eSIM Profile Provisioning

- Problem: profiles tightly controlled by the GSMA
- They may contain operator secrets…
- Enough money (~5 figure) buys access, powerful organizations have this capability
- But what about the rest?

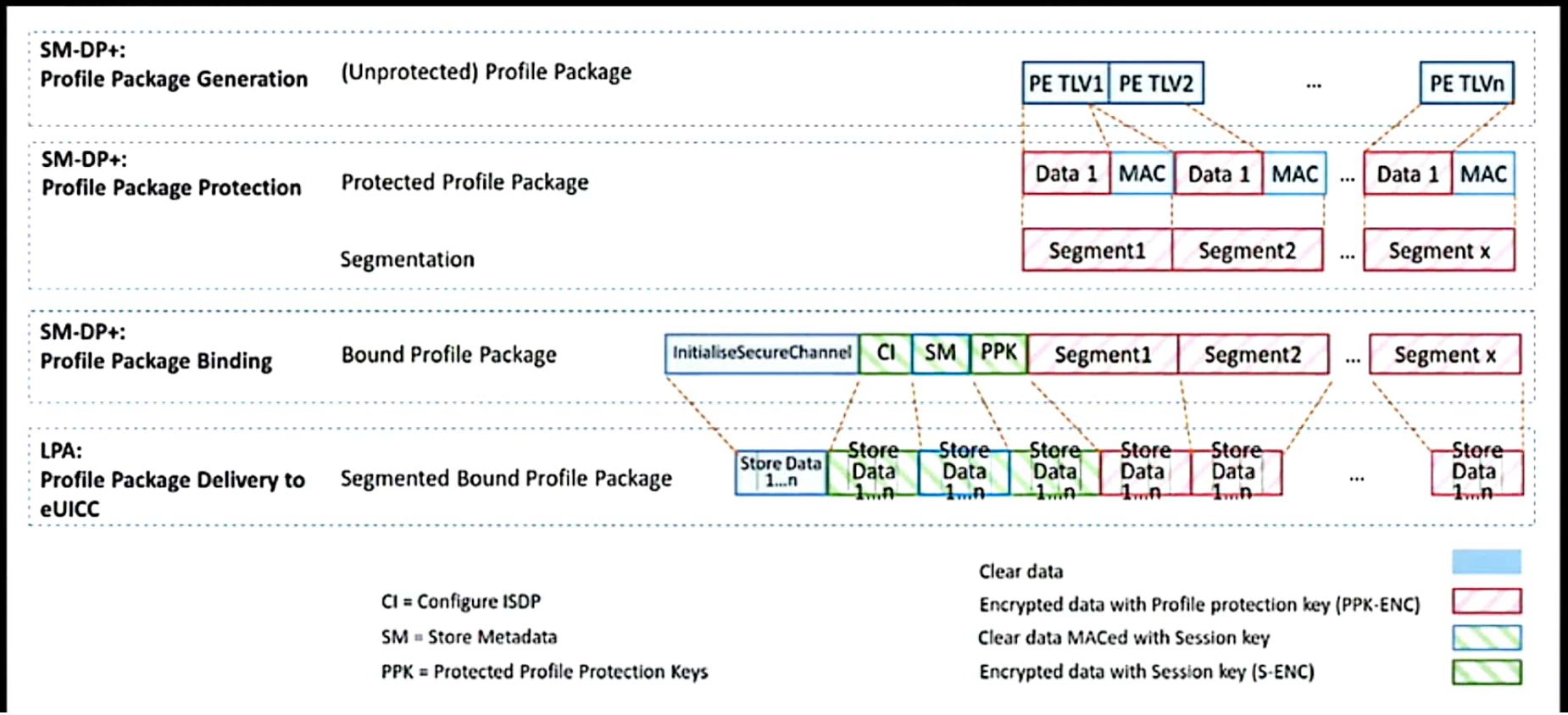
SM-DP+

- HTTPS service (Port 443) with a certificate issued by the ‘GSM Association - RSP2 Root CI1’
- Secret E2EE keys are stored in a special security domain called ECASD
  - SGP.22 hints that these keys might not be unique per device, but only per device revisions or type (see the parts about revocation)

SM-DP+

- Two layers of security:
  - ES9+: eUICC has this CA preloaded and the LPA only talks to servers “approved” by the eUICC
  - ES8+: During provisioning there is an additional E2EE protocol between SM-DP+ and eUICC, encapsulated within ES9+ and ES10{a,b,c}

eSIM Profile Formats



eUICC Security

- eUICC enforces isolation via security domains (ISDs)
- A eSIM profile exists in the context of a dedicated domain ISD-P
- Domains can have privileges, e.g. the ISD-R can manage nearly anything on the card and create new domains
- Bugs have been found, but overall the isolation is enforced quite tightly
- This means commands and applets of the currently active security domain cannot access files or send commands to applets and the filesystem in a different domain

Summary Analysis: Security of Profile Provisioning

- Did not find any obvious logic bug / blunder that doesn't involve obtaining some keys or using some implementation side bugs
- **That's it, case closed, Thank you for your time!**

Summary Analysis: Security of Profile Provisioning

- ~~That's it, case closed, Thank you for your time!~~^U
- **What's actually the goal here?** Not proving or disproving security of individual protocols or components…
- Approach this in a red team scenario, it's not about single vulns, but about **achieving your goals**!

OS / Host System Attack Surface: Windows

- The whole research started on Windows - we need **not** yet another C2
- Platform to test: Lenovo Fibocom modem with an integrated eUICC
- Motivation: We want to have our own out of band network in red teaming and adversary simulation scenarios

eSIM on Windows

- Windows 10 started supporting mobile devices and connectivity:
- Mobile plans / OneConnect apps have special capabilities for eSIM:

> The Microsoft.eSIMManagement_8wekyb3d8bbwe custom capability enables an app to perform configuration of a device's embedded SIM (eSIM). See the ESim class.

- eSIM provisioning does **NOT require Admin** by default
- Adversaries don´t even have to bring their own custom profile to get out of band network access

Red Team With the eUICC

- eUICCs are now widely deployed, but not used to their full capability (besides mobile devices, Lenovo and Dell Laptops have them now)
- Interesting technology for a red teamer:
  - Usage as C2 / Communication Channel → ✅ **We can bring our own mobile data connectivity**
  - Persistence / Run Apps → ✅ Cards run Java Applets, JavaCard / STK / USAT / CAT
  - Is it a phishing vector? → ✅ Applets can show UI elements

Red Team With the eUICC

- eSIM as a “smart card” is kind of HSM
- Like most security technology it’s dual use, can be used maliciously, can also be used to perform legitimate security tasks...
- Set some goals:
  - Goal 1: Deploy our own profile
  - Goal 2: Install Custom Apps to the eUICC
  - Goal 3: Find some vulnerabilities
  - BONUS: Create some nice tools

Goal 1: Deploy Our Own Profile

- Difficulty: Easy
- Solutions:

a. ~~Hack into some SM-DP+!~~: Obviously hacking random SM-DP+ is illegal, however we need to consider compromised infrastructure as realistic scenario

b. Obtain the secrets involved in profile provisioning: Requires vuln in an eUICC, hardware attacks or some other leak → potentially catastrophic scenario for the affected eUICC vendor

c. Lame but works: just play by the rules

Goal 1: Deploy Our Own Profile

- Paying for custom profiles and especially deployments is not really cheap
- You cannot just run your own production SM-DP+ without being certified by GSMA
- The process of creating custom profiles is quite tedious and involves nasty things such as dealing with ASN.1 and friends…
- Fortunately for a researcher there are test profiles as specified by GSMA TS.48 (currently v4.0)

✅ Goal 1: Deploy Our Own Profile

- So we can deploy GSMA TS.48 (currently v4.0) on some devices - why is that interesting?
- Remember: Any activated profile acts like a virtual classic UICC
- But we know or can even set **THE KEYS:**
  a. OTA management keys
  b. Admin keys

✅ Goal 1: Deploy Our Own Profile

- We can also preinstall Applets…or use the OTA keys to install them later, even via SMS (if we have connectivity!)
- So we cannot intercept profiles easily, but we can bring our own and modify them at our discretion → **good** for researchers
- Limitations: We don't have full control over the eUICC, "only" over the ISD-P (which is doing most of the interesting stuff anyway)

Goal 2: Install Custom Apps to the eUICC

- Well known previous research: SIM Toolkit and friends, STK - check sysmocom, sr-labs, …
- Threats classically from two directions:
  a. Malicious operators
  b. Attackers able to crack the OTA keys by attacking DES challenge-response and friends

OTA - RFM / RFA

- Multiple ways and technologies to manage the eUICC:
  - **GSMA Remote SIM Provisioning (RSP):** This is a global standard defined by the GSMA that allows for the remote management of eUICC profiles. Part of it was covered here so far, but it also enables to manage profiles *after* the provisioning is complete
  - **GlobalPlatform**: Technical standards often used in conjunction with the GSMA RSP
  - Proprietary methods from card vendors..especially if the eUICC is combined with the mobile broadband modem

High Level Overview: SPC02 Challenge-Response

- **Secure Channel Protocol '02' (SCP02)**: Method for establishing a secure channel between some on-card application and an external system
- Widely used in eUICC management
- Other algorithms SPC01, SPC03 (we skip the details)

High Level Overview: SPC02 Challenge-Response

- Simplified Protocol Overview:
  a. Challenge-Response Authentication Mechanism: This process is initiated when the off-card entity sends a challenge (random data) to the on-card application
  b. On-Card Application Response: The on-card application processes the challenge through an algorithm (AES, 3DES, older cards even DES), using a shared secret key, to generate a response
  c. Verification of Response: The off-card entity, having the same shared secret key, verifies the response. If the response matches its expectations, the entity authenticates the on-card application, thereby establishing a secure channel for communication

How to Crack 16 Byte Keys?

- Quick analysis of the challenge response:
  a. **Card(!)** sends the first challenge (cryptogram) which is encrypted with a key derived from a shared secret - did they assume a malicious card wants to capture the challenge?
  b. A diversification aims to derive a unique key from a master key that is combined with the unique card identifier
  c. 3DES still widely used

Custom Applets

- To install applets different methods were tried with varying results:
  - GlobalPlatform: Using KEY_ENC + KEY_MAC + KEY_DEK (discovered default keys or cracked) - ok for research purposes using tools such as GlobalPlatformPro
  - SMS-PP: Remote management via SMS - this worked fine, e.g. using the SIMAlliance CAP Loader
- SIM Toolkit was a compatibility and APIhell…why would anyone endure this?

STK / USAT / CAT_TP / etc. pp

- Common knowledge: AppStores of stock Android and iOS devices exclusively provide applications running on (non jailbroken) devices
- They cannot control applications on the SIM
- GSMA Specs mandate APIs that give the SIM control over the mobile device to some extent
- Vendors do not fully adhere to the specifications, probably for security reasons...

Proactive Commands

- The eUICC can send commands to the mobile device
- Examples are:
  - Profile Download Complete Command: Sent from the eUICC to the Mobile Device (MD) to indicate the completion of a profile download. It notifies the device that a new operator profile is ready for use.
  - Refresh Command: The eUICC sends this command to request the MD to refresh its internal data linked to the SIM card, for example when a network changes or operator settings are updated
  - Display Text Command: Allows the eUICC to request the MD to display a particular message on the screen. It could be used for important notifications or instructions to the user.
  - Launch Browser (!)

eUICC Applications

- Devices usually have pre installed applications such as Toolkits (SIM Toolkit STK, USAT, etc.)
- JavaCard applets can be installed on most eUICCs, the card runs a JVM that runs a subset of Java
- What we find on a card:
  - Packages: Container for the code, collection of classes
  - Instances: Object created from a class, memory is pre-allocated at install time and even the constructor is called
  - Applet: The actual Java program, it can receive and respond to APDU commands

Goal 3: Offensive Potential

- So far research focussed on testing the SIM cards with great results (e.g. SIMTester)
- Testing a mobile device via a SIM card required OTA keys (not handed out by the operators) or using a physical test card or device (e.g. via SIMTrace)
- Using physical hardware and cards is difficult when the eUICC is soldered into the mobile device
- Deploying a custom profile enables security testing without specific hardware skills and knowledge

Phishing Via Toolkit

- Toolkit applications can display text with highest priority on a mobile device
- Even when the Toolkit applications are not active
- Can also display forms
- Can send SMS back to the adversary

STK Proactive Fuzzer

- Writing a fuzzer to fuzz a hardware device so far required either:
  - A UICC emulator that needed to be physically connected
  - A physical test UICC
  - Emulation of the device's firmware (best option if done right - but a lot of effort and understanding required)
- We want to target a broad range of devices and not mess with physical hardware setups and dedicated reversing sessions for each device
- Why not implement a fuzzer as an Applet on the card?

STK Proactive Fuzzer

- Limitations:
  - APDUs are only sent from the host device
  - Proactive commands use a workaround: They use special responses to APDUs that are sent regularly
  - The card can define a polling interval
  - This makes communication slow
  - Broken APDUs and values are filtered out by the card in many cases
  - No coverage / instrumentation
- Why do we still want to do it:
  - Generic approach
  - Flexible since it's "just software" to change
  - Even if fuzzing via low powered cards is slow, they are cheap and we could scale it up to many devices!

Out of Band C2 via SMS (PoC)

- Scenario: Red Team engagement, we got temporary access (physical or payload execution)
- *Access is only temporary*
- *Cannot rely on the Internet being available*
- *Any TCP/DNS/UDP is monitored heavily*

- Solution: We bring our own connectivity and use an unusual OOB channel - SMS

Next Level Implant (not in the PoC)

- The SMS C2 is nice, but it can be reversed and analyzed
- However - as we have seen before, the eSIM profile is more of a HSM:
  - We deploy our own keys to lock it down
  - We leverage the encryption available for Toolkit applets (that´s proper AES, asymmetric crypto, etc.)
  - We implement that applet business logic inside the applet
- Good luck with forensics…
- Hint for defenders: make sure you have control over the actual eUICC security domains…although it´s still not easy to get installed applets off the card (at least it´s not documented)

Conclusion

- Research into eSIM opens up a vast space and Zoo of technologies
- We found new applications for using this technology for red teaming, research, etc.
- The environment is very tightly controlled, making it hard for researchers to get a foothold
- It is possible to follow the rabbit hole in different areas: attacking eUICCs themselves, protocol and infrastructure, attacking mobile platforms, attacking Windows desktop systems…
- **This could have been at least 3 talks**