KeePassNFC Applet
Code Improvement and Profiling

UCO 476354 : VIKAS LAMBA
UCO 476366 : AKHILESH KUMAR SONI
UCO 477950 : MARCO CIOTOLA

OVERVIEW

- Security Issues
- Best Security Practises
- Improvement in Code
- Profiling of Essential Functions

Security Issues

Integrity and Protocol

Read of sensitive unencrypted data

General Oracle

Padding Oracle Attack

BEST SECURITY PRACTISES

- ✓ Use of standard API
- ☒ -> ✓ Initialisation and clearing of sensitive data
- ✓ Sensitive data storage
- ☒ -> ✓ Use of exceptions
- ✓ Use of constructor
- ☒ -> ✓ Fault induction check
- ☒ -> ✓ Authorisation check
- ☒ -> ✓ Memory Initialization and bound checking

Code Improvement

Authentication based on PIN

- Master PIN
- User PIN
- Number of Trials

Memory Initialisation and Clearing of Sensitive data

Fault Induction Checks

Complete Commenting of Source code

Integration of Applet Functionality and Security Features in Readme

- Applet Configuration and Usage
- APDU Formats and Error Codes
- PIN Management
- Timings
- Future Work

Code Improvement

- Fixing of Security Flaws
- Partial State Management (User PIN verification + exceptions)
- Implementation of supported cryptographic algorithm (AES-256)
- Test Cases
- Profiling of relevant functions

Profiling

Whole-command timings

NXP J2E 081

- Card Key generation (2048b RSA) can take a minute
- Block (128B) decryption is fast (<100 ms), but still unusable (2.5 mins for whole db ~200KiB)
- Setup of Password Key (AES-256) slower than setup of Transaction Key (AES-128), even including initialization of ciphers

JCProfile

- Card Key generation: random operations bring to inconsistent results.
- Other operations (all dependant on Password Key) require multiple APDUs requests for setup.
- Tried editing PerfTests class to issue multiple APDUs for setup, but still insufficient
  - data encryption with card key is needed before sending…
  - even previous setup with no cleaning of sensitive data seems to help...
- Still, at least block decryption would be interesting to measure

Profiling Results

Header	Avg timing (ms)	Timing range (ms)	Notes
9072	7	7 - 8	(get lock reason/remaining PINs), direct read
9074	7	7 - 9	(get version), direct read
A080	37	37 - 39	(verify Master PIN)
A081	33	33 - 34	(set Master PIN)
A082	37	37 - 38	(verify User PIN)
A083	33	32 - 34	(set User PIN)
B070 010000	13	13 - 14	(get Card Key exponent) Supposed one execution every usage.
B070 020000	44	44 - 45	(get Card Key modulus - first part) Supposed one execution every usage.
B071	714	714 - 716	(set password key) Supposed one execution every long time.
B072	699	698 - 700	(set transaction key) Supposed one execution every usage.
B073 P1!=80	96	95 - 97	(decrypt one block) Decrypting 128 bytes unique block.
B075	21660	5557 - 54592	(generate card key) Supposed one execution every long time.
B076	14	14 - 15	(write to scratch) Writing 16 bytes to scratch ~encrypted transaction key.
B076	18	18	(write to scratch) Writing 32 bytes to scratch ~encrypted password key.
B076	26	26 - 27	(write to scratch) Writing 64 bytes to scratch, more than needed by keys.

Future Scope

Applet improvements:

- HMAC encrypted communication
- Drop usage of 01/02 to show success/failure of commands, and just use SW codes
- Offer method to know the maximum amount of data that can be sent each time
- Support for long APDUs
- Support for encryption of database
- Support for ChaCha20/Twofish/other algorithms supported by KeePass or its plugins
- Manage states of the applet to prevent wrong commands to be called
- Setting initial Master/User PINs during installation with a payload
- DH-like key derivation instead of PIN/transaction key (with shared secret set during installation)
- Optimization to speedup execution on real cards

Repository improvements:

- Leave a single build system, or correctly maintain/link both
- Provide multiple client implementations (different simulators/libraries)
- Run tests using different simulators and directly calling methods
- Add tests to cover all the applet code. Also, test and check coverage of clients.

Thanks for Attention !!