che verrà descritta a breve;
- **transactionId**: è il solito ID di transazione.

Qui di seguito si ha la definizione di prepareDownloadResponse [16].

PrepareDownloadResponse ::= [33] CHOICE { *-- Tag 'BF21'*
downloadResponseOk PrepareDownloadResponseOk,
downloadResponseError PrepareDownloadResponseError
}

PrepareDownloadResponseOk ::= SEQUENCE {
euiccSigned2 EUICCSigned2, -- Signed information
euiccSignature2 [APPLICATION 55] OCTET STRING -- tag '5F37'
}

EUICCSigned2 ::= SEQUENCE {
transactionId [0] TransactionId,
euiccOtpk [APPLICATION 73] OCTET STRING, -- otPK.EUICC.ECKA, tag '5F49'
hashCc Octet32 OPTIONAL -- Hash of confirmation code
}

Anche prepareDownloadResponse, come authenticateServerResponse nel messaggio authenticateClient, si articola in due casi: prepareDownloadResponseOk nel caso in cui tutti i controlli effettuati dal client alla ricezione di authenticateClientResponse abbiano avuto successo, e prepareDownloadResponseError in caso contrario. La struttura dati prepareDownloadResponseError, in modo perfettamente analogo ad authenticateResponseError, è composta dall'ID di transazione e da un codice di errore.
D'altra parte, prepareDownloadResponseOk è composto dalla struttura dati euiccSigned2 e dalla relativa digital signature euiccSignature2, dove euiccSigned2 è composto a sua volta da:

- **transactionId**: è l'ID di transazione;
- **euiccOtpk**: è la chiave pubblica otPK.EUICC.KA (sezione 2.3.1);
- **hashCc**: è un campo opzionale che è presente se il Confirmation Code è richiesto e corrisponde al valore SHA256(SHA256(Confirmation Code) | transactionID).

Figura 4.19: Messaggio di getBoundProfilePackageResponse visualizzato su Wireshark.

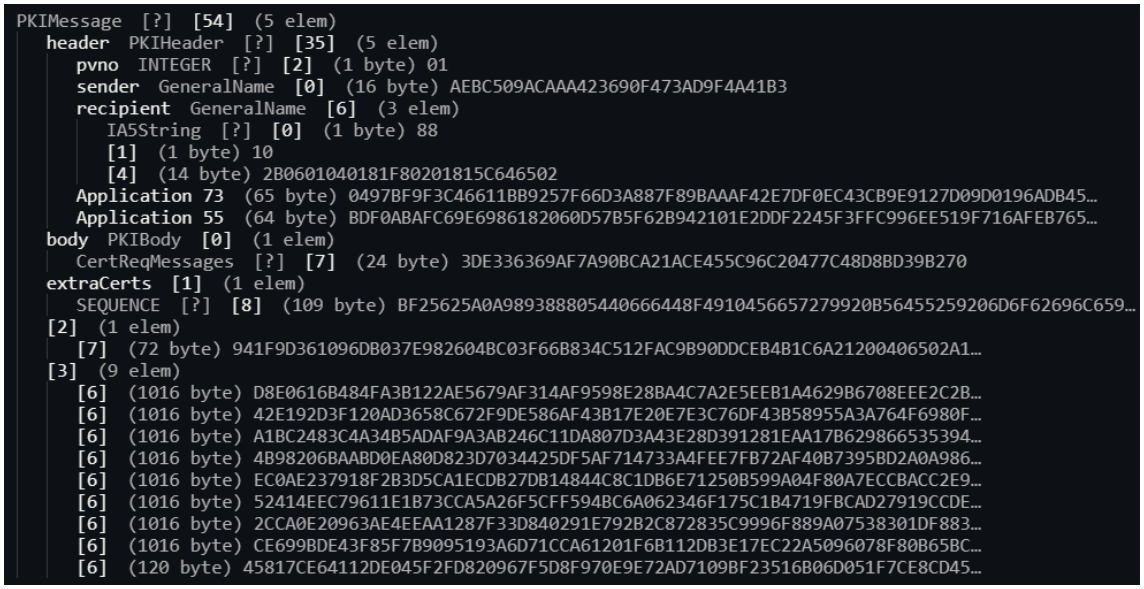


Figura 4.20: Decodifica del campo boundProfilePackage di getBoundProfilePackageResponse.

4.3.6 Analisi di getBoundProfilePackageResponse

La figura 4.19 mostra il contenuto della risposta del server al terzo messaggio inviato dal client (getBoundProfilePackageResponse).
Il payload è composto dai seguenti tre campi:

- **header**: contiene ancora una volta { *"status":"Executed-Success"*};
- **transactionId**: è l'ID di transazione;
- **boundProfilePackage**: è un campo codificato in base64 ed è molto grande, tant'è vero che non rientra per intero nella figura 4.19.

Com'è possibile osservare nella definizione riportata di seguito [16], il campo boundProfilePackage è composto da strutture di tipo TLV, ovvero sequenze di byte che a loro volta possono essere interpretate come elementi ASN.1 coi loro tag, le loro length e i loro value. Dalle figure 4.16, 4.20, è possibile accorgersi che il campo sequenceOf88 di boundProfilePackage, che in questo caso specifico corrisponde al campo da 109 byte, contiene una sequenza di byte che corrisponde all'intero TLV di profileMetadata. D'altra parte, il campo sequenceOf86, complice anche la sua dimensione, ha

tutta l'aria di essere il vero e proprio payload (cifrato) del profilo target.

```
BoundProfilePackage ::= [54] SEQUENCE { -- Tag 'BF36'
    initialiseSecureChannelRequest [35] InitialiseSecureChannelRequest, -- Tag 'BF23'
    firstSequenceOf87 [0] SEQUENCE OF [7] OCTET STRING, -- sequence of '87' TLVs
    sequenceOf88 [1] SEQUENCE OF [8] OCTET STRING, -- sequence of '88' TLVs
    secondSequenceOf87 [2] SEQUENCE OF [7] OCTET STRING OPTIONAL,
    sequenceOf86 [3] SEQUENCE OF [6] OCTET STRING -- sequence of '86' TLVs
}
```

4.3.7 Analisi di handleNotification

Nelle due tracce diverse da quella di riferimento sono stati catturati anche dei messaggi di handleNotification inviati al server SM-DP+ da parte dell'LPA. Può tornare utile analizzarne un campione, come quello riportato in figura 4.21.
Si ha un solo campo JSON, pendingNotification, la cui definizione è mostrata qui di seguito.

```
PendingNotification ::= CHOICE {
    profileInstallationResult [55] ProfileInstallationResult, -- Tag 'BF37'
    otherSignedNotification OtherSignedNotification
}

OtherSignedNotification ::= SEQUENCE {
    tbsOtherNotification NotificationMetadata,
    euiccNotificationSignature [APPLICATION 55] OCTET STRING, -- eUICC signature
    euiccCertificate Certificate, -- CERT.EUICC.ECDSA
    eumCertificate Certificate -- CERT.EUM.ECDSA
}

ProfileInstallationResult ::= [55] SEQUENCE { -- Tag 'BF37'
    profileInstallationResultData [39] ProfileInstallationResultData,
    euiccSignPIR EuiccSignPIR
}

ProfileInstallationResultData ::= [39] SEQUENCE { -- Tag 'BF27'
    transactionId[0] TransactionId, -- The Transaction ID generated by the SM-DP+
    notificationMetadata [47] NotificationMetadata,
    smdpOid OBJECT IDENTIFIER, -- SM-DP+ OID
    finalResult [2] CHOICE {
        successResult SuccessResult,
        errorResult ErrorResult
    }
}

SuccessResult ::= SEQUENCE {
    aid [APPLICATION 15] OCTET STRING (SIZE (5..16)), -- AID of ISD-P
    simaResponse OCTET STRING -- contains (multiple) 'EUICCResponse'
}

ErrorResult ::= SEQUENCE {
    bppCommandId BppCommandId,
    errorReason ErrorReason,
    simaResponse OCTET STRING OPTIONAL -- contains (multiple) 'EUICCResponse'
}
```

Si hanno due casi possibili: pendingNotification pari alla struttura dati profileInstallationResult e pendingNotification pari alla struttura dati otherSignedNotification.

Figura 4.21: Messaggio di handleNotification visualizzato su Wireshark.

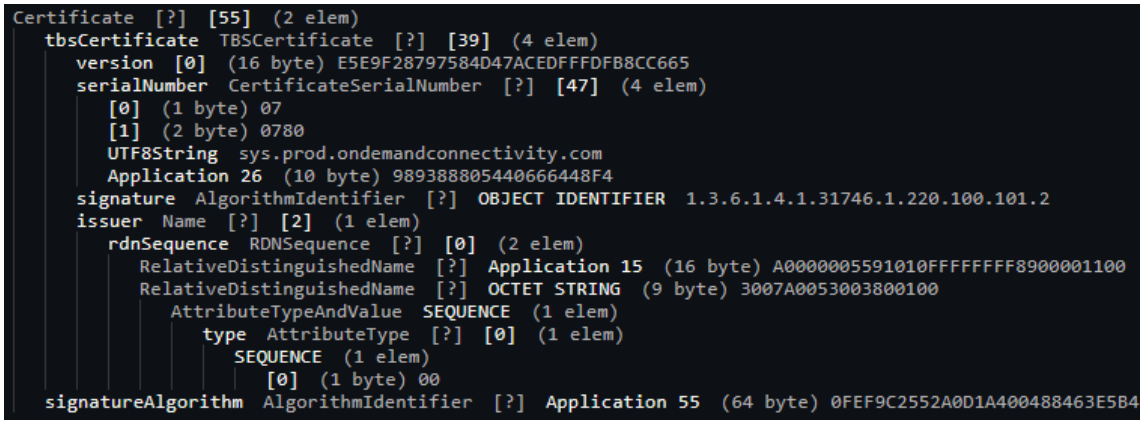


Figura 4.22: Decodifica del campo pendingNotification di handleNotification.

- Nel primo caso, si ha una struttura profileInstallationResultData composta da ID di transazione, alcuni metadati di notificazione, OID (i.e. l'identificatore) del server SM-DP+ ed esito finale dell'installazione del profilo all'interno dell'eUICC (che può essere un successResult o un errorResult). Oltre a profileInstallationResultData, il campo pendingNotification è composto anche da euiccSignPIR, che risulta essere la signature di profileInstallationResultData.
- Nel secondo caso, invece, si hanno alcuni metadati di notificazione, la relativa digital signature, il certificato dell'eUICC (CERT.EUICC.SIG) e il certificato dell'EUM (CERT.EUM.SIG). Intuitivamente, si ricade in questa casistica in contesti differenti da quello in cui l'LPA ha bisogno di comunicare l'esito dell'installazione del profilo (i.e. il successResult o l'errorResult).

Poiché, com'è stato già visto, la struttura dei certificati è piuttosto grande e poiché la stringa in base64 che codifica il campo pendingNotification del messaggio handleNotification, al contrario, è molto breve, è già possibile intuire che il messaggio catturato nella traccia ricada nel primo caso esposto precedentemente, dove pendingNotification è pari alla struttura dati profileInstallationResult che, verosimilmente, contiene un successResult. Tale intuizione trova conferma in figura 4.22 tratta dal sito *ASN.1 JavaScript decoder*, dove chiaramente non compaiono riferimenti a certificati.

4.4 Implementazione dei simulatori

Il sistema di simulazione sviluppato è un'applicazione Python composta da tre moduli principali:

- **rootCA**: è il modulo relativo al Certificate Issuer. È composto unicamente da un eseguibile per la generazione di certificati custom;
- **smdp**: è il modulo relativo al server SM-DP+. È composto a sua volta da:
  - un eseguibile per la generazione di certificati custom;
  - un eseguibile per la comunicazione con l'operatore per il rilascio di un nuovo profilo;
  - un eseguibile per le procedure di Common Mutual Authentication e di download e installazione del profilo;

  - un file per tenere traccia delle informazioni sui profili esistenti che sono il Matching ID, l'EID dell'eUICC a cui il profilo è destinato, l'ICCID, il numero di tentativi di download effettuati, l'ID dell'eventuale transazione in cui si sta tentando correntemente di installare il profilo, lo stato attuale del profilo, il tipo di profilo e l'eventuale Confirmation Code che serve per fornire la conferma finale sul download del profilo;
  - un eseguibile per effettuare il clean-up del file di cui sopra, in modo tale da lasciare esclusivamente i profili in uno stato diverso da 'Released' e da 'Error'.
  - un eseguibile per generare il QR code associato al profilo con un dato Matching ID.

- **euicc**: è il modulo che comprende la coppia eUICC + LPA e l'operatore telefonico. Questo accorpamento consente di semplificare la logica applicativa lato client e di bypassare, ad esempio, la comunicazione tra l'operatore e l'LPA che si ha nella fase di download initialization (sezione 2.4.3). Tale modulo è composto a sua volta da:
  - un eseguibile per la generazione di certificati custom sia per l'eUICC che per l'EUM;
  - un eseguibile per la comunicazione col server SM-DP+ per il rilascio di un nuovo profilo (qui l'attore impersonificato è l'operatore);
  - un eseguibile per le procedure di Common Mutual Authentication e di download e installazione del profilo (qui l'attore impersonificato è la coppia eUICC + LPA);
  - un file che rappresenta una RAT.

4.4.1 Librerie Python utilizzate

La tabella 4.1 racchiude tutte le librerie Python che sono state sfruttate per il sistema simulato e i relativi utilizzi.

4.4.2 Confronto tra specifiche GSMA e simulatori

Per quanto concerne i dettagli implementativi del sistema simulato, è stato possibile formattare i messaggi scambiati tra le controparti in modo perfettamente compatibile con le specifiche di GSMA sia grazie al livello di dettaglio fornito dalle specifiche stesse [3], sia grazie alla completezza delle definizioni ASN.1 per il protocollo RSP [16], anche se è stato necessario ritoccare queste ultime con piccole correzioni al fine di ottenere un funzionamento corretto. Un aspetto sicuramente più interessante da trattare riguarda i controlli che devono essere effettuati dalle controparti a ogni messaggio ricevuto.

Check da parte del server alla ricezione di initiateAuthetication

- Validità dell'indirizzo SM-DP+: l'implementazione prevede un controllo su se l'hostname specificato è uguale al vero hostname del server.
- Validità del contenuto di euiccInfo1: l'implementazione prevede un controllo su se in euiccCiPKIdListForVerification e in euiccCiPKIdListForSigning compare il valore di euiccCiPKIdToBeUsed.

Check da parte dell'LPA alla ricezione di initiateAuthenticationResponse

- Validità dell'OID (opzionale): non implementato.
- Validità dell'indirizzo SM-DP+: l'implementazione prevede un controllo su se l'hostname ricevuto corrisponde a quello inviato nel precedente messaggio di initiateAuthentication.
- Validità della chiave pubblica del Root della certificate chain (opzionale): non implementato.
- Eventuali altri controlli previsti per l'LPA non sono implementati.

Tabella 4.1: Librerie Python utilizzate.

Libreria	**Utilizzo**
asn1crypto	Viene usata a supporto della generazione e della verifica delle signature.
asn1tools	È la libreria di riferimento per convertire i dati in e dal formato ASN.1; prevede l'ausilio di alcuni file contenenti, tra le varie cose, le definizioni ASN.1 che vengono usate [16].
base64	Viene usata per convertire i dati in e dal formato base64.
cryptography	Viene usata a supporto della generazione dei certificati custom, della generazione e della verifica delle signature e di altre funzionalità crittografiche.
datetime	Viene usata per definire delle date o degli istanti di tempo; è utile ad esempio per inizializzare i campi *not valid before* e *not valid after* dei certificati custom.
gzip	Viene usata per estrarre il payload dei messaggi eventualmente compressi del server SM-DP+ (non è il caso di Very Mobile).
hashlib	Viene usata per generare fingerprint ad esempio con la funzione hash SHA256().
http	Viene sfruttata per utilizzare la parte client e/o la parte server per la comunicazione HTTP.
json	Viene usata per convertire i dati in e dal formato JSON, oltre che per leggere e scrivere sui file .json.
os	Viene usata per interagire col sistema operativo.
random	Viene usata per generare valori o stringhe in modo pseudo-randomico, come ad esempio i Matching ID.
requests	Viene usata in alternativa alla libreria http, in particolar modo per utilizzare la parte client per la comunicazione HTTP quando si ha a che fare con un server che con la libreria http dà problemi.
secrets	Viene usata per generare sequenze di byte pseudo-randomiche, come ad esempio le challenge.
signal	Viene usata per definire un timeout ove è previsto un input da parte dell'utente da immettere entro un tempo limitato.
six	Viene usata per verificare se determinate variabili sono istanze di uno specifico tipo di dato.
socket	Viene usata per definire e utilizzare le socket dalle quali verrà istanziata la connessione HTTP.
ssl	Viene usata per aggiungere la componente SSL (o TLS) alla connessione HTTP.
sslkeylog	Viene usata per generare un log su ciò che avviene all'interno di una connessione di rete.
string	Viene usata per la definizione di stringhe o di insiemi di caratteri; ad esempio, può indicare il set dei possibili caratteri che potenzialmente comporranno una stringa pseudo-randomica.
sys	Viene usata per interagire con l'ambiente runtime di Python.

Check da parte dell'eUICC alla ricezione di initiateAuthenticationResponse

- Validità della catena di certificati: l'implementazione prevede un controllo sul certificato CERT.DPauth.SIG mediante la chiave pubblica del CI (i.e. del Root).
- Validità di serverSignature1: l'implementazione prevede un controllo della signature contenuta nel messaggio mediante la chiave pubblica associata al certificato CERT.DPauth.SIG.
- Validità di euiccChallenge: l'implementazione prevede un controllo su se l'euiccChallenge ricevuta corrisponde a quella inviata nel precedente messaggio di initiateAuthentication.
- Supporto della chiave pubblica del CI ricevuta nel campo euiccCiPKIdToBeUsed: l'implementazione prevede un controllo su se euiccCiPKIdToBeUsed è incluso all'interno del campo euiccCiPKIdListForSigning.
- Validità della CRL (opzionale): non implementato.

È stato aggiunto anche un controllo su se il campo transactionId di initiateAuthenticationResponse corrisponde al sotto-campo transactionId della struttura serverSigned1 contenuta nel medesimo messaggio.

Check da parte del server alla ricezione di authenticateClient

- Validità dell'ID di transazione: l'implementazione prevede un controllo su se il campo transactionId di authenticateClient e il sotto-campo transactionId della struttura euiccSigned1 sono uguali all'ID di transazione originariamente comunicato dal server.
- Validità del Root della certificate chain usata dal client: l'implementazione prevede un controllo su se il campo *AuthorityKeyIdentifier* del certificato CERT.EUM.SIG corrisponde alla variabile euiccCiPKIdToBeUsed.
- Validità della catena di certificati: l'implementazione prevede un controllo sul certificato CERT.EUM.SIG mediante la chiave pubblica del CI, ma anche un controllo sul certificato CERT.EUICC.SIG mediante la chiave pubblica dell'EUM.
- Validità di euiccSignature1: l'implementazione prevede un controllo della signature contenuta nel messaggio mediante la chiave pubblica associata al certificato CERT.EUICC.SIG.
- Validità di serverChallenge: l'implementazione prevede un controllo su se la serverChallenge ricevuta corrisponde a quella inviata nel precedente messaggio di initiateAuthenticationResponse.
- Validità del Matching ID: l'implementazione prevede un controllo su se il campo matchingId della struttura ctxParams1 corrisponde a un Matching ID realmente esistente per il server e correntemente nello stato 'Released' o 'Download'.

A livello implementativo sono stati aggiunti anche i controlli elencati qui di seguito.

- Validità dell'indirizzo SM-DP+: si ha un controllo su se il campo serverAddress della struttura euiccSigned1 è uguale all'hostname del server.
- Validità dei campi di euiccInfo2 in comune con euiccInfo1: si ha un controllo su se i campi svn, euiccCiPKIdListForVerification ed euiccCiPKIdListForSigning assumono gli stessi valori in euiccInfo2 all'interno di authenticateClient e in euiccInfo1 all'interno di initiateAuthentication.

Check da parte dell'LPA alla ricezione di authenticateClientResponse

- Verifica della RAT: l'implementazione prevede una funzione che segue la logica introdotta nella sezione 2.5.1.
- Conferma da parte dell'utente: l'implementazione prevede due casi. Nel primo caso è richiesta l'acquisizione del Confirmation Code da parte dell'utente e, se questo è errato o scatta il timeout, viene invocata la procedura di Common Cancel Session. Nel secondo caso è richiesta la conferma semplice e, se l'utente risponde 'No', 'Not now' oppure fa scadere il timeout, viene invocata la procedura di Common Cancel Session.

Check da parte dell'eUICC alla ricezione di authenticateClientResponse

- Validità di CERT.DPpb.SIG: l'implementazione prevede un controllo su CERT.DPpb.SIG mediante la chiave pubblica del CI.
- Appartenenza di CERT.DPpb.SIG e di CERT.DPauth.SIG alla medesima entità: l'implementazione prevede un controllo su se l'Organization Name di CERT.DPpb.SIG corrisponde all'Organization Name di CERT.DPauth.SIG, e un controllo su se l'Organization Name dell'issuer di CERT.DPpb.SIG corrisponde all'Organization Name dell'issuer di CERT.DPauth.SIG.
- Validità di smdpSignature2: l'implementazione prevede un controllo della signature contenuta nel messaggio mediante la chiave pubblica associata al certificato CERT.DPpb.SIG.

- Validità dell'ID di transazione: l'implementazione prevede un controllo su se il campo transactionId di authenticateClientResponse e il sotto-campo transactionId della struttura smdpSigned2 sono uguali all'ID di transazione che identifica la sessione RSP corrente.

È stato aggiunto anche un controllo su se i campi serviceProviderName e profileName della struttura profileMetadata corrispondono a stringhe non vuote.

Check da parte del server alla ricezione di getBoundProfilePackage

- Validità di euiccSignature2: l'implementazione prevede un controllo della signature contenuta nel messaggio mediante la chiave pubblica associata al certificato CERT.EUICC.SIG.
- Validità del Confirmation Code (opzionale): l'implementazione prevede un controllo sull'hash del Confirmation Code ricevuto dal server; se il check non va a buon fine, è previsto un incremento del contatore dei tentativi di immissione del codice e un controllo su se il contatore ha sforato il numero massimo ammissibile.

È stato aggiunto anche un controllo su se il campo transactionId di getBoundProfilePackage e il sotto-campo transactionId della struttura euiccSigned2 sono uguali all'ID di transazione che identifica la sessione RSP corrente.

Check da parte del client alla ricezione di getBoundProfilePackageResponse

In questa fase, è stato deciso di sintetizzare tutti i check svolti da parte dell'LPA e dell'eUICC in cinque punti fondamentali.

- Validità del Bound Profile Package: l'implementazione prevede un controllo in modo particolare sul campo sequenceOf88 della struttura boundProfilePackage per verificare se corrisponde alla codifica ASN.1 dell'intero campo profileMetadata ricevuto nel precedente messaggio del server (authenticateClientResponse).
- Validità del Remote Operation Type: l'implementazione prevede un controllo su se il campo remoteOpId della struttura initialiseSecureChannelRequest corrisponde a un'operazione remota (Remote Operation) supportata.
- Validità dell'ID di transazione: l'implementazione prevede un controllo su se il campo transactionId di getBoundProfilePackageResponse e il sotto-campo transactionId della struttura initialiseSecureChannelRequest sono uguali all'ID di transazione che identifica la sessione RSP corrente.
- Validità della Profile Class: l'implementazione prevede un controllo su se la classe del profilo target (Profile Class) è supportata.
- Validità di smdpSign: l'implementazione prevede un controllo della signature contenuta nella struttura initialiseSecureChannelRequest del messaggio mediante la chiave pubblica associata al certificato CERT.DPpb.SIG.

Check da parte del server alla ricezione di handleNotification

Nonostante la sezione 2.4.7 non parli esplicitamente di controlli effettuati dal server SM-DP+ dopo la ricezione del messaggio handleNotification da parte dell'LPA, si è comunque deciso di inserire due check che sembravano ormai ovvi e doverosi.

- Validità dell'ID di transazione: l'implementazione prevede un controllo su se il campo transactionId della struttura profileInstallationResultData è uguale all'ID di transazione che identifica la sessione RSP corrente.
- Validità di euiccSignPIR: l'implementazione prevede un controllo della signature contenuta nel messaggio mediante la chiave pubblica associata al certificato CERT.EUICC.SIG.

Check da parte del server alla ricezione di cancelSession

All'interno del sistema simulato, il client avvia la procedura di Common Cancel Session nei casi riportati qui di seguito, come suggerito dalle specifiche di GSMA [3]:

- il messaggio authenticateClientResponse da parte del server contiene un solo campo (transactionId);
- il messaggio authenticateClientResponse contiene la struttura profileMetadata con serviceProviderName o profileName pari alla stringa vuota;
- fallisce un qualche controllo sulle Profile Policy Rules all'interno dei metadati del profilo (struttura profileMetadata);
- l'end user non inserisce correttamente in input l'eventuale Confirmation Code o comunque non fornisce la conferma sul download del profilo target;
- il campo sequenceOf88 della struttura boundProfilePackage all'interno del messaggio getBoundProfilePackageResponse non corrisponde alla codifica ASN.1 dell'intero campo profileMetadata del precedente messaggio di authenticateClientResponse.

Quando il server riceve un messaggio di cancelSession all'interno del sistema simulato, vengono implementati i seguenti check:

- Validità dell'ID di transazione: l'implementazione prevede un controllo su se il campo transactionId di cancelSession e il sotto-campo transactionId della struttura euiccCancelSessionSigned sono uguali all'ID di transazione che identifica una sessione RSP correntemente attiva.
- Validità di euiccCancelSessionSignature: l'implementazione prevede un controllo della signature contenuta nel messaggio mediante la chiave pubblica associata a CERT.EUICC.SIG.

L'unico aspetto da puntualizzare è la mancanza di un controllo sull'OID.

4.5 Validazione dei simulatori

Prima di sfruttare i simulatori implementati per condurre gli esperimenti successivi, è necessario dimostrare che essi siano corretti, assicurandosi che rispettino fedelmente le specifiche di GSMA, siano privi di bug e siano in grado di comunicare con successo con server e client reali. In altre parole, si vuole validare i simulatori.

4.5.1 Validazione del simulatore lato client

Per validare il simulatore del client, si è deciso di farlo comunicare con un server di test, ovvero un server SM-DP+ che non è stato concepito per rilasciare profili reali a utenti che vogliono ottenere un vero e proprio piano tariffario, bensì per effettuare delle prove e rilasciare quindi dei profili di test. Tipicamente, un server di test prevede l'utilizzo di catene di certificati di test, le più importanti delle quali vengono definite e messe a disposizione direttamente dalla GSMA [17]. Si tratta di due catene di certificati insieme alle quali è possibile scaricare anche le relative chiavi pubbliche e chiavi private. Inoltre, tali certificati coprono tutte le entità coinvolte nella catena, dalla RootCA, al server SM-DP+, all'EUM, all'eUICC. La prima certificate chain, indicata con BRP, fa uso dell'algoritmo Brainpool, che è un algoritmo utilizzato nell'ambito della crittografia basata sulle curve ellittiche [18], mentre la seconda, indicata con NIST, no.
Per aiutarsi con l'attività di debugging durante il processo di validazione del simulatore lato client, si è deciso di sfruttare un server di test open-source: **Osmo-smdpp**, che è possibile trovare su GitHub al link https://github.com/osmocom/pysim, ed è stato l'unico server SM-DP+ open-source reperibile al momento della validazione del simulatore. Si tratta di un server di test attualmente incompleto. Infatti, ha implementate tutte le funzioni necessarie per la mutua autenticazione e per il download dei profili di prova ma non prevede ancora tutti i controlli che, secondo le specifiche di GSMA [3], un qualunque server SM-DP+ deve effettuare a ogni messaggio ricevuto da parte dell'LPA coinvolta nella comunicazione; ciò è confermato dal fatto che all'interno del codice di Osmo-smdpp vi è un'ingente quantità di commenti marcati con l'espressione *TODO*. Comunque

Figura 4.23: Lettore di SIM utilizzato per validare il simulatore del server.

sia, per semplicità, si è assunto che tale server di test fosse sufficiente per debuggare, completare e validare il simulatore del client appena sviluppato.
Il simulatore è stato approvato, e quindi considerato valido, nel momento in cui è stato in grado di completare con successo la comunicazione col server Osmo-smdpp inviando nell'ordine i messaggi initiateAuthentication, authenticateClient, getBoundProfilePackage e handleNotification, potendo sfruttare sia la catena di certificati BRP, sia la catena di certificati NIST.

4.5.2 Validazione del simulatore lato server

Per validare il simulatore del server, si è deciso di farlo comunicare con un client di test, che è composto da un'eUICC programmabile della Sysmocom. Tale eUICC può essere acquistata al sito https://shop.sysmocom.de e può essere inserita all'interno di un apposito lettore di SIM, il quale è direttamente collegabile al computer con cui si sta lavorando grazie al suo attacco USB ed è riportato in figura 4.23. Inoltre, l'eUICC è pensata per comunicare col server di Osmocom (Osmo-smdpp) e, per questa ragione, supporta anch'essa la catena di certificati di test definita dalla GSMA, in particolare la NIST, che non fa uso dell'algoritmo Brainpool.
Per testare il simulatore del server, è stato sufficiente installare e compilare un'apposita LPA di test (**lpac**) reperibile su GitHub al link https://github.com/estkme-group/lpac, collegare al computer il lettore di SIM con all'interno l'eUICC programmabile e avviare sia il simulatore del server che LPA di test in modo tale da farli comunicare tra loro.
Il simulatore viene considerato valido nel momento in cui si riesce a installare con successo il profilo da lui offerto all'interno della SIM programmabile. Tuttavia, almeno per il momento, l'attività di debugging non ha consentito di raggiungere questo risultato: allo stato attuale, il simulatore è in grado di scambiare tutti i messaggi col client di test ma, dopo aver ricevuto il Bound Profile Package nel messaggio getBoundProfilePackageResponse, fornisce l'errore *scp03tSecurityError* e non installa effettivamente il profilo all'interno dell'eUICC programmabile. Per questo motivo, il simulatore del server risulta validato solo parzialmente ma, grazie alla capacità di portare a termine la comunicazione coi client, rimane comunque uno strumento sufficientemente adeguato per intraprendere lo studio successivo con i client reali.

Capitolo 5

Analisi di sicurezza dell'eSIM

5.1 Descrizione dell'analisi

L'analisi di sicurezza che ci si propone di effettuare si articola in due fasi.

1. Si individua un campione di server SM-DP+ da testare e, mediante il simulatore del client precedentemente sviluppato, si tenta di comunicare con ciascuno di loro in svariati modi diversi (e.g. modificando qualche campo di un messaggio, cambiando la certificate chain utilizzata, cambiando l'ordine dei messaggi inviati), costruendo così altrettanti casi di test. Per ciascun caso di test si analizzano le risposte di ogni server SM-DP+ del campione, col fine ultimo di stabilire se esse sono perfettamente conformi alle specifiche oppure lasciano intendere che il server abbia omesso dei check, aprendo le porte a eventuali vulnerabilità.

2. Si utilizzano dei dispositivi mobili (i.e. coppie eUICC - LPA) per definire ulteriori casi di test a parti invertite. Così, sfruttando il simulatore del server implementato in precedenza, si tenta di comunicare coi dispositivi mobili e si raccolgono i loro messaggi di risposta per ciascun caso di test, anche qui con lo scopo di verificare se i loro eUICC e le loro LPA sono conformi alle specifiche.

5.2 Ottenimento dei server da testare

L'idea originale consisteva nell'individuare tutti i server SM-DP+ del mondo in modo tale da effettuare dei test esaustivi e su larga scala. Tuttavia, non è stato possibile trovare in rete alcun riferimento a hostname o indirizzi IP dei server SM-DP+, eccezion fatta per i server di test e per alcuni server SM-DS. I server di test trovati sono quello di Google (con hostname *prod.smdp-plus.rsp.goog*), Infineon (con hostname *testsmdpplus.infineon.com*), Sysmocom (con hostname *smdpp.test.rsp.sysmocom.de*) e Truphone (con hostname *rsp.truphone.com*). I server SM-DS trovati sono quello di GSMA (con hostname *lpa.ds.gsma.com*), eSIM Discovery Production (con hostname *lpa.live.esimdiscovery.com*), eSIM Discovery Staging (con hostname *lpa.live.esimdiscovery.dev*), Stork (con hostname *prod.smds.rsp.goog*) e Verizon Wireless (con hostname *lpads.vzw.otgeuicc.com*). Tutti questi server, a differenza di Osmo-smdpp, non sono open-source e non devono essere installati sulla macchina locale per essere eseguiti, bensì possono essere visti come normali server SM-DP+. Per trovare dei server SM-DP+ che siano affiliati a degli operatori e permettano realmente di effettuare il download di profili, è stato necessario procedere come per il server di Very Mobile (vedere capitolo 4), ovvero acquistare un piano tariffario reale su un telefono che supporta l'eSIM, ricavando così l'hostname del server SM-DP+ a partire da un'email ricevuta dall'operatore (come quella in figura 4.13). Sono stati così individuati i seguent server SM-DP+:

- server di Iliad, con hostname *frm.prod.ondemandconnectivity.com*;
- server di Tim, con hostname *rsp-0001.oberthur.net*;
- server di Very Mobile, con hostname *sys.prod.ondemandconnectivity.com*.

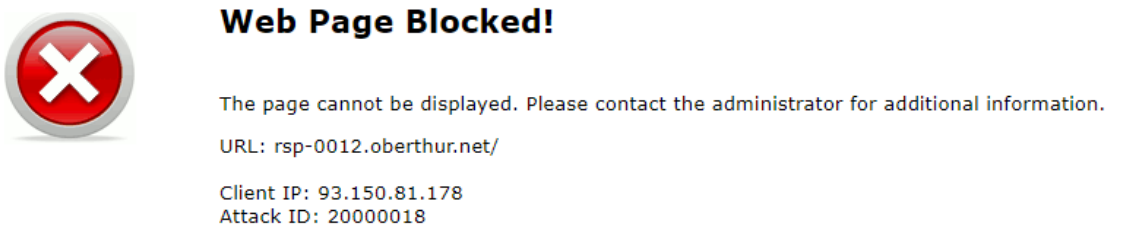


Figura 5.1: Pagina web con cui rsp-0012.oberthur.net risponde.

Successivamente, è stato effettuato un altro tentativo per ricavare l'hostname del maggior numero di server SM-DP+ possibile, che consiste nel provare a sfruttare l'indirizzo IP dei server già noti. In particolare, tramite il comando *dig*, sono state inviate delle query a un server DNS con lo scopo di ottenere l'indirizzo IP associato a ciascun hostname ottenuto, compreso quello dei server di test. La tabella 5.1 mette insieme hostname e indirizzo IP dei server SM-DP+ attualmente noti.

Tabella 5.1: Hostname e indirizzo IP dei server SM-DP+ noti.

Nome	**Hostname**	**Indirizzo IP**
Iliad	frm.prod.ondemandconnectivity.com	34.107.65.70
Tim	rsp-0001.oberthur.net	51.11.249.51
Very Mobile	sys.prod.ondemandconnectivity.com	34.107.65.70
eSIM Disc. Production	lpa.live.esimdiscovery.com	34.48.45.29
eSIM Disc. Staging	lpa.live.esimdiscovery.com	35.245.56.247
GSMA	lpa.ds.gsma.com	34.89.151.119
Stork	prod.smds.rsp.goog	34.95.109.237
Verizon Wireless	lpads.vzw.otgeuicc.com	35.245.232.18
Google	prod.smdp-plus.rsp.goog	34.95.109.237
Infineon	testsmdpplus.infineon.com	52.57.244.228; 18.196.185.236
Sysmocom	smdpp.test.rsp.sysmocom.de	213.95.46.137
Truphone	rsp.truphone.com	185.99.24.88

Dalla tabella 5.1 è possibile concludere che non c'è un pattern negli indirizzi IP dei server SM-DP+, per cui è impensabile ricavare ulteriori indirizzi IP a partire da quelli noti. Inoltre, è possibile osservare che gli operatori Iliad e Very Mobile si appoggiano sul medesimo server SM-DP+, così come Stork e Google.
Si potrebbe pensare di procedere con la forza bruta e, per ogni indirizzo IP che va da 0.0.0.0 a 255.255.255.255, provare a stabilire se esso è associato o meno a un server SM-DP. Tuttavia, né inviando query a un server DNS, né provando a contattare direttamente gli host specificando il loro indirizzo IP anziché il loro hostname si riescono a ottenere informazioni utili.
L'unico aspetto su cui si è potuto lavorare è l'hostname del server SM-DP+ associato a Tim, dove compare un numero N a quattro cifre (0001). Facendo variare N, si ottengono ancora degli hostname esistenti e validi per qualunque N a quattro cifre compreso tra 0001 e 0032. In particolare, i 32 hostname così trovati sono mappati soltanto su quattro indirizzi IP diversi, per cui molti di loro costituiscono degli alias. I quattro host individuati solo elencati di seguito:

- *rsp-0001.oberthur.net*, con indirizzo IP 51.11.249.51;
- *rsp-0002.oberthur.net*, con indirizzo IP 51.11.249.55;
- *rsp-0003.oberthur.net*, con indirizzo IP 185.200.48.11;
- *rsp-0012.oberthur.net*, con indirizzo IP 213.39.85.20.

Stabilendo una prima connessione con ciascuno dei quattro host, è stato possibile capire che dietro gli hostname *rsp-0001.oberthur.net* e *rsp-0002.oberthur.net* si cela con ogni probabilità il medesimo server SM-DP+, poiché nei due casi ha esibito esattamente lo stesso certificato CERT.DPauth.SIG. Inoltre, dietro l'hotname *rsp-0012.oberthur.net* si cela un server non più utilizzabile perché, indipendentemente dal messaggio di richiesta, risponde con l'errore "Web Page Blocked!" come riportato

in figura 5.1. Perciò, ai fini degli esperimenti, oltre a *rsp-0001.oberthur.net* (server di Tim), verrà considerato solo *rsp-0003.oberthur.net*, che verrà identificato come 'gemello del server di Tim'. La tabella 5.2 mostra tutti i server che verranno contattati durante l'esecuzione dei test.

Tabella 5.2: Elenco dei server SM-DP+ coinvolti negli esperimenti.

Nome	**Descrizione**	**Note**
Iliad & Very Mobile	Server SM-DP+ reale	Verrà contattato usando una certificate chain custom con CI = GSMA*.
Tim	Server SM-DP+ reale	Verrà contattato usando una certificate chain custom con CI = GSMA*.
Gemello di Tim	Server SM-DP+ reale	Verrà contattato usando una certificate chain custom con CI = GSMA*.
eSIM Disc. Production	Server SM-DS	Verrà contattato usando una certificate chain custom con CI = GSMA*.
eSIM Disc. Staging	Server SM-DS	Verrà contattato usando una certificate chain custom con CI = GSMA*.
GSMA	Server SM-DS	Verrà contattato usando una certificate chain custom con CI = GSMA*.
Verizon Wireless	Server SM-DS	Verrà contattato usando una certificate chain custom con CI = GSMA*.
Stork & Google	Server SM-DS / SM-DP+ di test	Verrà contattato usando una certificate chain di test**. Matching ID: 052X-UFXS-CQIY-PNGL.
Infineon	Server SM-DP+ di test	Verrà contattato usando una certificate chain di test. Matching ID: ApcLl-k0Ulo-kqSlk-GLhCl.
Sysmocom	Server SM-DP+ di test	Verrà contattato usando una certificate chain di test. Matching ID: TS48v1_A.
Truphone	Server SM-DP+ di test	Verrà contattato usando una certificate chain custom con CI = GSMA*. Matching ID: QR-G-5C-1LS-1W1Z9P7.
Osmo-smdpp	Server SM-DP+ di test open-source installato in locale	Verrà contattato usando una certificate chain di test. Matching ID: TS48v1_A.
Simulatore	Simulatore lato server precedentemente implementato	Può essere contattato usando una certificate chain qualsiasi e un Matching ID custom.

**Non è possibile disporre di un'intera certificate chain valida della GSMA assieme a tutte le chiavi private associate. È per questo motivo che il certificato presentato dal simulatore del client all'interno del messaggio authenticateClient non potrebbe essere considerato valido. Dunque, i successivi esperimenti non possono spingersi oltre alla procedura di Common Mutual Authentication.*

***La certificate chain di test supportata dal server di Stork e Google è diversa da quella prevista dagli altri server di test.*

5.3 Risposte dei server testati

5.3.1 Casi di test definiti

I casi di test che sono stati implementati ed eseguiti ai fini della presente trattazione sono riportati nella tabella 5.3.

Tabella 5.3: Elenco dei casi di test definiti per i server SM-DP+.

ID	Descrizione
1	Il simulatore del client si comporta perfettamente come da specifiche.
2	Manomissione del messaggio authenticateClient: invio della struttura authenticateResponseError anziché authenticateResponseOk.
3	Manomissione del messaggio authenticateClient: utilizzo di una certificate chain lato client differente da quella scelta di default.
4	Manomissione del messaggio authenticateClient: calcolo errato dalla signature (euiccSignature1).
5.1	Manomissione del messaggio authenticateClient (sotto-campo euiccInfo2): utilizzo di un SVN (Specification Version Number) non atteso (e.g. 0x000000, che corrisponderebbe alla versione 0.0.0).
5.2	Manomissione del messaggio authenticateClient (sotto-campo euiccInfo2): utilizzo di euiccCiPKIdListForVerification ed euiccCiPKIdListForSigning non attesi (pur mantenendo la certificate chain di default).
5.3	Manomissione del messaggio authenticateClient (sotto-campo euiccInfo2): sasAcreditationNumber = stringa vuota.
6.1	Manomissione del messaggio authenticateClient (sotto-campo ctxParams1): eliminazione del sotto-campo imei.
6.2	Manomissione del messaggio authenticateClient (sotto-campo ctxParams1): inserimento di un Matching ID non valido.
7.1	Manomissione del messaggio authenticateClient: utilizzo di un Transaction ID non valido all'interno del campo euiccSigned1.
7.2	Manomissione del messaggio authenticateClient: campo transactionId non valido direttamente all'interno di authenticateResponseOk.
8	Manomissione del messaggio authenticateClient: utilizzo di un smdpAddress non valido all'interno del campo euiccSigned1.
9	Manomissione del messaggio authenticateClient: utilizzo di una serverChallenge non valida all'interno del campo euiccSigned1.
10	Manomissione del messaggio initiateAuthentication: utilizzo di una euiccChallenge malforme (e.g. 0x00).
11	Manomissione del messaggio initiateAuthentication: utilizzo di un smdpAddress non valido.
12.1	Manomissione del messaggio initiateAuthentication (campo euiccInfo1): utilizzo di un SVN (Specification Version Number) non atteso (e.g. 0x000000, che corrisponderebbe alla versione 0.0.0).
12.2	Manomissione del messaggio initiateAuthentication (campo euiccInfo1): utilizzo di euiccCiPKIdListForVerification ed euiccCiPKIdListForSigning non attesi (pur mantenendo la certificate chain di default).
13.1	Avvio della Common Cancel Session Procedure già dal primo messaggio.
13.2	Avvio della Common Cancel Session Procedure a partire dal secondo messaggio.
14.1	Invio di getBoundProfilePackage come primo messaggio.
14.2	Invio di getBoundProfilePackage come secondo messaggio.
15	Invio di getBoundProfilePackage come primo messaggio col campo transactionId pari all'ID di transazione di una connessione instaurata da un altro processo client messo appositamente in sleep.

Caso di test 1

Consiste nel comunicare col server SM-DP+ seguendo fedelmente le specifiche, senza alcuna manomissione nel protocollo o nei messaggi inviati. Di seguito è descritto il comportamento dei server in esame in questo caso di test.

- **Iliad & Very Mobile**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "1.2"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "Function provider"*, *"message": "Internal error while verifying token signature: invalid Certificate Policies: [2.5.29.35, 2.23.146.1.2.1.2]"*.
- **Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **Gemello di Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **eSIM Discovery Production**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **eSIM Discovery Staging**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **GSMA**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Verizon Wireless**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Stork & Google**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"message": "Execution Failed"* → la verifica del certificato dell'EUM non è andata a buon fine.
- **Infineon**: HTTP_BAD_REQUEST (status code 400) al primo messaggio di risposta (initiateAuthenticationResponse).
- **Sysmocom**: *"status": "Executed-Success"*.
- **Truphone**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **Osmo-smdpp**: *"status": "Executed-Success"*.
- **Simulatore**: *"status": "Executed-Success"*.

Poiché il server Infineon dà problemi già dal primo messaggio (initiateAuthentication), non potrà essere oggetto dei casi di test che coinvolgono la manomissione del secondo messaggio (authenticateClient).

Caso di test 2

Consiste nel manomettere il messaggio authenticateClient inserendovi la struttura authenticateResponseError al posto di authenticateResponseOk, facendo credere al server SM-DP+ che la sua autenticazione non sia andata a buon fine.

- **Iliad & Very Mobile**: HTTP_NO_CONTENT (status code 204) al secondo messaggio di risposta (authenticateClientResponse).
- **Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.8"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "SMDP+"*, *"message": "Verification Failed"*.

- **Gemello di Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.8"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "SMDP+"*, *"message": "Verification Failed"*.
- **eSIM Discovery Production**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "eUICC"*, *"message": "serverChallenge is invalid"*.
- **eSIM Discovery Staging**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "eUICC"*, *"message": "serverChallenge is invalid"*.
- **GSMA**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "eUICC"*, *"message": "serverChallenge is invalid"*.
- **Verizon Wireless**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "eUICC"*, *"message": "serverChallenge is invalid"*.
- **Stork & Google**: Common Cancel Session Procedure completata con successo a seguito di un *"status": "Executed-Success"* al secondo messaggio di risposta (authenticateClientResponse).
- **Infineon**: n.d.
- **Sysmocom**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Truphone**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "1.6"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "Function"*, *"message": "Execution Error"*.
- **Osmo-smdpp**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Simulatore**: secondo messaggio di risposta (authenticateClientResponse) vuoto.

Caso di test 3

Consiste nel manomettere il messaggio authenticateClient inserendo all'interno dei campi euiccCertificate ed eumCertificate dei certificati appartenenti a una certificate chain differente da quella dichiarata precedentemente in euiccCiPKIdListForVerification ed euiccCiPKIdListForSigning.

- **Iliad & Very Mobile**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "1.2"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "Function provider"*, *"message": "Internal error while verifying token signature: "*.
- **Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.11.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "GSMA CI"*, *"message": "Unknown"* → chiave pubblica del CI sconosciuta: il CI utilizzato dal certificato dell'EUM non è una Root trusted per il server SM-DP+.
- **Gemello di Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.11.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "GSMA CI"*, *"message": "Unknown"* → chiave pubblica del CI sconosciuta: il CI utilizzato dal certificato dell'EUM non è una Root trusted per il server SM-DP+.
- **eSIM Discovery Production**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.11.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "CI Public Key"*, *"message": "Unknown CI Public Key. The CI used by the EUM Certificate is not a trusted root for the SM-DS"*.

- **eSIM Discovery Staging**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.11.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "CI Public Key"*, *"message": "Unknown CI Public Key. The CI used by the EUM Certificate is not a trusted root for the SM-DS"*.

- **GSMA**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.11.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "CI Public Key"*, *"message": "Unknown CI Public Key. The CI used by the EUM Certificate is not a trusted root for the SM-DS"*.

- **Verizon Wireless**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.11.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "CI Public Key"*, *"message": "Unknown CI Public Key. The CI used by the EUM Certificate is not a trusted root for the SM-DS"*.

- **Stork & Google**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.11.1"*, *"reasonCode": "3.9"*, *"message": "Execution Failed"* → chiave pubblica del CI sconosciuta: il CI utilizzato dal certificato dell'EUM non è una Root trusted per il server SM-DP+.

- **Infineon**: n.d.

- **Sysmocom**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.11.1"*, *"reasonCode": "3.9"*, *"message": "Unknown"* → chiave pubblica del CI sconosciuta: il CI utilizzato dal certificato dell'EUM non è una Root trusted per il server SM-DP+.

- **Truphone**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.11.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "Publick Key (PK)"*, *"message": "Unknown"*.

- **Osmo-smdpp**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.11.1"*, *"reasonCode": "3.9"*, *"message": "Unknown"* → chiave pubblica del CI sconosciuta: il CI utilizzato dal certificato dell'EUM non è una Root trusted per il server SM-DP+.

- **Simulatore**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.3"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "eUICC Certificate"*, *"message": "Verification Failed"* → il certificato dell'eUICC o un qualunque certificato nella catena è non valido.

Caso di test 4

Consiste nel manomettere il messaggio authenticateClient ponendo il campo euiccSignature1 pari a un valore diverso dalla signature ottenuta firmando il campo euiccSigned1 con la chiave privata dell'eUICC.

- **Iliad & Very Mobile**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "1.2"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "Function provider"*, *"message": "Internal error while verifying token signature: invalid Certificate Policies: [2.5.29.35, 2.23.146.1.2.1.2]"*.

- **Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.

- **Gemello di Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.

- **eSIM Discovery Production**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).

- **eSIM Discovery Staging**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **GSMA**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Verizon Wireless**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Stork & Google**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"message": "Execution Failed"* → la verifica del certificato dell'EUM non è andata a buon fine.
- **Infineon**: n.d.
- **Sysmocom**:
  - Caso in cui la lunghezza della signature è corretta → *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1"*, *"reasonCode": "6.1"*, *"message": "Verification failed"* → la signature dell'eUICC è non valida.
  - Caso in cui la lunghezza della signature è errata → HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Truphone**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **Osmo-smdpp**:
  - Caso in cui la lunghezza della signature è corretta → *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1"*, *"reasonCode": "6.1"*, *"message": "Verification failed"* → la signature dell'eUICC è non valida.
  - Caso in cui la lunghezza della signature è errata → HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Simulatore**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "eUICC"*, *"message": "Verification Failed"* → la signature dell'eUICC è non valida.

Caso di test 5.1

Consiste nel manomettere il messaggio authenticateClient ponendo l'SVN (Specification Version Number) all'interno della struttura euiccInfo2 del campo euiccSigned1 pari a un valore inatteso o non valido (come ad esempio 0x000000 che corrisponderebbe alla versione 0.0.0, oppure 0x0000 che dovrebbe avere una lunghezza non convenzionale, ovvero diversa da tre byte).

- **Iliad & Very Mobile**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "1.2"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "Function provider"*, *"message": "Internal error while verifying token signature: invalid Certificate Policies: [2.5.29.35, 2.23.146.1.2.1.2]"*.
- **Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **Gemello di Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **eSIM Discovery Production**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).

- **eSIM Discovery Staging**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **GSMA**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Verizon Wireless**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Stork & Google**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"message": "Execution Failed"* → la verifica del certificato dell'EUM non è andata a buon fine.
- **Infineon**: n.d.
- **Sysmocom**: *"status": "Executed-Success"*.
- **Truphone**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **Osmo-smdpp**: *"status": "Executed-Success"*.
- **Simulatore**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1"*, *"reasonCode": "3.11"*, *"subjectIdentifier": "eUICC"*, *"message": "Value has Changed"*.

Caso di test 5.2

Consiste nel manomettere il messaggio authenticateClient ponendo euiccCiPKIdListForVerification ed euiccCiPKIdListForSigning all'interno della struttura euiccInfo2 del campo euiccSigned1 pari a un valore inatteso o comunque differente dall'identificatore della RootCA della certificate chain utilizzata dal simulatore del client.

- **Iliad & Very Mobile**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "1.2"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "Function provider"*, *"message": "Internal error while verifying token signature: invalid Certificate Policies: [2.5.29.35, 2.23.146.1.2.1.2]"*.
- **Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **Gemello di Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **eSIM Discovery Production**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **eSIM Discovery Staging**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **GSMA**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Verizon Wireless**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Stork & Google**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"message": "Execution Failed"* → la verifica del certificato dell'EUM non è andata a buon fine.
- **Infineon**: n.d.

- **Sysmocom**: *"status": "Executed-Success"*.
- **Truphone**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **Osmo-smdpp**: *"status": "Executed-Success"*.
- **Simulatore**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.8.2"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Security configuration"*, *"message": "Unsupported"*.

Caso di test 5.3

Consiste nel manomettere il messaggio authenticateClient ponendo sasAcreditationNumber all'interno della struttura euiccInfo2 del campo euiccSigned1 pari alla stringa vuota.

- **Iliad & Very Mobile**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "1.2"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "Function provider"*, *"message": "Internal error while verifying token signature: invalid Certificate Policies: [2.5.29.35, 2.23.146.1.2.1.2]"*.
- **Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **Gemello di Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **eSIM Discovery Production**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **eSIM Discovery Staging**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **GSMA**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Verizon Wireless**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Stork & Google**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"message": "Execution Failed"* → la verifica del certificato dell'EUM non è andata a buon fine.
- **Infineon**: n.d.
- **Sysmocom**: *"status": "Executed-Success"*.
- **Truphone**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **Osmo-smdpp**: *"status": "Executed-Success"*.
- **Simulatore**: *"status": "Executed-Success"*.

Caso di test 6.1

Consiste nel manomettere il messaggio authenticateClient eliminando l'imei dalla struttura ctxParams1 del campo euiccSigned1. Trattandosi di un'informazione opzionale, questo caso di test non dovrebbe produrre un risultato differente dal caso di test 1.

- **Iliad & Very Mobile**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "1.2"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "Function provider"*, *"message": "Internal error while verifying token signature: invalid Certificate Policies: [2.5.29.35, 2.23.146.1.2.1.2]"*.
- **Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **Gemello di Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **eSIM Discovery Production**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **eSIM Discovery Staging**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **GSMA**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Verizon Wireless**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Stork & Google**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"message": "Execution Failed"* → la verifica del certificato dell'EUM non è andata a buon fine.
- **Infineon**: n.d.
- **Sysmocom**: *"status": "Executed-Success"*.
- **Truphone**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **Osmo-smdpp**: *"status": "Executed-Success"*.
- **Simulatore**: *"status": "Executed-Success"*.

Caso di test 6.2

Consiste nel manomettere il messaggio authenticateClient ponendo il Matching ID all'interno della struttura ctxParams1 del campo euiccSigned1 pari a un valore non valido. Questo caso di test è applicabile solo ai server SM-DP+ di cui si conosce un profilo valido scaricabile che, come specificato nella tabella 5.2, sono Stork+Google, Infineon, Sysmocom, Truphone, Osmo-smdpp e il simulatore. Tuttavia, poiché il server di Infineon, nelle condizioni descritte, non permette nemmeno di inviare un secondo messaggio di richiesta (authenticateClient), gli unici server SM-DP+ su cui in definitiva è possibile eseguire il presente caso di test sono Stork+Google, Sysmocom, Truphone, Osmo-smdpp e il simulatore.

- **Iliad & Very Mobile**: n.d.
- **Tim**: n.d.
- **Gemello di Tim**: n.d.
- **eSIM Discovery Production**: n.d.

- **eSIM Discovery Staging**: n.d.
- **GSMA**: n.d.
- **Verizon Wireless**: n.d.
- **Stork & Google**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"message": "Execution Failed"* → la verifica del certificato dell'EUM non è andata a buon fine.
- **Infineon**: n.d.
- **Sysmocom**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.2.6"*, *"reasonCode": "3.8"*, *"message": "Refused (profile not reliable)"*.
- **Truphone**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **Osmo-smdpp**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.2.6"*, *"reasonCode": "3.8"*, *"message": "Refused"* → il Matching ID non corrisponde a un valore valido.
- **Simulatore**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.2.6"*, *"reasonCode": "3.10"*, *"subjectIdentifier": "Matching ID"*, *"message": "Invalid Association"* → il Matching ID non corrisponde a quello effettivamente associato all'ICCID specificato.

Caso di test 7.1

Consiste nel manomettere il messaggio authenticateClient ponendo il Transaction ID del campo euiccSigned1 pari a un valore non valido.

- **Iliad & Very Mobile**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "1.2"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "Function provider"*, *"message": "Internal error while verifying token signature: invalid Certificate Policies: [2.5.29.35, 2.23.146.1.2.1.2]"*.
- **Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "Transaction ID"*, *"message": "Unknown"*.
- **Gemello di Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "Transaction ID"*, *"message": "Unknown"*.
- **eSIM Discovery Production**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **eSIM Discovery Staging**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **GSMA**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Verizon Wireless**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Stork & Google**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"message": "Execution Failed"* → la verifica del certificato dell'EUM non è andata a buon fine.
- **Infineon**: n.d.

- **Sysmocom**: *"status": "Executed-Success"*.
- **Truphone**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "TransactionId"*, *"message": "Unknown"*.
- **Osmo-smdpp**: *"status": "Executed-Success"*.
- **Simulatore**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "TransactionId"*, *"message": "Unknown"*.

Caso di test 7.2

Consiste nel manomettere il messaggio authenticateClient ponendo il Transaction ID pari a un valore non valido direttamente all'interno della struttura authenticateResponseOk.

- **Iliad & Very Mobile**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "<transactionId value>"*, *"message": "The transactionID is not correct"*.
- **Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "Transaction ID"*, *"message": "Unknown"*.
- **Gemello di Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "Transaction ID"*, *"message": "Unknown"*.
- **eSIM Discovery Production**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **eSIM Discovery Staging**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **GSMA**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Verizon Wireless**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Stork & Google**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"message": "Execution Failed"* → la verifica del certificato dell'EUM non è andata a buon fine.
- **Infineon**: n.d.
- **Sysmocom**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"message": "Unknown"* → la sessione RSP identificata dal Transaction ID è sconosciuta.
- **Truphone**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "TransactionId"*, *"message": "Unknown"*.
- **Osmo-smdpp**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"message": "Unknown"* → la sessione RSP identificata dal Transaction ID è sconosciuta.
- **Simulatore**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "TransactionId"*, *"message": "Unknown"*.

Caso di test 8

Consiste nel manomettere il messaggio authenticateClient ponendo il sotto-campo serverAddress del campo euiccSigned1 pari a una stringa che non corrisponde all'effettivo hostname del server SM-DP+ contattato.

- **Iliad & Very Mobile**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.8.1"*, *"reasonCode": "3.8"*, *"subjectIdentifier": "<smdpAddress value>"*, *"message": "The server address is not correct"*.
- **Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **Gemello di Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **eSIM Discovery Production**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **eSIM Discovery Staging**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **GSMA**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Verizon Wireless**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Stork & Google**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"message": "Execution Failed"* → la verifica del certificato dell'EUM non è andata a buon fine.
- **Infineon**: n.d.
- **Sysmocom**: *"status": "Executed-Success"*.
- **Truphone**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **Osmo-smdpp**: *"status": "Executed-Success"*.
- **Simulatore**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.8.1"*, *"reasonCode": "3.8"*, *"subjectIdentifier": "SM-DP+ Address"*, *"message": "Refused"*.

Caso di test 9

Consiste nel manomettere il messaggio authenticateClient ponendo il sotto-campo serverChallenge del campo euiccSigned1 pari a un valore che non corrisponde alla challenge inviata dal server nel messaggio precedente (initiateAuthenticationResponse).

- **Iliad & Very Mobile**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "1.2"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "Function provider"*, *"message": "Internal error while verifying token signature: invalid Certificate Policies: [2.5.29.35, 2.23.146.1.2.1.2]"*.
- **Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), senza ulteriori informazioni.
- **Gemello di Tim**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), senza ulteriori informazioni.

- **eSIM Discovery Production**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse
- **eSIM Discovery Staging**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse
- **GSMA**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse).
- **Verizon Wireless**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (authenticateClientResponse
- **Stork & Google**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"message": "Execution Failed"* → la verifica del certificato dell'EUM non è andata a buon fine.
- **Infineon**: n.d.
- **Sysmocom**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1"*, *"reasonCode": "6.1"*, *"message": "Verification failed"* → la signature dell'eUICC non è valida.
- **Truphone**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "EUM Certificate"*, *"message": "Verification Failed"*.
- **Osmo-smdpp**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1"*, *"reasonCode": "6.1"*, *"message": "Verification failed"* → la signature dell'eUICC non è valida.
- **Simulatore**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1"*, *"reasonCode": "6.1"*, *"subjectIdentifier": "eUICC"*, *"message": "Verification failed"*.

Caso di test 10

Consiste nel manomettere il messaggio initiateAuthentication utilizzando una euiccChallenge malforme (e.g. 0x00, che ha una lunghezza differente dai 16 byte di default, e comunque troppo ridotta per svolgere il ruolo di challenge).

- **Iliad & Very Mobile**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "1.6"*, *"reasonCode": "2.1"*, *"subjectIdentifier": "Function"*, *"message": "Euicc challenge format is not Octet[16]"*.
- **Tim**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "SM-DP+"*, *"message": "Execution Error"*.
- **Gemello di Tim**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "SM-DP+"*, *"message": "Execution Error"*.
- **eSIM Discovery Production**: invio di una euiccChallenge composta comunque da 16 byte nel primo messaggio di risposta (initiateAuthenticationResponse).
- **eSIM Discovery Staging**: invio di una euiccChallenge composta comunque da 16 byte nel primo messaggio di risposta (initiateAuthenticationResponse).
- **GSMA**: invio di una euiccChallenge composta comunque da 16 byte nel primo messaggio di risposta (initiateAuthenticationResponse).
- **Verizon Wireless**: invio di una euiccChallenge composta comunque da 16 byte nel primo messaggio di risposta (initiateAuthenticationResponse).

- **Stork & Google**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1.2"*, *"reasonCode": "6.1"*, *"message": "Execution Failed"* → la verifica del certificato dell'EUM non è andata a buon fine.
- **Infineon**: HTTP_BAD_REQUEST (status code 400) al primo messaggio di risposta (initiateAuthenticationResponse).
- **Sysmocom**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al primo messaggio di risposta (initiateAuthenticationResponse).
- **Truphone**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "1.6"*, *"reasonCode": "2.1"*, *"message": "Invalid"* → il formato di euiccChallenge è errato.
- **Osmo-smdpp**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al primo messaggio di risposta (initiateAuthenticationResponse).
- **Simulatore**: *"status": "Executed-Success"*.

Caso di test 11

Consiste nel manomettere il messaggio initiateAuthentication utilizzando un hostname (smdpAddress) che non corrisponde al server SM-DP+ contattato.

- **Iliad & Very Mobile**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.1"*, *"reasonCode": "3.8"*, *"subjectIdentifier": "<smdpAddress value>"*, *"message": "The SM-DP address sent by the LPA is unknown"*.
- **Tim**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.1"*, *"reasonCode": "3.8"*, *"subjectIdentifier": "SM-DP+ Address"*, *"message": "Refused"*.
- **Gemello di Tim**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.1"*, *"reasonCode": "3.8"*, *"subjectIdentifier": "SM-DP+ Address"*, *"message": "Refused"*.
- **eSIM Discovery Production**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.9.1"*, *"reasonCode": "3.8"*, *"subjectIdentifier": "SM-DS Address"*, *"message": "Invalid SM-DS Address"*.
- **eSIM Discovery Staging**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.9.1"*, *"reasonCode": "3.8"*, *"subjectIdentifier": "SM-DS Address"*, *"message": "Invalid SM-DS Address"*.
- **GSMA**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.9.1"*, *"reasonCode": "3.8"*, *"subjectIdentifier": "SM-DS Address"*, *"message": "Invalid SM-DS Address"*.
- **Verizon Wireless**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.9.1"*, *"reasonCode": "3.8"*, *"subjectIdentifier": "SM-DS Address"*, *"message": "Invalid SM-DS Address"*.
- **Stork & Google**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.1"*, *"reasonCode": "3.8"*, *"message": "Execution Failed"* → indirizzo del server SM-DP+ non valido.
- **Infineon**: HTTP_BAD_REQUEST (status code 400) al primo messaggio di risposta (initiateAuthenticationResponse).
- **Sysmocom**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.1"*, *"reasonCode": "3.8"*, *"message": "Invalid SM-DP+ Address"*.

- **Truphone**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.1"*, *"reasonCode": "3.8"*, *"subjectIdentifier": "SM-DP+ Address"*, *"message": "Refused"*.
- **Osmo-smdpp**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.1"*, *"reasonCode": "3.8"*, *"message": "Invalid SM-DP+ Address"*.
- **Simulatore**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.1"*, *"reasonCode": "3.8"*, *"subjectIdentifier": "SM-DP+ Address"*, *"message": "Refused"*.

Caso di test 12.1

Consiste nel manomettere il messaggio initiateAuthentication ponendo l'SVN (Specification Version Number) pari a un valore inatteso o non valido (come ad esempio 0x000000 che corrisponderebbe alla versione 0.0.0, oppure 0x0000 che dovrebbe avere una lunghezza non convenzionale, ovvero diversa da 3 byte).

- **Iliad & Very Mobile**:
  - Caso in cui la lunghezza dell'SVN è uguale a 3 byte → *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.3"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Specification Version Number"*, *"message": "The version SVN sent by the card is not in range [2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 3.0, 3.1, 3.2]"*.
  - Caso in cui la lunghezza dell'SVN è diversa da 3 byte → *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "1.6"*, *"reasonCode": "2.1"*, *"subjectIdentifier": "Function"*, *"message": "euiccInfo1 tlv is invalid"*.
- **Tim**:
  - Caso in cui la lunghezza dell'SVN è uguale a 3 byte → *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.3"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Specification Version Number"*, *"message": "Unsupported"*.
  - Caso in cui la lunghezza dell'SVN è diversa da 3 byte → *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "SM-DP+"*, *"message": "Execution Error"*.
- **Gemello di Tim**:
  - Caso in cui la lunghezza dell'SVN è uguale a 3 byte → *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.3"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Specification Version Number"*, *"message": "Unsupported"*.
  - Caso in cui la lunghezza dell'SVN è diversa da 3 byte → *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "SM-DP+"*, *"message": "Execution Error"*.
- **eSIM Discovery Production**:
  - Caso in cui la lunghezza dell'SVN è uguale a 3 byte → *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.9.3"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Specification Version Number"*, *"message": "The Specification Version Number indicated by the eUICC is not supported by the SM-DS"*.
  - Caso in cui la lunghezza dell'SVN è diversa da 3 byte → HTTP_INTERNAL_SERVER_ERROR (status code 500) al primo messaggio di risposta (initiateAuthenticationResponse).

- **eSIM Discovery Staging**:
  - Caso in cui la lunghezza dell'SVN è uguale a 3 byte → *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.9.3"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Specification Version Number"*, *"message": "The Specification Version Number indicated by the eUICC is not supported by the SM-DS"*.
  - Caso in cui la lunghezza dell'SVN è diversa da 3 byte → HTTP_INTERNAL_SERVER_ERROR (status code 500) al primo messaggio di risposta (initiateAuthenticationResponse).
- **GSMA**:
  - Caso in cui la lunghezza dell'SVN è uguale a 3 byte → *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.9.3"*, *"reasonCode": "3.1"*, *"message": "Execution Failed"* → SVN non valido.
  - Caso in cui la lunghezza dell'SVN è diversa da 3 byte → HTTP_INTERNAL_SERVER_ERROR (status code 500) al primo messaggio di risposta (initiateAuthenticationResponse).
- **Verizon Wireless**:
  - Caso in cui la lunghezza dell'SVN è uguale a 3 byte → *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.9.3"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Specification Version Number"*, *"message": "The Specification Version Number indicated by the eUICC is not supported by the SM-DS"*.
  - Caso in cui la lunghezza dell'SVN è diversa da 3 byte → HTTP_INTERNAL_SERVER_ERROR (status code 500) al primo messaggio di risposta (initiateAuthenticationResponse).
- **Stork & Google**:
  - Caso in cui la lunghezza dell'SVN è uguale a 3 byte → *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.2"*, *"reasonCode": "3.1"*, *"message": "Execution Failed"* → SVN non valido.
  - Caso in cui la lunghezza dell'SVN è diversa da 3 byte → HTTP_INTERNAL_SERVER_ERROR (status code 500) al primo messaggio di risposta (initiateAuthenticationResponse).
- **Infineon**: HTTP_BAD_REQUEST (status code 400) al primo messaggio di risposta (initiateAuthenticationResponse).
- **Sysmocom**: *"status": "Executed-Success"*.
- **Truphone**:
  - Caso in cui la lunghezza dell'SVN è uguale a 3 byte → *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.3"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Specification Version Number (SVN)"*, *"message": "Unsupported"*.
  - Caso in cui la lunghezza dell'SVN è diversa da 3 byte → *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "1.6"*, *"reasonCode": "2.1"*, *"subjectIdentifier": "Function"*, *"message": "Invalid"* → il formato di svn è errato.
- **Osmo-smdpp**: *"status": "Executed-Success"*.
- **Simulatore**: *"status": "Failed"* al secondo messaggio di risposta (authenticateClientResponse), con *"subjectCode: "8.1"*, *"reasonCode": "3.11"*, *"subjectIdentifier": "eUICC"*, *"message": "Value has Changed"*.

Caso di test 12.2

Consiste nel manomettere il messaggio initiateAuthentication ponendo i campi euiccCiPKIdListForVerification ed euiccCiPKIdListForSigning pari a un valore inatteso o comunque differente dall'identificatore della RootCA della certificate chain utilizzata dal simulatore del client.

- **Iliad & Very Mobile**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.3"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "SM-DP+ Certificate"*, *"message": "The SM-DP+ has no CERT.DPauth.SIG which chains to one of the eSIM CA RootCA Certificate with a Public Key supported by the eUICC"*.
- **Tim**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.2"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Security configuration"*, *"message": "Unsupported"*.
- **Gemello di Tim**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.2"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Security configuration"*, *"message": "Unsupported"*.
- **eSIM Discovery Production**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.9.2"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Security configuration"*, *"message": "None of the proposed Public Key Identifiers is supported by the SM-DS"*.
- **eSIM Discovery Staging**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.9.2"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Security configuration"*, *"message": "None of the proposed Public Key Identifiers is supported by the SM-DS"*.
- **GSMA**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.9.2"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Security configuration"*, *"message": "None of the proposed Public Key Identifiers is supported by the SM-DS"*.
- **Verizon Wireless**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.9.2"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Security configuration"*, *"message": "None of the proposed Public Key Identifiers is supported by the SM-DS"*.
- **Stork & Google**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.2"*, *"reasonCode": "3.1"*, *"message": "Execution Failed"* → nessun identificatore di chiave pubblica del CI (Public Key Identifier) tra quelli proposti è supportato dal server SM-DP+.
- **Infineon**: HTTP_BAD_REQUEST (status code 400) al primo messaggio di risposta (initiateAuthenticationResponse).
- **Sysmocom**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.2"*, *"reasonCode": "3.1"*, *"message": "None of the proposed Public Key Identifiers is supported by the SM-DP+"*.
- **Truphone**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.2"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Security configuration"*, *"message": "Unsupported"*.
- **Osmo-smdpp**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.2"*, *"reasonCode": "3.1"*, *"message": "None of the proposed Public Key Identifiers is supported by the SM-DP+"*.
- **Simulatore**: *"status": "Failed"* al primo messaggio di risposta (initiateAuthenticationResponse), con *"subjectCode: "8.8.2"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Security configuration"*, *"message": "Unsupported"*.

Caso di test 13.1

Consiste nell'avviare la Common Cancel Session Procedure appena si è stabilita la connessione col server SM-DP+, senza passare nemmeno per il messaggio di initiateAuthentication.

- **Iliad & Very Mobile**: Common Cancel Session Procedure completata con successo.
- **Tim**: Common Cancel Session Procedure completata con successo.
- **Gemello di Tim**: Common Cancel Session Procedure completata con successo.
- **eSIM Discovery Production**: HTTP_NOT_FOUND (status code 404) al primo messaggio di risposta (risposta a cancelSession).
- **eSIM Discovery Staging**: HTTP_NOT_FOUND (status code 404) al primo messaggio di risposta (risposta a cancelSession).
- **GSMA**: HTTP_NOT_FOUND (status code 404) al primo messaggio di risposta (risposta a cancelSession).
- **Verizon Wireless**: HTTP_NOT_FOUND (status code 404) al primo messaggio di risposta (risposta a cancelSession).
- **Stork & Google**: Common Cancel Session Procedure completata con successo.
- **Infineon**: HTTP_BAD_REQUEST (status code 400) al primo messaggio di risposta (risposta a cancelSession).
- **Sysmocom**: Common Cancel Session Procedure completata con successo.
- **Truphone**: Common Cancel Session Procedure completata con successo.
- **Osmo-smdpp**: Common Cancel Session Procedure completata con successo.
- **Simulatore**: Common Cancel Session Procedure completata con successo.

Caso di test 13.2

Consiste nell'avviare la Common Cancel Session Procedure subito dopo aver ricevuto il messaggio di initiateAuthenticationResponse.

- **Iliad & Very Mobile**: Common Cancel Session Procedure completata con successo.
- **Tim**: Common Cancel Session Procedure completata con successo.
- **Gemello di Tim**: Common Cancel Session Procedure completata con successo.
- **eSIM Discovery Production**: HTTP_NOT_FOUND (status code 404) al secondo messaggio di risposta (risposta a cancelSession).
- **eSIM Discovery Staging**: HTTP_NOT_FOUND (status code 404) al secondo messaggio di risposta (risposta a cancelSession).
- **GSMA**: HTTP_NOT_FOUND (status code 404) al secondo messaggio di risposta (risposta a cancelSession).
- **Verizon Wireless**: HTTP_NOT_FOUND (status code 404) al secondo messaggio di risposta (risposta a cancelSession).
- **Stork & Google**: Common Cancel Session Procedure completata con successo.
- **Infineon**: n.d.
- **Sysmocom**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (risposta a cancelSession).
- **Truphone**: Common Cancel Session Procedure completata con successo.
- **Osmo-smdpp**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (risposta a cancelSession).
- **Simulatore**: Common Cancel Session Procedure completata con successo.

Caso di test 14.1

Consiste nell'inviare getBoundProfilePackage come primo messaggio della comunicazione col server SM-DP+.

- **Iliad & Very Mobile**: *"status": "Failed"* al primo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "<transactionId value>"*, *"message": "Cannot retrieve the bound profile package with the current transactionId [<transactionId value>]"*.
- **Tim**: *"status": "Failed"* al primo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "Transaction ID"*, *"message": "Unknown"*.
- **Gemello di Tim**: *"status": "Failed"* al primo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "Transaction ID"*, *"message": "Unknown"*.
- **eSIM Discovery Production**: HTTP_NOT_FOUND (status code 404) al primo messaggio di risposta (getBoundProfilePackageResponse).
- **eSIM Discovery Staging**: HTTP_NOT_FOUND (status code 404) al primo messaggio di risposta (getBoundProfilePackageResponse).
- **GSMA**: HTTP_NOT_FOUND (status code 404) al primo messaggio di risposta (getBoundProfilePackageResponse).
- **Verizon Wireless**: HTTP_NOT_FOUND (status code 404) al primo messaggio di risposta (getBoundProfilePackageResponse).
- **Stork & Google**: *"status": "Failed"* al primo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"message": "Execution Failed"* → la sessione RSP identificata dal Transaction ID specificato è sconosciuta.
- **Infineon**: HTTP_BAD_REQUEST (status code 400) al primo messaggio di risposta (getBoundProfilePackageResponse).
- **Sysmocom**: *"status": "Failed"* al primo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"message": "The RSP session identified by the TransactionID is unknown"*.
- **Truphone**: *"status": "Failed"* al primo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "TransactionId"*, *"message": "Unknown"*.
- **Osmo-smdpp**: *"status": "Failed"* al primo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"message": "The RSP session identified by the TransactionID is unknown"*.
- **Simulatore**: *"status": "Failed"* al primo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "TransactionId"*, *"message": "Unknown"*.

Caso di test 14.2

Consiste nell'inviare getBoundProfilePackage subito dopo aver ricevuto il messaggio di initiateAuthenticationResponse, saltando così il messaggio di authenticateClient.

- **Iliad & Very Mobile**: *"status": "Failed"* al secondo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "<transactionId value>"*, *"message": "Cannot retrieve the bound profile package with the current transactionId [<transactionId value>]"*.

- **Tim**: *"status": "Failed"* al secondo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.8"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "SM-DP+"*, *"message": "Execution Error"*.
- **Gemello di Tim**: *"status": "Failed"* al secondo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.8"*, *"reasonCode": "4.2"*, *"subjectIdentifier": "SM-DP+"*, *"message": "Execution Error"*.
- **eSIM Discovery Production**: HTTP_NOT_FOUND (status code 404) al secondo messaggio di risposta (getBoundProfilePackageResponse).
- **eSIM Discovery Staging**: HTTP_NOT_FOUND (status code 404) al secondo messaggio di risposta (getBoundProfilePackageResponse).
- **GSMA**: HTTP_NOT_FOUND (status code 404) al secondo messaggio di risposta (getBoundProfilePackageResponse).
- **Verizon Wireless**: HTTP_NOT_FOUND (status code 404) al secondo messaggio di risposta (getBoundProfilePackageResponse).
- **Stork & Google**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (getBoundProfilePackageResponse).
- **Infineon**: n.d.
- **Sysmocom**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (getBoundProfilePackageResponse).
- **Truphone**: *"status": "Failed"* al secondo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.8.4"*, *"reasonCode": "3.7"*, *"subjectIdentifier": "SM-DP+ Certificate"*, *"message": "Unavailable"*.
- **Osmo-smdpp**: HTTP_INTERNAL_SERVER_ERROR (status code 500) al secondo messaggio di risposta (getBoundProfilePackageResponse).
- **Simulatore**: *"status": "Failed"* al primo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.8.2"*, *"reasonCode": "3.1"*, *"subjectIdentifier": "Security configuration"*, *"message": "Unsupported"*.

Caso di test 15

Consiste nell'inviare getBoundProfilePackage come primo messaggio della comunicazione col server SM-DP+, dove il messaggio ha il campo transactionId pari all'ID di transazione di una connessione instaurata da un altro processo client messo appositamente in sleep. È interessante applicare questo caso di test in particolar modo ai server SM-DP+ che hanno fornito risposte differenti nei casi di test 14.1 e 14.2.

- **Iliad & Very Mobile**: n.d.
- **Tim**: *"status": "Failed"* al primo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "Transaction ID"*, *"message": "Unknown"*.
- **Gemello di Tim**: *"status": "Failed"* al primo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "Transaction ID"*, *"message": "Unknown"*.
- **eSIM Discovery Production**: n.d.
- **eSIM Discovery Staging**: n.d.
- **GSMA**: n.d.
- **Verizon Wireless**: n.d.

- **Stork & Google**: *"status": "Failed"* al primo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"message": "Execution Failed"* → la sessione RSP identificata dal Transaction ID specificato è sconosciuta.
- **Infineon**: n.d.
- **Sysmocom**: *"status": "Failed"* al primo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"message": "The RSP session identified by the TransactionID is unknown"*.
- **Truphone**: *"status": "Failed"* al primo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"subjectIdentifier": "TransactionId"*, *"message": "Unknown"*.
- **Osmo-smdpp**: *"status": "Failed"* al primo messaggio di risposta (getBoundProfilePackageResponse), con *"subjectCode: "8.10.1"*, *"reasonCode": "3.9"*, *"message": "The RSP session identified by the TransactionID is unknown"*.
- **Simulatore**: il server non è in grado di processare più richieste in parallelo.

5.3.2 Certificati esibiti dai server

Un ulteriore aspetto interessante da analizzare nella comunicazione tra il simulatore del client e i server SM-DP+ in esame è il certificato CERT.DPauth.SIG esibito da ciascun server all'interno del messaggio initiateAuthenticationResponse. CERT.DPauth.SIG può essere ispezionato solo per i server che hanno risposto con successo al messaggio di richiesta initiateAuthentication in almeno uno dei casi di test sopra definiti. È perciò escluso da questa trattazione il server Infineon. Invece, i principali campi del certificato CERT.DPauth.SIG degli altri dodici server SM-DP+ sono elencati qui di seguito.

Certificato di Iliad e Very Mobile

- **Algorithm**: ecdsaWithSHA256;
- **issuer organizationName**: GSM Association;
- **issuer commonName**: GSM Association - RSP2 Root CI1;
- **validity notBefore**: 2021-10-28;
- **validity notAfter**: 2024-10-27;
- **countryName**: DE;
- **localityName**: Frankfurt;
- **organizationName**: ThalesSA;
- **organizationalUnitName**: IT;
- **commonName**: SMDP+ GCP EW3 MULTITENANT;
- **crLDistributionPoints**: http://gsma-crl.symauth.com/offlineca/gsma-rsp2-root-ci1.crl;
- **authorityKeyIdentifier**: rsp2.

Certificato di Tim

- **Algorithm**: ecdsaWithSHA256;
- **issuer organizationName**: GSM Association;
- **issuer commonName**: GSM Association - RSP2 Root CI1;
- **validity notBefore**: 2021-12-08;

- **validity notAfter**: 2031-12-07;
- **countryName**: FR;
- **localityName**: Paris;
- **organizationName**: IDEMIA;
- **commonName**: Azure SM-DP 401;
- **crLDistributionPoints**: http://gsma-crl.symauth.com/offlineca/gsma-rsp2-root-ci1.crl;
- **authorityKeyIdentifier**: rsp2.

Certificato del gemello di Tim

- **Algorithm**: ecdsaWithSHA256;
- **issuer organizationName**: GSM Association;
- **issuer commonName**: GSM Association - RSP2 Root CI1;
- **validity notBefore**: 2017-04-13;
- **validity notAfter**: 2027-04-12;
- **countryName**: RO;
- **localityName**: Otopeni;
- **organizationName**: Oberthur Technologies Romania SRL;
- **commonName**: Oberthur Technologies SM-DP 01;
- **crLDistributionPoints**: http://gsma-crl.symauth.com/offlineca/gsma-rsp2-root-ci1.crl;
- **authorityKeyIdentifier**: rsp2.

Certificato di eSIM Discovery Production

- **Algorithm**: ecdsaWithSHA256;
- **issuer organizationName**: GSM Association;
- **issuer commonName**: GSM Association - RSP2 Root CI1;
- **validity notBefore**: 2023-02-28;
- **validity notAfter**: 2026-02-27;
- **countryName**: US;
- **localityName**: Virginia;
- **organizationName**: Thales SA;
- **commonName**: Google SMDS UE4;
- **crLDistributionPoints**: http://gsma-crl.symauth.com/offlineca/gsma-rsp2-root-ci1.crl;
- **authorityKeyIdentifier**: rsp2.

Certificato di eSIM Discovery Staging

- **Algorithm**: ecdsaWithSHA256;
- **issuer organizationName**: GSM Association;
- **issuer commonName**: GSM Association - RSP2 Root CI1;
- **validity notBefore**: 2023-02-08;
- **validity notAfter**: 2026-02-07;
- **countryName**: US;
- **localityName**: Virginia;
- **organizationName**: Thales SA;
- **commonName**: Google SMDS UE4 LAB;
- **crLDistributionPoints**: http://gsma-crl.symauth.com/offlineca/gsma-rsp2-root-ci1.crl;
- **authorityKeyIdentifier**: rsp2.

Certificato di GSMA

- **Algorithm**: ecdsaWithSHA256;
- **issuer organizationName**: GSM Association;
- **issuer commonName**: GSM Association - RSP2 Root CI1;
- **validity notBefore**: 2023-06-21;
- **validity notAfter**: 2026-06-20;
- **countryName**: DE;
- **localityName**: Frankfurt;
- **organizationName**: Thales SA;
- **organizationalUnitName**: IT;
- **commonName**: GCP EW3 PROD SMDS;
- **crLDistributionPoints**: http://gsma-crl.symauth.com/offlineca/gsma-rsp2-root-ci1.crl;
- **authorityKeyIdentifier**: rsp2.

Certificato di Verizon Wireless

- **Algorithm**: ecdsaWithSHA256;
- **issuer organizationName**: GSM Association;
- **issuer commonName**: GSM Association - RSP2 Root CI1;
- **validity notBefore**: 2022-07-06;
- **validity notAfter**: 2025-07-05;
- **countryName**: US;
- **localityName**: Virginia;
- **organizationName**: Thales SA;
- **commonName**: VZW GSMA UE4 SMDS;
- **crLDistributionPoints**: http://gsma-crl.symauth.com/offlineca/gsma-rsp2-root-ci1.crl;
- **authorityKeyIdentifier**: rsp2.

Certificato di Stork e Google

- **Algorithm**: ecdsaWithSHA256;
- **issuer organizationName**: Symantec Corporation;
- **issuer commonName**: Symantec Corporation RSP Test Root CA - For Test Purposes Only;
- **validity notBefore**: 2019-05-14;
- **validity notAfter**: 2022-05-13;
- **organizationName**: Google LLC;
- **commonName**: Google LLC DPauth Prod;
- **crLDistributionPoints**: http://pki-crl.symauth.com/ca_a3dc2e3fea7708a11c889386d9d3a76f/LatestCRL.crl;
- **authorityKeyIdentifier**: symantec.

Certificato di Sysmocom (NIST)

- **Algorithm**: ecdsaWithSHA256;
- **issuer organizationName**: RSPTEST;
- **issuer organizationalUnitName**: TESTCERT;
- **issuer commonName**: Test CI ;
- **issuer countryName**: IT;
- **validity notBefore**: 2020-04-01;
- **validity notAfter**: 2030-03-30;
- **organizationName**: ACME;
- **commonName**: TEST SM-DP+;
- **crLDistributionPoints**: http://ci.test.example.com/CRL-A.crl; http://ci.test.example.com/crl-B.crl;
- **authorityKeyIdentifier**: sgp.

Certificato di Truphone

- **Algorithm**: ecdsaWithSHA256;
- **issuer organizationName**: GSM Association;
- **issuer commonName**: GSM Association - RSP2 Root CI1;
- **validity notBefore**: 2024-02-27;
- **validity notAfter**: 2027-02-26;
- **countryName**: GB;
- **localityName**: London;
- **organizationName**: TP Global Operations Limited;
- **organizationalUnitName**: RSP;
- **commonName**: 1GLOBAL SM-DP+;
- **crLDistributionPoints**: http://gsma-crl.symauth.com/offlineca/gsma-rsp2-root-ci1.crl;
- **authorityKeyIdentifier**: rsp2.

Certificato di Osmo-smdpp (BRP & NIST)

- **Algorithm**: ecdsaWithSHA256;
- **issuer organizationName**: RSPTEST;
- **issuer organizationalUnitName**: TESTCERT;
- **issuer commonName**: Test CI ;
- **issuer countryName**: IT;
- **validity notBefore**: 2020-04-01;
- **validity notAfter**: 2030-03-30;
- **organizationName**: ACME;
- **commonName**: TEST SM-DP+;
- **crLDistributionPoints**: http://ci.test.example.com/CRL-A.crl; http://ci.test.example.com/crl-B.crl;
- **authorityKeyIdentifier**: sgp.

Certificato del simulatore del server

- **Algorithm**: ecdsaWithSHA256;
- **issuer organizationName**: Fanfa Association;
- **issuer commonName**: Fanfa Association - RSP2 Root CI1;
- **validity notBefore**: 2024-04-12;
- **validity notAfter**: 2034-04-10;
- **countryName**: FR;
- **localityName**: Tours;
- **organizationName**: Thales DIS France SA;
- **organizationalUnitName**: ODC;
- **commonName**: SMDP Plus Torus Platform;
- **crLDistributionPoints**: http://gsma-crl.symauth.com/offlineca/gsma-rsp2-root-ci1.crl;
- **authorityKeyIdentifier**: fanfa.

A proposito di certificati, in rete sono disponibili quelli delle RootCA incontrate finora, ovvero GSM Association (i.e. GSMA), RSPTEST (a capo della certificate chain di test usata da Infineon e Osmo-smdpp) e Symantec Corporation (a capo della certificate chain di test usata da Stork e Google). Sempre con l'ausilio del sito *ASN.1 JavaScript decoder*, è stato possibile ispezionare anche questi altri tre certificati.

Certificato Root di GSM Association

- **Algorithm**: ecdsaWithSHA256;
- **issuer organizationName**: GSM Association;
- **issuer commonName**: GSM Association - RSP2 Root CI1;
- **validity notBefore**: 2017-02-22;
- **validity notAfter**: 2052-02-21;
- **organizationName**: GSM Association;
- **commonName**: GSM Association - RSP2 Root CI1;
- **crLDistributionPoints**: http://gsma-crl.symauth.com/offlineca/gsma-rsp2-root-ci1.crl;
- **authorityKeyIdentifier**: rsp2.

Certificato Root di RSPTEST (BRP & NIST)

- **Algorithm**: ecdsaWithSHA256;
- **issuer organizationName**: RSPTEST;
- **issuer organizationalUnitName**: TESTCERT;
- **issuer commonName**: Test CI;
- **issuer countryName**: IT;
- **validity notBefore**: 2020-04-01;
- **validity notAfter**: 2055-04-01;
- **countryName**: IT;
- **organizationName**: RSPTEST;
- **organizationalUnitName**: TESTCERT;
- **commonName**: Test CI;
- **crLDistributionPoints**: http://ci.test.example.com/CRL-A.crl; http://ci.test.example.com/crl-B.crl;
- **authorityKeyIdentifier**: sgp.

Certificato Root di Symantec Corporation

- **Algorithm**: ecdsaWithSHA256;
- **issuer organizationName**: Symantec Corporation;
- **issuer commonName**: Symantec Corporation RSP Test Root CA - For Test Purposes Only;
- **validity notBefore**: 2017-07-11;
- **validity notAfter**: 2049-12-31;
- **organizationName**: Symantec Corporation;
- **commonName**: Symantec Corporation RSP Test Root CA - For Test Purposes Only;
- **crLDistributionPoints**: http://pki-crl.symauth.com/SymantecRSPTestRootCA/LatestCRL.crl
- **authorityKeyIdentifier**: symantec.

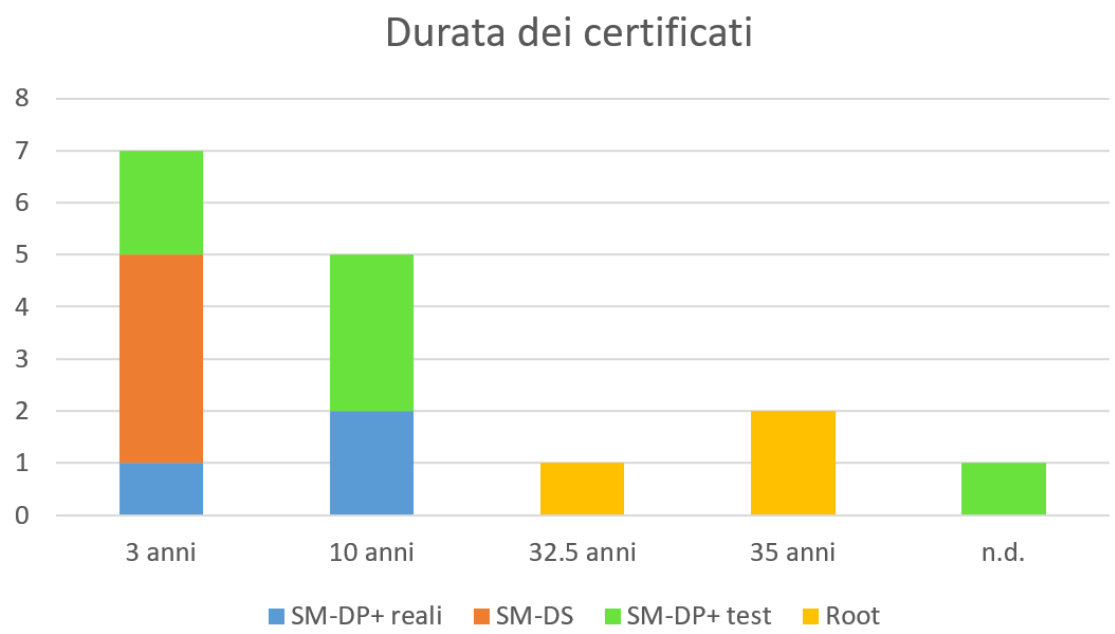


Figura 5.2: Durata dei certificati dei server SM-DP+ e delle Root CA.

5.3.3 Analisi delle risposte ricevute

A seguito di un'attenta analisi dei vari messaggi di risposta ricevuti dai server SM-DP+, il primo aspetto interessante che balza all'occhio è l'estrema somiglianza tra alcune implementazioni. In particolare, è possibile individuare tre sottoinsiemi di server:

1. Tim, Gemello di Tim, Truphone;
2. eSIM Discovery Production, eSIM Discovery Staging, GSMA, Verizon Wireless;
3. Sysmocom, Osmo-smdpp.

Solo il server di Iliad & Very Mobile, quello di Stork & Google e il simulatore precedentemente sviluppato assumono un comportamento abbastanza differente da tutti gli altri. D'altra parte, il server di Infineon solleva sempre l'errore HTTP_BAD_REQUEST indipendentemente dal messaggio di richiesta inviatogli: si tratta dunque di un server non particolarmente utile ai fini dello studio ma, per completezza, sarà preso in considerazione anche nelle valutazioni successive.
Ai fini dell'analisi finale, non si terrà conto della suddivisione dei server SM-DP+ basata sui relativi messaggi di risposta, bensì ha più senso fare una distinzione sulla tipologia dei server stessi: di fatto, la mancanza di un controllo nell'implementazione di un server SM-DP+ di test ha un peso differente dalla mancanza del medesimo controllo nell'implementazione di un server SM-DP+ reale. In definitiva, farà fede la classificazione riportata qui di seguito:

- **server SM-DP+ reali**: Iliad & Very Mobile, Tim, Gemello di Tim;
- **server SM-DS**: eSIM Discovery Production, eSIM Discovery Staging, GSMA, Verizon Wireless;
- **server SM-DP+ di test**: Stork & Google, Infineon, Sysmocom, Truphone, Osmo-smdpp, simulatore.

A partire da tale suddivisione e considerando anche le Root CA, è possibile anzitutto riassumere i tempi di validità di tutti i certificati ricevuti durante gli esperimenti: dalla figura 5.2 emerge che la durata dei certificati dei server varia tra i 3 e i 10 anni, mentre la durata dei certificati delle Root CA varia tra i 32 anni e mezzo e i 35 anni.
Sono state prodotte anche delle statistiche riassuntive sui casi di test più rilevanti, che sono raffigurate nelle pagine successive.

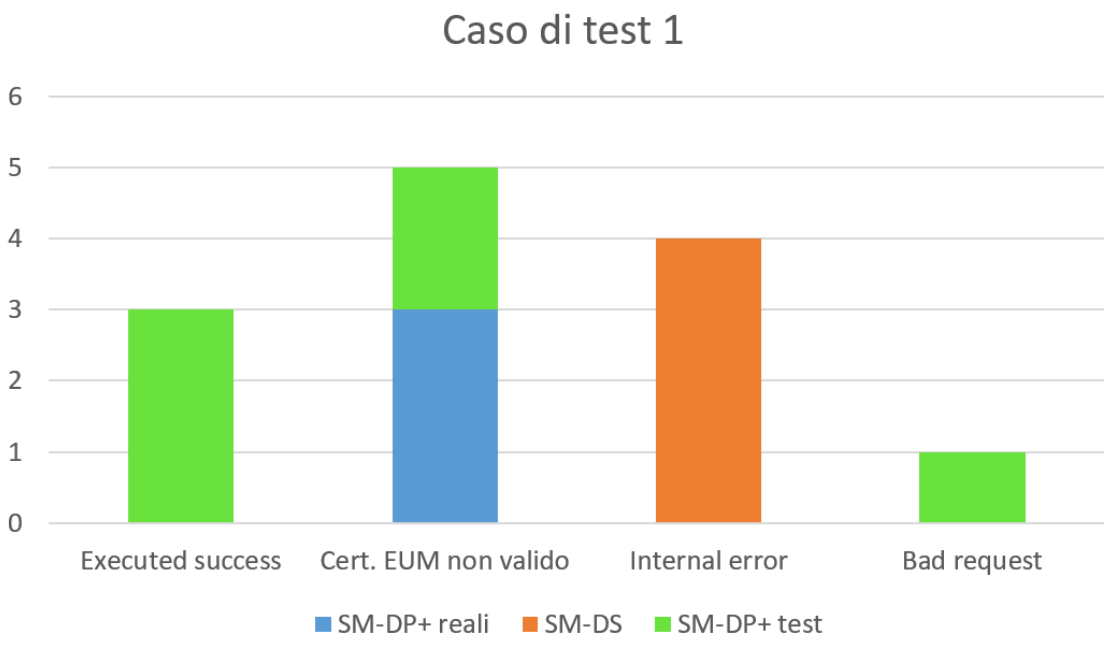


Figura 5.3: Risposte dei server SM-DP+ nel caso di test 1.

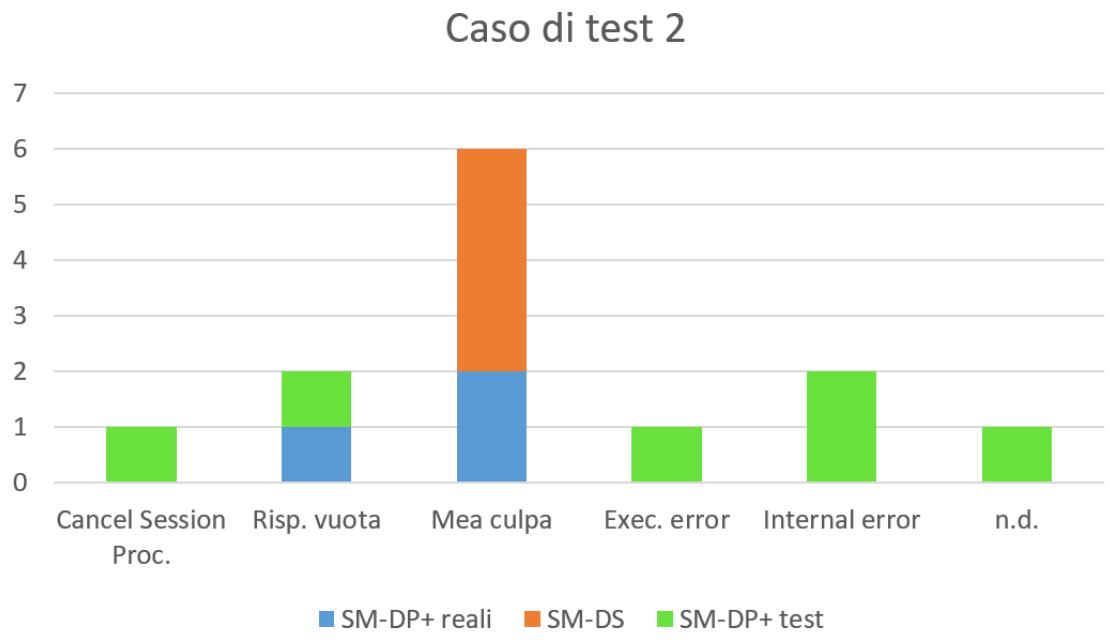


Figura 5.4: Risposte dei server SM-DP+ nel caso di test 2.

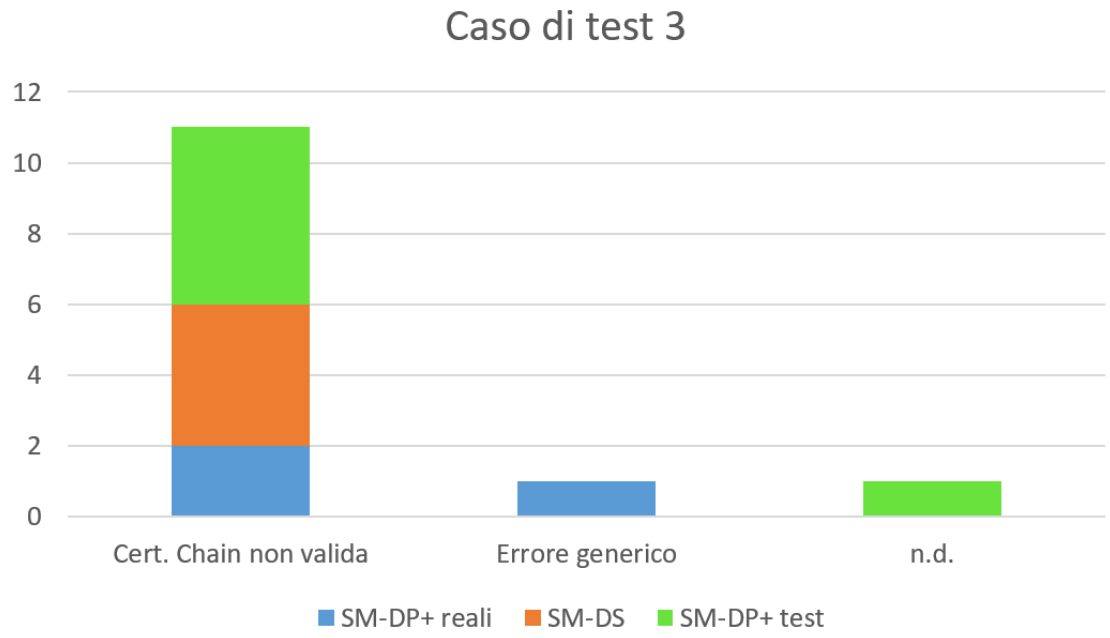


Figura 5.5: Risposte dei server SM-DP+ nel caso di test 3.

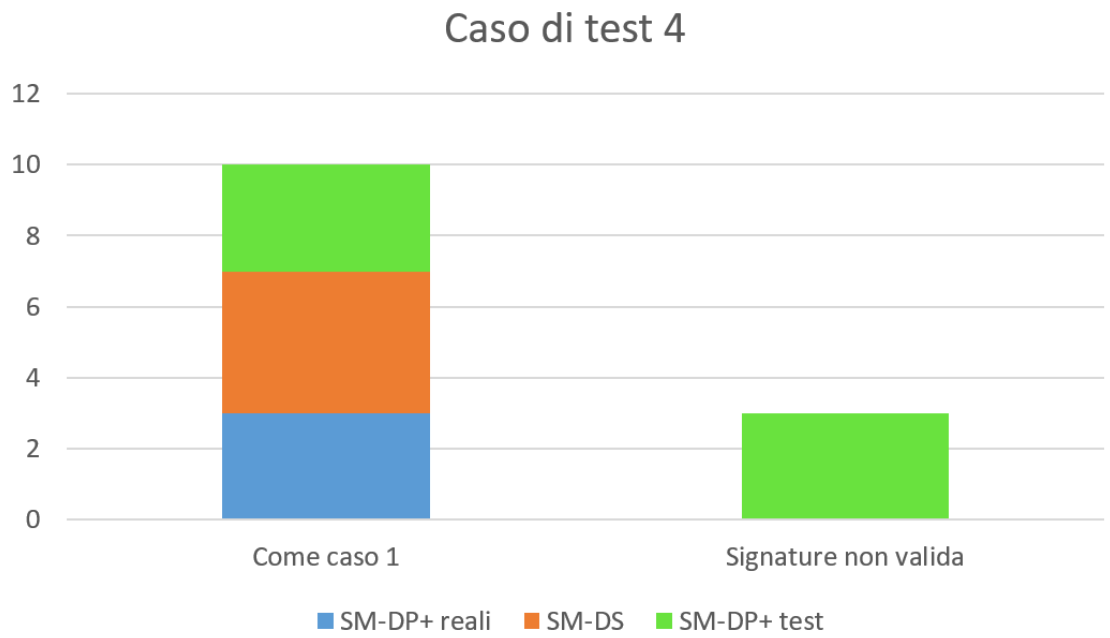


Figura 5.6: Risposte dei server SM-DP+ nel caso di test 4.

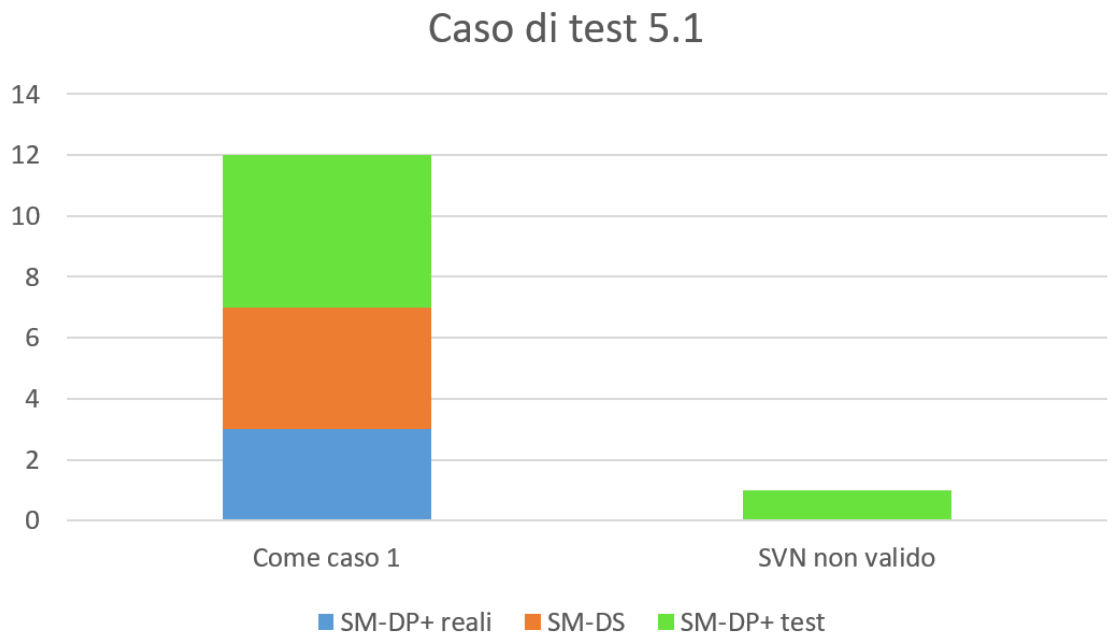


Figura 5.7: Risposte dei server SM-DP+ nel caso di test 5.1.

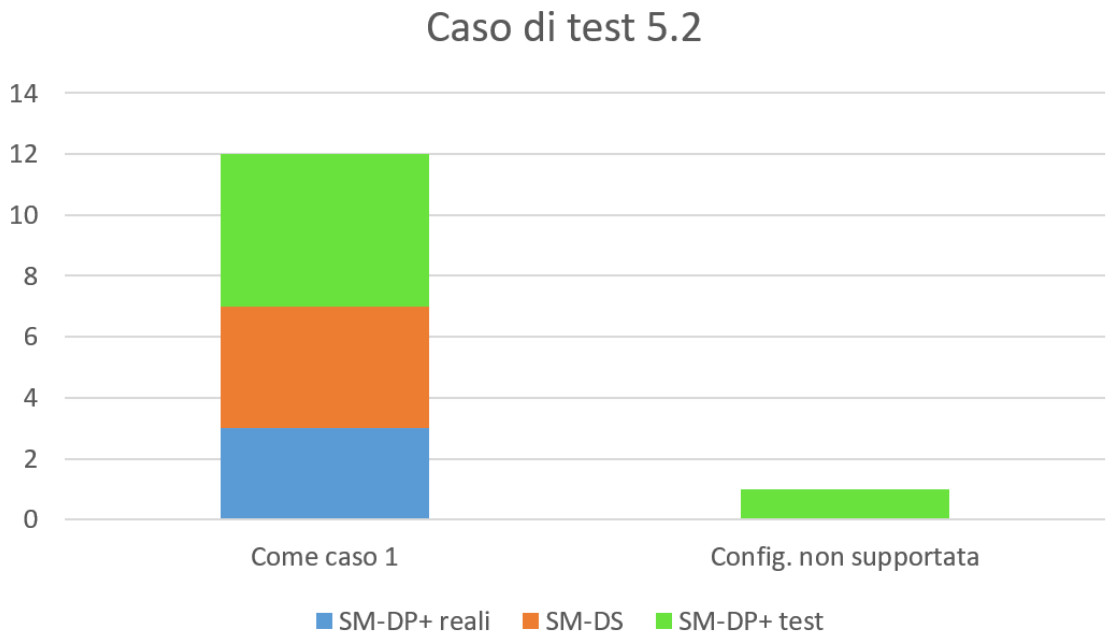


Figura 5.8: Risposte dei server SM-DP+ nel caso di test 5.2.

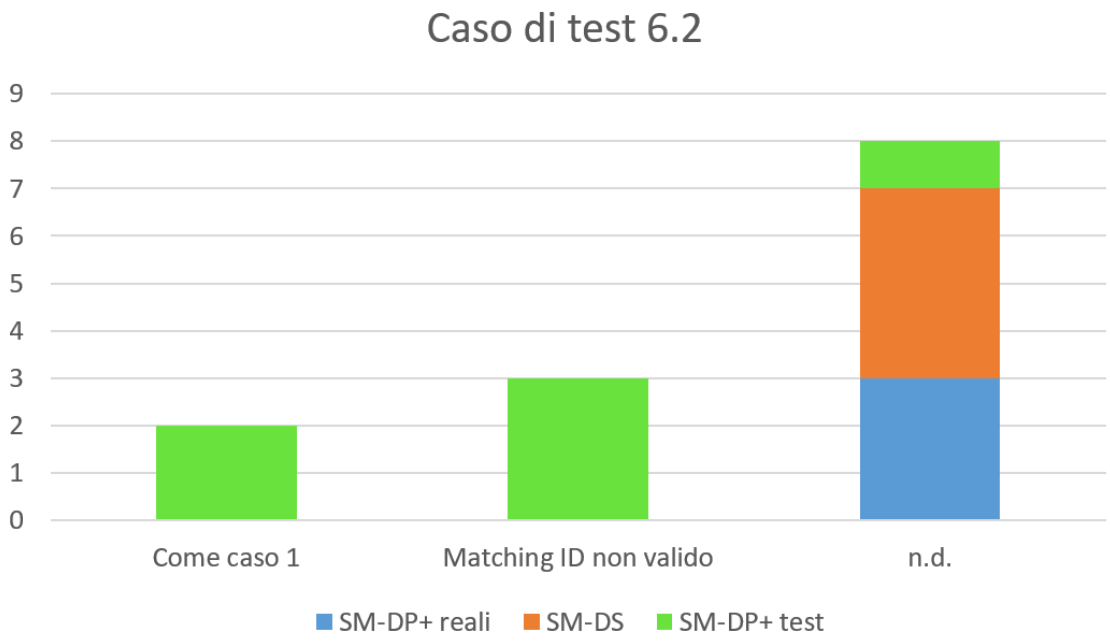


Figura 5.9: Risposte dei server SM-DP+ nel caso di test 6.2.

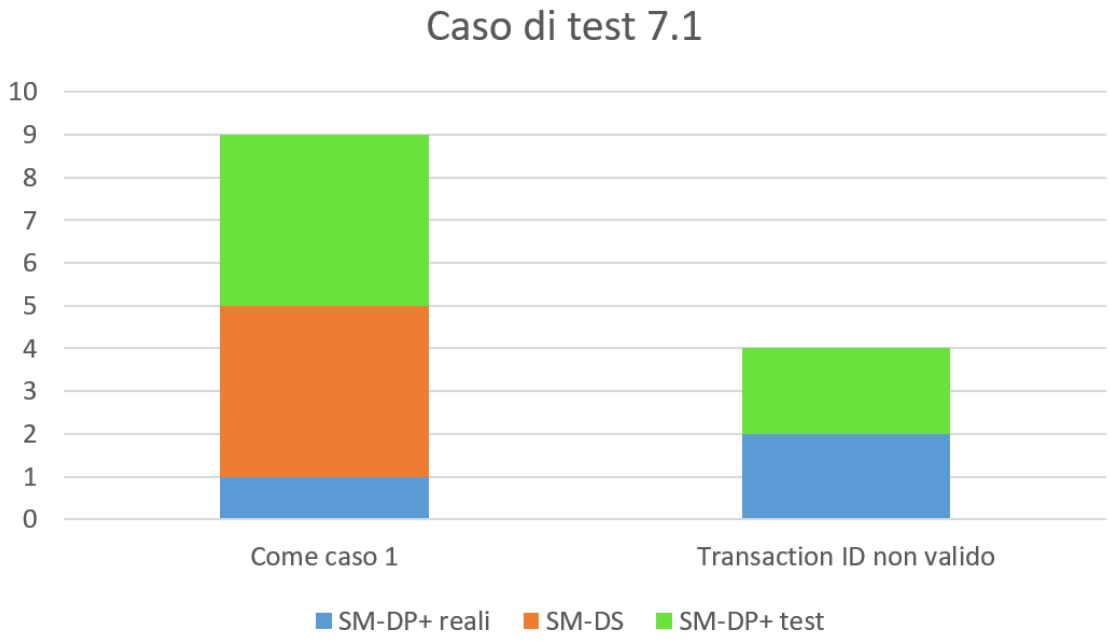


Figura 5.10: Risposte dei server SM-DP+ nel caso di test 7.1.

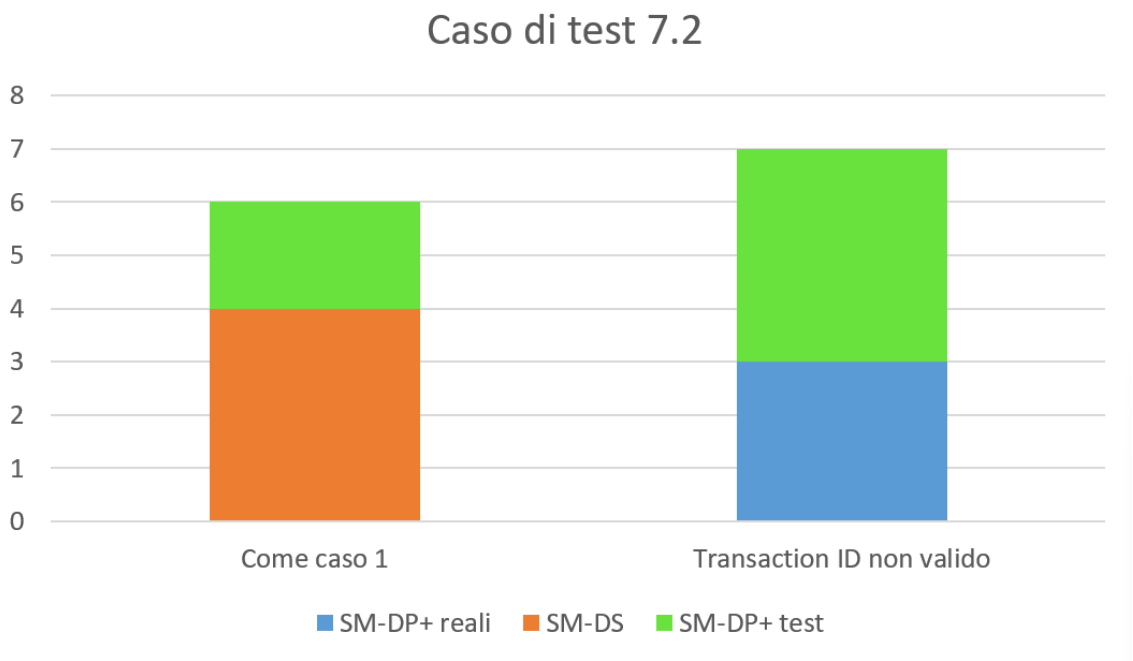


Figura 5.11: Risposte dei server SM-DP+ nel caso di test 7.2.

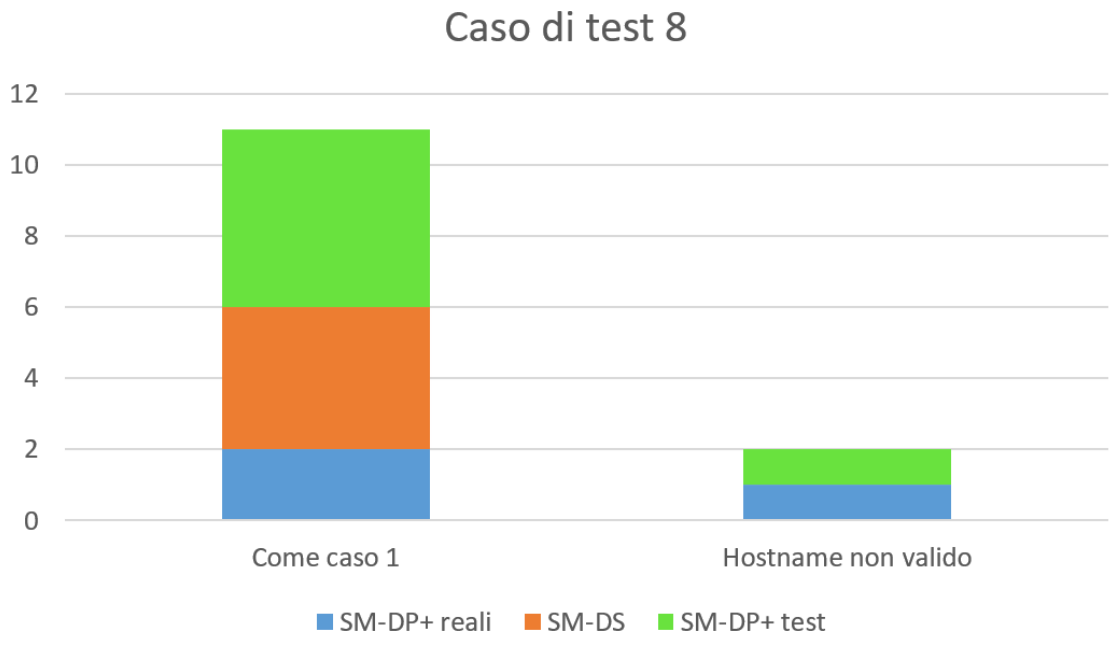


Figura 5.12: Risposte dei server SM-DP+ nel caso di test 8.

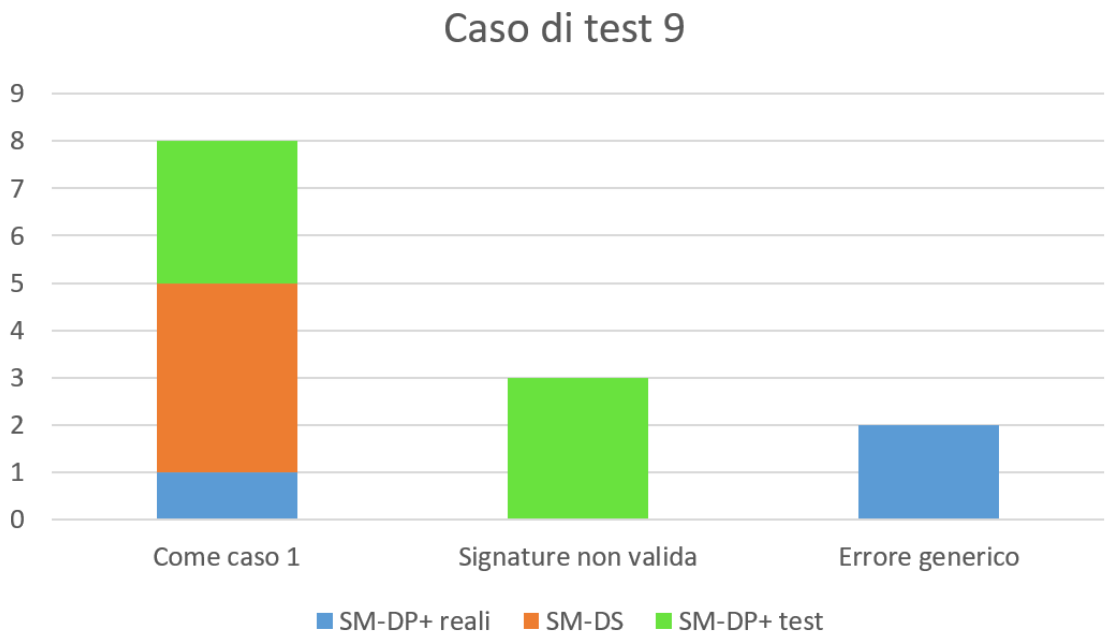


Figura 5.13: Risposte dei server SM-DP+ nel caso di test 9.

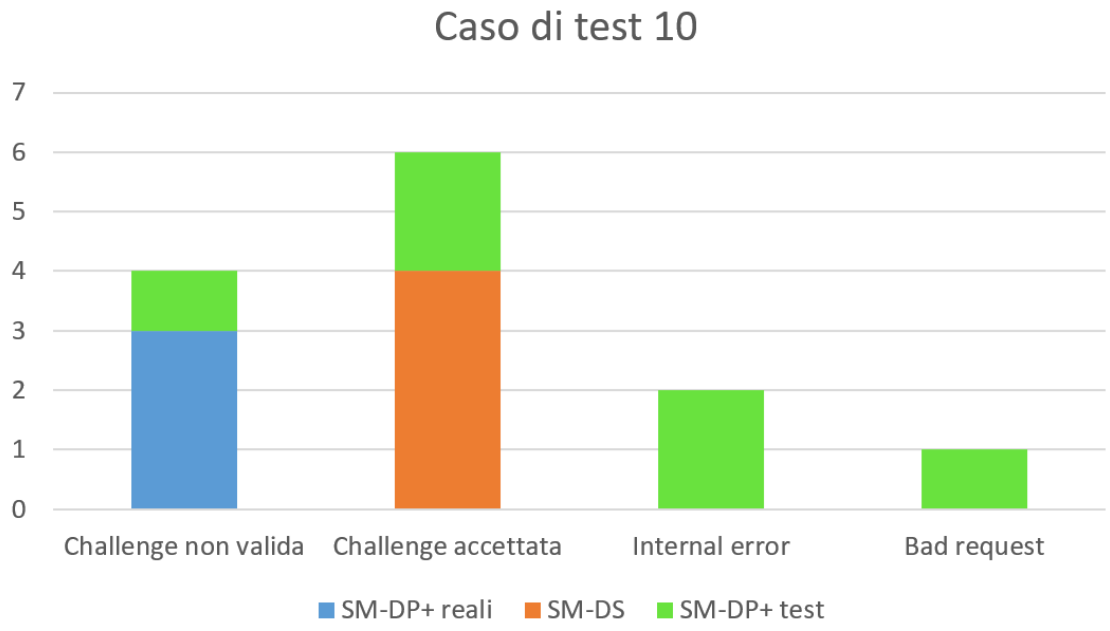


Figura 5.14: Risposte dei server SM-DP+ nel caso di test 10.

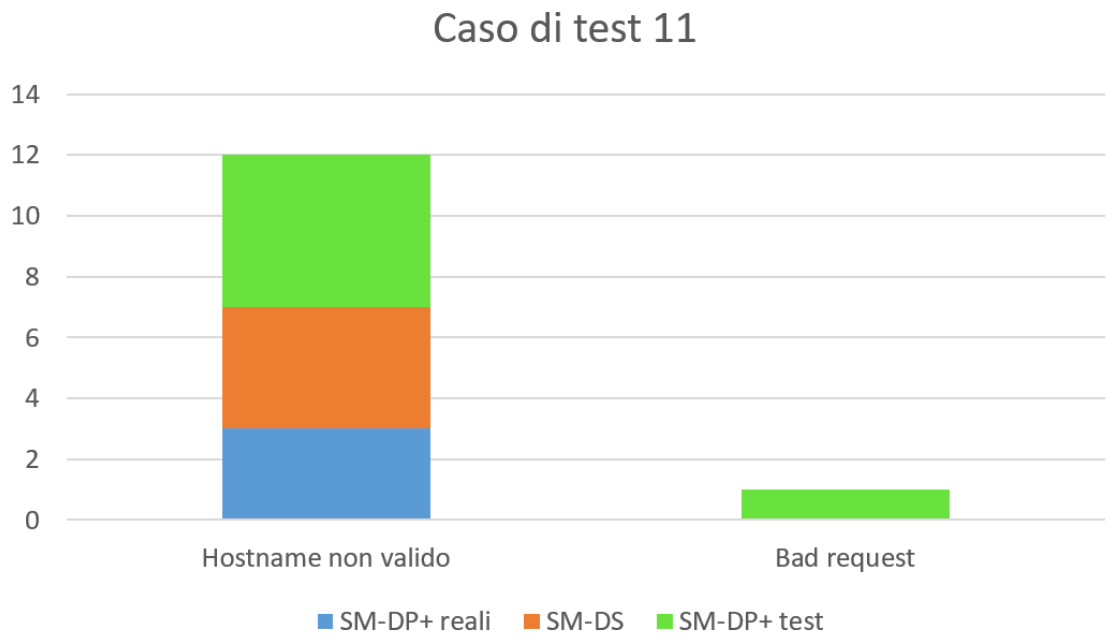


Figura 5.15: Risposte dei server SM-DP+ nel caso di test 11.

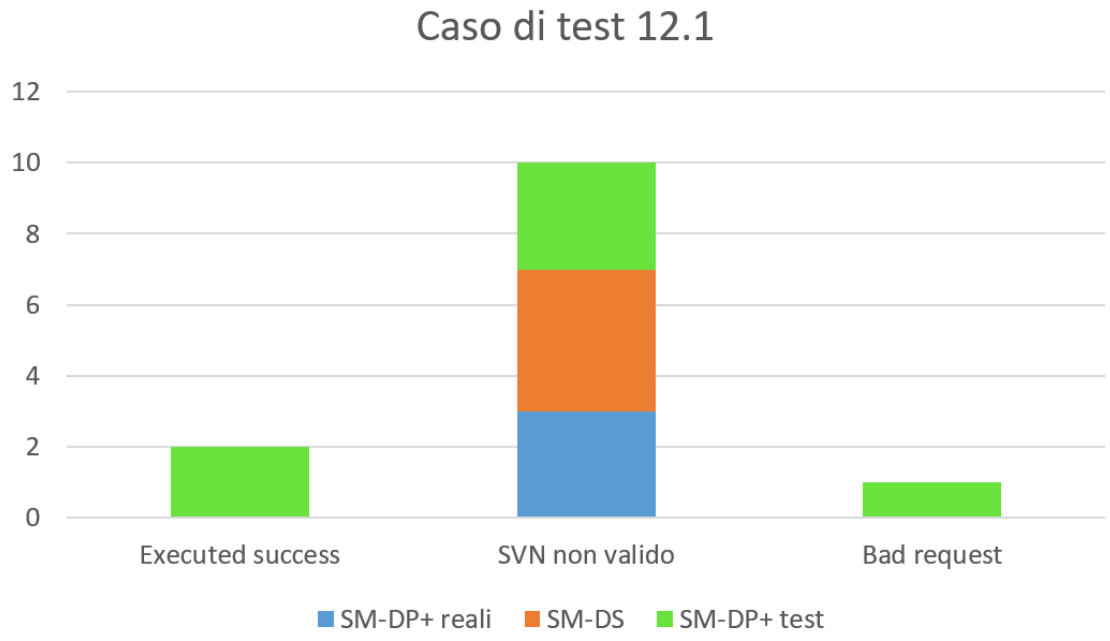


Figura 5.16: Risposte dei server SM-DP+ nel caso di test 12.1.

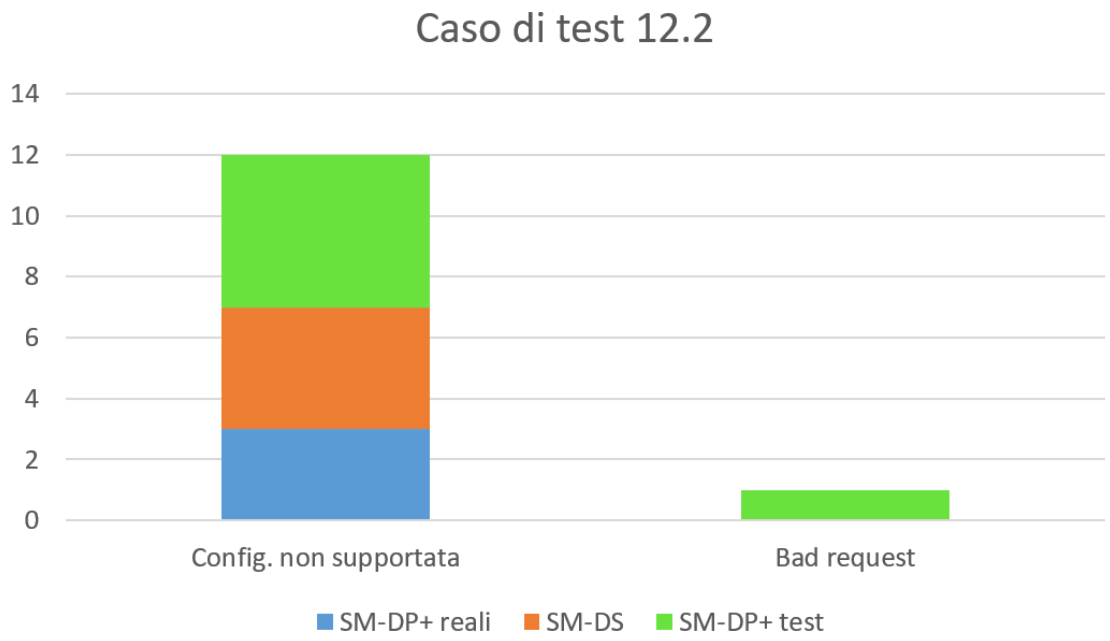


Figura 5.17: Risposte dei server SM-DP+ nel caso di test 12.2.

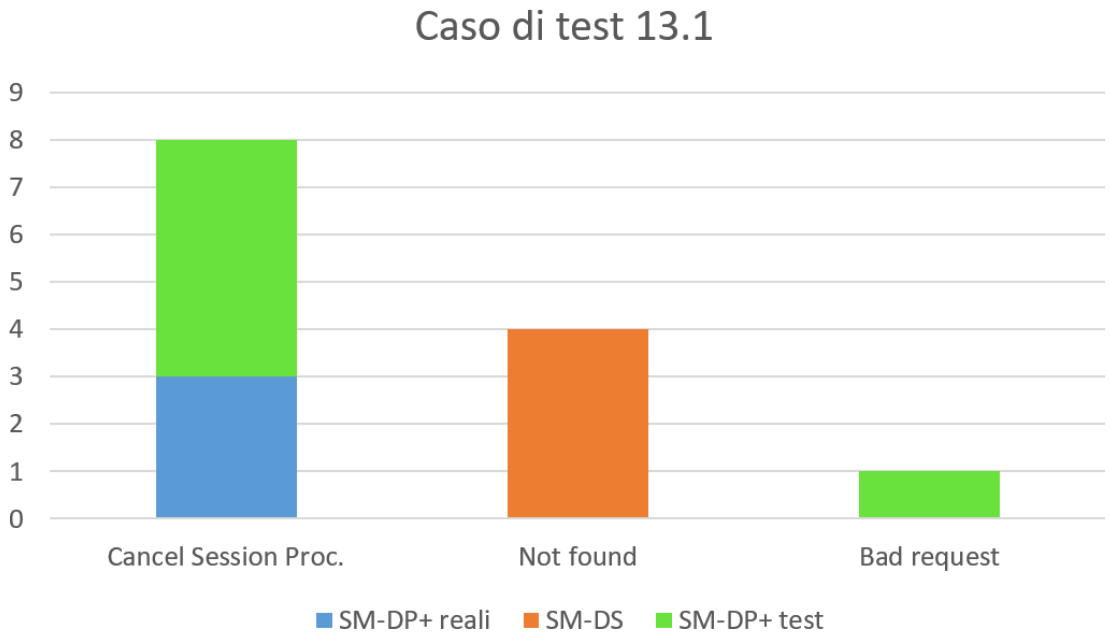


Figura 5.18: Risposte dei server SM-DP+ nel caso di test 13.1.

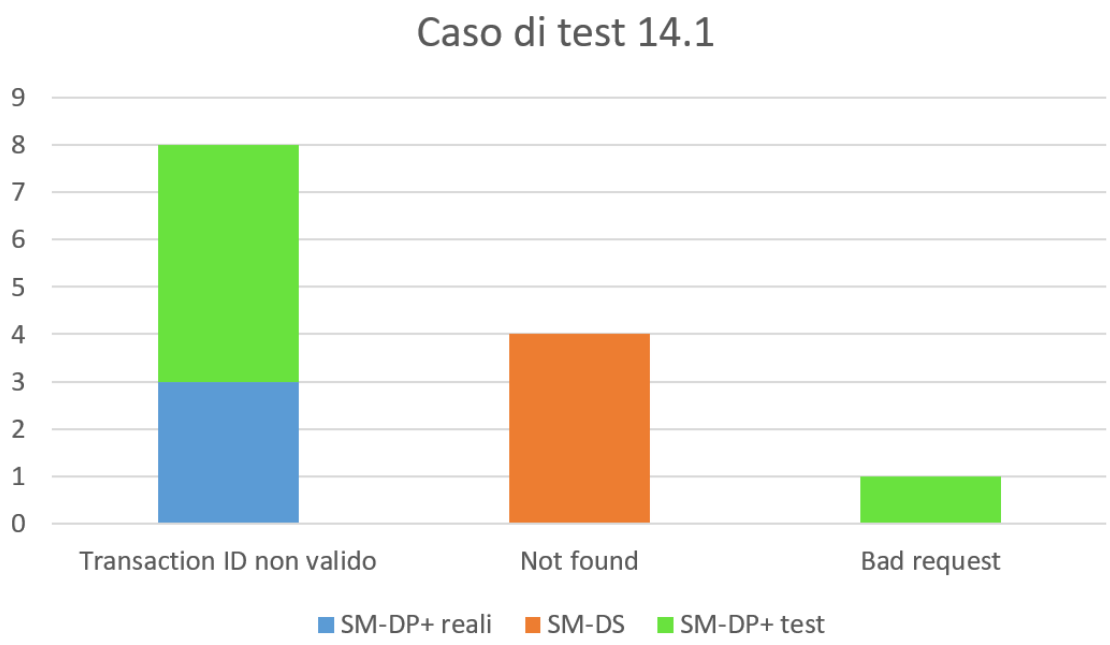


Figura 5.19: Risposte dei server SM-DP+ nel caso di test 14.1.

5.3.4 Osservazioni sulle risposte ricevute

Osservando i grafici riportati nel paragrafo precedente, è possibile fare le seguenti osservazioni:

- La maggior parte dei server SM-DP+ esaminati continua a fornire la medesima risposta del caso di test 1 anche nell'evenienza in cui viene inviata una euiccSignature1, un SVN (all'interno di euiccInfo2), un euiccCiPKIdListForVerification (all'interno di euiccInfo2), un euiccCiPKIdListForSigning (all'interno di euiccInfo2), un transactionId (all'interno di euiccSigned1), un smdpAddress (all'interno di euiccSigned1) oppure una serverChallenge (all'interno di euiccSigned1) non valida. Con ogni probabilità, ciò è indice del fatto che molti server effettuano prima un controllo sui certificati inviati dal client (eumCertificate, euiccCertificate) e solo poi sulle informazioni appena elencate. Tuttavia, è anche vero che i server di Sysmocom e Osmo-smdpp continuano a dare un esito positivo a seguito della manomissione dell'SVN, dell'euiccCiPKIdListForVerification, dell'euiccCiPKIdListForSigning, del transactionId e dell'smdpAddress: in altre parole, questo sottoinsieme di campi non viene valutato da Sysmocom e Osmo-smdpp. Ma poiché questi ultimi sono solo dei server di test, di base non vi è alcun problema. La faccenda diverrebbe più interessante se anche altri server SM-DP+ (e.g. quelli reali) fossero caratterizzati dalle stesse mancanze, seppure i campi in questione, se presi singolarmente, non sono particolarmente critici. Di fatto, SVN, euiccCiPKIdListForVerification ed euiccCiPKIdListForSigning vengono già negoziati a partire dal primo messaggio del client (initiateAuthentication) per cui all'interno di euiccInfo2, nel secondo messaggio (authenticateClient), sono dei campi ridondanti; anche l'hostname del server (smdpAddress) deve essere inviato già nel messaggio initiateAuthentication; infine, l'ID di transazione viene riportato nel messaggio authenticateClient anche esternamente alla struttura euiccSigned1, per cui è comprensibile, anche se probabilmente poco sicuro, che i server verifichino soltanto una delle due istanze dell'ID di transazione all'interno del messaggio.

- Per ovvie ragioni, il caso di test 6.2, che consiste nell'invio di un Matching ID errato, coinvolge esclusivamente i server SM-DP+ di test, i quali sono gli unici a mettere a disposizione dei profili di test coi relativi Matching ID. L'unico server di test escluso da questa analisi è Infineon, che risponde deterministicamente con HTTP_BAD_REQUEST. Gli unici due server che nel caso di test 6.2 si comportano esattamente come nel caso di test 1 sono quelli di Stork & Google e Truphone, i quali evidentemente effettuano prima il controllo sui certificati ricevuti da parte del client e solo successivamente il controllo sul Matching ID.

- Gli unici casi in cui i server SM-DS forniscono un errore previsto dalle specifiche e non vanno in crash dando un HTTP_INTERNAL_SERVER_ERROR sono:
  - invio di uno stato di errore nel messaggio authenticateClient;
  - invio di una certificate chain diversa da quella già stabilita in euiccCiPKIdToBeUsed;
  - invio di un hostname (serverAddress) non valido nel messaggio initiateAuthentication;
  - invio di un SVN non valido nel messaggio initiateAuthentication;
  - invio di un euiccCiPKIdListForVerification o di un euiccCiPKIdListForSigning non valido nel messaggio initiateAuthentication.

  Se invece i server SM-DS ricevono un messaggio inatteso, tendono a rispondere con HTTP_NOT_FOUND. Ciò fa pensare che l'implementazione dei server SM-DS che si possono trovare in rete sia incompleta e non del tutto affidabile.
- Se il client invia un'euiccChallenge non conforme alle specifiche (i.e. di una lunghezza diversa dai 16 byte), questa viene rifiutata solo dai server reali e da Truphone; d'altra parte, i server SM-DS, il server di Stork & Google e il simulatore accettano anche euiccChallenge di un solo byte, lasciando eventualmente maggiore spazio di manovra ad attacchi di tipo spoofing; infine, Sysmocom e Osmo-smdpp vanno in crash se ricevono una challenge di lunghezza differente dai 16 byte.
- Senza contare euiccChallenge, tutti i server rispondono con un opportuno messaggio di errore nel caso in cui ricevano il messaggio initiateAuthentication con almeno un campo non valido (serverAddress, SVN, euiccCiPKIdListForVerification o euiccCiPKIdListForSigning). L'unica eccezione è data dai server di test Sysmocom e Osmo-smdpp che accettano qualunque SVN, il che è comprensibile.
- Il protocollo studiato e le relative implementazioni dei server esaminati nei vari esperimenti prevedono che, quando si verifica una condizione di errore (e.g. un end-host invia un valore errato o non valido), l'entità che riscontra tale condizione risponda con un messaggio di errore in cui, mediante il subject code e il reason code, viene specificato con esattezza cos'è andato storto. Tuttavia, inviare dei messaggi di errore così dettagliati potrebbe aprire le porte a delle vulnerabilità, poiché potenzialmente potrebbe fornire a un attaccante delle informazioni in eccesso riguardo ciò che sta avvenendo durante la comunicazione.

5.4 Ottenimento dei client da testare

La selezione dei client da testare può essere effettuata sulla base delle risorse (i.e. eUICC e dispositivi mobili) che si hanno a disposizione durante la fase di sperimentazione. In particolare, sono stati individuati i seguenti client:

- eUICC programmabile e LPA EasyEUICC installati manualmente in un Google Pixel 6 (mostrato in figura 5.20);
- eUICC e LPA direttamente forniti dal Google Pixel 6.

Assumendo di voler considerare anche il lettore di SIM e il simulatore del client come possibili partecipanti dei successivi esperimenti, la tabella 5.4 mostra tutti i client che verranno impiegati durante l'esecuzione dei test.

Tabella 5.4: Elenco dei client coinvolti negli esperimenti.

Nome	**Descrizione**	**Note**
Google Pixel 6	Client reale	Utilizzerà una certificate chain di test.
EasyEUICC	eUICC programmabile + LPA EasyEUICC	Utilizzerà una certificate chain di test.
LPAC	eUICC programmabile + lpac	Utilizzerà una certificate chain di test.
Simulatore	Simulatore lato client precedentemente implementato	Utilizzerà una certificate chain di test.

5.5 Risposte dei client testati

5.5.1 Casi di test definiti

I casi di test che sono stati implementati per valutare l'interazione tra il simulatore del server e i client a disposizione sono riportati nella tabella 5.5.

Tabella 5.5: Elenco dei casi di test definiti per i client a disposizione.

ID	Descrizione
1	Il simulatore del server si comporta perfettamente come da specifiche.
2	Manomissione del messaggio getBoundProfilePackageResponse: invio di un profilo dummy non crittografato.
3.1	Manomissione del messaggio getBoundProfilePackageResponse: utilizzo di un Transaction ID non valido all'interno del sotto-campo initialiseSecureChannelRequest.
3.2	Manomissione del messaggio getBoundProfilePackageResponse: utilizzo di una chiave pubblica one-time smdpOtpk casuale e non generata tramite ECDSA all'interno del sotto-campo initialiseSecureChannelRequest.
3.3	Manomissione del messaggio getBoundProfilePackageResponse: calcolo errato della signature (smdpSign) all'interno del sotto-campo initialiseSecureChannelRequest.
4.1	Manomissione del messaggio authenticateClientResponse: utilizzo di un iccid casuale all'interno del campo profileMetadata.
4.2	Manomissione del messaggio authenticateClientResponse: eliminazione di profileClass, notificationConfigurationInfo e profileOwner dal campo profileMetadata.
4.3	Manomissione del messaggio authenticateClientResponse: inserimento delle Profile Policy Rules all'interno del campo profileMetadata.
5	Manomissione del messaggio authenticateClientResponse: ccRequiredFlag = True all'interno del campo smdpSigned2.
6.1	Manomissione del messaggio authenticateClientResponse: utilizzo di un Transaction ID non valido all'interno del campo smdpSigned2.
6.2	Manomissione del messaggio authenticateClientResponse: campo transactionId non valido direttamente all'interno del messaggio.
7	Manomissione del messaggio authenticateClientResponse: calcolo errato dalla signature (smdpSignature2).
8	Manomissione del messaggio authenticateClientResponse: invio di un certificato smdpCertificate diverso da quello previsto.
9.1	Manomissione del messaggio initiateAuthenticationResponse: utilizzo di una euiccChallenge non valida all'interno del campo serverSigned1.
9.2	Manomissione del messaggio initiateAuthenticationResponse: utilizzo di una serverChallenge malforme (e.g. 0x00) all'interno del campo serverSigned1.
9.3	Manomissione del messaggio initiateAuthenticationResponse: utilizzo di un serverAddress non corrispondente all'hostname realmente assegnato al simulatore del server.
10.1	Manomissione del messaggio initiateAuthenticationResponse: utilizzo di Transaction ID differenti all'interno del campo serverSigned1 e direttamente nel messaggio.
10.2	Manomissione del messaggio initiateAuthenticationResponse: utilizzo di un Transaction ID malforme direttamente all'interno del messaggio (e.g. 0x00).
11	Manomissione del messaggio initiateAuthenticationResponse: calcolo errato dalla signature (serverSignature1).
12	Manomissione del messaggio initiateAuthenticationResponse: invio di un certificato serverCertificate diverso da quello previsto.
13	Manomissione del messaggio initiateAuthenticationResponse: invio di una certificate chain identificata da euiccCiPKIdToBeUsed che non rientra in euiccCiPKIdListForVerification o euiccCiPKIdListForSigning.
14.1	Invio di getBoundProfilePackageResponse come primo messaggio.
14.2	Invio di getBoundProfilePackageResponse come secondo messaggio.

Figura 5.20: Il Google Pixel 6 coinvolto negli esperimenti.

Caso di test 1

Consiste nel comunicare col client (in particolare con l'LPA) seguendo fedelmente le specifiche, senza alcuna manomissione nel protocollo o nei messaggi inviati. Di seguito è descritto il comportamento dei client in esame in questo caso di test.

- **Google Pixel 6**: impossibile connettersi alla rete: non viene inviato nemmeno il messaggio initiateAuthentication. Ciò potrebbe essere dovuto al fatto che il client in questione non supporta i server SM-DP+ di test.
- **EasyEUICC**: *errorResult* al quarto messaggio di richiesta (pendingNotification), con *scp03tSecurityError*, che corrisponde all'errore riscontrato e non risolto durante la fase di validazione.
- **LPAC**: errore dopo il terzo messaggio di risposta (getBoundProfilePackageResponse), con *scp03t_security_error*, che è esattamente l'errore riscontrato e non risolto durante la fase di validazione.
- **Simulatore**: il profilo viene installato con successo.

Caso di test 2

Consiste nel manomettere il messaggio getBoundProfilePackageResponse inserendo all'interno dei sotto-campi firstSequenceOf87, sequenceOf88, secondSequenceOf87 e sequenceOf86 dei byte casuali (e.g. tutti pari a 0x00), in modo tale da rappresentare dei metadati e un profilo dummy non crittografato.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: *errorResult* al quarto messaggio di richiesta (pendingNotification), con *scp03tSecurityError*.
- **LPAC**: errore dopo il terzo messaggio di risposta (getBoundProfilePackageResponse), con *scp03t_security_error*.
- **Simulatore**: Common Cancel Session Procedure completata con successo a seguito del terzo messaggio di risposta da parte del server (getBoundProfilePackageResponse).

Caso di test 3.1

Consiste nel manomettere il messaggio getBoundProfilePackageResponse ponendo il Transaction ID all'interno del sotto-campo initialiseSecureChannelRequest pari a un valore non valido.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: *errorResult* al quarto messaggio di richiesta (pendingNotification), con *invalidTransactionId*.
- **LPAC**: errore dopo il terzo messaggio di risposta (getBoundProfilePackageResponse), con *invalid_transaction_id*.
- **Simulatore**: *errorResult* al quarto messaggio di richiesta (pendingNotification), con *invalidTransactionId*.

Caso di test 3.2

Consiste nel manomettere il messaggio getBoundProfilePackageResponse inserendo all'interno del sotto-campo initialiseSecureChannelRequest una chiave pubblica one-time smdpOtpk non generata tramite ECDSA, bensì espressa sottoforma di byte casuali.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: interruzione della comunicazione dopo il terzo messaggio di risposta (getBoundProfilePackageResponse).
- **LPAC**: errore dopo il terzo messaggio di risposta (getBoundProfilePackageResponse), con *unknown*.
- **Simulatore**: il profilo viene installato con successo.

Caso di test 3.3

Consiste nel manomettere il messaggio getBoundProfilePackageResponse ponendo il campo smdpSign pari a un valore diverso dalla signature ottenuta firmando tutte le altre entry del sotto-campo initialiseSecureChannelRequest con la chiave privata del server SM-DP+.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: *errorResult* al quarto messaggio di richiesta (pendingNotification), con *invalidSignature*.
- **LPAC**: errore dopo il terzo messaggio di risposta (getBoundProfilePackageResponse), con *invalid_signature*.
- **Simulatore**: *errorResult* al quarto messaggio di richiesta (pendingNotification), con *invalidSignature*.

Caso di test 4.1

Consiste nel manomettere il messaggio authenticateClientResponse inserendo all'interno del campo profileMetadata un iccid casuale e non quello originariamente assegnato al profilo che il client vuole scaricare.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: *errorResult* al quarto messaggio di richiesta (pendingNotification), con *scp03tSecurityError*.
- **LPAC**: errore dopo il terzo messaggio di risposta (getBoundProfilePackageResponse), con *scp03t_security_error*.
- **Simulatore**: il profilo viene installato con successo.

Caso di test 4.2

Consiste nel manomettere il messaggio authenticateClientResponse eliminando le entry relative a profileClass, notificationConfigurationInfo e profileOwner dal campo profileMetadata. Trattandosi solo di informazioni opzionali, questo caso di test non dovrebbe produrre un risultato differente dal caso di test 1.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: *errorResult* al quarto messaggio di richiesta (pendingNotification), con *scp03tSecurityError*.
- **LPAC**: errore dopo il terzo messaggio di risposta (getBoundProfilePackageResponse), con *scp03t_security_error*.
- **Simulatore**: il profilo viene installato con successo.

Caso di test 4.3

Consiste nel manomettere il messaggio authenticateClientResponse aggiungendo una entry relativa alle Profile Policy Rules all'interno del campo profileMetadata. In particolare, tale entry indica l'attivazione di entrambe le Profile Policy Rules (PPR1, PPR2).

- **Google Pixel 6**: n.d.
- **EasyEUICC**: *errorResult* al quarto messaggio di richiesta (pendingNotification), con *scp03tSecurityError*.
- **LPAC**: errore dopo il terzo messaggio di risposta (getBoundProfilePackageResponse), con *scp03t_security_error*.
- **Simulatore**: il profilo viene installato con successo dopo una conferma all'utente sulla sottoscrizione delle PPR.

Caso di test 5

Consiste nel manomettere il messaggio authenticateClientResponse ponendo ccRequiredFlag all'interno del campo smdpSigned2 pari a True, in modo tale da richiedere al client l'immissione di un Confirmation Code.

- **Google Pixel 6**: n.d.
- **EasyEUICC**:
  - Caso in cui l'utente ha immesso un qualunque Confirmation Code → *errorResult* al quarto messaggio di richiesta (pendingNotification), con *scp03tSecurityError*.
  - Caso in cui l'utente non ha immesso alcun Confirmation Code → *prepareDownloadResponseError* al terzo messaggio di richiesta (getBoundProfilePackage), con *undefinedError*.
- **LPAC**:
  - Caso in cui l'utente ha immesso un qualunque Confirmation Code → errore dopo il terzo messaggio di risposta (getBoundProfilePackageResponse), con *scp03t_security_error*.
  - Caso in cui l'utente non ha immesso alcun Confirmation Code → errore generico dopo il secondo messaggio di risposta (authenticateClientResponse).
- **Simulatore**:
  - Caso in cui l'utente ha immesso un qualunque Confirmation Code → il profilo viene installato con successo.
  - Caso in cui l'utente non ha immesso alcun Confirmation Code → Common Cancel Session Procedure completata con successo a seguito del secondo messaggio di risposta da parte del server (authenticateClientResponse).

Caso di test 6.1

Consiste nel manomettere il messaggio authenticateClientResponse ponendo il Transaction ID all'interno del campo smdpSigned2 pari a un valore non valido.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: *prepareDownloadResponseError* al terzo messaggio di richiesta (getBoundProfilePackage), con *invalidTransactionId*.
- **LPAC**: *prepareDownloadResponseError* al terzo messaggio di richiesta (getBoundProfilePackage), con *invalidTransactionId*.
- **Simulatore**: *prepareDownloadResponseError* al terzo messaggio di richiesta (getBoundProfilePackage), con *invalidTransactionId*.

Caso di test 6.2

Consiste nel manomettere il messaggio authenticateClientResponse ponendo il Transaction ID pari a un valore non valido direttamente all'interno messaggio stesso.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: *errorResult* al quarto messaggio di richiesta (pendingNotification), con *scp03tSecurityError*.
- **LPAC**: errore dopo il terzo messaggio di risposta (getBoundProfilePackageResponse), con *scp03t_security_error*.
- **Simulatore**: *prepareDownloadResponseError* al terzo messaggio di richiesta (getBoundProfilePackage), con *invalidTransactionId*.

Caso di test 7

Consiste nel manomettere il messaggio authenticateClientResponse ponendo il campo smdpSignature2 pari a un valore diverso dalla signature ottenuta firmando smdpSigned2+euiccSignature1 con la chiave privata del server SM-DP+.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: *prepareDownloadResponseError* al terzo messaggio di richiesta (getBoundProfilePackage), con *undefinedError*.
- **LPAC**: errore generico dopo il secondo messaggio di risposta (authenticateClientResponse).
- **Simulatore**: *prepareDownloadResponseError* al terzo messaggio di richiesta (getBoundProfilePackage), con *invalidSignature*.

Caso di test 8

Consiste nel manomettere il messaggio authenticateClientResponse inserendo all'interno del campo smdpCertificate un certificato appartenente a una certificate chain differente da quella dichiarata precedentemente in euiccCiPKIdToBeUsed.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: *prepareDownloadResponseError* al terzo messaggio di richiesta (getBoundProfilePackage), con *invalidCertificate*.
- **LPAC**: *prepareDownloadResponseError* al terzo messaggio di richiesta (getBoundProfilePackage), con *invalidCertificate*.
- **Simulatore**: *prepareDownloadResponseError* al terzo messaggio di richiesta (getBoundProfilePackage), con *invalidCertificate*.

Caso di test 9.1

Consiste nel manomettere il messaggio initiateAuthenticationResponse ponendo il sotto-campo euiccChallenge del campo serverSigned1 pari a un valore che non corrisponde alla challenge inviata dal client nel messaggio precedente (initiateAuthentication).

- **Google Pixel 6**: n.d.
- **EasyEUICC**: *authenticateResponseError* al secondo messaggio di richiesta (authenticateClient), con *euiccChallengeMismatch*.
- **LPAC**: *authenticateResponseError* al secondo messaggio di richiesta (authenticateClient), con *euiccChallengeMismatch*.
- **Simulatore**: *authenticateResponseError* al secondo messaggio di richiesta (authenticateClient), con *euiccChallengeMismatch*.

Caso di test 9.2

Consiste nel manomettere il messaggio initiateAuthenticationResponse utilizzando una serverChallenge malforme (e.g. 0x00, che ha una lunghezza differente dai 16 byte di default, e comunque troppo ridotta per svolgere il ruolo di challenge).

- **Google Pixel 6**: n.d.
- **EasyEUICC**: interruzione della comunicazione dopo il primo messaggio di risposta (initiateAuthenticationResponse).
- **LPAC**: errore generico dopo il primo messaggio di risposta (initiateAuthenticationResponse).
- **Simulatore**: il profilo viene installato con successo.

Caso di test 9.3

Consiste nel manomettere il messaggio initiateAuthenticationResponse ponendo il sotto-campo serverAddress del campo serverSigned1 pari a una stringa che non corrisponde all'effettivo hostname del simulatore del server.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: invio del serverAddress errato nel secondo messaggio di richiesta (authenticateClient).
- **LPAC**: invio del serverAddress errato nel secondo messaggio di richiesta (authenticateClient).
- **Simulatore**: *authenticateResponseError* al secondo messaggio di richiesta (authenticateClient), con *invalidOid*.

Caso di test 10.1

Consiste nel manomettere il messaggio initiateAuthenticationResponse ponendo il Transaction ID all'interno del campo serverSigned1 pari a un valore differente dal Transaction ID situato direttamente all'interno del messaggio stesso.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: invio del Transaction ID errato nel secondo messaggio di richiesta (authenticateClient).
- **LPAC**: invio del Transaction ID errato nel secondo messaggio di richiesta (authenticateClient).
- **Simulatore**: *authenticateResponseError* al secondo messaggio di richiesta (authenticateClient), con *undefinedError*.

Caso di test 10.2

Consiste nel manomettere il messaggio initiateAuthenticationResponse utilizzando un Transaction ID malforme (e.g. 0x00, che ha una lunghezza differente dai 16 byte di default, e comunque troppo ridotta per svolgere il ruolo di challenge).

- **Google Pixel 6**: n.d.
- **EasyEUICC**: *errorResult* al quarto messaggio di richiesta (pendingNotification), con *scp03tSecurityError*.
- **LPAC**: errore dopo il terzo messaggio di risposta (getBoundProfilePackageResponse), con *scp03t_security_error*.
- **Simulatore**: il profilo viene installato con successo.

Caso di test 11

Consiste nel manomettere il messaggio initiateAuthenticationResponse ponendo il campo serverSignature1 pari a un valore diverso dalla signature ottenuta firmando serverSigned1 con la chiave privata del server SM-DP+.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: *authenticateResponseError* al secondo messaggio di richiesta (authenticateClient), con *undefinedError*.
- **LPAC**: errore generico dopo il primo messaggio di risposta (initiateAuthenticationResponse).
- **Simulatore**: *authenticateResponseError* al secondo messaggio di richiesta (authenticateClient), con *invalidSignature*.

Caso di test 12

Consiste nel manomettere il messaggio initiateAuthenticationResponse inserendo all'interno del campo serverCertificate un certificato appartenente a una certificate chain differente da quella dichiarata in euiccCiPKIdToBeUsed.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: *authenticateResponseError* al secondo messaggio di richiesta (authenticateClient), con *invalidOid*.
- **LPAC**: *authenticateResponseError* al secondo messaggio di richiesta (authenticateClient), con *invalidOid*.
- **Simulatore**: *authenticateResponseError* al secondo messaggio di richiesta (authenticateClient), con *invalidCertificate*.

Caso di test 13

Consiste nel manomettere il messaggio initiateAuthenticationResponse ponendo il campo euiccCiPKIdToBeUsed pari a un valore che non rientra né in euiccCiPKIdListForVerification né in euiccCiPKIdListForSigning, in modo tale che venga indicata una certificate chain non inclusa tra quelle supportate dal client.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: *authenticateResponseError* al secondo messaggio di richiesta (authenticateClient), con *ciPKUnknown*.
- **LPAC**: *authenticateResponseError* al secondo messaggio di richiesta (authenticateClient), con *ciPKUnknown*.
- **Simulatore**: *authenticateResponseError* al secondo messaggio di richiesta (authenticateClient), con *ciPKUnknown*.

Caso di test 14.1

Consiste nell'inviare al client getBoundProfilePackageResponse come primo messaggio di risposta.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: interruzione della comunicazione dopo il messaggio di risposta getBoundProfilePackageResponse.
- **LPAC**: errore dopo il messaggio di risposta getBoundProfilePackageResponse, con *unknown*.
- **Simulatore**: interruzione della comunicazione dopo il messaggio di risposta getBoundProfilePackageResponse.

Caso di test 14.2

Consiste nell'inviare al client getBoundProfilePackageResponse subito dopo aver ricevuto il messaggio di authenticateClient, saltando così il messaggio di authenticateClientResponse.

- **Google Pixel 6**: n.d.
- **EasyEUICC**: interruzione della comunicazione dopo il messaggio di risposta getBoundProfilePackageResponse.
- **LPAC**: errore dopo il messaggio di risposta getBoundProfilePackageResponse, con *unknown*.
- **Simulatore**: interruzione della comunicazione dopo il messaggio di risposta getBoundProfilePackageResponse.

5.5.2 Certificati esibiti dai client

Anche contestualmente all'analisi delle risposte dei client può tornare utile analizzare i certificati esibiti dai client stessi e, in particolare, CERT.EUICC.SIG (proprio dell'eUICC) e CERT.EUM.SIG (proprio dell'EUM). Tali certificati possono essere ispezionati solo per i client che hanno inviato con successo il messaggio di richiesta authenticateClient in almeno uno dei casi di test sopra citati. È perciò esclusa da questa trattazione la coppia (eUICC originale, LPA originale) del Google Pixel 6. Di seguito sono riportati i principali campi dei certificati degli altri tre client, a partire dai CERT.EUICC.SIG; poiché i client EasyEUICC e LPAC condividono la medesima eUICC, devono necessariamente aver esposto lo stesso certificato.

Certificato di EasyEUICC e LPAC

- **Algorithm**: ecdsaWithSHA256;
- **issuer countryName**: ES;
- **issuer organizationName**: RSP TEST EUM;
- **issuer commonName**: EUM Test;
- **validity notBefore**: 2024-01-25;
- **validity notAfter**: 7499-11-18;
- **countryName**: DE;
- **organizationName**: sysmocom RSP Test EUM;
- **commonName**: sysmoEUICC-C2T Test eUICC.

Certificato del simulatore del client

- **Algorithm**: ecdsaWithSHA256;
- **issuer countryName**: ES;
- **issuer organizationName**: RSP TEST EUM;
- **issuer commonName**: EUM Test;
- **validity notBefore**: 2020-04-01;
- **validity notAfter**: 7496-01-24;
- **countryName**: ES;
- **organizationName**: RSP Test EUM;
- **commonName**: Test eUICC.

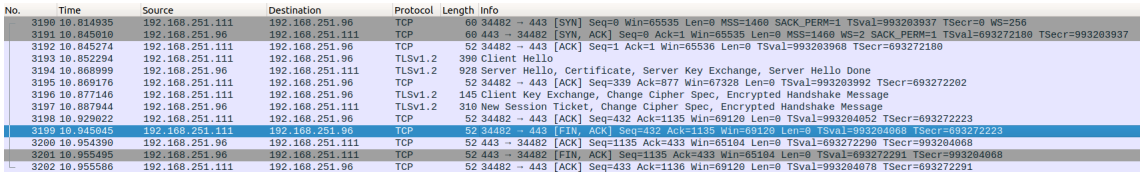
Naturalmente, sarebbe stato anche possibile far comunicare il simulatore del server col simulatore del client utilizzando la medesima certificate chain custom introdotta durante la fase degli esperimenti sui server SM-DP+ reali: non avrebbe fatto differenza.
Inoltre, dai campi *issuer countryName*, *issuer organizationName* e *issuer commonName*, è possibile constatare che EasyEUICC e LPAC e, per costruzione, il simulatore del client condividono il medesimo EUM, il cui certificato CERT.EUM.SIG è descritto qui di seguito.

Certificato dell'EUM

- **Algorithm**: ecdsaWithSHA256;
- **issuer countryName**: IT;
- **issuer organizationName**: RSPTEST;
- **issuer organizationalUnitName**: TESTCERT;
- **issuer commonName**: Test CI;
- **validity notBefore**: 2020-04-01;
- **validity notAfter**: 2054-03-24;
- **countryName**: ES;
- **organizationName**: RSP Test EUM;
- **commonName**: EUM Test;
- **crLDistributionPoints**: http://ci.test.example.com/crl-B.crl;
- **authorityKeyIdentifier**: sgp.

5.5.3 Analisi delle risposte ricevute

Per quanto concerne l'eUICC e l'applicazione LPA integrati nel Google Pixel 6, si potrebbe provare a capire quali siano le ragioni per cui la connessione non va a buon fine. È stato dunque catturato il traffico di rete nella comunicazione tra il dispositivo mobile e il simulatore del server ed è stato analizzato con Wireshark, come testimoniato dalla figura 5.21. Com'è possibile vedere, avviene l'intera procedura di handshake di TLS e, dopodiché, nel pacchetto evidenziato in blu nella figura, il dispositivo mobile invia un messaggio di [FIN, ACK] per interrompere la connessione TLS: qualcosa è andato storto durante l'handshake, come una configurazione non supportata da una delle due controparti. In particolare, sembra che il Google Pixel 6 abbia i certificati TLS caratterizzati dai PIN, grazie ai quali il dispositivo mobile è in grado di connettersi esclusivamente agli host che detengono certificati TLS specifici, e non con gli host che ne possiedono uno valido ma non previsto dai PIN.



No.	Time	Source	Destination	Protocol	Length	Info
3190	10.814935	192.168.251.111	192.168.251.96	TCP	60	34482 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=993203937 TSecr=0 WS=256
3191	10.845010	192.168.251.96	192.168.251.111	TCP	60	443 → 34482 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=2 SACK_PERM=1 TSval=693272180 TSecr=993203937
3192	10.845274	192.168.251.111	192.168.251.96	TCP	52	34482 → 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=993203968 TSecr=693272180
3193	10.852294	192.168.251.111	192.168.251.96	TLSv1.2	390	Client Hello
3194	10.868999	192.168.251.96	192.168.251.111	TLSv1.2	928	Server Hello, Certificate, Server Key Exchange, Server Hello Done
3195	10.869176	192.168.251.111	192.168.251.96	TCP	52	34482 → 443 [ACK] Seq=339 Ack=877 Win=67328 Len=0 TSval=993203992 TSecr=693272202
3196	10.877146	192.168.251.111	192.168.251.96	TLSv1.2	145	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
3197	10.887944	192.168.251.96	192.168.251.111	TLSv1.2	310	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
3198	10.929022	192.168.251.111	192.168.251.96	TCP	52	34482 → 443 [ACK] Seq=432 Ack=1135 Win=69120 Len=0 TSval=993204052 TSecr=693272223
3199	10.945045	192.168.251.111	192.168.251.96	TCP	52	34482 → 443 [FIN, ACK] Seq=432 Ack=1135 Win=69120 Len=0 TSval=993204068 TSecr=693272223
3200	10.954390	192.168.251.96	192.168.251.111	TCP	52	443 → 34482 [ACK] Seq=1135 Ack=433 Win=65104 Len=0 TSval=693272290 TSecr=993204068
3201	10.955495	192.168.251.96	192.168.251.111	TCP	52	443 → 34482 [FIN, ACK] Seq=1135 Ack=433 Win=65104 Len=0 TSval=693272291 TSecr=993204068
3202	10.955586	192.168.251.111	192.168.251.96	TCP	52	34482 → 443 [ACK] Seq=433 Ack=1136 Win=69120 Len=0 TSval=993204078 TSecr=693272291

Figura 5.21: Cattura Wireshark della comunicazione tra il Google Pixel 6 e il simulatore del server.

Dalla figura 5.21 emerge anche che la versione di TLS che stava per essere utilizzata è la 1.2: una versione ancora accettata dalla GSM Association ma non la migliore possibile. È molto probabile che la maggioranza delle comunicazioni basate sul protocollo SM-DP+ si appoggino tuttora sulla versione 1.2 di TLS.

Passando invece ai client sui quali è stato possibile testare il protocollo SM-DP+, balza subito all'occhio la perfetta analogia fra le risposte fornite da EasyEUICC nei vari esperimenti e quelle fornite da LPAC. Ciò era assolutamente prevedibile, dal momento in cui i due client fanno uso della medesima eUICC programmabile e differiscono soltanto dall'applicazione LPA. Si hanno dunque solo due client significativi dal punto di vista delle risposte ricevute, di cui uno è identificabile o con EasyEUICC o con LPAC, mentre l'altro è il simulatore del client sviluppato nel presente lavoro, per cui non è neanche un client in commercio. Conseguentemente, non ha molto senso graficare il comportamento dei client nei vari esperimenti come è stato fatto per i server SM-DP+; piuttosto, possono essere direttamente fatte delle osservazioni.

- All'interno del messaggio di risposta authenticateClientResponse si hanno due istanze di Transaction ID: una interna alla struttura smdpSigned2 e una esterna. Dai casi di test 6.1 e 6.2 emerge che EasyEUICC e LPAC controllano la correttezza soltanto dell'ID di transazione situato in smdpSigned2. Si tratta di una situazione perfettamente analoga a quella riscontrata durante l'esecuzione dei test dei server: anche nei messaggi di richiesta dei client si ha il Transaction ID ripetuto due volte e anche i server SM-DP+ verificano la correttezza del valore di una sola istanza dell'ID di transazione.

- Anche all'interno del messaggio di risposta initiateAuthenticationResponse si hanno due istanze di Transaction ID: una interna alla struttura serverSigned1 e una esterna. Dal caso di test 10.1 si ha di nuovo che, dal punto di vista di EasyEUICC e LPAC, fa fede esclusivamente l'ID di transazione interno. Infatti, nel caso di test 10.1 il valore dei due ID è differente e i client acquisiscono quello situato in serverSigned1.

- Nel caso di test 10.2, il Transaction ID proposto dal simulatore del server è composto da un unico byte e viene accettato così com'è dai client testati. Tuttavia, trattandosi di un nonce, sarebbe stato ideale forzare l'ID di transazione ad avere una certa entropia minima (i.e. una certa dimensione minima), in modo tale da evitare il più possibile il riuso del medesimo ID in più connessioni successive.

- All'interno del messaggio initiateAuthenticationResponse viene specificato l'hostname del server SM-DP+ che sta inviando tale messaggio. Sulla base del caso di test 9.3, sembra succedere ciò che viene descritto qui di seguito. Il client contatta il server SM-DP+ specificando nel messaggio initiateAuthentication l'hostname effettivo del server mediante il campo smdpAddress; il server risponde specificando nel messaggio initiateAuthenticationResponse un hostname differente; a quel punto, il client si convince che l'hostname corretto del server sia quello riportato in initiateAuthenticationResponse. Tale fenomeno può essere giustificato dal fatto che ciasucn server può avere più hostname (alias). Tuttavia, cià potrebbe far sospettare che l'hostname del server potrebbe essere spoofato all'interno del messaggio initiateAuthenticationResponse.

- Le condizioni di errore legate al calcolo errato della signature sono le uniche che vengono elaborate di EasyEUICC e LPAC come errori generici (o *undefined*). In tutti gli altri casi, i client specificano sempre i dettagli sull'errore riscontrato. Si tratta nuovamente di un comportamento che è stato già osservato durante i test dei server SM-DP+.

Comunque sia, nel complesso, nonostante siano stati esaminati perlopiù client di test, il loro comportamento osservato rispecchia abbastanza fedelmente le specifiche di GSMA e, quindi, il comportamento atteso.

5.6 Scenari di attacchi ipotetici

[TODO]

Capitolo 6

Conclusione

6.1 Resoconto dei risultati ottenuti

[TODO]

6.2 Sviluppi futuri

[TODO]

Bibliografia

[1] Corcom. *"Telefonia mobile, cosa sono le eSim? E perché sono più sicure?"*, 2023. https://corrierecomunicazioni.it/telco/telefonia-mobile-cosa-sono-le-esim-e-perche-sono-piu-sicure/.

[2] GSM Association. *"The what and how of Remote SIM Provisioning"*, 2018. https://gsma.com/esim/wp-content/uploads/2018/12/esim-whitepaper.pdf.

[3] GSM Association. *"RSP Technical Specification Version 3.0"*, 2022. https://gsma.com/esim/wp-content/uploads/2022/10/SGP.22-v3.0-1.pdf.

[4] Abu Shohel Ahmed, Aleksi Peltonen, Mohit Sethi, and Tuomas Aura. *"Security Analysis of the Consumer Remote SIM Provisioning Protocol"*, 2022. https://arxiv.org/pdf/2211.15323.pdf.

[5] GSM Association. *"RSP Technical Specification Version 2.0"*, 2016. https://gsma.com/newsroom/wp-content/uploads/SGP.22_v2.0.pdf.

[6] Team di Android. *"Implementing eSIM"*, 2023. https://source.android.com/docs/core/connect/esim-overview.

[7] IETF. *"The Transport Layer Security (TLS) Protocol Version 1.2"*, 2008. https://rfc-editor.org/rfc/rfc5246.

[8] IETF. *"The Transport Layer Security (TLS) Protocol Version 1.3"*, 2018. https://rfc-editor.org/rfc/rfc8446.

[9] Bruno Blanchet. *"Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif"*, 2016. https://www.nowpublishers.com/article/Details/SEC-004.

[10] IETF. *"The Base16, Base32, and Base64 Data Encodings"*, 2006. https://datatracker.ietf.org/doc/html/rfc4648.

[11] OSS Nokalva. *"Distinguished Encoding Rules"*, 2022. https://www.oss.com/asn1/resources/asn1-made-simple/asn1-quick-reference/distinguished-encoding-rules.html.

[12] OSS Nokalva. *"Basic Encoding Rules"*, 2012. https://www.oss.com/asn1/resources/asn1-made-simple/asn1-quick-reference/basic-encoding-rules.html.

[13] IETF. *"ASN.1 Translation"*, 2010. https://datatracker.ietf.org/doc/html/rfc6025.

[14] AJ ONeal. *"ASN.1 for Dummies"*, 2018. https://coolaj86.com/articles/asn1-for-dummies/.

[15] Microsoft. *"DER Encoding of ASN.1 Types"*, 2021. https://learn.microsoft.com/en-us/windows/win32/seccertenroll/about-der-encoding-of-asn-1-types.

[16] Harald Welte. *"RSPDefinitions.asn"*, 2022. https://gitea.osmocom.org/sim-card/gsma-esim-iot/src/branch/master/asn/RSPDefinitions.asn.

[17] GSM Association. *"RSP Test Certificates definitions Version 1.4"*, 2020. https://www.gsma.com/solutions-and-impact/technologies/esim/wp-content/uploads/2020/07/SGP.26-v1.4.pdf.

[18] IETF. *"Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation"*, 2010. https://datatracker.ietf.org/doc/html/rfc5639.

Ringraziamenti

[TODO]