Mobiflage:

Deniable Storage Encryption for Mobile Devices

Adam Skillen and Mohammad Mannan

a_skil@ciise.concordia.ca

Concordia Institute for Information Systems Engineering
Concordia University
Montréal, Canada

NDSS 2013, San Diego, CA
February 26, 2013

Why do we need plausible deniable encryption (PDE)?

PDE can protect a user when apprehended with controversial data

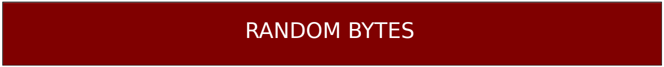
E.g., Syrian refugee smuggles evidence of atrocities under skin
`http://www.thestar.com/news/world/article/1145824`

- A user can feign compliance when coerced to reveal decryption keys/passwords
- Tools such as TrueCrypt provide PDE for desktop/laptop PCs
- PDE is arguably more important for mobile devices
- We explore inherent challenges by implementing PDE for Android

Mobiflage overview

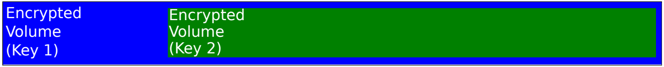
Two encrypted storage areas on physical medium

1. Encrypted disk appears as uniformly random bytes



RANDOM BYTES

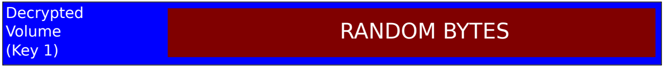
2. Encrypted volumes at different offsets with different keys
   Each volume is formatted to **consume all remaining space**



Encrypted Volume (Key 1)

Encrypted Volume (Key 2)

3. Decrypted outer volume appears to **consume the entire disk**
   Hidden volumes look like random bytes in decrypted free space



Decrypted Volume (Key 1)

RANDOM BYTES

Contributions

1. First mobile implementation

   Parts of Mobiflage design and implementation are Android specific

2. Despite simple theoretical design, the implementation has non-trivial complications

   (e.g., boot process, Flash storage, filesystems, etc.)

3. Explore sources of leakage/compromise inherent to mobile devices

   Several have not been analyzed for existing desktop PDE solutions

4. Sheds light on considerations beyond design requirements

   (e.g., FS and storage design, application permissions, communication channels)

Background on mobile storage encryption

1. File based encryption
    - Selected individual files are encrypted with unique keys
    - Keys are wiped from RAM when device is “screen locked”
    - BlackBerry and Apple iOS
      (iOS behaviour is file-based, actual implementation closer to FDE)

2. System/Full Disk Encryption (FDE)
    - Block ciphers act on individual disk sectors
    - Pre-boot authenticator to unlock/mount disk
    - On-the-fly (transparent to users/apps)
    - Key stays in RAM while “screen locked” (for background IO)
    - Google Android and Microsoft Windows Phone

Mobiflage modes

User boots into a given mode based on the supplied password

1. Standard Mode
   - Encryption without deniability
   - For day-to-day use of mobile device
   - Mounts “outer” volumes
2. PDE Mode
   - Encryption with deniability
   - Used to store data and later deny existence
   - Mounts “hidden” volumes

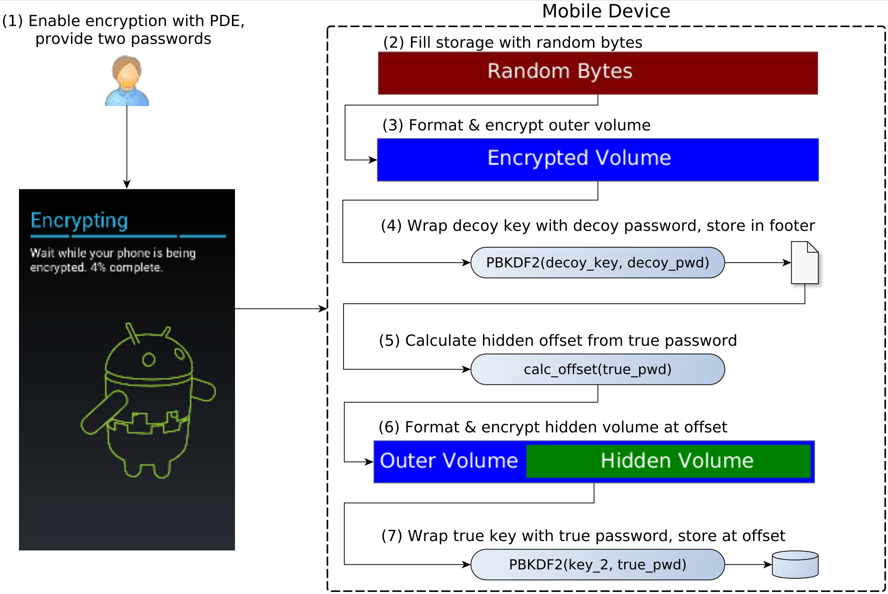
Apps and data in each mode are independent
Essentially two isolated installations are present

Storage considerations

1. Android has two storage locations for user data
   - `/data` – app packages and settings
   - `/sdcard` – file data (photos, music, maps)
2. Mobiflage creates hidden volumes for both mount-points, to facilitate hidden apps and hidden data
3. Hidden volumes consume 25% – 50% of SD card storage (actual size derived from hidden password)
4. Some devices have shared internal/external storage (i.e., no real/emulated SD card)

OS and kernel partitions are Read-Only and shared between Mobiflage modes

Mobiflage initialization



(1) Enable encryption with PDE, provide two passwords

Encrypting

Wait while your phone is being encrypted. 4% complete.

Mobile Device

(2) Fill storage with random bytes

Random Bytes

(3) Format & encrypt outer volume

Encrypted Volume

(4) Wrap decoy key with decoy password, store in footer

PBKDF2(decoy_key, decoy_pwd)

(5) Calculate hidden offset from true password

calc_offset(true_pwd)

(6) Format & encrypt hidden volume at offset

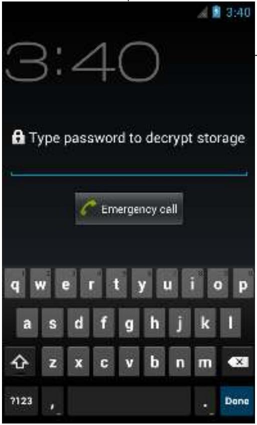
Outer Volume | Hidden Volume

(7) Wrap true key with true password, store at offset

PBKDF2(key_2, true_pwd)

Mobiflage usage – standard mode

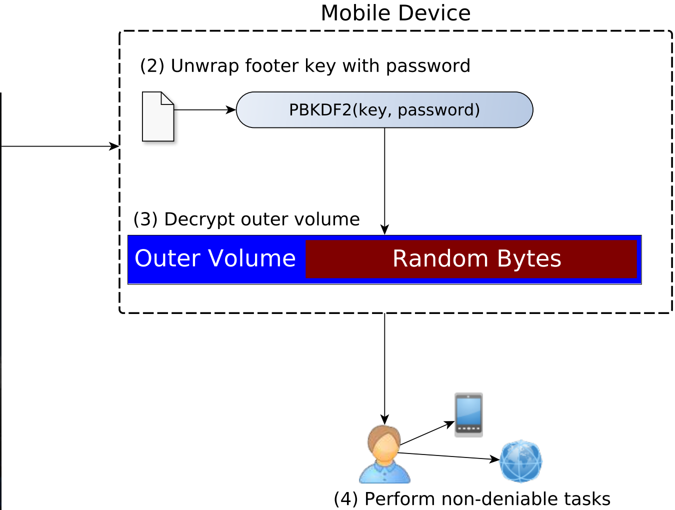
(1) Boot device and enter decoy password



3:40

3:40

Type password to decrypt storage



Mobile Device

(2) Unwrap footer key with password

PBKDF2(key, password)

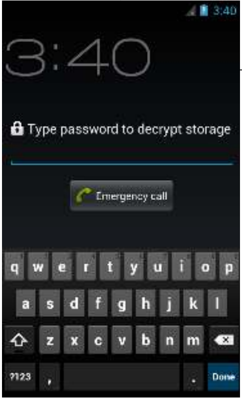
(3) Decrypt outer volume

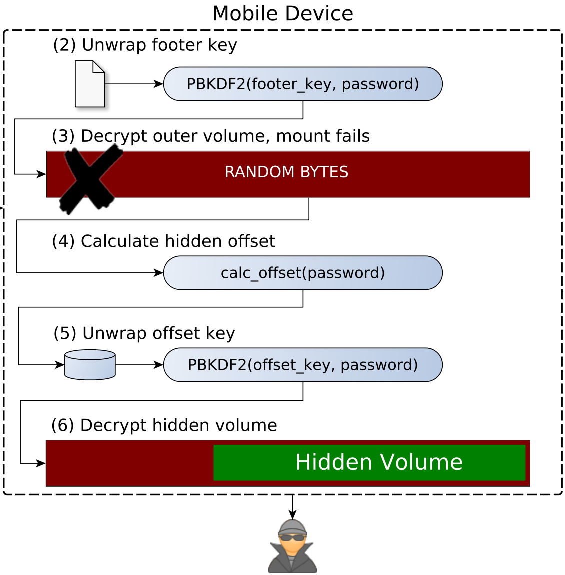
Outer Volume	Random Bytes

(4) Perform non-deniable tasks

Mobiflage usage – PDE mode

(1) Boot device and enter true password





Mobile Device

(2) Unwrap footer key

PBKDF2(footer_key, password)

(3) Decrypt outer volume, mount fails

RANDOM BYTES

(4) Calculate hidden offset

calc_offset(password)

(5) Unwrap offset key

PBKDF2(offset_key, password)

(6) Decrypt hidden volume

Hidden Volume

(7) Store data & perform tasks deniably

Enhancements to Android FDE

Mobiflage makes 3 changes to default Android FDE:

1. XTS-AES-256 cipher instead of CBC-AES-128

   Prevents known weaknesses in CBC for FDE[1]

2. Wipe external storage with random bytes

   Necessary to conceal hidden volumes

3. Enable encryption of removable storage

   Hidden volumes are stored on SD card

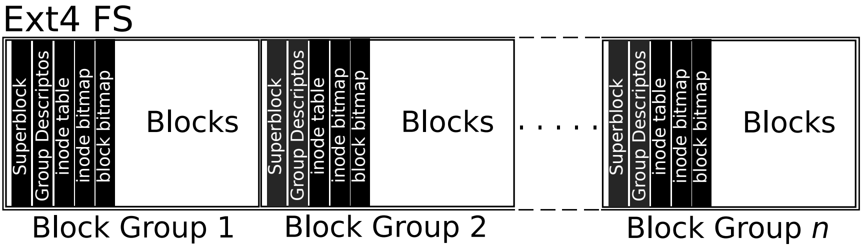
PDE is optional – users can still use default FDE
Changes are still applied to ensure PDE/FDE are indiscernible

[1] C. Fruhwirth. New methods in hard disk encryption. Technical report (July 2005). `http://clemens.endorphin.org/nmihde/nmihde-A4-ds.pdf`

Filesystem considerations

Android default FS is Ext4

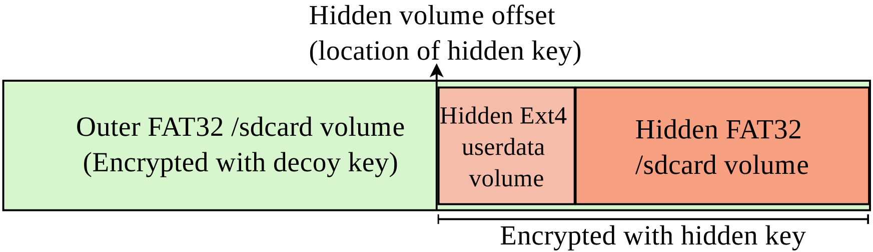
1. Volume divided into *block groups* and data blocks
2. Each group has meta-data structures
   (inode table, block bitmap, backup superblock, etc.)
3. Ext4 spreads directories (and hence files) across block groups
4. Hidden volumes can overwrite meta-data structures and file data



Mobiflage storage layout

Mobiflage uses FAT32 formatted *external* storage to hide volumes

- All meta-data at beginning of volume
- Remaining space is continuous data blocks



Sources of compromise addressed by Mobiflage

1. Flash storage
   - Data remanence
2. Leakage from software
   - Filesystem collisions
   - Logs, swap-space, temp files, (e.g., `/cache`, `/devlog`)
3. Crypto-primitives
   - FDE attacks – watermarking, *copy-and-paste*, etc.
   - Statistical deviations between RNG and cipher output

Remaining sources of compromise

4. Leakage from hardware
   - Flash wear-leveling (partial snapshots)
   - Device identifiers (e.g., MAC, IMEI)
   - Hardware component on-board cache (e.g., camera)

5. Password guessing
   - Only 2000 PBKDF2 (PKCS#5) iterations
   - Outer/hidden share salt value

6. Storage snapshots (e.g., border crossing)

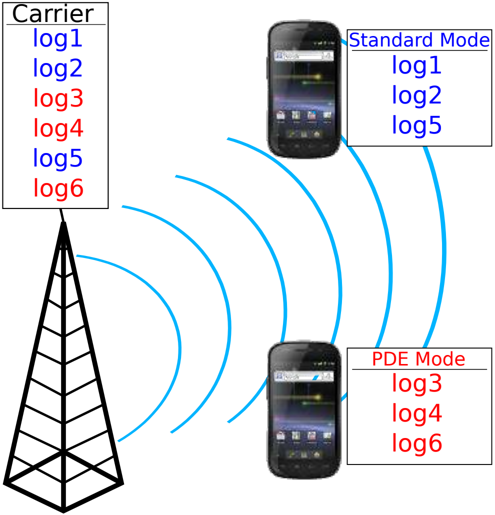
7. Other threats exist (malware, baseband attacks, etc.)

Collusion with carriers

Discrepancies between device logs and carrier/web service logs

Some defenses include:

1. Disable cell antenna
2. Spoof identifiers (IMEI, MAC)
3. Use anonymous SIM
4. Use public WiFi or Tor/VPN
5. Use pseudonymous accounts
6. This is **not** a comprehensive list!



Carrier
log1
log2
log3
log4
log5
log6

Standard Mode
log1
log2
log5

PDE Mode
log3
log4
log6

Mobiflage performance

1. Initialization time – two-pass wipe of external storage
2. Boot time – three invocations of PBKDF2 (negligible)
3. Power consumption – affects all FDE implementations
4. IO performance – DMA enabled hardware

Cipher-spec	Key-length (bits)	Speed (KB/s)		Speed reduction	
		Nexus S	Xoom	Nexus S	Xoom
Unencrypted	N/A	5880±260	4767±238	-	-
AES-CBC-ESSIV (Android 4.x)	128	5559±76	4168±186	5.46%	12.57%
AES-XTS-Plain64 (Mobiflage)	512 (256+256)	5288±69	3929±146	10.07%	17.58%

Observed read/write performance of external storage
(≈ 5% reduction over Android FDE)

Limitations

1. Currently requires removable SD card or internal FAT32 partition
2. User cannot choose size of hidden volumes
3. No clean solution to transfer data between modes
4. Denial-of-service: adversary can wipe/confiscate device
5. Only 50% of SD card can be used safely
6. Requires wide deployment so capability alone is not a red flag

Summary

1. Mobiflage hides encrypted volumes in external storage

   incurring a tolerable impact on performance

2. Requires conscientious users to maintain deniability:

   Mobiflage seeks to prevent known sources of leakage/compromise; but is not fool-proof

3. Different hardware profiles present non-trivial complications

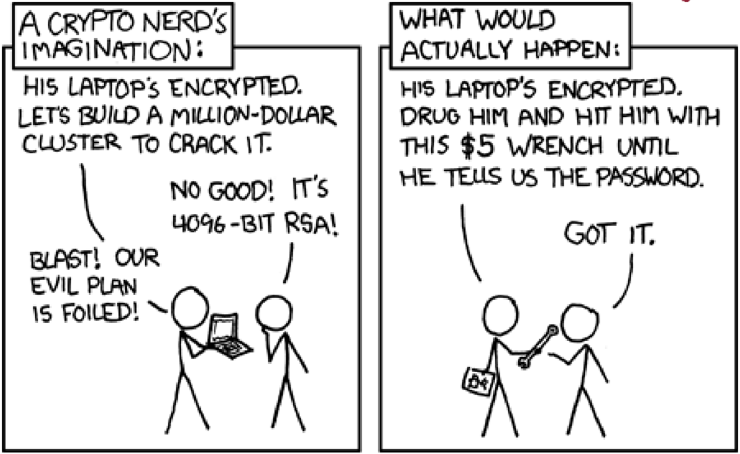
4. Unique challenges in mobile environment may lead to new design considerations

   (e.g., storage, filesystems, permission systems)

Mobiflage project website:

`http://users.encs.concordia.ca/~a_skil/mobiflage`

Questions?


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

Source: XKCD

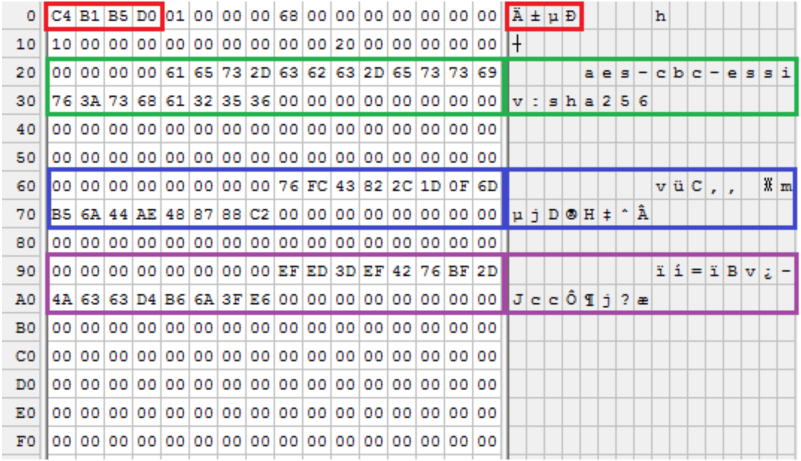
Offset calculation

Mobiflage offset is derived from deniable password:

$$\mathit{offset} = \lfloor 0.75 \times \mathit{vlen} \rfloor - \left(\mathrm{H}(\mathit{pwd} || \mathit{salt}) \bmod \lfloor 0.25 \times \mathit{vlen} \rfloor \right)$$

- Calculations are 512-Byte sector aligned
- Avoids new fields in Android footer
- Complicates large-scale dictionary attack campaign as compared to using a fixed offset (e.g., $\lfloor 0.5 \times \mathit{vlen} \rfloor$) (must capture at least 25% of each disk to mount attack)

Android FDE footer



```
 0  C4 B1 B5 D0 01 00 00 00 68 00 00 00 00 00 00 00  Ä±µÐ    h
10  10 00 00 00 00 00 00 00 00 00 20 00 00 00 00 00  ┼
20  00 00 00 00 61 65 73 2D 63 62 63 2D 65 73 73 69      aes-cbc-essi
30  76 3A 73 68 61 32 35 36 00 00 00 00 00 00 00 00  v:sha256
40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60  00 00 00 00 00 00 00 00 76 FC 43 82 2C 1D 0F 6D          vüC‚,  ☼m
70  B5 6A 44 AE 48 87 88 C2 00 00 00 00 00 00 00 00  µjD®H‡ˆÂ
80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
90  00 00 00 00 00 00 00 00 EF ED 3D EF 42 76 BF 2D          ïí=ïBv¿-
A0  4A 63 63 D4 B6 6A 3F E6 00 00 00 00 00 00 00 00  JccÔ¶j?æ
B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Magic Number

Cipher Spec

Key (16 Bytes)

Salt (16 Bytes)

Android storage volumes

Typical volumes found on common Android devices:

Volume	Mount point	Mode	Description
Boot	N/A	N/A	Bootloader and kernel image
Recovery	N/A	N/A	Recovery tools and backup kernel
System	`/system`	RO	OS binaries, Dalvik VM, etc.
Cache	`/cache`	RW	Temporary space for OS and apps (e.g., OTA updates and downloaded .apk)
Device log	`/devlog`	RW	Persistent system logs
Userdata	`/data`	RW	Apps and settings
External	`/mnt/sdcard` or `/storage/sdcard0`	RW	App and user data (e.g., photos, maps, music)