

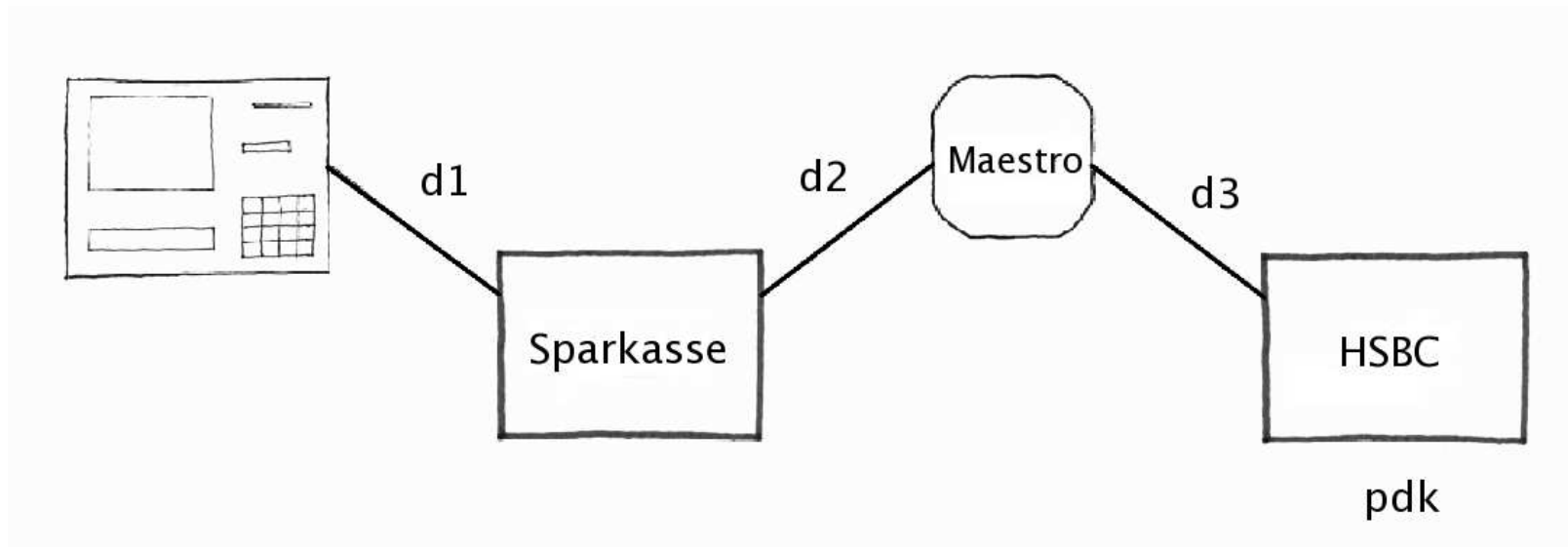
---

# Formal Analysis of Security APIs

Graham Steel



# Automated Teller Machines



## PIN Derivation

PIN derived by:

Write account number (PAN) as 0000AAAAAAAAAAAA

3DES encrypt under a PDK (PIN Derivation Key)

Decimalise first 4 digits of result obtain IPIN

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

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 → HSM : { pdk }<sub>kek⊕pin</sub>, data, { kek ⊕ pin ⊕ data }<sub>km⊕imp</sub>

HSM → Host : { pdk }<sub>km⊕data</sub>

### Encrypt data

Host → HSM : { pdk }<sub>km⊕data</sub>, pan

HSM → Host : { pan }<sub>pdk</sub> (= PIN!)

## 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

Are they secure?

## Formal Modelling

Following the classical ‘Dolev-Yao’ style:

Bitstrings modelled as terms

Cryptography modelled as function on terms

Model each command as 2-step protocol

Have used:

- Model checker CL-AtSe – to find attacks
- Ad-Hoc decision procedure – for verification, based on theoretical results in Cortier & Steel, TACAS ’07.

## Attack on Recommendation 1?

Discovered by CL-AtSe

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

Should split access to PKA\_Import and Key\_Import

Then Cl-Atse finds no attack to depth 10

## 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.

**Corollary:**

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

## Decision Procedure

$2^{12}$  possible unencrypted terms

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

Encode terms as integers

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

Each rule is a simple operation on integers

e.g.  $x1, x2 \rightarrow x1 \oplus x2 \equiv n, m \rightarrow n \oplus m$

Implemented in C, a few hundred lines

Verifies all fixes in  $\sim 100$ s each

## PIN Blocks

64-bit strings to encode a guessed PIN for encryption

e.g.

VISA format 3

PPPPFFFFFFFFFFFFFF

ISO 9564 format 0

04PPPPFFFFFFFFFFFF

0000AAAAAAAAAAAA

## PIN Block Operations

Verify\_PIN

Verify\_Offset\_PIN

Verify\_PVV

Translate\_Encrypted\_PIN

Clear\_PIN\_Encrypt

Customer chooses which ones are enabled at installation time

## ISO-0 Reformatting attack

(Clulow, 2003)

```
04PPPPFFFFFFFFFFFF
```

```
0000AAAAAAAAAAAAAA
```

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

**Extend:**

Masquerade ISO-0 as VISA format 3

```
0604PPPPFFFFFFFFFFFF
```

Can now uniquely determine digits

## Dectab Attacks

Controls decimalisation of  $\{ \text{pan} \}_{\text{pdk}}$

Cleartext input to verify commands

Standard

0123456789ABCDEF

0123456789012345

Attack 1

0123456789ABCDEF

1123456789112345

Alter offset to establish position of digits (Bond + Zieliński, 2003)

## Formal Modelling

Take a customer configuration and an API spec. as input

Using CLP, generate tree of all possible attacks

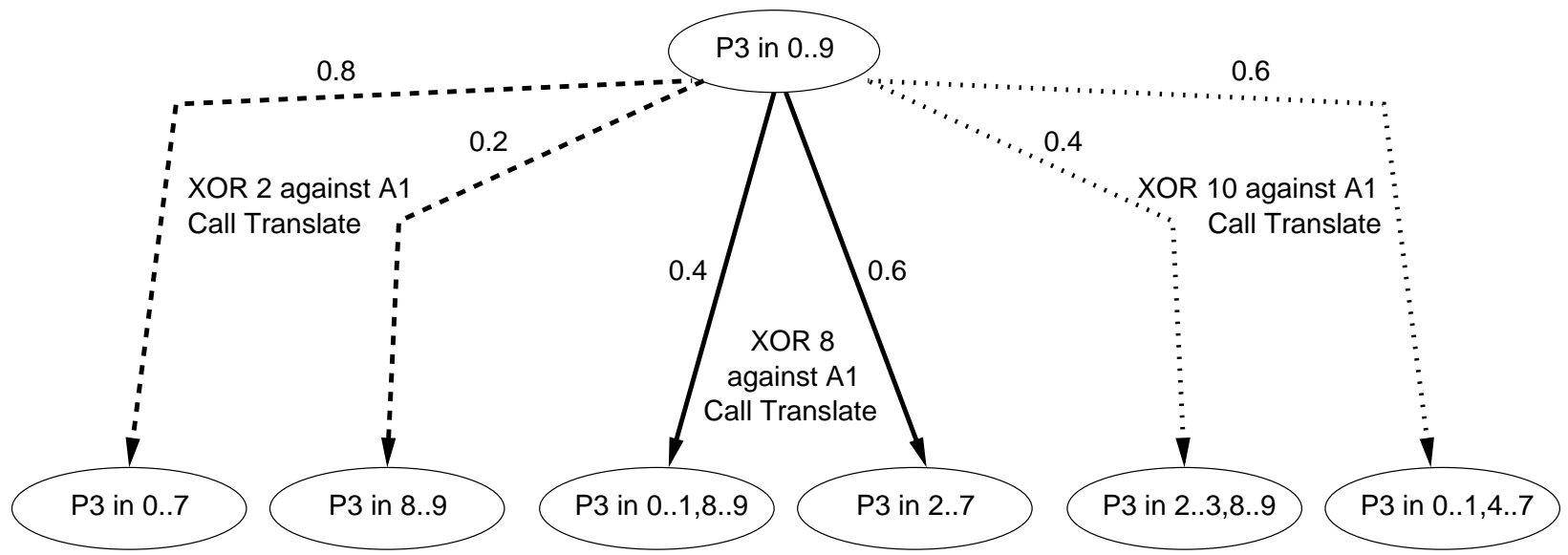
Meta-logical predicates allow us to calculate transition probabilities

Apply PRISM (Kwiatkowska et. al, 2004)

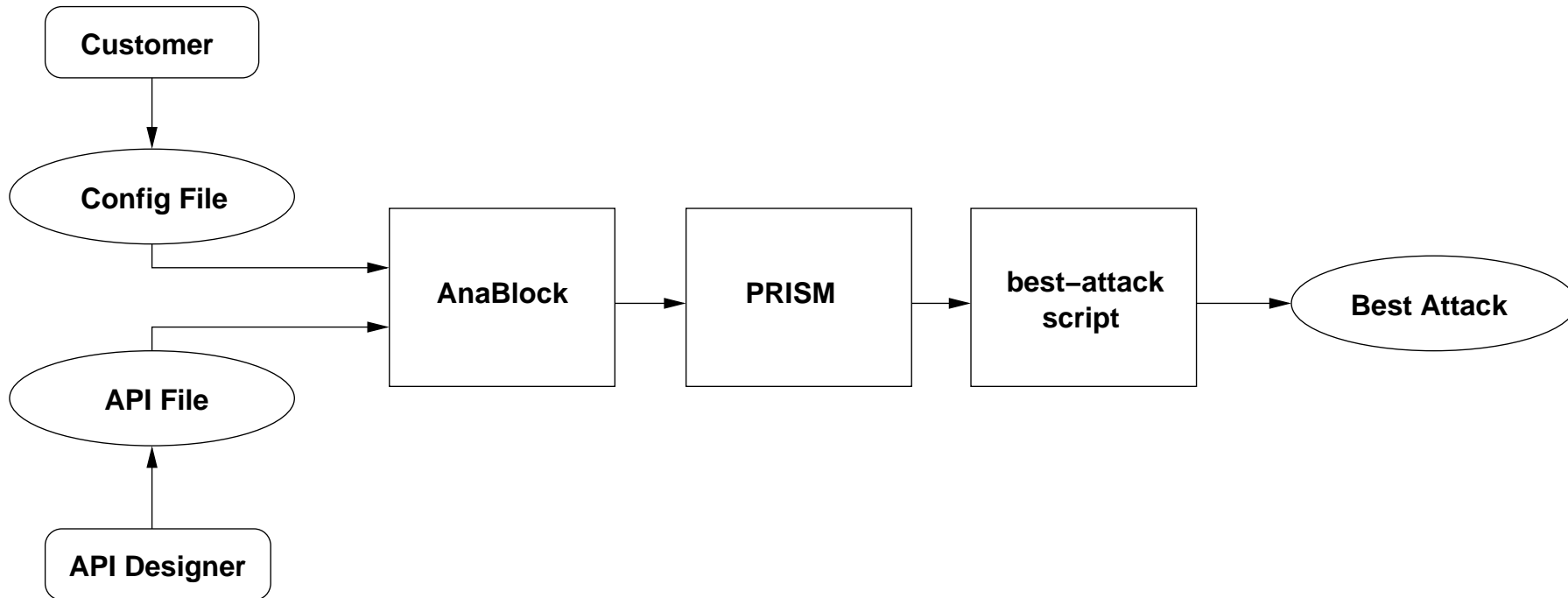
Get minimum expected number of steps to determine PIN

Generate tree for best attack

# Attack Trees



# AnaBlock



## PCTL Properties

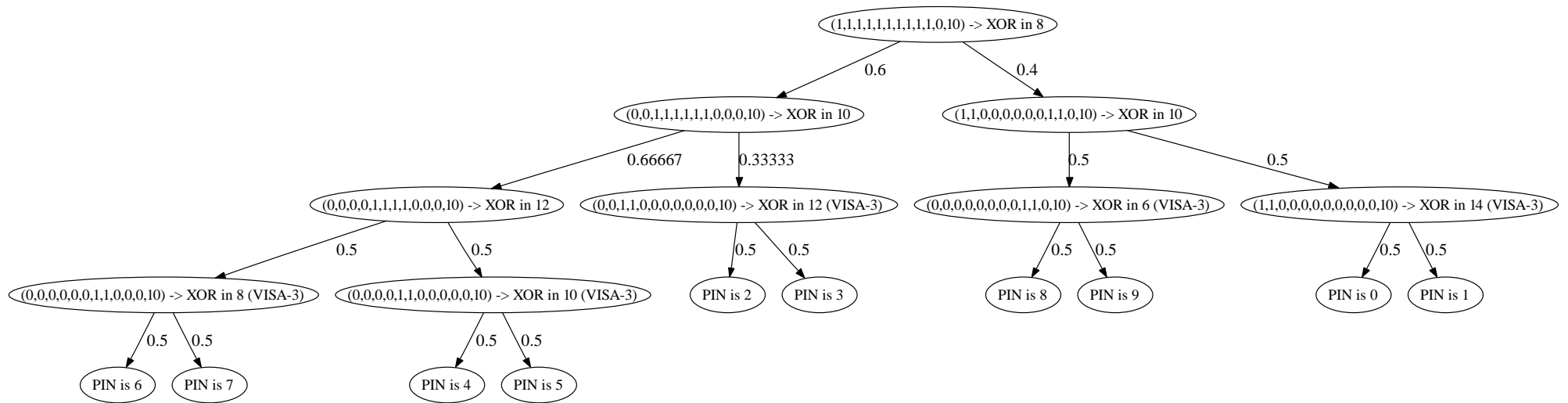
What chance of obtaining a PIN?

$$Pmax =? \quad [true \ U \ ( \ P1\_guessed \wedge \ P2\_guessed \wedge \\ \ P3\_guessed \wedge \ P4\_guessed ) ]$$

What's the fastest way to do it?

$$Rmin =? \quad [F \ ( \ P1\_guessed \wedge \ P2\_guessed \wedge \\ \ P3\_guessed \wedge \ P4\_guessed ) \{ "init" \} \{ min \} ]$$

# Optimised Attack

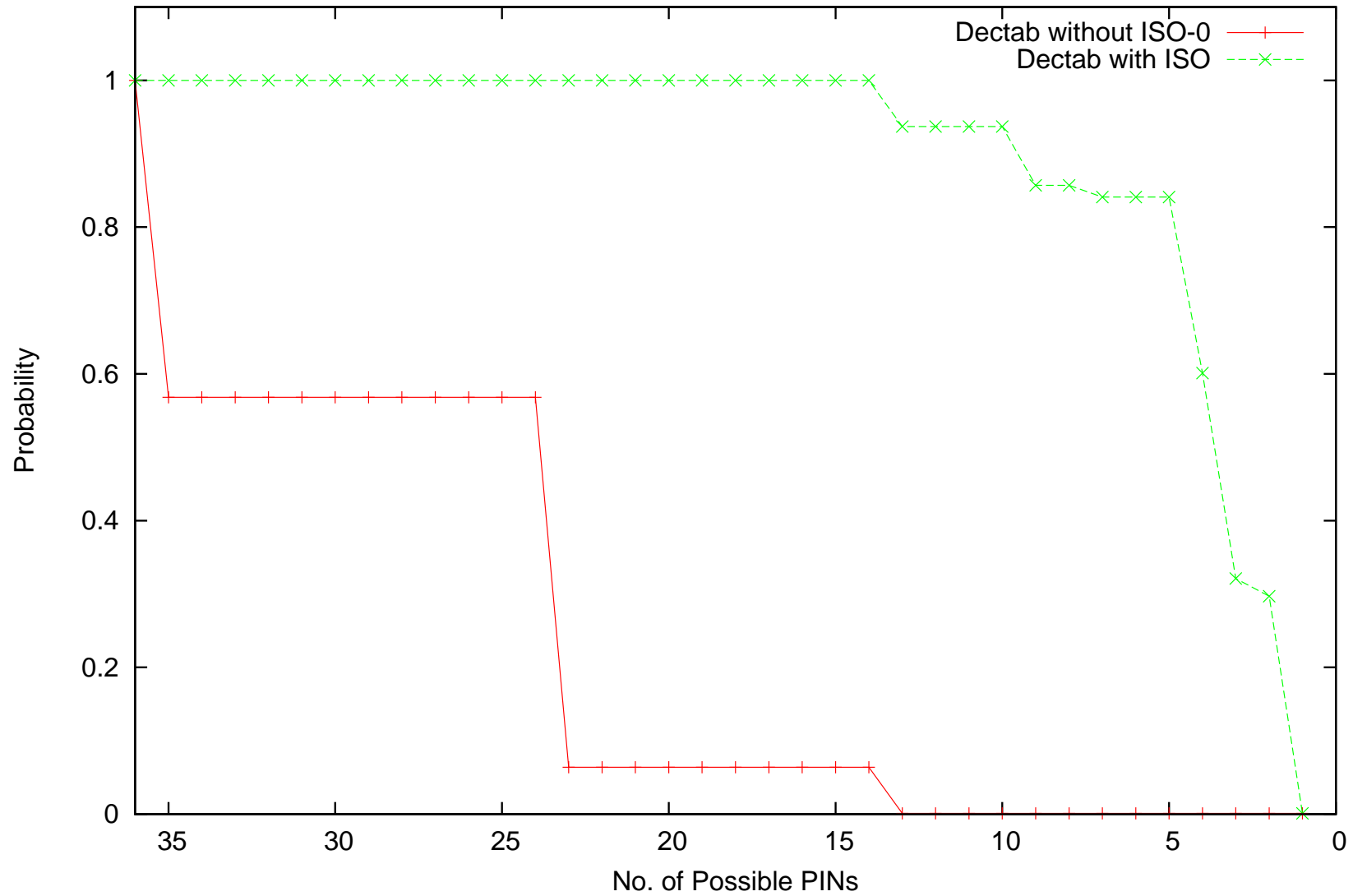


## Results from Analysis (Generic API)

No.	Attack	$P(determined)$	$E(Steps)$
(1)	ISO-0 (extended)	1	13.6
(2)	Dectab	1	16.145
(3)	Dectab & ISO (restricted)	1	15.275

No.	Attack	$k = 400$	$k = 36$	$k = 24$	$k = 14$	$k = 1$
(4)	ISO-0 (restricted)	1	0	0	0	0
(5)	Dectab no offset	1	1	0.568	0.064	0.001
(6)	Dectab no offset & ISO-0 (restricted)	1	1	1	1	0.001

Performance of Dectab attack without offset



## Comparison to Protocol Analysis

- One-shot analysis of a specification not sufficient
  - Legacy support issues
  - Varying configurations required
- Size:
  - Typical protocol 3-6 messages
  - Typical API 8-12 two-message protocols
  - but messages typically short
- Modelling algebraic properties of cryptofunctions essential
- Guess complexity must be modelled

## Future Plans

Broaden scope to other tamper-proof hardware

- Automotive
- TCA
- Other HSM applications

Challenges:

- Specification of policies
- Scaling up
- State

## Summary

- API analysis emerging as important area
- Have made some progress
  - Verification of revised CCA API
  - Framework for PIN Block attack analysis
- First Workshop on Analysis of Security APIs, Venice, July

---

EPSRC Project “Automated Analysis of Security Critical Systems”

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