Secure upgrade of hardware security modules in bank networks*

Riccardo Focardi[1] Flaminia Luccio[1]

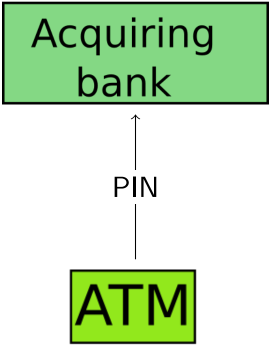
[1]Università Ca' Foscari di Venezia, Italy
{focardi,luccio}@dsi.unive.it

ARSPA-WITS'10
Paphos, Cyprus March 27-28, 2010

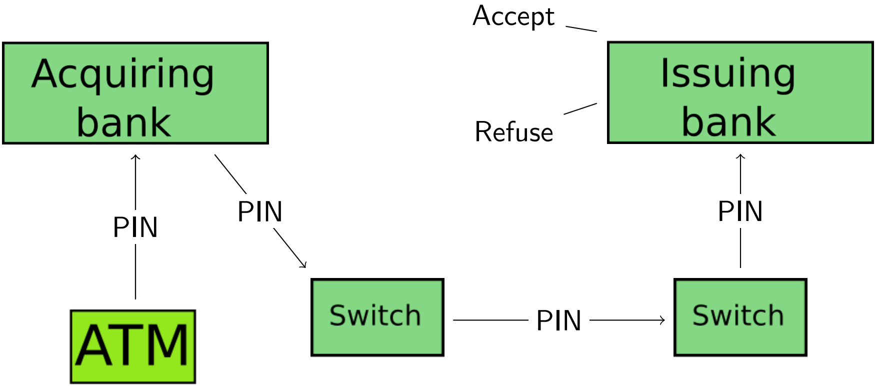
*Work partially supported by:
Miur'07 Project SOFT: "*Security Oriented Formal Techniques*"

PIN processing infrastructure

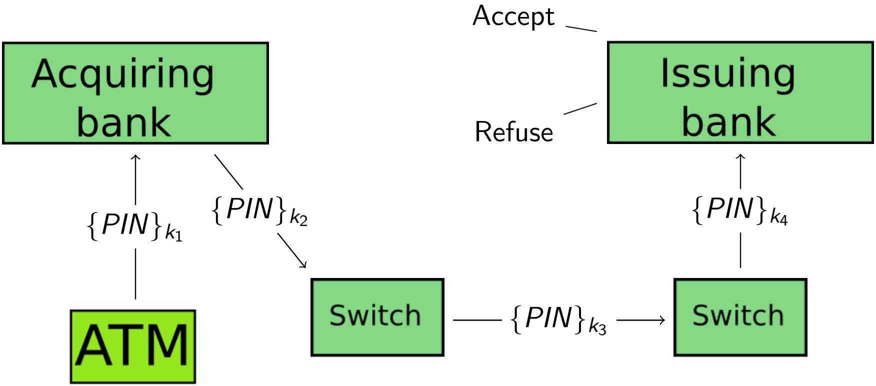
PIN processing infrastructure



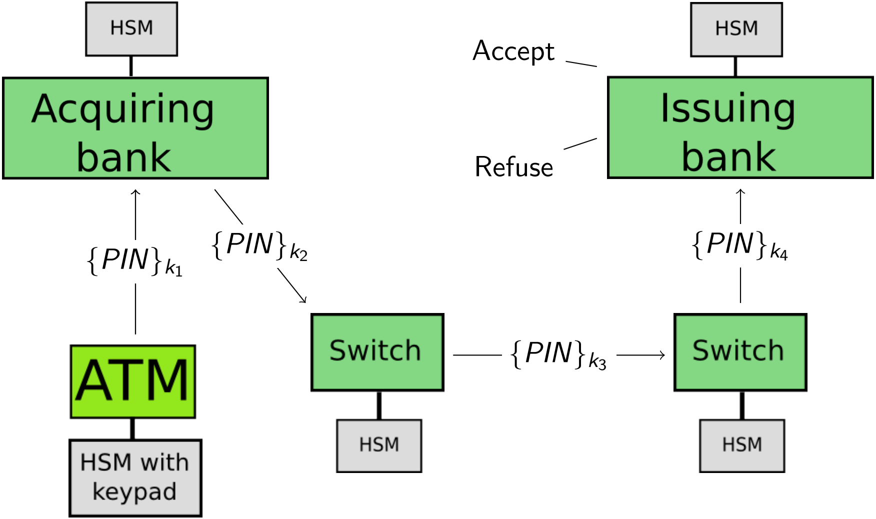
PIN processing infrastructure



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

Hardware Security Module (HSM)

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

... but still, attacks are possible

Hardware Security Module (HSM)

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

... but still, attacks are possible

Our goal:

propose 'cheap' HSM upgrading strategies

1. securing subnetworks while keeping service up
2. trade-off between hardware and manpower cost

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

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

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

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. $dec_k(\{4104, r\}_k) = \quad 4104, r$
   4104
2. $enc_{pdk}(\text{vdata}) = \quad A47295FDE32A48B1$
   $0472 \oplus 4732 \bmod 10 = 4104$
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. $dec_k(\{4104, r\}_k) = 4104, r$
   4104
2. $enc_{pdk}(vdata) = A47295FDE32A48B1$
   $0472 \oplus 4732 \bmod 10 = 4104$
3. PIN_V returns ‘true’

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

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

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

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

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

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

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

   $1472 \oplus 4732 \bmod 10 = 5104$

3. PIN_V returns 'false'

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

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

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

This kind of attack is practical

- an average of 13.463 PIN_V calls for a four-digit PIN [Focardi, Luccio, FUN'10]
- ... an insider might disclose thousands of PINs in a lunch-break!

Verizon Breach Report 2008

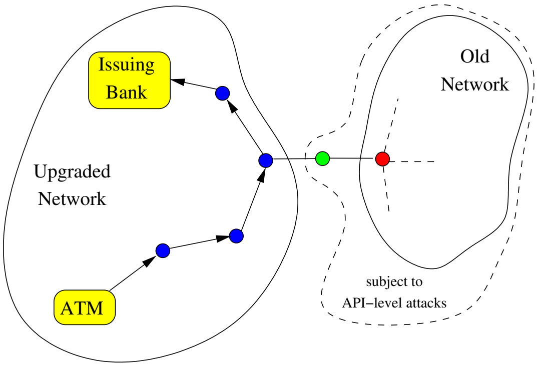
"Were seeing entirely new attacks that a year ago were thought to be only academically possible"
"What we see now is people going right to the source [..] and stealing the encrypted PIN blocks and using complex ways to un-encrypt the PIN blocks."
(Quotes from Wired Magazine interview with report author,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

How 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

HSM upgrade

- replace old, flawed, functionalities with new, patched, APIs
- keep the service up: new and old HSMs should 'talk'
- IDEA: special *borderline* HSMs placed temporarily
  - supporting both old and new APIs (still flawed!)
  - translating from/to upgraded and non-upgraded subnetworks



The HSM upgrading problem

- initially *non-upgraded* tree network
- *U technicians* moving on the network and upgrading nodes
- technicians place borderline HSMs, when needed
- borderline HSMs can be moved when all the neighbouring nodes are upgraded

HSM upgrading strategy

A sequence of moves that upgrades an initially non-upgraded network

HSM upgrading number $uhn(T, U)$

The number of borderline HSMs needed to solve the HSM problem on a given tree T and with a given number U of technicians

The Connected Monotone Decontamination problem [Barrière et al., SPAA'02]

- initially *contaminated* tree network
- a set of *agents* moving on the network
- agents decontaminate nodes they traverse
- decontaminated nodes left unguarded are recontaminated

Decontamination strategy

A sequence of moves that clears an initially contaminated network

Connected search number $csn(T)$

The number of agents needed to solve the CMD problem on a given tree T

The two problems are strictly related

Theorem

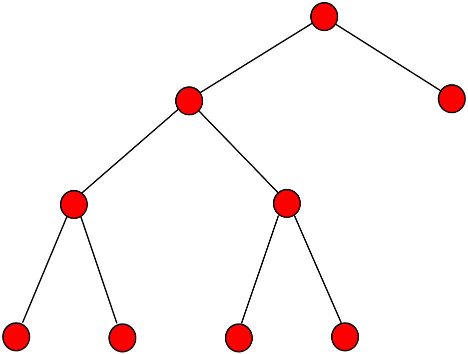
Given a tree T, we have $uhn(T, 1) \leq csn(T) \leq uhn(T, 1) + 1$

Intuitively:

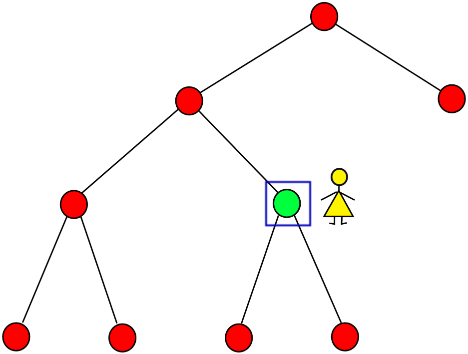
- Borderline HSMs as ‘still’ agents transported by the *unique* technician
- Agent moves simulated by the technician reaching a borderline HSM and moving it

reuse known algorithms and generalize them to U technicians

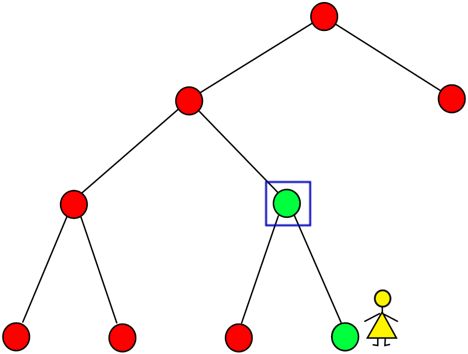
The algorithm with 1 technician



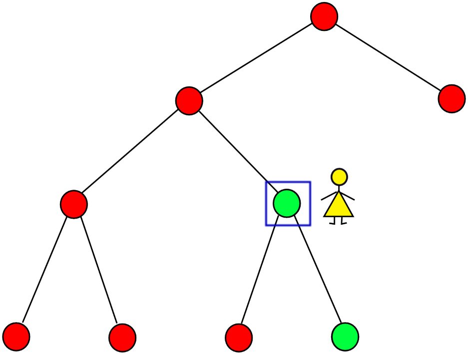
The algorithm with 1 technician



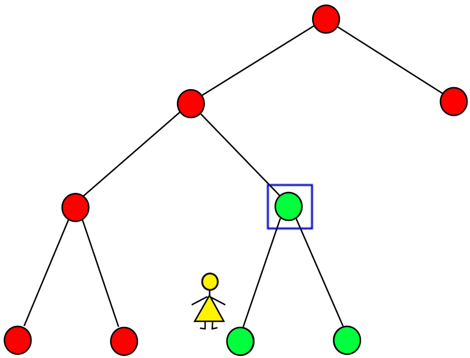
The algorithm with 1 technician



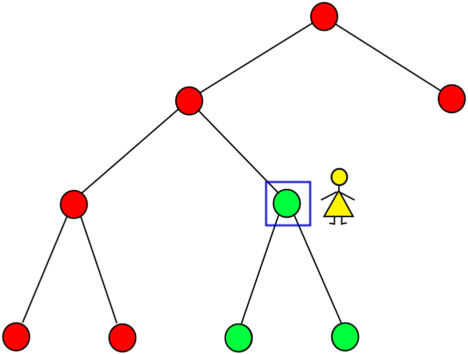
The algorithm with 1 technician



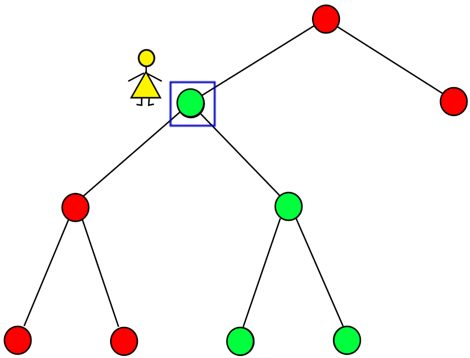
The algorithm with 1 technician



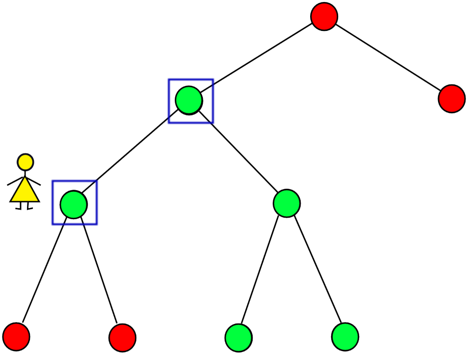
The algorithm with 1 technician



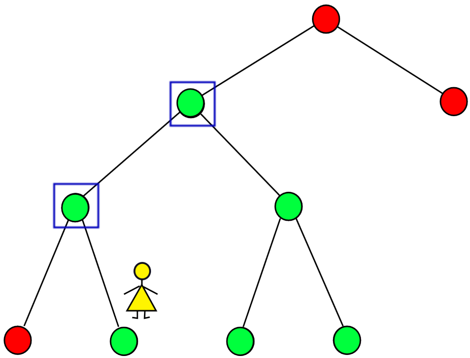
The algorithm with 1 technician



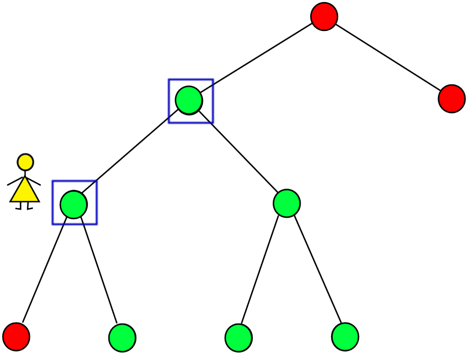
The algorithm with 1 technician



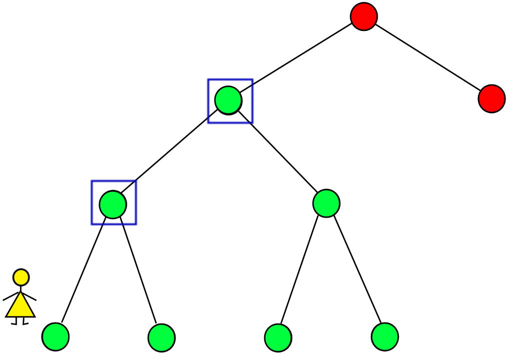
The algorithm with 1 technician



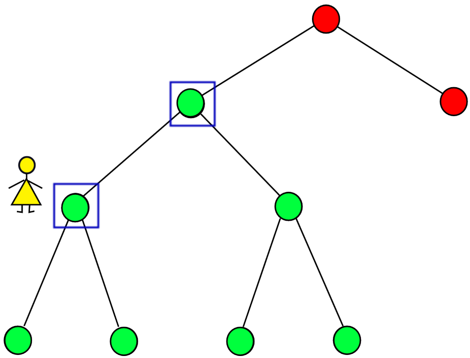
The algorithm with 1 technician



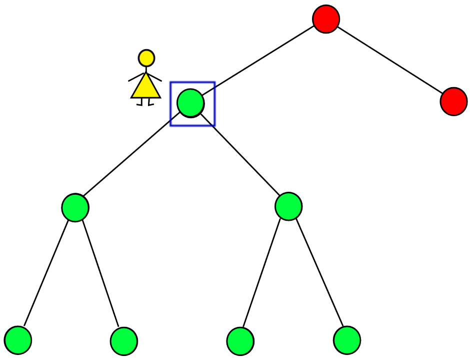
The algorithm with 1 technician



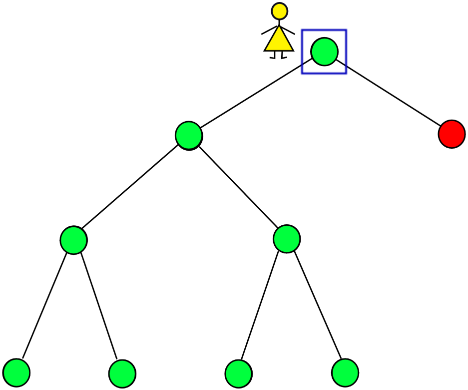
The algorithm with 1 technician



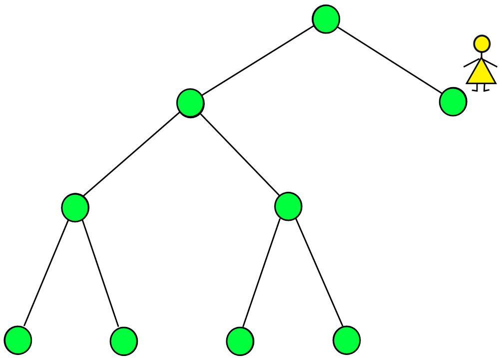
The algorithm with 1 technician



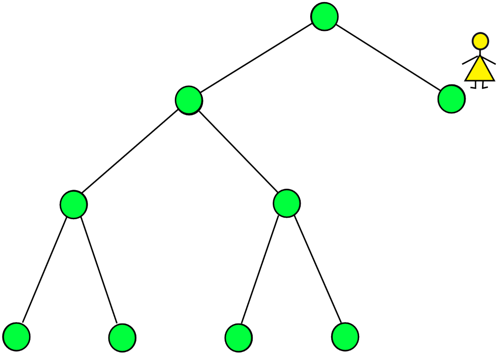
The algorithm with 1 technician



The algorithm with 1 technician

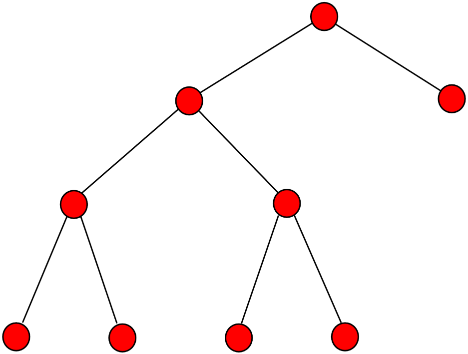


The algorithm with 1 technician

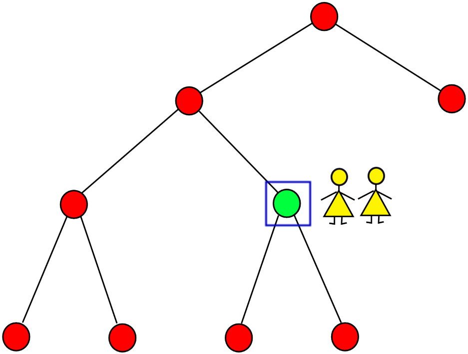


- Two borderline HSMs needed

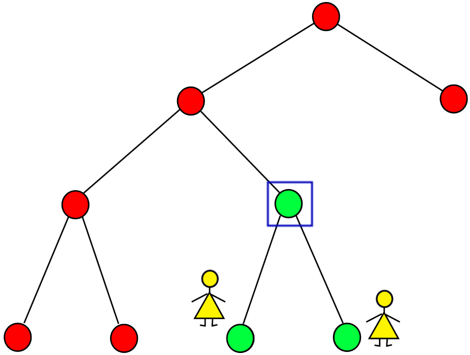
The algorithm with 2 technicians



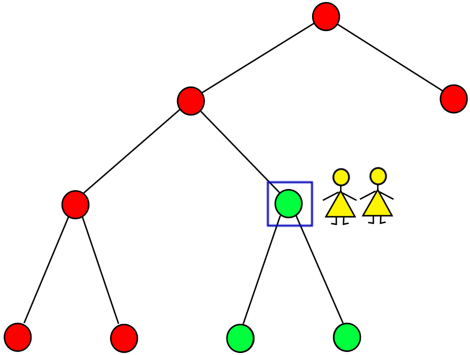
The algorithm with 2 technicians



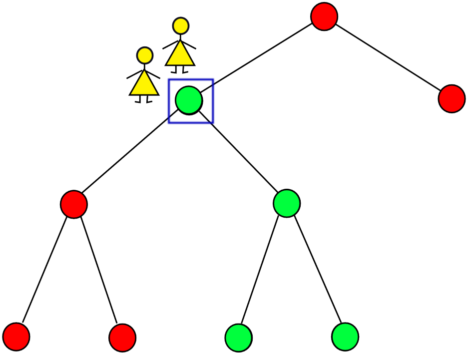
The algorithm with 2 technicians



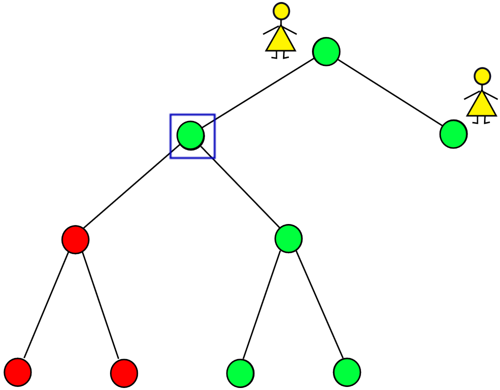
The algorithm with 2 technicians



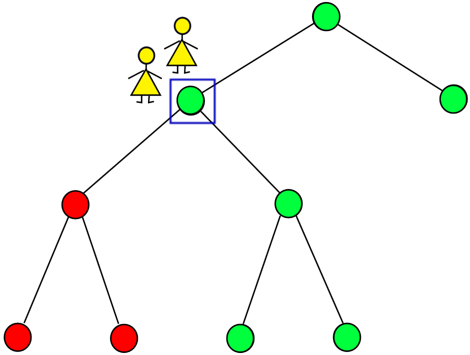
The algorithm with 2 technicians



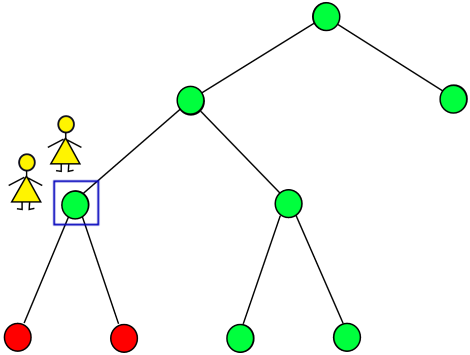
The algorithm with 2 technicians



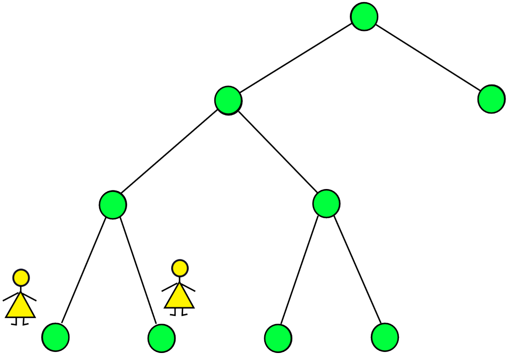
The algorithm with 2 technicians



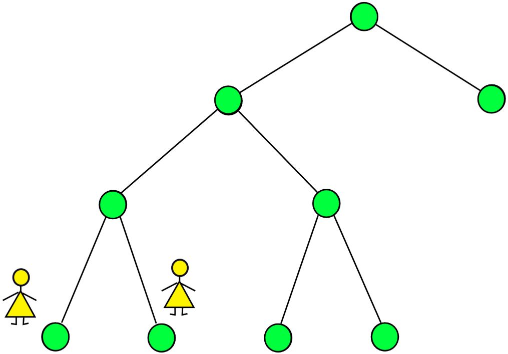
The algorithm with 2 technicians



The algorithm with 2 technicians



The algorithm with 2 technicians



- Only one borderline HSM needed!

Cost trade-off: an example

- Let C_H be the cost for one HSM and C_U the cost for one technician
- $2C_H + C_U$ versus $C_H + 2C_U$
- Suppose $C_H = 10000€$ and $C_U = 5000€$ we obtain
  - 25000€ versus 20000€

- In general, $BC_H + UC_U$ where B is derived by applying the strategy

Conclusion

- strategy for HSM upgrading on tree networks
- trade-off between hardware and manpower cost

Open problems

- placing HSMs on edges instead of nodes
- trade-off between cost and security
    - counting the number of secured paths
- measuring the travelling cost
    - weighted graph
    - independent distance matrix
- extensions to more topologies

References

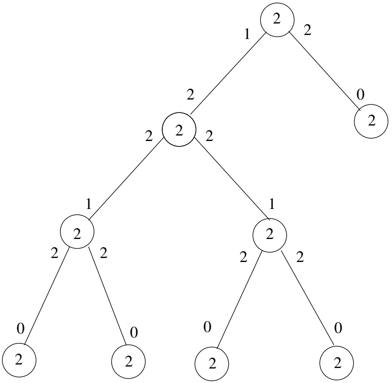
L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro.
Capture of an intruder by mobile agents.
In proceedings of SPAA'02.

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

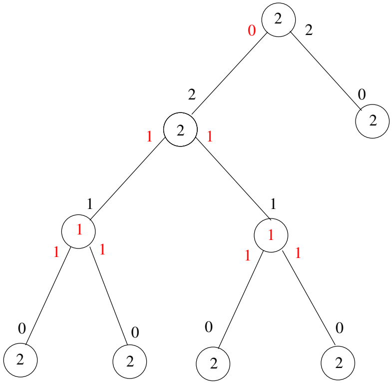
M. Centenaro, R. Focardi, F.L. Luccio, G. Steel.
Type-Based Analysis of PIN Processing APIs
In proceedings of ESORICS'09, September 2009.

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

R. Focardi, F.L. Luccio.
Cracking bank PINs by playing Mastermind
to appear in FUN'10, June 2010, Ischia Island.


2
1 2
2 0
2 2 2 2
1 1
2 2
2 2 2 2
0 0 0 0
2 2 2 2

U=1


2
0 2
2 0
1 2 1 2
1 1
1 1
1 1 1 1
0 0 0 0
2 2 2 2

U=2