MobiWear: A Plausibly Deniable Encryption System for Wearable Mobile Devices

Niusen Chen[1], Bo Chen[1(✉)], and Weisong Shi[2]

[1] Department of Computer Science, Michigan Technological University, Houghton, MI, USA
bchen@mtu.edu

[2] Department of Computer Science, Wayne State University, Detroit, MI, USA

Abstract. Mobile computing devices are widely used in our daily life. With their increased use, a large amount of sensitive data are collected, stored, and managed in the mobile devices. To protect sensitive data, encryption is often used but, traditional encryption is vulnerable to coercive attacks in which the device owner is coerced by the adversary to disclose the decryption key. To defend against the coercive attacks, Plausibly Deniable Encryption (PDE) has been designed which can allow the victim user to deny the existence of hidden sensitive data. The PDE systems have been explored broadly for smartphones. However, the PDE systems which are suitable for wearable mobile devices are still missing in the literature.

In this work, we design MobiWear, the first PDE system specifically for wearable mobile devices. To accommodate the hardware nature of wearable devices, MobiWear: 1) uses image steganography to achieve PDE, which suits the resource-limited wearable devices; and 2) relies on various sensors equipped with the wearable devices to input passwords, rather than requiring users to enter them via a keyboard or a touchscreen. Security analysis and experimental evaluation using a real-world prototype (ported to an LG G smartwatch) show that MobiWear can ensure deniability with a small computational overhead as well as a small decrease of image quality.

Keywords: Confidentiality · Plausibly deniable encryption · Wearable mobile devices · Image steganography · Digital watermarking

1 Introduction

Mobile computing devices are ubiquitous today. More and more people choose to use their mobile devices, e.g. smartphones, tablets, smartwatches, to manage their personal private or even mission critical data. To protect confidentiality of sensitive data, full disk encryption (FDE) has been integrated into major mobile operating systems including Android [29] and iOS [15]. FDE can encrypt/decrypt data transparently to users, such that without having access to the secret key, the attacker

B. Chen and X. Huang (Eds.): AC3 2021, LNICST 386, pp. 138–154, 2021.
https://doi.org/10.1007/978-3-030-80851-8_10

will not be able to access plaintext of the data. FDE however, can only defend against a passive attacker which tries to steal sensitive data from external storage [21] of the mobile devices. It cannot defend against an active attacker which can capture the device owner and force him/her to disclose the secret key, i.e., a coercive attack. This type of coercive attackers can be found broadly in the real world. For example, a journalist uses a mobile device to collect criminal evidence of atrocities in a region of oppression, and stores the evidence encrypted; when he/she crosses the border, a border inspector may notice this encrypted ciphertext and require him/her to hand in the decryption key [27]. Plausibly deniable encryption (PDE) has been designed to ensure confidentiality of data against this type of coercive attacks. PDE can allow a victim user to deny the very existence of the hidden sensitive data upon being coerced. Its rationale is, data are encrypted in a special way so that, sensitive private data will be revealed only if a true key is used for decryption but, if a decoy key is used, only non-sensitive public data will be disclosed; when a device owner is coerced, he/she can simply disclose the decoy key and, using the decoy key, the attacker can obtain the non-sensitive data but will be unaware of the existence of the hidden sensitive data.

The concept of PDE has been broadly adapted to mobile devices [5–7,13,14,16,17,21,25,27,28,33] to protect hidden sensitive data against coercive attacks. We have a few observations on those existing mobile PDE systems. First, most of them are specifically built for smartphones [5,7,13,14,25,27,28,33], rather than wearable mobile devices like smartwatches. The PDE systems for the wearable devices have broader applications compared to smartphones. This is because, compared to using a smartphone, using a wearable device (e.g., an Apple watch) to capture criminal evidence (e.g., taking photos or recording videos) is more convenient and less likely to be noticeable[1]. Second, most of them rely on either the hidden volume technique [5,6,9,16,17,21,27,28,33] or the steganographic file system [7,25], and both techniques incline to hide sensitive data among randomness which suffers from several limitations: 1) Filling the randomness will cause expensive extra overhead. 2) An implied assumption needs to be made that, filling the randomness itself is a normal behavior and will not lead to compromise PDE.

This work aims to build a PDE system for wearable mobile devices, using smartwatches as a representative. The resulted system, MobiWear, is the first system which can allow a wearable device user to deny the very existence of hidden sensitive data when facing coercive attacks. Our key insights are two-fold: First, we do not rely on randomness to hide sensitive data; instead, we utilize image steganography. Specifically, having observed that images usually use digital watermarking to protect intellectual property, we choose to embed the sensitive data in the watermarks of images, so that upon being coerced, the hidden sensitive data can be denied as the regular image watermarks. Our image steganography does not incur too much overhead and is suitable for the light-weight wearable devices. In addition, we do not rely on the implied assumption that filling the randomness is a normal system behavior. Second, we carefully adapt the PDE system to the

[1] Note that the examples we show here are only limited to the scope of capturing criminal evidence in a region of oppression or conflict.

wearable devices. A PDE system usually requires some sorts of secrets, i.e., some low-security-level secrets which can be disclosed to the adversary (e.g., the decoy keys), as well some high-security-level secrets (e.g., the true keys) which should be unknown to the adversary. In traditional PDE systems for smartphones [6,7,27], the secrets are entered by users using a keyboard or a touchscreen; in a wearable device however, the screen is usually small, rendering it a "bad" practice to enter the secrets using either a keyboard or a touchscreen. Therefore, we rely on various sensors equipped with the wearable devices to enter the secrets.

Contributions. We summarize our major contributions as follows:

- We have designed the first PDE system specifically for wearable mobile computing devices, by combining the concept of PDE, image steganography, as well as digital watermarking, and adapting the design to the wearable devices. MobiWear is a light-weight PDE system which is well integrated with the hardware features of the wearable devices and well suits the resources-limited wearable devices.
- We have implemented a real-world prototype of MobiWear in an LG G smartwatch and evaluated its performance.
- We have also analyzed the security as well as discuss a few potential security issues.

2 Background

2.1 Wearable Mobile Devices

A wearable mobile computing device is a mobile device which can be worn on the body. The wearable devices can be used for the purpose of general computing, as well as special purposes like fitness tracker. They usually integrate a few special sensors like accelerometers, gyroscopes, magnetometers, heart rate sensors, and pedometers. The most popular wearable device is the smartwatch. Besides the basic functionality of a regular watch, the modern smartwatch may include various extra peripherals to achieve "smartness", e.g., digital cameras, tiny speakers, GPS receivers, pedometers, heart rate sensors, thermometers, accelerometers, altimeters, barometers, compasses, gyroscopes. Compared to a smartphone which is usually much larger in size and equipped with more powerful hardware (e.g., much larger touchscreens, more powerful processors, RAM, and batteries), the smartwatch is small in size and equipped with less powerful hardware, e.g., using small screens which do not well support user input, being equipped with less powerful processors, RAM, and batteries.

2.2 Plausibly Deniable Encryption

Plausibly deniable encryption (PDE) systems are designed to protect sensitive information when a device owner is coerced by an adversary. Upon being coerced, the device owner only reveals the decoy key which can be used to decrypt non-sensitive data. The actual secret key (i.e., true key) which can be used to decrypt

the sensitive data will be kept confidential and therefore, sensitive data are protected.

Currently, there are two major techniques which can implement the PDE concept in systems, namely, the steganography [4] and the hidden volume. The steganography-based PDE system hides sensitive data in regular files or randomness arbitrarily filled. However, since the system which manages the public non-sensitive data should not know the existence of the hidden sensitive data and, therefore, the hidden sensitive data may be overwritten by the public data. To mitigate this overwrite issue, several copies of hidden sensitive data are usually maintained across the entire disk. The hidden volume-based PDE system hides the sensitive data in a hidden volume. Its idea is: initially, the entire disk is filled with random data, and two volumes, a public and a hidden volume, are created; the public volume stores the public non-sensitive data, which are encrypted with a decoy key and placed across the entire disk; the hidden volume stores the sensitive data, which are encrypted with a true key and placed at the end of the disk starting from a secret offset; the hidden volume is completely embedded in the empty space of the public volume and, the attacker cannot detect its existence since he/she can not differentiate the encrypted hidden data from the randomness filled initially.

2.3 Image Steganography

Image steganography is often used to hide information in a cover image. Its process is as follows (Fig. 1): Secret data (e.g., texts or images) are stored invisibly in a cover image, generating a stego-image; this stego-image can then be sent to a receiver, where any third party will not be able to find out that the stego-image has hidden the secret data [19]; after having received the stego-image, the receiver can simply extract the secret data with or without a key [22].

In general, image steganography can hide secret data in two domains, the spatial domain and the transform domain. The least significant bit (LSB) steganographic embedding is an extremely simple technique of hiding secret data in the spatial domain. In an RGB image, each pixel consists of 4 channels, alpha (A), red (R), green (G) and blue (B), each of which occupies one byte. Alpha represents the value of transparency, and red, green and blue represent the value of three different colors. The last bit of each byte is called the least significant bit since its value only has a small effect on the pixel value [26] and, therefore, this bit can be used to hide sensitive data. A typical algorithm [3,20] for *embedding* the secret data is as follows: Given a secret key, a cover image and the secret data to be embedded, we first add two flags to the head and the tail of the secret data, respectively, generating the extended secret data. The two flags mark the beginning and the end of the secret data. We then encrypt the extended secret data using the secret key, and the resulted ciphertext will be treated as a collection of bits, which will be embedded sequentially to the pixels of the cover image (i.e., the least significant bit of each byte in a pixel will be used), generating a stego-image. Given the stego-image and the secret key, the *extraction* process of the LSB technique is: we extract the least significant bits from pixels of the

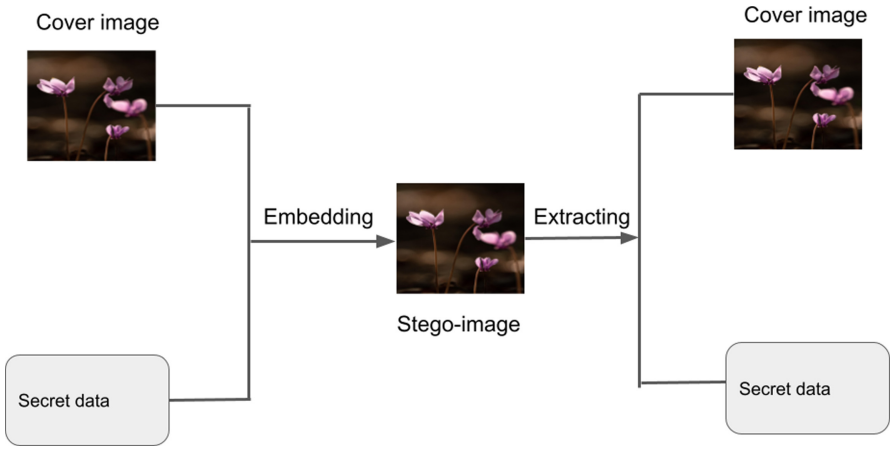


Fig. 1. Image steganography.

stego-image, decrypting them via the secret key, and can identify the beginning and the end of the secret data using the flags added during the embedding process; in general, it is no need to decrypt all the LSB bits, since we can treat the entire LSB bits as a collection of units (e.g., each unit can be 128 bits for AES-128), and decrypt each unit sequentially from the beginning until the "end" flag is found.

There are some variants of the LSB technique, e.g., pixel value differencing (PVD) [31], random pixel embedding method (RPE) [23], and pixel intensity based method [18]. The LSB technique has some advantages, including: 1) being hard to be detected by human eyes; and 2) simplicity of implementation; and 3) high payload compared to transform domain technique. It also has some disadvantages: 1) it is less robust compared to the transform domain technique; 2) the hidden data can easily be destroyed by simple attacks such as scaling and cropping.

2.4 Digital Watermarking

Digital watermarking can reinforce the security of multimedia data, by providing a solution to ensure tamper resistance as well as ownership protection of intellectual property [32]. A simple type of image watermarking is to embed a logo, which is a *visible* watermark. This is usually used for public identification and recognition. Another type of image watermarking is to embed an *invisible* watermark, which has been used broadly in multimedia data (e.g., images, videos) to claim copyrights.

2.5 Peak Signal-to-Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio, PSNR, represents a ratio between the maximal possible power of a signal and the power of the noise. A higher PSNR usually

indicates a good image quality. PSNR can be computed via Eq. 1, in which MAX_I is the maximal pixel value of the image, and MSE represents the mean square error. The MSE of an m × n image can be computed via Eq. 2, in which I is the original image and K is its noisy approximation.

$$PSNR = 20 \cdot log_{10}(MAX_I) - 10 \cdot log_{10}(MSE) \tag{1}$$

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{2}$$

3 Model and Assumptions

3.1 System Model

We consider a wearable mobile device, and its architecture is shown in Fig. 2. The architecture mainly contains three layers. The top layer is the application layer, which contains various user apps that directly interact with users and accept I/Os from users, e.g., an image viewer or editor. The middle layer is the operating system for wearable devices which, 1) manages the device's hardware resources, and 2) allows the apps to use the hardware resources via the system APIs. A popular operating system is Wear OS [2]. The bottom layer is the hardware which includes the processor, the RAM, and the flash storage. Note that a flash-based block device like a microSD card is used broadly in wearable devices, in which the flash memory is managed internally by flash translation layer (FTL), exposing a regular block access interface.

3.2 Adversarial Model

We consider a computationally bounded adversary. The adversary is able to capture a victim user together with his/her mobile device. The adversary notices the existence of ciphertext in the device and may coerce the victim user to disclose keys which can be used to decrypt the ciphertext. We need to rely on a few reasonable assumptions:

- The adversary is rationale and will stop coercing the victim user after being convinced that the keys have been disclosed. This is a common assumption for all the PDE designs [6,7,21,27].
- The adversary cannot capture a victim user when he/she is right processing the hidden sensitive data. Otherwise, the sensitive data will be obtained by the adversary trivially.
- We do not consider the attack that the adversary injects malware into the device before capturing it. Otherwise, the malware can monitor the process of embedding hidden data and compromise PDE trivially.

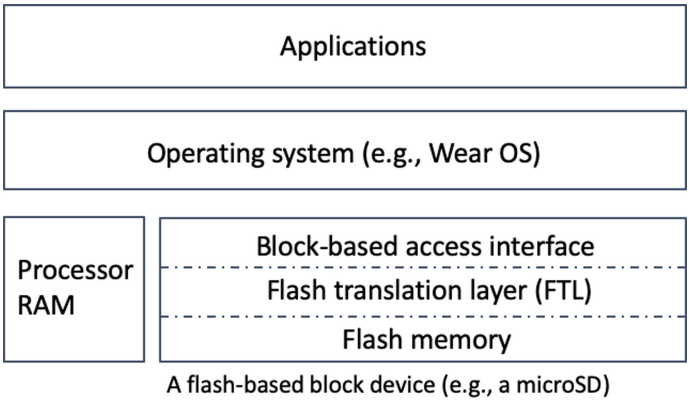


Fig. 2. The architecture of a wearable mobile device.

– The adversary is assumed to be able to obtain both the original cover image and the stego-image (i.e., the resulted image after the sensitive data are embedded), and to perform forensic analysis over them. The stego-image can be easily obtained once the wearable device is captured. The cover image may be obtained by the adversary considering that the cover image may be obtained from external sources, e.g., the device owner purchased it from others. However, the adversary is assumed to be not able to obtain the watermark image which is created locally by the device owner and has not been disclosed to the public. Note that a cautious user should delete the watermark image locally once used.

4 Design

4.1 Design Rationale

MobiWear is a PDE system specifically designed for wearable mobile devices. To ensure easy deployment of MobiWear, we integrate it into the application layer (Fig. 2) of a wearable mobile device. The design rationale of MobiWear is described in the following.

Upon being coerced, the victim should be able to convince the adversary that the noticeable embedding secrets are nothing but just some normal information. In the hidden volume-based technique [6,27] or steganographic file systems [25], the secret sensitive data are denied as randomness, which are filled by the device. This requires making the assumption that filling randomness is a normal system behavior [6,21,27]. Having observed that images can embed watermarks for intellectual property protection, we choose to hide the sensitive data into the images stored in the device, and deny them as the watermarks embedded into the images.

The process for embedding secret sensitive data into images is as follows (Fig. 3): Given some secret sensitive data, we first pick a cover image as well as a watermark image which will be embedded into the cover image. We then perform two steps: 1) We encrypt the sensitive data using a true key and, the resulted ciphertext will be embedded into the watermark image, obtaining a stego-watermark; 2) We encrypt the stego-watermark using a decoy key and, the resulted ciphertext will be embedded into the cover image, obtaining a stego-image. The stego-image will be stored to the wearable mobile device.

After the victim user is captured together with his/her wearable device, the adversary may notice that something is stored hidden in the stego-image. Note that the adversary is assumed to be able to obtain both the original cover image and the stego-image (Sect. 3.2). The adversary will coerce the victim user for the hidden sensitive data. The victim user will disclose the decoy key and claim that there is a watermark embedded in the stego-image. Utilizing the decoy key, the adversary can successfully extract the watermark (i.e., the stego-watermark) from the stego-image. The adversary will not be able to notice anything special in the stego-watermark, since it does not have access to the actual watermark (Sect. 3.2), and convince that there are no secret sensitive data stored. Only by utilizing the true key, the actual secret sensitive data can be extracted.

4.2 Design Details

How to Input Keys. Due to the small size of a wearable mobile device, using a keyboard or a touchscreen to enter keys is very inconvenient and impractical. We therefore rely on the embedded sensors in the wearable device to enter keys. There are many sensors such as accelerometer, gyroscope, heart rate sensor and compass sensor equipped with the wearable mobile devices. In MobiWear, we choose the gyroscope, which can be used to measure the rotation rate of x-axis, y-axis, and z-axis. Compared to other types of sensors, the gyroscope is a more convenient and robust choice for a wearable device's user to generate different input, each corresponding to a different key. For example, a user who is wearing a smartwatch can simply rotate his/her wrist differently to generate unique keys and, after obtaining the rotation rate from the gyroscope, the device can calculate the rotation degree in a time interval, generating unique keys. In MobiWear, two different keys, a decoy key and a true key, are generated. The decoy key is used to hide/extract the stego-watermark into/from the stego-image, and the true key is used to hide/extract the hidden sensitive data into/ from the stego-watermark.

Information Hiding and Extracting. The process of hiding secret sensitive data includes two steps: 1) hiding the sensitive data into the watermark image, obtaining a stego-watermark; and 2) hiding the stego-watermark into the cover image, obtaining a stego-image. For step 1, the sensitive data are encrypted using the true key and embedded into the watermark image, using the LSB steganographic embedding technique (Sect. 2.3). Similarly for step 2, the stego-watermark is encrypted using the decoy key and embedded into the cover image

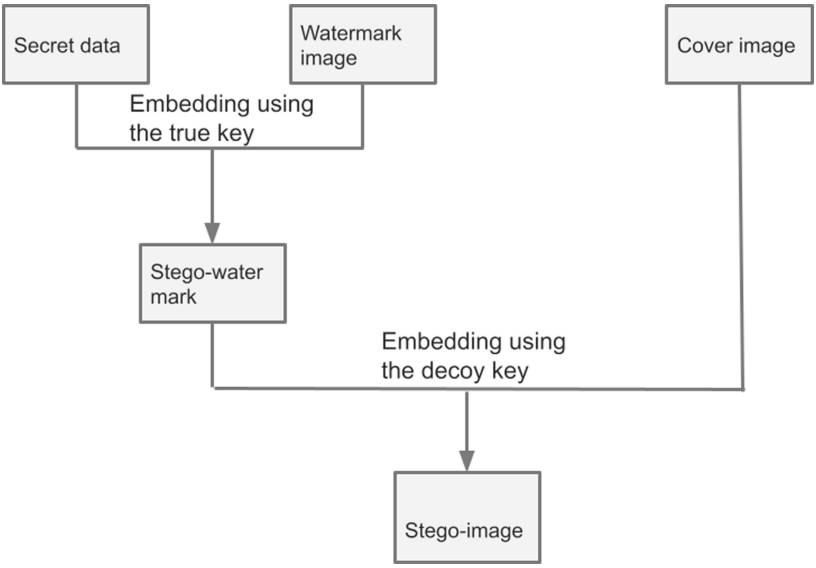


Fig. 3. The process of embedding secret sensitive data.

(via the LSB technique). Extracting sensitive information is the reverse operation of the information hiding. Once both the decoy and the true key are provided, MobiWear can use the decoy key to extract the stego-watermark from the cover image (i.e., via the extraction process of the LSB technique in Sect. 2.3), and then use the true key to extract the sensitive data from the stego-watermark (via the extraction process of the LSB technique). If only the decoy key is provided, only the stego-watermark can be extracted, denying the existence of the hidden sensitive data.

User Authentication. The user is required to enter the decoy key first and, if the decoy key is correct, MobiWear will wait for a certain amount of time (e.g., a few seconds). During this time interval, the user can enter the true key and, if the true key is entered within this time period and it is correct, the secret sensitive data will be extracted and displayed; otherwise, the stego-watermark will be displayed. The entire process for user authentication is shown in Fig. 4. Upon being coerced, the victim will disclose the decoy key and, using the decoy key, the adversary will be able to see the extracted stego-watermark, but will not be aware of the existence of the hidden sensitive data. Although there is a short delay after entering the decoy key, this can be simply denied as the system delay due to the limited computational power of a wearable device.

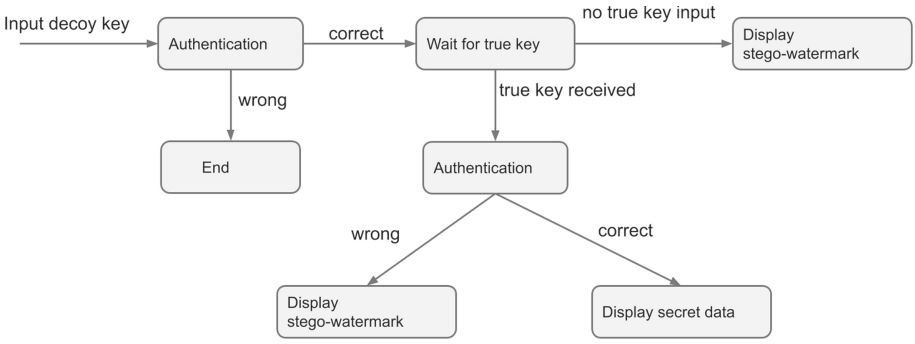


Fig. 4. The process of user authentication.

5 Implementation and Evaluation

5.1 Implementation

We have implemented MobiWear in an LG G watch [1], which has been released by LG and Google. It is equipped with Qualcomm Snapdragon 400 processor, 512 MB memory and 4 GB flash storage. Its embedded sensors include accelerometer, digital compass and gyroscope. The default operating system is Android Wear 1.5.0, which does not well support image viewing, and we have ported Android Wear OS 2.0 instead to resolve this issue. For image steganography, we relied on an open-source image steganography library [3], which has implemented the LSB steganographic embedding technique for Android. The encryption is instantiated using AES-128. Our major implementations are: 1) We support the password input via the embedded sensor gyroscope of the LG G watch; 2) We implement the PDE authentication via both the decoy and the true key; 3) We implement the information hiding and extracting process via the decoy and the true key.

5.2 Evaluation

Computational Overhead. We first evaluate the computational time of MobiWear in hiding/extracting data under different lengths of the secret data, while fixing the size of both the cover image and the watermark image. The results are shown in Fig. 5. We can observe that: 1) For longer secret data, MobiWear needs more time in hiding/extracting secret data. This is because: for longer secret data, MobiWear will need to embed/extract more secret bits into/ from the watermark image, which will increase the time; but the time for embedding/extracting the watermark image into/from the cover image will be identical. 2) Extracting the secret data is slower than hiding them. This is because: when embedding the data upon generating the stego-watermark and the stego-image (Sect. 2.3), two flags which indicate the begin and the end of the data are added and, when extracting them, we need to decrypt the LSB

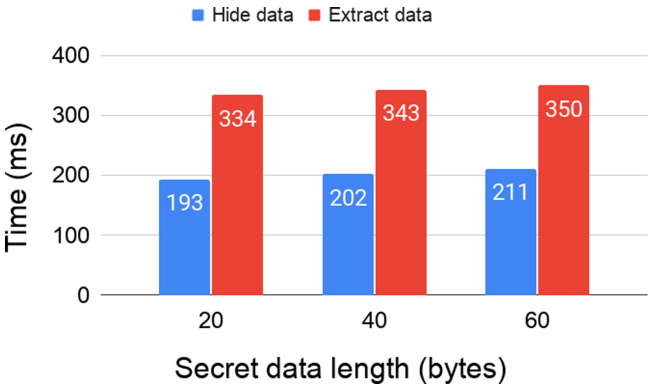


Fig. 5. Computational time for hiding and extracting sensitive data when the size of the cover image and the watermark image is fixed (the cover image is 1.5 MB in size, and the watermark image is 20 KB in size).

bits in terms of units until finding the end flag; this leads to extra overhead in decrypting more data and locating the end flag.

We then evaluate how the size of the watermark image and the cover image will affect the time of hiding/extracting secret data. We fix the length of the secret data as 20 bytes, and evaluate the hiding/extracting time under different sizes of the watermark and the cover image. The results are shown in Fig. 6. We can observe that: 1) The time for hiding/extracting the secret data slightly increases when the size of the cover/watermark image increases. This is because: both the cover and the watermark image need to be loaded into the memory for further processing, which slightly increases the computational time. 2) The time for extracting the secret data is more than that for hiding them. The reason has been mentioned before.

Assessing PSNR. To understand how MobiWear affects the image quality, we compute the PSNR values by varying the lengths of the secret data. We first hide the secret data with length 20, 40 and 60 (in bytes) in a watermark image, generating a corresponding stego-watermark, which is then embedded into the cover image. The size of the watermark image is fixed as 20 KB and the size of the cover image is fixed as 1.5 MB. We calculate the PSNR of each stego-image. The results are shown in Table 1. We can observe that, longer secret data will result in lower PSNR values, which indicates a worse image quality. This is because the longer secret data will occupy more least significant bits and increase the noise ratio. But still, the difference between the cover image and the stego-images is hard to be detected by human, which is justified in Fig. 7.

We also compare the PSNR values between original watermark image and the stego-watermark by varying the lengths of the secret data. The results are shown in Table 2. We can observe that a stego-watermark with longer secret data embedded has a lower PSNR value. This is because longer secret data will occupy more bits in the watermark image, decreasing its image quality. But, the

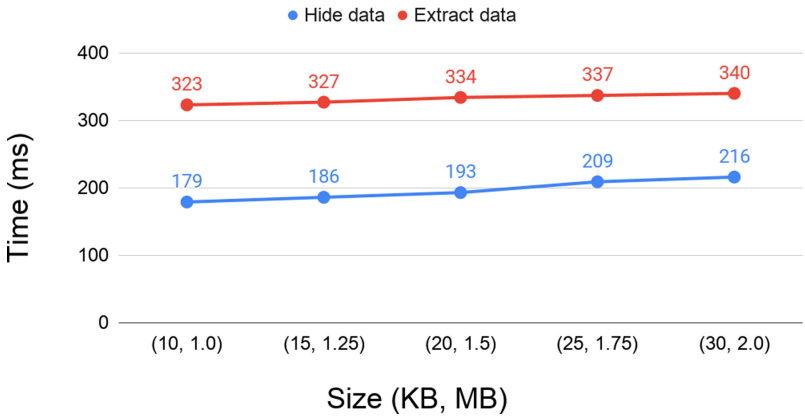


Fig. 6. Computational time for hiding and extracting sensitive data when the size of both the watermark image and the cover image varies. For the x-axis, (a, b) represents (size of watermark image in KBs, size of cover image in MBs).

Table 1. PSNR of stego-images under different lengths of secret data.

Length of secret data (bytes)	20	40	60
PSNR	30.6257	30.6153	30.5500

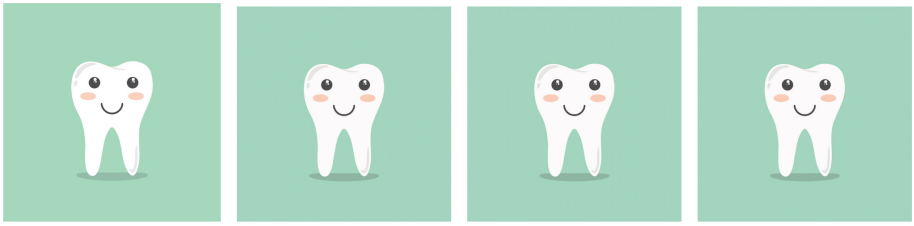
difference between the original watermark image and the stego-watermarks is also hard to be detected by human, which is justified in Fig. 8.

Table 2. PSNR of stego-watermarks under different lengths of secret data.

Length of secret data (bytes)	20	40	60
PSNR	33.8356	33.7800	30.7610

(a) Cover Image (b) Length=20 (c) Length=40 (d) Length=60

Fig. 7. The visualized comparison between the cover image (a) and stego-images (b–d) hiding secret data of different lengths.



(a) Watermark (b) Length=20 (c) Length=40 (d) Length=60

Fig. 8. The visualized comparison between the original watermark image (a) and the stego-watermarks (b–d) hiding secret data of different lengths.

6 Security Analysis and Discussion

6.1 Security Analysis

After having captured a victim wearable device, the adversary will obtain the stego-image from the device. For simplicity, we assume there is only one image in the device, which can be easily generalized to the actual case that there are multiple images. The adversary will obtain the corresponding cover image somehow (Sect. 3.2). By comparing the cover image and the stego-image, the adversary can identify differences between them and notice something has been embedded stealthily in the cover image. Note that since we use the invisible watermark (Sect. 2.4) in MobiWear and, without having access to the original cover image, the adversary should not be able to notice something has been hidden. By coercing the victim user, the adversary will be able to obtain the decoy key and, using the decoy key, the adversary will then extract the stego-watermark.

The adversary further performs forensic analysis over the extracted stego-watermark. This however, will not give the adversary any advantage of identifying the existence of hidden sensitive data because: The adversary cannot have access to the original watermark image (Sect. 3.2) and, without being able to compare the stego-watermark with the original watermark, there is no way for the adversary to identify whether there are any modifications over the LSBs of the stego-watermark's pixels (solely viewing the stego-watermark will not result in any visualized abnormality).

Therefore, we can conclude that, the adversary will not be able to identify the existence of PDE and the security of MobiWear can be ensured.

6.2 Discussion

About the Length of Sensitive Data which can be Hidden. MobiWear hides the sensitive data into the image and, the length of the hidden data will be limited by the size of the image and the corresponding watermark. Given an ARGB cover image with N pixels and each pixel consisting of 4 bytes (alpha, red,

green, blue), we analyze the length of the sensitive data which can be hidden. Using the LSB technique, the maximal size of watermark the cover image can embed is $\frac{N}{2}$ bytes considering 1 bit out of each byte can be used to embed the watermark. Correspondingly, the maximal length of secret sensitive data which can be hidden in the watermark is $\frac{N}{16}$ bytes. For example, a 4 MB cover image will have 1M pixels, and can hide up to 0.0625 MB sensitive data. To hide more sensitive data, we can use more least significant bits, which however will decrease the quality of the image. A practical mitigation is to cut the sensitive data of large size into small chunks, and to hide each chunk using a different cover image.

Deniability Compromises in Memory. Secret sensitive data may leave traces in the memory, leading to compromise of PDE. Considering the volatile nature of RAM, an immediate mitigation strategy is to power-off the device to remove the traces of secret sensitive data in the memory. Other mitigation strategies include utilizing hardware isolation techniques like ARM TrustZone to isolate the memory region which processes the hidden sensitive data [5].

Defending Against Multi-snapshot Adversaries. MobiWear can defend against an adversary which can have access to the victim device once. Note that by using image steganography, MobiWear remains secure even if the adversary can have access to different layers of the system (Fig. 2). If the adversary can capture the device and its owner multiple times, it will have multiple access to the device over time, and a potential PDE compromise could be: If the secret sensitive data are modified and a new stego-watermark needs to be re-embedded into the original cover image and, by comparing the stego-image at different points of time, the adversary may be aware of the existence of hidden sensitive data. A potential mitigation strategy can be, each time when the sensitive data are modified, the device owner should discard the corresponding stego-image which turns obsolete, and hide the new data using a new cover image.

Mitigating Data Corruptions. MobiWear uses the LSB technique to hide secret sensitive data. This is vulnerable to cutting and cropping attack which will destroy the sensitive data. However, the goal of PDE is to ensure confidentiality of the sensitive data, rather than to prevent data from being corrupted. A recommendation for mitigating corruption attacks is to periodically back up the sensitive data, e.g., to a remote cloud server or an offline personal computer.

7 Related Work

7.1 Plausibly Deniable Encryption Systems

Plausibly deniable encryption has been designed to defend against coercive attacks so that even though the key is forced to be disclosed, the critical sensitive data can remain protected. There are mainly two types of PDE systems, the steganography-based PDEs and the hidden volume-based PDEs. There are also PDE systems relying on other techniques like side channel [10] and WOM codes [11].

The Steganography-Based PDE Systems. The first steganography-based PDE system [4] was proposed by Anderson et al. They designed two schemes. The first one is to hide sensitive data in cover files, which however, requires the system to store a large number of cover files. The second one is to fill the entire disk with random data and, to encrypt and hide the secret data in those random data. Based on the second scheme of Anderson et al., McDonald et al. designed StegFS [24], in which they extended a standard Linux file system (EXT2) with PDE support. Peters et al. proposed DEFY [25], a flash file system which supports PDE. Its features include authenticated encryption, fast secure deletion, and support for multiple layers of deniability.

The Hidden-Volume Based PDE Systems. The other technique which can be used to achieve PDE is the hidden volume technique. TrueCrypt [30] and VeraCrypt [12] are open-source projects for disk encryption, with deniability support using the hidden volume technique. Skillen et al. [27,28] proposed Mobiflage which moved the hidden volume technique to mobile computing devices. Mobiflage requires the user to re-boot the device to enter the hidden mode, which is inconvenient. To mitigate this issue, Yu et al. proposed MobiHydra [33] which supports data hiding without the need of rebooting the device. MobiHydra also solved a boot-time attack on the PDE systems. Chang et al. proposed Mobipluto [5,6], a file system friendly PDE system such that any block-based file systems can be deployed on top of the public volume, without worrying about overwriting the hidden sensitive data. Jia et al. [21] further moved the hidden volume to the flash translation layer, eliminating the deniability compromised in the low-level flash memory medium. Having observed that the prior mobile PDE systems cannot defend against a multi-snapshot adversary, Chang et al. designed MobiCeal [7], which combines both the hidden volume technique and the dummy write technique to enable defend against multi-snapshot adversaries.

7.2 Image Steganograyphy

Image steganography has been widely used to claim the ownership or copyright of the products. Wu et al. [31] proposed an efficient steganographic method to embed secret messages into cover images, which is based on a simple visual effect of the human visual perception. Ibrahim et al. [20] proposed an algorithm to hide data in cover images, by using binary codes and pixels inside an image. Based on the proposed algorithm, a system called Steganography Imaging System (SIS) is built which can hide secret message without a noticeable distortion. Chaumontet et al. [8] proposed a DCT-based data hiding method which can hide color information in a compressed gray-level image. Their proposed method consists of three steps, color quantization, color ordering, and data hiding.

8 Conclusion

In this work, we design MobiWear, a plausibly deniable encryption system for wearable mobile devices. MobiWear uses image steganography to hide sensitive

data and utilizes the integrated sensors to input secrets. The experiment results indicate that MobiWear can achieve deniability with a small overhead as well as a slight decrease of image quality.

Acknowledgments. This work was supported by US National Science Foundation under grant number 1928349-CNS, 1928331-CNS, and 1938130-CNS.

References

1. LG G watch (2016). https://www.lg.com/us/smart-watches/lg-W100-lg-watch
2. Wear OS (2016). https://wearos.google.com/#stay-connected
3. Agarwal, A.: Image-steganography-library-android (2011). https://github.com/aagarwal1012/Image-Steganography-Library-Android
4. Anderson, R., Needham, R., Shamir, A.: The steganographic file system. In: Aucsmith, D. (ed.) IH 1998. LNCS, vol. 1525, pp. 73–82. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-49380-8_6
5. Chang, B., et al.: User-friendly deniable storage for mobile devices. Comput. Secur. **72**, 163 (2017)
6. Chang, B., Wang, Z., Chen, B., Zhang, F.: Mobipluto: file system friendly deniable storage for mobile devices. In: Proceedings of the 31st Annual Computer Security Applications Conference, pp. 381–390 (2015)
7. Chang, B., et al.: 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), pp. 454–465. IEEE (2018)
8. Chaumont, M., Puech, W.: A DCT-based data-hiding method to embed the color information in a jpeg grey level image. In: 2006 14th European Signal Processing Conference, pp. 1–5. IEEE (2006)
9. Chen, B., Chen, N.: Poster: a secure plausibly deniable system for mobile devices against multi-snapshot adversaries. In: 2020 IEEE Symposium on Security and Privacy Poster Session (2020)
10. Chen, C., Chakraborti, A., Sion, R.: Infuse: invisible plausibly-deniable file system for nand flash. Proc. Priv. Enhancing Technol. **4**, 239–254 (2020)
11. Chen, C., Chakraborti, A., Sion, R.: Pearl: plausibly deniable flash translation layer using WOM coding. In: The 30th Unsenix Security Symposium (2021)
12. CodePlex. Veracrypt ssd. https://veracrypt.codeplex.com/, 2017
13. Trnka, D.: Steganography software.version 1.3 (2014). https://play.google.com/store/apps/details?id=com.dinaga.photosecret&hl=en_US&gl=US
14. EDS. Free open source on-the-fly disk encryption software.version 2.0.0.243 (2012). http://www.sovworks.com/
15. How to encrypt your devices (2017). https://spreadprivacy.com/how-to-encrypt-devices/
16. Feng, W., et al.: Mobigyges: a mobile hidden volume for preventing data loss, improving storage utilization, and avoiding device reboot. Future Gener. Comput. Syst. **109**, 158 (2020)
17. Hong, S., Liu, C., Ren, B., Huang, Y., Chen, J.: Personal privacy protection framework based on hidden technology for smartphones. IEEE Access **5**, 6515–6526 (2017)
18. Hussain, M., Hussain, M.: Pixel intensity based high capacity data embedding method. In: 2010 International Conference on Information and Emerging Technologies, pp. 1–5. IEEE (2010)

19. Hussain, M., Hussain, M.: A survey of image steganography techniques (2013)
20. Ibrahim, R., Teoh, S.K.: Teganography algorithm to hide secret message inside an image. J. Comput. Technol. Appl. (JCTA) **1**(2), 102–108 (2011)
21. Jia, S., Xia, L., Chen, B., Liu, P.: DEFTL: implementing plausibly deniable encryption in flash translation layer. In: Proceedings of the 24th ACM Conference on Computer and Communications Security. ACM (2017)
22. Johnson, N.F., Jajodia, S.: Exploring steganography seeing the unseen. Computer **31**(2), 26–34 (1998)
23. Liu, L., Chen, T., Cao, C., Wen, X., Xie, R.: A novel data embedding method using random pixels selecting. Inf. Technol. J. **12**(7), 1299 (2013)
24. McDonald, A.D., Kuhn, M.G.: StegFS: a steganographic file system for linux. In: Pfitzmann, A. (ed.) IH 1999. LNCS, vol. 1768, pp. 463–477. Springer, Heidelberg (2000). https://doi.org/10.1007/10719724_32
25. Peters, T.M., Gondree, M.A., Peterson, Z.N.J.: DEFY: a deniable, encrypted file system for log-structured storage. In: 22th Annual Network and Distributed System Security Symposium, NDSS (2015)
26. Singh, A.K., Singh, J., Singh, H.V.: Steganography in images using LSB technique. Int. J. Latest Trends Eng. Technol. (IJLTET) **5**(1), 426–430 (2015)
27. Skillen, A., Mannan, M.: On implementing deniable storage encryption for mobile devices. In: 20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA (2013)
28. Skillen, A., Mannan, M.: Mobiflage: deniable storage encryption for mobile devices. IEEE Trans. Depend. Secure Comput. **11**(3), 224–237 (2014)
29. Source. Android full disk encryption (2016). https://source.android.com/security/encryption/
30. TrueCrypt. Free open source on-the-fly disk encryption software. version 7.1a (2012). http://www.truecrypt.org/
31. Da-Chun, W., Tsai, W.-H.: A steganographic method for images by pixel-value differencing. Pattern Recogn. Lett. **24**(9–10), 1613–1626 (2003)
32. You, W., Chen, B., Liu, L., Jing, J.: Deduplication-friendly watermarking for multimedia data in public clouds. In: Chen, L., Li, N., Liang, K., Schneider, S. (eds.) ESORICS 2020. LNCS, vol. 12308, pp. 67–87. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58951-6_4
33. Yu, X., Chen, B., Wang, Z., Chang, B., Zhu, W.T., Jing, J.: Mobihydra: pragmatic and multi-level plausibly deniable encryption storage for mobile devices. In: Information Security - 17th International Conference, ISC 2014, Hong Kong, China. Proceedings, pp. 555–567 (2014)