Smartphones as Practical and Secure Location Verification Tokens for Payments

Claudio Marforio, Nikolaos Karapanos, Claudio Soriente,
Kari Kostiainen and Srdjan Čapkun
Institute of Information Security
ETH Zurich
{firstname.lastname}@inf.ethz.ch

***Abstract*—We propose a novel location-based second-factor authentication solution for modern smartphones. We demonstrate our solution in the context of point of sale transactions and show how it can be effectively used for the detection of fraudulent transactions caused by card theft or counterfeiting. Our scheme makes use of Trusted Execution Environments (TEEs), such as ARM TrustZone, commonly available on modern smartphones, and resists strong attackers, even those capable of compromising the victim phone applications and OS. It does not require any changes in the user behavior at the point of sale or to the deployed terminals. In particular, we show that practical deployment of smartphone-based second-factor authentication requires a secure enrollment phase that binds the user to his smartphone TEE and allows convenient device migration. We then propose two novel enrollment schemes that resist targeted attacks and provide easy migration. We implement our solution within available platforms and show that it is indeed realizable, can be deployed with small software changes, and does not hinder user experience.**

I. Introduction

Fraudulent transactions at points of sale made with stolen or duplicated payment cards are a major problem. In 2010 alone, these transactions constituted one third of the 1.26 billion EUR total fraud in the Single Euro Payments Area [1]. To improve the security of existing payment systems, companies and researchers have suggested usage of mobile devices as a second-factor authentication mechanism [2], [3]. As most users already have smartphones, deployment of such second-factor authentication is practical. Many online service providers already employ second-factor authentication using smartphones. Examples of this approach are online banking applications [4] and Google 2-Step Verification [5]. In a typical implementation, the user reads a one-time *passcode* off the smartphone screen and enters it on the service's web page during login. Login operations to services like online banking are typically performed when the user has time to interact with his smartphone to complete the authentication process. In addition, web services are easily modifiable. Thus, in most cases, an extra authentication step can be added to the login procedure of a web service at little cost. This approach cannot be, however, integrated in point of sale transactions, because interactions, at a shop counter, with a smartphone are inconvenient and add undesirable transaction delay. Additionally, the payment terminal infrastructure is hard to modify.

Recent proposals leverage location data from the user's phone as the second authentication factor for payments [2], [3]. During a transaction, either the card issuer [2] or the user [3] can verify that the location of the user's smartphone matches the location of the point of sale terminal used for the transaction. Previous work, however, overlooks important usability and security aspects, as it requires changes in both the point of sale infrastructure and the user experience. Furthermore, it assumes a trustworthy mobile OS, even though compromise of mobile operating systems has become commonplace [6], [7].

To secure mobile services despite mobile OS compromise, researchers have proposed using system-wide Trusted Execution Environments (TEEs), such as ARM TrustZone [8], which provide isolated execution of applications and secure storage of credentials [9], [10]. Integrating system-wide TEEs with any second-factor authentication protocol, however, requires the verifying party (e.g., the card issuer) to correctly bind a user's identity to the TEE running on his mobile device through an enrollment scheme. How to establish this binding in the presence of a compromised OS is an open problem [11].

In this paper we propose a smartphone-based second-factor authentication solution for payments at points of sale that uses location data to identify fraudulent transactions. In contrast to previous work, our solution does not require changes to established user interaction models and is compatible with the existing point of sale infrastructure. We leverage system-wide TEE architectures to provide a secure system, despite mobile OS compromise. As part of our solution, we design two secure enrollment schemes for smartphones to bootstrap second-factor authentication for payments, which may also be used in other application scenarios. To summarize, We make the following contributions.

- We propose a smartphone-based second-factor authentication solution for payments at points of sale, that uses the phone's location as the second authentication factor. Our solution makes use of smartphone TEEs to resist mobile OS compromise.
- As part of our solution, we construct two secure enrollment schemes that allow a card issuer to bind

NDSS '14, 23-26 February 2014, San Diego, CA, USA
Copyright 2014 Internet Society, ISBN 1-891562-35-5
http://dx.doi.org/10.14722/ndss.2014.23165

the identity of a user with the TEE running on his device. The first enrollment scheme leverages the unique identity of the user's SIM card and resists adversaries that can remotely compromise the victim's mobile OS. The second enrollment scheme uses specially crafted SMS messages that are processed within the device baseband OS. This scheme provides protection against more powerful adversaries that can additionally perform hardware attacks on devices to which they have physical access.

- Through prototype implementation using an open-source baseband OS, Android devices, and an ARM TrustZone development board, we show that our solution can be easily deployed. It requires small changes to existing smartphones, and no changes to the point of sale infrastructure and the user experience. Our experiments show that, during a transaction, location verification takes less than 4 seconds on average, which is a tolerable delay for most payment scenarios.

- We survey known approaches for second-factor authentication using mobile phones and argue why they cannot be deployed in the considered application scenario. We also analyze commonly suggested enrollment schemes and show why they do not withstand strong attackers that we consider in this work.

The rest of the paper is structured as follows. Section II defines the problem we address is more detail. In Section III, we give background information on mobile device architectures, and Section IV defines the adversarial model. Our second-factor authentication solution, including the enrollment schemes, is explained in Section V. We analyze the security properties of our solution in Section VI and Section VII provides implementation details. We provide performance evaluation in Section VIII, and in Section IX we discuss payment system integration, privacy considerations, and applicability of our solution to other related use cases. Section X discusses alternative second-factor authentication mechanisms and enrollment schemes. Section XI surveys related work, and we end the paper with a brief summary.

II. Problem Statement

Our goal is to design a smartphone-based second-factor authentication mechanism that prevents fraudulent transactions at points of sale. In the following we detail the *requirements* for any deployable and secure solution to this problem.

We first aim to design mechanisms that must not change the user interaction model and the current point of sale infrastructure. Previous work shows that introducing changes to established user interaction models makes adoption of new security mechanisms impractical [12]. Similarly, having to update the deployed points of sale makes adoption of additional security mechanisms hard [13].

Second, a solution must remain secure despite a targeted adversary that may compromise the mobile OS on the victim's device. Current systems [4], [5] and related research proposals [14], [15], which use smartphones for second-factor authentication, assume the mobile OS to be trustworthy. This assumption is too strong, as the complexity of smartphone platforms has increased, mobile operating system vulnerabilities have become commonplace [6], [7].

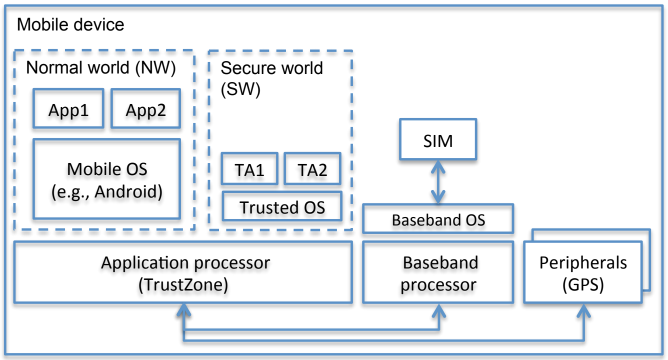


Fig. 1: Architecture overview of a TrustZone-enabled device.

Third, any second-factor authentication mechanism that replaces dedicated tokens with smartphones, must have an *enrollment scheme*, where the verifying party binds the identity of a user to his device. A dedicated security token is a user-specific device, which the service provider binds to the user identity before the token is shipped to the user. As smartphones replace such tokens, the service provider can only bind the user identity to his device after the user has already purchased the smartphone. In addition to initial enrollment, a practical solution must also support device migration. In applications like payments at points of sale, it is realistic to assume a one-time service registration performed, for example, when the user visits a branch of his bank in person. Requiring a similar operation every time the user starts using a new smartphone becomes both expensive for the bank and inconvenient for the user.

III. Mobile Device Architecture

A standard mobile device architecture has two processors. An *application processor* runs the mobile OS (e.g., Android) and the applications on top of it. A *baseband processor*, running the baseband OS, handles cellular communication and mediates communication between the application processor and the SIM card. Each SIM card has a unique identifier called IMSI (International Mobile Subscriber Identity).

Most mobile devices support system-wide TEEs, like ARM TrustZone [8]. In a TrustZone-enabled device, the application processor supports two execution states, namely, *secure world* and *normal world* (or non-secure world). The processor switches between these states in a time-slicing manner, so that only one state is active at a time. The normal world runs the mobile OS and regular applications on top of it. The secure world runs *trusted applications* (TAs), which are executed on top of a small layer of software called the *trusted OS*. The architecture of a TrustZone-enabled mobile device is shown in Figure 1.

Applications in the secure world are isolated from the mobile OS in the normal world. In contrast to dedicated security elements like smart cards, system-wide TEEs like TrustZone allow secure access to various device hardware

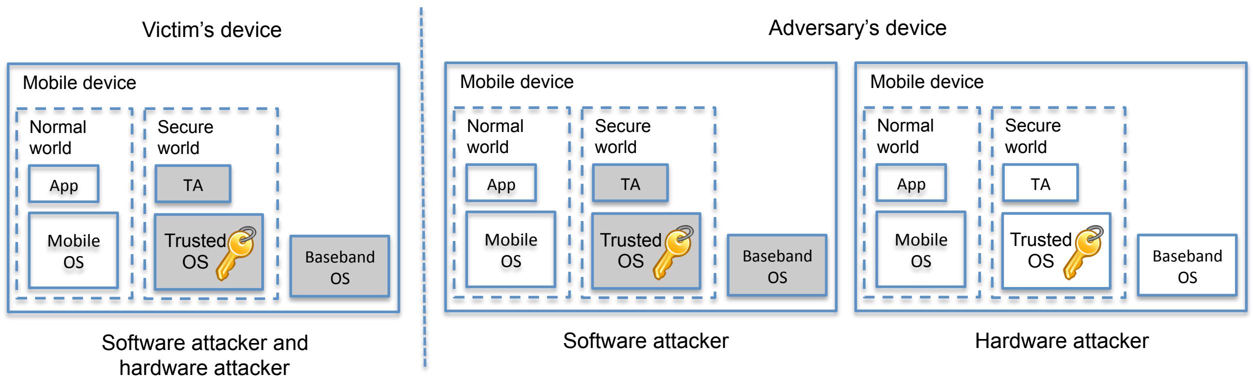


Fig. 2: Adversarial model. Grey boxes show trusted components for each type of adversary. The victim's device (on the left) can be only targeted with remote attacks and the TCB is the same in both attacker models. Trusted components on the adversary's device (on the right) depend on the considered attacker model: a software attacker cannot tamper with the TCB on his device; an hardware attacker has complete access to any component of his device.

resources from the TEE, and to configuration of secure mobile device event handling. Access to device peripherals and the baseband processor environment is typically possible by both the trusted OS in the secure world and the mobile OS in the normal world. In addition, certain peripherals can be reserved for secure world access only. Access to memory areas can be configured in a similar manner, and in a mobile device a small amount of memory is reserved for the secure world. Access control to hardware resources is implemented through specific control hardware and signals on the system communication bus. Hardware interrupts can also be configured for secure world processing if need be. For more details on TrustZone, see [8].

A standard trusted OS only allows execution of code that has been signed by a trusted authority, such as the device manufacturer. Typically, the device manufacturer ships each device with a device-specific key-pair (we refer to it as the *device key*). The public part of the device key is certified by the manufacturer, and the issued *device certificate* contains an immutable device identifier, such as the IMEI number (International Mobile Equipment Identity). The corresponding private key is only accessible by software that runs in the secure world [16].

To limit the size of the TCB, a typical trusted application handles only security-critical processing, such as user credential processing or data encryption. A *companion application*, running in the normal world, handles the communication, the UI rendering and other complex tasks. The rationale is that inclusion of complex libraries in the trusted OS, like network stacks or video drivers, considerably increases the size of the device TCB, and with that the attack surface of the secure world.

Device manufacturers have shipped their mobile devices with system-wide TEEs like ARM TrustZone for almost a decade. The usage of these environments, thus far, has been primarily limited to a few manufacturer-specific use cases, like implementation of subsidy locks and secure boot [16]. Deployment of third-party applications in system-wide TEEs has been limited, because the installation of new trusted applications is subject to the approval of the device manufacturer. Nevertheless, recent research shows that system-wide TEEs can be safely opened up for third-party trusted application development [17], and on-going TEE API standardization activities [18] are likely to make trusted application deployment more accessible to third-parties.

IV. Adversarial Model

We consider a targeted adversary who possesses the victim's payment card (or a clone) and knows its PIN code (if any). His goal is to perform a fraudulent transaction at a point of sale. The adversary does not have physical access to the victim's smartphone. He does, however, have access to other similar devices. We regard the device hardware, the trusted OS and the baseband OS as the TCB on the victim's device, and distinguish between two types of adversary.

A *software* attacker can remotely compromise the mobile OS on the victim's smartphone, but cannot compromise its TrustZone secure world nor the baseband OS execution. Likewise, he cannot compromise the TrustZone secure world nor the baseband OS on any other device. Software attacks against the mobile OS are a real threat [6], [7], while software attacks against the TCB (i.e., the baseband OS and the TrustZone secure world) are significantly harder, due to its limited size and attack surface.

A *hardware* attacker can additionally perform hardware attacks against devices that he owns or has physical access to. On such devices, he may compromise the baseband OS execution, the TrustZone secure world and, ultimately, extract the TrustZone-protected secrets, such as the device key. This adversarial model is justified by the fact that neither a TrustZone-enabled processor nor the baseband processor provide tamper resistance properties, commonly found in smart cards or hardware security modules. Figure 2 illustrates trusted software components (gray boxes) in these different scenarios.

Neither of the attackers controls the cellular network communication. Furthermore, they cannot launch GPS spoofing attacks on the victim's device. Finally, we do not address denial-of-service attacks.

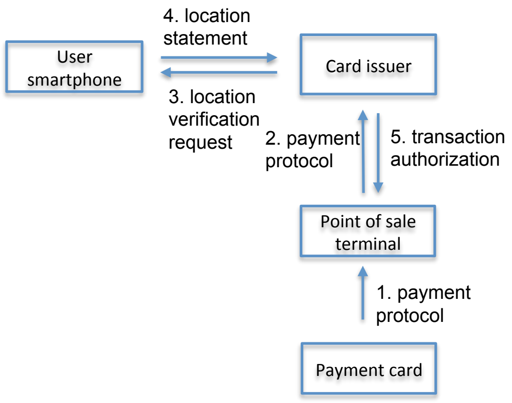


Fig. 3: Overview of location-based second-factor authentication for payments at point of sale. During a payment transaction, the card issuer queries the user's smartphone for its location over an Internet connection.

V. Our Solution

We use the location of the user's phone as the second authentication factor during a transaction at a point of sale. Our solution leverages system-wide TEEs available on mobile devices to provide card issuers with trustworthy location information despite a potentially compromised mobile OS on the user's smartphone. We focus on ARM TrustZone since it is currently the most widely deployed system-wide TEE on mobile devices. The schemes we propose can, nevertheless, be used with other system-wide TEEs as well.

Figure 3 shows an overview of our scheme. Prior to payments at point of sales, we assume that the user has already installed two applications provided by the card issuer on his device: a companion application running in the normal world and a trusted application running in the secure world. Additionally, the user has also completed an enrollment scheme (see below), and the card issuer has established a binding between the user and the TEE on his device. During payment, the user inserts or swipes his payment card in a point of sale terminal and optionally enters its PIN code (step 1). The terminal sends the transaction information to the card issuer (step 2). The card issuer contacts the TEE on the user's smartphone (step 3), which replies with a location statement (step 4). The card issuer then checks whether the location statement was sent by the correct device and compares it against the location of the terminal. Finally, the card issuer sends the transaction decision (authorize or deny) to the terminal (step 5).

We leverage location data due to two main reasons. First, we want a solution that does not change the user interaction model and the hardware infrastructure (see Section II). We therefore resort to the sensing capabilities of modern smartphones. Second, among available smartphone sensors, GPS units are almost ubiquitous, and previous work has shown that GPS coordinates are a practical and useful means that card issuers can leverage to identify fraudulent transactions [2], [3]. Our solution could, in principle, use any sensor available on the device.

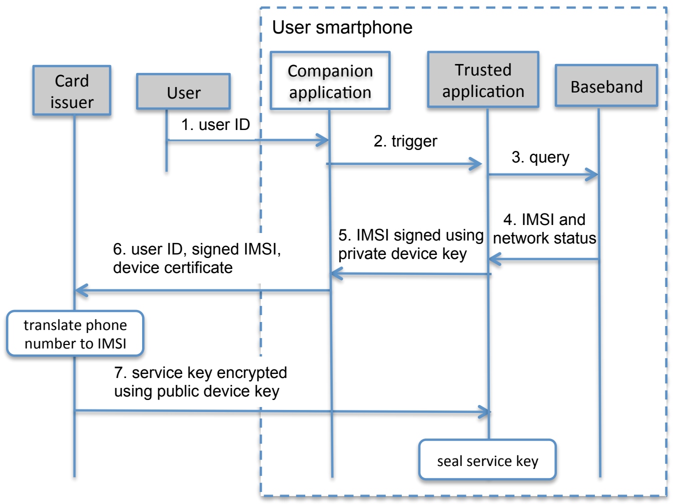

User smartphone
Card issuer
User
Companion application
Trusted application
Baseband
1. user ID
2. trigger
3. query
4. IMSI and network status
5. IMSI signed using private device key
6. user ID, signed IMSI, device certificate
translate phone number to IMSI
7. service key encrypted using public device key
seal service key

Fig. 4: Signed-IMSI enrollment scheme. Gray boxes are trusted entities. The trusted application fetches the IMSI of the installed SIM card from the baseband processor. It signs the IMSI with the private device key and forwards it to the card issuer (through the companion application). The card issuer can link the IMSI to a previously registered phone number.

A. *User Enrollment*

Before the card issuer can verify the location of the user's smartphone, it needs to bind the user identity to the TEE running on his mobile device. To achieve this binding, we present two enrollment schemes. The *signed-IMSI enrollment* scheme is easier to deploy but can only withstand software attackers; the *baseband-assisted enrollment* scheme is also secure against hardware attackers. However, it requires minor software changes to the baseband OS. Both schemes leverage the implicit binding between the user and his SIM card. They require a one-time registration in which the user provides his phone number to the card issuer in a reliable manner, for example, by visiting his bank's branch in person. The goal of both enrollment schemes is to establish a shared *service key*, between the card issuer and the trusted application running in the TEE on the user's device.

Signed-IMSI enrollment

The card issuer uses the SIM identifier (i.e., the IMSI) and the mobile network infrastructure to verify that the enrolling device is indeed the one where the user's SIM card is installed. Figure 4 illustrates the steps of the enrollment scheme.

The user starts the companion application and provides his user ID, e.g., the bank customer number (step 1). This application triggers the execution of the trusted application (step 2) that queries the baseband OS for the IMSI of the SIM card (steps 3-4).[1] The trusted application also verifies from the baseband OS that the device is connected to the mobile network, to discard the possibility of a fake SIM card reporting a false IMSI.[2] The trusted application signs the IMSI

[1]The IMSI is needed for cellular protocols and is available to the baseband OS through standardized interfaces.

[2]A false SIM card lacks the correct keying material to connect to the cellular network

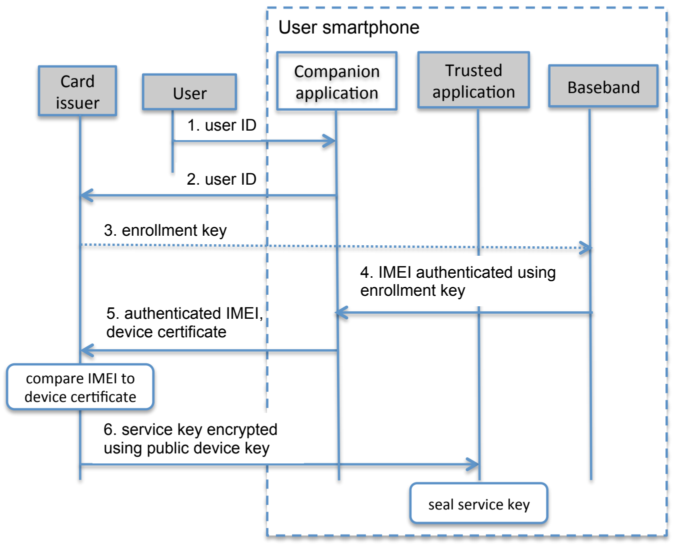

User smartphone
Card issuer
User
Companion application
Trusted application
Baseband
1. user ID
2. user ID
3. enrollment key
4. IMEI authenticated using enrollment key
5. authenticated IMEI, device certificate
compare IMEI to device certificate
6. service key encrypted using public device key
seal service key

Fig. 5: Baseband-assisted enrollment scheme. Gray boxes are trusted entities. The card issuer sends an SMS message with an enrollment key (dotted line); the baseband OS uses that key to authenticate the device's IMEI and deletes the key. The card issuer can check the authenticated IMEI against the one received in the device certificate.

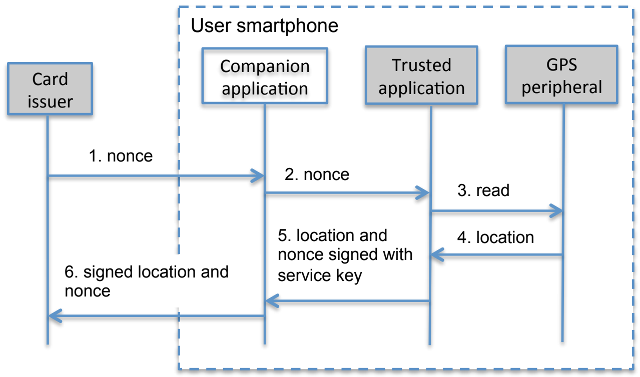


Fig. 6: Location verification is a simple challenge-response protocol using the service key established during enrollment. Gray boxes are trusted entities. The card issuer sends a nonce; the trusted application reads the GPS coordinates and authenticates them together with the nonce.

using its device private key (step 5) and passes the signature to the companion application. At this point the companion application sends the signed IMSI, the user ID, and the device certificate to the card issuer over the Internet (step 6). The card issuer uses the received user ID to retrieve the user's phone number and queries the mobile infrastructure for the corresponding IMSI (see Section VII for details). The card issuer checks that the IMSI received from the user's phone matches the one reported by the mobile infrastructure. If the two IMSIs match, the card issuer proceeds to verify (i) the validity of the device public key using the device certificate and the public key of the manufacturer, and (ii) the validity of the signature over the IMSI. If all checks are successful, the card issuer picks a fresh service key and encrypts it under the device public key; the ciphertext is sent to the user's smartphone (step 7). The companion application passes the encrypted service key to the trusted application that decrypts it using the private part of the device key and encrypts it using a symmetric storage key available only in the secure world (*sealing*). The sealed service key can be stored by the companion application in the normal world.

Baseband-assisted Enrollment

In this scheme the card issuer sends an SMS message carrying an *enrollment key* to the phone number provided by the user during registration. We augment the baseband OS to use this key and compute an authentication tag on the device's IMEI.[3] The steps of this scheme are in Figure 5.

The user starts the companion application and provides his user ID (step 1), which is forwarded to the card issuer over the Internet (step 2). The card issuer sends an enrollment SMS message to the corresponding user's phone number, containing a fresh enrollment key (step 3). The baseband OS on the user's device intercepts the SMS message and extracts the enrollment key. The baseband OS uses the enrollment key to authenticate the device's IMEI,[4] provides the authentication tag to the companion application (step 4), and deletes the enrollment key. The companion application forwards the authenticated IMEI and the device certificate to the card issuer (step 5). The card issuer checks (i) the validity of the device certificate, (ii) the validity of the authentication tag, and (iii) that the IMEI authenticated with the enrollment key matches the one in the received certificate. If all checks are successful, the card issuer picks a fresh service key and encrypts it under the device public key extracted from the device certificate; the ciphertext is sent to the user's smartphone (step 6). The companion application passes the encrypted service key to the trusted application that seals it.

B. Location Verification

After successful enrollment, the user's device shares a service key with the card issuer that can be used to create an authentic channel between the two parties. During a transaction payment, therefore, the card issuer can query the device for an authenticated location statement. As detailed in Figure 6, the location verification is a simple challenge-response protocol.

The card issuer picks a fresh nonce (for replay protection) and sends it to the trusted application through the companion application (steps 1-2).[5] The trusted application reads the location coordinates from the device GPS unit (steps 3-4), unseals the service key and uses it to authenticate the nonce and the current coordinates. The authenticated message is sent

[3]The IMEI is bound to the device key by the device certificate.

[4]The IMEI is read from a read-only memory on the device, written by the device manufacturer. The baseband OS has access to the IMEI to handle cellular communication.

[5]Alternatively, the card issuer could verify the freshness of location statements checking the timestamp provided by the GPS unit. As one of our goals is not to change the payment transaction user experience, the location verification must be initiated by the card issuer. Thus, a location verification requires the exchange of two messages between the card issuer and the user smartphone in either cases (i.e., usage of the GPS timestamp does not allow a more efficient implementation).

back to the card issuer through the companion application (steps 5-6). The card issuer verifies the authenticity of the location statement using the service key. At this point, the card issuer matches the location of the user's smartphone against the one of the terminal used for the transaction, to decide whether to authorize or deny the transaction.

We note that neither the companion application nor the trusted application need to be continuously running in the background. The execution of the trusted application is triggered by the companion application which, in turn, is started by the mobile OS when receiving a request for that application (e.g., through a push notification).

C. Device Migration

Both enrollment schemes support device migration by re-running the enrollment operation. When the user switches to a new device and moves his SIM card to it, he can start the enrollment process from the companion application of the new device. As a result of either the signed-IMSI or the baseband-assisted enrollment, the card issuer invalidates the previously used service key, re-associates the user identity to the device key of the new device, and sends a fresh service key to the TEE on that device. Device migration does not require out-of-band communication with the card issuer as long as the user keeps his phone number (even if he gets a new SIM card associated with his old phone number).

VI. Security Analysis

Recall that our adversary holds the victim's payment card (or a clone) and his goal is to make fraudulent transactions at points of sale. The adversary does not have physical access to the victim's smartphone but he does have remote control over that smartphone's mobile OS.

With the deployment of our system, the adversary must convince the card issuer that the enrolled user's smartphone is close to the terminal where the fraudulent transaction is taking place. To do so, the adversary must either succeed in an impersonation attack during enrollment or tamper with the location verification protocol.[6]

In an impersonation attack, the adversary must halt the enrollment scheme on the victim's device (he can do so since he controls that device's mobile OS), and use the victim's ID to run the enrollment scheme on a device that he owns. In particular, the adversary has two possible *strategies*: he must induce the card issuer to encrypt the service key either

(a) under the public key of the adversary's device (the adversary will thus be able to make fraudulent transactions if his phone is in proximity of the terminal where the transaction takes place), or

(b) under a public key for which the adversary knows the private key (so that the adversary will be able to generate arbitrary location statements when the card issuer requests them).

In the following we provide an informal analysis of the enrollment schemes, with respect to the two strategies above. Finally, we argue that the location verification mechanism is secure after successful user enrollment.

[6]We acknowledge that the adversary may still succeed in his goal if the fraudulent transaction takes place close to where the victim's device is located.

A. Signed-IMSI enrollment

This enrollment scheme is secure against a software attacker as defined in Section IV. This attacker can control the mobile OS on any device (including the victim's) but does not have sufficient capabilities to control any baseband OS or the TrustZone secure world execution environment.

Strategy (a) requires the adversary to start an enrollment scheme on his device using the victim's ID. Since the card issuer knows the victim's phone number and can retrieve the corresponding IMSI, the adversary must force the trusted application on his device to send the IMSI of the victim's SIM card. To do so, the adversary may use a custom SIM card where he can manipulate the IMSI. However, such SIM card misses the key that the victim's SIM card uses for authentication with the network operator and, thus, cannot connect to the cellular network. When the trusted application on the adversary's device queries the baseband OS for cellular network status, it detects that the phone is not connected and will abort the enrollment scheme.

Strategy (b) requires the adversary to hold a private key corresponding to a valid (i.e., certified by the device manufacturer) public key. This is not possible since a software adversary cannot compromise the ARM TrustZone architecture of any device and leak the secrets stored therein.

B. Baseband-assisted Enrollment

This enrollment scheme is secure against an hardware attacker, as defined in Section IV, that controls the mobile OS on any device (including the victim's), as well as the baseband OS and the TrustZone secure world execution environment on devices to which he has physical access.

Strategies (a) and (b) require the adversary to either intercept the enrollment SMS message and extract the enrollment key, or provide a crafted IMEI to the baseband OS on the victim's device. Since the adversary does not control the GSM network, SMS messages cannot be intercepted. Furthermore, the enrollment key is deleted by the baseband OS, so that the normal world cannot read it. Finally, the IMEI is stored on read-only memory during device manufacturing [16], thus the adversary cannot feed an arbitrary IMEI to the baseband OS on the victim's device.

C. Location Verification

After a successful enrollment, the trusted application running in the secure world on the victim's device shares a service key with the card issuer. At this time, the adversary can only try to force the victim's device to report a location statement with GPS coordinates matching the location where the fraudulent transaction takes place. Since none of the considered adversaries (i.e., software and hardware) control the secure world on the victim's device, the adversary can only try to change the coordinates provided by the GPS unit on that device. We note that GPS units on modern smartphones only allow to reset the GPS sensor. That is, the adversary cannot feed the trusted application on the user's phone with arbitrary coordinates. The trusted application can, nevertheless, detect a reset and restart the GPS sensor.

VII. Implementation

We implement three prototypes to evaluate the feasibility of the proposed second-factor authentication solution and enrollment schemes. First, we modify an open-source baseband OS to show that the changes required to existing baseband operating systems are small. We test our baseband modifications on an older mobile phone, because the baseband environment on modern smartphones is not modifiable by third-party developers.

Second, we implement the trusted application on a TrustZone development board to show that its deployment on TrustZone-enabled devices is straightforward, and that the time overhead to generate location statements is negligible compared to network delays. We use a TrustZone development board because installation of trusted applications on current smartphones requires approval by the device manufacturer.

Third, we implement a client-server prototype using an Android smartphone to evaluate the end-to-end performance of the location verification mechanism at the time of a payment transaction.

A. Baseband Implementation

To accommodate the baseband-assisted enrollment scheme of Section V-A, we augment osmocomBB [19], which is the only available open-source baseband OS. It is implemented for Motorola mobile phones like the C123 or C118, introduced in 2005. The GSM layer 1 (the physical layer) executes directly on the mobile phone, while layers 2 and 3 (respectively the data-link layer and the third layer, subdivided in the Radio Resource management, the Mobility Management, and the Connection Management) run as the `mobile` application on a host PC connected with the device through a USB-to-serial cable.

We leverage the widely used SMS Protocol Data Unit mode, standardized in [20], to format the enrollment SMS message sent by the card issuer to the user's phone number. In the standard, a User Data Header structure can contain so called Information Element Identifier elements, that are reserved for future use. We encode the enrollment key in the Information Element Data field of one such identifier. We add to the baseband OS the logic to identify and handle enrollment SMS messages. Once the key is found in the SMS message header, the baseband OS extracts it and computes an authentication tag over the device's IMEI. We use the HMAC-SHA1 implementation provided by the PolarSSL [21] library as the authentication algorithm.

We test the prototype baseband OS in a Motorola C118 connected through a USB-to-serial cable to a host PC running Ubuntu 12.10. The original `mobile` application provided by osmocomBB consists of 19,482 lines of C code; we add a total of 523 lines of code, where `polar_sha1.c` accounts for 451 lines of code. Our changes increase the code size by 2.7%. In terms of binary size, a compiled version of the original `mobile` application is 2029 kB; our modified version accounts for 2077 kB in total (i.e., 2.3% larger). The layer1 firmware (`layer1.compalram.bin`) that is installed on the mobile phone accounts for 63 kB and remains unmodified.

B. Trusted Application Implementation

We implement a trusted application that provides location statements, on a TrustZone-enabled development board. We use it to evaluate the implementation complexity and the time required to produce location statements. The board is an ARM Motherboard Express uATX [22] coupled with an ARM CoreTile Express A9 [23]. The board features a Cortex-A9 processor which is clocked at 400 MHz. The development board contains no GPS unit or baseband processor. In the normal world of the system, we run Android version 4.1.1 with Linux kernel version 2.6.38.7, properly patched to support the ARM board as well as Android. In the secure world, we run Open Virtualization SierraTEE [24], release "02 June 2013". SierraTEE is an open-source framework that provides a basic secure world kernel, compliant with the GlobalPlatform TEE specifications [25].

The implementation of the trusted application accounts for less than 150 lines of code. Thus, incorporating our trusted application into an existing trusted OS, that already provides the necessary cryptographic functions and system calls, would hardly change the existing memory and storage requirements.

Location Verification

The application that generates location statements runs on top of Open Virtualization, in the secure world, while the companion application runs in the normal world, on top of Android. When the card issuer initiates a location verification protocol with the user's smartphone, that device switches from normal world to secure world and executes the trusted application that generates the location statement.

We set up an experiment where the companion application, running in the normal world, invokes the trusted application and provides it with a 128-bit nonce. As the development board has no GPS unit, we emulate it by creating a system call in the secure kernel that just returns longitude, latitude and accuracy values. The trusted application runs HMAC-SHA256 over the data fetched from the system call and the provided nonce. The location statement is returned to the companion application in the normal world. A shared memory buffer is used for exchanging data between the two worlds.

We measure the total time required for the companion application to receive a location statement from the trusted application. This time includes (i) the performance delay introduced by the context switching and required data copying between the normal world and the secure world, and (ii) the time it takes for the trusted application to generate a location statement. The above experiment is repeated 1000 times. Average completion time is 3.0 milliseconds, with a standard deviation of 0.04 milliseconds. The time spent in context switching between the normal world and the secure world is below one millisecond.

Enrollment

Since our board is not equipped with a baseband processor, we do not implement the enrollment schemes of Section V-A. Nevertheless, we now explain how they can be realized.

During the signed-IMSI enrollment scheme, the trusted application must query the baseband OS for the IMSI of the installed SIM card, and for the cellular network status. In a mobile device, communication between the baseband OS and the mobile OS (e.g., Android) is implemented through a manufacturer-supplied binary (e.g., a driver). A stripped-down version of this binary may be as well installed in the secure world by the device manufacturer. Given that subsidy lock is one of the most used services in TrustZone-enabled devices, it is reasonable to assume that the secure world is able to communicate with the SIM card in a modern smartphone [16]. For reference, the full binary in the Samsung Galaxy S3 phone (i.e., `libril.so`) is 49 kB. The complete API offers roughly 200 function calls (extracted by looking at the *strings* of the binary) to the baseband OS. In contrast, the stripped-down version to support enrollment only requires the function calls `GET_SIM_STATUS` and `GET_IMSI`.

In the baseband-assisted enrollment scheme, the trusted application is only invoked at the end of the process to decrypt and seal the service key sent by the card issuer. Hence, there is no requirement for direct communication between the secure world and the baseband OS.

C. Client-Server Implementation

To evaluate the performance of the location verification protocol, the client prototype provides the functionalities of both the companion application running in normal world, and the trusted application running in secure world. This implementation does not account for the needed context switching between the normal world and the secure world. As mentioned before, this time is below one millisecond, and thus negligible compared to networking delays of a full end-to-end implementation.

We develop against the API level 16 of the Android SDK (version 4.1, "Jelly Bean") [26]. Cryptographic operations are based on the Bouncy Castle crypto library [27]. We use 2048-bit RSA keys as device keys. Authentication of location statements leverages HMAC-SHA256 with an 128-bit service key. Communication between the server and the client uses the push notification feature of Google Cloud Messaging (GCM) [28]; the reverse channel is a standard HTTP connection.

The client provides functionalities for the signed-IMSI enrollment scheme (cf. Section V-A) and the location verification mechanism (cf. Section V-B). During enrollment, the application queries the baseband OS through the Android Java API provided by the `TelephonyManager` service, for the IMSI of the SIM card and the network connection status. During location verification, the application reads the GPS location (latitude and longitude, accuracy and satellite fix time) using the `LocationManager` system service.

The server-side processing is implemented in python, using the CherryPy Web Framework [29] and SQLite [30]. This web service is accessed through a RESTful web API that provides enrollment and location verification operations. During the signed-IMSI enrollment scheme, the server translates a phone number to the corresponding IMSI using an HLR-lookup query with an external service provider. An HLR-lookup query is carried out by network operators using the Signaling System #7 (SS7) protocols. In particular, the Home Location Register (HLR) of a network operator holds information about its users such as their phone numbers and to which network a device is currently connected. Among other information, the HLR holds the IMSI of the SIM card connected to the network. Several HLR lookup services, such as [31], are available to third-parties.

While our client prototype implementation is tailored towards a device using Android OS, similar functionalities are easily done on other smartphone platforms.

	Static tests			**Field study (3G)**	
	WiFi (n=101)	3G (n=101)	Edge (n=101)	Orange (n=46)	Sunrise (n=34)
average (sec)	0.60	1.82	2.20	2.54	3.68
std dev (sec)	0.08	0.05	0.30	0.78	1.45

TABLE I: Completion time for location verification during payment transactions. *n* denotes the number of samples in each scenario.

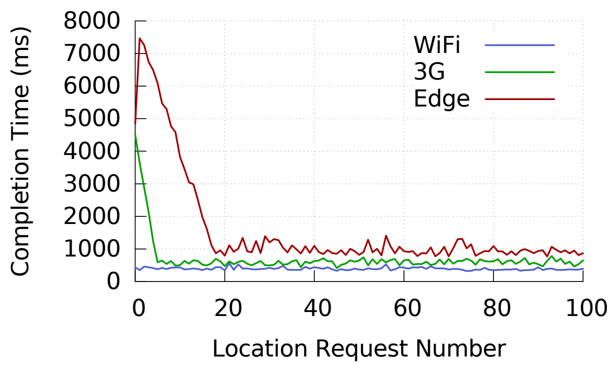
VIII. Experimental Evaluation

The previous section shows that context switching between the two TrustZone execution states (i.e., normal and secure world) and cryptographic operations to produce location statements, require only a few milliseconds. Network delays, therefore, account for the majority of the time required to verify the location of the user's device by the card issuer. In this section we analyze the time to complete location verification and present the experimental evaluation of our client-server prototype.

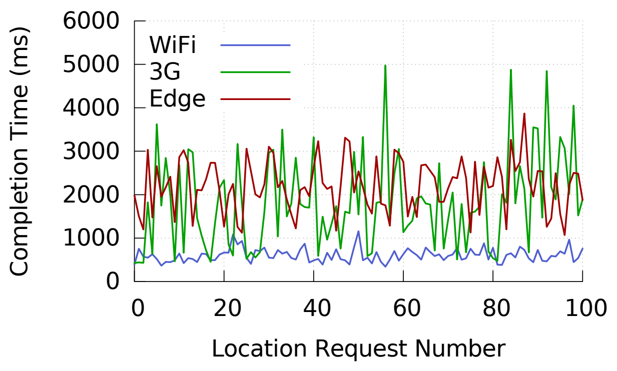
We focus on the location verification protocol as the enrollment procedure is a one-time operation and its performance is less critical. The client prototype is installed on a Samsung Galaxy S3 smartphone with the latest software updates (as of the time of writing), after a factory reset. The server is running on a standard laptop and shares a service key with the client. We provide results for both *static tests* run with the phone in a fixed location (office environment) and a *field study* run in a scenario close to actual deployment. Table I provides an overview of our results which we elaborate below.

A. Static Tests

During static tests, the client device is in a fixed position, on a desk in our office environment. We run tests for Edge (GSM only mode), 3G (WCDMA only), and WiFi (mobile data turned off) connections. For each connection setting, we measure the completion time, i.e., how long it takes from the moment the server issues a request, until the moment it receives the location statement and verifies its authenticity. The experiment is repeated 100 times (the server issues one request per second), and Figure 7a shows the completion time for each location verification. Results show longer completion times during the first runs, for Edge and 3G connections. This behavior is presumably caused by the time it takes to "activate" the radio on the phone. To validate our hypothesis, we set the interval of two consecutive server requests to 30 seconds, allowing the radio of the phone to "deactivate" after each request. Results are shown in Figure 7b. Confirming



(a)



(b)

Fig. 7: Completion time for 100 location verifications. In Figure (a) the server initiates one request per second. In Figure (b) the server waits 30 seconds before issuing each request.

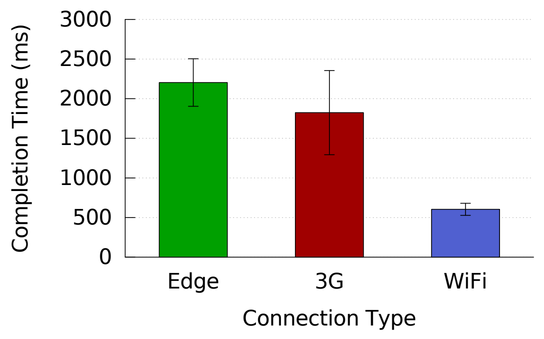


Fig. 8: Average time and standard deviation required to complete a location verification, for different connection types.

our hypothesis, completion time in this scenario has greater variance, especially when using Edge or 3G networks.

Finally, Figure 8 shows the average and the standard deviation for the measurements of Figure 7b. The completion time for our solution is, on average, 2.2 seconds with an Edge connection, 1.82 seconds with a 3G connection, and 0.6 seconds with a WiFi connection, respectively.

B. Field Study

To test our solution in a setting close to the actual payment scenario, we use two client devices and carry out a two-day field study with two subjects in Zürich. Each subject carries a client device and a triggering device. The two clients have SIM cards issued by different operators (*Orange* and *Sunrise*).

The triggering device initiates a server request. The server runs on a standard laptop and listens for incoming triggers. We use a separate trigger device to make sure that, at the time of a location verification, the radio on the client device is not active. In an actual deployment, if the radio happens to be active when a location verification request is received, completion time is expected to be smaller than the ones reported below.

For two days, each subject uses the triggering device to initiate a location verification each time he is close to a payment terminal in, for example, shops, cafes, museums, parking lots, etc. Completion time, for the device with the Orange SIM card, is 2.54 seconds on average (standard deviation 0.78 seconds). The smartphone with the Sunrise SIM card shows slightly worse performance as the completion time raises to 3.68 seconds on average (standard deviation 1.45 seconds).

GPS accuracy (m)			**GPS fix delay (ms)**		
average	max	min	average	max	min
17.40	48.0	4.0	256.83	3430.00	0.00

TABLE II: Location accuracy results during the field study.

Table II shows the accuracy of the location information and the elapsed time between the server request and the actual GPS fix on the client. The reported accuracy is within an acceptable range to distinguish shops next to each other (17.40 meters on average), and the location fix is available with a very short delay (less than 257 milliseconds on average).

IX. Discussion

This section further discusses the proposed mechanisms in terms of integration with current payment systems, deployment considerations, and privacy issues. Finally, we discuss the applicability of our solution to other scenarios.

A. Integration with Payment Systems

Our protocols can increase the security of any payment card transaction (either "chip and PIN" or "swipe and sign") where the card issuer is contacted by the terminal to authorize or deny the payment. We now detail the integration of our solution with deployed payment systems and, in particular, with the EMV payment standards [32].

An EMV transaction involves the card, the terminal, the card issuer and an *acquirer*. The acquirer is a banking institution that processes credit and debit card payments for merchants. Third-party payment processors may be also involved, or the issuer and the acquirer may be the same banking institution. EMV specifications for transactions with

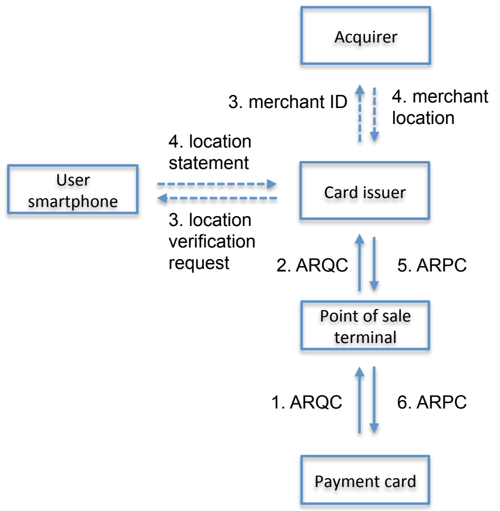


Fig. 9: Integration of the location verification protocol within the EMV payment standards. Dashed lines indicate the additional messages required for our solution.

online authorization dictate two *cryptograms* (authenticated messages) exchanged between the card and its issuer. The cryptogram sent by the card is denoted as the Application Request Cryptogram (ARQC) and accounts for a number of fields that are supplied either by the card or by the terminal. Mandatory fields for ARQC include the transaction amount, the transaction date, and a random nonce generated by the terminal. EMV also defines optional fields that individual payment systems (e.g., Mastercard M/Chip or Visa VSDC) may require. The card issuer replies with an Application Reply Cryptogram (ARPC) that notifies the card whether the transaction has been approved. Figure 9 illustrates the messages defined in the EMV standards; dashed arrows show the additional messages required to implement our solution.

Our solution requires the card issuer to know the physical location of the terminal used for a transaction. In the EMV standards an *acquirer ID* globally identifies a banking institution, while a *merchant ID* identifies a merchant within a banking institution. Either the card or the terminal can request that these elements are included in the ARQC message. Once the card issuer learns the acquirer and merchant identifiers of a transaction, it can contact the acquirer with the purported merchant ID in order to retrieve the merchant location. Popular payment systems are already providing merchant information, such as location, through standardized APIs [33], [34].

In current EMV payment systems, the acquirer ID and the merchant ID fields are defined by the standard but are not mandatory for ARQC messages. Through card software updates it is possible to include these fields in ARQC messages of every transaction. The EMV standards allow for remote update of payment card software through *issuer scripts* that may be included in the ARPC cryptogram.

A noteworthy benefit of our solution is that it enables gradual and secure deployment for selected users. A card issuer can enable location verification for a subset of its customers, for example, by updating the payment cards of customers that decide to opt-in. Once location verification is enabled for these users, an adversary that has obtained a payment card of any such user cannot circumvent the system. In comparison, solutions that require gradual terminal updates do not provide similar protection, as the adversary can always bypass the added security mechanisms by utilizing not yet updated terminals.

B. *Deployment Considerations*

Even though fraud reduction is a clear incentive for banks or payment card issuers to adopt our solution, it is the user adoption that will be the driving factor. The proposed schemes do not change the user experience at point of sales, but they do consume internet bandwidth and battery on the user's device. Since payment card issuers are currently covering the costs of frauds, it is an open question whether users would pay the additional costs in terms of bandwidth and battery on their mobile devices, without any apparent benefit. To boost the user adoption, the card issuer may offer lower fees to users who opt-in, just like car insurance companies do for customers who install anti-theft mechanisms on their cars.

Our solution assumes the user's smartphone to have Internet connectivity at the time of a transaction. This may not hold for two reasons. First, high roaming charges induce many users to turn Internet access off while traveling abroad. International regulatory bodies have started to forbid excessive roaming charges. For example, the EU has recently decided to completely remove roaming charges within its member countries by 2015 [35]. A cost-effective alternative to avoid current roaming charges is to use SMS messages for communication between the user's device and the card issuer. SMS-based communication, can however experience high delays (SMS message delivery is based on a best-effort basis and can take up to 12 seconds in normal load conditions [36]). Second, Internet connectivity might not be available in remote areas or underground locations (although many underground shopping centers or transport stations have cellular coverage).

Card issuers can handle lack of connectivity based on transaction value, merchant location or user specific policies. For example, high-value transactions in areas where Internet connectivity is expected to be available may only be authorized after a successful location verification with the user's smartphone. A possible solution to handle temporary lack of connectivity for low-value transactions could be to keep, on the device, an authenticated log of timestamped locations and report the one that is closest to the transaction time, once Internet connectivity is again available. While this solution does not allow for real-time fraud prevention, it allows card issuers to perform offline fraud detection.

C. *Privacy Considerations*

The card issuer can ask the user's device for location statements and track the user over time. However, if the protocol is triggered at times of genuine transactions, our solution does not leak extra information, since the card-terminal transaction already reveals the user location to the card issuer. The card issuer may also abuse the system and query the user's device for a location statement when the card is not involved in a transaction. We argue that the system abuse can be prevented through precise terms of agreement; card issuers that break those terms will damage their reputation and lose customers. Another solution is to let the card issuer send the location of the point of sale terminal to the device and let the device compare it against its current location, as done in [3].

With respect to third-parties (i.e., law-enforcement authorities), location statements issued by the user's device can be denied. Since the (symmetric) service key used to authenticate location statements is shared with the card issuer, no third-party can identify who produced a location statement (either the user's device or the card issuer). Finally, we remark that an adversary in control of the mobile OS on the victim's device can query the GPS unit at will and track the user, independently of the solution presented in this paper.

D. Other Application Scenarios

Our protocols can be applied to other application scenarios beside payments at points of sale. In particular, they can be used in any scenario where the verifying party (e.g., a service provider) knows the user's phone number and the location of the infrastructure used to perform transactions. Two prominent examples are public transportation ticketing and building access.

In the public transportation scenario the user holds a transport ticket (e.g., an NFC card) that is used at dedicated machines to access the transportation network (e.g., at the entrance of metro stations). Our solution can provide assurance to the public transportation authority that a lost or stolen transport ticket is not used by any party but the rightful owner.

Similarly, building access control systems require a user to carry an access token with a short-range wireless interface. Entry is granted if the token is presented to a dedicated reader and the valid PIN is entered by the user. Location statements can increase the security of such access control systems. Upon presenting the access token to the reader, the user phone is queried for its location; if the purported location matches the one of the reader, the building access authority can grant access.

As part of our field study (described in Section VIII), we tested completion time and accuracy of our location verification protocol in public transportation and building entrance scenarios. Results are summarized in Table III and Table IV.

Completion time at public transport stations takes 3.03 seconds on average, using a 3G connection, for the device using Orange, and 3.39 seconds on average for the smartphone using Sunrise. We argue that three seconds to grant access to the transportation network may be an undesirable delay. Nevertheless, our solution could be used for offline ticket abuse monitoring. The public transportation authority could disable a ticket after witnessing a number of consecutive fraudulent uses. In this scenario, the measured location accuracy (see Table IV) is around 14 meters on average. In most public transportation applications this accuracy is sufficient to distinguish entrances to the transportation network from one another.

We also test the time it takes to run our protocol when entering buildings in two campuses of ETH Zurich, one in the city center and the other in the city suburbs. Completion time takes 2.31 and 4.40 seconds on average, depending on the network operator used as shown in Table III. The location accuracy in this scenario is around 15 meters, which is enough to differentiate building doors from each other in most cases.

	Building access		**Public transport**	
	Orange (n=59)	Sunrise (n=40)	Orange (n=43)	Sunrise (n=63)
average (sec)	2.31	4.40	3.03	3.39
std dev (sec)	0.57	1.74	0.66	1.33

TABLE III: Completion time for location verification for public transport and building access tests. *n* denotes the number of samples in each scenario.

Scenario	**GPS accuracy (m)**			**GPS fix delay (ms)**		
	avg	max	min	avg	max	min
Building access	14.04	48.0	4.0	139.31	3087.00	0.00
Public transport	15.47	48.0	6.0	210.52	4035	0.00

TABLE IV: Location accuracy for public transport and building access tests.

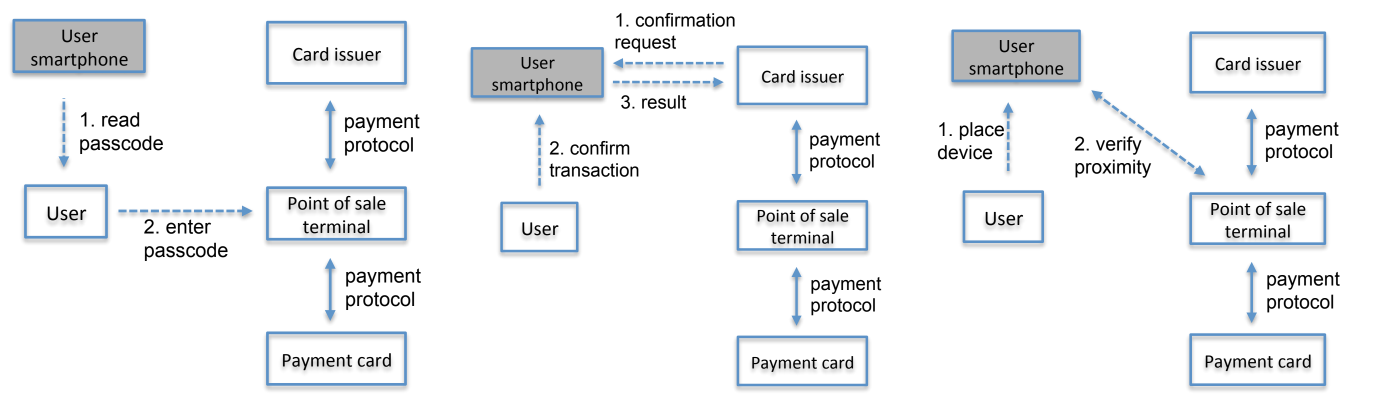
X. Alternative Approaches

In this section we discuss alternative ways in which smartphones could provide a second authentication factor for payments at points of sale, and conclude that location verification provides a practical means for card issuers to identify fraudulent transactions. We also analyze commonly suggested enrollment schemes and show how they fail to provide secure user-to-device binding, given our realistic attacker model. Finally, we describe alternative TEEs available on current smartphones and their shortcomings, compared to system-wide TEEs such as ARM TrustZone.

A. Second-factor Authentication Approaches

In a typical transaction at a point of sale, the user enters or swipes his payment card into a terminal and optionally types its PIN code. The card runs a protocol with the terminal that contacts the card issuer for online transaction verification. Figure 10 illustrates common second-factor authentication approaches: *(1) Authentication token replacement (Figure 10a).* The smartphone acts as a dedicated authentication token and displays one-time passcodes that the user must type into the terminal. Google 2-Step Verification [5] is a prominent example of this approach in the context of web login authentication. A similar approach can be applied to payments at points of sale. *(2) User confirmation device (Figure 10b).* The card issuer contacts the user's device which presents a confirmation dialog to the user. The confirmation result is sent back to the card issuer. Authentication solutions like this one have already been deployed for online banking [37]. *(3) Distance-verification device* (Figure 10c). The user places his smartphone next to the payment terminal, which starts a distance-verification protocol over a short-range wireless connection, such as NFC [38].

The given approaches require additional user interaction at the time of the transaction. Changes to established user interaction models hinder the introduction of new security mechanisms [12]. Additionally, the majority of point of sale terminals



(a) *Smartphone as an authentication token replacement:* the user reads a passcode off his smartphone and enters it into the payment terminal.

(b) *Smartphone as a user confirmation device:* the user confirms the transaction using his smartphone; the devices delivers the user's decision to card issuer.

(c) *Smartphone as a distance verification device:* the user positions the phone next to the terminal; the terminal verifies the proximity of the phone over short-range wireless connection (e.g, NFC).

Fig. 10: Common smartphone second-factor authentication approaches applied to payment systems. Solid arrows represent standard transaction messages, while dashed arrows show additional messages for second-factor authentication.

do not have the required software components for passcode entry (Figure 10a) or hardware interfaces for distance-verification (Figure 10c). The replacement of deployed terminals would be gradual and optional, which allows the adversary to target the terminals that have not been updated yet.

B. Common Enrollment Solutions

We now explain why commonly suggested enrollment schemes are not secure or feasible to deploy, assuming an adversary that controls the mobile OS on the victim's device.

Device Identifier Enrollment

A simple way to bind a user identity to the device key of his TEE, is to leverage the device's IMEI, typically included in the device certificate. The IMEI is available on the device's package or displayed on-screen. During enrollment, the user provides the IMEI of his device to the card issuer using a reliable out-of-band channel, for example, visiting a branch of the card issuer in person. The card issuer then verifies the device certificate with respect to the IMEI provided by the user.

Communicating the IMEI to the card issuer in a trustworthy way is more complicated than it seems. Device sales packages are not always available. Also, if the mobile OS is compromised, the adversary controls the IMEI shown on the device screen. Additionally, IMEI-based enrollment does not provide flexible device migration: every time the user changes devices, he must provide the IMEI of the new device to the card issuer, using an out-of-band channel.

Password Enrollment

The user-to-device binding can also be implemented by asking the user to enter a password or a similar initialization secret, known by the card issuer, in his device. A trusted application can authenticate itself to the card issuer, using the certified device key and the user-provided password.

The user should type in the password only when a trusted application can securely receive it. The compromised mobile OS can otherwise intercept and forward the password to the adversary, who can then launch an impoersonation attack. A reliable communication interface from the user to a trusted application is called *trusted path* [39], [40]. The device hardware and software resources used for user interaction (e.g., the display buffer or the touchscreen input events) can be temporarily reserved for system-wide TEEs such as ARM TrustZone. A *security indicator*, such as a colored bar on the top of the screen [41] or a dedicated LED [42] can be used to inform the user about the type of application he is communicating with (trusted or untrusted). However, current smartphones do not support this division of user interface resources, nor do they provide dedicated security indicators. Furthermore, several academic studies, and a few decades of practical experience, have shown that users tend to ignore security indicators [43], [44], [45].

Previous work assumes that password-based enrollment is secure if the enrollment is done early in the device life cycle, before the adversary has the opportunity to compromise the mobile OS [14], [46]. This assumption is hard to justify since not every service enrollment happens at the beginning of a device life time.

SMS Enrollment

If the user provides his phone number during registration, the card issuer can send an SMS message that will be received by the device in which the user's SIM card is installed. Similarly to our solution, the SMS message could carry an initialization secret to bootstrap security services. The problem with this approach is that SMS messages provide a trustworthy channel to the the baseband OS of the device where the SIM card is installed, but not to the secure world on that device. In current mobile device architectures, the baseband OS is accessible by both the mobile OS in the normal word and by the trusted applications running in the secure world. Therefore,

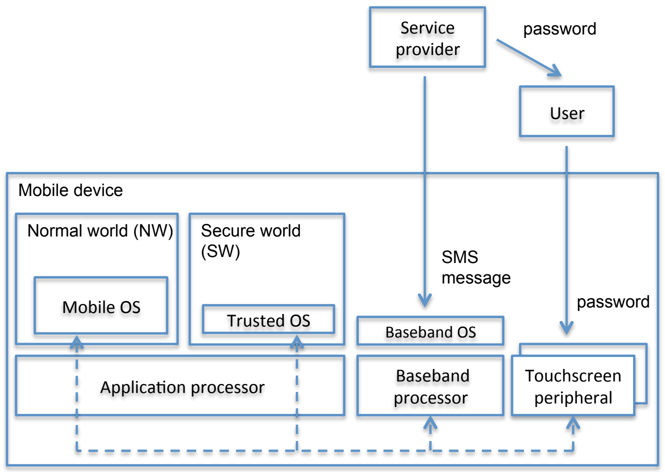


Fig. 11: Commonly suggested user enrollment schemes. Solid arrows illustrate trustworthy communication channels. Dashed arrows illustrate communication channels in which one of the end points can be either the normal world or the secure world.

when the baseband OS receives an SMS message, it notifies the mobile OS, which can read any initialization secret and leak it to the attacker. Assuming a mobile device architecture in which the baseband OS interacts only with the secure world is not feasible, as the mobile OS needs to interact with the baseband for, e.g., phone calls. To overcome this limitations, the baseband-assisted enrollment scheme uses an enhanced baseband OS to achieve secure enrollment.

Enrollment Using Initialization Secrets

Figure 11 illustrates the user enrollment problem when using initialization secrets. Any channel to the secure world can be intercepted by the mobile OS in the normal world, be it the baseband OS, the user-interface, or any other interface such as NFC or Bluetooth. For example, the baseband OS cannot pass the initialization secret received over SMS messages to the application processor because it does not know which processor state (normal world or secure world) is active.

C. Alternative Trusted Execution Environments

In addition to ARM TrustZone, SIM cards are TEEs widely available on smartphones. A SIM card can store secrets and execute small pieces of code (often referred to as *SIM applications*) in isolation from the mobile OS. Therefore, one could argue that the location verification mechanism can be implemented as a SIM application within the SIM card processing environment. This approach has two drawbacks. First, provisioning of SIM applications to a SIM card requires the service provider to negotiate with the network operator that issued that SIM card. Providers who target global applications must negotiate with a large number of network operators separately. Second, in current architectures, SIM cards cannot directly access the device peripherals, such as the GPS unit. When the SIM card needs location information, the application processor must read the coordinates from the GPS unit and provide them to the SIM card. The SIM application has no means of knowing whether these coordinates have been tampered with. Some recent device configurations support a dedicated connection between the SIM card and the device NFC unit [47]. Similar dedicated lines could be added for secure access to the GPS unit. However, targeted hardware changes to allow for specific security services based on SIM applications are costly and hard to justify for widespread adoption.

Some mobile devices are equipped with a slot for SD cards. The latter may be used as a TEE to run code in isolation from the mobile OS or to securely store credentials [48], [49]. However, just like SIM cards, SD cards do not have direct access to the device peripherals, such as the GPS unit, and remote provisioning of applications by third-parties to SD cards is not currently available.

XI. Related Work

Secure payments with location data. Similar to our approach, the authors of [3] propose to mitigate fraud in payments at points of sale, using the phone location as an additional evidence to distinguish between legitimate and fraudulent transactions. In [3], the bank sends a message to the user phone with the details of the transactions (including the location of the merchant and the one of the phone, as provided by the network operator) and asks the user to confirm the payment. This solution requires changes to (i) the GSM infrastructure to provide the user with the current location of his phone, (ii) the points of sale to handle extra messages and additional cryptographic operations, and ultimately (iii) the overall user experience. Furthermore, the protocols in [3] do not account for a compromised mobile OS, nor they address enrollment. The authors do assume an adversary who can intercept any communication channel and require that the bank shares pair-wise keys with the phone, the mobile network operator, and the point of sale.

Recently, MasterCard has proposed to use the location of the card-holder's smartphone to improve fraud detection in payments at points of sale [2]. However, the proposal, neither addresses mobile OS compromise, nor enrollment, nor migration.

Secure payments with other approaches. Securing online payments with multi-factor authentication is the focus of a number of work that leverage hardware and software tokens to generate one-time passwords [4], [50], [51], biometric scanners [52], [53], or simply user-remembered passwords [54]. Financial institutions are deploying systems that leverage modern smartphones to replace traditional payment cards and using NFC-enabled point of sale terminals [38], [55]. Such solutions can only be used at selected stores or face large investments for hardware upgrades at merchants. Other systems that enhance the security of payment card transactions, require the user to confirm the payment through SMS messages or phone calls [37].

Secure Enrollment. Secure enrollment for trusted platforms [56] and TEEs on mobile devices [11] is a challenging research problem. Both deployed systems [4], [5] and academic work [3], [10], [57] have overlooked it or assume a non-compromised OS at enrollment time [14], [46].

Trusted Sensors. Saroiu and Wolman [9] put forward the problem of trustworthy sensor readings on mobile phones.

They propose to leverage virtualization to handle sensors readings and provide signed measurements to applications that run in guest VMs. A similar approach is considered in [58] where the authors consider participatory sensing scenarios and study how to protect user privacy and how to allow for local data processing while providing (TPM) signed sensor readings. Trustworthy participatory sensing applications are also considered in [59], where the authors design and implement a trusted sensing platform. The latter is a board equipped with sensors and a TPM that communicates with the mobile device over Bluetooth and provides signed sensor measurements.

Liu et al. [10] build on top of Credo [60] and TrustZone to propose software abstractions that expose trusted sensor services to mobile applications. In particular, [10] focuses on scenarios where trustworthy reading must be processed locally by applications (e.g., blurring people's faces from a photo taken by the phone's camera) before being sent. Therefore the framework in [10] aims at protecting the integrity of a sensor reading as well as the local code that must process it. The authors provide a sample application scenario where sensor readings account for GPS traces, while data processing applies noise to the aggregated outcome in order to achieve differential privacy [61].

Plug-n-Trust [62] focuses on mHealth scenarios and provides a system to protect integrity and confidentiality of data collected by body-area network of sensors. Their approach leverages a smart card that plugs into a phone's microSD slot and creates a trusted computing environment for collecting and processing data provided by surrounding sensors. Security rests upon tamper resistance of both the smart card and the sensor nodes where encryption/authentication keys are stored.

YouProve [57] addresses scenarios where sensor readings must be modified by local applications for, e.g., privacy issues. The authors of [57] propose a framework to verify that a modified data item preserves the meaning of the original sensor reading. YouProve uses TaintDroid [63] to track the measurement from the moment it is produced by the sensor, until the moment it is uploaded to a service provider. Each modification is tracked, and a summary of all changes is signed using the secret key of the phone's TPM.

To the best of our knowledge, all previous work on trusted sensors focuses on designing or developing a trusted computing environment on a mobile device, to sign sensor readings. However, it does not address secure deployment aspects, including secure enrollment and device migration. To the best of our knowledge, we are the first to propose a complete and deployable solution that allows a number of application scenarios to leverage on-smartphone trusted sensor measurements.

XII. Conclusion

In this paper we proposed a practical solution that adds the required security to recently proposed location-based second-factor authentication mechanisms for payments at points of sale. We have identified the necessary requirements for a deployable solution, including no changes to the user experience and to the deployed infrastructure. We have further proposed two novel enrollment schemes that enable secure enrollment and convenient device migration despite a compromised OS. Through prototype implementations we have shown deployment feasibility of our solution; a location verification operation causes an acceptable delay during a payment transaction and requires only minimal software changes to mobile devices. As future work we plan to integrate the changes we proposed to TrustZone and the baseband OS in a smartphone device, in order to better assess the performance of our solution.

Acknowledgments

The authors would like to thank Ramya Jayaram Masti and Mario Strasser for the useful discussions while writing this paper. This work was partially supported by the Zurich Information Security Center (ZISC). It represents the views of the authors.

References

[1] European Central Bank, "Report on card fraud," July 2012, http://www.ecb.int/pub/pdf/other/cardfraudreport201207en.pdf.

[2] P. Fourez and Mastercard International Inc., "Location controls on payment card transactions," http://patentscope.wipo.int/search/en/WO2011022062, 2011, Patent No. WO/2011/022062.

[3] F. S. Park, C. Gangakhedkar, and P. Traynor, "Leveraging cellular infrastructure to improve fraud prevention," in *Proceedings of the 25th Annual Computer Security Applications Conference*, ser. ACSAC '09, 2009, pp. 350–359.

[4] Barclays, "Mobile PINsentry," http://goo.gl/pcuGo, last access 2013.

[5] Google Inc., "Google 2-Step Verification," http://www.google.com/landing/2step/, last access 2013.

[6] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," in *ACM workshop on Security and privacy in smartphones and mobile devices*, ser. SPSM'11, 2011.

[7] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *IEEE Symposium on Security and Privacy*, ser. SP'12, 2012.

[8] ARM, "Building a Secure System using TrustZone Technology," http://www.arm.com, 2009.

[9] S. Saroiu and A. Wolman, "I am a sensor, and i approve this message," in *Proceedings of ACM International Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2010.

[10] H. Liu, S. Saroiu, A. Wolman, and H. Raj, "Software abstractions for trusted sensors," in *The 10th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2012.

[11] C. Marforio, N. Karapanos, C. Soriente, K. Kostiainen, and S. Capkun, "Secure enrollment and practical migration for mobile trusted execution environments," in *Proceedings of the third ACM workshop on Security and privacy in smartphones and mobile devices*, ser. SPSM'13, 2013.

[12] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes," in *2012 IEEE Symposium on Security and Privacy*, May 2012. [Online]. Available: http://www.cl.cam.ac.uk/~jcb82/doc/BHOS12-IEEESP-quest_to_replace_passwords.pdf

[13] R. Guida, R. Stahl, T. Bunt, G. Secrest, and J. Moorcones, "Deploying and using public key technology: lessons learned in real life," *IEEE Security Privacy*, vol. 2, no. 4, 2004.

[14] A. Czeskis, M. Dietz, T. Kohno, D. S. Wallach, and D. Balfanz, "Strengthening user authentication through opportunistic cryptographic identity assertions," in *Proceedings of the 19th ACM Conference on Computer and Communications Security*, ser. CCS'12, 2012, pp. 404–414.

[15] M. Mannan, B. H. Kim, A. Ganjali, and D. Lie, "Unicorn: two-factor attestation for data security," in *Proceedings of the 18th ACM conference on Computer and communications security*, ser. CCS '11, 2011, pp. 17–28.

[16] K. Kostiainen, E. Reshetova, J.-E. Ekberg, and N. Asokan, "Old, new, borrowed, blue – a perspective on the evolution of mobile platform security architectures," in *ACM conference on Data and application security and privacy*, ser. CODASPY'11, 2011.

[17] K. Kostiainen, J.-E. Ekberg, N. Asokan, and A. Rantala, "On-board credentials with open provisioning," in *International Symposium on Information, Computer, and Communications Security*, ser. ASIACCS'09, 2009.

[18] GlobalPlatform, "Device specifications," available at: http://www.globalplatform.org/specificationsdevice.asp.

[19] osmocomBB, http://bb.osmocom.org, last access 2013.

[20] 3GPP, "3GPP TS 23.040 - Technical realization of the Short Message Service (SMS)," http://www.3gpp.org/ftp/Specs/html-info/23040.htm, last access 2013.

[21] Offspark B.V., "PolarSSL," https://polarssl.org/, last access 2013.

[22] ARM, "ARM Motherboard Express," http://www.arm.com/products/tools/development-boards/versatile-express/motherboard-express.php.

[23] ——, "ARM Coretile Express," http://www.arm.com/products/tools/development-boards/versatile-express/coretile-express.php.

[24] Sierraware, "Open Virtualization - ARM TrustZone and ARM Hypervisor Open Source Software," http://www.openvirtualization.org/.

[25] GlobalPlatform, "GlobalPlatform Device Specifications," http://www.globalplatform.org/specificationsdevice.asp, last access 2013.

[26] Android Development Team, "Android 4.1 APIs - Jelly Bean," http://developer.android.com/about/versions/jelly-bean.html, 2013.

[27] Bouncy Castle Crypto APIs, http://www.bouncycastle.org, last access 2013.

[28] G. Inc., "Google Cloud Messaging for Android," http://developer.android.com/google/gcm/index.html, last access 2013.

[29] CherryPy Team, "CherryPy - A Minimalistic Python Web Framework," http://www.cherrypy.org/, 2013.

[30] SQLite Development Team, "SQLite," http://www.sqlite.org, last access 2013.

[31] Comcetera Ltd., "Number Portability Lookup," http://www.numberportabilitylookup.com/, 2013.

[32] EMV, "Integrated Circuit Card Specifications for Payment Systems, Book 1-4, Version 4.3," 2011. [Online]. Available: http://www.emvco.com/

[33] Mastercard, "Mastercard Developer Zone," 2013. [Online]. Available: https://developer.mastercard.com/

[34] VISA, "Visa Developer Program," 2013. [Online]. Available: https://developer.visa.com/

[35] Telegraph, "EU to end mobile roaming charges next year," June 2013, http://www.telegraph.co.uk/finance/newsbysector/mediatechnologyandtelecoms/telecoms/10119159/EU-to-end-mobile-roaming-charges-next-year.html.

[36] R. Pries, T. Hobfeld, and P. Tran-Gia, "On the suitability of the short message service for emergency warning systems," in *Proceedings of the 63rd Vehicular Technology Conference*, ser. VTC 2006-Spring, vol. 2, 2006, pp. 991–995.

[37] ValidSoft, "ValidPOS," 2013. [Online]. Available: http://www.validsoft.com/

[38] Google Inc., "Google wallet," http://www.google.com/wallet/, last access 2013.

[39] K.-P. Yee, "User interaction design for secure systems," in *International Conference on Information and Communications Security*, ser. ICICS '02, 2002.

[40] Z. Ye, S. W. Smith, and D. Anthony, "Trusted paths for browsers," *ACM Trans. Inf. Syst. Secur.*, vol. 8, no. 2, pp. 153–186, 2005.

[41] M. Selhorst, C. Stüble, F. Feldmann, and U. Gnaida, "Towards a trusted mobile desktop," in *International conference on Trust and trustworthy computing*, ser. TRUST'10, 2010.

[42] ARM, "Securing the system with trustzone ready program," http://www.arm.com/files/pdf/Tech_seminar_TrustZone_v7_PUBLIC.pdf, last access 2013.

[43] S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer, "The emperor's new security indicators," in *IEEE Symposium on Security and Privacy*, ser. SP'07, 2007.

[44] S. Egelman, L. F. Cranor, and J. Hong, "You've been warned: an empirical study of the effectiveness of web browser phishing warnings," in *SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI'08, 2008, pp. 1065–1074.

[45] C. Jackson, D. R. Simon, D. S. Tan, and A. Barth, "An evaluation of extended validation and picture-in-picture phishing attacks," in *International Conference on Financial cryptography and International conference on Usable Security*, ser. FC'07/USEC'07, 2007, pp. 281–293.

[46] K. Kostiainen and N. Asokan, "Credential life cycle management in open credential platforms (short paper)," in *ACM workshop on Scalable trusted computing*, ser. STC'11, 2011, pp. 65–70.

[47] GSM Association, "Requirements for Single Wire Protocol NFC Handsets," http://www.gsma.com/mobilenfc/, 2011.

[48] Giesecke & Devrient GmbH, "G&D Makes Mobile Terminal Devices Even More Secure with New Version of Smart Card in MicroSD Format," http://www.gi-de.com/en/about_g_d/press/press_releases/G%26D-Makes-Mobile-Terminal-Devices-Secure-with-New-MicroSD%E2%84%A2-Card-g3592.jsp, last access 2013.

[49] SD Association, "smartSD Memory Cards," hhttps://www.sdcard.org/developers/overview/ASSD/smartsd/, last access 2013.

[50] PayPal, "PayPal Security Key," https://www.paypal.com/securitykey/.

[51] RSA, "RSA SecurID," http://www.emc.com/security/rsa-securid.htm/.

[52] "United Bankers' Bank Authenticates Customers Online," *Biometric Technology Today*, vol. 12, no. 6, p. 4, 2004.

[53] "Bank of Utah Adopts Keystroke Dynamics," *Biometric Technology Today*, vol. 15, no. 5, p. 5, 2007.

[54] VISA, "Verified by VISA," http://www.visaeurope.com/en/cardholders/verified_by_visa.aspx.

[55] JVL Ventures, LLC, "ISIS," https://www.paywithisis.com/.

[56] B. Parno, "Bootstrapping trust in a "trusted" platform," in *Proceedings of the third Conference on Hot Topics in Security*, ser. HOTSEC'08, 2008, pp. 9:1–9:6.

[57] P. Gilbert, J. Jung, K. Lee, H. Qin, D. Sharkey, A. Sheth, and L. P. Cox, "Youprove: authenticity and fidelity in mobile sensing," in *SenSys*, J. Liu, P. Levis, and K. Römer, Eds. ACM, 2011, pp. 176–189.

[58] P. Gilbert, L. P. Cox, J. Jung, and D. Wetherall, "Toward trustworthy mobile sensing," in *Proceedings of ACM International Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2010.

[59] A. Dua, N. Bulusu, and W. chang Feng, "Towards trustworthy participatory sensing," in *4th USENIX Workshop on Hot Topics in Security (HotSec)*, 2009, pp. 1–6.

[60] H. Raj, D. Robinson, T. Tariq, P. England, S. Saroiu, and A. Wolman, "Credo: Trusted computing for guest vms with a commodity hypervisor," Microsoft Research, Tech. Rep. MSR-TR-2011-130, 2011.

[61] C. Dwork, "Differential privacy," in *ICALP (2)*, ser. Lecture Notes in Computer Science, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds., vol. 4052. Springer, 2006, pp. 1–12.

[62] J. Sorber, M. Shin, R. A. Peterson, and D. Kotz, "Plug-n-trust: practical trusted sensing for mhealth," in *International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys'12, 2012.

[63] W. Enck, P. Gilbert, B. gon Chun, L. P. Cox, J. Jung, P. McDaniel, and A. Sheth, "Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones," in *OSDI*, 2010, pp. 393–407.