

---

# Automated Analysis of the Security of XOR-Based Key Management Schemes

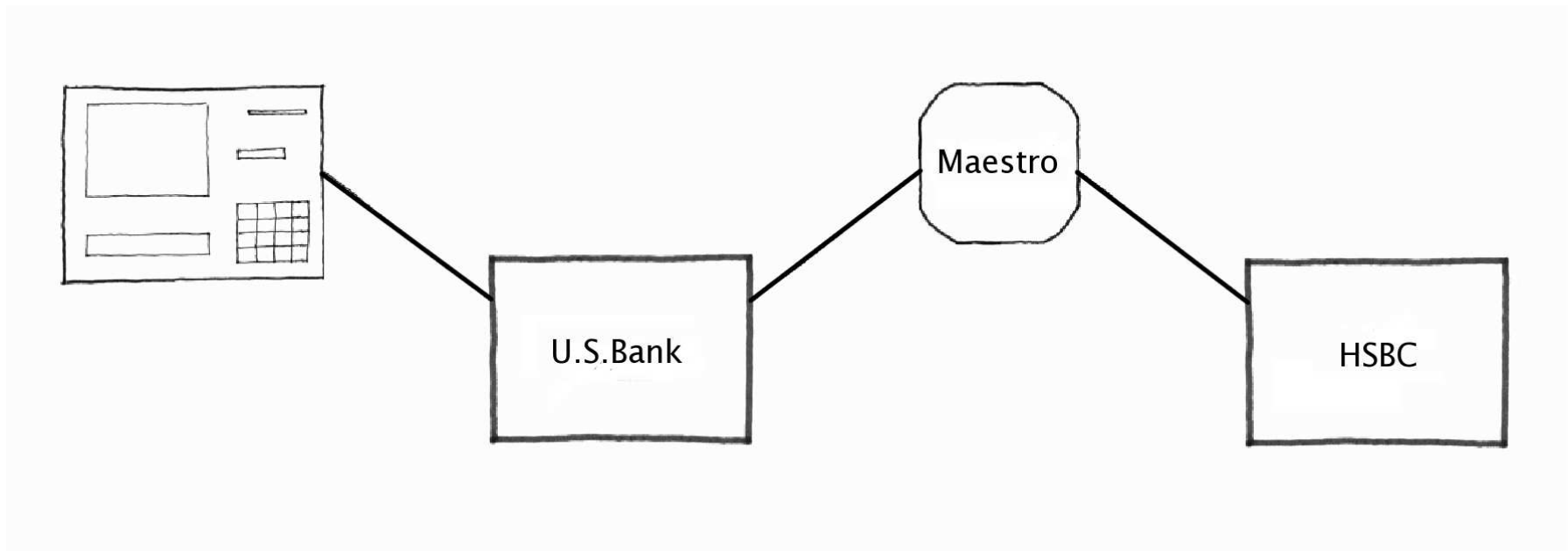
Véronique Cortier, Gavin Keighren, **Graham Steel**



 School of  
**informatics**

---

# Automated Teller Machines





## Criticality of Key Management

PIN derived by:

Write account number (PAN) as 0000AAAAAAAAAAAA

## Criticality of Key Management

PIN derived by:

Write account number (PAN) as 0000AAAAAAAAAAAA

3DES encrypt under a PDK (PIN Derivation Key)

Decimalise first 4 digits of result

## Criticality of Key Management

PIN derived by:

Write account number (PAN) as 0000AAAAAAAAAAAA

3DES encrypt under a PDK (PIN Derivation Key)

Decimalise first 4 digits of result

$\text{PIN} = \text{IPIN} + \text{Offset (modulo 10 each digit)}$

Offset NOT secure!





km  
data, pin, imp, ...

data, pin, imp,...



km  
data, pin, imp, ...

data, pin, imp,...

{ d1 }<sub>km⊕data</sub>

{ pdk1 }<sub>km⊕pin</sub>

## CCA API - Examples

Encrypt Data:

Host  $\rightarrow$  HSM :  $\{ D1 \}_{km \oplus data}$ , Message

HSM  $\rightarrow$  Host :  $\{ Message \}_{D1}$

## 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  $k = k_1 \oplus k_2$

Host  $\rightarrow$  HSM :  $k_1, \text{TYPE}$

HSM  $\rightarrow$  Host :  $\{ k_1 \}_{k_m \oplus k_p \oplus \text{TYPE}}$

## 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  $\text{KEK} \oplus \text{TYPE}$

Key Import:

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

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

## Attack (Bond, 2001) (part 1)

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

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

## Attack (Bond, 2001) (part 1)

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

Have key part  $\{ \text{kek} \oplus k2 \}_{\text{km} \oplus \text{imp} \oplus \text{kp}}$  for known  $k2$

Host  $\rightarrow$  HSM :  $\{ \text{kek} \oplus k2 \}_{\text{km} \oplus \text{kp} \oplus \text{imp}}, k2 \oplus \text{pin} \oplus \text{data}, \text{imp}$

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 :  $\{ \text{pdk} \}_{\text{km} \oplus \text{data}}$

## 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 :  $\{ \text{pan} \}_{\text{pdk}} (= \text{PIN!})$

## **IBM Recommendations**

Published in response to attacks

## IBM Recommendations

Published in response to attacks

1. Use asymmetric key crypto for key import
  - 2 officer protocol to generate key pair at destination, transfer public key to source
  - PKA\_SYMMETRIC\_KEY\_IMPORT command

## IBM Recommendations

Published in response to attacks

1. Use asymmetric key crypto for key import
  - 2 officer protocol to generate key pair at destination, transfer public key to source
  - PKA\_SYMMETRIC\_KEY\_IMPORT command
2. More access control
  - security officers access fewer commands

## IBM Recommendations

Published in response to attacks

1. Use asymmetric key crypto for key import
  - 2 officer protocol to generate key pair at destination, transfer public key to source
  - PKA\_SYMMETRIC\_KEY\_IMPORT command
2. More access control
  - security officers access fewer commands
3. Procedural controls to check entered key parts

## IBM Recommendations

Published in response to attacks

1. Use asymmetric key crypto for key import
  - 2 officer protocol to generate key pair at destination, transfer public key to source
  - PKA\_SYMMETRIC\_KEY\_IMPORT command
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

## Formal Modelling

Following the classical 'Dolev-Yao' style:

Bitstrings modelled as terms

Cryptography modelled as function on terms

## Formal Modelling

Following the classical 'Dolev-Yao' style:

Bitstrings modelled as terms

Cryptography modelled as function on terms

API rules modelled as Horn clauses

## Formal Modelling

Following the classical ‘Dolev-Yao’ style:

Bitstrings modelled as terms

Cryptography modelled as function on terms

API rules modelled as Horn clauses

Security problem modelled as derivability of a secret term

## Examples

Encrypt data:

$$x, \{ \text{xd1} \}_{k \oplus \text{data}} \rightarrow \{ x \}_{\text{xd1}}$$

## Examples

Encrypt data:

$$x, \{ x d1 \}_{k \oplus \text{data}} \rightarrow \{ x \}_{d1}$$

Intruder rules:

$$x, y \rightarrow \{ x \}_y$$

$$\{ x \}_y, y \rightarrow x$$

$$x, y \rightarrow x \oplus y$$

## Examples

Encrypt data:

$$x, \{ x d1 \}_{k \oplus \text{data}} \rightarrow \{ x \}_{d1}$$

Intruder rules:

$$x, y \rightarrow \{ x \}_y$$

$$\{ x \}_y, y \rightarrow x$$

$$x, y \rightarrow x \oplus y$$

Secrecy problem undecidable in general

## Characterisation of Class

Finite set of atoms (km, imp, data, pin, ...)

XOR term ::= atom

atom  $\oplus$  XOR term

Encryption term ::= { XOR term }<sub>XOR term</sub>

Well Formed Term ::= Encryption term

XOR term

Well Formed Rule ::= WFT, ..., WFT  $\rightarrow$  WFT

## Theorem

**If:**

$R$  finite set of well-formed rules

$S$  finite set of well-formed ground terms

$u$  some ground well-formed term

**Then:**

$S \vdash_R u \iff S \vdash_R u$  using only well-formed terms.

## Theorem

**If:**

$R$  finite set of well-formed rules

$S$  finite set of well-formed ground terms

$u$  some ground well-formed term

**Then:**

$S \vdash_R u \iff S \vdash_R u$  using only well-formed terms.

**Corollary:**

The question of whether  $S \vdash_R u$  is decidable

## Decision Procedure

$2^{12}$  possible unencrypted terms

$2^{24}$  possible encrypted terms ( $\{ \cdot \}$ .)

## Decision Procedure

$2^{12}$  possible unencrypted terms

$2^{24}$  possible encrypted terms ( $\{ \cdot \}$ .)

Encode terms as integers

$\text{kek} \oplus \text{pin} \oplus \text{data}$	$\rightarrow$	km	kp	kek	imp	exp	data	pin
19	$\leftarrow$	0	0	1	0	0	1	1

## Decision Procedure

$2^{12}$  possible unencrypted terms

$2^{24}$  possible encrypted terms ( $\{ \cdot \}$ .)

Encode terms as integers

$$\begin{array}{cccccccc} \text{kek} \oplus \text{pin} \oplus \text{data} & \rightarrow & \text{km} & \text{kp} & \text{kek} & \text{imp} & \text{exp} & \text{data} & \text{pin} \\ 19 & \leftarrow & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{array}$$

Each rule is a partial function  $f : \mathbb{N}^k \rightarrow \mathbb{N}$  for  $k$  inputs

e.g.  $f_1 : x_1, x_2 \rightarrow x_1 \oplus x_2 \quad \equiv \quad x_1, x_2 \rightarrow x_1 \oplus x_2$

## Decision Procedure

$2^{12}$  possible unencrypted terms

$2^{24}$  possible encrypted terms ( $\{ \cdot \}$ .)

Encode terms as integers

$$\begin{array}{ccccccc}
 \text{kek} \oplus \text{pin} \oplus \text{data} & \rightarrow & \text{km} & \text{kp} & \text{kek} & \text{imp} & \text{exp} & \text{data} & \text{pin} \\
 19 & \leftarrow & 0 & 0 & 1 & 0 & 0 & 1 & 1
 \end{array}$$

Each rule is a partial function  $f : \mathbb{N}^k \rightarrow \mathbb{N}$  for  $k$  inputs

e.g.  $f_1 : x_1, x_2 \rightarrow x_1 \oplus x_2 \quad \equiv \quad x_1, x_2 \rightarrow x_1 \oplus x_2$

$$f_2 : [xky|x], xtype, [xkek|km \oplus imp] \rightarrow [xky|km \oplus xtype] \quad \text{IF} \quad x = xkek \oplus xtype$$

## Decision Procedure

1. Allocate sufficient memory for all possible terms
2. Set to 1 locations corresponding to initial knowledge, rest to 0
3. Exhaustively apply each rule, setting newly discovered terms to 1
4. Repeat 3 until no new terms are discovered

## Decision Procedure

1. Allocate sufficient memory for all possible terms
2. Set to 1 locations corresponding to initial knowledge, rest to 0
3. Exhaustively apply each rule, setting newly discovered terms to 1
4. Repeat 3 until no new terms are discovered

Some optimisations used

## Decision Procedure

1. Allocate sufficient memory for all possible terms
2. Set to 1 locations corresponding to initial knowledge, rest to 0
3. Exhaustively apply each rule, setting newly discovered terms to 1
4. Repeat 3 until no new terms are discovered

Some optimisations used

Possible attack on recommendation 1!

## Decision Procedure

1. Allocate sufficient memory for all possible terms
2. Set to 1 locations corresponding to initial knowledge, rest to 0
3. Exhaustively apply each rule, setting newly discovered terms to 1
4. Repeat 3 until no new terms are discovered

Some optimisations used

Possible attack on recommendation 1!

After fix, verifies all recommendations in a few seconds

## Evaluation

### **Attack**

As serious as the original Bond attack

### **API Modelling Methodology**

Works well for this 'stateless' API, will need further work to broaden scope

### **XOR/integer representation trick**

Highly efficient, but requires finite vocabulary of terms

## Related Work

Decidable classes for protocols using XOR:

Comon and Cortier 03, Verma et al. 05

– incomparable

Otter work, Youn et. al 05:

Rediscovered old attacks, used heuristics without proof

Courant & Monin 06:

Verified Bond's proposed fixes

## Summary

- Found a new attack on the CCA API
- Proved decidability for a class containing the API
- Devised new representation and decision procedure
- Verified a fixed version of the API
- More to do (key conjuring, parallel key search...)

## Summary

- Found a new attack on the CCA API
- Proved decidability for a class containing the API
- Devised new representation and decision procedure
- Verified a fixed version of the API
- More to do (key conjuring, parallel key search...)

---

EPSRC Project “Automated Analysis of Security Critical Systems”

<http://dream.inf.ed.ac.uk/projects/aascs/>