MOBICEAL: TOWARDS SECURE AND PRACTICAL PLAUSIBLY DENIABLE ENCRYPTION ON MOBILE DEVICES

Bing Chang, Fengwei Zhang, Bo Chen, Yingjiu Li, Wen-Tao Zhu, Yangguang Tian, Zhan Wang and Albert Ching

OVERVIEW

WHAT IS PDE?

WHAT IS PLAUSIBLY DENIABLE ENCRYPTION?

- Traditional encryption (full disk encryption, or FDE) is ineffective if user can be coerced into disclosing decryption key
- Allows for user to plausibly deny the existence of sensitive material on the encrypted disk
- Data can be decrypted in two or more different ways – only one being the actual protected data
- The user can reveal the public volume decryption key under coercion

HIDDEN VOLUMES

- The PDE tool encrypts two volumes of data – the hidden volume containing the actual data, and a public decoy volume
- Public volume placed on entire disk, hidden volume placed at a defined offset from the end of the disk
- Works for one-time access from an adversary
- Multiple disk snapshots taken at different points in time by an adversary can detect presence of hidden volume

STEGANOGRAPHIC FILE SYSTEMS

- Hides sensitive data among regular public files
- Done by either introducing many cover files, or hiding the data in dummy file blocks
- Main challenge is to avoid overwriting hidden data
- Requires a large amount of redundancy and disk space

PDE ON MOBILE DEVICES

PDE ON MOBILE DEVICES

- Existing PDE solutions on mobile devices hide encrypted data in randomness on disk
- An adversary can detect changes to this randomness over multiple snapshots
- Reboot is required for access to hidden mode – may not be feasible in situations where sensitive information needs to be captured quickly
- PDE solutions to multi-snapshot adversaries not suitable for mobile devices

PDE ON MOBILE DEVICES

- HIVE and DataLair, two desktop PDE solutions, require "write-only oblivious RAM" (ORAM) to prevent data differentiation over multiple snapshots
- HIVE and DataLair assume that the adversary can obtain a snapshot after every write to the disk
- This has poor I/O performance, and not suitable for mobile devices
- MobiCeal assumes "on-event" adversary that can obtain multiple snapshots after the user is prepared, making it more lightweight and suitable to mobile devices

PDE ON MOBILE DEVICES

- Unlike HIVE and DEFY (another PDE tool), MobiCeal is not vulnerable to side-channel attacks. Both HIVE and DEFY do not sufficiently isolate hidden data from public data
- DEFY is designed for mobile devices and to prevent multiple-snapshot attacks, however it relies on properties of the flash filesystem YAFFS, limiting device compatibility
- Other tools do not provide enough detail on correct usage, which may lead to information exposure if used incorrectly
- Some tools require a reboot to switch between public and hidden mode, which is time consuming and may not be feasible in all situations

MOBICEAL

- MobiCeal provides plausible deniability against multiple-snapshot adversaries
- Not vulnerable to side channel attacks
- Designed to be implemented on mainstream mobile devices
- Built into the block layer of the Linux kernel, allowing deployment of any block filesystems
- Lightweight "dummy write" mechanism, with relatively low overhead
- Fast switching between public and hidden mode

MODEL AND ASSUMPTIONS

ADVERSARIAL MODEL & ASSUMPTIONS

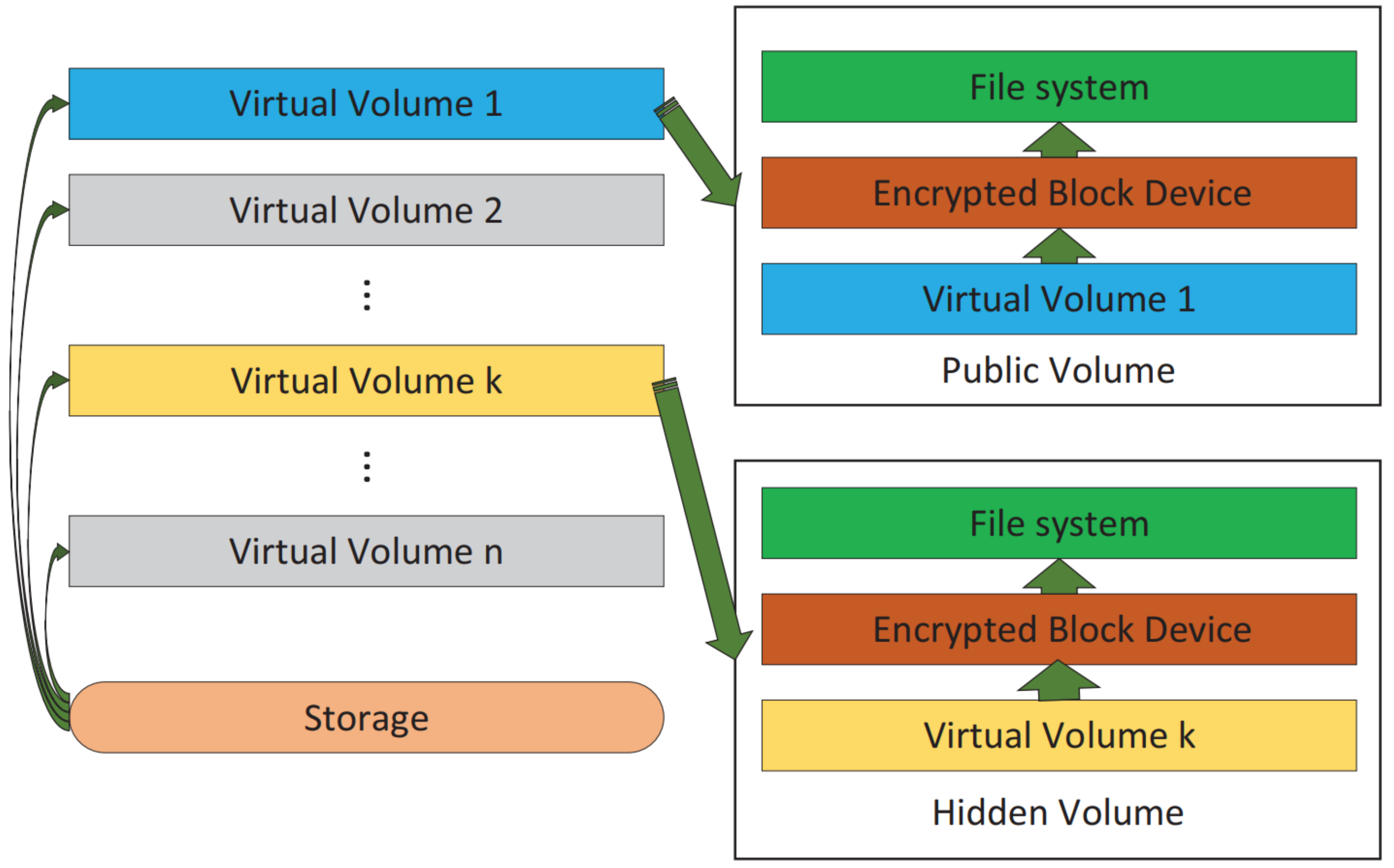
- Assumes a computationally bounded (i.e. reasonable computation power) adversary that takes snapshots of the device at different points in time (entering and leaving a country or secured facility)
- Adversary is allowed full knowledge of MobiCeal's design
- Coercion of the device owner is allowed
- Does not know encryption key or password of the hidden volume
- Can obtain root privilege and access internal and external storage during each snapshot

ADVERSARIAL MODEL & ASSUMPTIONS

- Adversary assumed to not be able to capture the device owner when the hidden volume is being worked with
- Assumed that coercion will stop when the device owner discloses the decoy password/encryption key
- Assumed that MobiCeal is merged with the Android code stream, so that its availability is not suspicious
- The OS, kernel, bootloader, firmware, and baseband OS, and other apps are not compromised

DESIGN

MOBICEAL SYSTEM ARCHITECTURE



DESIGN OVERVIEW

- Uses a dummy write approach, allowing for similar benefits to ORAM approaches with less overhead
- When writing public data, MobiCeal performs some additional random writes to the dummy volume (following an expontential distribution)
- Comparision between snapshots by an adversary will show changes, but the use can attribute these to the dummy volume instead of the hidden volume
- The dummy data is created using the same encryption algorithm as the hidden data with random input and keys

DESIGN OVERVIEW

- The public mode has no knowledge of the hidden volume, so a way is needed to ensure that public data does not overwrite hidden data
- The hidden-volume technique resolves this by placing the hidden volume at the end of the disk
- This is only suitable for file systems that write sequentially on the disk, and overwrites are still possible when the disk is under heavy load
- Borrows "global bitmap" from steganographic file system, storing it in the block layer. This keeps track of blocks being used by the public, dummy, and hidden data, marking blocks as allocated and preventing overwrite by public/dummy data

A BASIC MOBICEAL SCHEME

- Three types of virtual volumes:
  1. Public volume – used for operations not involving sensitive operation, provides encryption without deniability. This is accessed with the decoy key, computed with the decoy password
  2. Hidden volume – used when storing sensitive data, provides deniability. Accessed with hidden password at boot
  3. Dummy volume – stores data created by dummy writes, obfuscates hidden volume. Adversary cannot differentiate between this and a hidden volume

A BASIC MOBICEAL SCHEME

- Dummy write mechanism is used to obfuscate writes to the hidden volume
- Dummy write will be performed with a certain probability when blocks are allocated to the public volume
- Dummy writes are only performed when the condition is met, in order to prevent adversary from determining dummy write pattern

Condition:

```
rand <= stored_rand mod x
```

- ***x*** is a positive integer constant,
- ***stored_rand*** is a random number that is periodically updated (PRNG or noise)
- ***rand*** is an integer chosen uniformly from `1` to `2*x` (ensuring probability of dummy write is always under 50%)

A BASIC MOBICEAL SCHEME

- When performing a dummy write, *m* free blocks are allocated and marked as such in the global bitmap
- Blocks are filled with random noise, which should be indistinguishable from encrypted data

$$m = \mathrm{floor}(m')$$

$$m' = -(\ln(1-f))/\lambda$$

- f is a random number in (0,1)
- $\lambda$is the rate parameter, making m' follow exponential distribution
- The mean value of m'/λ is 1, meaning if 1 is chosen as λ, the dummy write will be allocated 1 free block on average
- An exponential distribution means large variance, good for deniability

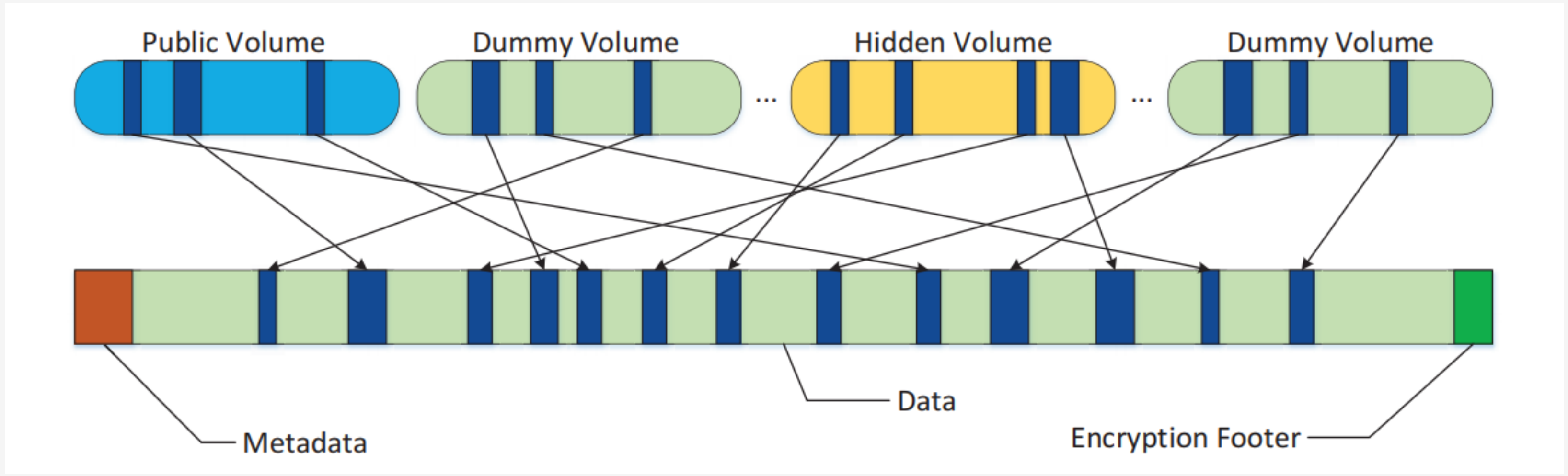
A BASIC MOBICEAL SCHEME

- A common allocation strategy is sequential block allocation, where data blocks are allocated to volumes sequentially from the disk, example:

  Dh || Dv || Dh || Dh || Dh || Dh || Dh || Dh || Dh || Dv

- Where Dh are blocks written to the hidden volume, and Dv are writes to the public volume
- From this example, an adversary can observe that seven blocks are allocated between the two allocations to the public volume. The user can claim these are allocations to the dummy volume, but dummy writes are limited, and the adversary may notice that the allocations to the dummy block exceeds this limit
- To avoid this issue, MobiCeal uses random block allocation, eliminating large sequences of blocks that are apparently written to the dummy volume

MOBICEAL STORAGE LAYOUT



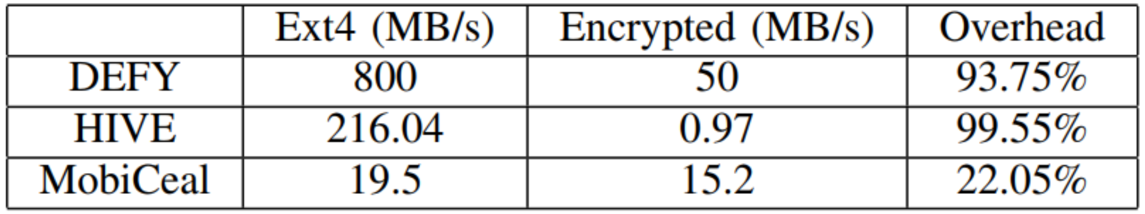
SIDE CHANNEL ATTACKS

- MobiCeal isolates the hidden volume from the public volume, preventing hidden volume information from being recorded in the public volume
- The hidden password is entered in public mode, the password is not recorded, assuming other parts of the system are not compromised
- The partitions containing */devlog* and */cache* are unmounted immediately after the hidden password is verified, and RAM disks are mounted in their place, preventing data leakage to the public volume. The hidden volume is then decrypted, and mounted as the userdata partition
- Fast switching between modes is only supported from public mode to hidden mode, ensuring that anything stored in memory from hidden mode is cleared from the RAM on reboot

EVALUATION

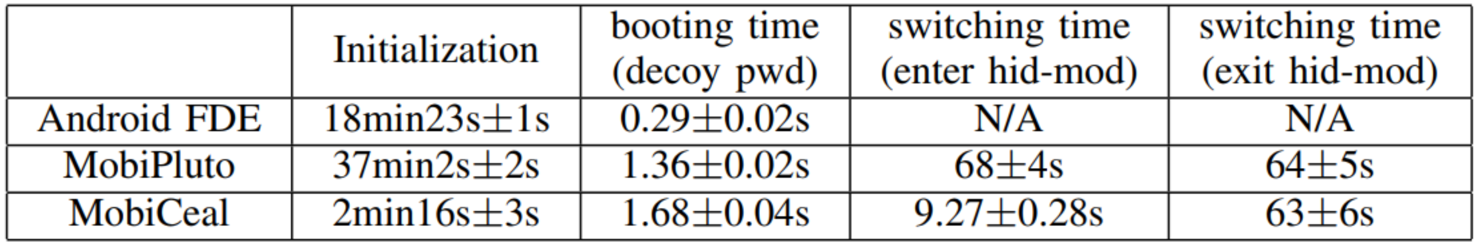
MOBICEAL PERFORMANCE

OVERHEAD COMPARISON. THE VALUES OF DEFY ARE FROM THE FIGURE 6 IN [33]. TEST ENVIRONMENT: DEFY: UBUNTU 13.04, SINGLE PROCESSOR, 4GB RAM, SIMULATED FLASH DEVICE; HIVE: ARCH LINUX X86-64, I7-930, 9GB RAM, SAMSUNG 840 EVO SSD; MOBICEAL: ANDROID 4.2.2, SNAPDRAGON APQ 8064, 2GB RAM, NEXUS 4 INTERNAL STORAGE.



	Ext4 (MB/s)	Encrypted (MB/s)	Overhead
DEFY	800	50	93.75%
HIVE	216.04	0.97	99.55%
MobiCeal	19.5	15.2	22.05%

INITIALIZATION TIME, BOOTING TIME, AND SWITCHING TIME.



	Initialization	booting time (decoy pwd)	switching time (enter hid-mod)	switching time (exit hid-mod)
Android FDE	18min23s±1s	0.29±0.02s	N/A	N/A
MobiPluto	37min2s±2s	1.36±0.02s	68±4s	64±5s
MobiCeal	2min16s±3s	1.68±0.04s	9.27±0.28s	63±6s

CONCLUSION

CONCLUSION

- MobiCeal is a practical PDE solution for mobile devices
- First block-layer PDE scheme that is resistant to multi-snapshot adversaries
- File system friendly and supports fast switching
- Prototype implemented on an LG Nexus 4 and tested on Huawei Nexus 6P
- Significantly lower overhead relative to other multi-snapshot defensive PDE systems