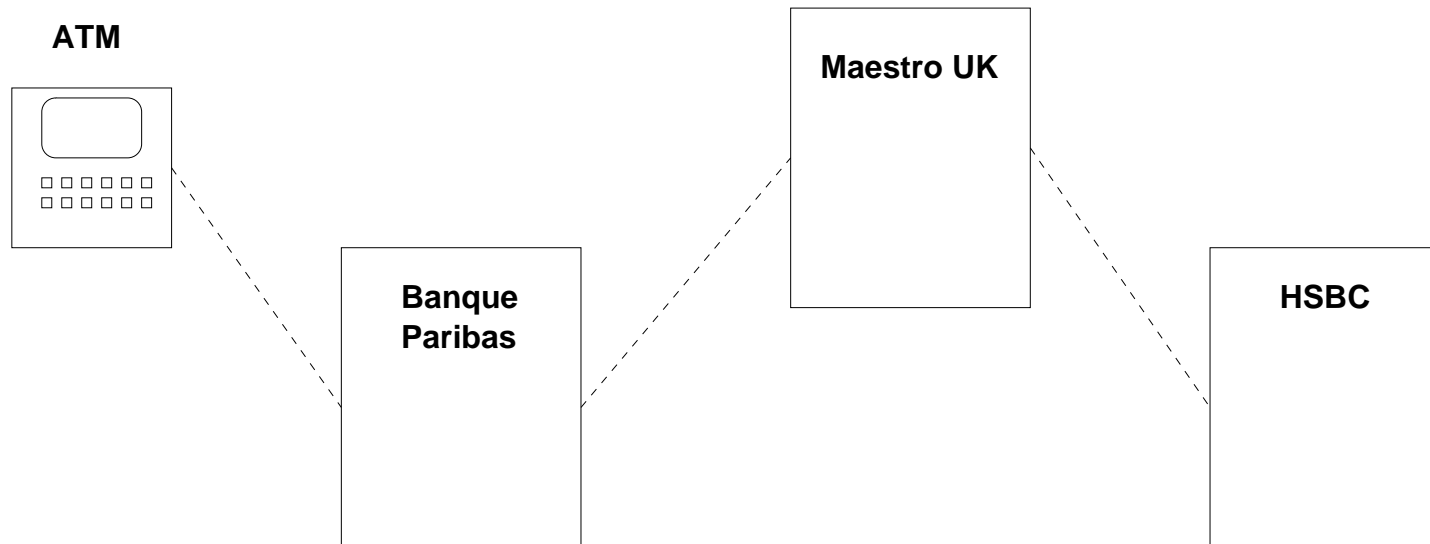

Model Checking Security APIs

Gavin Keighren and Graham Steel



 School of
informatics

Automated Teller Machines



Hardware Security Modules



Criticality of Key Management

Original PIN (IPIN) derived by:

3DES encrypting 0000AAAAAAAAAAAA
under a PDK (PIN Derivation Key)

Where the As are your account number (PAN)

Decimalise result

0123456789ABCDEF

0123456789012345

$\text{PIN} = \text{IPIN} + \text{Offset} \pmod{10}$ each digit

Offset NOT secure!



km
data, pin, imp, ...

data, pin, imp,...

{ d1 } $_{km \oplus data}$

{ pdk1 } $_{km \oplus pin}$

CCA API - Examples

Encrypt Data:

Host → HSM : $\{ d1 \}_{km \oplus data}$, message

HSM → Host : $\{ message \}_{d1}$

Verify PIN:

Host → HSM : $\{ PINBlock \}_{p1}$, PAN, $\{ pdk1 \}_{km \oplus pin}$,

OFFSET, $\{ p1 \}_{km \oplus ipinenc}$

HSM → Host : yes/no

Importing Key Parts

‘Separation of duty’

Key $k = k1 \oplus k2$

Host \rightarrow HSM : $k1, TYPE$

HSM \rightarrow Host : $\{ k1 \}_{km \oplus kp \oplus TYPE}$

Host \rightarrow HSM : $\{ k1 \}_{km \oplus kp \oplus TYPE}, k2, TYPE$

HSM \rightarrow Host : $\{ k1 \oplus k2 \}_{km \oplus TYPE}$

Usually used to import a ‘key encrypting key’ ($\{ KEK \}_{km \oplus imp}$)

Importing Encrypted Keys

Exported from another 4758 encrypted under $KEK \oplus TYPE$

Key Import:

Host \rightarrow HSM : $\{ KEY1 \}_{KEK \oplus TYPE}, TYPE, \{ KEK \}_{km \oplus imp}$

HSM \rightarrow Host : $\{ KEY1 \}_{km \oplus TYPE}$

Attack (Bond, 2001) (part 1)

PIN derivation key: $\{ \text{pdk} \}_{\text{kek} \oplus \text{pin}}$

Have key part $\{ \text{kek} \oplus k_2 \}_{\text{km} \oplus \text{imp} \oplus \text{kp}}$ for known k_2

Host \rightarrow HSM : $\{ \text{kek} \oplus k_2 \}_{\text{km} \oplus \text{kp} \oplus \text{imp}}, k_2 \oplus \text{pin} \oplus \text{data}, \text{imp}$

HSM \rightarrow Host : $\{ \text{kek} \oplus \text{pin} \oplus \text{data} \}_{\text{km} \oplus \text{imp}}$

Attack (Bond, 2001) (part 2)

Key Import

Host \rightarrow HSM : $\{ \text{pdk} \}_{\text{kek} \oplus \text{pin}}, \text{data}, \{ \text{kek} \oplus \text{pin} \oplus \text{data} \}_{\text{km} \oplus \text{imp}}$

HSM \rightarrow Host : $\{ \text{pdk} \}_{\text{km} \oplus \text{data}}$

Encrypt data

Host \rightarrow HSM : $\{ \text{pdk} \}_{\text{km} \oplus \text{data}}, \text{pan}$

HSM \rightarrow Host : $\{ \text{pan} \}_{\text{pdk}} (= \text{PIN!})$

Modelling in CL-Atse

- Suitable because supports XOR unification
- One role for each command
 - otherwise optimisation takes too long
- Only terms of type *message* can be XORed
 - so untyped mode used
- Initial optimisation step usually disabled
 - apart from a few examples, made performance worse

Rediscovering Known Attacks

Rediscovery of:

- Typecast attack shown above
- Import/Export loop attack
- IBM's variation

Various different models...

Times compare favourably to Youn et. al. 2005 (U. of Cambridge Tech. Report)

– but not for IBM attack

IBM Recommendations

Published in response to attacks

1. Use asymmetric key crypto for key import
2. More access control
 - security officers access fewer commands
3. Procedural controls to check entered key parts

Not to be confused with Bond's own recommendations

1. Use Asymmetric Crypto

Procedure:

1. Key pair generated on destination 4758
2. Public key registered at source by two separate individuals
3. KEK generated at source 4758, and exported under public key
4. KEK imported at destination 4758

Our model assumes intruder about to carry out step 4,
PKA_SYMMETRIC_KEY_IMPORT command added.

Attack found by CL-AtSe...

The Attack

1. Intruder encrypts known value k under registered public key
2. Intruder imports k as an export KEK
3. Intruder obtains transported key under k (using key translate or key export)
4. Intruder knows k , so can obtain transported key

Could the intruder encrypt a suitable block?

– format given in CCA manual

If not, CI-Atse finds no attack to depth 10

2. More Access Control

Three security officers, each with particular restrictions

1. has no access to middle/last key part import
2. only access to key part import complete, and no access to key being transferred
3. no access to any key part import command

CI-AtSe found no attacks

Depth bounds: 6, 3, and 10 respectively

3. Check Integrity of Key Parts

All key parts checked on entry

Final import key checked using `KEY_TEST`

Modelled by only allowing correct key to be imported

CI-AtSe finds no attack (depth bound 3, then MemOut)

Conclusions

Recommendation 1: If intruder cannot fake key blocks, safe to depth bound 10

Recommendation 2: Safe for all three security officers to bounds 6,3 and 10 resp.

Recommendation 3: Safe to bound 3 if only correct key import allowed.

CL-AtSe for API Analysis

- Pretty good
- Interpretation of bound unsuitable for APIs
 - e.g. Import/Export attack, 2 commands used twice, 8 commands total so must search paths up to 16 long
- Since intruder knowledge monotonic, possible optimisations missed?

Further Work

Have now implemented decision procedure based on work in Cortier & Steel, FCS-ARSPA 2006.

Performs fast verification, but no counterexamples in case of attack.

Hope to conclude story of 4758 CCA typecast attacks...

... but many open problems remain (parallel key search attacks, PIN block attacks, other HSMs and APIs inc. 'stateful' ones, etc.)

Summary

- IBM recommendations sound, **provided** (for recommendation 1)
intruder cannot encrypt own key blocks
- CL-Atse performed pretty well
- Some features for an ideal API analyser identified
- Much scope for further work on key management schemes
e.g. other kinds of attack, TPM, smartcards...

FIN