



OTA Flash Forum
Refreshing the Mobile World

Firmware Update Security

Hagai Bar-El, Discretix
www.discretix.com

The Need to Protect Updates

- The need is trivial, actually...
- Functionally, the device is its firmware
- Firmware compromise invalidates all security methods by all application
 - ...and thus FW update security upper-bounds the security level of the entire system
- Firmware hacks, for being low-level, lead to the most comprehensive set of exploit capabilities
 - These exploits are beyond the scope of the application privileges and also are cross-app.

Protecting Update Files

- Relies on the following technologies:
 - Encryption
 - Digital Signatures for Authenticity and Integrity
 - Both can be done using
 - Symmetric algorithms
 - Asymmetric algorithms
- Can be implemented either at the application layer or at the transport layer

Update Files: Encryption

- Assures secrecy
 - Not always required, but must be implemented
- Methods: Symmetric, Asymmetric and Hybrid
- Does not provide integrity
- Can be implemented at
 - Application level
 - Transport level

Authenticity & Integrity: Symmetric

- Using block ciphers
- Using one-way keyed functions

Advantages	Disadvantages
Easier to implement	Require a shared secret, which is hard to do right
Faster to run	Does not allow non-repudiation, which is not necessarily a significant limitation in the DM case
Low space requirement both for code and data	

Authenticity & Integrity: Asymmetric

- By asymmetric encryption algorithms together with hash functions

Advantages	Disadvantages
Ease key exchange and server-side key management for unique keys	A more complex infrastructure is required
	More complex (and slow) code is required
Can provide non-repudiation as well	Larger space requirements

Symmetric vs. Asymmetric

	Symmetric	Asymmetric
Implementation Complexity	Code :Simple Keys: Difficult	Code: Complex Keys: Simple
Space Required	Code: Small Keys: Small	Code: Large Keys: Large
Execution Time	Fast	Slow

Application Layer vs. Transport Layer

	Application	Transport
Implementation	Difficult	Easy
Reusable cred.	App.Reusable	Separate
Code length	Minimal	Longer
RAM requirement	Minimal	Larger
Confidentiality	End-2-End	Transport
Authenticity	Retained	Lost
Hack risk	Higher	Lower
Bearer agnostic	Yes	No

Receipts: Why and How

■ The need

- User may oppose some of the updates
 - Removal of SW
 - Fix of bugs for which he/she has useful exploit code for
 - Revocation of services, statuses, etc.
- Legal circumstances may at times require proof of execution

Receipts: Why and How

■ Secure Implementation Guidelines

- Regular ACK is not enough
- Digital signature is a must
- Cryptographically bound to the update message, not to its content
- Optimally, and if applicable - also bound to object result
- Time bound by TTS on device and/or on server

Biggest Challenges

■ Key Generation

- Symmetric: Can be generated by client or server, so server does
- Asymmetric: Server generates or generated on-board and certified

Biggest Challenges

- Enrollment and Key-Exchange
 - Based on pre-provisioning
 - Based on proximity and lifecycle state awareness of the device
 - Semi-secure delivery using semi-authenticated mechanisms
 - Possibly assisted by semi-secure off-band information – simple solution but with security pains

Biggest Challenges

- Maintaining Keys Securely
 - Within agent
 - Plain constant
 - Using anti-RE manipulations
 - Using encryption
 - Using device hardware
 - Using SIM/SC
 - Using user – very problematic
- ...and the inevitable need for a secure environment

In Conclusion

- There is no one-size-fits-all formula in DM security
- Every vendor acts as he sees fit and according to its constraints to obtain as much security as possible
- Yet, there are some obvious “no-no’s” that must be adhered to so the system does not collapse along with the entire platform which was built on top of it



OTA Flash Forum
Refreshing the Mobile World

Thank You!

hagai.bar-el@discretix.com

www.discretix.com