Powerflexible Cryptography
With Python and Flightbox

FOSDEM 2024
The Struggle

Some data is confidential and must be strongly protected
Server or device data might be compromised at any moment due to:

- **Zero-day exploit of the OS**
- **Obsolete software dependencies**
- **Privilege escalation bug**
- **Unprotected endpoint**
... or due to:

- Poorly protected backups
- Theft of admin login/password
- Dishonest system administrators
- Phishing and other human engineering
- Etc.
We need something better than that…
Which won't explode during a random software update
We need Cryptography
Cryptography Basics

**Encryption**: Use a "cipher" algorithm to transform a "plaintext" media content into an incomprehensible but decryptable "ciphertext"

**Hashing**: Compute a fingerprint of a content

**Signing**: Authenticate/timestamp a content
Symmetric cipher: Principles

A chest with a keyhole

+ a secret key (or symkey) to open/close it

100% random

... or a CRYPTEX
Examples: AES, Chacha20, Blowfish...

Typically, tons of two-way maths/boolean/shift operations between content and secret key

Can work on byte blocks, or byte-by-byte

High performance, often hardware-accelerated...

Different modes of operation (CBC, XTS…), optional integrity tags (MACs)…
Most famous: AES
Performance: GREAT
Asymmetric cipher: Principles

A chest with a padlock

Uses a key pair

- Private key: actual key
- Public key: padlock
Asymmetric cipher: Details

Examples: RSA-OAEP, ECC/ECIES...

Leverages math problems like the factorization of integers, elliptical curves...

Low performance, works only on small contents (fingerprints, other keys...)

Asymmetric cipher: TLDR!

Most famous: RSA
Performance: BAD
Digital signature

A seal of authenticity (timestamped?)

Typically applied on a fingerprint of the content

Proof of anteriority, but hardly posteriority

Uses a keypair too

- Private key: seal stamp
- Public key: verifier

Examples: PSS, DSS...
A weird way of separating a secret

- Content is split into $N$ parts (or shards)
- You need any $M$ parts ($M \leq N$) to restore it
Shared secret of Shamir (2)
Real life: Hybrid Encryption Scheme

- Encrypt (big) content with a symmetric cipher
- Then encrypt its secret key with an asymmetric cipher
- Sign the plaintext and/or ciphertext
OK, ENOUGH CRYPTO PRIMITIVES FOR TODAY
MAIN POINTS TO KNOW ABOUT CRYPTOGRAPHY
Cryptography is DANGEROUS
• Do not try to implement “primitives” yourself, trust cryptoanalysts and popular libs

• The order of operations (cipher, sign, mac…) is important *(Stackoverflow is your friend...)*

• Never use the same key for different purposes (and never reuse initialization vectors, nonces...)

• Ensure your source of randomness is good

• Sometimes it gets metaphysical (ECC Curves...)*
Cryptography is STRONG
Need to brute-force a symmetric cipher?

*Not enough energy in the known Universe*
Need to break an asymmetric cipher?
Solve one of the hardest maths problems in History
TIME FOR INNOVATION
Witness Angel Project

A “black box” for humans, securing proofs in case of future crimes

Goal 1: No more judicial errors, nor “It’s your word against theirs”

Goal 2: Preserve our privacy!
What’s a typical “Witness Angel” recorder?

- By default, nobody can read what has been recorded and encrypted
- Concept of “write-only device”
Witness Angel Device: Revelation

But when a decryption operation is launched:

- Device owner MUST grant authorization

- Then, among 5 external key guardians, 3 at least MUST grant authorization

- BUT if the owner is killed, then we query a Special Assembly of 6 key guardians, and 4 of them MUST grant authorization
Our technical quest

• We want a very high security level, with multiple ciphers involved

• We want multiple key guardians
  • Some of them mandatory for decryption (ex. the device owner)
  • Some of them optional for decryption (ex. remote trusted third parties)

• We want “E.B.A.C”
  i.e. Encryption Based Access Control
Chain (a)symmetric ciphers

- Use different cipher algos and secret keys
- Result: a logical AND between key guardians
Step 2

Use a shared secret algorithm

- Choose a threshold $M$, for $N$ key guardians
- Give each key guardian a shard

Logical “OR“ if $M==1$
Logical “AND“ if $M==N$
Shared secret if $1<M<N$
Putting it all together

• Data passes through several symmetric ciphers

• Secret keys and shards recursively go through:
  • Symmetric ciphers
  • Asymmetric ciphers
  • Shared secrets

• Signatures can be applied at every stage
Example cipher tree

Cleartext → AES-CBC → CHACHA20 → Ciphertext

AES-CBC

RSA (me)

RSA (mom)

CHACHA20

AES-EAX

RSA (John)

RSA (Jane)

AES-CBC

RSA (Jess)

ECC (Jill)

RSA (mom)
Example cipher tree

Cleartext → AES-CBC → CHACHA20 → Ciphertext

- AES-CBC
  - RSA (me)
  - RSA (mom)

- CHACHA20
  - AES-EAX
    - RSA (John)
    - RSA (Jane)
    - RSA (Jess)

- AES-CBC
  - ECC (Jill)

Riddle me this!
LET’S CALL IT... FLIGHTBOX
Flightbox workflow (1)

- Each key guardian generates own keypair(s), and publishes own public key(s) to web registries or USB storages
- A "cryptoconf" describes the whole encryption pipeline (e.g. in Extended-JSOn format)
- Recorder devices recursively encrypt their content using relevant public keys
• The resulting "cryptainers" contain the ciphertext and its metadata (cryptoconf, random UUIDs, integrity tags, signatures...)

• They can be stored remotely, over-encrypted...

• When a decryption is allowed, key guardians:
  • Decrypt the keys/shards dispatched by a supervisor
  • Send them back via a secure channel (PGP-style)
HOW TO USE FLIGHTBOX
We provide a reference implementation in Python

WACRYPTOLIB

Audited in October 2023,
thanks @Raccoon
Wacryptolib API

- Key generation and serialization (PEM format)
- Key Guardian management
- USB device support
- JsonRPC client with error translator
- Sensor device handlers (push/pull modes)
- Cryptainer encryption, storage and decryption
Use the Python API

- `data = b"abcd12345"
- `cryptoconf = load_from_json_file("mycryptoconf.json")
- `storage = CryptainerStorage("~/mydir/", <keystore-pool>)
- `storage.encrypt_file("rec", data, cryptoconf=cryptoconf)
- # File ",~/mydir/rec.crypt" should now exist

→ Import necessary key guardians into <keystore-pool> first
→ We can also directly use encrypt_xxx() functions
→ Encryption is either one-shot or streamed chunk-by-chunk
Use the Cmd Line Interface (1)

$ flightbox foreign-keystore import --from-usb

$ flightbox foreign-keystore list

<table>
<thead>
<tr>
<th>Keystore UID</th>
<th>Owner</th>
<th>Public keys</th>
<th>Private Keys</th>
<th>Created at (UTC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0f0c0988-80c1-9362-11c1-b06909a3a53c</td>
<td>Maaa</td>
<td>7</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0f702b36-1a14-fc7e-79f8-7ea4c47eef38</td>
<td>John Doe</td>
<td>3</td>
<td>0</td>
<td>2023-10-16 08:06</td>
</tr>
</tbody>
</table>
Use the Cmd Line Interface (2)

$ flightbox cryptoconf generate-simple [...] > conf.json

$ flightbox cryptoconf summarize conf.json

Data encryption layer 1: AES_CBC
Key encryption layers:
  RSA_OAEP via trustee 'local device'
Signatures: None
Data encryption layer 2: CHACHA20_POLY1305
Key encryption layers:
  Shared secret with threshold 1:
    Shard 1 encryption layers:
      RSA_OAEP via trustee 'authenticator 0f0c0988-80c1-9362-11c1-b06909a3a53c'
    Shard 2 encryption layers:
      RSA_OAEP via trustee 'authenticator 7a25db2c-4c4e-42bb-a064-8da2007a4fd7'
Signatures: None
Use the Cmd Line Interface (3)

$ flightbox encrypt data.mp4 --cryptoconf conf.json

$ flightbox cryptainer list

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Offloaded</th>
<th>Created at (UTC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>readme.rst.crypt</td>
<td>102 KB</td>
<td>X</td>
<td>2023-10-16 08:06</td>
</tr>
</tbody>
</table>
Use the Cmd Line Interface (4)

$ flightbox cryptainer purge --max-age 30

$ flightbox cryptainer validate data.mp4.crypt

$ flightbox cryptainer decrypt data.mp4.crypt

Decryption report:
[
  {
    'entry_criticity': 'INFO',
    'entry_exception': None,
    'entry_message': 'Skipping retrieval of remotely predecrypted symkeys (requires requestor-uid and gateway urls),
    'entry_nesting': 0,
    'entry_type': 'INFORMATION'},
  {
    'entry_criticity': 'INFO',
    'entry_exception': None,
    'entry_message': 'Starting decryption of payload cipher layer 1/1 (algo: ' 'AES_CBC)',
    'entry_nesting': 0,
    'entry_type': 'INFORMATION'}]
Use the W.A Recorder software

Available as standalone program or Raspberry Pi image

Thanks Kivy!
BUT WHAT ABOUT KEY GUARDIANS?
Authenticator for Key Guardians

Thanks Kivy again!
Our cute prototypes

W.A Handbag

W.A Dashcam

W.A Network Video Recorder
Our next steps

- **C implementation of Flightbox (for microcontrollers)**

- **Highly optimized audio/video recorder chip**
  - Must last 24h on tiny battery
  - BIG challenge
Our next steps

- C implementation of Flightbox (for microcontrollers)
- Highly optimized audio/video recorder chip
  - Must last 24h on tiny battery
  - BIG challenge

Help & Contacts are Welcome
Flightbox use cases

- Securing unattended recordings for judicial proofs (e.g. Witness Angel devices)
- Protecting enterprise credentials
- Protecting sensitive couple/family docs
  
  See iCloud nude leaks, identity thefts...
- Sending files very securely
- It’s your shot!
Thanks for your attention!

- Any questions?

- Feedbacks & contributions are welcome
  - https://witnessangel.com
  - https://www.instagram.com/witnessangel_fr/
  - https://www.linkedin.com/company/witnessangelfr/
  - https://github.com/WitnessAngel/

Say hi to Shivi and its comic!