A Simple Mobile Plausibly Deniable System Using Image Steganography and Secure Hardware

Lichen Xia
University of Delaware
United States
lxia@udel.edu

Jinghui Liao
Wayne State University
United States
jinghui@wayne.edu

Niusen Chen
Michigan Technological University
United States
niusenc@mtu.edu

Bo Chen
Michigan Technological University
United States
bchen@mtu.edu

Weisong Shi
University of Delaware
United States
weisong@udel.edu

ABSTRACT

Traditional encryption methods cannot defend against coercive attacks in which the adversary captures both the user and the possessed computing device, and forces the user to disclose the decryption keys. Plausibly deniable encryption (PDE) has been designed to defend against this strong coercive attacker. At its core, PDE allows the victim to plausibly deny the very existence of hidden sensitive data and the corresponding decryption keys upon being coerced. Designing an efficient PDE system for a mobile platform, however, is challenging due to various design constraints bound to the mobile systems.

Leveraging image steganography and the built-in hardware security feature of mobile devices, namely TrustZone, we have designed a Simple Mobile Plausibly Deniable Encryption (SMPDE) system which can combat coercive adversaries and, meanwhile, is able to overcome unique design constraints. In our design, the encoding/decoding process of image steganography is bounded together with Arm TrustZone. In this manner, the coercive adversary will be given a decoy key, which can only activate a DUMMY trusted application that will instead sanitize the sensitive information stored hidden in the stego-image upon decoding. On the contrary, the actual user can be given the true key, which can activate the PDE trusted application that can really extract the sensitive information from the stego-image upon decoding. Security analysis and experimental evaluation justify both the security and the efficiency of our design.

CCS CONCEPTS

• **Security and privacy → Mobile platform security**.

KEYWORDS

Plausibly Deniable Encryption, Mobile Devices, TrustZone, Image Steganography

SaT-CPS '24, June 21, 2024, Porto, Portugal

ACM ISBN 979-8-4007-0555-7/24/06.
https://doi.org/10.1145/3643650.3658607

ACM Reference Format:
Lichen Xia, Jinghui Liao, Niusen Chen, Bo Chen, and Weisong Shi. 2024. A Simple Mobile Plausibly Deniable System Using Image Steganography and Secure Hardware. In *Proceedings of the 2024 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems (SaT-CPS '24), June 21, 2024, Porto, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3643650.3658607

1 INTRODUCTION

Mobile devices, such as smartphones, tablets, and wearable devices, have been increasingly utilized to store and process private or even mission-critical information. To protect the confidentiality of those sensitive data, disk encryption is typically incorporated. Traditional encryption methods, however, convert the protection of the data to that of the decryption keys. They remain vulnerable to a strong coercive attack [5], in which the adversary is able to capture the device as well as the owner of the device, and forces the owner to disclose the decryption keys. An illustrative example is that a journalist works in a country of oppression and has captured criminal evidence in his/her smartwatch; later, the journalist is caught at the border and forced to disclose the encrypted criminal evidence. Plausibly deniable encryption (PDE) has been designed to combat such coercive attackers. The PDE encrypts the sensitive data into ciphertext in such a way that, when a true key is used for decryption, the original sensitive data will be revealed; but, when a decoy key is used, another non-sensitive data will be revealed. Upon being coerced, the victim can simply disclose the decoy key, plausibly denying the existence of both the true key and the sensitive data. Therefore, the PDE essentially hides the sensitive data with plausible deniability.

Implementing the PDE concept in the mobile setting is not straightforward, as mobile systems present unique design constraints. First, mobile systems typically use flash memory as external storage, which exhibits a unique hardware nature leading to unique attack avenues towards deniability compromises [14, 20]. Second, the processing of sensitive data typically occurs within an insecure system environment [23], which will unavoidably leave traces of sensitive data within the insecure system. Given the variety of mobile operating systems, eliminating those traces poses a significant challenge. Third, mobile systems typically possess limited computational and storage capacities, and the incorporation of a PDE

system may considerably increase the system's overhead, thereby affecting the system performance as well as the user experience.

This work aims to design a **S**imple **M**obile **PDE** system, SMPDE, which can meet the aforementioned design constraints of mobile systems. At its core, SMPDE "simply" enhances image steganography with deniability support, by utilizing ARM TrustZone, a prevalent hardware security technology found in today's mobile computing devices. Image steganography [31] has been broadly used to embed sensitive data into cover images, generating stego-images which hide the sensitive data. However, the stego-images are vulnerable to image steganalysis [21] and, once the image steganalysis has detected the presence of hidden information embedded within the stego-images, this existence cannot be denied. Therefore, image steganography itself cannot achieve plausible deniability. SMPDE incorporates TrustZone with image steganography to provide this property. Our key idea is that the stego-image will be encrypted with a key bound to the TrustZone, i.e., only the TrustZone can decrypt this encrypted image. In addition, the TrustZone maintains two trusted applications (TA), a PDE TA and a DUMMY TA. The PDE TA will be activated by a true key, while the DUMMY TA will be activated by a decoy key. When the true key is provided, the PDE TA will decrypt the stego-image, extract the sensitive data from the stego-image, and then return the sensitive data. However, when the decoy key is provided, the DUMMY TA will decrypt the stego-image, delete the hidden data from the image, and then return this altered image. Upon being coerced by the adversary, the victim can simply provide the decoy key, and convince the adversary that the encrypted image does not hide sensitive information.

By concealing sensitive data within the images that are considered purely application-layer data, SMPDE can eliminate deniability concerns associated with the complicated flash memory system. In addition, the use of TrustZone can isolate the PDE system in a trusted world, and avoid leaking sensitive data to the untrusted environment, regardless of the mobile operating systems being used. Furthermore, the design is "thin", by merely processing the image steganography in a TrustZone secure world, which can avoid imposing too much additional overhead on the mobile system. Such a simple PDE system design does come with some limitations which will be discussed in Sec. 9.

Contributions. Our primary contributions are summarized in the following:

- We have introduced SMPDE,a simple mobile PDE system design. SMPDE is able to resolve various design constraints unique to mobile systems, by smartly integrating image steganography and Arm TrustZone.
- We have developed a prototype of SMPDE in a Raspberry Pi 3 Model B+ with ARM TrustZone enabled. Experimental results justify the efficiency of SMPDE.

2 BACKGROUND

2.1 Image Steganography

Steganography is a technique which can be used to hide sensitive information within ordinary, non-secret objects like images, audio, videos, texts, etc. in an imperceptible manner [17]. The carrier objects themselves do not attract suspicion and the hidden information is not discernible through direct examination. This provides a layer of obscurity and deniability for covert communication. Image steganography specifically refers to the steganography technique which hides information in image files. It usually operates in two domains, the spatial domain and the transform domain. A typical approach in the spatial domain is the least significant bit (LSB) technique. This approach will embed sensitive data into the least significant bit of each byte in an RGB image's pixel channels. Each pixel in an RGB image includes four channels: alpha (A), red (R), green (G), and blue (B), with each channel being represented by one byte. The alpha channel determines the transparency, while the red, green, and blue channels represent the respective color values. The least significant bit of each channel has minimal impact on the overall pixel value, making it an ideal candidate for hiding secret data without visible modifications to the image. In the transform domain, the most frequently used methods are the discrete cosine transform (DCT) [3] and the discrete wavelet transform (DWT) [29]. These techniques transfer the image from the spatial domain to the frequency domain. Subsequently, the sensitive data are embedded into the high or middle-frequency components, as those components have minimal visual impact on the original image. Compared to performing steganography in the transform domain, the LSB technique is much easier to be implemented. Other advanced techniques have been developed recently for image steganography, such as neural networks [26], generative adversarial networks [25], etc.

Steganography is vulnerable to steganalysis [22], a practice which aims to defeat steganography by detecting the presence of hidden data within digital mediums. It involves examining digital files to detect hidden information embedded within them, via a variety of techniques from simple statistical analysis [21] to more complex machine learning algorithms [32].

2.2 ARM TrustZone

ARM TrustZone represents a set of hardware security extensions introduced by ARM since 2004 to improve security in modern computing systems against an evolving landscape of threats [1]. TrustZone enables the creation of a secure, isolated environment on the main system-on-chip (SoC) by virtually partitioning resources into a *secure world* and a *normal world*. The normal world corresponds to the traditional rich operating system (OS) environment used to support general-purpose applications and functionality. On the contrary, the secure world enables the execution of security-sensitive code in an isolated trusted execution environment (TEE), along with access to peripherals and memory exclusively assigned to it. By design, code running in the normal world cannot have access to resources that belong exclusively to the secure world. This isolation prevents potentially vulnerable rich OSes and applications from compromising the security of trusted code running in the TEE. The normal world can only invoke TEE services through carefully controlled communication channels. Sensitive data processed within the TEE is also secured in the protected memory reserved for the secure world.

The TrustZone hardware security feature has seen widespread adoption across mobile, embedded, and IoT segments, providing robust security for tasks such as mobile payments, biometric authentication, digital rights management and secure boot [4]. By

leveraging an isolated secure environment, TrustZone-based techniques can deliver advanced security properties like confidentiality, integrity, and anti-cloning assurances.

3 MODELS AND ASSUMPTIONS

Threat model. We consider an adversary which is computationally bounded, i.e., the adversary cannot have access to unlimited computational power. The adversary can capture a victim user and his/her computing device multiple times at different intervals, known as a multi-snapshot adversary. The adversary can access the device's external storage and internal memory and, may coerce the victim into revealing decryption keys. The access to the external storage can be extended to various layers of the storage stack, including the application, the file system, the block device, and the raw flash memory.

User assumptions. We assume that the user will not intentionally disclose the existence of the PDE system or install potentially adversary-controlled malicious applications. The user is expected to share only the decoy key with the adversary and should avoid using publicly accessible images as cover images to conceal sensitive data. In addition, the user will process hidden sensitive data only in the PDE to address deniability concerns that may arise from operating on an untrusted system. Furthermore, the user should not use the PDE system to process sensitive data in the presence of the adversary. Last, the user should use an image steganography technique which is sufficiently secure.

Device assumptions. We consider an embedded device which is equipped with an ARM processor that features TrustZone capability. The TrustZone is assumed to be secure, and its ability to safeguard trusted applications (TA) codes is in line with other trusted execution environment (TEE)-based solutions. Attacks against TrustZone itself are not the focus of this work. In addition, we assume that the TrustZone hardware can provide or protect a master key that is not accessible to the adversary. Furthermore, the operating system, kernel, and bootloader are assumed to be free of malware. This is ensured through regular antivirus scans and caution about untrusted apps. Lastly, we assume the device is equipped with secure I/O [19] which can be used to input/output the data that need to be kept secret from the adversary, e.g., the hidden sensitive data, the cover images.

Adversary assumptions. The adversary cannot obtain the internal processing logic of the PDE system through reverse-engineering the PDE binary file [20]. In addition, the adversary is not able to physically access the on-chip memory and will cease coercion once convinced that the decryption key has been disclosed.

4 SMPDE DESIGN

To plausibly deny the existence of sensitive data, SMPDE encodes the sensitive data into a cover image using image steganography, generating a stego-image. Subsequently, this stego-image will be encrypted in the TrustZone secure world using a key solely accessible to the TrustZone TAs, and the encrypted stego-image will be stored in the external storage. Upon retrieval, the encrypted stego-image is read and can only be decrypted by TrustZone TAs in two cases: 1) If a true key is used, a PDE TA is invoked which will decrypt the encrypted stego-image, extract the hidden sensitive data from the stego-image, and return the hidden sensitive data. 2) If a decoy key is used, a DUMMY TA is invoked which will decrypt the encrypted stego-image, and return the stego-image after having removing the hidden sensitive data from it. Therefore, upon being coerced, the victim can simply disclose the decoy key to avoid being tortured. Using the decoy key, the adversary cannot identify the existence of the hidden sensitive data. However, the user can extract the hidden sensitive data using the true key.

To prevent the adversary from compromising deniability by playing with TrustZone using the decoy key, we employ a one-time key (OTK) mechanism for encryption. In other words, the TrustZone TAs (including both the DUMMY TA and the PDE TA) always use a different one-time key upon encryption. This can ensure that the identical plaintext will always be encrypted into different ciphertext in a different encryption round.

4.1 Overview

An overview of SMPDE is illustrated in Figure 1. There are two TAs running in the TrustZone secure world: a PDE TA (i.e., a PDE application running in the trusted OS) that handles PDE operations, and a DUMMY TA (i.e., a DUMMY application running in the trusted OS) that processes decoy operations. When a user seeks access to the PDE system, he/she begins this procedure through a client entry application running in the normal world, sending data and commands to the entry application running in the secure world, which acts as a gateway to the TrustZone TAs. This TrustZone entry application asks for a key from the user. If the user provides the true key, the entry application loads the PDE TA, facilitating the execution of the PDE operations (Sec. 4.2.2). On the contrary, if the user enters a decoy key, it results in activating the dummy TA designed to execute operations that appear superficially legitimate (Sec. 4.2.3). The workflow in Figure 1 is described in the following:

(1) The user enters the non-secure system, initiating the client entry point application running in the normal world (aka, rich execution environment, REE) which encompasses modules for TEE communication.
(2) The non-secure client entry application retrieves files through the normal world file system (note that for PDE operations, both the cover image and the sensitive data should be entered from the secure I/O rather than being read from the REE file system to avoid being compromised).
(3) Commands and files from the user are conveyed to the TEE, where the TrustZone entry application assesses the user's directives to ascertain subsequent handling of the data.
(4) Depending on the input key value, the TrustZone entry application will activate either the PDE TA which hides/extracts the sensitive data, or the dummy TA which executes the decoy operations.

4.2 Design Details

We present the design details of the SMPDE protocol. A high-level description of the protocol is shown in Algorithm 1, while each detailed operation is elaborated in Algorithm 2.

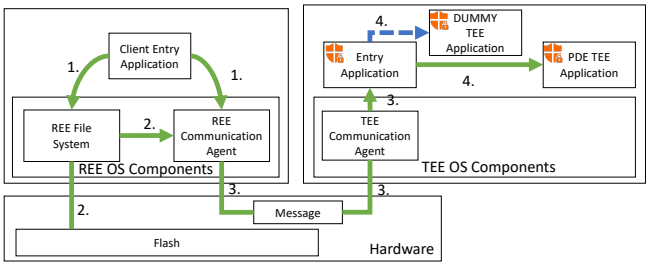

Figure 1: An overview of SMPDE. The user operations are marked with arrows. The blue dash line indicates the dummy TA triggered by the decoy key.

4.2.1 Notations and Primitives. We denote the secret data to be hidden as m, the cover image used to hide m as C, and the stego-image that contains m is denoted as C_m. MK is the TrustZone master key which is secret and only accessible in the TrustZone secure world. The true key is $\mathcal{K}_t$ while the decoy key is $\mathcal{K}_d$.

Key Generation Function (KGF). Upon each encryption, we need to derive a new one-time key (OTK) from the TrustZone master key (MK). In addition, this ephemeral key should be able to be re-generated upon decryption. We define our KGF function as follows:

$$\text{KGF}(MK, cont) \rightarrow k \tag{1}$$

where k is the derived OTK key, and $cont$ is the contextual information which can be extracted from the image file. We use both the image ID and the timestamp as $cont$. Note that the image ID for the cover image, and the stego-image after embedding sensitive data, and the stego-image after removing the sensitive data remain the same. This ID is committed to the metadata area of the image and will not be affected by encryption/decryption. In addition, each time upon encrypting an image, a new timestamp should be generated and committed to the metadata area of the image. The KGF function can be implemented using crypto primitives like HMAC-SHA2.

Encryption/Decryption. The encryption should be performed using a one-time key (OTK). If m is the plaintext, k is the OTK key, and the encrypted message is $E(m)$, then the $Encrypt$ can be represented as:

$$E(m) = \textsc{Encrypt}(m, k) \tag{2}$$

and the $Decrypt$ function is:

$$m = \textsc{Decrypt}(E(m), k) \tag{3}$$

Note that the $Encrypt/Decrypt$ functions can be instantiated using secure symmetric encryption algorithms like AES.

Sensitive data embedding/extracting. Image steganography involves embedding a hidden message m within a cover image C to produce a stego-image C_m. The embedding function, denoted as $Embed$, that performs the embedding operations can be represented as:

$$\textsc{Embed}(C, m) \rightarrow C_m \tag{4}$$

The extraction function, denoted as $Extract$, that extracts m from C_m can be represented as:

$$\textsc{Extract}(C_m) \rightarrow m \tag{5}$$

Note that the $Embed/Extract$ functions can be implemented using various image steganography techniques and their reverse operations (Sec. 2.1).

4.2.2 SMPDE Protocol with True Key $\mathcal{K}_t$. When the user enters the true key, the entry application loads the PDE TA, allowing the user to interact with the PDE TA to process hidden sensitive data. As shown in Algorithm 1, the client entry application accepts the true key $\mathcal{K}_t$ and passes it to the TrustZone to activate the TrustZone entry application. Then the TrustZone entry application process is as follows:

Data Encrypting and Saving with $\mathcal{K}_t$. This process is depicted in Algorithm 2 as function **TAUserStore**. It is elaborated as follows:

(1) **PDE TA Activation:** The system first activates the PDE TA within TrustZone using $\mathcal{K}_t$. This secure environment within TrustZone is crucial as it isolates the encryption process from the rest of the device, mitigating the risks associated with potential security breaches.
(2) **Image Steganography:** m is processed through image steganography, by which it is embedded within a cover image C, generating a stego-image C_m. Note that both m and C should not be obtained from the REE, which is insecure and may cause deniability compromises. Instead, both m and C should be entered into the secure world from the secure I/O.
(3) **OTK Generation:** To ensure that the ciphertext is always different upon a new encryption, SMPDE derives a new OTK k from the TrustZone master key MK, the image ID as well as a new timestamp.
(4) **Encryption:** The stego-image C_m is encrypted using k, obtaining $E(C_m)$. This encryption ensures that even if the data is accessed, its true nature remains protected without accessing the master key. The data can only be decrypted by PDE TA or DUMMY TA, not even by the device owner.
(5) **Storage:** The storage of the encrypted data $E(C_m)$ is processed by the REE file system. As the data are encrypted, the non-secure system cannot know their content.

Data Decrypting and Retrieval with $\mathcal{K}_t$. Retrieving the sensitive data involves a reverse process. This process is depicted in Algorithm 2 as a function **TAUserRetrieve**. It is elaborated as follows:

(1) **Decryption:** PDE TA reads the encrypted data $E(C_m)$ from the non-secure file system. It then re-generates the OTK k. Last, it decrypts the data $E(C_m)$ using k, obtaining C_m with.
(2) **Sensitive data extraction:** Once decrypted, the PDE TA extracts the sensitive data m from C_m using the reverse operation of image steganography, and returns the sensitive data m to the user.

4.2.3 SMPDE Protocol with Decoy Key $\mathcal{K}_d$. When the user enters the decoy key $\mathcal{K}_d$, the TruztZone entry application loads the DUMMY TA, allowing the user to interact with the DUMMY TA to process input images, triggering a different set of procedures to maintain the plausible deniability of the system.

Data Encrypting and Saving with $\mathcal{K}_d$. The protocol's ingenious design ensures that the use of $\mathcal{K}_d$ does not reveal the existence of sensitive data. This process is represented in Algorithm 2 as a function **TAAversaryStore**. The function is elaborated as follows:

Algorithm 1 The SMPDE Protocol for User and Adversary Operations.

Require: User input key $\mathcal{K}_i$ where $i \in \{t, d\}$
Ensure: Execution of the appropriate operations based on $\mathcal{K}_i$
function ClientEntryApplication
  $\mathcal{K}_i \leftarrow$ ReceiveInputKey ▷ Receive key from the user
  Invoke the TrustZone entry application ▷ Enter the TrustZone by calling the entry application
  if $\mathcal{K}_i = \mathcal{K}_t$ **then** ▷ Check if input key is the true key ▷ Call methods designed for the legitimate user
    PDE-TA(UserOperations) where $UserOperations \in \{TAUserStore, TAUserRetrieve\}$
  else ▷ Call methods designed for the adversary
    DUMMY-TA(AdversaryOperations) where $AdversaryOperations \in \{TAAversaryStore, TAAversaryRetrieve\}$
  end if
end function

Algorithm 2 Detailed Operations in SMPDE.

function GenerateOTK(MK, image)
  $k \leftarrow$ KGF(MK, $image\ ID||time\ stamp$)
  return k
end function

function TAUserStore(m, C, MK) ▷ PDE-TA
  Obtain system time, update C with new timestamp
  $k \leftarrow$ GenerateOTK(MK, C) ▷ Generate the OTK key
  $C_m \leftarrow$ Embed(C, m) ▷ Embed sensitive data
  $E(C_m) \leftarrow$ Encrypt(C_m, k) ▷ Encrypt stego-image for user
  Store $E(C_m)$ to external storage via REE
end function

function TAUserRetrieve(MK) ▷ PDE-TA
  Read $E(C_m)$ from external storage via REE
  $k \leftarrow$ GenerateOTK($MK, E(C_m)$) ▷ Re-generate OTK key
  $C_m \leftarrow$ Decrypt($E(C_m), k$) ▷ Decrypt stego-image
  $m \leftarrow$ Extract(C_m) ▷ Extract sensitive data
  return m
end function

function TAAversaryStore(C', MK) ▷ DUMMY-TA
  Obtain system time, update C' with new timestamp
  $k \leftarrow$ GenerateOTK(MK, C') ▷ Generate the OTK key
  $E(C') \leftarrow$ Encrypt(C', k) ▷ Directly encrypt adversary's image
  Store $E(C')$ to external storage via REE
end function

function TAAversaryRetrieve(MK) ▷ DUMMY-TA
  Read $E(C_m)$ from external storage via REE
  $k \leftarrow$ GenerateOTK($MK, E(C_m)$) ▷ Re-generate OTK key
  $C_m \leftarrow$ Decrypt($E(C_m), k$) ▷ Decrypt image for adversary
  $C' \leftarrow$ *sanitize sensitive data from C_m*
  return C' ▷ Return sanitized image to adversary
end function

(1) **DUMMY TA Activation:** TrustZone's secure environment is also utilized, activated with $\mathcal{K}_d$. This DUMMY TA ensures that the operations carried out do not compromise the hidden data.
(2) **OTK Generation:** The DUMMY TA generates a new OTK key k from the TrustZone master key MK, the image ID as well as a new time stamp.
(3) **Encryption:** The input image C' is then encrypted using k, obtaining $E(C')$.
(4) **Storage:** The storage of the encrypted data $E(C')$ is processed by the REE file system.

Data Decrypting and Retrieval with $\mathcal{K}_d$. The protocol's design ensures that both genuine and decoy operations are indistinguishable from each other from external observers, thus preserving the user's ability to deny the existence of sensitive data. This process is depicted in Algorithm 2 as a function **TAAversaryRetrieve**. It is elaborated as follows:

(1) **Decryption:** The DUMMY TA receives the encrypted data $E(C_m)$ from the REE file system. It first re-generates the OTK k. It then decrypts the $E(C_m)$, obtaining C_m using k.
(2) **Sanitizing sensitive data:** Once decrypted, the DUMMY TA performs a reverse operation of image steganography, locating m from C_m and replacing m with randomness, generating C'. This is essentially equal to sanitizing m from the stego-image before returning it to the adversary.

5 SECURITY ANALYSIS

SMPDE achieves plausible deniability by creating two TAs in TrustZone, a PDE TA and a DUMMY TA, and hiding sensitive data into a cover image through image steganography and TrustZone-bounded encryption. Note that having TAs running itself does not raise any deniability issues as TrustZone TAs are popular for secure computing applications in today's mobile devices. The encrypted stego-image that hides sensitive data cannot be differentiated from a random image that does not contain sensitive data without decryption. We show in the following that SMPDE can ensure plausible deniability.

When the adversary captures both the device and the device owner, it coerces the owner and the owner will disclose the decoy key to avoid further coercion. We consider three cases in which the adversary tries to compromise PDE:

Case 1: The adversary performs forensic analysis over the memory and the external storage. For the memory, the PDE operations are performed in the TrustZone secure world and, all the sensitive data stay isolated in a secure memory region which is inaccessible to the untrusted operating system; in addition, the sensitive data will

be entered into the system through secure I/O and, their traces will not appear in the untrusted operating system; furthermore, the adversary is assumed to be unable to physically access the on-chip memory of the ARM chip (Section 3). Therefore, we can conclude that the adversary is not able to obtain any footprint of the sensitive data in the system memory over time. For the external storage, the stego-images are stored encrypted in the external storage and, the adversary could not tell whether a stego-image contains sensitive data or not before decrypting it. However, the decryption would not be possible without the help of TrustZone as the encryption is bound to the TrustZone (this will be further discussed in Case 2). In conclusion, the adversary cannot compromise the deniability by simply utilizing the multiple snapshots obtained from the memory and the external storage.

Case 2: The adversary obtains the decoy key after coercing the victim and invokes the DUMMY TA, and through the DUMMY TA, it decrypts the encrypted stego-image $E(C_m)$ and performs steganalysis over it, hoping to compromise deniability. However, the internal processing of the DUMMY TA is unknown to the adversary and, taking advantage of this, the DUMMY TA decrypts $E(C_m)$ correctly; however, before directly returning the plaintext C_m to the adversary, it will first remove the sensitive data m from C_m, generating C' which is then returned to the adversary. Clearly, the adversary is not able to learn anything about m from C'. In addition, the adversary will not be able to notice that C' is a modified version of the original cover image C, considering that 1) the adversary does not have access to C, and 2) the size of C' is equal to that of C and $E(C_m)$, and 3) the image steganography technique is sufficiently secure.

Case 3: The adversary plays with the TrustZone. Specifically, the adversary obtains the decoy key and activates the DUMMY TA to decrypt the encrypted stego-image $E(C_m)$. After having obtained the plaintext C', it encrypts C' by activating the TrustZone again using the decoy key, obtaining the ciphertext $E(C')$, and compares $E(C')$ with $E(C_m)$. This however, will not lead to the compromise of deniability, because: 1) both the PDE TA and the DUMMY TA in SMPDE adopt an "ephemeral" encryption mechanism by which the identical plaintext will always be encrypted into different ciphertext when a new encryption process is invoked, and 2) the size of $E(C')$ is always equal to the size of C' which is equal to that of C and $E(C_m)$.

6 IMPLEMENTATION

SMPDE was implemented using OP-TEE (Open Portable Trusted Execution Environment) [27], an open-source TEE that utilizes ARM TrustZone technology.

Encryption and Image Steganography. For encryption, we utilize AES symmetric encryption in CBC mode, with a 256-bit key. For image steganography, we use LSB[1], a lightweight image steganography technique.

OP-TEE TAs and PTAs. Within the OP-TEE architecture, programs are segregated into two privilege layers: the user level, where trusted applications (TAs) function, and the kernel level, which is designated for pseudo-trusted applications (PTAs). The PDE ingress program is structured as a TA operating at the user level, while its core logic is implemented as a PTA. Additionally, a dummy PTA is established to handle the logic related to dummy keys. TAs are typically called by client applications in the normal world through OP-TEE's client API (`libteec`), starting with `TEEC_OpenSession` to initiate a session, followed by `TEEC_InvokeCommand` to transmit commands. PTAs, being static components within the kernel, can be invoked by a TA or another PTA using `TEE_InvokeTACommand`. Encryption and decryption play a crucial role in data writing or reading, and OP-TEE provides a comprehensive range of cryptographic APIs to support these operations. This set of APIs is flexible enough to accommodate the AES encryption algorithm.

To support the PDE logic layer, we adapt the OP-TEE OS to enable the execution of PDE-related operations within TrustZone TAs. We introduce new secure system calls (syscalls) for creating, accessing, and managing PDE-protected data. These calls establish a hidden mode within the TEE, allowing users to securely store and process sensitive data without leaving traces in the REE. Moreover, we implement a two-layer approach [24] within the TEE, comprising a dummy layer as an entry point to the actual PDE layer. This design enhances usability and user experience while maintaining security. The dummy layer enables users to input decoy passwords, granting access to non-sensitive data and concealing the actual PDE-protected data. In the SMPDE system, the encryption of TAs is accomplished using OP-TEE's REE filesystem TA mechanism. Each TA is represented by an ELF file that is signed and, if needed, encrypted. PTAs undergo a similar encryption and decryption process as TAs. However, given their closer integration with core services, the encryption and decryption mechanisms for PTAs may require additional security measures. However, the fundamental approach remains consistent with that of TAs.

7 EXPERIMENTAL EVALUATION

7.1 Testbed

A proof of concept prototype of SMPDE was implemented and deployed on a Raspberry Pi 3 Model B development board. This embedded board has a Broadcom BCM2837 System-on-Chip (SoC) featuring a 64-bit CPU architecture, which is equipped with a 1.2 GHz quad-core ARM Cortex A53 (ARMv8) processor. The SoC has the built-in support for ARM TrustZone. The OP-TEE we used is version 3.20, and the non-secure software realm is anchored in the Linux kernel with version rpi3-optee-5.17 [16]. The board is equipped with 1GB LPDDR2 SDRAM. For external storage, we used a 32GB SanDisk high capacity micro SD Card.

7.2 Performance

The performance of SMPDE is shown in Table 1, which captures the computational time needed for hiding/extracting sensitive data into/from the cover image, encrypting/decrypting the cover image, and reading/writing the SD card. The results were measured by varying the sizes of both the cover image and the hidden sensitive data.

Hiding and Extracting Sensitive Data. The time needed for hiding sensitive data in cover images ranges from 13,882 μs for a 192x144 cover image to 704,463 μs for a 640x360 cover image.

[1]We used LSB for image steganography for a proof-of-concept implementation. LSB, though simple and efficient, is not a strong image steganography technique and may be vulnerable to image steganalysis [34].

Table 1: Performance of SMPDE in various operations when the sizes of cover images vary. The size of the hidden sensitive data is $\frac{1}{8}$ of that of the cover image.

	192x144	256x144	320x240	352x240	352x288	512x288	480x360	568x320	640x360
Hide sensitive data (μs)	13,882	39,350	73,744	138,043	208,627	305,947	416,452	556,141	704,463
Extract sensitive data (μs)	20,869	53,715	96,994	172,665	255,397	368,100	491,867	650,493	819,026
Encrypt/Decrypt cover image (μs)	18,258	35,093	70,208	108,703	154,987	214,518	293,526	376,431	481,618
Write stego-image to SD (μs)	24,782	30,987	59,914	65,896	77,867	99,935	131,240	138,401	173,842
Read stego-image from SD (μs)	30,865	38,276	70,142	76,225	89,754	113,045	146,954	153,858	192,718

Similarly, the time for extracting sensitive data from cover images shows an increase from 20,869 μs to 819,026 μs when the cover image sizes vary. Note that the size of the sensitive data is always $\frac{1}{8}$ of that of the cover image, and therefore, when the size of the cover image increases, the size of the sensitive data will also increase. To hide/extract sensitive data, the LSB technique needs to operate over a certain number of bytes in the cover image, which is linear to the size of the sensitive data. Therefore, the computation needed is approximately linear to the size of the hidden sensitive data, which is linear to the size of the cover image.

Encrypting/Decrypting Stego-images. The time needed for encrypting/decrypting the stego-image increases when the size of the cover image increases, starting at 18,258 μs and reaching up to 481,618 μs. Note that the size of the stego-image is always equal to that of the corresponding cover image. The encryption/decryption time exhibits an approximately (but not exactly) linear growth when the image size grows, which is consistent with the linear complexity of the AES CBC mode.

SD Card Read/Write. For the *SD Write* operation, we observe that the time needed increases from 24,782 μs for the smallest image to 173,842 μs for the largest one, growing approximately linearly. The *SD Read* operation also exhibits a similar pattern, ranging from 30,865 μs to 192,718 μs. It is reasonable that handling a larger image via the SD card takes more time because it requires proportionally more resources and involves retrieving or storing proportionally more data.

8 RELATED WORK

The application-layer PDE systems. MobiWear [13] hides sensitive data in images using steganography and, once the victim is coerced, the hidden information is denied as the watermark embedded in the image, though the actual sensitive data is hidden inside the watermark. However, MobiWear is vulnerable to steganalysis on the watermark images. In addition, traces of hidden data will be present in the insecure memory and may be observed by the adversary. HiPDS [12] is another application-layer PDE system which encoded the sensitive data into the non-sensitive cover data by leveraging Chameleon hash. However, the Chameleon hash is expensive, leading to a significant decrease in throughput [12] and is hence not suitable to be used for low-power mobile computing devices. In contrast, SMPDE is a lightweight PDE system design that fits resource-limited mobile systems and can resist again steganalysis on both external storage and memory.

The block-layer PDE systems. Multiple existing PDE systems [6, 7, 9, 18, 30, 33] have integrated hidden volumes at the block layer and, the hidden volumes can be used to store hidden sensitive data. A common idea is, that each hidden volume is encrypted by a true key and placed stealthily at a secret offset of a public volume encrypted by a decoy key. Upon being coerced, the victim can disclose the decoy key. The adversary can then decrypt the public volume but remains unaware of the hidden volumes concealed among the randomness filled initially. All of them suffer from two limitations. First, they suffer from deniability compromises on the lower-layer flash memory [14]. Second, they suffer from deniability compromises in the insecure memory. SMPDE can address both limitations by 1) hiding sensitive data in the images of the application layer (this will not create any traces at the underlying flash memory), and 2) utilizing TrustZone to protect hidden sensitive data from being leaked to the insecure memory.

The flash memory-layer PDE systems. There were a few mobile PDE systems designed to accommodate the unique nature of flash memory hardware equipped with mobile devices. They have been integrated into either the flash translation layer (FTL) [11, 15, 20, 23, 24] or the flash-specific file systems [10, 28]. A major limitation for all of them is that the PDE functionality is strongly coupled with the underlying flash memory systems, making them hard to deploy. On the contrary, SMPDE is an application-layer PDE system which does not require modifying the underlying flash memory systems and can be deployed easily.

9 DISCUSSION

Applications of SMPDE in the CPS/IoT realm. The amount of critical and sensitive data being created, stored, and processed by cyber-physical systems (CPS) and the Internet of Things (IoT) has been rising. Protecting the confidentiality of such data has become of utmost importance, particularly in situations involving wearable gadgets and other IoT devices. These gadgets frequently manage sensitive data that, if breached, could have significant repercussions on user privacy and system stability. Nowadays, the adversary has become extremely sophisticated and may employ various methods to breach the confidentiality of critical data within the CPS/IoT environments (e.g., the mission-critical data from the smart grids, industry control systems, and autopilot systems, the personally private data from the healthcare monitoring systems, home robotics systems, and self-driving cars). Such methods include rubber-hose cryptanalysis [2] or other strategies that could potentially put the lives of victims at risk. Different from traditional encryption methods, SMPDE could add an extra layer of protection against coercive threats, protecting the data in critical CPS/IoT scenarios.

About embedding/extracting sensitive data. To facilitate extraction of the hidden sensitive data, the locations of the hidden data in the cover image should be kept track of. This can be realized by adding to the cover image "extra information" which can keep track of the embedding locations. This "extra information" should

be also sanitized after the DUMMY TA decrypts the stego-image and sanitizes the hidden data.

Extended memory protection with Secure IO. The sensitive data will be processed within TrustZone so that the adversary cannot have access to it in the secure memory. However, the transfer of data from an unsecured system to TrustZone can result in traces of sensitive data being left in the insecure memory which will lead to the deniability compromise. To mitigate this issue, secure I/O [19] needs to be leveraged to protect peripherals that are used to input/output sensitive data (including cover images which need to be kept secret from the adversaries for deniability assurance). When the peripherals are configured to operate in the secure mode, both the peripherals and their DMZ gain the protection of TrustZone, and any input received by the peripherals is sent directly to TrustZone. We will further investigate the integration of secure I/O and SMPDE in our future work.

Limitations of SMPDE. SMPDE is a "simple" PDE system design at the cost of a few accompanied limitations. First, it relies on the assumption that the adversary does not have access to the original cover images. Otherwise, the adversary can simply compare the original cover images with the images output by the DUMMY TA and notice that "some special data" have been removed from the cover images, compromising the deniability. Second, each image can only conceal a limited amount of sensitive data as otherwise, the image quality of the cover image will be affected significantly, raising a red flag. For example, for LSB, every 8 bits can be used to hide at most 1 bit, and a 1MB image can at most hide 0.12MB sensitive data. The rate is much lower than the hidden volume technique [8, 20] which can hide as much as 0.5MB per 1MB of public data.

10 CONCLUSION

In this work, we have developed SMPDE, a simple PDE system design tailored for mobile hardware. By integrating ARM TrustZone and image steganography, SMPDE can resist a coercive adversary which can have access to both the external storage and the memory of a victim device at multiple checkpoints over time. Security analysis and experimental evaluation using a real-world prototype implementation demonstrate that SMPDE can achieve plausible deniability while imposing a reasonable overhead on mobile platforms.

ACKNOWLEDGMENTS

This work was supported by US National Science Foundation under grant number CNS-2313139, CNS-1928331 and CNS-1928349. Niusen Chen and Bo Chen were also supported by US National Science Foundation under grant number CNS-2225424.

REFERENCES

[1] 2009. ARM TrustZone Technology. https://developer.arm.com/ip-products/security-ip/trustzone
[2] 2022. The Best Defense Against Rubber-Hose Cryptanalysis. https://pluralistic.net/2022/03/27/the-best-defense-against-rubber-hose-cryptanalysis/
[3] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. 1974. Discrete cosine transform. *IEEE transactions on Computers* 100, 1 (1974), 90–93.
[4] ARM. 2023. Silicon IP Security. https://www.arm.com/products/silicon-ip-security. Accessed: 2024-02-25.
[5] Hristo Bojinov, Daniel Sanchez, Paul Reber, Dan Boneh, and Patrick Lincoln. 2014. Neuroscience meets cryptography: Crypto primitives secure against rubber hose attacks. *Commun. ACM* 57, 5 (2014), 110–118.
[6] Bing Chang, Yao Cheng, Bo Chen, Fengwei Zhang, Wen-Tao Zhu, Yingjiu Li, and Zhan Wang. 2018. User-friendly deniable storage for mobile devices. *computers & security* 72 (2018), 163–174.
[7] Bing Chang, Zhan Wang, Bo Chen, and Fengwei Zhang. 2015. Mobipluto: File system friendly deniable storage for mobile devices. In *Proceedings of the 31st Annual Computer Security Applications Conference*. ACM, 381–390.
[8] Bing Chang, Zhan Wang, Bo Chen, and Fengwei Zhang. 2015. Mobipluto: File system friendly deniable storage for mobile devices. In *Proceedings of the 31st annual computer security applications conference*. 381–390.
[9] Bing Chang, Fengwei Zhang, Bo Chen, Yingjiu Li, Wen-Tao Zhu, Yangguang Tian, Zhan Wang, and Albert Ching. 2018. Mobiceal: Towards secure and practical plausibly deniable encryption on mobile devices. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 454–465.
[10] Chen Chen, Anrin Chakraborti, and Radu Sion. 2020. INFUSE: Invisible plausibly-deniable file system for NAND flash. *Proc. Priv. Enhancing Technol.* 2020, 4 (2020), 239–254.
[11] Chen Chen, Anrin Chakraborti, and Radu Sion. 2021. PEARL: Plausibly Deniable Flash Translation Layer using WOM coding. In *The 30th Usenix Security Symposium*.
[12] Niusen Chen and Bo Chen. 2023. HiPDS: A Storage Hardware-independent Plausibly Deniable Storage System. *IEEE Transactions on Information Forensics and Security* (2023).
[13] Niusen Chen, Bo Chen, and Weisong Shi. 2021. MobiWear: A Plausibly Deniable Encryption System for Wearable Mobile Devices. In *EAI International Conference on Applied Cryptography in Computer and Communications*. Springer, 138–154.
[14] Niusen Chen, Bo Chen, and Weisong Shi. 2022. The block-based mobile pde systems are not secure-experimental attacks. In *EAI International Conference on Applied Cryptography in Computer and Communications*. Springer, 139–152.
[15] Niusen Chen, Bo Chen, and Weisong Shi. 2022. A Cross-layer Plausibly Deniable Encryption System for Mobile Devices. In *International Conference on Security and Privacy in Communication Systems*. Springer, 150–169.
[16] OP-TEE Contributors. 2021. OP-TEE/manifest. https://github.com/OP-TEE/manifest GitHub repository.
[17] John Doe and Jane Smith. 2021. Fundamentals of Steganography: Hiding Information in Plain Sight. *Journal of Cybersecurity and Digital Forensics* 15, 3 (2021), 105–120. https://doi.org/10.1000/jcdf.2021.15.3.105
[18] Wendi Feng, Chuanchang Liu, Zehua Guo, Thar Baker, Gang Wang, Meng Wang, Bo Cheng, and Junliang Chen. 2020. MobiGyges: A mobile hidden volume for preventing data loss, improving storage utilization, and avoiding device reboot. *Future Generation Computer Systems* (2020).
[19] Seung-Kyun Han and Jinsoo Jang. 2023. MyTEE: Own the Trusted Execution Environment on Embedded Devices.. In *NDSS*.
[20] Shijie Jia, Luning Xia, Bo Chen, and Peng Liu. 2017. Deftl: Implementing plausibly deniable encryption in flash translation layer. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2217–2229.
[21] Konstantinos Karampidis, Ergina Kavallieratou, and Giorgos Papadourakis. 2018. A review of image steganalysis techniques for digital forensics. *Journal of information security and applications* 40 (2018), 217–235.
[22] Mohammad Khan and Linda White. 2019. Steganalysis Techniques: Detecting the Undetectable. *Computer Security Review* 14, 2 (2019), 89–102. https://doi.org/10.1000/csr.2019.14.2.89
[23] Jinghui Liao, Bo Chen, and Weisong Shi. 2021. TrustZone enhanced plausibly deniable encryption system for mobile devices. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 441–447.
[24] Jinghui Liao, Niusen Chen, Lichen Xia, Bo Chen, and Weisong Shi. 2024. FSPDE: A Full Stack Plausibly Deniable Encryption System for Mobile Devices. In *2024 ACM Conference on Data and Application Security and Privacy (CODASPY)*. ACM.
[25] Jia Liu, Yan Ke, Zhuo Zhang, Yu Lei, Jun Li, Minqing Zhang, and Xiaoyuan Yang. 2020. Recent advances of image steganography with generative adversarial networks. *IEEE Access* 8 (2020), 60575–60597.
[26] Shao-Ping Lu, Rong Wang, Tao Zhong, and Paul L Rosin. 2021. Large-capacity image steganography based on invertible neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10816–10825.
[27] OP-TEE. 2019. Open Portable Trusted Execution Environment. *https://www.op-tee.org/* (2019).
[28] Timothy M Peters, Mark A Gondree, and Zachary NJ Peterson. 2015. DEFY: A deniable, encrypted file system for log-structured storage. (2015).
[29] Mark J Shensa et al. 1992. The discrete wavelet transform: wedding the a trous and Mallat algorithms. *IEEE Transactions on signal processing* 40, 10 (1992), 2464–2482.
[30] Adam Skillen and Mohammad Mannan. 2013. On Implementing Deniable Storage Encryption for Mobile Devices. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27*.
[31] Nandhini Subramanian, Omar Elharrouss, Somaya Al-Maadeed, and Ahmed Bouridane. 2021. Image steganography: A review of the recent advances. *IEEE*

access 9 (2021), 23409–23423.

[32] Weike You, Hong Zhang, and Xianfeng Zhao. 2020. A Siamese CNN for image steganalysis. *IEEE Transactions on Information Forensics and Security* 16 (2020), 291–306.

[33] Xingjie Yu, Bo Chen, Zhan Wang, Bing Chang, Wen Tao Zhu, and Jiwu Jing. 2014. Mobihydra: Pragmatic and multi-level plausibly deniable encryption storage for mobile devices. In *International conference on information security*. Springer, 555–567.

[34] Tao Zhang and Xijian Ping. 2003. A new approach to reliable detection of LSB steganography in natural images. *Signal processing* 83, 10 (2003), 2085–2093.