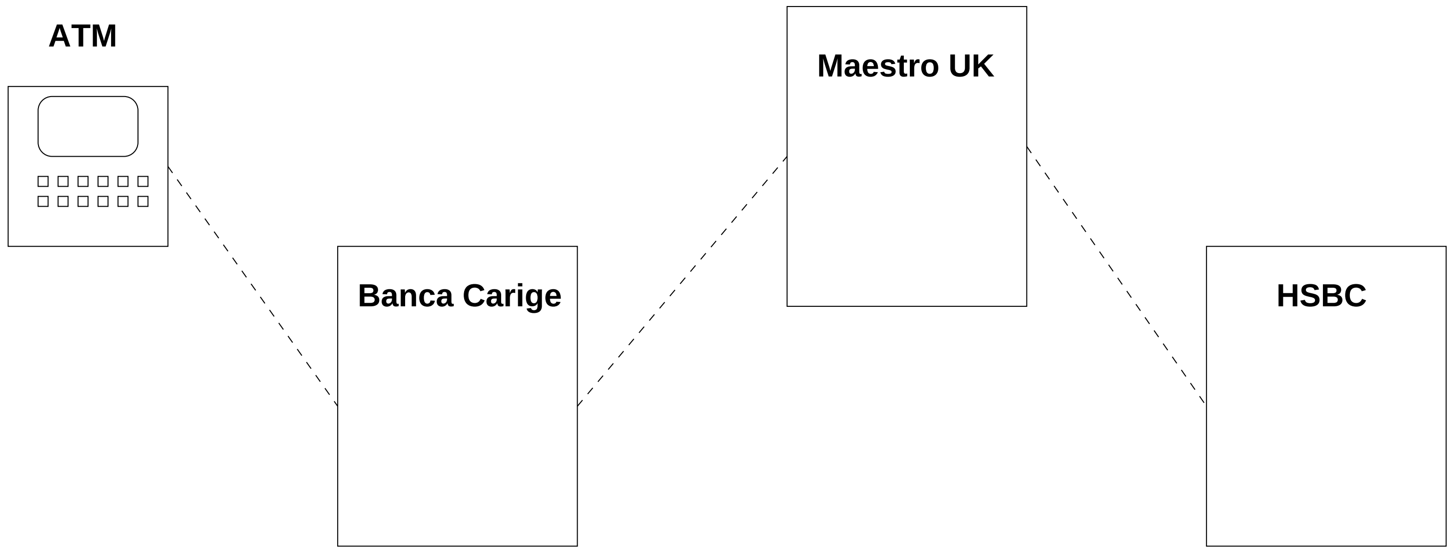
Formalised PIN Cracking

Graham Steel

THE UNIVERSITY OF EDINBURGH

School of informatics

Automated Teller Machines



Hardware Security Modules

PIN Block Attacks

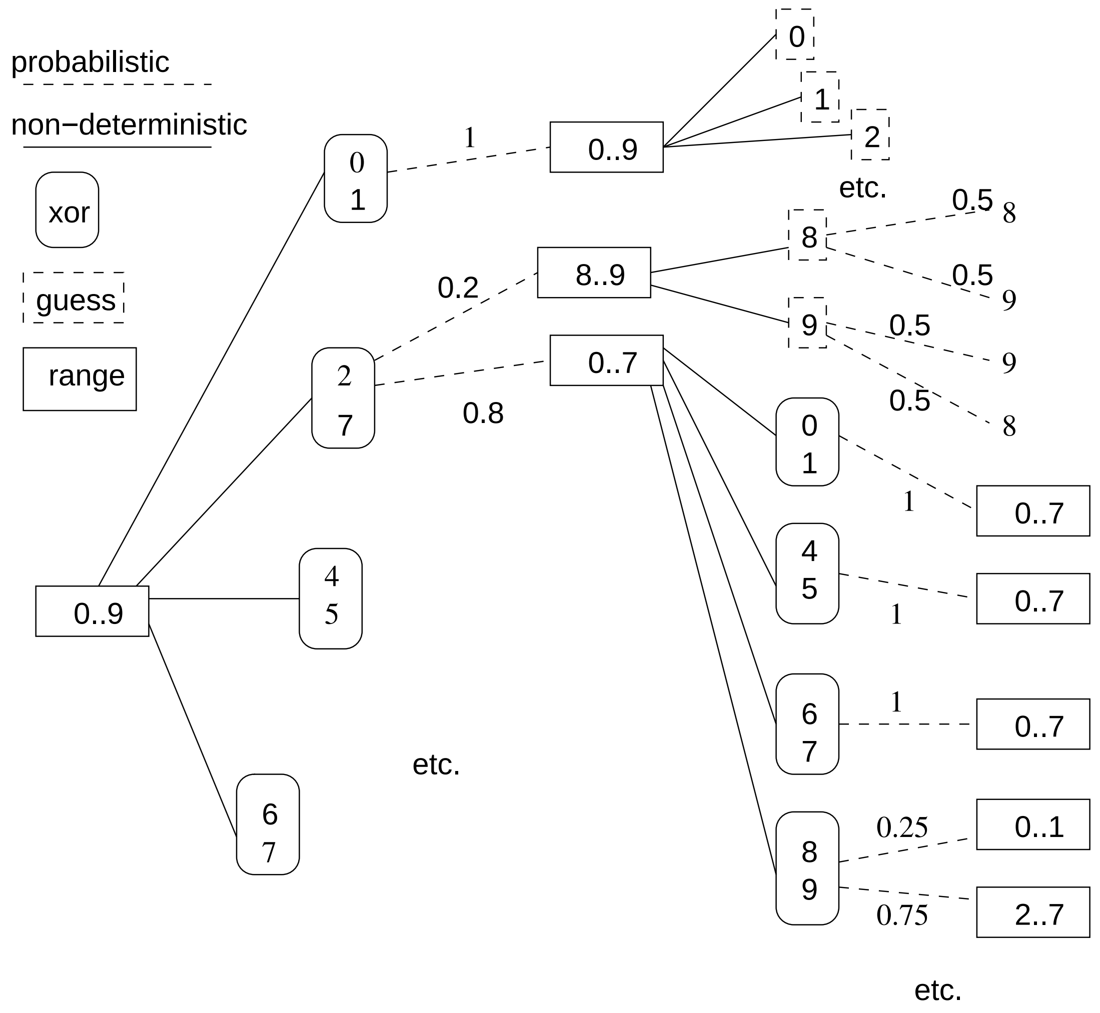
ISO format 0

```
04PPPPFFFFFFFFFF
0000AAAAAAAAAAAA
```

VISA format 3

```
PPPPFFFFFFFFFFFF
```

Error check ($0 \leq P \leq 9$) leaks information


probabilistic
non−deterministic
xor
guess
range
0..9
0 1
2 7
4 5
6 7
1
0.2
0.8
0..9
8..9
0..7
0
1
2
etc.
8
9
0.5
8
0.5
9
0.5
9
0.5
8
0 1
4 5
6 7
8 9
1
1
1
0.25
0.75
0..7
0..7
0..7
0..1
2..7
etc.
etc.

Optimising the Attack

Generate full tree of possible attacks from command specs.

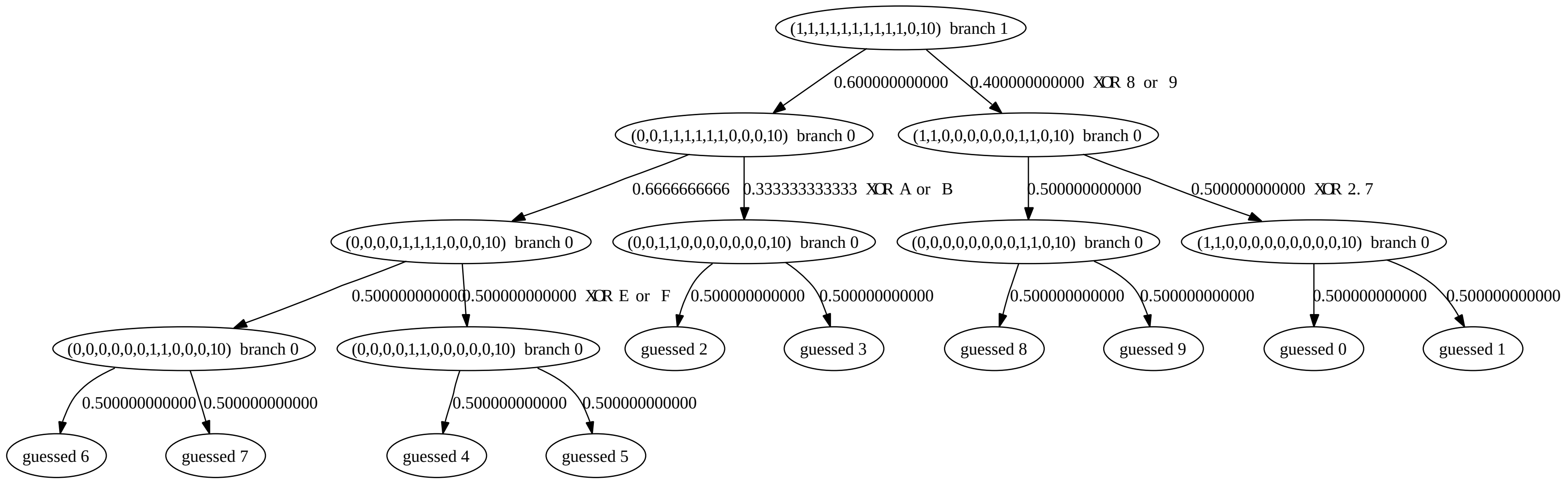
e.g. `command(xor_in_E_F,(6..9),(0..5)).`

Apply PRISM - probabilistic model checker (Birmingham)

Get minimum expected number of steps to determine PIN digit: 3.4

Generate tree for best attack

Optimised Attack



(1,1,1,1,1,1,1,1,1,1,0,10) branch 1
0.600000000000
0.400000000000 XOR 8 or 9
(0,0,1,1,1,1,1,1,0,0,0,10) branch 0
(1,1,0,0,0,0,0,0,1,1,0,10) branch 0
0.6666666666
0.333333333333 XOR A or B
0.500000000000
0.500000000000 XOR 2. 7
(0,0,0,0,1,1,1,1,0,0,0,10) branch 0
(0,0,1,1,0,0,0,0,0,0,0,10) branch 0
(0,0,0,0,0,0,0,0,1,1,0,10) branch 0
(1,1,0,0,0,0,0,0,0,0,0,10) branch 0
0.500000000000
0.500000000000 XOR E or F
0.500000000000
0.500000000000
0.500000000000
0.500000000000
0.500000000000
0.500000000000
(0,0,0,0,0,0,1,1,0,0,0,10) branch 0
(0,0,0,0,1,1,0,0,0,0,0,10) branch 0
guessed 2
guessed 3
guessed 8
guessed 9
guessed 0
guessed 1
0.500000000000
0.500000000000
0.500000000000
0.500000000000
guessed 6
guessed 7
guessed 4
guessed 5

A Word About Your PIN

Original PIN (IPIN) derived by:

3DES encrypting 0000AAAAAAAAAAAA

(The As are your PAN digits)

Decimalise result

```
0123456789ABCDEF
0123456789012345
```

PIN = IPIN + Offset (modulo 10 each digit)

Offset NOT secure!

IBM 4758 - Control Vectors

Mechanism to support many types of key: ‘role based access’

Keys stored outside box encrypted under master key XOR control vector

E.g. data keys

$\{\!|\ \text{d1}\ |\!\}_{km \oplus data}$

Encrypt Data:

Host	$\rightarrow$	HSM	:	$\{\!|\ \text{d1}\ |\!\}_{km \oplus data}$, message
HSM	$\rightarrow$	Host	:	$\{\!|\ \text{message}\ |\!\}_{d1}$

Importing Key Parts

'Separation of duty'

Typically used to import a 'key encrypting key' (kek)

Key kek = 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	:	$\{\!\|\, k1 \oplus k2 \,\|\!\}_{km \oplus TYPE}$

Importing Encrypted Keys

Exported from another 4758 under $KEK \oplus TYPE$

First import KEK, obtaining $\{\!|\ \text{KEK}\ |\!\}_{\text{km}\oplus\text{imp}}$

$$\begin{array}{llll} \text{Host} & \rightarrow & \text{HSM} & : & \{\!|\ \text{KEY1}\ |\!\}_{\text{KEK}\oplus\text{TYPE}},\ \text{TYPE},\ \{\!|\ \text{KEK}\ |\!\}_{\text{km}\oplus\text{imp}} \\ \text{HSM} & \rightarrow & \text{Host} & : & \{\!|\ \text{KEY1}\ |\!\}_{\text{km}\oplus\text{TYPE}} \end{array}$$

Attack (Bond, 2001)

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

Have key part $\{\!|\ \text{kek}\oplus\text{k3}\ |\!\}_{\text{km}\oplus\text{imp}\oplus\text{kp}}$ for known k3

Host	$\rightarrow$	HSM	:	$\{\!|\ \text{kek}\oplus\text{k3}\ |\!\}_{\text{km}\oplus\text{kp}\oplus\text{imp}},\ \text{k3}\oplus\text{pin}\oplus\text{data}$, imp
HSM	$\rightarrow$	Host	:	$\{\!|\ \text{kek}\oplus\text{pin}\oplus\text{data}|\!\}_{\text{km}\oplus\text{imp}}$

Attack (Bond, 2001) (part 2)

Key Import

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

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

Encrypt data

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

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

Formal Modelling

HSMs are 'stateless'

$P(x)$ if x is 'public' - i.e. outside HSM

One clause for each command

$$\begin{array}{llll} \text{Host} & \rightarrow & \text{HSM} & : \{\!|\ \text{d1}\ |\!\}_{\text{km}\oplus\text{data}},\ \text{message} \\ \text{HSM} & \rightarrow & \text{Host} & : \{\!|\ \text{message}\ |\!\}_{\text{d1}} \end{array}$$

$$P(Msg) \wedge P(crypt(km \oplus data, D1)) \Rightarrow P(crypt(D1, Msg))$$

The Problem with XOR

$P(x) \wedge P(y) \rightarrow P(x \oplus y)$

Associativity and Commutativity

Self-Inverse ($a \oplus b \oplus a \equiv b$)

XOR constraints

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

$$P(crypt(X,Key)) \wedge P(Type) \wedge P(crypt(km \oplus imp, Kek))$$
$$\Rightarrow P(crypt(km \oplus Type, decrypt(Kek \oplus Type, crypt(X,Key)))).$$
$$\Rightarrow decrypt(K, crypt(K,X)) = X.$$

$$P(crypt(X,Key)) \wedge P(Type) \wedge P(crypt(km \oplus imp, Kek))$$
$$\Rightarrow P(crypt(km \oplus Type, Key)) \quad IF \quad Kek \oplus Type =_{xor} X.$$

Checking Solubility

Permit only inferences which leave soluble constraints

Check:

- If there are any variables at XOR positions, it is soluble
- Otherwise count up all terms. If there are an even number of each term, it is soluble. If not, insoluble.

Store in normal form

$$x_1 \oplus \ldots \oplus x_n \quad = \quad t_1 \oplus \ldots \oplus t_n$$

Subsumption Checking

If C_1 subsumes C_2 without consideration of XOR constraints, then it is a valid subsumer iff:

1. C_1 has no XOR constraint

or

2. C_1 and C_2 have the same XOR constraints after substitutions applied

Results

Implemented in daTac, [Vigneron, 1994]

- Bond’s attack shown above
- Import/Export Attack (also due to Bond)
- IBM’s own attack
- Attack on NSPKL variant - Jacquemard et al. model

PIN Decimalisation Table

Standard

```
0123456789ABCDEF
0123456789012345
```

Attack

```
0123456789ABCDEF
1123456789012345
```

Alter offset to establish effect of change

Further Work

- PIN Block format analysis
- Improvement to XOR constraint solving
- Ideas for decimalisation table attacks

Summary

- API analysis exciting new area!

  Used also in smartcards, POS devices, mobile phones, DRM, . . .

- Some early successes, many problems remain

  XOR constraints, probabilistic model checking look good

  Need ideas for decimalisation table attacks

  There are also other kinds of attacks

`http://dream.inf.ed.ac.uk/projects/aascs/`