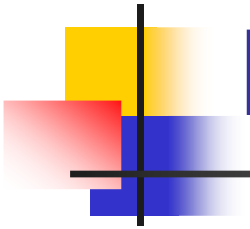
Security vulnerability found in On-Board Credentials validation activity

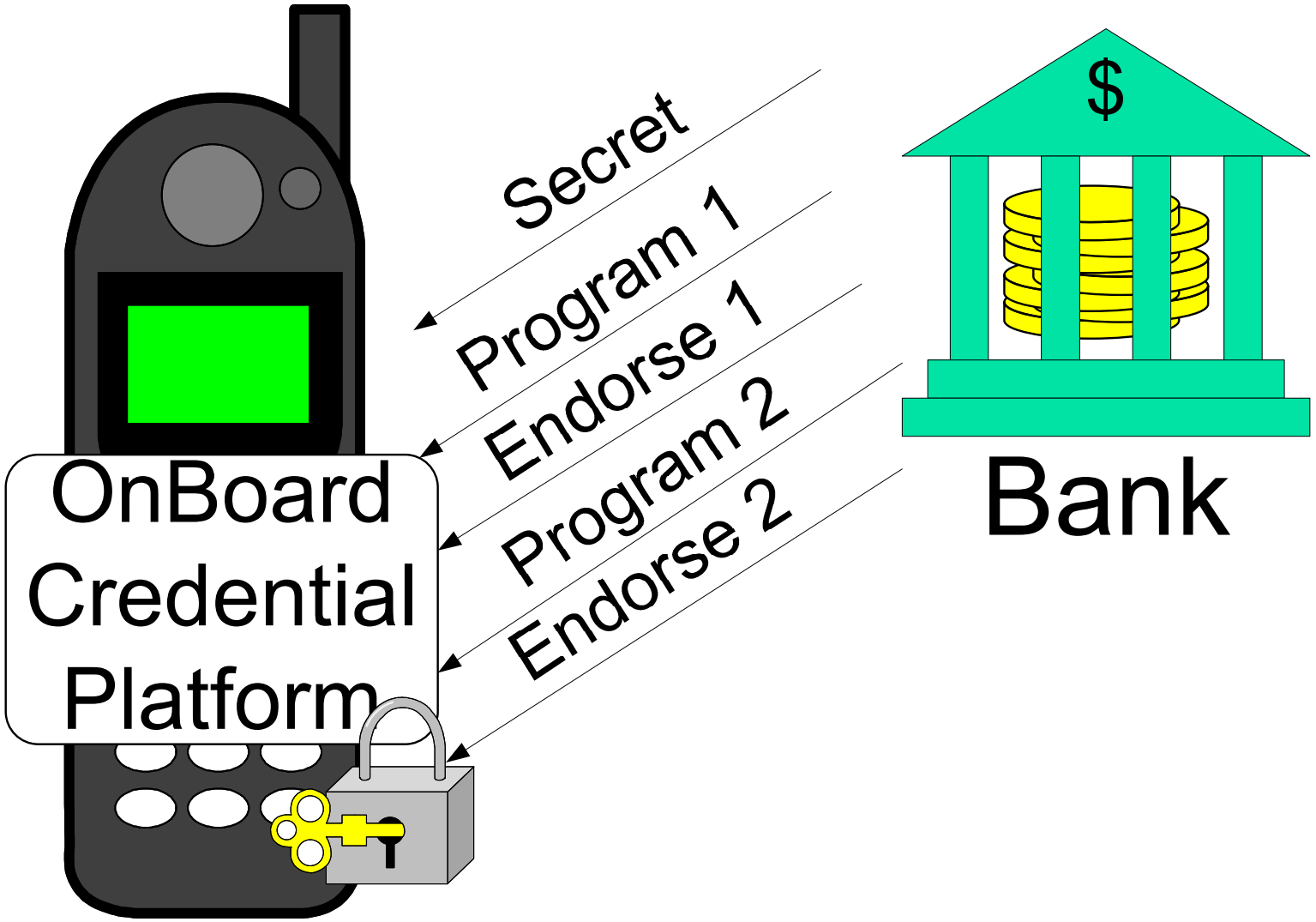
Afanasyeva Alexandra, SUAI
Ekberg Jan-Erik, NRC
alra@vu.spb.ru



Background

- On-Board Credentials is a framework by Nokia Research Center for secure execution of third-party credentials on e.g. embedded devices
- The framework also includes a provisioning protocol by which any third part can provision credentials onto the platform
- A Fruct project (SUAI) was set up to analyze parts of the framework, among other things the provisioning protocol
- A vulnerability was found by which the integrity of the provisioned 3rd-party credentials programs could be compromised
- The implementation of ObC was changed to correct the found security vulnerability

OnBoard Credential Platform



Provisioning Protocol

- Goal: To allow ***any*** entity to provision secure data and program on the device.
- Steps:
  - User send to device
    - *Init* = header || Enc_{PK_D}(FK)
    - *Xfer* = AE_{FK}(header, <secret>)
    - *Xfer* = AE_{FK}(header, <program>) or <program>
    - *Endorse* = AE_{FK}(header, H(<program>))
  - Device:
    - If program wants to read secret it should have appropriate Endorse

Attack on Provisioning Protocol

- Intruder
  - sniffs Endorse = AE(FK, header, H(<program1>))
  - generates program2 for disclosing of secret
  - generates program2 in such a way that H(<program2>)= H(<program1>)

Problem statement:

Find second pre-image for hash-function used in provisioning protocol

Cryptographic Primitives

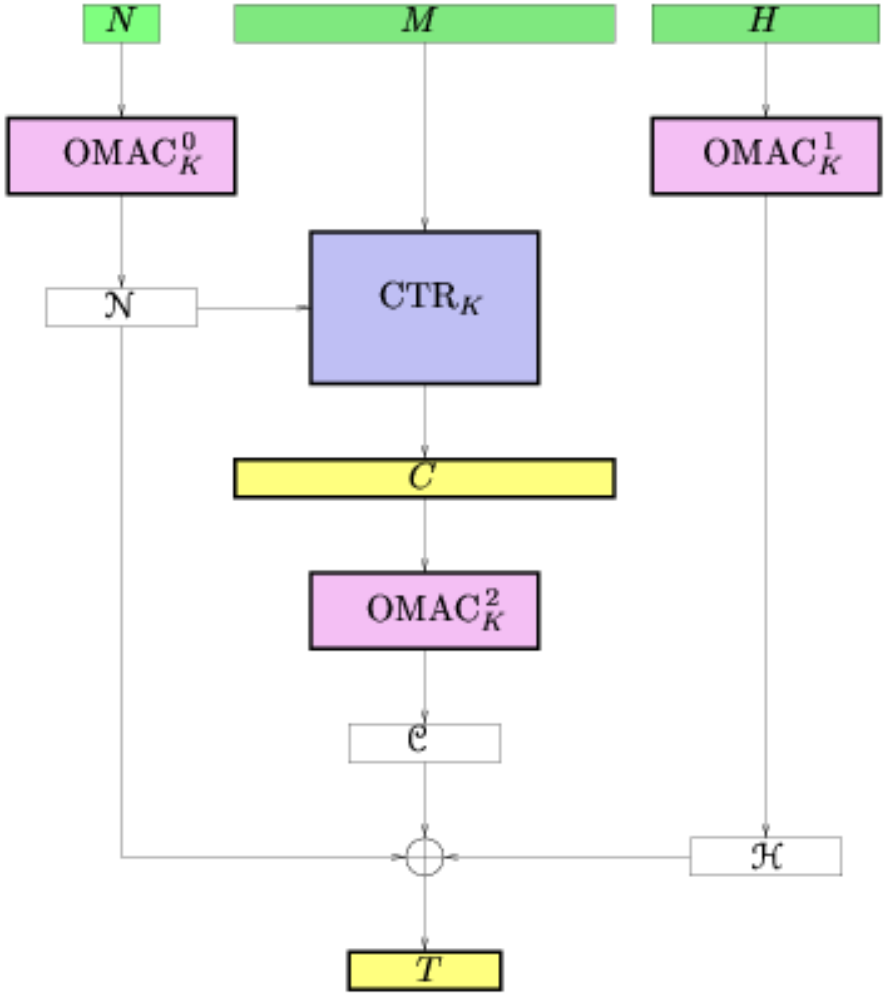
- There is a restriction on size of code which implements all cryptographic functions (encryption, hash, …)
- So, only one crypto-primitive (AES-EAX) was used as basis for all these functions
  - Authenticated encryption

$$AE = AES - EAX(key, header, nonce, data)$$

  - Hash function

$$HASH = AES - EAX(public\_hash\_key, data)$$

AES-EAX Encryption Mode



- X = {x_1,x_2, … x_n}
- OMAC$_K$(X)={

$$x_n = x_n \oplus pad, c_0 = 0$$

for i = 1 to n

$$c_i = AES_K^{ENC}(x_i \oplus c_{i-1})$$

return(c_n) }

- CTR$_K$(X,N) = {

for i = 1 to n

$$c_i = x_i \oplus AES_K^{ENC}(N + i)$$

return(c_1,c_2,…,c_n) }

Hash Function Vulnerability

- Given:
  - M, Hash(M)=T
  - Arbitrary

  $M' = \{m_1, m_2, \ldots m_n\}$

- Find:
  - m_{n+1} :

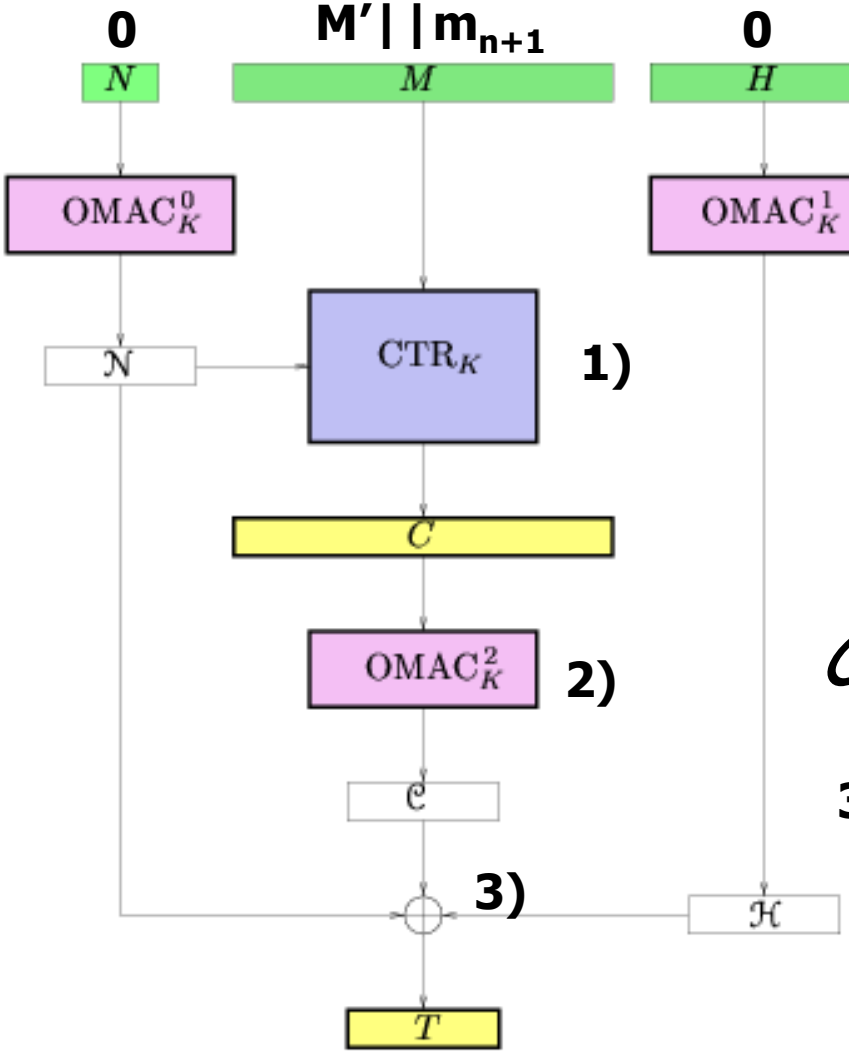
  $Hash(M' || m_{n+1}) = T$

The only restriction on M' is $|M'| = n*128$

Finding m_{n+1}

$$m_{n+1} = \left(AES_{HK}^{DEC}\left(T \oplus OMAC_{HK}^{0}(0) \oplus OMAC_{HK}^{1}(0)\right) \oplus OMAC_{HK}^{2}(CTR_{KH}(M', \mathcal{N})) \oplus pad \right) \oplus$$

$$\oplus AES_{HK}^{ENC}\left(\mathcal{N}+n\right)$$



1)

$$C = \left(CTR_{KH}\left(M', \mathcal{N}\right)\right) \|$$

$$\| AES_{HK}^{DEC}\left(T \oplus OMAC_{HK}^{0}(0) \oplus OMAC_{HK}^{1}(0)\right) \oplus$$

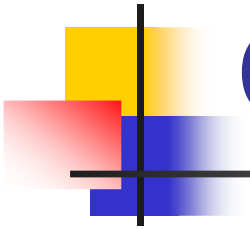
$$\oplus OMAC_{HK}^{2}\left(CTR_{KH}\left(M', \mathcal{N}\right)\right) \oplus pad$$

2)

$$\mathcal{C} = T \oplus OMAC_{HK}^{0}(0) \oplus OMAC_{HK}^{1}(0)$$

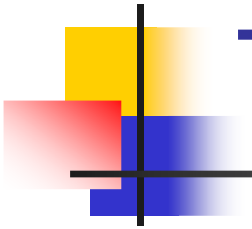
3)

$$\text{Hash} = T$$



Conclusions

- A good, and flawless security design in the end benefits the customer
  - independent design validation
  - validating implementations prior to deployment can find problems before they occur in the field
  - build in-field upgradeability
- The correct use of cryptographic primitives is often essential



Thank you!

Q & A