Profiles as Payloads: Exploiting eSIM Applet Behavior

Lorenzo Valeriani[1,2,*], Simone Festa[1,3], Matteo Fanfarillo[1,2] and Giuseppe Bianchi[1,2]

[1]CNIT National Network Assurance and Monitoring (NAM) Lab, Rome, Italy
[2]University of Rome Tor Vergata, Rome, Italy
[3]Roma Servizi per la Mobilità, Rome, Italy

Abstract

As an emerging technology, eSIMs represent a major breakthrough, radically simplifying the way users obtain and manage mobile profiles. Yet, this modern workflow still supports a legacy, largely opaque component whose design has changed little in decades: applets. These small and largely understudied code snippets can be embedded in downloadable profiles remotely delivered by a fragmented and weakly regulated reseller ecosystem at consumer scale and executed without user awareness. In this paper, we demonstrate how embedded applets can extract sensitive information without consent and how they can disrupt basic device usability. We present two attacks. The first silently exfiltrates location data via text messages, without visibility to the user. The second is a novel, persistent user-facing denial-of-service attack triggered by repeated browser launches, creating a self-sustaining loop that survives reboot and reset. We then outline realistic mitigation techniques for Android devices and argue for enforced transparency and stricter runtime controls on applet-initiated actions within the rapidly expanding eSIM ecosystem.

Keywords

eSIM, SIM Applets, Mobile Security, Android Security

1. Introduction

The embedded SIM (eSIM) is a programmable Subscriber Identity Module (SIM) promoted as a fully digital and modern replacement [1, 2] for the plastic SIM card. Profile installation is no longer tied to a physical token or to a retail distribution channel, but is instead performed remotely through software-assisted provisioning, typically initiated by scanning a QR code, and completed without requiring the user to handle any hardware component.

Buried inside this modern architecture, there is however something that dates back to the 1990s: eSIMs still run *Java Card Applets*. These are tiny programs originally designed for early SIM cards, long before smartphones, app stores, or remote provisioning even existed. The technology surrounding them has changed completely, but the applets have not, and can be embedded in an eSIM profile to perform various functions that can involve significant interactions with the mobile device's operating system.

While such capabilities are fundamental to enabling advanced telecom services and value-added features, and while the global eSIM market is boosting to conservative forecasts that estimate growth rates up to USD 17.5 billion by 2030 [3], eSIMs also introduce security risks when misused [4, 5, 6, 7, 8, 9]. In the past, this logic was tied to a physical card with known provenance; now it is invisible, delivered remotely in seconds, and never inspected by the user. More critically, the trust boundary has changed: applets are no longer confined to operator-issued SIMs subject to established compliance regimes, but can also be embedded by a growing array of third-party sellers that operate outside traditional telecom certification and auditing processes[10]. This moves the primary security assumption from physical control of the SIM to trust in a distributed and partially commercial provisioning supply chain.

Existing works such as *SIMjacker* [11] and *SIMurai* [12] have investigated applet abuse, primarily in contexts where attackers exploited vulnerabilities in pre-installed applets or could tamper with physical

Joint National Conference on Cybersecurity (ITASEC & SERICS 2026), February 09-13, 2026, Cagliari, IT
*Corresponding author.
lorenzo.valeriani@cnit.it (L. Valeriani); simone.festa@romamobilita.it (S. Festa); matteo.fanfarillo@cnit.it (M. Fanfarillo); giuseppe.bianchi@uniroma2.it (G. Bianchi)
http://netgroup.uniroma2.it/GiuseppeBianchi/biografia.html (G. Bianchi)
0009-0003-9373-9575 (L. Valeriani); 0009-0001-3319-9066 (M. Fanfarillo); 0000-0001-7277-7423 (G. Bianchi)

SIM cards. In particular, *SIMjacker* was a remote attack targeting the S@T Browser applet on physical SIMs. Remote eSIM provisioning fundamentally changes the threat model: applets can be deployed at scale via standard consumer activation flows, without any hardware interaction. Once embedded, these applets are non-removable, persist across reboots, and remain active as long as the profile is installed, complicating mitigation compared to physical SIMs, which can simply be removed during attacks such as denial-of-service. In this emerging eSIM context, where the carrier profile itself becomes the delivery vector, we aim to address the following three research questions.

RQ1: Are known applet injection techniques still feasible within modern eSIM provisioning workflows? Here the answer is positive: applets can be injected in *the same way as in ordinary SIMs*. The eSIM workflow does not introduce a new loading mechanism: an applet can simply be embedded inside the application elements of the profile before provisioning. Once the user installs the profile through standard activation flows, the embedded Universal Integrated Circuit Card (eUICC) loads and executes the embedded code with the permissions declared by the issuer. Thus, techniques long known from the physical SIM context apply directly and with minimal adaptation to eSIMs; the main difference is that they now operate within routine, consumer-facing eSIM remote activation workflows.

RQ2. What can applets do once they are installed? To answer this question, we developed two proof-of-concept attacks running on real devices and provisioned eSIMs. The first replicates the silent location exfiltration mechanism reported in earlier SIM Toolkit (STK) attacks [11]: an applet requests location parameters and exfiltrates them via SMS without any user prompt or visible interface. While the underlying logic mirrors what has been shown for physical SIMs, the impact is amplified in the eSIM setting, as the applet executes within a remotely provisioned profile that may originate from loosely regulated or poorly vetted third-party resellers. The second attack, to the best of our knowledge, has not been previously documented. In this case, an applet repeatedly triggers browser launches via STK, at rates as high as one per second, creating a persistent denial-of-service condition that severely degrades device usability and whose root cause is difficult for users to trace back to the eSIM profile.

RQ3. Can we mitigate, block, or render such attacks visible? Given that applets are persistent and widely distributable by multiple actors outside traditional operator channels, mitigation becomes more challenging than in scenarios involving removable SIM cards. In this work, we explore two classes of countermeasures. The first attempts to block specific applet-initiated actions at the operating system layer. We show that such blocking can work selectively, but it does not generalize to all STK-triggered behavior, and in particular fails to neutralize our first attack. The second mitigation direction focuses on visibility, i.e., making applet-initiated actions observable rather than silent. This approach applies to both attacks but introduces open questions, particularly the need for future usability evaluation: while alerts or prompts may reduce silent exploitation, they also transfer responsibility to the end user, who may not consistently attend to such notifications.

The remainder of the paper is organized as follows. Section 2 provides the minimal background on eSIM and applets necessary to follow the rest of the paper. Section 3 presents the threat model. Section 4 describes the two attacks. Section 5 details the experimental setup. Section 6 illustrates how the attacks operate and discusses potential defenses. Related work is reviewed in Section 7, and conclusions are drawn in Section 8. An additional Appendix provides a concise primer on how applets are programmed and deployed within eSIM profiles.

2. Background

2.1. Embedded SIM and eSIM profiles

An embedded SIM (eSIM) is a programmable, tamper-resistant secure element—referred to as an *embedded Universal Integrated Circuit Card* (*eUICC*)—integrated into the device and fully functional only after being configured with a provisioned profile. It performs the same fundamental functions as a traditional physical SIM card: authenticating the user and allowing access to mobile networks, but without requiring a removable component. This design offers several practical advantages [1].

For example, eSIMs enable users to remotely download and manage carrier profiles, significantly reducing roaming costs by allowing connection to local networks while traveling. They also support the coexistence of multiple profiles on a single device, facilitating seamless switching between carriers or phone numbers. From a hardware perspective, the elimination of the physical SIM tray allows manufacturers to optimize internal space for other components, such as larger batteries or additional sensors. Furthermore, because the eSIM is soldered to the device, it is less susceptible to loss or physical tampering [13].

A core abstraction in the eSIM architecture is the *profile*, the software-based successor to the physical SIM card. Rather than existing as a removable object, the profile is remotely provisioned onto the eUICC and stored as a structured set of Profile Elements, each encoded in ASN.1 and processed independently by the eUICC. The profile includes identifiers (IMSI), authentication material (keys, etc), network access applications, operator policies, and optional executable logic such as applets, see next Section 2.3. Its format follows Tag-Length-Value encoding with strict ordering rules, ensuring deterministic installation. Once activated, the profile defines how the device authenticates to the network, how data are managed, and which privileged STK operations an applet may trigger. Unlike physical SIMs, profiles can be downloaded, enabled, disabled, or deleted on demand, making them both flexible and opaque to end users, as there is no user interface for applet inspection.

2.2. The eSIM Ecosystem

The transition from removable SIMs to eSIMs is not a mere technological refinement, but constitutes a structural shift in the mobile-services marketplace, fundamentally reshaping established business models [14]. By decoupling hardware from operator-specific configuration, eSIM technology expands the ecosystem to include a wider and more heterogeneous array of stakeholders, ranging from traditional operators to platform providers, device manufacturers, intermediaries, and third-party resellers operating outside of conventional certification or audit regimes [15, 16].

This enlarged actor landscape introduces new operational dependencies, redistributes trust relationships, and significantly increases systemic complexity. As several studies have observed [5, 17, 18], the resulting environment is not only more intricate but also substantially less bounded: actors may vary widely in technical capability, regulatory oversight, commercial incentives, and security posture. Such heterogeneity gives rise to a more dynamic and, at times, unpredictable ecosystem, where responsibilities once concentrated within mobile network operators now span multiple loosely coupled entities. The key stakeholders and components include:

- **Operator**: A *Mobile Network Operator* (*MNO*) or *Mobile Virtual Network Operator* (*MVNO*) that provides cellular service and issues eSIM profiles to users [19].
- **eUICC (Embedded Universal Integrated Circuit Card)**: The tamper-resistant secure element that stores and executes eSIM profiles. Certified under international standards, it supports multiple profiles, though only one can be active at a time. It enforces security policies and protects sensitive subscriber data.
- **SM-DP+ (Subscription Manager - Data Preparation Plus)**: This entity is responsible for securely preparing, encrypting, and delivering eSIM profiles to devices. It guarantees cryptographic protection throughout the provisioning process and operates within the trust domain of the network provider, which maintains full control over subscription data [17]. Rather than managing the SM-DP+ infrastructure in-house, many mobile operators delegate its deployment and operation to certified third-party providers. In practice, an operator may rely on multiple SM-DP+ providers, while a single SM-DP+ provider may simultaneously serve multiple operators.
- **LPA (Local Profile Assistant)**: A client-side software component that manages profile downloads, installation, and deletion. It may reside on the device (LPAd) or on the eUICC (LPAe). It also handles QR code scanning and communicates with SM-DP+ during profile provisioning.

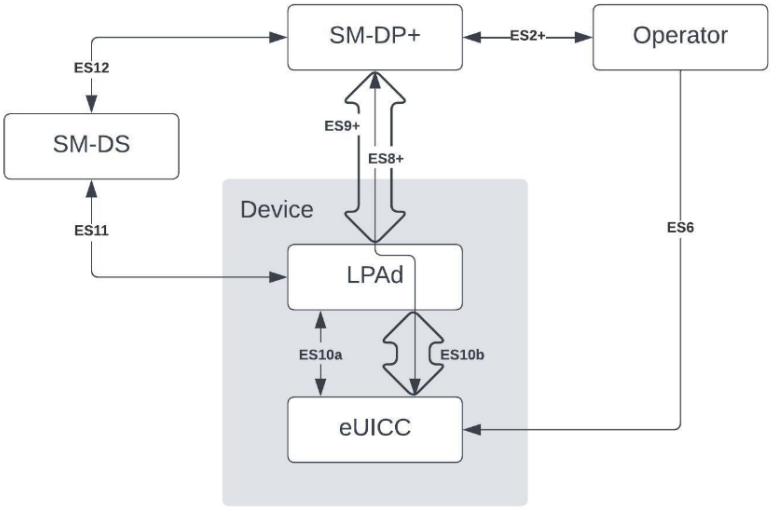

Figure 1: Overview of the eSIM ecosystem and its key components, together with the ESx interfaces defining standardized communication between them.

- **SM-DS (Subscription Manager - Discovery Server)**: Acts as a routing intermediary that helps SM-DP+ reach devices even when their IP addresses are unknown or change frequently. The LPA polls the SM-DS to check for notifications depending on the eUICC status and user actions.

Within this evolving architectural and organizational context, ensuring the integrity of remotely provisioned platforms becomes a critical challenge [20, 21]. As illustrated in Figure 1 and detailed in the following paragraphs, the principal components of the eSIM ecosystem interact through interdependent operational and trust relationships that collectively govern the lifecycle of eSIM profiles and maintain secure communication between devices and mobile operators. These interactions are governed by the GSMA Remote SIM Provisioning (RSP) protocol suite, which defines the roles, message flows, and trust relationships involved in secure eSIM provisioning and relies on a public key infrastructure to authenticate and authorize participating components [22]. However, RSP focuses on secure delivery and authentication, not on validating the internal logic of profiles, creating an attack surface for actors able to manipulate profile contents within the provisioning chain.

2.3. Java Card Applets

Much like traditional SIM cards, a crucial, yet often underexamined, aspect of eSIM technology is its support for **Java Card Applets**. Applets constitute a core programmable component of SIM and eSIM architectures, enabling the deployment of application logic directly within the secure execution environment of the UICC/eUICC. As highlighted in Figure 2, they appear as a distinct building block of the eSIM profile alongside security domains, network access applications, and file system elements. Their formal integration is defined through the `PE-Application` Profile Element, which specifies the load package and possible runtime instances of the applet, enabling both code deployment and customization at installation time.

Applets are typically written in *Java Card*, a constrained dialect of Java optimized for secure EEPROM-backed environments, with strict memory quotas and no support for strings, floating-point types, or multithreading. Their binary representation is transported in a `.cap` package, parsed into components such as `Header`, `Directory`, `Applet`, `ConstantPool`, and `Method`, which are embedded in the eSIM profile through the `loadBlockObject` field of the `PE-Application` element.

Applets power the *SIM Toolkit (STK)* layer and allow operators to trigger actions such as SMS dispatch or browser launch. Although originally conceived for operator-controlled service delivery, the same mechanism has been abused in documented attacks (e.g., SIMjacker [11]), where remotely triggered applet logic silently retrieves device and network identifiers via STK commands.

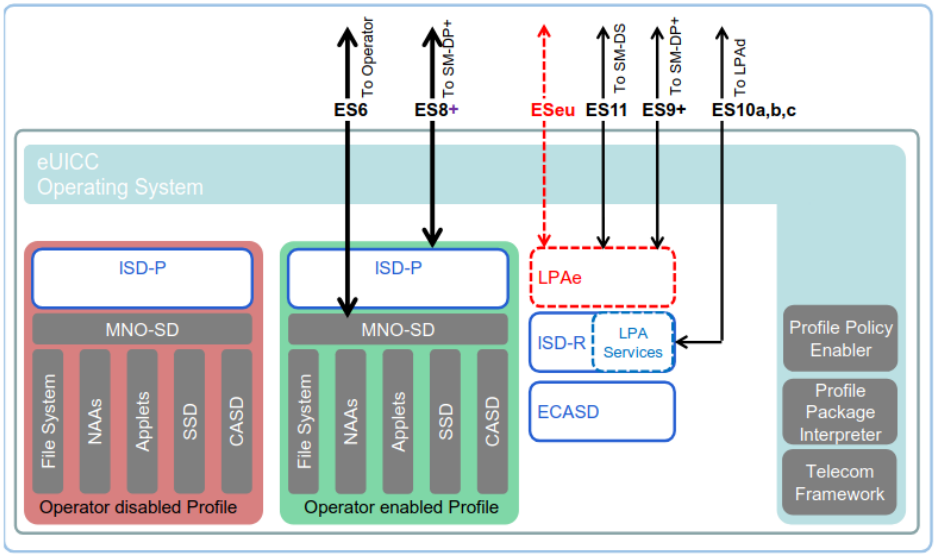

Figure 2: Schematic Representation of the eUICC.

In the eSIM context, the programmability of applets extends from a hardware-bound model to a remotely provisioned software lifecycle: applets can be inserted into profiles prior to download, bypassing historical requirements for physical card access. As demonstrated later on, malicious applets can exfiltrate location information or execute timed actions without user visibility, while current mitigation approaches tend to rely on assumptions of provisioning-channel trust rather than enforce runtime inspection or disclosure of applet-initiated behavior.

3. Threat Model

This section defines the threat model for malicious eSIM applets, highlighting how the distribution of provisioning responsibilities across multiple actors weakens traditional trust assumptions. The threats considered here *do not* rely on vulnerabilities in the mobile operating system, baseband firmware, or eUICC hardware, but instead abuse *legitimate provisioning privileges* and standardized SIM Toolkit functionality. Compared to the traditional SIM model, the eSIM provisioning process involves a broader set of entities operating under heterogeneous operational, regulatory, and trust assumptions. By virtue of their role in the architecture, each of these parties can potentially[1] introduce components that execute within the secure element.

Mobile Service Providers. Mobile Service Providers retain the authority to embed applets within the profiles they issue. Although typically used for service provisioning or device management, these applets could be abused to implement covert communication channels or to exfiltrate subscriber-related information to external entities.

Subscription Manager Data Preparation Plus (SM-DP+). SM-DP+ is responsible for the creation and delivery of eSIM profiles to target eUICCs. As part of this role, it has full visibility into the internal structure of the profile and the ability to modify its contents prior to distribution. A compromised or malicious SM-DP+ can therefore inject arbitrary applet code while still producing a valid profile, making such modifications indistinguishable from legitimate provisioning activity from the perspective of the eUICC.

[1]These scenarios describe *potential attack capabilities* rather than expected operational behavior; in other words, while the entities considered may technically be able to tamper with eSIM profiles, we obviously do **not** imply that they may be doing this in current practice.

Third-party Resellers and Intermediaries. The eSIM ecosystem also includes third-party resellers or intermediaries that distribute or manage profiles on behalf of service providers. Some of these actors may legitimately insert specific access-control components within the profiles they manage, such as Access Rule Files (ARF) or Access Rule Applications (ARA), in order to enable specific privileges (e.g., Android carrier privileges [23]) for their associated LPA. Although not all resellers have access to profile content, a malicious or compromised reseller with such capabilities can serve as a vector for introducing harmful applets that remain fully compliant with the provisioning specifications.

eUICC Manufacturers. Threats are not limited to provisioning entities alone; eUICC manufacturers occupy a uniquely privileged position, as they control the firmware and the Java Card runtime. Vulnerabilities or intentional backdoors at this level could enable malicious behaviors that are functionally indistinguishable from legitimate secure-element operations. Such attacks would be particularly difficult to detect, attribute, or mitigate, given the closed and security-critical nature of eUICC implementations.

From a practical point of view, the likelihood that these stakeholders become a threat varies significantly across the ecosystem. Established mobile network operators and certified eUICC manufacturers typically operate under mature regulatory, contractual and auditing frameworks, making intentional abuse unlikely and more plausibly limited to misconfigurations or insider threats. In contrast, the rapid expansion of third-party intermediaries and resellers, often operating across jurisdictions and outside traditional telecom certification regimes [10], introduces actors with more heterogeneous incentives and security maturity. Although this does not imply malicious intent, it increases the probability that legitimate provisioning capabilities may be misused, repurposed, or inadequately controlled, thus expanding the effective attack surface of the eSIM ecosystem.

Finally, we emphasize that the ability to install malicious applets does not eliminate the relevance of attacks based on remotely triggering vulnerabilities in preinstalled applets, eUICC firmware, or OTA SMS handling logic. Such vulnerabilities may themselves be intentionally or unintentionally introduced at different stages of the eSIM supply chain and later exploited through remote triggering mechanisms. Attacks that exploit implementation flaws in existing components to achieve remote code execution or unauthorized profile installation remain applicable in the eSIM context.

4. Attacks

The SIM Toolkit (STK) defines a set of powerful commands (full list documented extensively by ETSI in [24]) through which a SIM or eSIM can request actions from the device, including sending SMS messages, retrieving location parameters, launching applications, opening browser sessions, or interacting with user-interface elements. Originally introduced in the late 1990s, the STK predates modern mobile operating systems and their permission models; as a consequence, it exposes a powerful legacy control surface that operates below awareness of the device's security architecture. When invoked by an authenticated applet, STK commands can therefore trigger device-level actions with minimal user visibility or without user interaction at all.

The writing of applets that interact with the device via the STK is straightforward, and the eSIM architecture does not fundamentally change this model. Once an applet is compiled, embedding it into a profile, together with the permissions required to issue STK commands, is mechanically simple, and such a profile can be provisioned remotely like any other. The technical steps for creating and integrating an applet are outlined in the Appendix. Naturally, an applet must still be granted the appropriate privileges by the profile issuer, a point we discuss later when analyzing the feasibility and limitations of the attacks.

The potential attack vectors include the malicious use of operations that applets are legitimately allowed to perform (e.g., proactive commands) and the exploitation of security bugs or misconfigurations within the eUICC software stack. Recent efforts [25] have demonstrated that hostile applets can introduce significantly more severe security risks within the eSIM ecosystem. In particular, the researchers identified a type-confusion vulnerability in the Java Card environment of commercial eUICCs. This

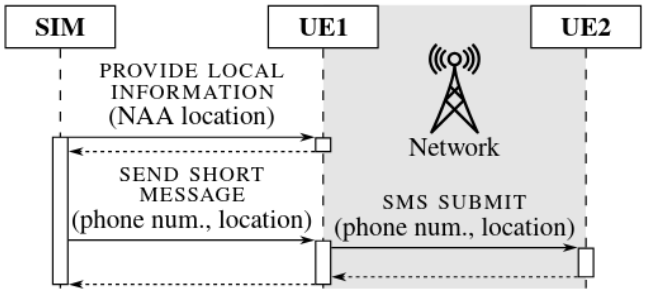

Figure 3: Concept of the Spy-SMS-Ware attack.

vulnerability enables an attacker to extract the eUICC's private key. The implications of such an attack are significant, particularly when considering the exfiltration capabilities that are already present in standard applets. Under these circumstances, a malicious profile has the potential to leak the confidential material of all other profiles installed on the card. By compromising the private key of the eUICC, a malicious actor could mount higher-level attacks within the Remote SIM Provisioning (RSP) infrastructure, with severe consequences for the entire ecosystem.

In the following subsections, we illustrate how these long-standing mechanisms can be misused through two concrete examples. The first shows how an applet can exfiltrate device-specific information via SMS by leveraging location-related STK primitives, while the second demonstrates how an applet can induce a persistent and self-sustaining denial-of-service condition by repeatedly triggering browser-launch operations. Together, these examples highlight how legacy STK capabilities, combined with modern eSIM provisioning channels, enable attack vectors that are both practical and scalable.

4.1. Spy-SMS-Ware: Silent Location Exfiltration via STK

This attack, brought to public attention by the Simjacker attack [11] and here ported to eSIMs, demonstrates how an applet embedded within an eSIM profile can silently extract device-specific information and leak it through SMS.

Attack overview. The objective is to collect coarse-grained location and network parameters from the device and exfiltrate them to an attacker-controlled endpoint. The attack relies on the following two STK commands:

1. `PROVIDE LOCAL INFORMATION`: it allows to query device and network parameters such as MCC, MNC, LAC/TAC, and Cell ID;
2. `SEND SHORT MESSAGE`: as the name implies, it enables an applet to transmit SMS messages directly over the mobile network.

By combining these long-standing capabilities, the applet can autonomously collect location-related data and exfiltrate it via SMS with minimal user visibility. This interaction occurs entirely within the secure element, and the resulting SMS traffic may be indistinguishable from operator-triggered communication, making detection challenging. A conceptual overview of this attack is shown in Figure 3.

Execution flow. After the profile is installed and the applet is instantiated, the attack proceeds autonomously:

1. The applet issues a `PROVIDE LOCAL INFORMATION` request; the device supplies the relevant network-identification parameters, which the applet records internally.
2. The applet constructs an SMS embedding these values, potentially using operator-like formatting or binary encodings to minimize visibility.

3. The applet issues a SEND SHORT MESSAGE command to an attacker-controlled destination. The transmission occurs without user interaction, and without the messages appearing in the device's messaging application [26].

The sequence can be triggered periodically, opportunistically, or upon specific events, depending on the applet's logic. In particular, the applet can register for EVENT DOWNLOAD LOCATION STATUS events to be notified by the terminal of a change in the device's network location status.

Impact and observations. The information obtainable through PROVIDE LOCAL INFORMATION is typically sufficient to infer the approximate location of the user, monitor movements, or identify the serving network. When combined with the ability to exfiltrate data silently, the attack provides a practical surveillance mechanism that is both simple to implement and difficult to detect. Unlike its prior implementation over physical SIM cards [11], our eSIM-based Spy-SMS-Ware can scale to any number of devices installing a compromised eSIM profile. The user is not required to grant permissions, and the applet's execution remains confined to the secure element, outside the visibility of conventional security tooling. Finally, we remark that the SMS-based exfiltration attack could, in principle, be extended by exploiting additional STK commands [24]. For instance, instead of relying solely on SEND SHORT MESSAGE, an applet could use OPEN CHANNEL and SEND DATA to achieve higher-bandwidth data leakage over a covert data connection, further broadening the impact of the attack.

4.2. BrowserLoop Attack: Persistent Applet-Controlled Browser Invocation

This second attack, which we named BrowserLoop, is (to the best of our knowledge) a novel and previously unexplored vector. It demonstrates how a malicious applet can induce persistent usability degradation and open multiple attack surfaces by repeatedly triggering browser-launch operations through the SIM Toolkit (STK). Unlike conventional device-level attacks, this technique relies on a loop entirely within the secure element that survives across reboots as long as the corresponding eSIM profile remains active.

Attack overview. BrowserLoop exploits two STK commands: LAUNCH BROWSER, which instructs the device to open a specific URL, and TIMER MANAGEMENT, which schedules callbacks. By combining these, the applet creates a self-sustaining loop. The applet can explicitly define its re-execution interval, with a configurable timer resolution that can be set as low as $\Delta t = 1$ s. In practice, however, the effective execution frequency is constrained by the device's processing overhead. As a result, the attack manifests as periodic browser launches that repeatedly steal focus from the user's foreground application.

Execution flow. After installation, the malicious applet executes the following cycle:

1. It issues a TIMER MANAGEMENT command to start an internal timer.
2. Upon expiration, the applet receives a TIMER EXPIRATION event and resumes execution.
3. The applet responds by issuing a LAUNCH BROWSER command targeting a specific URL.
4. Crucially, the applet restarts the timer before yielding control, ensuring the cycle continues indefinitely.

Impact and observations. The processing latency allows for a brief window of interactivity, preventing a complete device freeze. However, the attack introduces significant security risks beyond simple usability degradation. We identify four primary threat vectors enabled by similar mechanisms:

- **Limited Denial-of-Service (DoS):** The constant "focus stealing" caused by the browser launching renders the device frustrating to use, disrupting legitimate applications and draining system resources.
- **Phishing:** The applet can direct the browser to a deceptive landing page (e.g., a fake carrier login or system update page), socially engineering the user into surrendering credentials immediately upon unlocking the device.

- **Malware Distribution:** The target URL can host malicious payloads, initiating drive-by downloads or exploiting browser vulnerabilities without explicit user navigation.
- **Data Exfiltration:** Sensitive data harvested by the applet (such as location or IMEI via `PROVIDE LOCAL INFORMATION`) can be appended to the URL as HTTP GET parameters (e.g., `http://attacker.com/exfiltrate?loc=XYZ`), effectively exfiltrating data through the browser request itself.

Unlike traditional malware, this behavior cannot be stopped by a factory reset, as the malicious logic is anchored in the eSIM profile. Mitigation requires the user to successfully navigate the settings menu within the short inter-arrival windows to disable or delete the offending profile. Importantly, the attack is difficult to attribute from the user's perspective. Repeated browser launches are unlikely to be intuitively associated with a compromised eSIM profile. This lack of clear attribution significantly complicates diagnosis and remediation, increasing the likelihood that the attack persists over time despite user intervention attempts.

5. Experimental Setup

To evaluate the feasibility and impact of applet-level attacks in the eSIM ecosystem, we constructed a fully controlled experimental environment that reproduces the complete Remote SIM Provisioning (RSP) pipeline, from profile creation to on-device execution. The setup consists of three main steps: (i) obtaining the capability to generate and sign valid eSIM profiles, (ii) embedding custom Java Card applets into those profiles, and (iii) delivering the profiles to real devices through a functional RSP infrastructure and a test mobile network.

5.1. Profile provider setup and provisioning credentials.

The first requirement was the ability to generate eSIM profiles that would be accepted by a real eUICC. In commercial deployments, this capability is restricted to authorized profile providers operating within the GSMA RSP ecosystem. To replicate this role in a laboratory setting, we relied on programmable consumer eUICCs and GSMA test credentials.

Specifically, we used a consumer eUICC compatible with the GSMA SGP.26 [27] specification and provisioned with the GSMA test root certificates. This configuration allows the eUICC to authenticate Subscription Manager entities operating in a test environment. On the server side, we employed an open-source SM-DP+ server[2] based on the Osmocom toolchain (pySIM), which provides profile provisioning capabilities suitable for research and testing purposes. We then generated a TLS certificate for our SM-DP+ server instance, signed by the GSMA test Certificate Authority as defined in the SGP.26 test specifications. This certificate is required to establish a mutually authenticated TLS channel between the Local Profile Assistant on the device and the SM-DP+ during profile download.

In addition to the provisioning infrastructure, profile creation capabilities were acquired by modifying SGP.26 test profiles in their DER-encoded (Distinguished Encoding Rules) format. This step was necessary for two reasons. First, it allowed us to customize subscriber-specific parameters, such as the IMSI (International Mobile Subscriber Identity) and AKA (Authentication and Key Agreement) credentials, in order to connect the provisioned profiles to a dedicated test network. Second, we modified the ICCID (Integrated Circuit Card Identifier) values to enable the concurrent installation on a single eUICC of multiple profiles derived from the same baseline test profile.

Overall, with this setup, we achieved the same technical capabilities as a legitimate profile provider, i.e., the ability to prepare and deliver cryptographically valid eSIM profiles that are accepted by the target eUICC.

[2] https://github.com/osmocom/pysim/blob/master/osmo-smdpp.py

5.2. Embedding applets into eSIM profiles.

Once profile creation capabilities were in place, the next step was to embed malicious Java Card applets into the eSIM profiles. The general process of developing, compiling, and packaging applets is described in detail in the Appendix; here we focus on how those applets were practically integrated into the profiles used in our experiments.

Each applet was first compiled into a Converted Applet (CAP) file, the standard binary format used to deploy Java Card applets onto SIMs/eSIMs, details in the Appendix. The CAP components were then extracted and reordered according to the Java Card specifications to construct a valid load block. This load block was embedded into the profile as part of a `PE-Application` Profile Element, using the `loadBlockObject` field to carry the applet's binary code. The same Profile Element also declares one or more `ApplicationInstance` entries, which instantiate the applet at installation time.

Crucially, the Toolkit Application Specific Parameters associated with each instance were configured to grant access to the required SIM Toolkit (STK) commands, such as timers, SMS submission, and browser launch. From the perspective of the eUICC, the resulting profile is indistinguishable from a legitimate operator-issued profile containing service-related applets: the applet code is loaded, installed, and executed entirely within the secure element, with privileges determined solely by the profile contents. No modification of the eUICC firmware or of the mobile operating system was required.

5.3. Profile delivery and network environment.

The final step consisted in delivering the crafted profiles to real devices and ensuring full network connectivity to observe the effects of the attacks.

Profile delivery via SM-DP+ and LPA. For profile delivery, the SM-DP+ server is responsible for hosting and serving the encoded eSIM profiles. The server exposes the standard RSP interfaces and, in our laboratory setting, uses the GSMA test certificates to authenticate itself to the eUICC during the provisioning process. On the device side, profile installation is initiated through a Local Profile Assistant (LPA), which establishes a secure channel with the SM-DP+ and downloads the profile using standard consumer eSIM activation flows (e.g., QR-code–based provisioning). In our experiments, we employed *EasyEUICC*[3], an open-source implementation of the LPA that can run as a regular user-space application. Unlike the privileged, OEM-integrated LPAs normally present on consumer smartphones, EasyEUICC is intentionally designed to enable experimentation and profile management without requiring system-level privileges.

Programmable eUICC hardware. To support our experiments, we used the *sysmoEUICC1-C2x*, a programmable eUICC platform that supports the download, installation, and management of multiple eSIM profiles and is compatible with GSMA SGP.26 test certificates. Unlike a traditional SIM, which typically exposes a single fixed profile, this eUICC supports a full profile lifecycle (download, enable/disable, delete), making it suitable for repeated experiments with multiple profiles and applet configurations. While typical consumer eUICCs are soldered onto the device's mainboard, the sysmoEUICC1-C2x adopts a removable SIM-like form factor (2FF/3FF/4FF). This design makes it compatible with smart-card readers and devices that lack native eSIM support, and facilitates rapid iteration over profile contents and provisioning flows during development and testing.

Enabling unprivileged access to the eUICC. On Android, an unprivileged LPA requires carrier privileges to interact with the eUICC. These privileges are granted via Access Rule Applications (ARA) or via Access Rule Files (ARF), either at card level or embedded within individual profiles. On sysmoEUICC1, the card-level ARA-M includes EasyEUICC's certificate hash, authorizing the app for eUICC-wide profile management operations across all hosted profiles. This configuration is primarily

[3] https://gitea.angry.im/PeterCxy/OpenEUICC

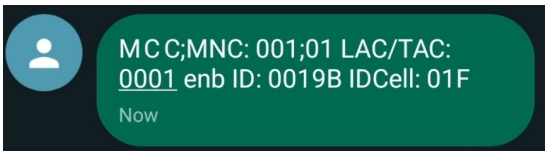

Figure 4: Spy-SMS-Ware implementation: SMS message sent from the victim to the attacker. Screenshot taken from attacker's phone.

intended for development and research eUICCs and enables controlled experimentation in environments where privileged access to the device vendor's LPA is not available.

Test mobile network (RAN + core). To support network-dependent functionality, such as SMS transmission and cellular registration, we deployed a complete test mobile network. The Radio Access Network (RAN) was implemented using a software-defined radio device (Ettus B210) running the open-source *srsRAN* software suite (specifically, the LTE eNodeB component). The RAN was connected to an LTE core network implemented using *Open5GS*, which provides the standard EPC functions for authentication, mobility management, and service enablement. SMS capabilities were enabled via SMS over SGs using *OsmoMSC* and *OsmoHLR*. Subscriber credentials were synchronized across the eSIM profile, Open5GS HSS, and OsmoHLR. Service identifiers required for SMS routing (e.g., MSISDN) were configured on the core-network side. As user equipment, we employed a set of commercial smartphones.

6. Evaluation and Defense

This section evaluates the practical impact of the two applet-based attacks introduced earlier and discusses potential defensive measures. The attacks were experimentally validated on a heterogeneous set of commercial devices from different vendors and running various operating systems, including both Android- and iOS-based platforms. For a complete list of tested devices and experimental outcomes, see Table 1. Defensive measures were implemented and tested on a Pixel 6 device, whose modified operating system facilitated system-level interventions and rapid experimentation.

6.1. SMS-Based Applet-Initiated Communication

As established by prior work on physical SIM cards [11], applets can issue SIM Toolkit commands to retrieve coarse-grained location and network information and transmit it via SMS without user interaction. Our experiments confirm that this behavior remains fully functional when the applet is embedded within a remotely provisioned eSIM profile. Figure 4 reports a sample SMS generated by our proof-of-concept implementation. The message content is intentionally formatted in a human-readable form to facilitate validation and demonstrate feasibility, and includes network identifiers sufficient to infer the user's approximate location. The transmission does not require any runtime permission or explicit user awareness, and the resulting SMS is not recorded in the device's messaging history. Finally, the applet registers for network location update events and automatically transmits a new SMS each time a change in the reported location status occurs. In our experiments, all tested devices transmitted these SMS messages without presenting any user-visible notification.

6.2. Applet-Controlled Browser Launches

We then evaluated BrowserLoop, a novel denial-of-service attack based on repeated browser launches triggered by an applet. Unlike the SMS attack, BrowserLoop directly impacts device usability.

The attack was assessed on our test set of commercial smartphones. Our experiments revealed three distinct classes of device behavior. The first group of Android devices executed the browser-launch command without user interaction, resulting in repeated openings at a reduced but still intrusive

Table 1
Impact of applet-based attacks across tested devices

Device	OS Version	Spy-SMS-Ware Outcome	BrowserLoop Outcome
Google Pixel 9 Pro	Android 16 QPR2	SMS sent silently	Browser opened
Google Pixel 6	LineageOS 21 Android 14	SMS sent silently	Browser opened
Samsung Galaxy S23	One UI 6.1 Android 14	SMS sent silently	Browser opened
OnePlus Nord 5G	OxygenOS 12.1 Android 12	SMS sent silently	User confirmation requested
Apple iPhone 13	iOS 26.1	SMS sent silently	Requests ignored
Apple iPhone 16 Pro	iOS 18.6.2	SMS sent silently	Requests ignored

rate of approximately one every 3–4 seconds. This effective rate is substantially lower than the timer resolution requested by the applet and appears constrained by the processing pipeline, including baseband handling, the Android telephony framework, and additional mechanisms in Android's inter-process communication (IPC). The second class, represented by the OnePlus Nord 5G, mitigated the attack by prompting the user for confirmation before opening the browser, thereby interrupting the automated loop but still exposing the user to persistent and intrusive prompts. Finally, iOS devices were observed to systematically ignore browser launch requests, effectively preventing the attack.

Despite these platform-specific variations, on affected Android devices, the repeated focus stealing caused by browser launches and confirmation dialogs disrupts normal operation and complicates profile removal. The attack further persists across reboots and factory resets, as the triggering logic is entirely contained within the eSIM profile.

6.3. Blocking Applet-Initiated Attacks

We further examined whether applet-initiated actions can be actively *blocked* on Android devices. SIM Toolkit proactive commands are handled by a combination of framework-level services and user-facing components, most notably `CatService` (radio interface parsing) and `StkAppService` (user interface handling).

The framework-level mitigations were initially implemented directly in the Android codebase, requiring full system recompilation. The same logic was subsequently adapted into an LSPosed framework module, enabling rapid deployment without recompiling or reflashing the OS, and allowing the mitigation to be generalized across multiple Android devices, provided they are rooted and have LSPosed installed.

To suppress applet-triggered behavior, we operated at the Android framework level. In particular, we focused on the `LAUNCH_BROWSER` proactive command, which is responsible for triggering browser launches in the BrowserLoop attack scenario. We instrumented the command-processing logic in `CatService`, intercepting proactive commands of type `LAUNCH_BROWSER` before they are forwarded to `StkAppService`. By selectively discarding these commands in the `handleCommand()` path, browser launches initiated by the applet are effectively suppressed.

The modification was applied without requiring changes to the eSIM profile or the applet logic. This approach allows selective blocking of user-facing STK actions while preserving the remainder of the SIM Toolkit functionality. In practice, this technique successfully neutralizes the BrowserLoop behavior by preventing repeated browser invocations.

However, this approach does not generalize to all applet-initiated actions. Notably, `SEND_SMS` proactive commands *are processed below the Android framework and handled directly by the modem*[4], bypassing

[4]See `config_stk_sms_send_support` in `frameworks/base/core/res/res/values/config.xml`

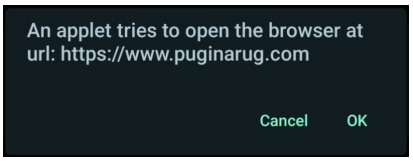

Figure 5: Browser-launch warning example.

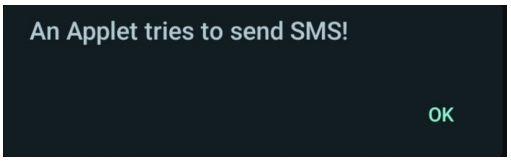

Figure 6: SMS delivery warning example.

`CatService` and evading framework-level blocking.

6.4. User Awareness and Visibility-Based Mitigations

The analysis above shows that preventive blocking at the Android framework level is inherently command-dependent: while certain user-facing STK actions such as `LAUNCH_BROWSER` can be selectively suppressed, modem-handled commands such as `SEND_SMS` cannot be reliably blocked at the same layer. To address this limitation, we explore a complementary mitigation strategy based on *visibility* rather than suppression.

For browser-related actions, visibility was introduced by modifying the SIM Toolkit user interaction path rather than discarding commands. Instead of suppressing the `LAUNCH_BROWSER` proactive command, we instrumented `StkAppService` to enforce an explicit user-facing confirmation dialog before execution. This dialog exposes the target URL and clearly attributes the action to SIM Toolkit logic, ensuring that browser launches initiated by an applet are no longer silent. Figure 5 shows an example of the resulting browser warning dialog.

Similarly, for SMS-related actions, we implemented a visibility mechanism that raises a user-facing alert for every `SEND_SMS` proactive command issued by the SIM. Rather than attempting to block SMS transmission, which is handled below the framework level, we modified the STK handling path to restore user awareness of otherwise silent SIM-initiated communication. This mechanism is effective when the modem reports the proactive command to the upper layers[5], as observed on the Pixel 6. Figure 6 shows an example of such an STK SMS alert. This approach preserves network functionality while enabling user oversight.

Unlike preventive blocking, visibility-based mitigation applies to both classes of behavior demonstrated in this work. However, it shifts responsibility to the user, who must interpret and respond to the presented warnings. While this improves transparency and accountability, its effectiveness ultimately depends on user attention and interface design.

Applet-Initiated SMS Signaling in Real-World Deployments. Since applet-initiated SMS transmission is straightforward for profile providers, we were curious to see whether similar behavior appears in real-world deployments. We conducted a small number of exploratory tests on operator-issued SIM and eSIM profiles to observe whether SMS messages are automatically transmitted without explicit user interaction, a behavior previously reported in practice [28]. We modified the Card Application Toolkit (CAT) logic within the Android framework. These modifications revealed not only the issuance of `SEND_SMS` commands, but also critical details including destination number, message body, and complete Protocol Data Unit (PDU). In these tests, we identified at least one nationally deployed operator profile that sends an SMS upon first insertion into a mobile device.

Figure 7 shows a representative log from such a modified deployment, where a pre-installed applet triggers the transmission of an SMS as part of the initial activation process. While the destination address is visible, the message payload itself is not human-readable, likely due to the use of binary encoding or non-printable characters. We do not attribute malicious intent to this behavior; on the contrary, given its association with a major operator, it is reasonable to assume a legitimate operational purpose, such as

[5]On such devices, STK-triggered SMS events are observable without root via Android radio logs, e.g.,

```
adb logcat -b radio | grep SEND_SMS
```

This approach requires no system changes but does require continuous monitoring, since log entries are transient.

Figure 7: Log showing applet-initiated SMS transmission upon SIM insertion by a national operator. Blurred to safeguard operator and device anonymity.

service activation, provisioning confirmation, billing procedures, or network diagnostics. Nevertheless, this observation highlights an important property of the current ecosystem: such communications occur without user awareness and are indistinguishable, at the device level, from potentially abusive applet-triggered transmissions. This lack of transparency complicates both user understanding and technical detection of malicious behavior.

7. Related works

This work is motivated by (and builds upon) prior investigations into SIM and eSIM applet security. The Simjacker attack [11] demonstrated that SIM Toolkit commands can be abused to silently extract device and network data from physical SIMs at scale. More recently, Security Explorations [25] uncovered severe Java Card runtime vulnerabilities in commercial eUICCs, enabling private key leakage and compromising the secure element's root of trust.

Complementing these, Motallebighomi et al. [10] examined the privacy implications of the modern eSIM ecosystem, emphasizing supply chain fragmentation among operators, SM-DP+ providers, and resellers. While they observed that some commercial eSIM profiles make use of SIM Toolkit proactive commands, their study did not focus on the threat model of malicious applets.

Some prior works addressed security aspects of the RSP protocol by explicitly introducing the notion of stakeholders. In particular, the 2023 ENISA report [2] emphasized the risks associated with *Sensitive Processes* [29], such as platform management, profile lifecycle management, and key provisioning. Similarly, the Security Analysis of the Consumer Remote SIM Provisioning Protocol [30] defined 15 security goals, investigated two protocol variants, and evaluated 11 scenarios, each assuming the compromise of a single stakeholder.

Wen et al. [31] proposed defenses against SMS-based attacks by operating at the radio interface layer, enabling interception of SMS traffic that bypasses the Android framework. While their approach offered strong prevention guarantees, it did not consider the eSIM provisioning model.

Faisal and Marilly [32] presented an architecture for monitoring the integrity of remote industrial devices in multi-stakeholder environments. The proposed solution leveraged secure applets deployed in the eUICC and a distributed ledger to verify and manage device integrity without requiring additional hardware, such as Trusted Platform Modules (TPMs).

8. Conclusion

While programmable applets have long existed in physical SIM cards, their deployment within remotely provisioned eUICCs fundamentally alters the security landscape. The expanded eSIM ecosystem introduces new provisioning actors that are not subject to the same scrutiny as traditional operators, transforming applet misuse from isolated incidents into systemic risks driven by ecosystem fragmentation.

Our attacks demonstrate that long-standing SIM Toolkit capabilities can be repurposed for silent data exfiltration and persistent device disruption, and that such logic can be deployed at scale through ordinary consumer activation workflows without hardware access or user awareness. Crucially, our

attack examples only scratch the surface: the STK exposes a much broader range of commands, including telephony primitives such as SET UP CALL, event-driven monitoring via SET UP EVENT LIST, UI-manipulation commands such as DISPLAY TEXT and GET INPUT, data-channel management through OPEN CHANNEL and SEND/RECEIVE DATA, and even modem-level interactions enabled by RUN AT COMMAND. When accessible to unregulated or lightly supervised profile issuers, these features constitute a significant potential misuse vector.

As eSIM adoption accelerates, securing the provisioning pipeline and constraining third-party access to SIM Toolkit capabilities are essential, as the threats analyzed here represent only the beginning of a broader and urgently needed reassessment of programmable eSIM interfaces.

Acknowledgements

This work was partially supported by the EU under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnerships "SEcurity and RIghts In the CyberSpace" (PE00000014 - program "SERICS") and "Telecommunications of the Future" (PE00000001 - program "RESTART"). The first author was partially funded by the project I-Nest (G.A. 101083398 - CUP B97H22004950001) - Italian National hub Enabling and enhancing networked applications and Services for digitally Transforming Small, Medium Enterprises and Public Administration. The third author was partially supported by *Agenzia per la cybersicurezza nazionale* under the programme for promotion of XL cycle PhD research in cybersecurity –- CUP number: E83C24002610001. The views expressed are those of the authors and do not represent the funding institutions.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] Saily, What is an eSIM card?, 2025. https://saily.com/what-is-esim/.

[2] E. U. A. for Cybersecurity (ENISA), Embedded SIM (eSIM) Ecosystem: Security Risks and Measures, Technical Report, European Union Agency for Cybersecurity (ENISA), 2021. URL: https://www.enisa.europa.eu/sites/default/files/publications/Embedded%20Sim%20Ecosystem%20Security%20Risks%20and%20Measures.pdf, accessed: 2025-04-17.

[3] F. business insights, eSIM Market size, 2025. https://www.fortunebusinessinsights.com/industry-reports/embedded-sim-esim-technology-market-100372.

[4] J. Zhao, Enhancing System Resiliency for 5G and 5G IoT: A Plug-and-Play, SIM/eSIM-based Approach, University of California, Los Angeles, 2023.

[5] T. J. Gerpott, S. May, Embedded subscriber identity module esim, Springer Business & Information Systems Engineering (2017).

[6] M. M. Joghal, M. A. Doostari, K. S. Hosseini, A. Golzarnia, Improving the security of sim applets by using tee, in: 27th Iranian Conf. on Electrical Engineering (ICEE), 2019.

[7] S. F. Mjolsnes, R. F. Olimid, Private identification of subscribers in mobile networks: Status and challenges, IEEE Communications Magazine (2019). doi:10.1109/MCOM.2019.1800511.

[8] D. E. Pertiwi, L. H. Kusumah, Identification of operational risk of embedded subscriber identity module (sim) technology based on iso 31000: Systematic literature review, SINERGI (2023).

[9] H. D. S. Medeiros, L. de Souza Bezerra, F. T. España, J. T. S. de Oliveira, Embedded-sim (e-sim) an overview in latin america: implementation, availability, advantages and disadvantages, Journal of Communication and Information Systems 39 (2024) 46–57.

[10] M. Motallebighomi, J. Veara, E. Bitsikas, A. Ranganathan, eSIMplicity or eSIMplification? Privacy and Security Risks in the eSIM Ecosystem, in: 34th USENIX Security Symposium (USENIX Security 25), 2025, pp. 5425–5444.

[11] AdaptiveMobile, Simjacker Technical Paper, Technical Report, AdaptiveMobile, 2019. https://info.enea.com/Simjacker-Technical-Paper?pk_vid=a01b627fbc8f9d7c174462625908d5c3.

[12] T. P. Lisowski, M. Chlosta, J. Wang, M. Muench, SIMurai: Slicing through the complexity of SIM card security research, in: 33rd USENIX Security Symposium (USENIX Security 24), USENIX Association, 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/lisowski.

[13] Y. Zhou, Y. Yu, F.-X. Standaert, J.-J. Quisquater, On the need of physical security for small embedded devices: A case study with comp128-1 implementations in sim cards, in: 17th Int. Conf. on Financial Cryptography and Data Security, Springer, 2013. Online: https://eprint.iacr.org/2013/224.pdf.

[14] Ramneek, P. Hosein, S. Pack, Pricing esim services: Ecosystem, challenges, and opportunities, IEEE Communications Magazine 61 (2023) 18–24. doi:10.1109/MCOM.008.2200702.
[15] B. A. Abdou, Commercializing esim for network operators, in: 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), 2019. doi:10.1109/WF-IoT.2019.8767260.
[16] Fridh, Albin, eSIM Re-Selling on Mobile App, 2020. Student Paper.
[17] H. Yuan, A. Baloian, J. Janak, H. Schulzrinne, esim technology in iot architecture, 2024. URL: https://arxiv.org/abs/2401.04302. arXiv:2401.04302.
[18] E. Vahidian, Evolution of the SIM to eSIM, Master's thesis, Institutt for telematikk, 2013. https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/262765/617036_FULLTEXT01.pdf?sequence=2&isAllowed=y.
[19] N. B. Khalifa, A. Benhamiche, A. Simonian, M. Bouillon, Profit and strategic analysis for mno-mvno partnership, in: 2018 IFIP networking conference (IFIP networking) and workshops, IEEE, 2018, pp. 325–333.
[20] A. Vesselkov, H. Hammainen, P. Ikalainen, Value networks of embedded sim-based remote subscription management, in: 2015 Conference of Telecommunication, Media and Internet Techno-Economics (CTTE), 2015, pp. 1–7. doi:10.1109/CTTE.2015.7347220.
[21] J. Zhao, B. Ding, Y. Guo, Z. Tan, S. Lu, Securesim: rethinking authentication and access control for sim/esim, in: Proc. 27th Annual Int. Conf. on Mobile Computing and Networking, ACM MobiCom '21, 2021.
[22] GSMA, RSP Technical Specification Version 3.1, Technical Specification, Global System for Mobile Communications Association, 2023. https://gsma.com/esim/wp-content/uploads/2023/12/SGP.22-v3.1.pdf.
[23] Android Open Source Project, Implement eSIM: Carrier privileges, 2025. https://source.android.com/docs/core/connect/esim-overview#carrier-privileges.
[24] ETSI, ETSI TS 102 223 V13.2.0 Smart Cards; Card Application Toolkit (CAT), 2025. https://www.etsi.org/deliver/etsi_ts/102200_102299/102223/13.02.00_60/ts_102223v130200p.pdf.
[25] Security Explorations, eSIM security, https://security-explorations.com/esim-security.html, 2025.
[26] D. A. Burgess, Proactive SIMs, https://deepsec.net/docs/Slides/2021/Proactive_SIMs_David_Burgess.pdf, 2021. Presented at DeepSec 2021.
[27] GSMA, SGP.26 v1.4: RSP Test Certificates Definitions, Technical Report, Global System for Mobile Communications Association, 2020. https://www.gsma.com/solutions-and-impact/technologies/esim/gsma_resources/sgp-26-v1-4-rsp-test-certificates-defintions/.
[28] D. A. Burgess, What is AT&T doing at 1111340002?, 2021. https://medium.com/telecom-expert/what-is-at-t-doing-at-1111340002-c418876c212c.
[29] GSMA, GSMA SAS Standard for Subscription Manager Roles Version 4.1, Technical Report, Global System for Mobile communications Association (GSMA), 2024. URL: https://www.gsma.com/solutions-and-impact/industry-services/wp-content/uploads/2024/07/FS.08-SAS-SM-Standard-v4.1.pdf, accessed: 2025-12-16.
[30] A. S. Ahmed, A. Peltonen, M. Sethi, T. Aura, Security analysis of the consumer remote sim provisioning protocol, ACM Trans. Priv. Secur. 27 (2024). URL: https://doi.org/10.1145/3663761. doi:10.1145/3663761.
[31] H. Wen, P. A. Porras, V. Yegneswaran, Z. Lin, Thwarting smartphone sms attacks at the radio interface layer., in: NDSS, 2023.
[32] T. Faisal, E. Marilly, Dimsim – device integrity monitoring through isim applets and distributed ledger technology, 2024. URL: https://arxiv.org/abs/2405.09916. arXiv:2405.09916.

Appendix: Programming an Applet into an eSIM Profile

The SIM Toolkit (STK) specifies how a SIM or eSIM can request actions from the device. These interactions are implemented by applets, small Java Card programs executed within the UICC/eUICC. In eSIMs, such applets can be embedded directly into a downloadable profile.

Embedding an applet into an eSIM profile consists of four main steps: developing and compiling the applet into a CAP file; extracting and reordering its components to construct the load block; embedding this load block into a `PE-Application` element; and finally declaring one or more runtime instances equipped with the appropriate permissions and STK capabilities. After provisioning, the eUICC installs, activates, and executes autonomously the applet's code within the secure element. The next subsections provide a short primer on how these applets are programmed and incorporated into a valid profile.

Applet development and CAP compilation

Applet development follows the Java Card programming model, a constrained dialect of Java designed for secure elements with strict memory and execution limits. Once implemented, the applet is compiled into a Converted Applet (CAP) file, which represents the standard binary format consumed by the Java Card Virtual Machine. A CAP file is a structured binary archive containing the executable logic of the applet together with metadata required by the Java Card Virtual Machine. When extracted, the CAP archive exposes components such as `Header`, `Directory`, `Import`, `Applet`, `Class`, `Method`, `StaticField`, `ConstantPool`, and `RefLocation`, each encoding definitions, methods, static initializers, imports, or linking information.

CAP Component Extraction and Reordering

Although the CAP file does not impose a specific internal ordering, the load order that must be respected during installation is standardized by the Java Card Forum and later adopted by GlobalPlatform. This standard defines how components must be arranged to construct a valid load block for consumption by the eUICC.

Although a Java Card CAP file is distributed as a single archive, it internally consists of the above mentioned multiple logical components (e.g., `Header`, `Directory`, etc.). While the CAP format itself does not impose a strict ordering on these components when stored as an archive, the Java Card and GlobalPlatform specifications mandate a precise load order during installation. To construct a valid `loadBlockObject` suitable for inclusion in an eSIM profile, the CAP file must therefore undergo an explicit extraction and reordering phase, as deviations from the prescribed component sequence result in installation failures at the secure element. Concretely, the process consists of the following steps:

1. The CAP archive is extracted into its constituent component files.
2. The extracted components are reordered according to the standardized component load order defined by the Java Card Forum and adopted by GlobalPlatform.
3. The reordered components are concatenated into a single byte sequence, excluding non-essential metadata such as optional debugging information.

Constructing the Application Load Package

To embed a Java Card applet into an eSIM profile, the CAP components must be transformed into the `loadBlockObject` field of the `PE-Application` profile element. The key parameters involved are:

- `loadPackageAID`: the unique Application Identifier of the applet package;
- `securityDomainAID`: the Security Domain under which the applet is managed (default if omitted);
- `loadBlockObject`: the concatenation of the ordered CAP components, represented in hexadecimal form and stripped of non-essential metadata.

Constructing the `loadBlockObject` involves the following steps: (i) extracting the CAP archive into its constituent component files; (ii) reassembling those components in the standardized load order; and (iii) encoding the resulting byte sequence in the DER-TLV format required by the eSIM profile. The final output is a deterministic load block that the eUICC can parse and install without ambiguity.

Declaring applet instances and configuration

Once the load block is defined, the `PE-Application` element may include one or more entries named `ApplicationInstance`. Each instance specifies a class AID, an instance AID, privilege flags, and optional personalization parameters. A relevant category of configuration values concerns the *Toolkit Application Specific Parameters* (Tag 'CA'), which define the resources and STK functionalities available to the applet. These parameters determine which elementary files can be accessed, whether timers are available, and which STK commands may be invoked by the applet. They are encoded within the `systemSpecificParameters` field of the profile.

Profile example

An example of a GSMA-compliant `PE-Application` structure within an eSIM profile, illustrating a load block and corresponding applet instance, is shown below:

```
value1 ProfileElement ::= application : {

  app-Header {
                mandated NULL,
                identification 29
             },

  loadBlock {
       loadPackageAID ‘D07002CA44'H,
       loadBlockObject '010012decaffed010204000108a0000005591010040200280304…
                         …090023101b008b4432343066800a10b680063680020'H
            },

  instanceList {
                 {
                  applicationLoadPackageAID 'D07002CA44'H,
                  classAID 'D07002CA44900101'H,
                  instanceAID 'D07002CA4490010101'H,
                  applicationPrivileges '000000'H,
                  applicationSpecificParametersC9 ''H,
                  systemSpecificParameters {
                    ts102226SIMFileAccessToolkitParameter '010001000A03010202030305000…
                                  …003555555'H
                                         }
                 }
                }
}
```