# Poster: A Secure Plausibly Deniable System for Mobile Devices against Multi-snapshot Adversaries

Bo Chen, Niusen Chen
*Department of Computer Science*, *Michigan Technological University*
{bchen, niusenc}@mtu.edu

***Abstract*—Mobile computing devices have been used broadly to store, manage and process critical data. To protect confidentiality of stored data, major mobile operating systems provide full disk encryption, which relies on traditional encryption and requires keeping the decryption keys secret. This however, may not be true as an active attacker may coerce victims for decryption keys. Plausibly deniable encryption (PDE) can defend against such a coercive attacker by disguising the secret keys with decoy keys. Leveraging concept of PDE, various PDE systems have been built for mobile devices. However, a practical PDE system is still missing which can be compatible with mainstream mobile devices and, meanwhile, remains secure when facing a strong multi-snapshot adversary. This work fills this gap by designing the first mobile PDE system against the multi-snapshot adversaries.**

***Index Terms*—plausibly deniable encryption, mobile devices, flash memory, multi-snapshot adversaries**

## I. Introduction

People today are turning to mobile computing devices (e.g., smartphones, tablets, smart home assistants like Amazon Echo and Google Home) for social networking, e-commerce, online banking, etc. This brings a lot of convenience, but also creates a significant security concern since mobile devices are increasingly used to store, manage, and process sensitive data. Especially, users from federal agencies/military departments increasingly use mobile devices to handle mission critical data daily. This makes it an urgent need to protect data in mobile devices. Major mobile operating systems have incorporated full disk encryption (FDE) [1] to protect data confidentiality. This however, cannot defend against a coercive attack, in which the device's owner is forced to disclose the decryption key. For example, a human rights worker collects criminal evidence using her phone in a region of oppression; to keep the evidence secret, she stores it encrypted; when she tries to cross border, the inspector may notice existence of encrypted data and coerce her to disclose the decryption key.

Plausibly Deniable Encryption (PDE) has been proposed to defend against an adversary which can coerce users into revealing secret keys. PDE works as follows: Original sensitive data are encrypted into ciphertexts such that, when using a decoy key, reasonable and innocuous plaintexts will be generated; only when using a true key, original sensitive data will be obtained; therefore upon coerced, the victim can disclose the decoy key and, the adversary uses the decoy key for decryption, obtaining the innocuous plaintexts rather than the original sensitive data. Instantiating PDE in cryptography could be a challenging problem. Instead, a variety of deniable storage systems [3], [4], [7], [8] were proposed to provide deniability for mobile devices, leveraging PDE concept. However, they are built on the block layer and suffer from deniability compromise in the underlying flash memory, due to not being able to "touch" the internal raw flash [6].

DEFTL [6] takes an initial step of incorporating PDE into flash memory[1]. However, it is vulnerable to a multi-snapshot adversary, which can have access to the raw flash multiple times over time and, by comparing different disk snapshots captured, it can detect unaccountable changes on the random data caused by writes performed by the hidden volume, compromising deniability. A PDE system designed for mobile devices which can resist against multi-snapshot adversaries is still missing in the literature. This work thus aims to fill this gap by designing *the first secure PDE system for mobile devices which can combat multi-snapshot adversaries*.

## II. Background on Flash Memory

As a non-volatile storage medium which can be electrically erased and reprogrammed, flash memory is broadly used in mainstream mobile devices including smartphones, tablets, IoT devices. Flash memory stores information in an array of memory cells, which are grouped into blocks, and each block is divided into pages. Compared to mechanical disks, NAND flash has some unique characteristics: 1) It uses an erase-before-write design, i.e., a flash cell needs to be erased before it can be overwritten. 2) Its unit of read/program is a page, but its unit of erase is a block. Therefore, overwriting a page requires first erasing the entire encompassing block which is expensive. Thus, it usually uses an out-of-place update strategy. 3) A flash block usually has a finite number (e.g., 10K) of program-erase cycles, and wear leveling is usually required to distribute writes/erasures across the flash.

The most popular form of using flash memory is to emulate it as a block device to be compatible with traditional block file systems like EXT4 and FAT32. This is commonly found in flash memory cards like eMMC cards, SD/microSD/MiniSD cards. A piece of special firmware, Flash Translation Layer (FTL), is introduced between the block device and the raw flash. The FTL transparently handles unique nature of flash, and implements functions like address translation, garbage collection, wear leveling, and bad block management.

[1]Another work DEFY [7] also tries to build PDE into flash memory. Their design strongly relies on system properties provided by a flash-specific file system YAFFS, which is rarely used today.

## III. Adversarial Model

We consider a computationally bounded adversary, which can have access to external storage of a victim mobile device at different points of time, i.e., a multi-snapshot adversary. Each snapshot can be a physical image of raw NAND flash, obtainable by forensic data recovery tools [2]. The adversary is assumed to be not able to capture the device owner when she is working on storing hidden sensitive data.

## IV. Design

Compared to a single-snapshot adversary [3], [6], [8], the multi-snapshot adversary is much more difficult to be combated. *MobiCeal* [4] combats an adversary which can obtain multiple snapshots from the block device layer by: 1) introducing a dummy write on the block device to deny unaccountable changes caused by hidden sensitive data; and 2) writing all the data (public, dummy and hidden data) to random locations of the block device to avoid deniability compromise. This however, cannot combat an adversary which can obtain multiple snapshots from the raw flash memory (Sec. III), because: To accommodate special nature of flash memory, FTL usually follows a log-structured writing manner, i.e., regardless how data are written on the block device, they are written sequentially to the flash memory; thus, random writes on the block device will become useless and deniability may be compromised by the snapshot adversary using the raw flash memory. To combat the multi-snapshot attacks, we carefully modify the FTL as follows:

First, two modes, a public and a hidden mode, are incorporated into the FTL. Correspondingly, there are two volumes, a public and a hidden volume introduced. The public volume is encrypted using FDE with a decoy key while the hidden volume is encrypted using FDE with a true key. In the public mode, the user can write public non-sensitive data to the public volume and, in the hidden mode, the user can write hidden sensitive data to the hidden volume. The public volume is initially filled with randomness and the hidden volume is stored hidden among the randomness.

Second, upon writing the public volume, the FTL will perform additional dummy writes of random data. A multi-snapshot adversary can compare two snapshots captured, and will notice changes among the randomness caused by hidden data writes. Therefore, by performing dummy writes, the hidden data writes can be denied as the dummy writes. One security issue is that, rarely, there are no public data writes between two snapshots captured by the adversary and, if any hidden data writes are performed, the adversary will detect existence of hidden writes, compromising deniability. To avoid this compromise, the FTL can actively perform a few dummy writes if there are no public data writes for a long time.

Third, the log-structure writing strategy of the FTL is changed to random writing. With log-structure writing, when a user writes a large hidden file in the hidden mode,it will be stored in flash blocks following those storing public data. This can be noticed by the adversary, leading to deniability compromise. With random writing, this compromise will be mitigated. In addition, the random writing strategy is exclusively feasible for flash memory, since random seeks are efficient on flash memory. In addition, random placements inherently distribute data evenly among flash, achieving good wear leveling.

Other techniques include introducing a global bitmap to FTL to keep track of usage of flash pages to prevent public data from overwriting hidden data. A more complete description of our design can be found in our technical report [5].

## V. Preliminary Experimental Results

A major change of the FTL in our design is to convert the conventional log-structure writing to random writing as well as introduce dummy writes to obfuscate writes performed by the hidden mode. To access this impact, we implemented random writing and dummy writes into an open-source NAND flash controller framework, OpenNFM. We currently implemented dummy writes in a simple way that, when a normal write is performed, we will perform a dummy write if a randomness-based dummy write condition [4] is satisfied. We also ported our prototype to LPC-H3131, an electronic development board equipped with 180MHz ARM micro-controller, 512MB SLC NAND flash. To simulate a mobile device, we used another embedded board, Firefly AIO-3399J, as the host device to read/write the external flash storage provided by LPC-H3131 via a USB interface. We ran benchmark tool "fio" in the host device. The write throughput is shown in Table I. We can observe that, compared to the original OpenNFM, the write throughput of our design decreases around 60%. This is due to extra overhead caused by random writing as well as dummy writes associated with each normal write.

| fio pattern | OpenNFM | Our Scheme |
|---|---|---|
| sequential write | 2550 KB/s | 784 KB/s |
| random write | 2110 KB/s | 760 KB/s |

TABLE I
Throughput comparison.

## VI. Acknowledgement

We would like to thank Fengwei Zhang for the initial discussion of the design. This work was supported by National Science Foundation (1928349-CNS and 1938130-CNS).

## References

[1] Android full disk encryption. https://source.android.com/security/encryption/, 2016.

[2] M. Breeuwsma, M. D. Jongh, C. Klaver, R. V. D. Knijff, and M. Roeloffs. Forensic data recovery from flash memory. *Small Scale Digital Device Forensics Journal*, 1(1):1–17, 2007.

[3] B. Chang, Z. Wang, B. Chen, and F. Zhang. Mobipluto: File system friendly deniable storage for mobile devices. In *ACSAC*. ACM, 2015.

[4] B. Chang, F. Zhang, B. Chen, Y. Li, W. Zhu, Y. Tian, Z. Wang, and A. Ching. Mobiceal: Towards secure and practical plausibly deniable encryption on mobile devices. In *DSN*, pages 454–465. IEEE, 2018.

[5] B. Chen. Towards designing a secure plausibly deniable system for mobile devices against multi-snapshot adversaries–a preliminary design. *arXiv preprint arXiv:2002.02379*, 2020.

[6] S. Jia, L. Xia, B. Chen, and P. Liu. Deftl: Implementing plausibly deniable encryption in flash translation layer. In *CCS*. ACM, 2017.

[7] T. M. Peters, M. A. Gondree, and Z. N. J. Peterson. DEFY: A deniable, encrypted file system for log-structured storage. In *NDSS*, 2015.

[8] A. Skillen and M. Mannan. On implementing deniable storage encryption for mobile devices. In *NDSS*, 2013.