Analysis of Security APIs (part I)

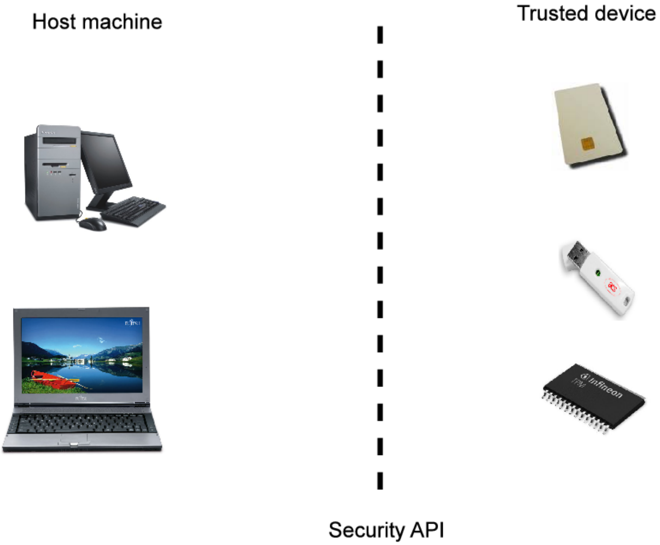
Riccardo Focardi

Università Ca' Foscari di Venezia, Italy
focardi@dsi.unive.it

http://www.dsi.unive.it/~focardi
http://secgroup.ext.dsi.unive.it/

FOSAD 2010
Bertinoro, Italy, September 6-11, 2010

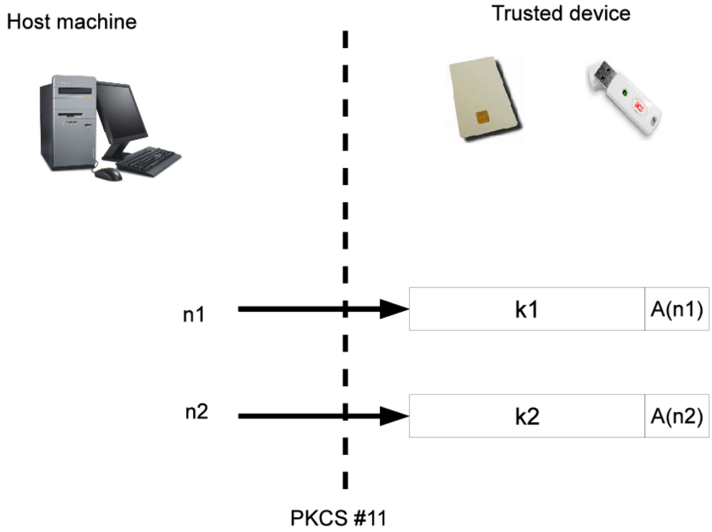
Security APIs



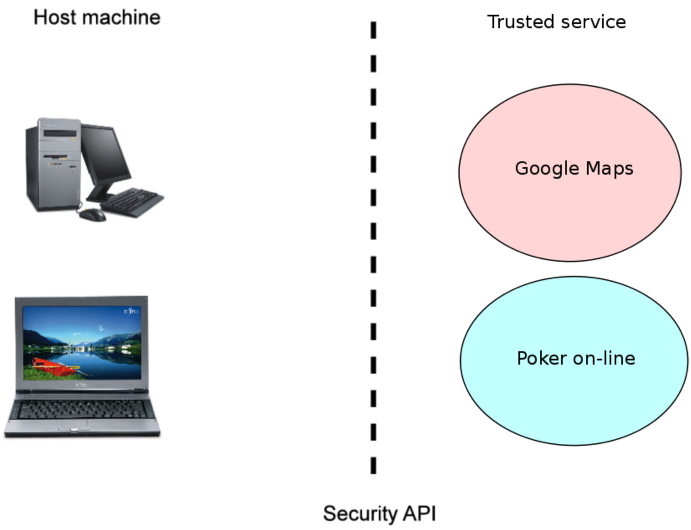
Example 1: Hardware Security Module (HSM)

- Used in the ATM Bank network
- Tamper resistant
- Security API for
  - Managing cryptographic keys
  - Decrypting/re-encrypting the PIN
  - Checking the validity of the PIN

Example 2: PKCS#11 API for tokens/smarcards



Example 3: API to a service or on-line game

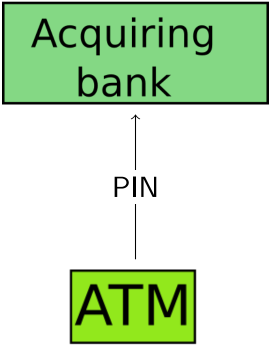


Outline of the course

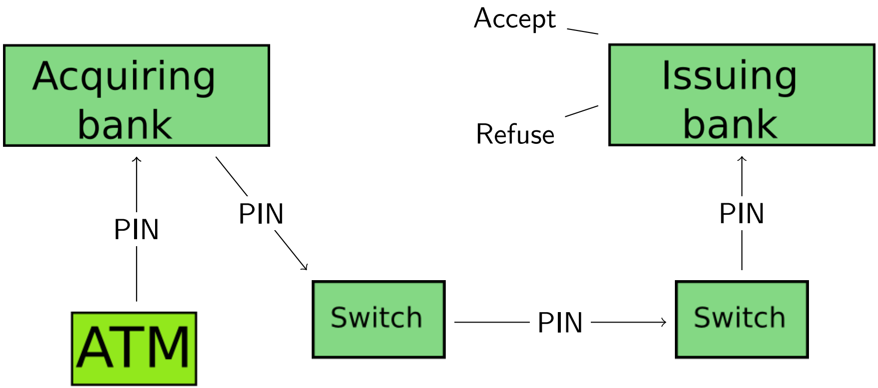
- Today: PIN processing APIs
  - Attacks to guess bank PINs
  - Best strategies to break PINs
  - Language-based analysis and fixes
- Tomorrow: PKCS#11 devices
  - Attacks to compromise a sensitive key
  - A formal model of PKCS#11
  - How to secure PKCS#11: a software token
  - Tookan: Analysis of real tokens

PIN processing infrastructure

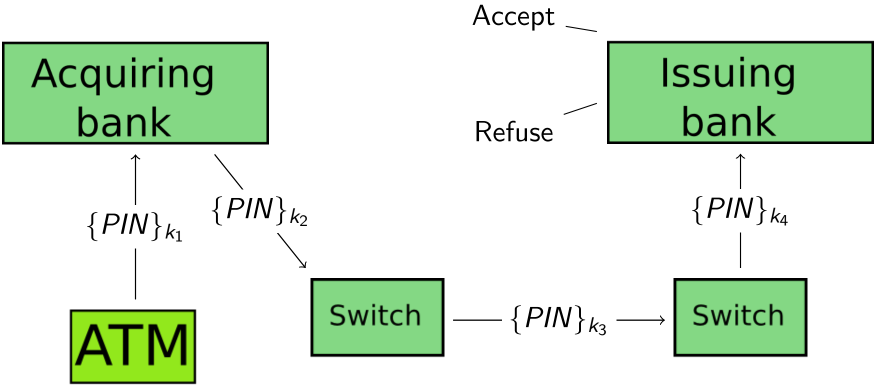
PIN processing infrastructure



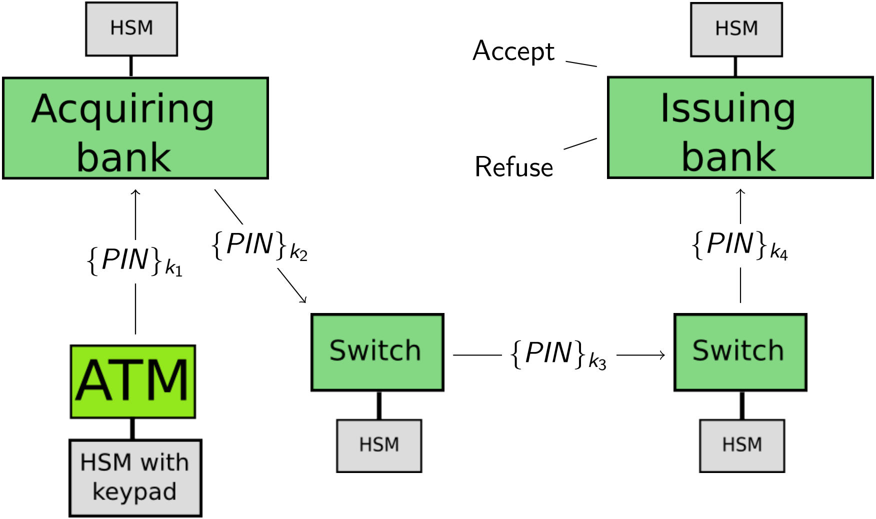
PIN processing infrastructure



PIN processing infrastructure



PIN processing infrastructure



Hardware Security Module (HSM)

- Tamper resistant
- Security API for
  - Managing cryptographic keys
  - Decrypting/re-encrypting the PIN
  - Checking the validity of the PIN

The PIN verification API

- *Encrypted PIN Block* : contains the PIN at the ATM

PIN_V(EPB , vdata,len,dectab,offset)

- Data for computing the user PIN
- Returns the equality of the two PINs

The PIN verification API

- *Encrypted PIN Block* : contains the PIN at the ATM

PIN_V(EPB , vdata,len,dectab,offset)

- Data for computing the user PIN
- Returns the equality of the two PINs

Example: PIN_V($\{4104, r\}_k$,vdata,4,0123456789012345,4732)

The PIN verification API

- *Encrypted PIN Block* : contains the PIN at the ATM

PIN_V(EPB , vdata,len,dectab,offset)

- Data for computing the user PIN
- Returns the equality of the two PINs

Example: PIN_V($\{4104, r\}_k$,vdata,4,0123456789012345,4732)

1. $\text{dec}_k(\{4104, r\}_k) = 4104, r$

The PIN verification API

- *Encrypted PIN Block* : contains the PIN at the ATM

PIN_V(EPB , vdata,len,dectab,offset)

- Data for computing the user PIN
- Returns the equality of the two PINs

Example: PIN_V($\{4104, r\}_k$,vdata,4,0123456789012345,4732)

1. $dec_k(\{4104, r\}_k) = 4104, r$
   4104

The PIN verification API

- *Encrypted PIN Block* : contains the PIN at the ATM

PIN_V(EPB , vdata,len,dectab,offset)

- Data for computing the user PIN
- Returns the equality of the two PINs

Example: PIN_V($\{4104, r\}_k$,vdata,4,0123456789012345,4732)

1. $\mathrm{dec}_k(\{4104, r\}_k) = \quad 4104, r$
   4104
2. $\mathrm{enc}_{pdk}(\mathrm{vdata}) = \quad A47295FDE32A48B1$

The PIN verification API

- *Encrypted PIN Block* : contains the PIN at the ATM

PIN_V(EPB , vdata,len,dectab,offset)

- Data for computing the user PIN
- Returns the equality of the two PINs

Example: PIN_V($\{4104, r\}_k$,vdata,4,0123456789012345,4732)

1. $dec_k(\{4104, r\}_k) = \quad 4104, r$

   4104

2. $enc_{pdk}(\text{vdata}) = \quad A47295FDE32A48B1$

The PIN verification API

- *Encrypted PIN Block* : contains the PIN at the ATM

PIN_V(EPB , vdata,len,dectab,offset)

- Data for computing the user PIN
- Returns the equality of the two PINs

Example: PIN_V($\{4104, r\}_k$,vdata,4,0123456789012345,4732)

1. $dec_k(\{4104, r\}_k) = 4104, r$
   4104
2. $enc_{pdk}(\text{vdata}) = A4729 5FDE32A48B1$
   0472

The PIN verification API

- *Encrypted PIN Block* : contains the PIN at the ATM

PIN_V(EPB , vdata,len,dectab,offset)

- Data for computing the user PIN
- Returns the equality of the two PINs

Example: PIN_V($\{4104, r\}_k$,vdata,4,0123456789012345,4732)

1. $\text{dec}_k(\{4104, r\}_k) = \quad 4104, r$
   4104
2. $\text{enc}_{pdk}(\text{vdata}) = \quad A47295FDE32A48B1$
   $0472 \oplus 4732 \bmod 10 = 4104$

The PIN verification API

- *Encrypted PIN Block* : contains the PIN at the ATM

PIN_V(EPB , vdata,len,dectab,offset)

- Data for computing the user PIN
- Returns the equality of the two PINs

Example: PIN_V($\{4104, r\}_k$,vdata,4,0123456789012345,4732)

1. $\mathrm{dec}_k(\{4104, r\}_k) = \quad 4104, r$
   4104
2. $\mathrm{enc}_{pdk}(\mathrm{vdata}) = \quad A47295FDE32A48B1$
   $0472 \oplus 4732 \bmod 10 = 4104$
3. The two values coincide: PIN_V returns 'true'

The code for PIN verification

$$
\begin{array}{l}
\mathrm{PIN\_V}(\mathit{EPB}, \mathit{vdata}, \mathit{len}, \mathit{dectab}, \mathit{offset})\ \{ \\
\quad x_1 := \mathrm{enc}_{pdk}(\mathit{vdata}); \\
\quad x_2 := \mathrm{left}(\mathit{len}, x_1); \\
\quad x_3 := \mathrm{decimalize}(\mathit{dectab}, x_2); \\
\quad \mathit{u\_pin} := \mathrm{sum\_mod10}(x_3, \mathit{offset}); \\
\\
\quad x_4 := \mathrm{dec}_k(\mathit{EPB}); \\
\quad \mathit{t\_pin} := \mathrm{fcheck}(x_4); \\
\\
\quad \text{if } (\mathit{t\_pin} = \bot) \text{ then return}(''\mathit{format\ wrong}''); \\
\quad \text{if } (\mathit{t\_pin} = \mathit{u\_pin}) \text{ then return}(''\mathit{PIN\ is\ correct}''); \\
\qquad \text{else return}(''\mathit{PIN\ is\ wrong}'') \\
\}
\end{array}
$$

Secure? Against who ... ?

Security property

- Confidentiality: PIN should never be disclosed
- Cards can be cloned, user PIN is the only secret

Secure? Against who ... ?

Security property

- Confidentiality: PIN should never be disclosed
- Cards can be cloned, user PIN is the only secret

Worst case scenario

the attacker works inside the bank

- breaks into the system and have direct access to the HSM
- can perform any sequence of API calls
- ... but no direct access to HSM keys, memory or resources

Secure? Against who ... ?

Security property

- Confidentiality: PIN should never be disclosed
- Cards can be cloned, user PIN is the only secret

Worst case scenario

the attacker works inside the bank

- breaks into the system and have direct access to the HSM
- can perform any sequence of API calls
- ... but no direct access to HSM keys, memory or resources

- But, how can we break PIN secrecy by just calling the API?
  ... have you ever played Mastermind?

The Mastermind Game

- Invented by the Israeli postmaster and telecommunications expert Mordecai Meirowitz in 1970;
- 4 pegs from 6 possible colors, duplicates are allowed.
- The *codemaker* chooses a sequence of 4 pegs, the *codebreaker* has to guess it
- Goal: Minimize the number of guesses

The Mastermind 'API'

- The codebreaker 'calls'

  MasterMind(guess)

  where 'guess' is a sequence of 4 pegs
- The return value is a set of 4 markers:
  - red marker: right color and position;
  - white marker: right color and wrong position.
- Partial information about the secret sequence ...

Can we 'play mastermind' on this API?

- *Encrypted PIN Block* : contains the PIN at the ATM

PIN_V(EPB , vdata,len,dectab,offset)

- Data for computing the user PIN
- Returns the equality of the two PINs

Example: PIN_V($\{4104, r\}_k$,vdata,4,0123456789012345,4732)

1. $\text{dec}_k(\{4104, r\}_k) = 4104, r$
   4104
2. $\text{enc}_{pdk}(\text{vdata}) = A47295FDE32A48B1$
   $0472 \oplus 4732 \bmod 10 = 4104$
3. The two values coincide: PIN_V returns 'true'

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,0123456789012345,4732)

1. $\mathrm{dec}_k(\{4104, r\}_k) = \quad 4104, r$
   4104
2. $\mathrm{enc}_{pdk}(\mathrm{vdata}) = \quad A47295FDE32A48B1$
   $0472 \oplus 4732 \bmod 10 = 4104$
3. PIN_V returns 'true'

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,0123456789012345,4732)

1. $\mathrm{dec}_k(\{4104, r\}_k) = \quad 4104, r$
   4104
2. $\mathrm{enc}_{pdk}(\mathrm{vdata}) = \quad A47295FDE32A48B1$
   $0472 \oplus 4732 \bmod 10 = 4104$
3. PIN_V returns 'true'

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,4732)

1. $\mathrm{dec}_k(\{4104, r\}_k) =$ $4104, r$
   4104
2. $\mathrm{enc}_{pdk}(\mathrm{vdata}) =$ $A47295FDE32A48B1$
   $0472 \oplus 4732 \bmod 10 = 4104$
3. PIN_V returns 'true'

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,4732)

1. $\mathrm{dec}_k(\{4104, r\}_k) = 4104, r$
   4104
2. $\mathrm{enc}_{pdk}(\mathrm{vdata}) = A47295FDE32A48B1$
   $0472 \oplus 4732 \bmod 10 = 4104$
3. PIN_V returns 'true'

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,4732)

1. $\mathrm{dec}_k(\{4104, r\}_k) = \;\; 4104, r$
   4104
2. $\mathrm{enc}_{pdk}(\mathrm{vdata}) = \;\; A47295FDE32A48B1$
   $1472 \oplus 4732 \bmod 10 = 4104$
3. PIN_V returns 'true'

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,4732)

1. $\text{dec}_k(\{4104, r\}_k) = 4104, r$
   4104
2. $\text{enc}_{pdk}(\text{vdata}) = A47295FDE32A48B1$
   $1472 \oplus 4732 \bmod 10 = 4104$
3. PIN_V returns 'true'

The ‘decimalization’ attack on PIN_V [Bond, Zielinski ’03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,4732)

1. $\text{dec}_k(\{4104, r\}_k) = \quad 4104, r$
   4104
2. $\text{enc}_{pdk}(\text{vdata}) = \quad A47295FDE32A48B1$
   $1472 \oplus 4732 \bmod 10 = 5104$
3. PIN_V returns ‘true’

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,4732)

1. $\mathrm{dec}_k(\{4104, r\}_k) = \quad 4104, r$
   4104
2. $\mathrm{enc}_{pdk}(\mathrm{vdata}) = \quad A47295FDE32A48B1$
   $1472 \oplus 4732 \bmod 10 = 5104$
3. PIN_V returns 'true'

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,4732)

1. $dec_k(\{4104, r\}_k) =$ $4104, r$
   4104
2. $enc_{pdk}(\text{vdata}) =$ $A47295FDE32A48B1$
   $1472 \oplus 4732 \bmod 10 = 5104$
3. PIN_V returns 'false'

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,4732)

1. $\text{dec}_k(\{4104, r\}_k) = \quad 4104, r$
   4104
2. $\text{enc}_{pdk}(\text{vdata}) = \quad A47295FDE32A48B1$
   $1472 \oplus 4732 \text{ mod } 10 = 5104$
3. PIN_V returns 'false'

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,3732)

1. $\mathrm{dec}_k(\{4104, r\}_k) = \quad 4104, r$
   4104
2. $\mathrm{enc}_{pdk}(\mathrm{vdata}) = \quad A47295FDE32A48B1$
   $1472 \oplus 3732 \bmod 10 = 5104$
3. PIN_V returns 'false'

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,3732)

1. $\text{dec}_k(\{4104, r\}_k) = \quad 4104, r$
   4104
2. $\text{enc}_{pdk}(\text{vdata}) = \quad A47295FDE32A48B1$
   $1472 \oplus 3732 \bmod 10 = 5104$
3. PIN_V returns 'false'

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,3732)

1. $dec_k(\{4104, r\}_k) =$ $4104, r$
   4104
2. $enc_{pdk}(\text{vdata}) =$ $A47295FDE32A48B1$
   $1472 \oplus 3732 \bmod 10 = 4104$
3. PIN_V returns 'false'

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,3732)

1. $\text{dec}_k(\{4104, r\}_k) = \quad 4104, r$
   4104
2. $\text{enc}_{pdk}(\text{vdata}) = \quad A47295FDE32A48B1$
   $1472 \oplus 3732 \bmod 10 = 4104$
3. PIN_V returns 'false'

The ‘decimalization’ attack on PIN_V [Bond, Zielinski ’03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,3732)

1. $\mathrm{dec}_k(\{4104, r\}_k) = \quad 4104, r$
   4104
2. $\mathrm{enc}_{pdk}(\mathrm{vdata}) = \quad A47295FDE32A48B1$
   $1472 \oplus 3732 \bmod 10 = 4104$
3. PIN_V returns ‘true’

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,3732)

1. $\text{dec}_k(\{4104, r\}_k) = \quad 4104, r$
   4104
2. $\text{enc}_{pdk}(\text{vdata}) = \quad A47295FDE32A48B1$
   $1472 \oplus 3732 \bmod 10 = 4104$
3. PIN_V returns 'true'

- We discover that the first digit is 4 with 2 API calls, being lucky

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,3732)

1. $\text{dec}_k(\{4104, r\}_k) =$ $4104, r$
   4104
2. $\text{enc}_{pdk}(\text{vdata}) =$ $A47295FDE32A48B1$
   $1472 \oplus 3732 \bmod 10 = 4104$
3. PIN_V returns 'true'

- We discover that the first digit is 4 with 2 API calls, being lucky
- Has this kind of attack been tried on real bank systems?

The 'decimalization' attack on PIN_V [Bond, Zielinski '03]

PIN_V($\{4104, r\}_k$,vdata,4,1123456789112345,3732)

1. $\text{dec}_k(\{4104, r\}_k) = \quad 4104, r$
   4104
2. $\text{enc}_{pdk}(\text{vdata}) = \quad A47295FDE32A48B1$
   $1472 \oplus 3732 \bmod 10 = 4104$
3. PIN_V returns 'true'

- We discover that the first digit is 4 with 2 API calls, being lucky
- Has this kind of attack been tried on real bank systems?
- How long does it take to discover the whole PIN?

Reports suggest something has been going on ...

Verizon Breach Report 2008

"Were seeing entirely new attacks that a year ago were thought to be only academically possible"
"What we see now is people going right to the source [..] and stealing the encrypted PIN blocks and using complex ways to un-encrypt the PIN blocks."
(Quotes from Wired Magazine interview with report author, Bryan Sartin)

How many API calls are needed?

For a four digit PIN:

- [Bond, Zielinski '03] Average 16.5 API calls
- [Steel, TCS06] Average 16.145 API calls
- [Focardi, Luccio, FUN'10] Average 14.47 API calls (as instance of Mastermind)
- Lower-bound of 13.362 API calls

The Extended Mastermind Game

- Colors: $\mathcal{C} = \{0, 1, \ldots, N-1\}$
- Secret sequence: $(c_1, c_2, \ldots, c_k)$, with $c_1, \ldots, c_k \in \mathcal{C}$
- Extended guess: $(S_1, S_2, \ldots, S_k)$, with $S_1, \ldots, S_k \subseteq \mathcal{C}$

The Extended Mastermind Game

- Colors: $\mathcal{C} = \{0, 1, \ldots, N-1\}$
- Secret sequence: $(c_1, c_2, \ldots, c_k)$, with $c_1, \ldots, c_k \in \mathcal{C}$
- Extended guess: $(S_1, S_2, \ldots, S_k)$, with $S_1, \ldots, S_k \subseteq \mathcal{C}$

Example

- 6 colors: $\mathcal{C} = \{0, 1, \ldots, 5\}$
- Secret: $(1, 2, 3, 1)$
- Extended guess: $(\{1\}, \{3\}, \{1\}, \{1, 3\})$
- what's the answer?

The Extended Mastermind Game

- Colors: $\mathcal{C} = \{0, 1, \ldots, N-1\}$
- Secret sequence: $(c_1, c_2, \ldots, c_k)$, with $c_1, \ldots, c_k \in \mathcal{C}$
- Extended guess: $(S_1, S_2, \ldots, S_k)$, with $S_1, \ldots, S_k \subseteq \mathcal{C}$

Example

- 6 colors: $\mathcal{C} = \{0, 1, \ldots, 5\}$
- Secret: $(1, 2, 3, 1)$
- Extended guess: $(\{1\}, \{3\}, \{1\}, \{1, 3\})$
- what's the answer?

2 red and 1 white markers

Red Markers

- Colors: $\mathcal{C} = \{0, 1, \ldots, N-1\}$
- Secret sequence: $(c_1, c_2, \ldots, c_k)$, with $c_1, \ldots, c_k \in \mathcal{C}$
- Extended guess: $(S_1, S_2, \ldots, S_k)$, with $S_1, \ldots, S_k \subseteq \mathcal{C}$

Definition (Red markers)

The number b of red markers is computed as $r = |\{i \in [1, k] \mid c_i \in S_i\}|$.

Red Markers

- Colors: $\mathcal{C} = \{0, 1, \ldots, N-1\}$
- Secret sequence: $(c_1, c_2, \ldots, c_k)$, with $c_1, \ldots, c_k \in \mathcal{C}$
- Extended guess: $(S_1, S_2, \ldots, S_k)$, with $S_1, \ldots, S_k \subseteq \mathcal{C}$

Definition (Red markers)

The number b of red markers is computed as $r = |\{i \in [1, k] \mid c_i \in S_i\}|$.

Example

- Secret: $(1, 2, 3, 1)$
- Extended guess: $(\{1\}, \{3\}, \{1\}, \{1, 3\})$
- $r = |\{i \in [1, k] \mid c_i \in S_i\}| = 2$

White Markers

- Secret : $(c_1, c_2, \ldots, c_k)$, Extended guess: $(S_1, S_2, \ldots, S_k)$

- $p_j = |\{i \in [1, k] \mid j = c_i\}|$ occurrences of a color j in the secret
- $q_j = |\{i \in [1, k] \mid j \in S_i\}|$ occurrences of a color j in the guess

Definition (White markers)

The number w of white markers is computed as $w = \sum_{j=1}^{N} min(p_j, q_j) - r$.

White Markers

- Secret : $(c_1, c_2, \ldots, c_k)$, Extended guess: $(S_1, S_2, \ldots, S_k)$
- $p_j = |\{i \in [1,k] \mid j = c_i\}|$ occurrences of a color j in the secret
- $q_j = |\{i \in [1,k] \mid j \in S_i\}|$ occurrences of a color j in the guess

Definition (White markers)

The number w of white markers is computed as $w = \sum_{j=1}^{N} min(p_j, q_j) - r$.

Example

- Secret $(1,2,3,1)$ and extended guess $(\{1\},\{3\},\{1\},\{1,3\})$:
- $p_1 = |\{1,4\}| = 2,\ q_1 = |\{1,3,4\}| = 3, min(p_1, q_1) = 2$
- $p_2 = 1,\ q_2 = 0, min(p_2, q_2) = 0;\ p_3 = 1,\ q_3 = 2, min(p_3, q_3) = 1$
- $w = \sum_{j=1}^{N} min(p_j, q_j) - r = 2 + 0 + 1 - 2 = 1$

We can still play Mastermind

Proposition

The Mastermind game is an instance of the Extended game

Proof.

Trivial: just restrict the sets in the estended guesses to singletons. □

Cracking a PIN by playing extended Mastermind

Theorem

PIN cracking is an instance of the Extended Mastermind game

Proof.

Intuition: Restrict to cases in which guesses $(S_1, S_2, \ldots, S_k)$ minus offset provide either equal or disjoint sets.

1. Modify the dectab mapping of all elements of the i-th set from d to $d + i \pmod{10}$
2. Compensate by $-i \pmod{10}$ the offset in the corresponding positions to find out whether those PIN digits are in the set.

The answer is four red markers if and only if PIN verification succeeds. □

Example

Example: PIN_V($\{4104, r\}_k$,vdata,4, 0123456789012345,4732)

1. $dec_k(\{4104, r\}_k) = \; 4104, r$
   4104
2. $enc_{pdk}(\text{vdata}) = \; A47295FDE32A48B1$
   $0472 \oplus 4732 \bmod 10 = 4104$

- We play: $(\{4, 5, 6, 7, 8\}, \{0, 1, 7, 8, 9\}, \{0, 1\}, \{2, 3, 4, 5, 6\})$

Example

Example: PIN_V($\{4104, r\}_k$,vdata,4, 0123456789012345,4732)

1. $\mathrm{dec}_k(\{4104, r\}_k) = 4104, r$

   4104

2. $\mathrm{enc}_{pdk}(\mathrm{vdata}) = A47295FDE32A48B1$

   $0472 \oplus 4732 \bmod 10 = 4104$

- We play: $(\{4,5,6,7,8\},\{0,1,7,8,9\},\{0,1\},\{2,3,4,5,6\})$
- Subtract the offset: $(\{0,1,2,3,4\},\{0,1,2,3,4\},\{7,8\},\{0,1,2,3,4\})$
- Two disjoint sets: $\{0,1,2,3,4\},\{7,8\}$, change the dectab

Example

Example: PIN_V($\{4104, r\}_k$,vdata,4, 0123456789012345,4732)

1. $\text{dec}_k(\{4104, r\}_k) = 4104, r$
   4104
2. $\text{enc}_{pdk}(\text{vdata}) = A47295FDE32A48B1$
   $0472 \oplus 4732 \bmod 10 = 4104$

- We play: $(\{4, 5, 6, 7, 8\}, \{0, 1, 7, 8, 9\}, \{0, 1\}, \{2, 3, 4, 5, 6\})$
- Subtract the offset: $(\{0, 1, 2, 3, 4\}, \{0, 1, 2, 3, 4\}, \{7, 8\}, \{0, 1, 2, 3, 4\})$
- Two disjoint sets: $\{0, 1, 2, 3, 4\}, \{7, 8\}$, change the dectab

Example

Example: PIN_V($\{4104, r\}_k$,vdata,4, 1234556909123455,4732)

1. $\text{dec}_k(\{4104, r\}_k) =$ $4104, r$
   4104
2. $\text{enc}_{pdk}(\text{vdata}) =$ $A47295FDE32A48B1$
   $0472 \oplus 4732 \bmod 10 = 4104$

- We play: $(\{4, 5, 6, 7, 8\}, \{0, 1, 7, 8, 9\}, \{0, 1\}, \{2, 3, 4, 5, 6\})$
- Subtract the offset: $(\{0, 1, 2, 3, 4\}, \{0, 1, 2, 3, 4\}, \{7, 8\}, \{0, 1, 2, 3, 4\})$
- Two disjoint sets: $\{0, 1, 2, 3, 4\}, \{7, 8\}$, change the dectab

Example

Example: PIN_V($\{4104, r\}_k$,vdata,4, 1234556909123455,4732)

1. $\mathrm{dec}_k(\{4104, r\}_k) = 4104, r$
   4104
2. $\mathrm{enc}_{pdk}(\mathrm{vdata}) = A47295FDE32A48B1$
   $0472 \oplus 4732 \bmod 10 = 4104$

- We play: $(\{4, 5, 6, 7, 8\}, \{0, 1, 7, 8, 9\}, \{0, 1\}, \{2, 3, 4, 5, 6\})$
- Subtract the offset: $(\{0, 1, 2, 3, 4\}, \{0, 1, 2, 3, 4\}, \{7, 8\}, \{0, 1, 2, 3, 4\})$
- Two disjoint sets: $\{0, 1, 2, 3, 4\}, \{7, 8\}$, change the dectab
- Compensate the offset

Example

Example: PIN_V($\{4104, r\}_k$,vdata,4, 1234556909123455,3611)

1. $\text{dec}_k(\{4104, r\}_k) = 4104, r$
   4104
2. $\text{enc}_{pdk}(\text{vdata}) = A47295FDE32A48B1$
   $0472 \oplus 4732 \bmod 10 = 4104$

- We play: $(\{4, 5, 6, 7, 8\}, \{0, 1, 7, 8, 9\}, \{0, 1\}, \{2, 3, 4, 5, 6\})$
- Subtract the offset: $(\{0, 1, 2, 3, 4\}, \{0, 1, 2, 3, 4\}, \{7, 8\}, \{0, 1, 2, 3, 4\})$
- Two disjoint sets: $\{0, 1, 2, 3, 4\}, \{7, 8\}$, change the dectab
- Compensate the offset

Example

Example: PIN_V($\{4104, r\}_k$,vdata,4, 1234556909123455,3611)

1. $\text{dec}_k(\{4104, r\}_k) = 4104, r$
   4104
2. $\text{enc}_{pdk}(\text{vdata}) = A47295FDE32A48B1$
   $0472 \oplus 4732 \bmod 10 = 4104$

- We play: $(\{4,5,6,7,8\},\{0,1,7,8,9\},\{0,1\},\{2,3,4,5,6\})$
- Subtract the offset: $(\{0,1,2,3,4\},\{0,1,2,3,4\},\{7,8\},\{0,1,2,3,4\})$
- Two disjoint sets: $\{0,1,2,3,4\},\{7,8\}$, change the dectab
- Compensate the offset

Example

Example: PIN_V($\{4104, r\}_k$,vdata,4, 1234556909123455,3611)

1. $\mathrm{dec}_k(\{4104, r\}_k) = 4104, r$
   4104
2. $\mathrm{enc}_{pdk}(\mathrm{vdata}) = A47295FDE32A48B1$
   $1593 \oplus 3611 \bmod 10 = 4104$

- We play: $(\{4, 5, 6, 7, 8\}, \{0, 1, 7, 8, 9\}, \{0, 1\}, \{2, 3, 4, 5, 6\})$
- Subtract the offset: $(\{0, 1, 2, 3, 4\}, \{0, 1, 2, 3, 4\}, \{7, 8\}, \{0, 1, 2, 3, 4\})$
- Two disjoint sets: $\{0, 1, 2, 3, 4\}, \{7, 8\}$, change the dectab
- Compensate the offset

Example

Example: PIN_V($\{4104, r\}_k$,vdata,4, 1234567909123455,3611)

1. $\mathrm{dec}_k(\{4104, r\}_k) = 4104, r$
   4104
2. $\mathrm{enc}_{pdk}(\mathrm{vdata}) = A47295FDE32A48B1$
   $1593 \oplus 3611 \bmod 10 = 4104$

- We play: $(\{4, 5, 6, 7, 8\}, \{0, 1, 7, 8, 9\}, \{0, 1\}, \{2, 3, 4, 5, 6\})$
- Subtract the offset: $(\{0, 1, 2, 3, 4\}, \{0, 1, 2, 3, 4\}, \{7, 8\}, \{0, 1, 2, 3, 4\})$
- Two disjoint sets: $\{0, 1, 2, 3, 4\}, \{7, 8\}$, change the dectab
- Compensate the offset
- PIN_V returns ‘true’ iff PIN digits are in the sets

An algorithm for the Extended Mastermind Problem

Based on [Knuth JRM76]: an algorithm for the solution of the standard Mastermind problem (quasi optimal solutions).

1. Tries all the possible guesses. For each guess, computes the number of 'surviving' solutions related to each possible outcome of the guess;
2. Picks the guess from the previous step which minimizes the maximum number of surviving solutions among all the possible outcomes and performs the guess.

An algorithm for the Extended Mastermind Problem

Based on [Knuth JRM76]: an algorithm for the solution of the standard Mastermind problem (quasi optimal solutions).

1. Tries all the possible guesses. For each guess, computes the number of ‘surviving’ solutions related to each possible outcome of the guess;
2. Picks the guess from the previous step which minimizes the maximum number of surviving solutions among all the possible outcomes and performs the guess.

Focus on two kinds of guesses:

- $(\{0,1,2,3,4,5\},\{0,1,2,3,4,5\},\{0,1,2,3,4,5\},\{0,1,2,3,4,5\})$, the same set repeated: checks if 6,7,8,9 are in the PIN
- $(\{1,3\},\{0,2,4,5,6,7,8,9\},\{0,2,4,5,6,7,8,9\},\{1,3\})$, one set and its complementary
- perform very well and still find a complete strategy

Summary of results for PIN cracking

Four digit PINs

- [Bond, Zielinski '03] Average 16.5 API calls
- [Steel, TCS06] Average 16.145 API calls
- [Focardi, Luccio, FUN'10] Average 14.47 API calls
  with the previous algorithm in Python on this laptop $\approx$ 18 seconds

Summary of results for PIN cracking

Four digit PINs

- [Bond, Zielinski '03] Average 16.5 API calls
- [Steel, TCS06] Average 16.145 API calls
- [Focardi, Luccio, FUN'10] Average 14.47 API calls
  with the previous algorithm in Python on this laptop $\approx$ 18 seconds

Five digit PINs

- [Focardi, Luccio, FUN'10] Average 19.3 API calls $\approx$ 18 minutes

Summary of results for PIN cracking

Four digit PINs

- [Bond, Zielinski '03] Average 16.5 API calls
- [Steel, TCS06] Average 16.145 API calls
- [Focardi, Luccio, FUN'10] Average 14.47 API calls
  with the previous algorithm in Python on this laptop $\approx$ 18 seconds

Five digit PINs

- [Focardi, Luccio, FUN'10] Average 19.3 API calls $\approx$ 18 minutes

Lower bounds

- The lower bounds for 4 and 5 digit PINs are 13.362 and 16.689, for the average case

The ‘lunch-break’ attack

A realistic scenario

gaining access to the HSM and intercepting incoming data an insider might disclose thousands of PINs in a lunch-break!

The ‘lunch-break’ attack

A realistic scenario

gaining access to the HSM and intercepting incoming data an insider might disclose thousands of PINs in a lunch-break!

How to prevent the attack?

- low-impact CVV-based fix [Focardi, Luccio, Steel, NORDSEC’09]
  - **mitigates** the attack (50000 times slower)
- point-to-point MAC-based fix and type-based proof of security [Centenaro, Focardi, Luccio, Steel, ESORICS’09]
  - **prevents** the attack but requires modifying each HSM
- efficient HSM upgrading strategies [Focardi, Luccio, ARSPA-WITS’10]
  - **securing subnetworks** while keeping service up

What kind of attack?

What kind of attack?

- no cryptoanalysis and no broken protocols

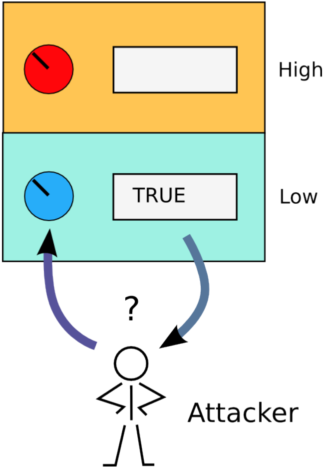
What kind of attack?

- no cryptoanalysis and no broken protocols
- Information-flow: variations on the input produce unintended *information leakage*

Absence of information leakage [Goguen, Meseguer'82]

Noninterference

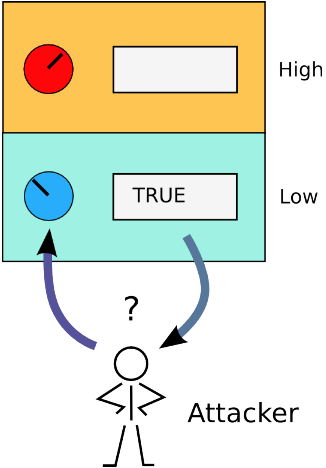
High behaviour is not observable by the Low attacker



Absence of information leakage [Goguen, Meseguer'82]

Noninterference

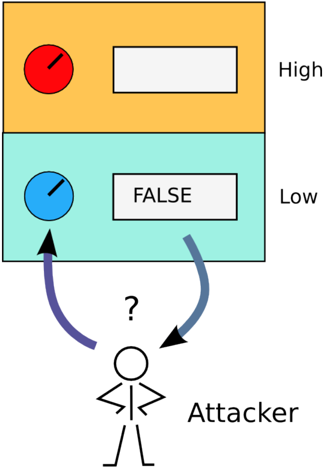
High behaviour is not observable by the Low attacker



Absence of information leakage [Goguen, Meseguer'82]

Noninterference

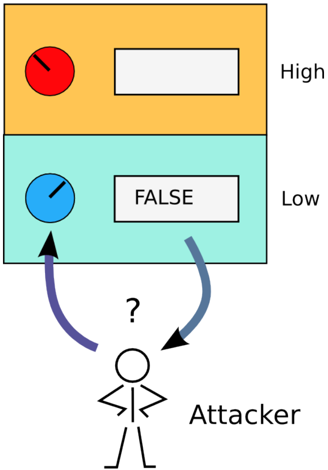
High behaviour is not observable by the Low attacker



Absence of information leakage [Goguen, Meseguer'82]

Noninterference

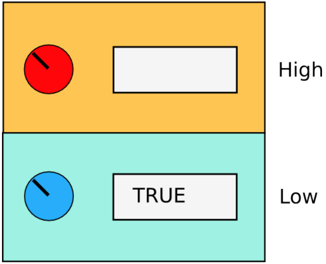
High behaviour is not observable by the Low attacker



Noninterference is too much

$$\text{PIN\_V}(\{\,4104\,, r\}_k, \text{vdata}, 4, 0123456789012345, 4732)$$

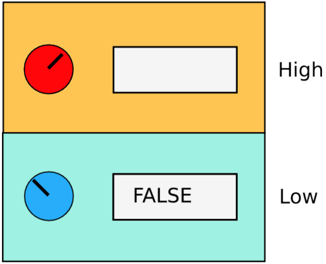
- PIN_V intentionally 'leaks' the correctness of the PIN



Noninterference is too much

PIN_V($\{$ 5832 $, r\}_k$,vdata, 4, 0123456789012345, 4732)

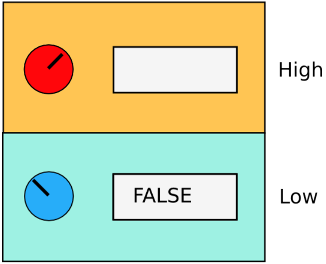
- PIN_V intentionally ‘leaks’ the correctness of the PIN



Noninterference is too much

PIN_V($\{$5832$, r\}_k$,vdata, 4, 0123456789012345, 4732)

- PIN_V intentionally 'leaks' the correctness of the PIN
- PIN correctness is *declassified*

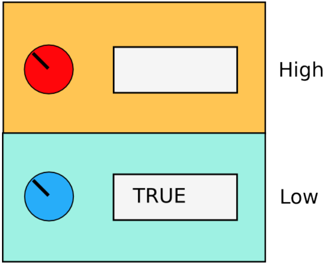


Robust declassification [Myers, Sabelfeld, Zdancewic '06]

Robustness

Declassification is independent of the attacker behaviour

- the attacker cannot cause to release more information than intended

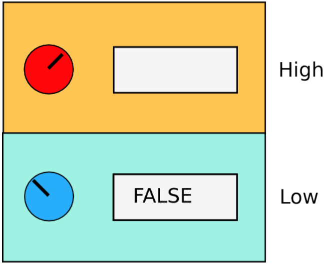


Robust declassification [Myers, Sabelfeld, Zdancewic '06]

Robustness

Declassification is independent of the attacker behaviour

- the attacker cannot cause to release more information than intended

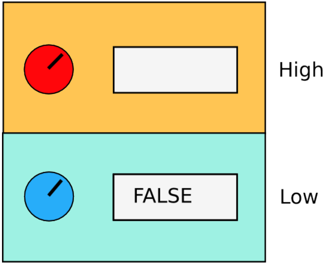


Robust declassification [Myers, Sabelfeld, Zdancewic '06]

Robustness

Declassification is independent of the attacker behaviour

- the attacker cannot cause to release more information than intended

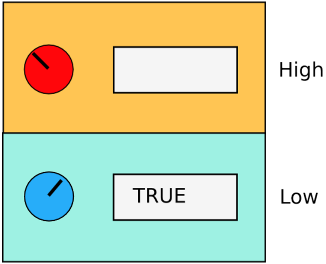


Robust declassification [Myers, Sabelfeld, Zdancewic '06]

Robustness

Declassification is independent of the attacker behaviour

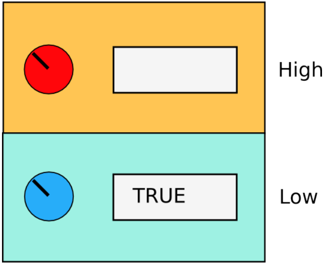
- the attacker cannot cause to release more information than intended



PIN_V is not robust!

PIN_V($\{$ 4104 $, r\}_k$,vdata, 4, 0123456789012345 , 4732)

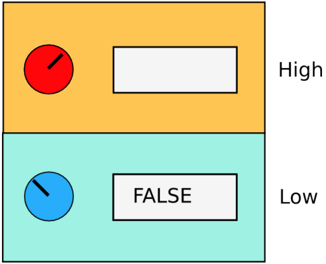
- PIN correctness is declassified



PIN_V is not robust!

PIN_V({ 5832 , $r\}_k$,vdata, 4, 0123456789012345 , 4732)

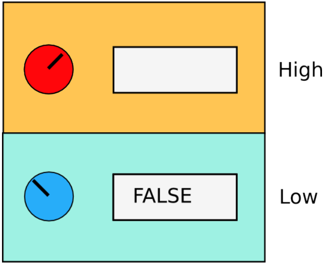
- PIN correctness is declassified



PIN_V is not robust!

PIN_V({ 5832 , $r\}_k$,vdata, 4, 0123456789012345 , 4732)

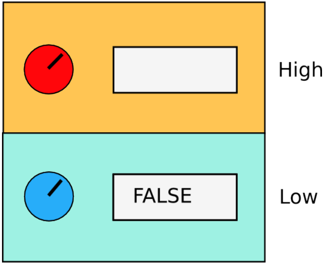
- PIN correctness is declassified
- the insider tries a decimalization attack



PIN_V is not robust!

PIN_V({ 5832 , r}$_k$,vdata, 4, 1123456789112345 , 4732)

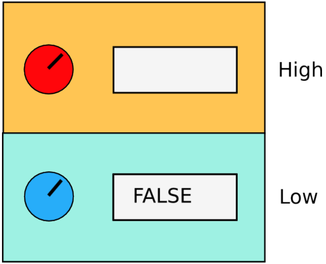
- PIN correctness is declassified
- the insider tries a decimalization attack



PIN_V is not robust!

PIN_V({ 5832 , $r\}_k$,vdata, 4, 1123456789112345 , 4732)

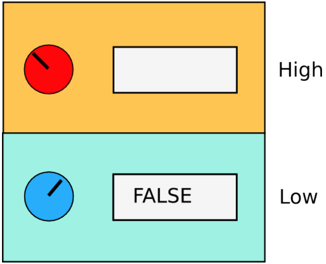
- PIN correctness is declassified
- the insider tries a decimalization attack
- PIN_V now fails in both cases



PIN_V is not robust!

PIN_V({ 4104 , $r\}_k$,vdata, 4, 1123456789112345 , 4732)

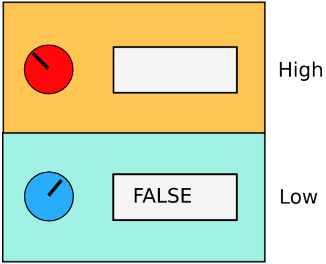
- PIN correctness is declassified
- the insider tries a decimalization attack
- PIN_V now fails in both cases



PIN_V is not robust!

PIN_V($\{$ 4104 , $r\}_k$,vdata, 4, 1123456789112345 , 4732)

- PIN correctness is declassified
- the insider tries a decimalization attack
- PIN_V now fails in both cases
- the attacker has influenced declassification



The code for PIN verification, what is wrong ... ?

$PIN\_V(EPB, vdata, len, dectab, offset)$ {

$x_1 := enc_{pdk}(vdata)$;

$x_2 := left(len, x_1)$;

$x_3 := decimalize(dectab, x_2)$;

$u\_pin := sum\_mod10(x_3, offset)$;

$x_4 := dec_k(EPB)$;

$t\_pin := fcheck(x_4)$;

if $(t\_pin = \perp)$ then return("*format wrong*");

if $(t\_pin = u\_pin)$ then return("*PIN is correct*");

else return("*PIN is wrong*")

}

How to be robust?

- declassify high-integrity data in high-integrity program points

How to be robust?

- declassify high-integrity data in high-integrity program points

  declassify($x_H = y_H$);

How to be robust?

- declassify high-integrity data in high-integrity program points
  - declassify($x_H = y_H$);

How to be robust?

- declassify high-integrity data in high-integrity program points
  - 😎 declassify($x_H = y_H$);
    declassify($x_H = z_L$);

How to be robust?

- declassify high-integrity data in high-integrity program points
  - declassify($x_H = y_H$);
  - declassify($x_H = z_L$);

How to be robust?

- declassify high-integrity data in high-integrity program points
  - declassify($x_H = y_H$);
  - declassify($x_H = z_L$);
    if z_L declassify(x_H);

How to be robust?

- declassify high-integrity data in high-integrity program points
  - 😎 declassify($x_H = y_H$);
  - 👎 declassify($x_H = z_L$);
  - 👎 if z_L declassify(x_H);

How to be robust?

- declassify high-integrity data in high-integrity program points
  - 😎 declassify($x_H = y_H$);
  - 👎 declassify($x_H = z_L$);
  - 👎 if z_L declassify(x_H);
- user PIN u_pin is computed from low-integrity data

PIN_V({t_pin, r}$_k$, vdata,len,dectab,offset)

How to be robust?

- declassify high-integrity data in high-integrity program points
  - 😎 declassify($x_H = y_H$);
  - 👎 declassify($x_H = z_L$);
  - 👎 if z_L declassify(x_H);

- user PIN u_pin is computed from low-integrity data

PIN_V($\{$t_pin, r$\}_k$, vdata,len,dectab,offset)

declassify(t_pin = u_pin)

How to be robust?

- declassify high-integrity data in high-integrity program points
  - 😎 declassify($x_H = y_H$);
  - 👎 declassify($x_H = z_L$);
  - 👎 if z_L declassify(x_H);

- user PIN u_pin is computed from low-integrity data

PIN_V($\{$t_pin, r$\}_k$, vdata,len,dectab,offset)

- 👎 declassify(t_pin = u_pin)

Fixing PIN_V

- add a *Message Authentication Code*

$$m = \langle \{\text{t\_pin}, r\}_k, \text{vdata}, \text{len}, \text{dectab}, \text{offset} \rangle_j$$

```
PIN_V+( {t_pin, r}_k, vdata,len,dectab,offset, m ) {
    if mac_j( {t_pin, r}_k, vdata,len,dectab,offset ) = m
        ... old PIN_V code ...
    else
        FAIL
}
```

- MAC guarantees that data come from a specific user

Integrity representatives

- MAC creation has to be regulated
- with these three MACs the attacker can get the first PIN digit

$$\langle\ \{4104, r\}_k,\ \text{vdata}, 4, 0123456789012345, 4732\ \rangle_j$$
$$\langle\ \{4104, r\}_k,\ \text{vdata}, 4, 1123456789112345, 4732\ \rangle_j$$
$$\langle\ \{4104, r\}_k,\ \text{vdata}, 4, 1123456789112345, 3732\ \rangle_j$$

Integrity representatives

- MAC creation has to be regulated
- with these three MACs the attacker can get the first PIN digit

$$\langle\ \{4104,\ r\}_k,\ \text{vdata},\ 4,\ 0123456789012345,\ 4732\ \rangle_j$$
$$\langle\ \{4104,\ r\}_k,\ \text{vdata},\ 4,\ 1123456789112345,\ 4732\ \rangle_j$$
$$\langle\ \{4104,\ r\}_k,\ \text{vdata},\ 4,\ 1123456789112345,\ 3732\ \rangle_j$$

vdata is the *integrity representative*: len, dectab, offset and the PIN are 'determined' by vdata

Integrity representatives

- MAC creation has to be regulated
- with these three MACs the attacker can get the first PIN digit

$$\langle\ \{4104, r\}_k,\ \text{vdata}, 4, 0123456789012345, 4732\ \rangle_j$$
$$\langle\ \{4104, r\}_k,\ \text{vdata}, 4, 1123456789112345, 4732\ \rangle_j$$
$$\langle\ \{4104, r\}_k,\ \text{vdata}, 4, 1123456789112345, 3732\ \rangle_j$$

vdata is the *integrity representative*: len, dectab, offset and the PIN are 'determined' by vdata

More precisely, in the two MACs

$$\langle\{pin_1, r_1\}_k, \boxed{\text{vdata}}, len_1, dectab_1, offset_1\rangle_j$$
$$\langle\{pin_2, r_2\}_k, \boxed{\text{vdata}}, len_2, dectab_2, offset_2\rangle_j$$

we require $len_1 = len_2$, $dectab_1 = dectab_2$, $offset_1 = offset_2$, $pin_1 = pin_2$

... still not robust!

user 1 has inserted the correct PIN:

$$PIN\_V^{+}(EPB_1, vdata_1, len_1, dectab_1, offset_1, m_1) \rightarrow \text{true}$$

user 2 has typed a wrong PIN:

$$PIN\_V^{+}(EPB_2, vdata_2, len_2, dectab_2, offset_2, m_2) \rightarrow \text{false}$$

- the attacker affects declassification by calling PIN_V^{+} with data from completely different users

... still not robust!

user 1 has inserted the correct PIN:

$$\text{PIN\_V}^{+}(\text{EPB}_1, \text{vdata}_1, \text{len}_1, \text{dectab}_1, \text{offset}_1, \text{m}_1) \rightarrow \text{true}$$

user 2 has typed a wrong PIN:

$$\text{PIN\_V}^{+}(\text{EPB}_2, \text{vdata}_2, \text{len}_2, \text{dectab}_2, \text{offset}_2, \text{m}_2) \rightarrow \text{false}$$

- the attacker affects declassification by calling PIN_V^{+} with data from completely different users
- ... he only gets the expected outputs

... still not robust!

user 1 has inserted the correct PIN:

$$\text{PIN\_V}^{+}(\text{EPB}_1, \text{vdata}_1, \text{len}_1, \text{dectab}_1, \text{offset}_1, \text{m}_1) \rightarrow \text{true}$$

user 2 has typed a wrong PIN:

$$\text{PIN\_V}^{+}(\text{EPB}_2, \text{vdata}_2, \text{len}_2, \text{dectab}_2, \text{offset}_2, \text{m}_2) \rightarrow \text{false}$$

- the attacker affects declassification by calling PIN_V^{+} with data from completely different users
- ... he only gets the expected outputs

disallow changes to vdata

PIN_V$^+$ is robust, for each user

$$\text{PIN\_V}^+(\ \{\text{pin, r}\}_k, \text{vdata, len, dectab, offset,} \\ \langle\ \{\text{pin, r}\}_k, \text{vdata, len, dectab, offset}\ \rangle_j\ )$$

PIN_V$^+$ is robust, for each user

$$\text{PIN\_V}^+(\ \{\text{pin, r}\}_k, \text{vdata}, \text{len, dectab, offset}, \langle\ \{\text{pin, r}\}_k, \text{vdata, len, dectab, offset}\ \rangle_j\ )$$

1. vdata cannot be manipulated (focus on one user)

PIN_V$^+$ is robust, for each user

$$\text{PIN\_V}^+(\ \{\text{pin, r}\}_k, \text{vdata, len, dectab, offset,}\ \langle\ \{\text{pin, r}\}_k, \text{vdata, len, dectab, offset}\ \rangle_j\ )$$

1. vdata cannot be manipulated (focus on one user)
2. vdata in the MAC must correspond

PIN_V$^+$ is robust, for each user

$$\text{PIN\_V}^+(\ \{\text{pin}, \text{r}\}_k, \text{vdata}, \text{len}, \text{dectab}, \text{offset}, \langle\ \{\text{pin}, \text{r}\}_k, \text{vdata}, \text{len}, \text{dectab}, \text{offset}\ \rangle_j\ )$$

1. vdata cannot be manipulated (focus on one user)
2. vdata in the MAC must correspond
3. pin, len, dectab and offset in the MAC are determined

PIN_V$^+$ is robust, for each user

$$\text{PIN\_V}^+(\ \{\text{pin}, \text{r}\}_k, \text{vdata}, \text{len}, \text{dectab}, \text{offset}, \langle\ \{\text{pin}, \text{r}\}_k, \text{vdata}, \text{len}, \text{dectab}, \text{offset}\ \rangle_j\ )$$

1. vdata cannot be manipulated (focus on one user)
2. vdata in the MAC must correspond
3. pin, len, dectab and offset in the MAC are determined
4. ... and must correspond to the ones outside

PIN_V$^+$ is robust, for each user

$$\text{PIN\_V}^+(\ \{\text{pin}, \text{r}\}_k, \text{vdata}, \text{len}, \text{dectab}, \text{offset}, \langle\ \{\text{pin}, \text{r}\}_k, \text{vdata}, \text{len}, \text{dectab}, \text{offset}\ \rangle_j\ )$$

1. vdata cannot be manipulated (focus on one user)
2. vdata in the MAC must correspond
3. pin, len, dectab and offset in the MAC are determined
4. ... and must correspond to the ones outside

Note: The only possible change is in the random padding r

$\Rightarrow$ PIN_V$^+$ result cannot be influenced by the attacker

$\Rightarrow$ PIN_V$^+$ is robust

PIN_V$^+$ is robust, for each user

$$\text{PIN\_V}^+(\ \{\text{pin}, \text{r}\}_k, \text{vdata}, \text{len}, \text{dectab}, \text{offset}, \langle \{\text{pin}, \text{r}\}_k, \text{vdata}, \text{len}, \text{dectab}, \text{offset} \rangle_j\ )$$

1. vdata cannot be manipulated (focus on one user)
2. vdata in the MAC must correspond
3. pin, len, dectab and offset in the MAC are determined
4. ... and must correspond to the ones outside

Note: The only possible change is in the random padding r

$\Rightarrow$ PIN_V$^+$ result cannot be influenced by the attacker

$\Rightarrow$ PIN_V$^+$ is robust

- vdata : [VDATA] , dectab : [DTAB$\hookleftarrow$VDATA]

The ISO0 PIN block format

- encrypting a secret produces a low-confidentiality ciphertext
- Noninterference: changes to the secret plaintexts should be unobservable
- not true in general: $\{1234\}_k = \{1234\}_k$

The ISO0 PIN block format

- encrypting a secret produces a low-confidentiality ciphertext
- Noninterference: changes to the secret plaintexts should be unobservable
- not true in general: $\{1234\}_k = \{1234\}_k$
  $\{1234\}_k \neq \{5678\}_k$

The ISO0 PIN block format

- encrypting a secret produces a low-confidentiality ciphertext
- Noninterference: changes to the secret plaintexts should be unobservable
- not true in general: $\{1234\}_k = \{1234\}_k$
  $\{1234\}_k \neq \{5678\}_k$
- randomization helps: $\{1234, r\}_k \neq \{1234, r'\}_k$

The ISO0 PIN block format

- encrypting a secret produces a low-confidentiality ciphertext
- Noninterference: changes to the secret plaintexts should be unobservable
- not true in general: $\{1234\}_k = \{1234\}_k$
  $\{1234\}_k \neq \{5678\}_k$
- randomization helps: $\{1234, r\}_k \neq \{1234, r'\}_k$
  $\{1234, r\}_k \neq \{5678, r'\}_k$

The ISO0 PIN block format

- encrypting a secret produces a low-confidentiality ciphertext
- Noninterference: changes to the secret plaintexts should be unobservable
- not true in general: $\{1234\}_k = \{1234\}_k$
  $\{1234\}_k \neq \{5678\}_k$
- randomization helps: $\{1234, r\}_k \neq \{1234, r'\}_k$
  $\{1234, r\}_k \neq \{5678, r'\}_k$
- ISO0 uses (a part of) vdata: $\{1234, vdata\}_k$
  - for different users with the same PIN, vdata is like randomization $\{1234, vdata\}_k \neq \{1234, vdata'\}_k$
  - for one user the PIN is determined by vdata

The ISO0 PIN block format

- encrypting a secret produces a low-confidentiality ciphertext
- Noninterference: changes to the secret plaintexts should be unobservable
- not true in general: $\{1234\}_k = \{1234\}_k$
  $\{1234\}_k \neq \{5678\}_k$
- randomization helps: $\{1234, r\}_k \neq \{1234, r'\}_k$
  $\{1234, r\}_k \neq \{5678, r'\}_k$
- ISO0 uses (a part of) vdata: $\{1234, vdata\}_k$
  - for different users with the same PIN, vdata is like randomization $\{1234, vdata\}_k \neq \{1234, vdata'\}_k$
  - for one user the PIN is determined by vdata

$\Rightarrow$ noninterference holds

Summary: fixing PIN management APIs

- robustness in a non-randomized cryptographic setting: existing PIN processing APIs are not robust
- a MAC-based fix of PIN_V (and PIN_T in the paper)
  - low-impact CVV-based fix [Focardi, Luccio, Steel, NORDSEC'09]
- integrity w.r.t. a representative, e.g., dectab : [DTAB↩VDATA]
- a type system to type-check APIs

Theorem

1. $\Gamma \vdash P$ *then* P *is robust*
2. *if* P *does not declassify data, then* P *satisfies noninterference, too*

- More detail in [Centenaro, Focardi, Luccio, Steel, ESORICS'09]

Conlusion

- ✓ API-level attacks to guess bank PINs
(much more can be found in [Bond, Zielinski '03, Clulow '03])
- ✓ How to become rich by playing Mastermind:
almost-optimal strategies to break PINs [Focardi, Luccio, FUN'10]
- ✓ Language-based analysis and fixes
[Centenaro, Focardi, Luccio, Steel, ESORICS'09,
Focardi, Luccio, Steel, NORDSEC'09]

Conlusion

- ✓ API-level attacks to guess bank PINs
  (much more can be found in [Bond, Zielinski '03, Clulow '03])
- ✓ How to become rich by playing Mastermind:
  almost-optimal strategies to break PINs [Focardi, Luccio, FUN'10]
- ✓ Language-based analysis and fixes
  [Centenaro, Focardi, Luccio, Steel, ESORICS'09,
  Focardi, Luccio, Steel, NORDSEC'09]

- Tomorrow we will see why some smarcard and crypto-tokens can be easily cloned

References

M. Bond and P. Zielinski.
Decimalization table attacks for PIN cracking.
UCAM-CL-TR-560, Univ. Cambridge, Computer Lab., 2003.

M. Centenaro, R. Focardi, F.L. Luccio, G. Steel.
Type-Based Analysis of PIN Processing APIs
In proceedings of ESORICS'09, September 2009.

J. Clulow.
The design and analysis of cryptographic APIs for security devices.
Master's thesis, University of Natal, Durban, 2003.

R. Focardi, F.L. Luccio,
Cracking bank pins by playing mastermind.
In Proceedings of FUN'10, June 2010.

R. Focardi, F.L. Luccio, G. Steel.
Blunting Differential Attacks on PIN Processing APIs
In proceedings of NORDSEC'09, Obtober 2009.

References

R. Focardi, F.L. Luccio.
Secure upgrade of hardware security modules in bank networks
ARSPA-WITS10 Paphos, Cyprus March 27-28, 2010.

IBM Inc.
CCA Basic Services Reference and Guide for the IBM 4758 PCI
Technical report, 2006. Rel. 2.53–3.27.

A. Myers, A. Sabelfeld, and S. Zdancewic.
Enforcing robust declassification and qualified robustness.
Journal of Computer Security, 14(2):157–196, May 2006.

G. Steel.
Formal Analysis of PIN Block Attacks
Theoretical Computer Science, 367, 1-2, pages 257-270, November 2006.

D. Knuth,
The Computer as a Master Mind,
Journal of Recreational Mathematics, 9, pages 1-6, 1976.