

A Formal Model for Detecting Parallel Key Search Attacks

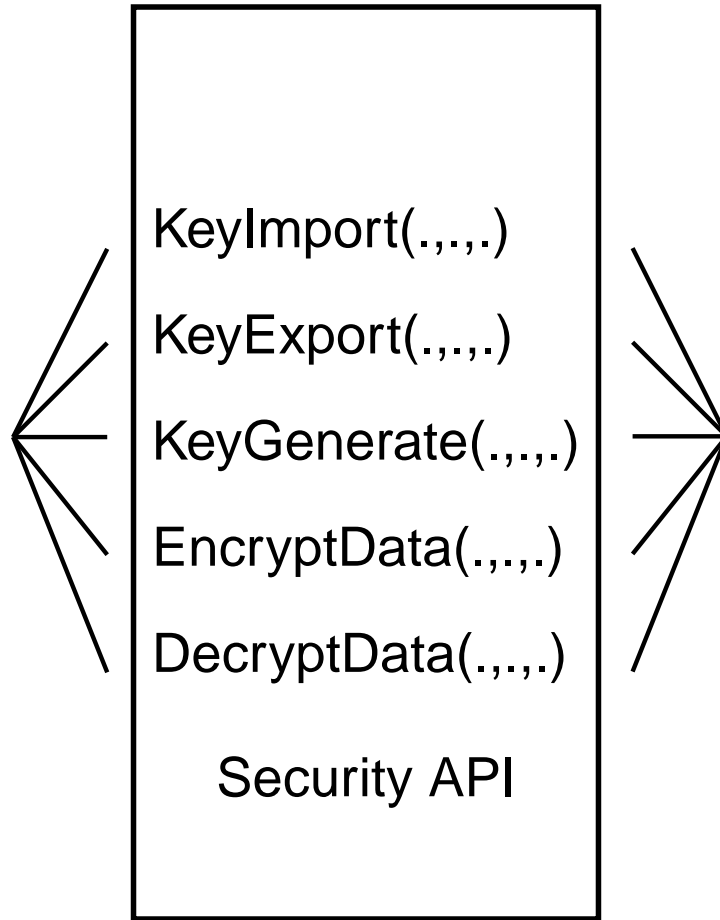
Graham Steel and Judicaël Courant



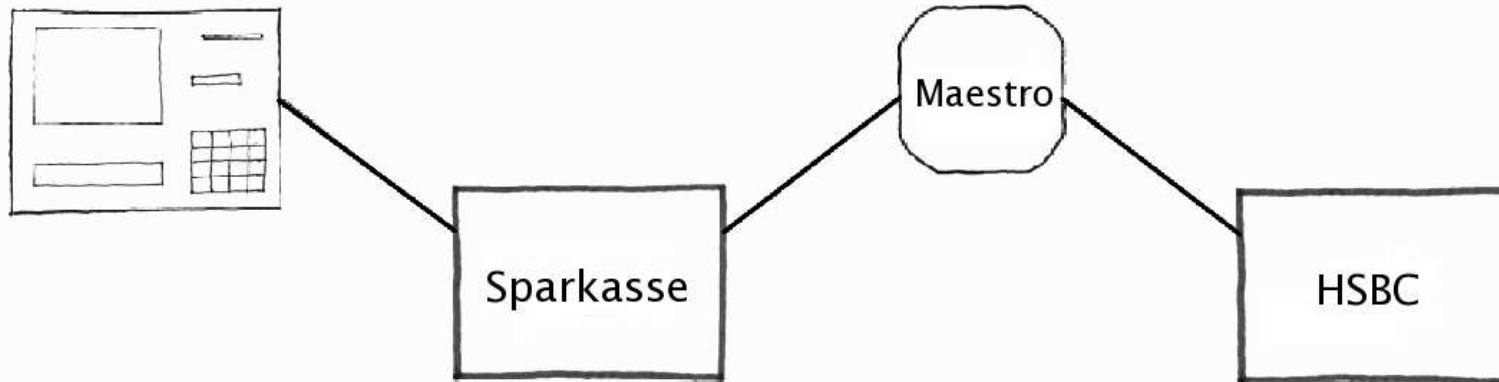
 School of
informatics



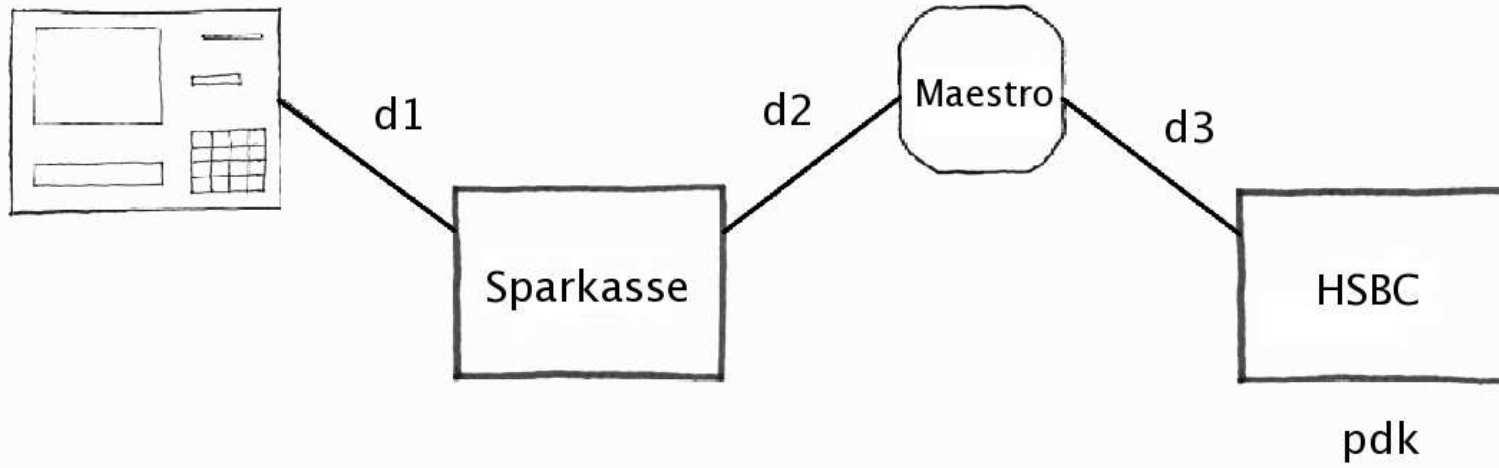




Automated Teller Machines



Automated Teller Machines







km
data, pin, imp, ...

data, pin, imp,...



km
data, pin, imp, ...

data, pin, imp,...

{ d1 }_{km⊕data}

{ pdk1 }_{km⊕pin}

E.g. CCA API

Encrypt Data:

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

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

E.g. CCA API

Encrypt Data:

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

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

Export Key:

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

HSM \rightarrow Host : $\{ D1 \}_{KEK \oplus data}$

Parallel Key Search

Key Generate:

$$TYPE \rightarrow \{k\}_{km \oplus TYPE}$$

Parallel Key Search

Key Generate:

$$TYPE \rightarrow \{k\}_{km \oplus TYPE}$$

Key test:

$$\{k\}_{km \oplus TYPE}, TYPE \rightarrow \{00000000\}_k$$

Parallel Key Search

Key Generate:

$$TYPE \rightarrow \{k\}_{km \oplus TYPE}$$

Key test:

$$\{k\}_{km \oplus TYPE}, TYPE \rightarrow \{00000000\}_k$$

Obtain test values for large number of k

Parallel Key Search

Key Generate:

$$TYPE \rightarrow \{k\}_{km \oplus TYPE}$$

Key test:

$$\{k\}_{km \oplus TYPE}, TYPE \rightarrow \{00000000\}_k$$

Obtain test values for large number of k

Offline, encrypt 00000000 under random r until collision occurs

Formalism

Encrypt data v0.1

$$\text{xmsg}, \{\text{xkey}\} \text{ km} \oplus \text{data} \rightarrow \{\text{xmsg}\} \text{ xkey}$$

Formalism

Encrypt data v0.1

$$\text{xmsg}, \{\text{xkey}\} \xrightarrow{\text{km} \oplus \text{data}} \{\text{xmsg}\}_{\text{xkey}}$$

Encrypt Data v0.2

$$x, y \rightarrow \{x\}_{\text{dec}(y, \text{km} \oplus \text{data})}$$

Formalism

Encrypt data v0.1

$$\text{xmsg}, \{\text{xkey}\}_{\text{km} \oplus \text{data}} \rightarrow \{\text{xmsg}\}_{\text{xkey}}$$

Encrypt Data v0.2

$$x, y \rightarrow \{x\}_{\text{dec}(y, \text{km} \oplus \text{data})}$$

Encrypt Data v0.21

$$\text{chkOdd}(\text{dec}(y, \text{km} \oplus \text{data})), x, y \rightarrow \{x\}_{\text{dec}(y, \text{km} \oplus \text{data})}$$

Key Conjuring

$\text{chkOdd}(\text{dec}(y, \text{km} \oplus \text{data})), x, y \rightarrow \{x\}_{\text{dec}(y, \text{km} \oplus \text{data})}$

Key Conjuring

$$\text{chkOdd}(\text{dec}(y, \text{km} \oplus \text{data})), x, y \rightarrow \{x\}_{\text{dec}(y, \text{km} \oplus \text{data})}$$

Attacker tries random values instead of y

Obtains an encrypted data key.

Key Conjuring

$$\text{chkOdd}(\text{dec}(y, km \oplus \text{data})), x, y \rightarrow \{x\}_{\text{dec}(y, km \oplus \text{data})}$$

Attacker tries random values instead of y

Obtains an encrypted data key.

Encrypt Data:

$$x \xrightarrow{\text{new } n} \{x\}_{\text{dec}(n, km \oplus \text{data})}, n, \text{chkOdd}(\text{dec}(n, km \oplus \text{data}))$$

Parallel Key Search

Key Generate:

$$y \xrightarrow{\text{new } n} \{n\}_{km \oplus y}$$

Parallel Key Search

Key Generate:

$$y \xrightarrow{\text{new } n} \{n\}_{km \oplus y}$$

Key test:

$$x, y \rightarrow \{0\}_{\text{dec}(x, km \oplus y)}$$

Parallel Key Search - 2

Offline generation:

new n
→ n

Parallel Key Search - 2

Offline generation:

$$\begin{array}{c} \text{new } n \\ \rightarrow \\ n \end{array}$$

Collision:

$$n, f(n), f(g(n')) \rightarrow g(n')$$

Example Parallel Key Search

Key Generate:

$$\text{exp} \xrightarrow{\text{new } n_1} \{n_1\}_{km \oplus \text{exp}}$$

Example Parallel Key Search

Key Generate:

$$\text{exp} \xrightarrow{\text{new } n_1} \{n_1\}_{km \oplus \text{exp}}$$

Key test:

$$\{n_1\}_{km \oplus \text{exp}}, \text{exp} \rightarrow \{0\}_{n_1}$$

Example Parallel Key Search - 2

Offline generation and encryption:

$$\begin{array}{ccc} & \xrightarrow{\text{new } n_2} & n_2 \\ n_2, 0 & \longrightarrow & \{0\}_{n_2} \end{array}$$

Example Parallel Key Search - 2

Offline generation and encryption:

$$\begin{array}{ccc} & \xrightarrow{\text{new } n_2} & n_2 \\ n_2, 0 & \longrightarrow & \{0\}_{n_2} \end{array}$$

Collision:

$$n_2, \{0\}_{n_2}, \{0\}_{n_1} \longrightarrow n_1$$

Concerns

- Collision rule?

- Cryptographic foundations?

HSM as encryption oracle for parameterised analysis?

- How to search in such a formalism?

- Quantitative treatment of cost?

Summary

Parallel key search important challenge in our API analysis project

Have sketched a treatment here

Future work to improve treatment of collisions, account for costs