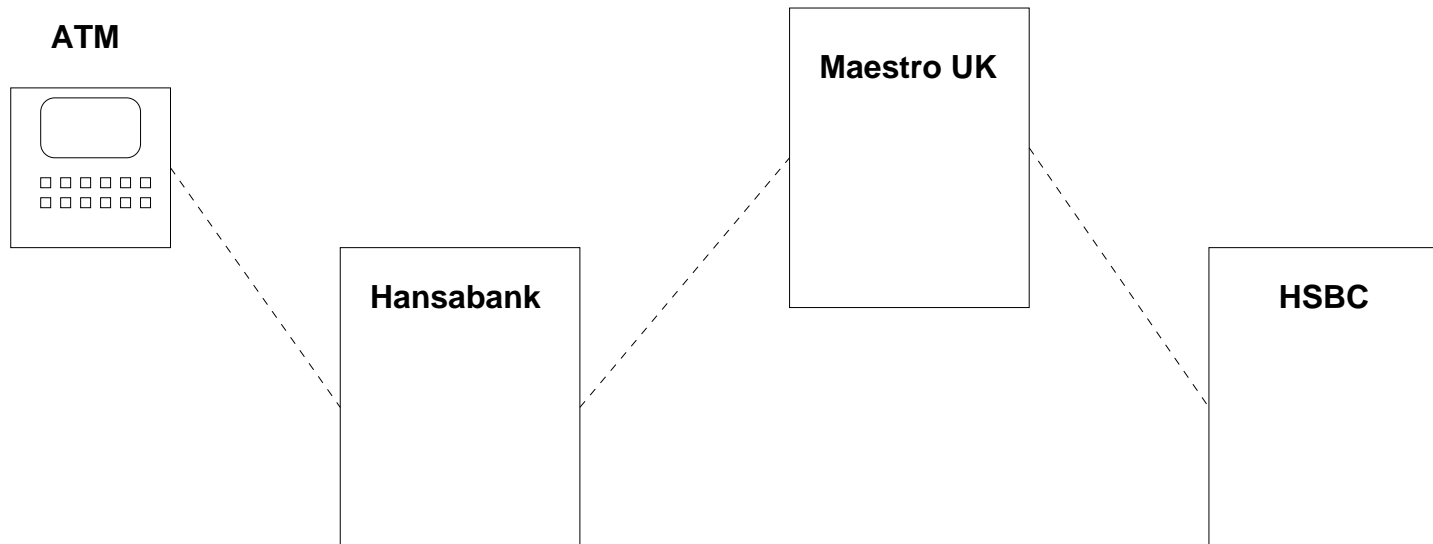

Deduction with XOR Constraints in Security API Modelling

Graham Steel



 School of
informatics

Automated Teller Machines



Hardware Security Modules



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

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

Encrypt Data:

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

HSM \rightarrow Host : $\{ message \}_{d1}$

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 $\{ KEK \}_{km \oplus imp}$

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

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

Attack (Bond, 2001)

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

Have key part $\{ \text{kek} \oplus k_3 \}_{\text{km} \oplus \text{imp} \oplus \text{kp}}$ for known k_3

Host \rightarrow HSM : $\{ \text{kek} \oplus k_3 \}_{\text{km} \oplus \text{kp} \oplus \text{imp}}, k_3 \oplus \text{pin} \oplus \text{data}, \text{imp}$

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

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

Encrypt data

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

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

Formal Modelling

HSMs are 'stateless'

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

One clause for each command

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

HSM \rightarrow Host : $\{ \text{message} \}_{\text{d1}}$

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

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

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

$$\begin{aligned}
 &P(\text{crypt}(X, \text{Key})) \wedge P(\text{Type}) \wedge P(\text{crypt}(\text{km} \oplus \text{imp}, \text{Kek})) \\
 &\Rightarrow P(\text{crypt}(\text{km} \oplus \text{Type}, \text{decrypt}(\text{Kek} \oplus \text{Type}, \text{crypt}(X, \text{Key}))))). \\
 &\Rightarrow \text{decrypt}(K, \text{crypt}(K, X)) = X.
 \end{aligned}$$

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

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

$$x_1 \oplus \dots \oplus x_n = t_1 \oplus \dots \oplus t_n$$

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

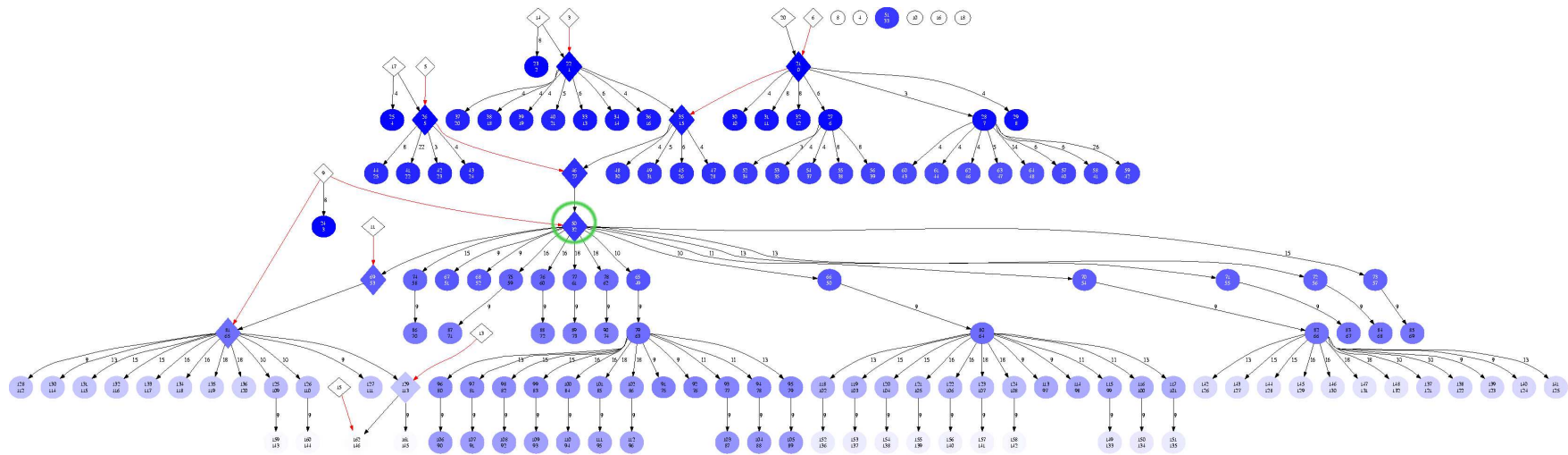
2. C_1 and C_2 have the same XOR constraints after substitutions applied

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

4758 Attack 1



Related Work

Security APIs:

- Longley & Rigby, 1992 - Key management scheme without XOR
- Ganapathy et al, 2005 - Model checking for fragment of first attack
- Bond & Clulow - Work in progress on first-order model

Protocols with XOR:

- Chevalier et al. , Comon-Shmatikov, 2003 - Insecurity shown decidable (bounded runs, NP).
- Basin, Mödersheim, Viganò, 2005 - General framework for OFMC (unimplemented)

Further Work

- Improve solving of final XOR constraint
- Look at new APIs for novel attacks
- Comparison to (special purpose) model checking
- PIN Block format analysis

Conclusions

XOR constraints considerably improve reasoning capabilities of a FOTP when dealing with bitwise XOR

- Allow implicit encryption model to be used
- Allow forward, backward and mixed strategies
- Reduce explicit construction of terms by XOR

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