Cyber bank robbery

Erik Poll

Digital Security group

Radboud University, Nijmegen, the Netherlands

Biggest cyber bank robbery to date

$ 951 million stolen via SWIFT global payment system from the Bangladesh Central Bank

- Most of the money recuperated
- ‘Only’ $ 81 million really lost, via casinos on the Philippines
- Attackers installed custom malware on computers at bank & clearly had insider knowledge
  - malware removed transactions from local database & physical print outs

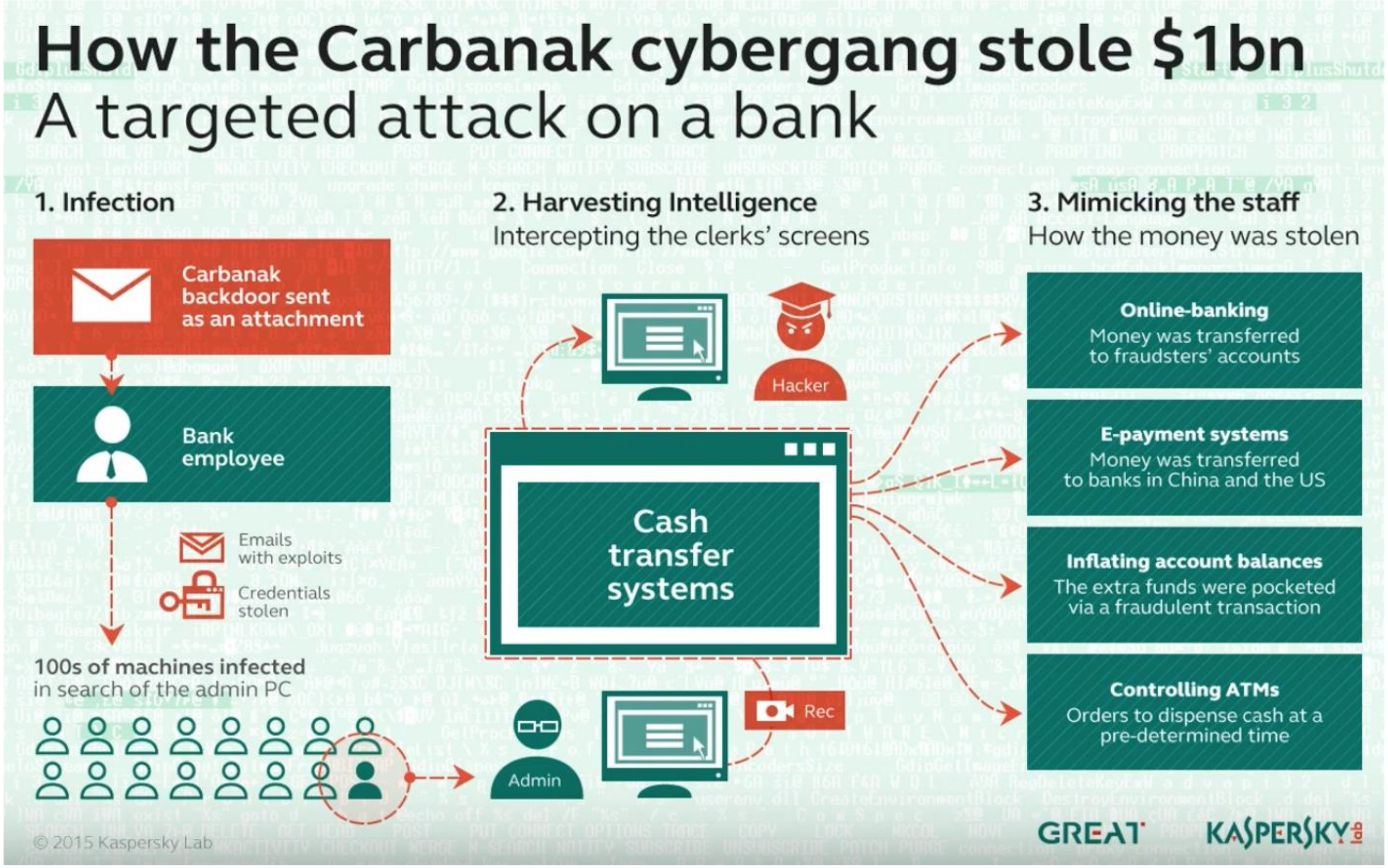
These are no script kiddies, but serious organised crime

[http://baesystemsai.blogspot.com/2016/04/two-bytes-to-951m.html]

[http://www.reuters.com/assets/iframe/cmsyovideo?videoId=370707923]

[https://www.nettitude.com/wp-content/uploads/2016/12/Nettitude-SWIFT-Threat-Advisory-Report-client.pdf]

Carbanak hack



How the Carbanak cybergang stole $1bn

A targeted attack on a bank

1. Infection

Carbanak backdoor sent as an attachment

Bank employee

Emails with exploits

Credentials stolen

100s of machines infected in search of the admin PC

2. Harvesting Intelligence

Intercepting the clerks' screens

Hacker

Cash transfer systems

Admin

Rec

3. Mimicking the staff

How the money was stolen

Online-banking
Money was transferred to fraudsters' accounts

E-payment systems
Money was transferred to banks in China and the US

Inflating account balances
The extra funds were pocketed via a fraudulent transaction

Controlling ATMs
Orders to dispense cash at a pre-determined time

© 2015 Kaspersky Lab

GREAT KASPERSKY lab

Darknet Diaries podcast https://darknetdiaries.com/episode/35

DARKNET DIARIES

Overview

1. Skimming

2. EMV and complexities of EMV

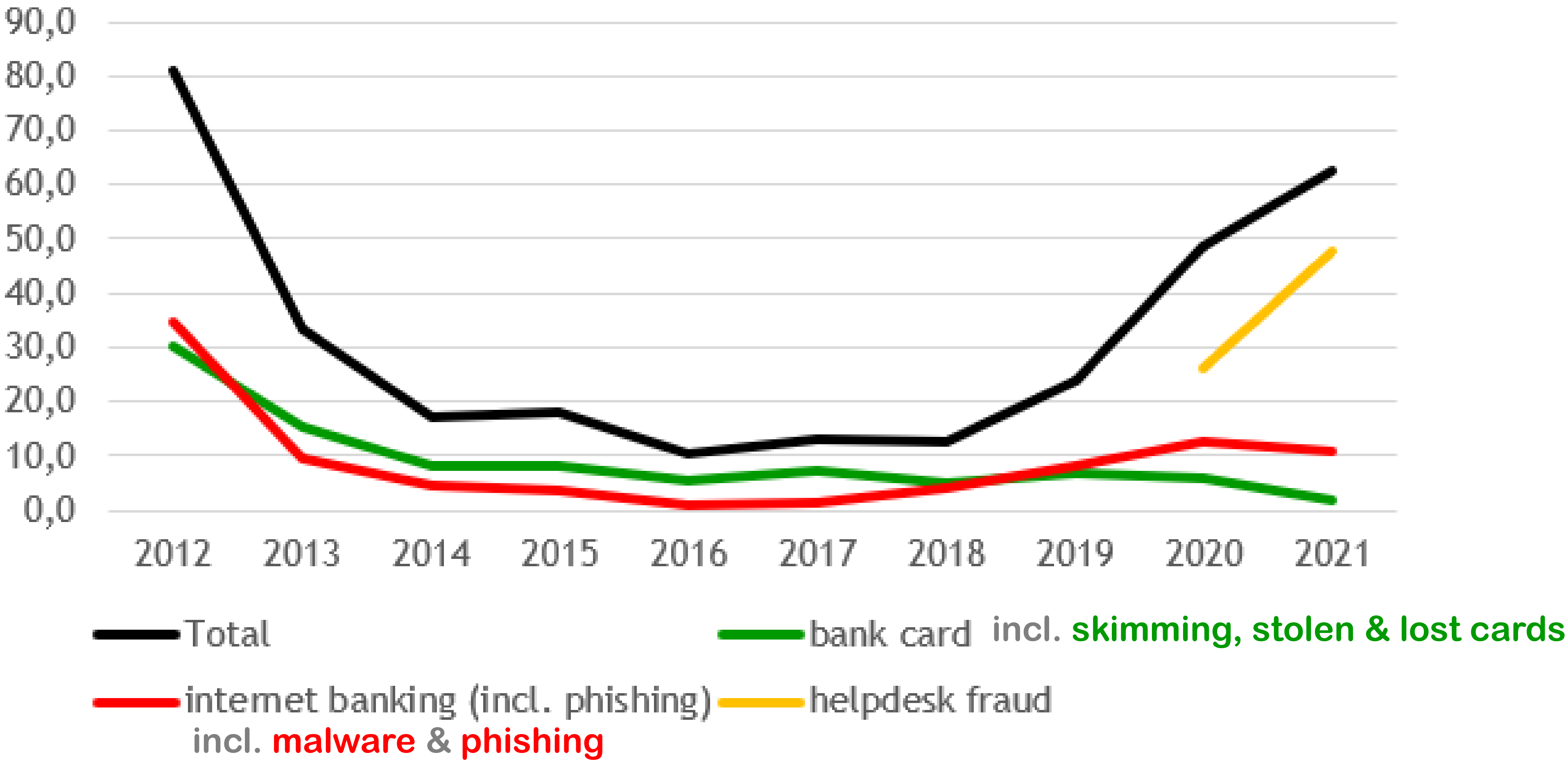
3. Online banking

4. Contactless payments

Things that go wrong:
Complexity, backward compatibility, UI as weakest link

Techniques to combat this (a little bit):
formal specification using finite state machines, fuzzing

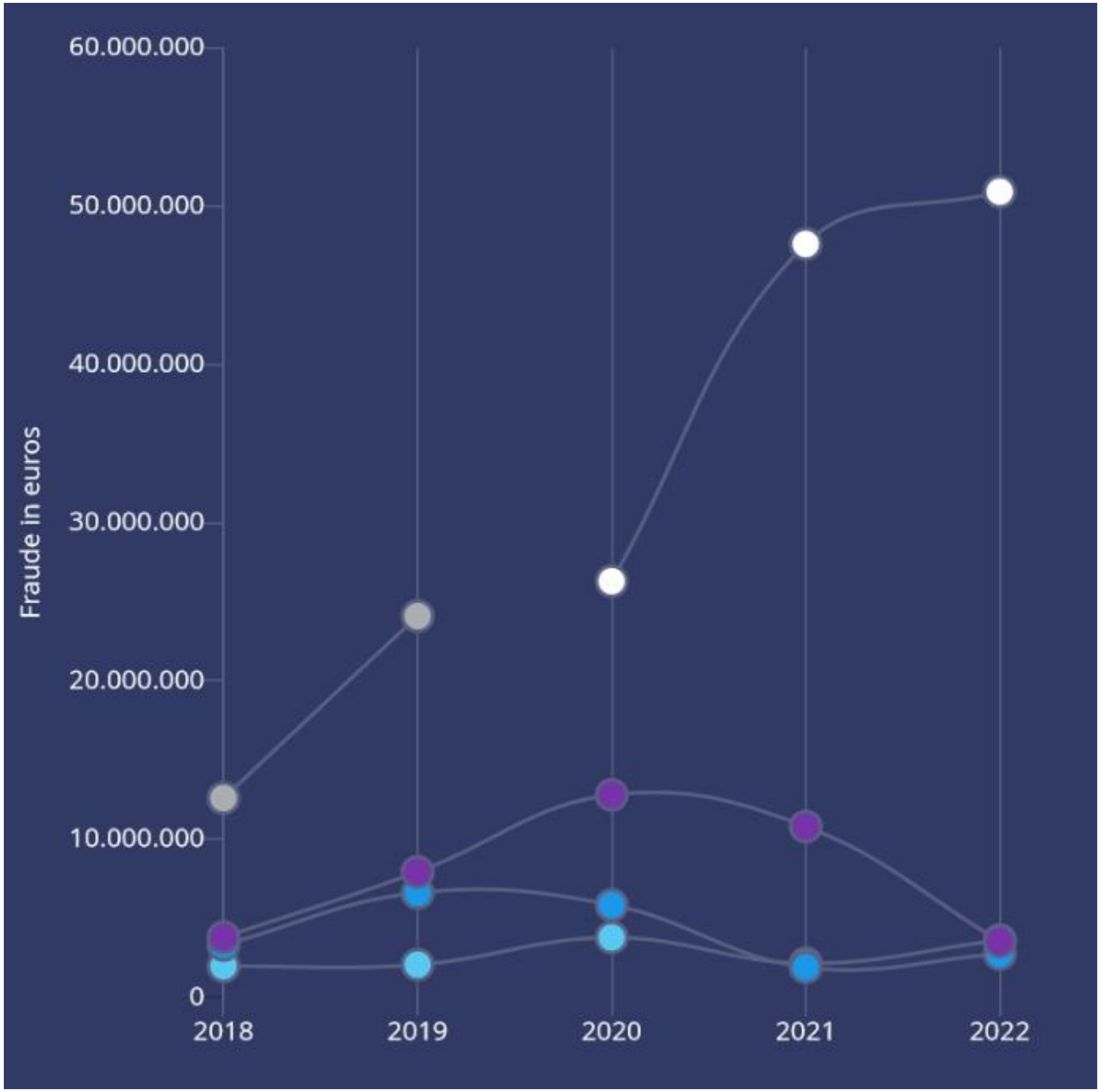
Payment fraud in Netherlands – longer term trends

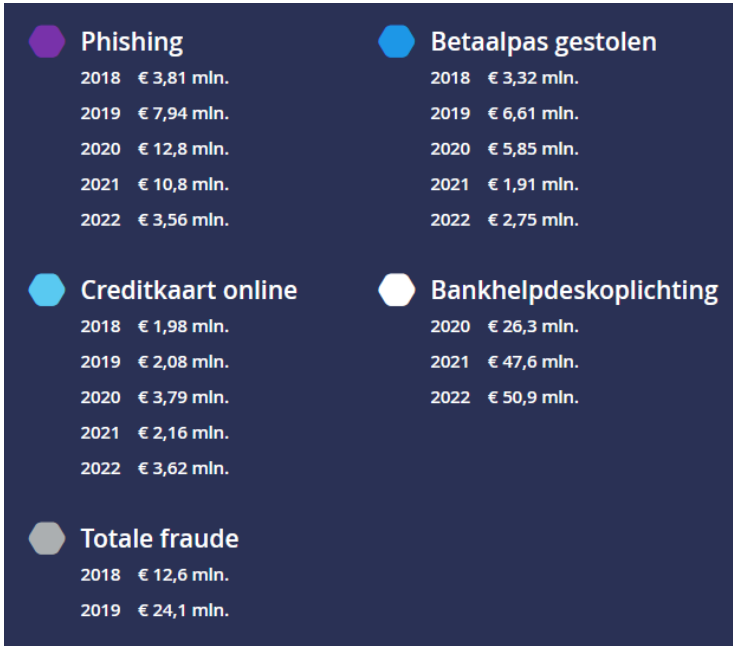


incl. **skimming, stolen & lost cards**

incl. **malware** & **phishing**

Payment fraud in the Netherlands – recent years





🟣 Phishing	2018	€ 3,81 mln.
	2019	€ 7,94 mln.
	2020	€ 12,8 mln.
	2021	€ 10,8 mln.
	2022	€ 3,56 mln.
🔵 Betaalpas gestolen	2018	€ 3,32 mln.
	2019	€ 6,61 mln.
	2020	€ 5,85 mln.
	2021	€ 1,91 mln.
	2022	€ 2,75 mln.
🔵 Creditkaart online	2018	€ 1,98 mln.
	2019	€ 2,08 mln.
	2020	€ 3,79 mln.
	2021	€ 2,16 mln.
	2022	€ 3,62 mln.
⚪ Bankhelpdeskoplichting	2020	€ 26,3 mln.
	2021	€ 47,6 mln.
	2022	€ 50,9 mln.
⚪ Totale fraude	2018	€ 12,6 mln.
	2019	€ 24,1 mln.

https://factsheet.betaalvereniging.nl

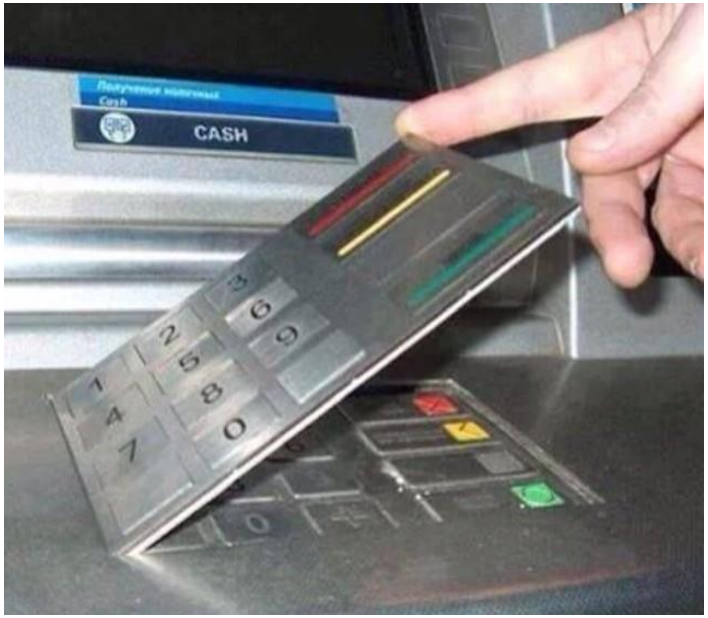
Skimming

Skimming

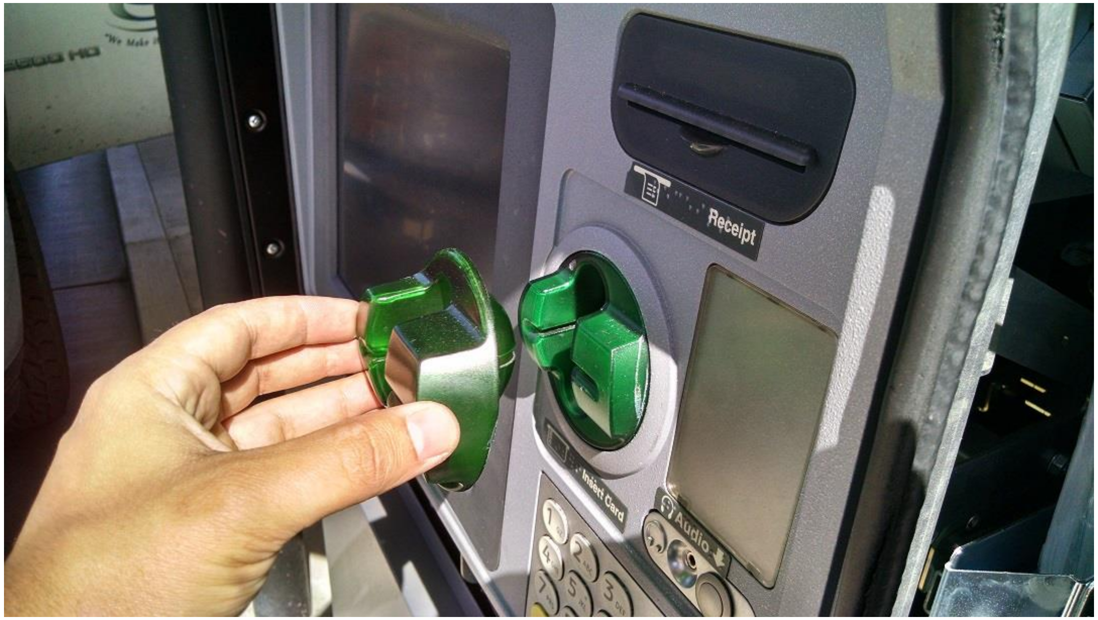
Magnetic-stripe (mag-stripe) on bank card contains digitally signed information

but… this info can be copied

Skimming

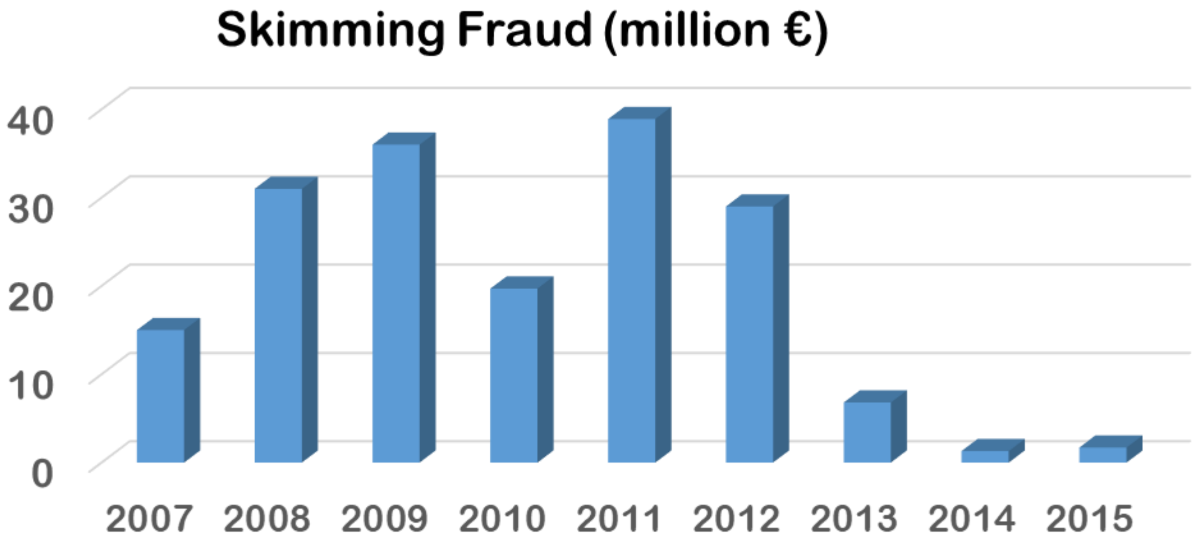


Fake keyboard
to intercept PIN code



Fake cover
that copies magnetic stripe

Skimming fraud in the Netherlands



[Source: NVB & Betaalvereniging]

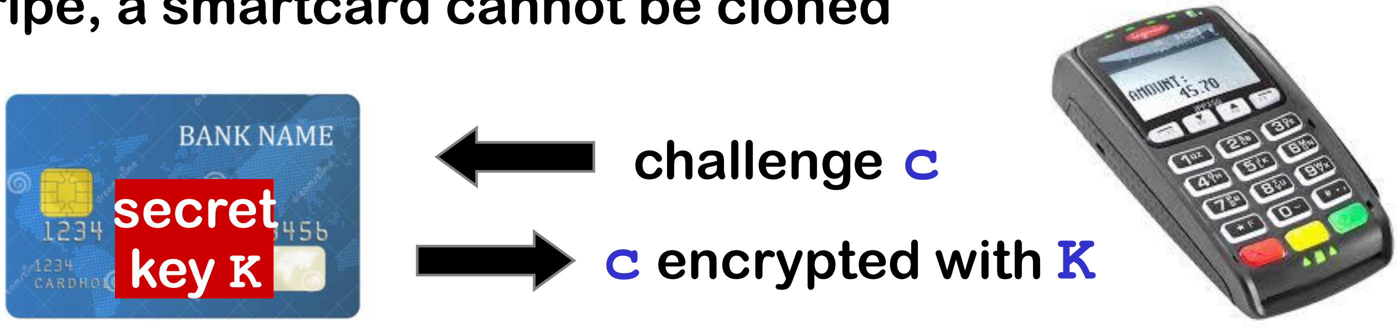
Fraud under control thanks to

- better monitoring & response (incl. blocking cards)
- replacing of mag-stripe by chip in 2012

EMV (Europay-Mastercard-Visa)

- Standard used by all chip cards for banking
- Specs controlled by EMVCo which is owned by MasterCard VISA AMERICAN EXPRESS JCB

- Unlike magstripe, a smartcard cannot be cloned



secret key K

challenge c

c encrypted with K

  - Payment terminal sends a different challenge c every time, so card gives a different response each time
  - Card proves it knows the secret key K without revealing it

Does EMV chip reduce skimming?

- UK introduced EMV in 2006

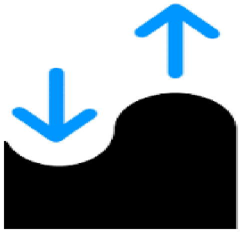
	2005	2006	2007	2008
domestic	79	46	31	36
foreign	18	53	113	134

Skimming fraud with UK cards, in millions £

Copied magstripes can still be used in countries that don’t use the chip

- Blocking cards for use outside EU (geoblocking) helps a lot!
- Skimmers have now moved to the US, and the US is now migrating to EMV

Such water bed effects are a recurring phenomenon

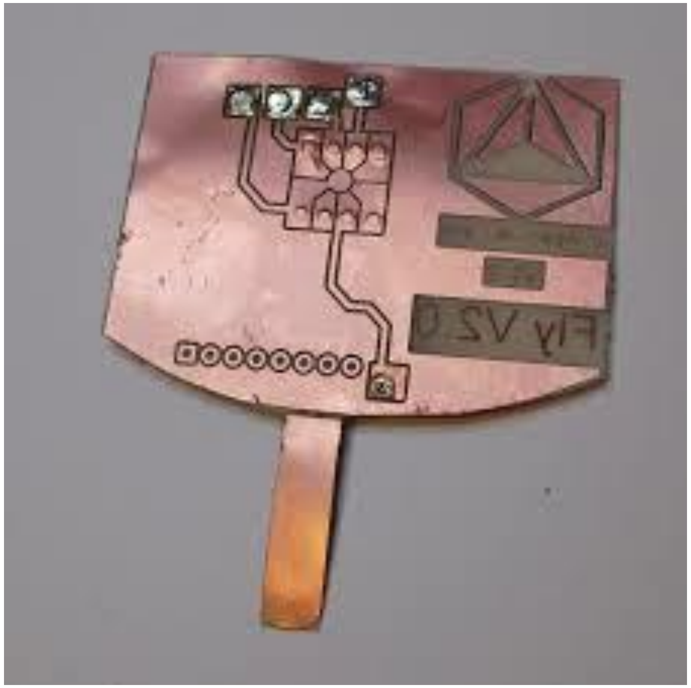


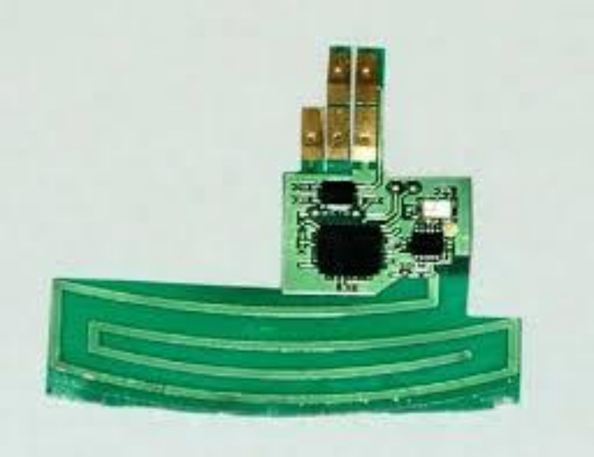
Recurring problem: BACKWARD COMPATIBILITY

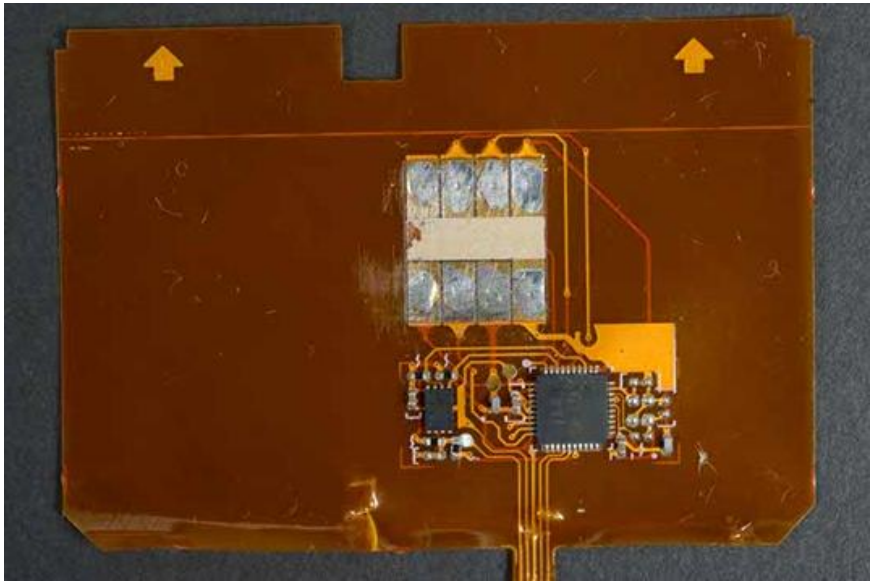
- In 2009, criminals put tampered card readers *inside* Dutch bank branches to skim cards
  - For *backwards compatibility*, the chip reports the mag-stripe data...
  - Both mag-stripe data and PIN code sent unencrypted from card to this reader
  - Criminals caught & convicted in 2011

- Cards have been improved to avoid this:
  mag-stripe data should now be different from info on the chip

Shims to eavesdrop on communication







https://krebsonsecurity.com/tag/atm-shimming/

Problem: COMPLEXITY

EMV is not a protocol, but a ‘protocol toolkit suite’ with *lots* of configuration options

- Original EMV specs : 4 books, > 700 pages
  - 3 types of cards (SDA,DDA, CDA), 5 authentication mechanism (online PIN, online PIN, offline encrypted PIN, signature, none), 2 types of transactions (offline, online), ...
- Contactless EMV: 7 books, > 2000 pages

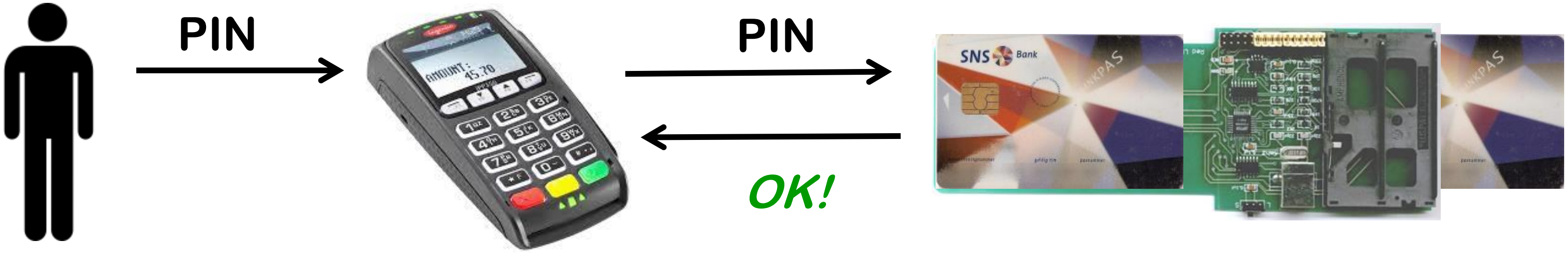
Sample sentence

> “If the card responds to GPO with SW1 SW2 = x9000 and AIP byte 2 bit 8 set to 0, and if the reader supports qVSDC and contactless VSDC, then if the Application Cryptogram (Tag '9F26') is present in the GPO response, then the reader shall process the transaction as qVSDC, and if Tag '9F26' is not present, then the reader shall process the transaction as VSDC.”

Complexity: example protocol flaw

Terminal can choose to do offline PIN

- **ie. terminal asks the card to check the PIN code**



PIN

PIN

OK!

The response of the card is *not authenticated*

(not cryptographically signed)

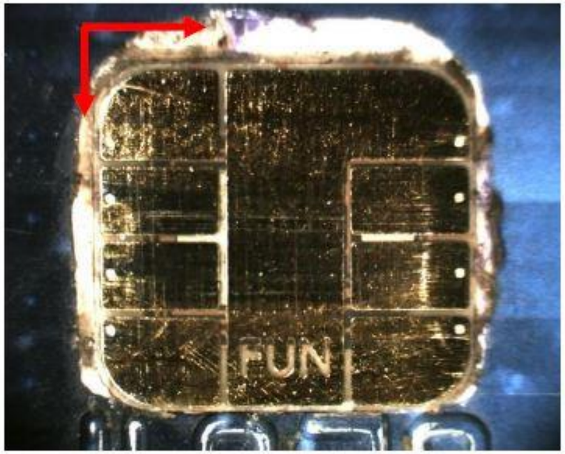
so terminal can be fooled by a Man-in-the-Middle attack

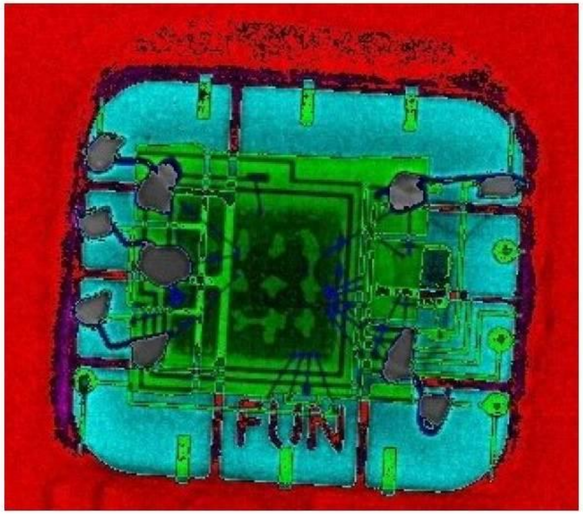
The transaction data will reveal the transaction was PIN-less, so the bank back-end will know the PIN was *not* entered

[Stephen Murdoch et al., *Chip & PIN is broken*, FC'2010]

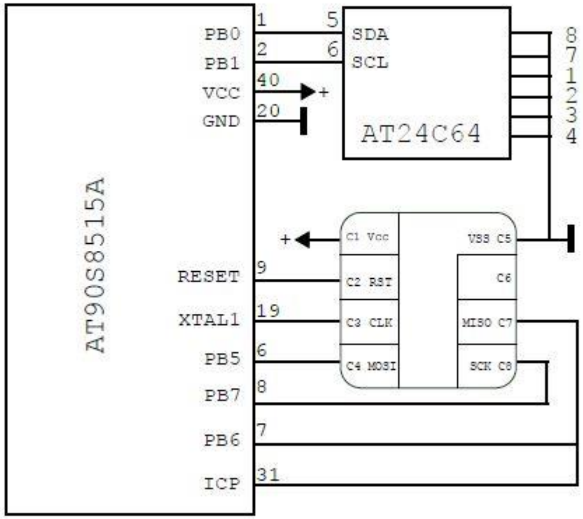
Criminal Man-in-the-Middle set-up

Chips from stolen cards inserted under another chip, which faked the PIN OK response





xray reveals
green stolen chip under
blue microcontroller



[Houda Ferradi et al., *When Organized Crime Applies Academic Results: A Forensic Analysis of an In-Card Listening Device*, Journal of Cryptographic Engineering, 2015]

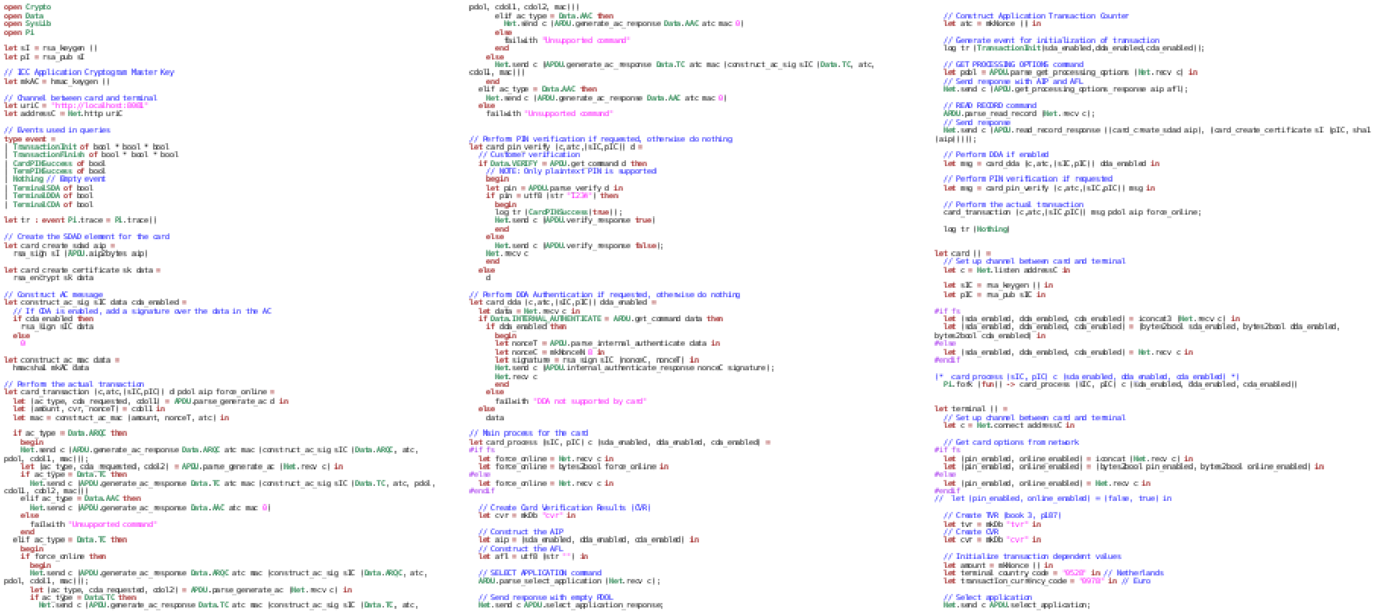
Complexity of EMV specs

- Specifications very complex to understand
  - long documents
  - no discussion of security goals or design choices
  - little abstraction or modularity

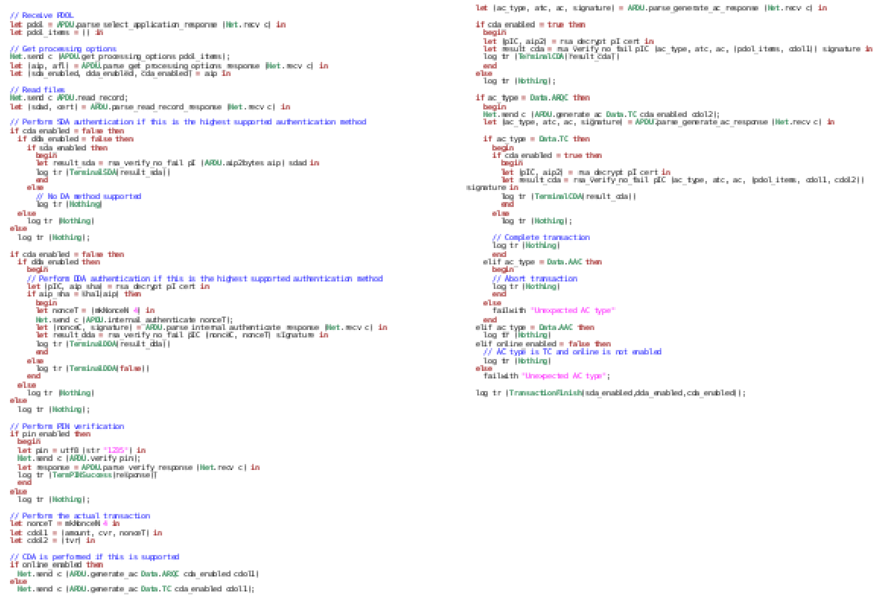
- Who really takes responsibility for ensuring these specs are secure? EMVCo, the credit card companies behind EMVCo, or individual banks?

- Can we provide some scientific rigour?

Formalisation of EMV in F#



```
open Crypto
open Data
open Syslib
open Pi

let sI = rsa_keygen ()
let pI = rsa_pub sI

// ICC Application Cryptogram Master Key
let mkAC = hmac_keygen ()

// Channel between card and terminal
let uriC = "http://localhost:8081"
let addressC = Net.http uriC

// Events used in queries
type event =
| TransactionInit of bool * bool * bool
| TransactionFinish of bool * bool * bool
| CardPINSuccess of bool
| TermPINSuccess of bool
| Nothing // Empty event
| TerminalSDA of bool
| TerminalDDA of bool
| TerminalCDA of bool

let tr : event Pi.trace = Pi.trace()

// Create the SDAD element for the card
let card_create_sdad aip =
  rsa_sign sI (APDU.aip2bytes aip)

let card_create_certificate sk data =
  rsa_encrypt sK data

// Construct AC message
let construct_ac_sig sIC data cda_enabled =
  // If CDA is enabled, add a signature over the data in the AC
  if cda_enabled then
    rsa_sign sIC data
  else
    0

let construct_ac_mac data =
  hmacsha1 mkAC data

// Perform the actual transaction
let card_transaction (c,atc,(sIC,pIC)) d pdol aip force_online =
  let (ac_type, cda_requested, cdol1) = APDU.parse_generate_ac d in
  let (amount, cvr, nonceT) = cdol1 in
  let mac = construct_ac_mac (amount, nonceT, atc) in

  if ac_type = Data.ARQC then
    begin
    Net.send c (APDU.generate_ac_response Data.ARQC atc mac (construct_ac_sig sIC (Data.ARQC, atc,
pdol, cdol1, mac)));
     let (ac_type, cda_requested, cdol2) = APDU.parse_generate_ac (Net.recv c) in
     if ac_type = Data.TC then
       Net.send c (APDU.generate_ac_response Data.TC atc mac (construct_ac_sig sIC (Data.TC, atc, pdol,
cdol1, cdol2, mac)))
     elif ac_type = Data.AAC then
       Net.send c (APDU.generate_ac_response Data.AAC atc mac 0)
     else
       failwith "Unsupported command"
     end
  elif ac_type = Data.TC then
    begin
    if force_online then
      begin
      Net.send c (APDU.generate_ac_response Data.ARQC atc mac (construct_ac_sig sIC (Data.ARQC, atc,
pdol, cdol1, mac)));
      let (ac_type, cda_requested, cdol2) = APDU.parse_generate_ac (Net.recv c) in
      if ac_type = Data.TC then
        Net.send c (APDU.generate_ac_response Data.TC atc mac (construct_ac_sig sIC (Data.TC, atc,
pdol, cdol1, cdol2, mac)))
        elif ac_type = Data.AAC then
          Net.send c (APDU.generate_ac_response Data.AAC atc mac 0)
        else
          failwith "Unsupported command"
        end
      else
        Net.send c (APDU.generate_ac_response Data.TC atc mac (construct_ac_sig sIC (Data.TC, atc,
cdol1, mac)))
      end
    elif ac_type = Data.AAC then
      Net.send c (APDU.generate_ac_response Data.AAC atc mac 0)
    else
      failwith "Unsupported command"


// Perform PIN verification if requested, otherwise do nothing
let card_pin_verify (c,atc,(sIC,pIC)) d =
  // Customer verification
  if Data.VERIFY = APDU.get_command d then
    // NOTE: Only plaintext PIN is supported
    begin
    let pin = APDU.parse_verify d in
    if pin = utf8 (str "1234") then
      begin
      log tr (CardPINSuccess(true));
      Net.send c (APDU.verify_response true)
      end
    else
      Net.send c (APDU.verify_response false);
    Net.recv c
    end
  else
    d

// Perform DDA Authentication if requested, otherwise do nothing
let card_dda (c,atc,(sIC,pIC)) dda_enabled =
  let data = Net.recv c in
  if Data.INTERNAL_AUTHENTICATE = APDU.get_command data then
    if dda_enabled then
      begin
      let nonceT = APDU.parse_internal_authenticate data in
      let nonceC = mkNonceN 8 in
      let signature = rsa_sign sIC (nonceC, nonceT) in
      Net.send c (APDU.internal_authenticate_response nonceC signature);
      Net.recv c
      end
    else
      failwith "DDA not supported by card"
  else
    data

// Main process for the card
let card_process (sIC, pIC) c (sda_enabled, dda_enabled, cda_enabled) =
#if fs
  let force_online = Net.recv c in
  let force_online = bytes2bool force_online in
#else
  let force_online = Net.recv c in
#endif

  // Create Card Verification Results (CVR)
  let cvr = mkDb "cvr" in

  // Construct the AIP
  let aip = (sda_enabled, dda_enabled, cda_enabled) in
  // Construct the AFL
  let afl = utf8 (str "") in

  // SELECT APPLICATION command
  APDU.parse_select_application (Net.recv c);

  // Send response with empty PDOL
  Net.send c APDU.select_application_response;

  // Construct Application Transaction Counter
  let atc = mkNonce () in

  // Generate event for initialization of transaction
  log tr (TransactionInit(sda_enabled,dda_enabled,cda_enabled));

  // GET PROCESSING OPTIONS command
  let pdol = APDU.parse_get_processing_options (Net.recv c) in
  // Send response with AIP and AFL
  Net.send c (APDU.get_processing_options_response aip afl);

  // READ RECORD command
  APDU.parse_read_record (Net.recv c);
  // Send response
  Net.send c (APDU.read_record_response ((card_create_sdad aip), (card_create_certificate sI (pIC, sha1
(aip)))));

  // Perform DDA if enabled
  let msg = card_dda (c,atc,(sIC,pIC)) dda_enabled in

  // Perform PIN verification if requested
  let msg = card_pin_verify (c,atc,(sIC,pIC)) msg in

  // Perform the actual transaction
  card_transaction (c,atc,(sIC,pIC)) msg pdol aip force_online;

  log tr (Nothing)


let card () =
  // Set up channel between card and terminal
  let c = Net.listen addressC in

  let sIC = rsa_keygen () in
  let pIC = rsa_pub sIC in

#if fs
  let (sda_enabled, dda_enabled, cda_enabled) = iconcat3 (Net.recv c) in
  let (sda_enabled, dda_enabled, cda_enabled) = (bytes2bool sda_enabled, bytes2bool dda_enabled,
bytes2bool cda_enabled) in
#else
  let (sda_enabled, dda_enabled, cda_enabled) = Net.recv c in
#endif

(*  card_process (sIC, pIC) c (sda_enabled, dda_enabled, cda_enabled) *)
  Pi.fork (fun() -> card_process (sIC, pIC) c (sda_enabled, dda_enabled, cda_enabled))


let terminal () =
  // Set up channel between card and terminal
  let c = Net.connect addressC in

  // Get card options from network
#if fs
  let (pin_enabled, online_enabled) = iconcat (Net.recv c) in
  let (pin_enabled, online_enabled) = (bytes2bool pin_enabled, bytes2bool online_enabled) in
#else
  let (pin_enabled, online_enabled) = Net.recv c in
#endif
//  let (pin_enabled, online_enabled) = (false, true) in

  // Create TVR (book 3, p187)
  let tvr = mkDb "tvr" in
  // Create CVR
  let cvr = mkDb "cvr" in

  // Initialize transaction dependent values
  let amount = mkNonce () in
  let terminal_country_code = "0528" in // Netherlands
  let transaction_currency_code = "0978" in // Euro

  // Select application
  Net.send c APDU.select_application;
```



```
  // Receive PDOL
  let pdol = APDU.parse_select_application_response (Net.recv c) in
  let pdol_items = () in

  // Get processing options
  Net.send c (APDU.get_processing_options pdol_items);
  let (aip, afl) = APDU.parse_get_processing_options_response (Net.recv c) in
  let (sda_enabled, dda_enabled, cda_enabled) = aip in

  // Read files
  Net.send c APDU.read_record;
  let (sdad, cert) = APDU.parse_read_record_response (Net.recv c) in

  // Perform SDA authentication if this is the highest supported authentication method
  if cda_enabled = false then
    if dda_enabled = false then
      if sda_enabled then
        begin
        let result_sda = rsa_verify_no_fail pI (APDU.aip2bytes aip) sdad in
        log tr (TerminalSDA(result_sda))
        end
      else
        // No DA method supported
        log tr (Nothing)
    else
      log tr (Nothing)
  else
    log tr (Nothing);

  if cda_enabled = false then
    if dda_enabled then
      begin
      // Perform DDA authentication if this is the highest supported authentication method
      let (pIC, aip_sha) = rsa_decrypt pI cert in
      if aip_sha = sha1(aip) then
        begin
        let nonceT = (mkNonceN 4) in
        Net.send c (APDU.internal_authenticate nonceT);
        let (nonceC, signature) = APDU.parse_internal_authenticate_response (Net.recv c) in
        let result_dda = rsa_verify_no_fail pIC (nonceC, nonceT) signature in
        log tr (TerminalDDA(result_dda))
        end
      else
        log tr (TerminalDDA(false))
      end
    else
      log tr (Nothing)
  else
    log tr (Nothing);

  // Perform PIN verification
  if pin_enabled then
    begin
    let pin = utf8 (str "1235") in
    Net.send c (APDU.verify pin);
    let response = APDU.parse_verify_response (Net.recv c) in
    log tr (TermPINSuccess(response))
    end
  else
    log tr (Nothing);

  // Perform the actual transaction
  let nonceT = mkNonceN 4 in
  let cdol1 = (amount, cvr, nonceT) in
  let cdol2 = (tvr) in

  // CDA is performed if this is supported
  if online_enabled then
    Net.send c (APDU.generate_ac Data.ARQC cda_enabled cdol1)
  else
    Net.send c (APDU.generate_ac Data.TC cda_enabled cdol1);

  let (ac_type, atc, ac, signature) = APDU.parse_generate_ac_response (Net.recv c) in

  if cda_enabled = true then
    begin
    let (pIC, aip2) = rsa_decrypt pI cert in
    let result_cda = rsa_Verify_no_fail pIC (ac_type, atc, ac, (pdol_items, cdol1)) signature in
    log tr (TerminalCDA(result_cda))
    end
  else
    log tr (Nothing);

  if ac_type = Data.ARQC then
    begin
    Net.send c (APDU.generate_ac Data.TC cda_enabled cdol2);
    let (ac_type, atc, ac, signature) = APDU.parse_generate_ac_response (Net.recv c) in

    if ac_type = Data.TC then
      begin
      if cda_enabled = true then
        begin
        let (pIC, aip2) = rsa_decrypt pI cert in
        let result_cda = rsa_Verify_no_fail pIC (ac_type, atc, ac, (pdol_items, cdol1, cdol2))
signature in
        log tr (TerminalCDA(result_cda))
        end
      else
        log tr (Nothing);

      // Complete transaction
      log tr (Nothing)
      end
    elif ac_type = Data.AAC then
      begin
      // Abort transaction
      log tr (Nothing)
      end
    else
      failwith "Unexpected AC type"
    end
  elif ac_type = Data.AAC then
    log tr (Nothing)
  elif online_enabled = false then
    // AC type is TC and online is not enabled
    log tr (Nothing)
  else
    failwith "Unexpected AC type";

  log tr (TransactionFinish(sda_enabled,dda_enabled,cda_enabled));
```

Essence of EMV in functional programming language F#

Formal Analysis of EMV

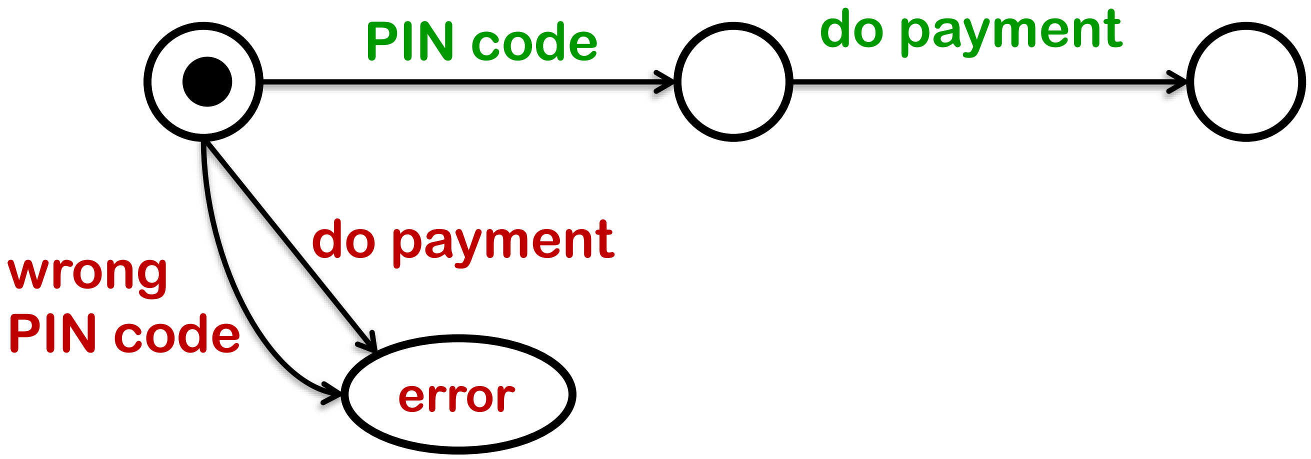
- Essence of EMV (all variants) can be formalized in less than 700 lines of F# code
- This model be analysed for security flaws using ProVerif tool
- No new attacks found, but existing attacks inevitably (re)discovered

[Joeri de Ruiter and Erik Poll, *Formal Analysis of the EMV protocol suite*, TOSCA 2012]

This still leaves the question if the software implementing these standards is correct!

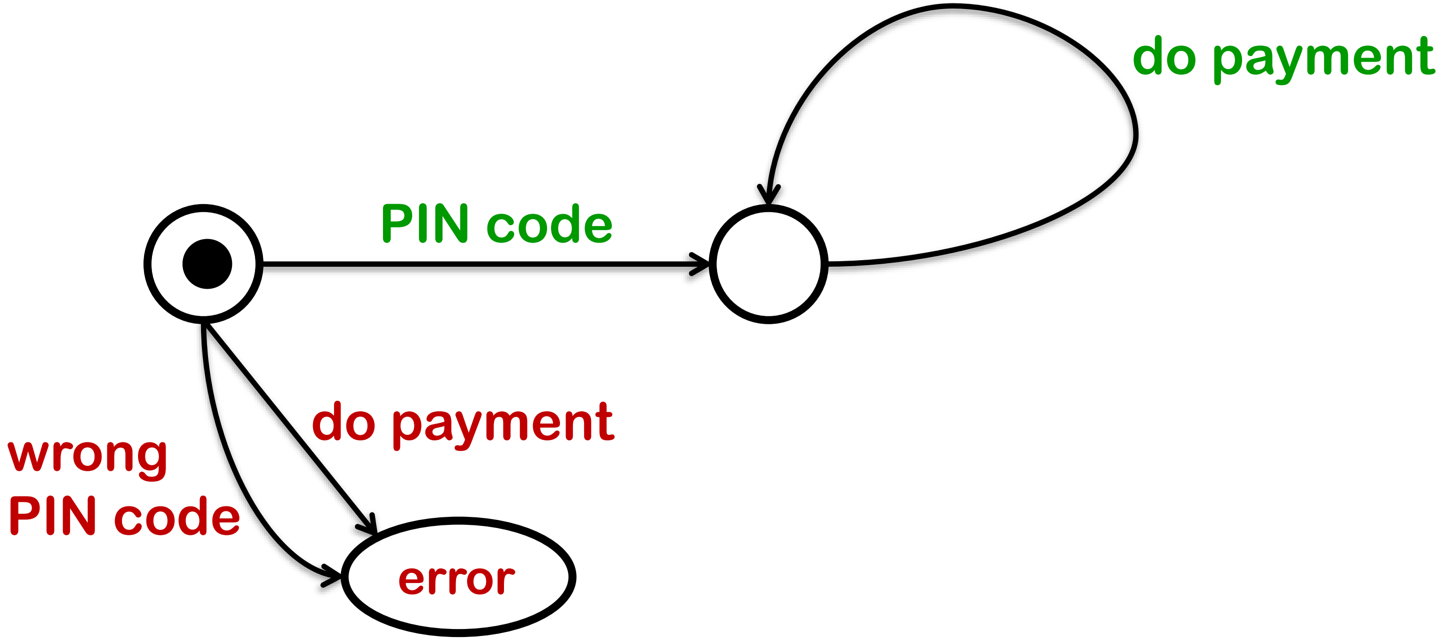
Finite State Machines

A bank card, like any program implementing a protocol, implements a finite state machine

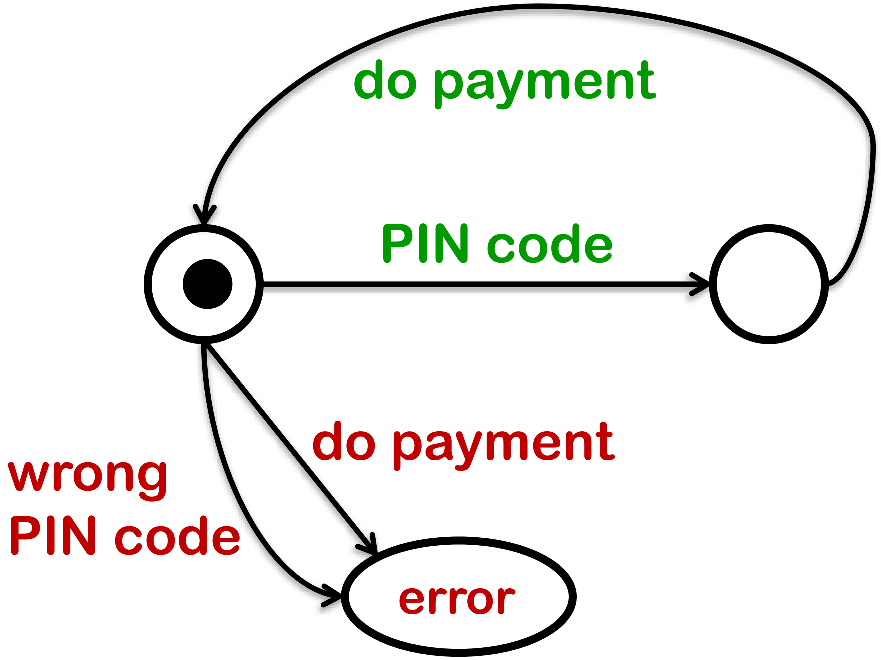


Such a state machine specifies (dis)allowed sequence of actions

Finite State Machines



Finite State Machines

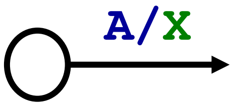


Finite state machines is a great formalism!
Easy to understand & precisely
captures subtle differences

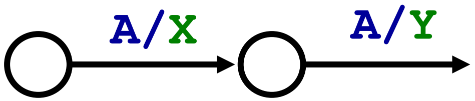
Automated inference of state machines

Just try out many sequences of inputs, and observe outputs

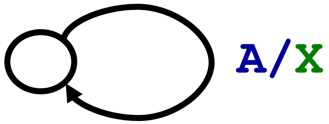
Suppose input A results in output X



- If second input A results in *different* output Y

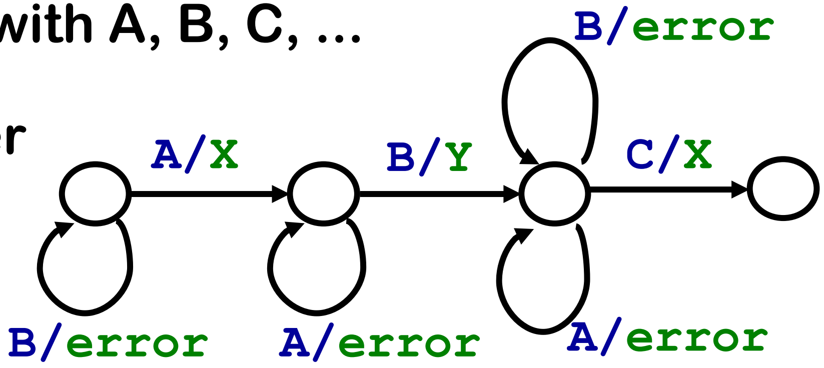


- If second input A results in the *same* output X



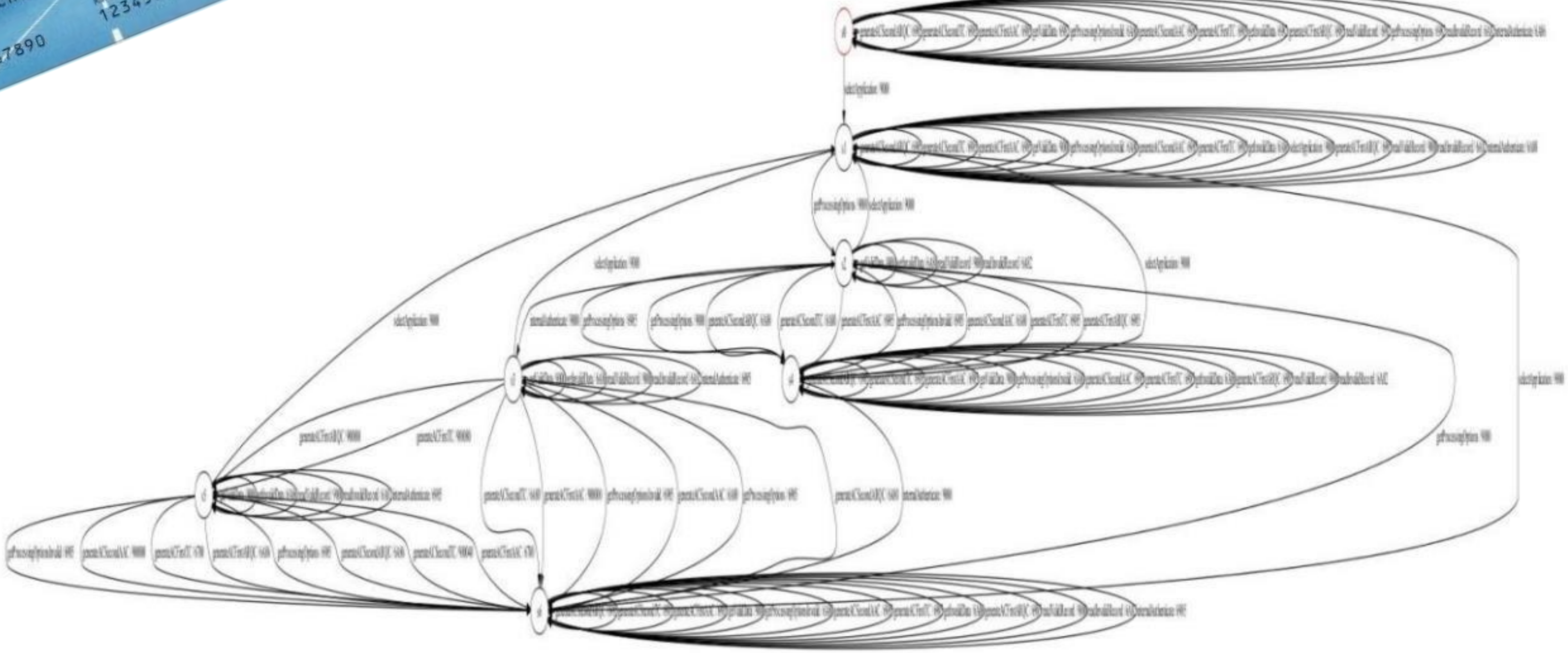
Now try more sequences of inputs with A, B, C, ...

to e.g. infer



The inferred state machine is under-approximation of real system

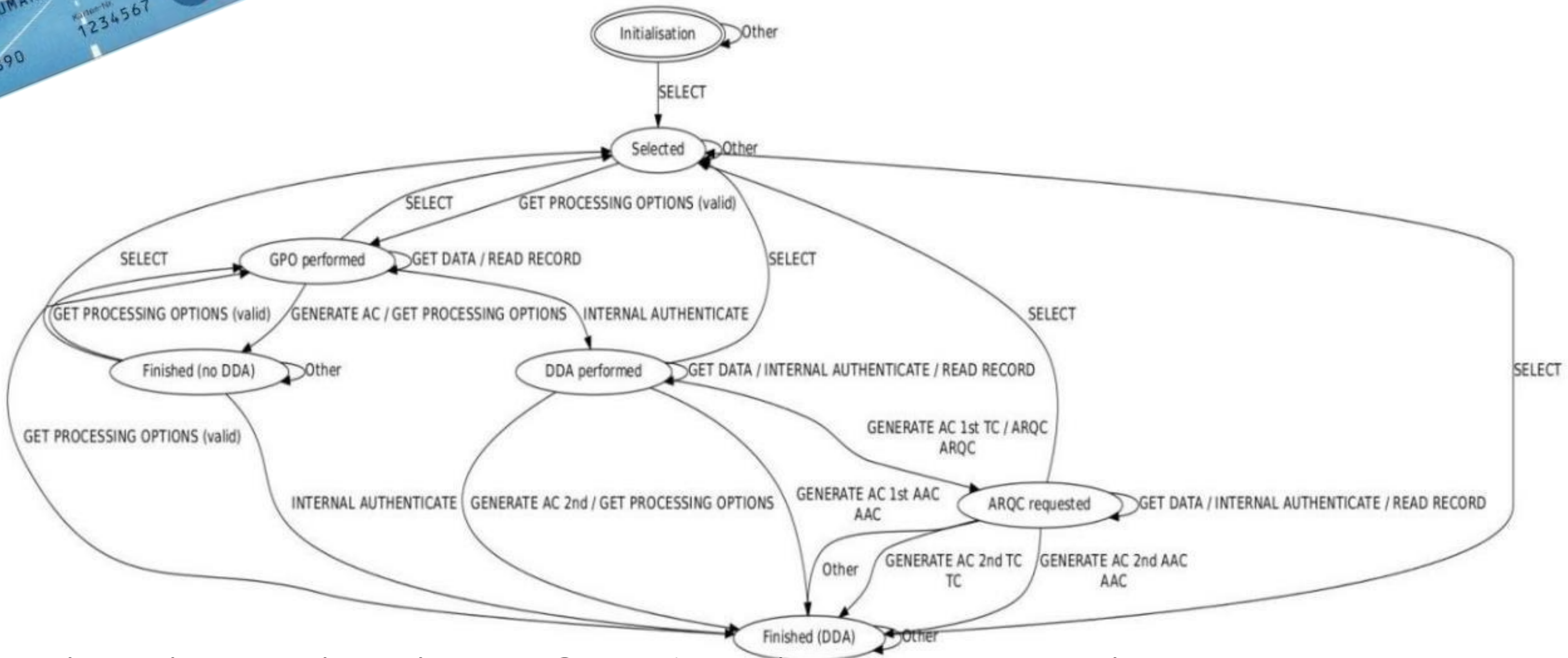
State machine inference of Maestro card



Using the LearnLib tool

State machine inference of Maestro card

merging arrows resulting
in error response

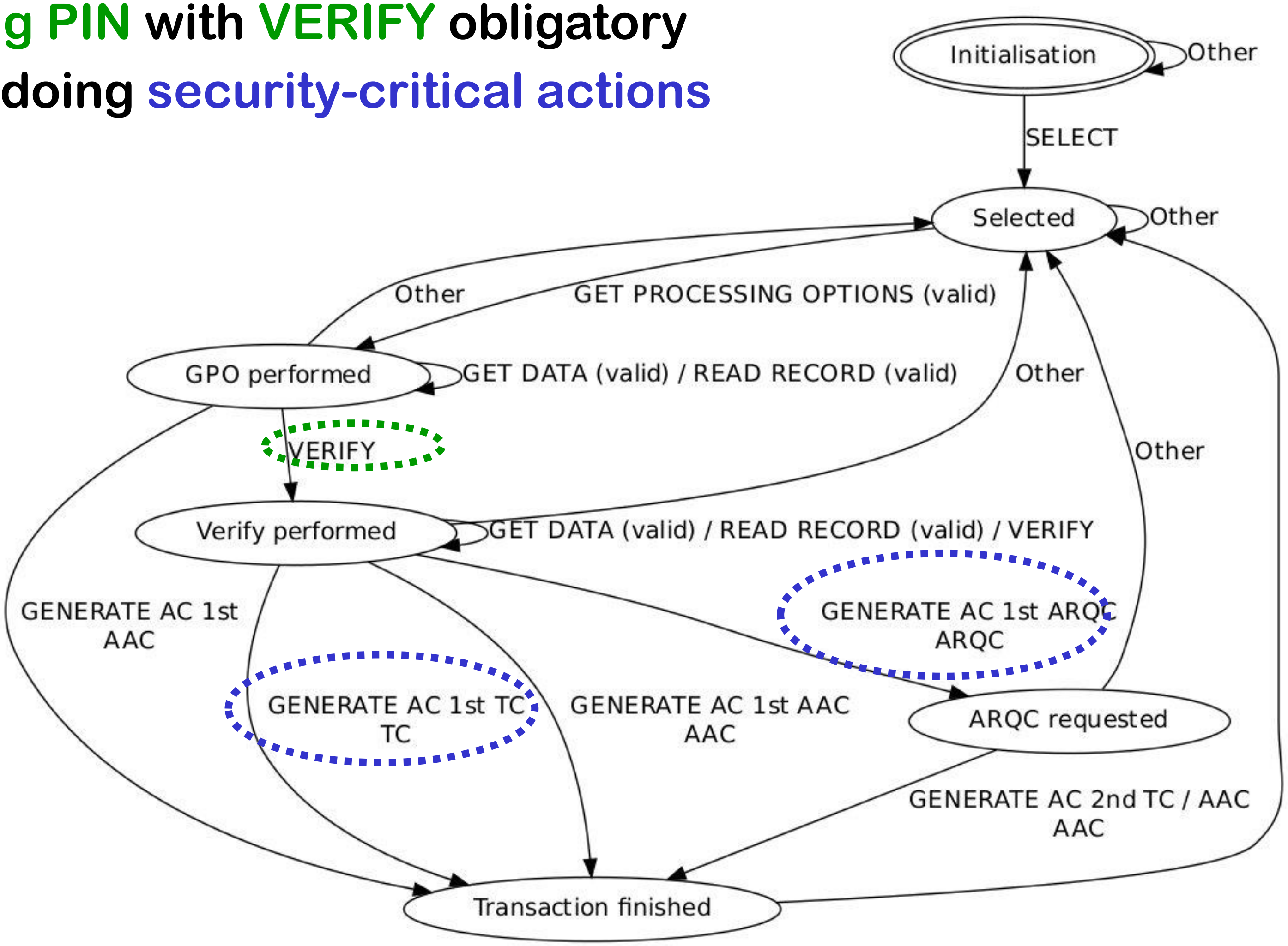


We found no bugs, but lots of variety between cards.

[Fides Aarts et al., Formal models of bank cards for free, SECTEST 2013]

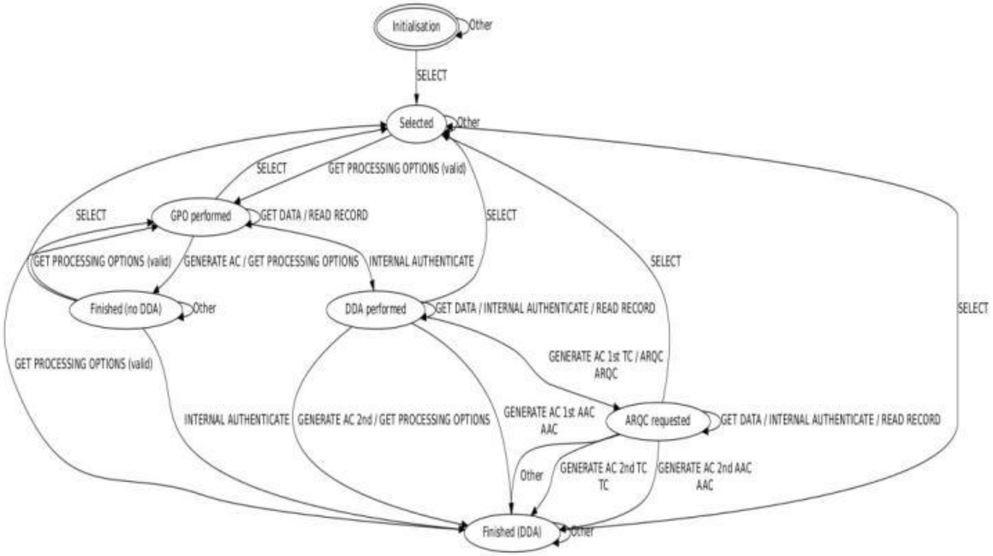
Using state machine to check security properties

entering PIN with VERIFY obligatory
before doing security-critical actions

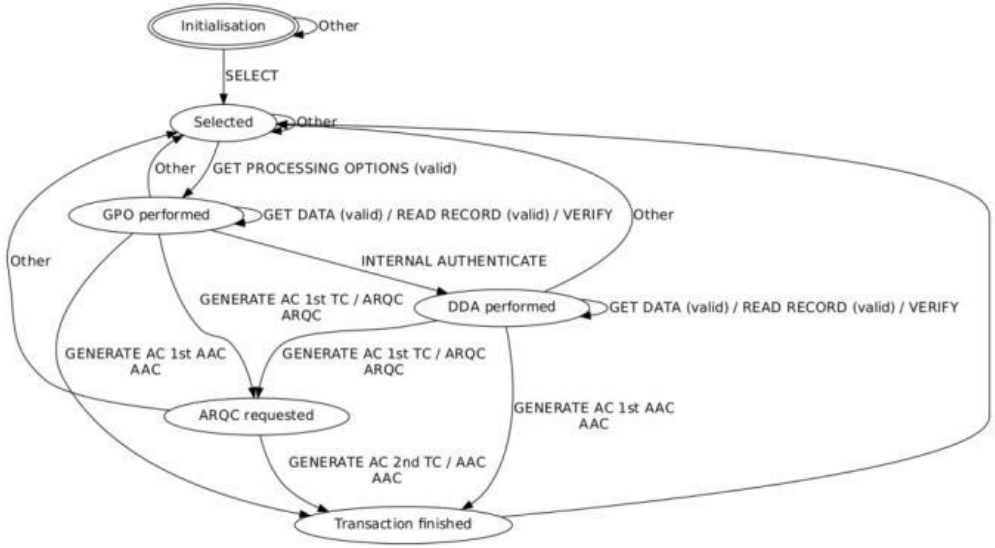


State machine of SecureCode application on Rabobank card

Understanding & comparing EMV implementations



Volksbank Maestro implementation



Rabobank Maestro implementation

Are both implementations correct & secure? And compatible?

Presumably they both pass a Maestro compliance test-suite...

So some paths (and maybe some states) are superfluous?

Contactless payments

Contactless payments)))

Contactless version of EMV with bank card or NFC smartphone

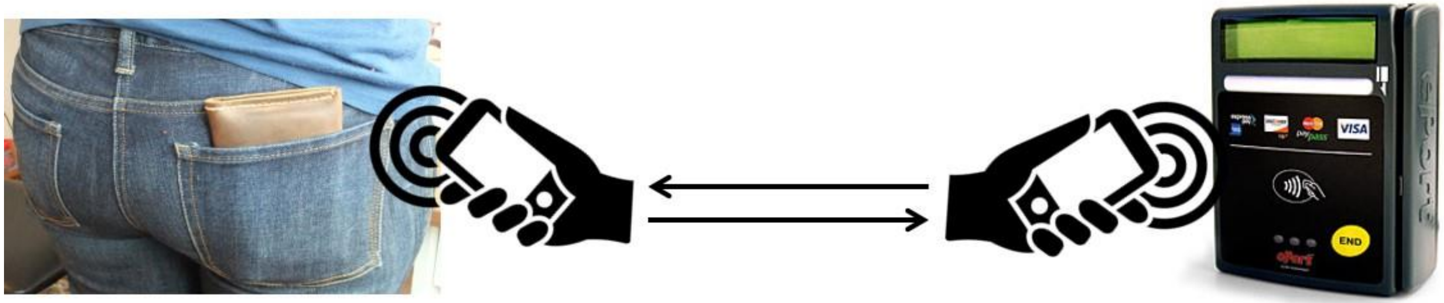
In Netherlands, for a maximum of 25 euro per individual transaction and a cumulative total of 50 euro until PIN has to be entered again.

Contactless payments

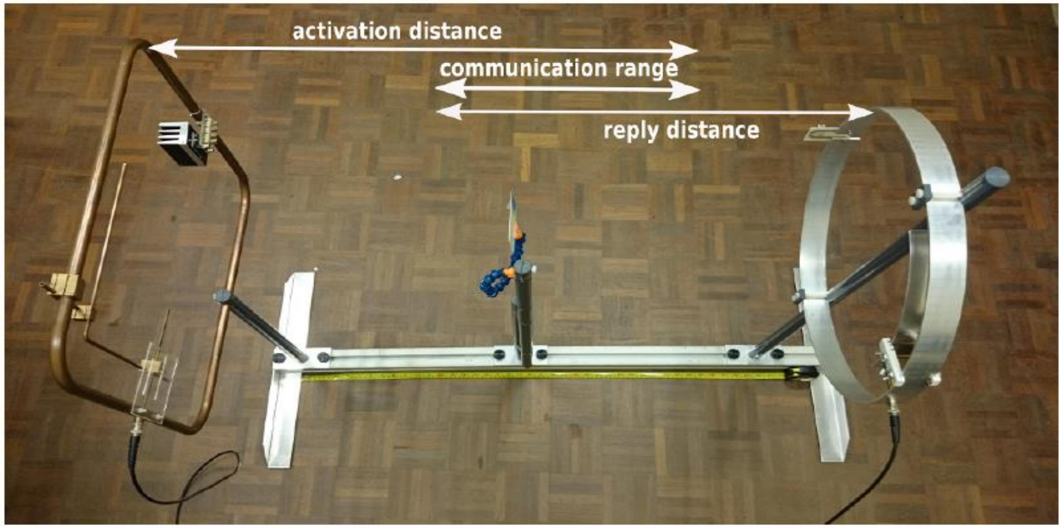
- *Who thinks that contactless payments without PIN are less secure than contact payment with PIN?*

Security of contactless payments

- It is not possible to clone a contactless card
- It is possible to do a relay attack



- But is there a good criminal business model? Probably not…
- Max. distance to activate card ≈ 40 cm



[René Habraken et al., An RFID Skimming Gate Using Higher Harmonics, RFIDSec 2015]

Risks of contactless payments

1. Risks of contactless payment ***without*** PIN

   a) You loose max. € 50 if your card is stolen

   b) You loose max. € 25 euro if you fall victim to a relay attack

   Dutch banks typically cover these losses.

2. Risks of contact payment ***with*** PIN

   a) You don't loose any money if your card is stolen

   b) You can loose €1000 or more if your card is stolen after attacker snooped on your PIN code

   Banks will typically not cover these losses…

So the 'extra security' of the PIN probably *in*creases risk for customers.

Note: technical weakness in the security ≠ risk

where risk = likelihood x impact

Configuration & Implementation mistakes

- Mistake in most first generation Dutch contactless cards:

  functionality to check the PIN code offline,
  which should only be accessible via the contact interface
  was also accessible via the contactless interface

  Possible risk for DoS attacks, rather than financial fraud?

  Flaw discovered by Anton Jongsma, Robert Kleinpenning, and Peter Maandag.

- Contactless payment terminals of one manufacturer could be crashed with a legal – but unusual – input

  - buffer overflow triggered by extended length APDU

  Flaw discovered by Jordi van den Breekel

  Why are terminals not tested for this as part of certification?

emv CERTIFIED

Payments with mobile phones

Cons

- Phone is easier to hack than a smartcard

  Cryptographic key material may be stored in secure hardware:

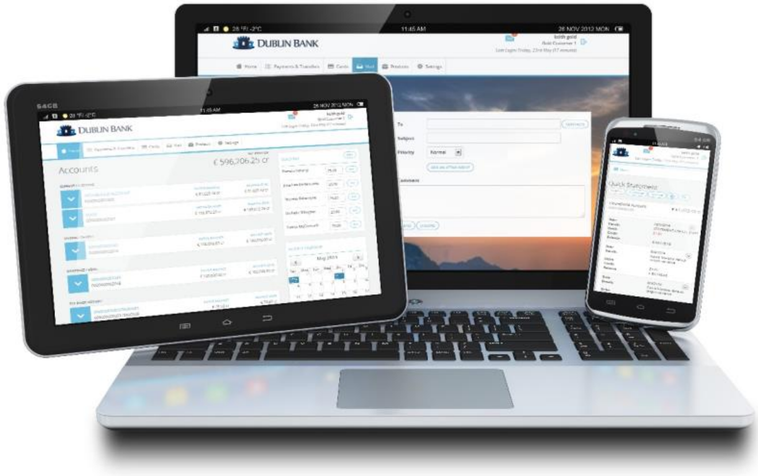
  - SIM card
  - smartcard-like hardware inside phone
    - Apple Secure Enclave on iPhone,
    - hardware-backed keystore (aka Strongbox Keymaster) on Android

Pros

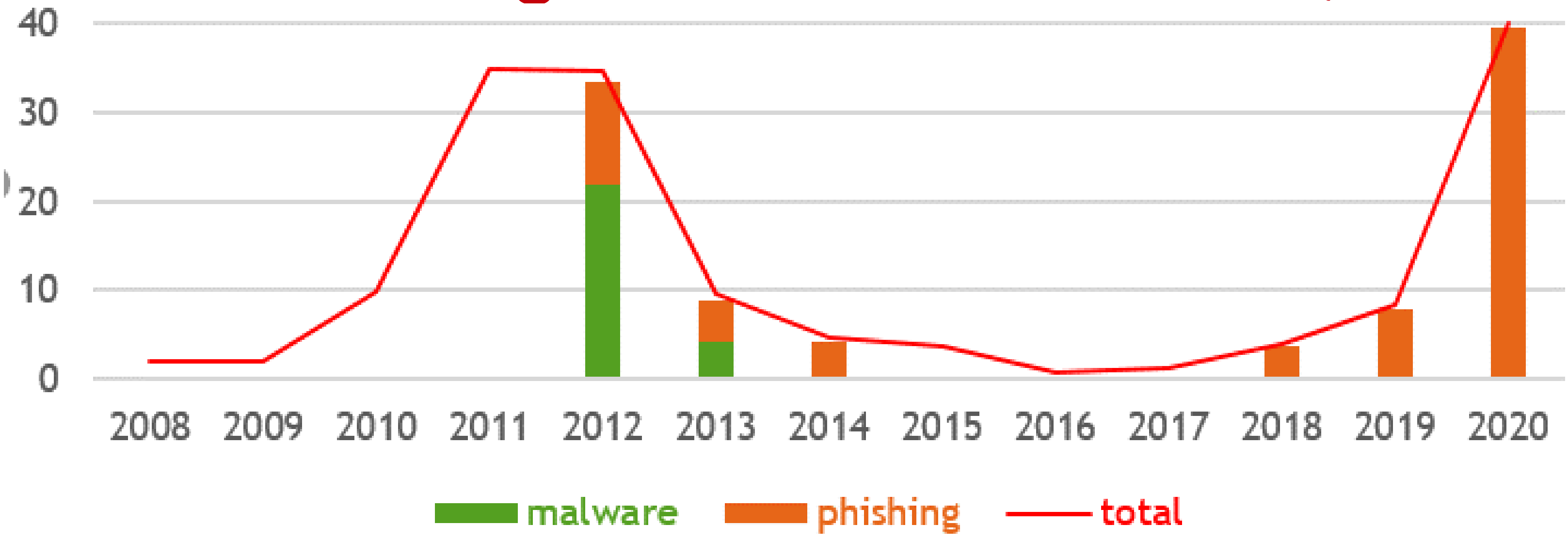
- Biometric authentication on phone is security advantage over smartcards
- Entering PIN code on phone could be more secure than on payment terminal?
- Stolen/lost phone reported faster than stolen/lost bank card?

But more online banking app is a interesting way to monetise a stolen phone…

Internet banking



Internet banking fraud in Netherlands (millions euro)



After 2012, up to last year, fraud under control thanks to

1. better monitoring - for suspicious transactions & money mules

   Recruiting money mules, to extract money from the system without being caught, is the bottleneck for attackers

2. awareness campaigns

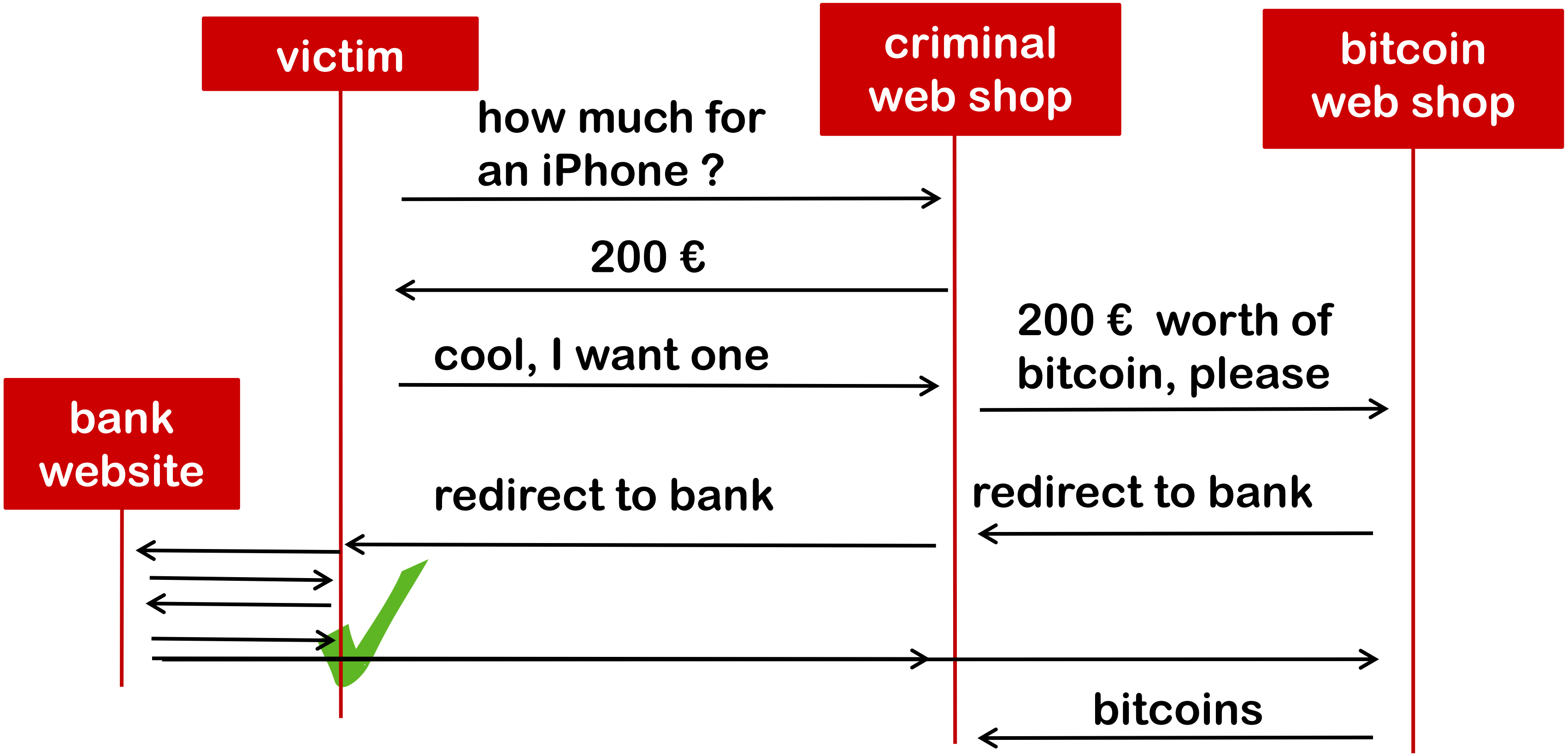
3. criminal switching to ransomware as better business model?

[Source: Dutch Payment Association]

Example attack on internet banking: malware

- Your online bank statement shows you received 3000 euro from some company you never heard of
- You get a phone call from the bank, saying that this is a mistake and asking you to transfer the money back
- You never received 3000 euro, but malware in your browser inserts the fake transaction
  - this is a so-called Man-in-the-Browser attack
- When you transfer the money back, that is not a fake transaction…

Example attack on internet banking: tricking users



- Problem: messages to user not very informative, so user does not spot the attack
- Solution: better monitoring, and banks impose extra rules on bitcoin shops & online casinos for allowing internet payments

Example attack on internet banking: SIM swapping

- For banks that use m-TANs, ie. one time passwords sent by SMS, criminals can obtain a second SIM card for your phone number
- *How?*
  - *bribe someone at the Vodaphone shop!*
- Typical countermeasure:
  - *banks make deal with telco to be told about re-issued SIMs, and then block internet payments for that SIM*

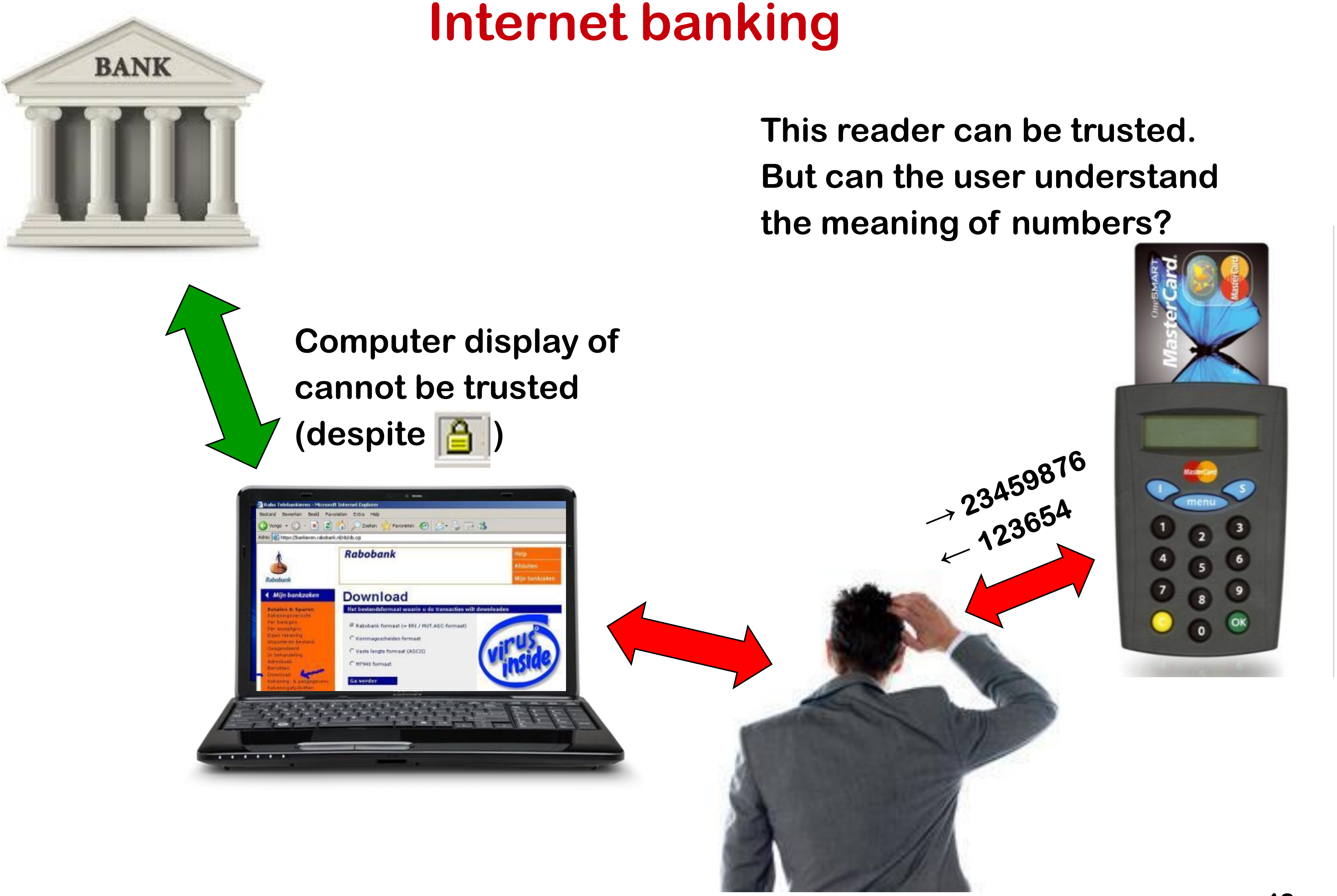
Strong authentication for online banking

- For authentication, most Dutch banks use stronger mechanisms than just username & password
  - TAN codes: one time passwords on a printed list
  - m-TAN: one time password received by SMS
  - hand-held reader that generates one-time code using bank card

  aka two-factor authentication

- Still, these mechanisms are not fool-proof…
  - eg. criminals have resorted to phoning people, pretending to be from the bank, to obtain these one-time codes

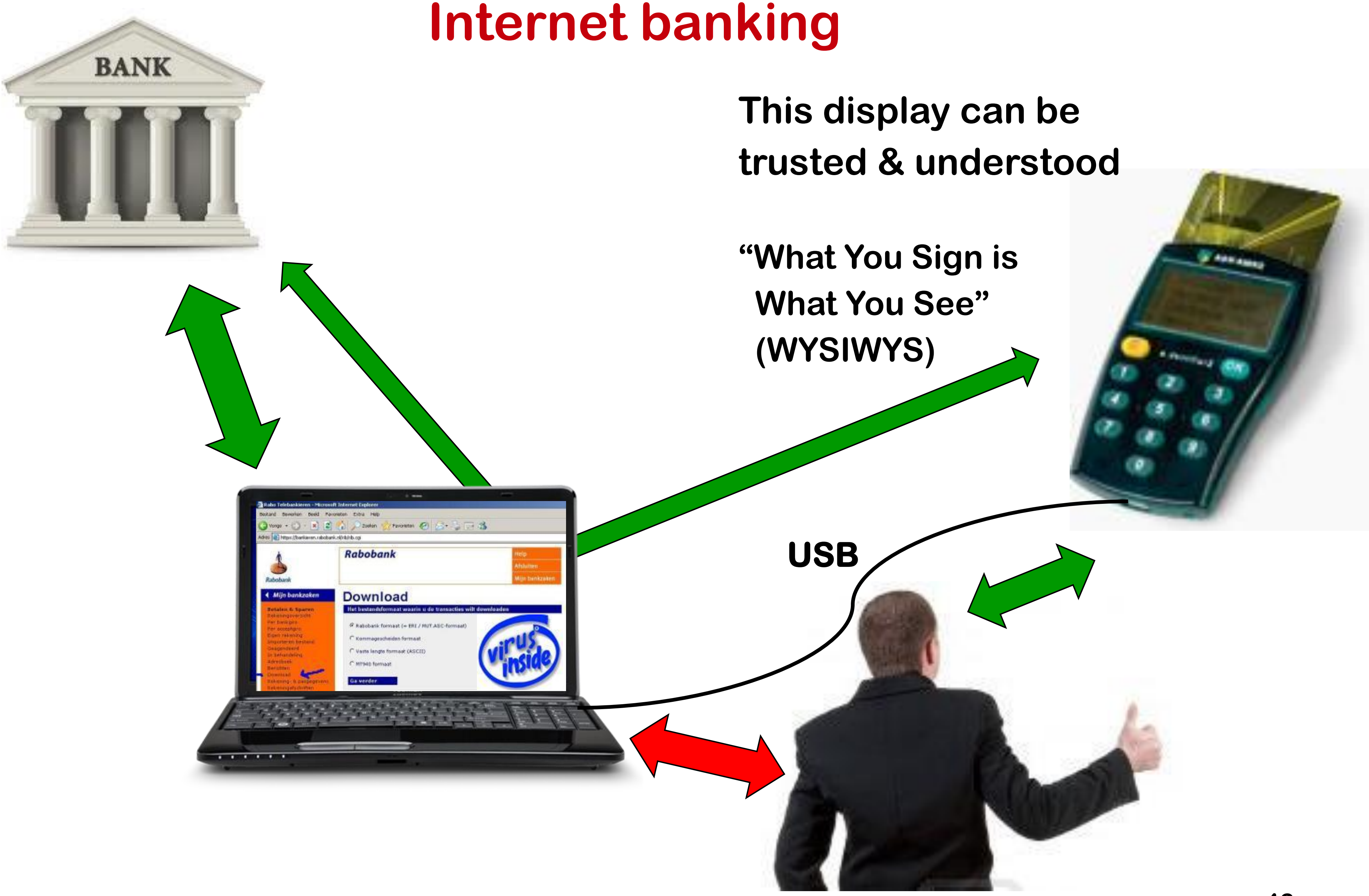
Internet banking



This reader can be trusted.
But can the user understand
the meaning of numbers?

Computer display of
cannot be trusted
(despite 🔒)

Internet banking

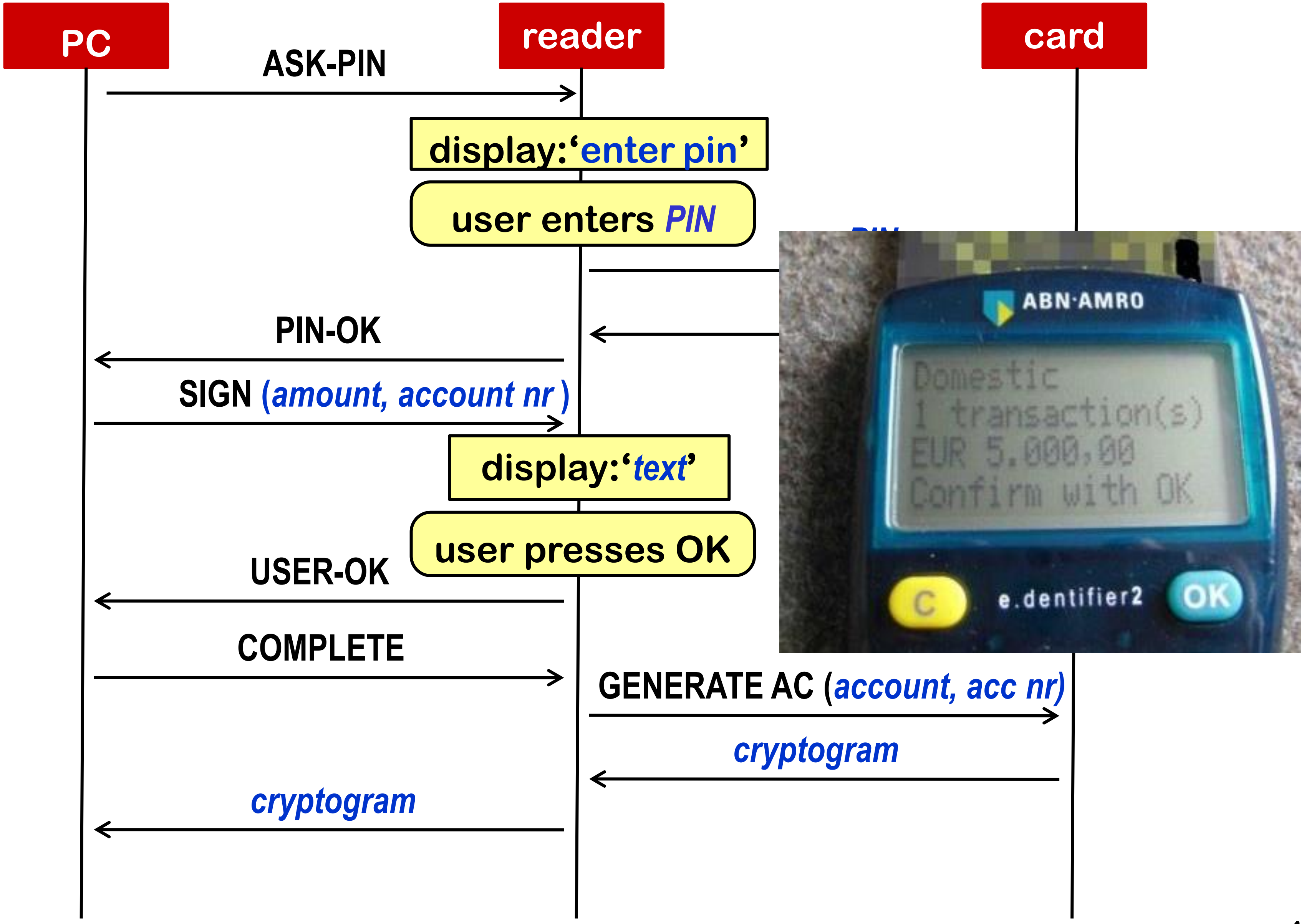


This display can be trusted & understood

“What You Sign is What You See” (WYSIWYS)

USB

Reverse-Engineered Protocol



PC

reader

card

ASK-PIN

display:‘enter pin’

user enters *PIN*

PIN-OK

SIGN (*amount, account nr*)

display:‘*text*’

user presses OK

USER-OK

COMPLETE

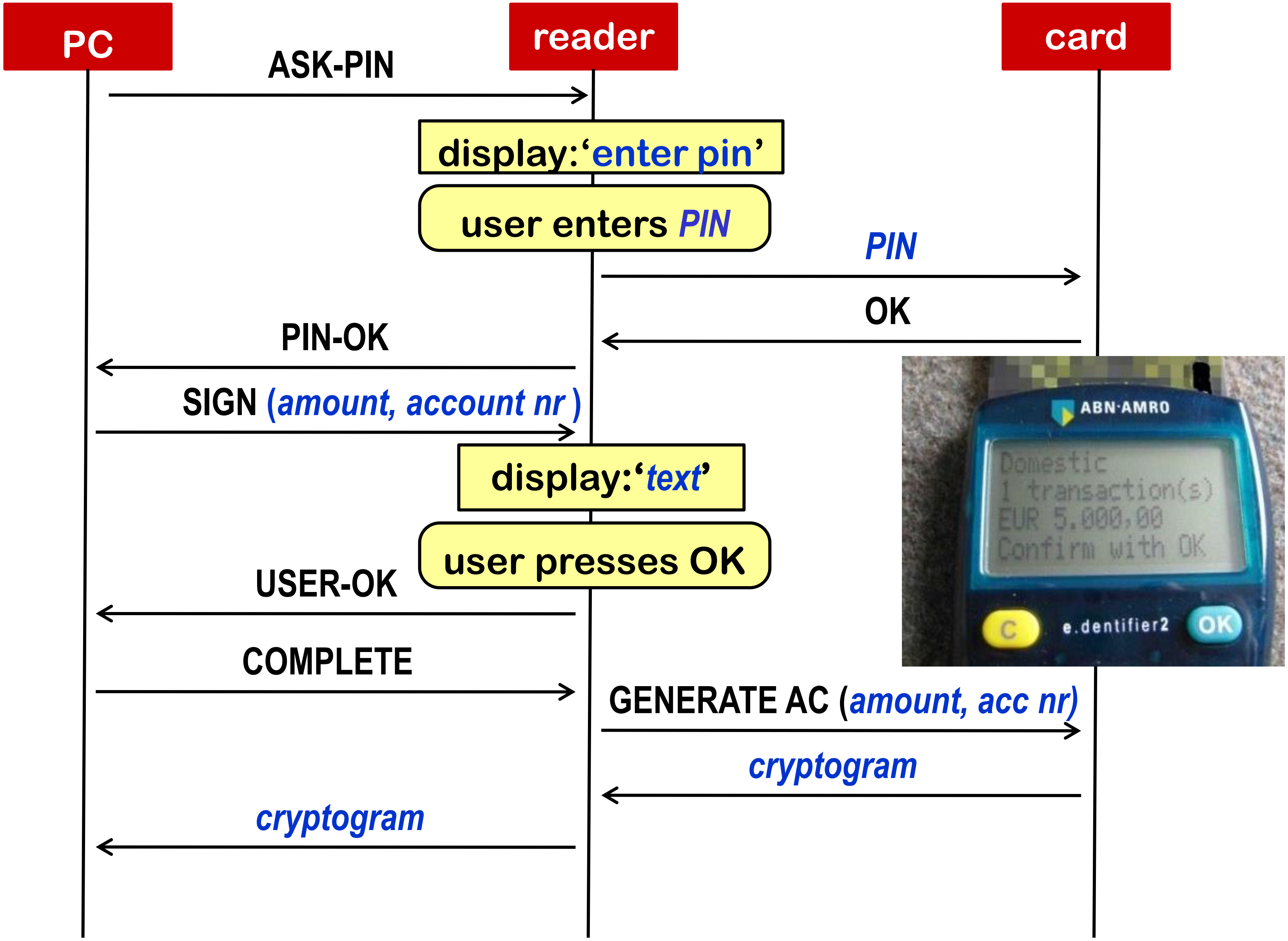
GENERATE AC (*account, acc nr)*

cryptogram

cryptogram

Domestic
1 transaction(s)
EUR 5.000,00
Confirm with OK

Reverse-Engineered Protocol



PC

reader

card

ASK-PIN

display:‘*enter pin*’

user enters *PIN*

PIN

OK

PIN-OK

SIGN (*amount, account nr*)

display:‘*text*’

user presses OK

USER-OK

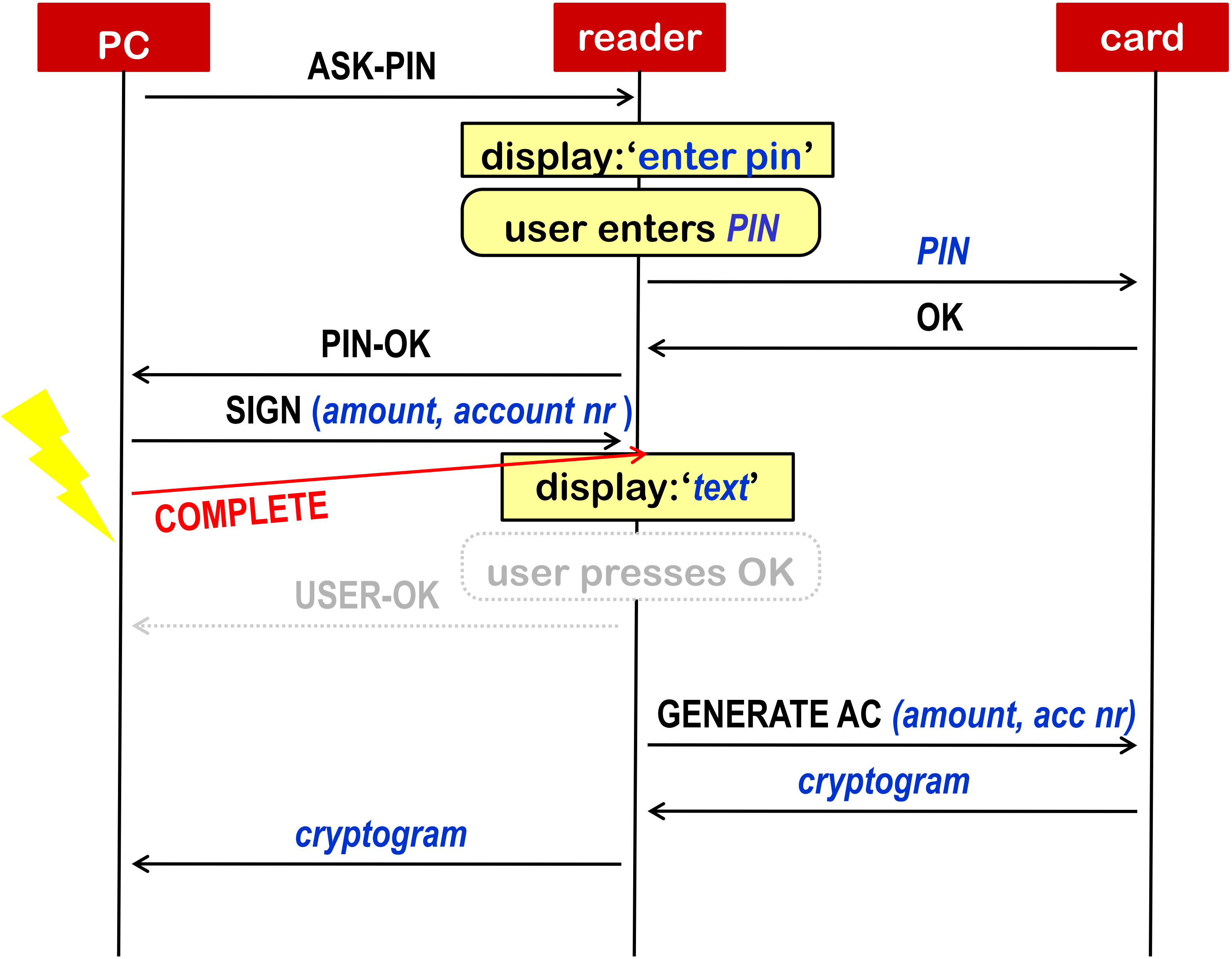
COMPLETE

GENERATE AC (*amount, acc nr)*

cryptogram

cryptogram

Attack!


PC
reader
card
ASK-PIN
display:'enter pin'
user enters PIN
PIN
OK
PIN-OK
SIGN (amount, account nr)
COMPLETE
display:'text'
user presses OK
USER-OK
GENERATE AC (amount, acc nr)
cryptogram
cryptogram

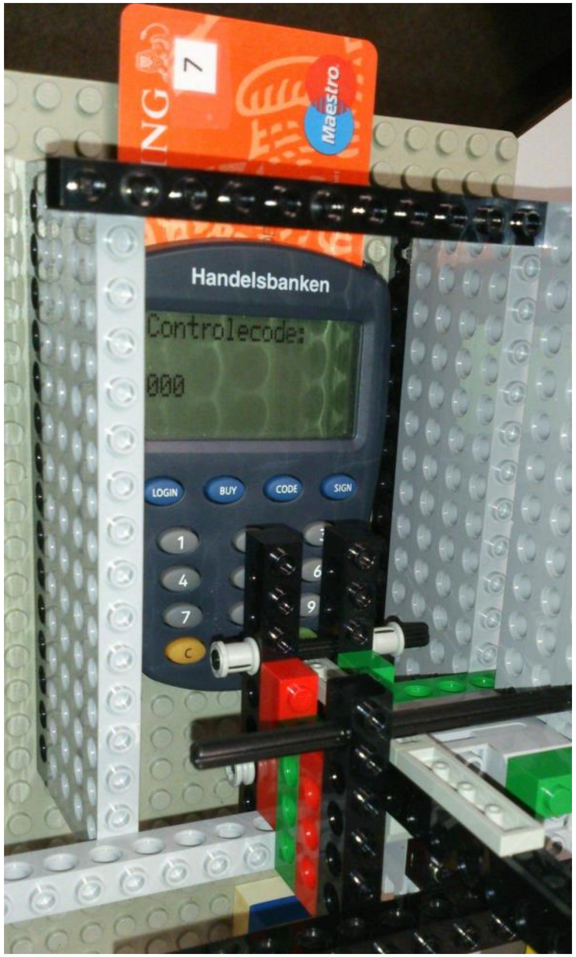
Problem with Todos/Gemalto e.dentifier2

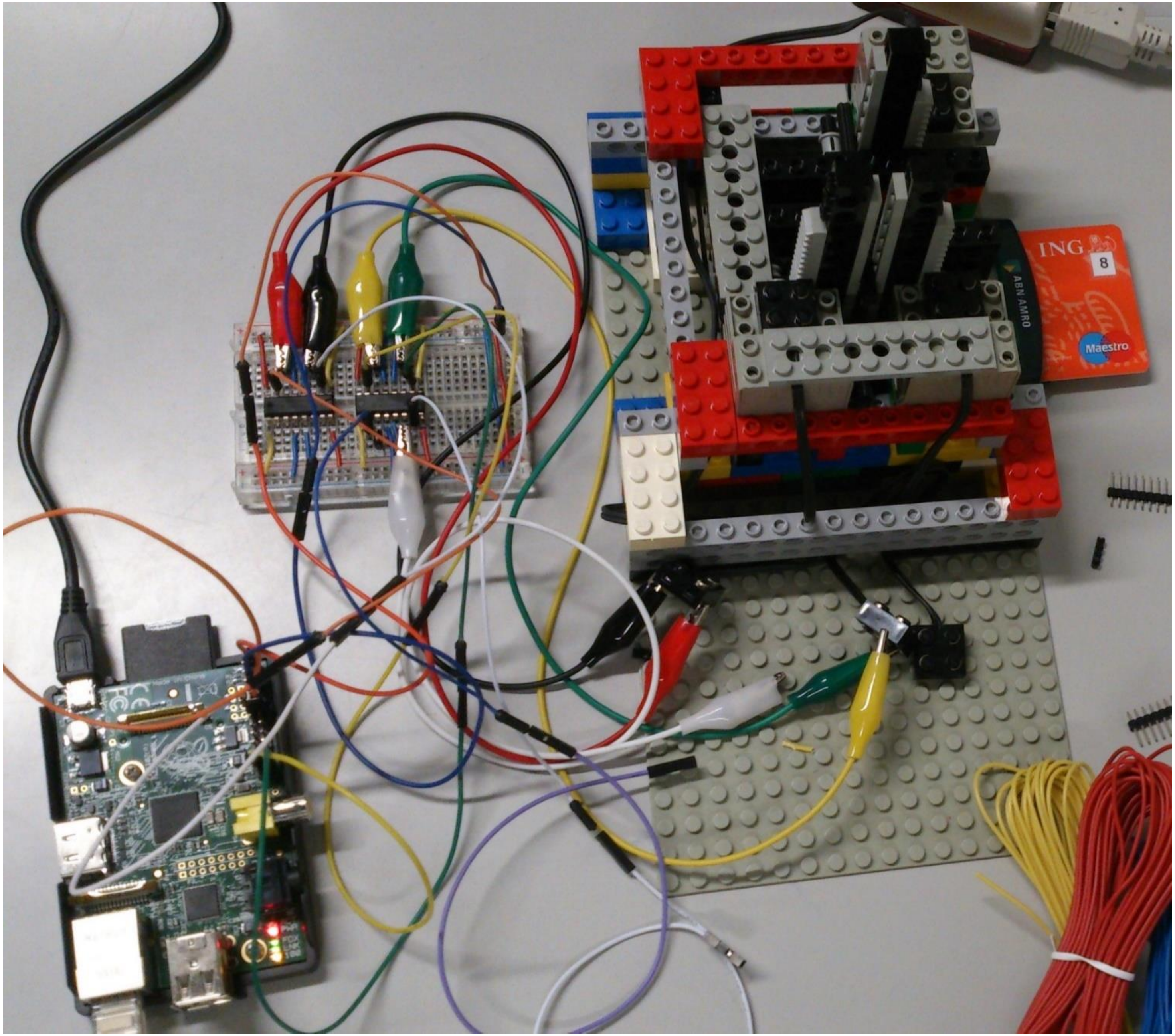
It's possible to press OK via USB cable...

Malware on an infected PC could change all the transaction details and press OK

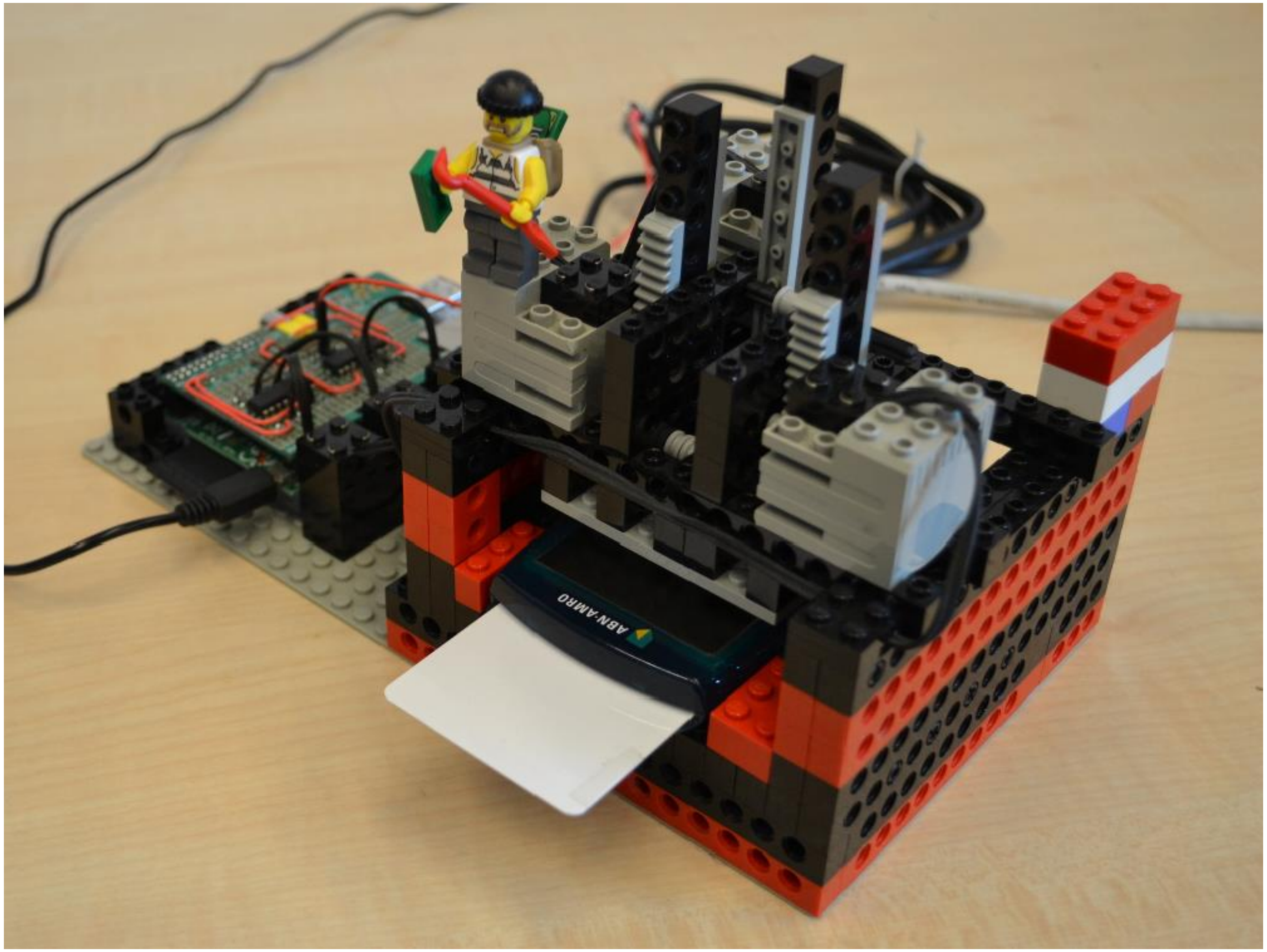
This is a flaw in the state machine!
Can we find it automatically?

Operating the keyboard using LEGO



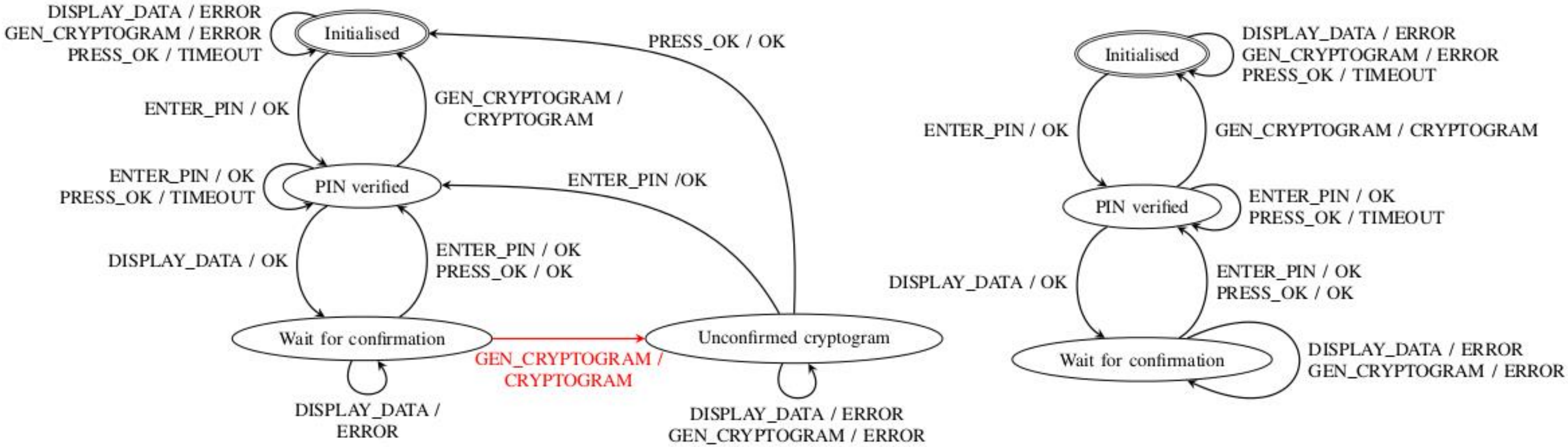

ING
8
ABN AMRO
Maestro

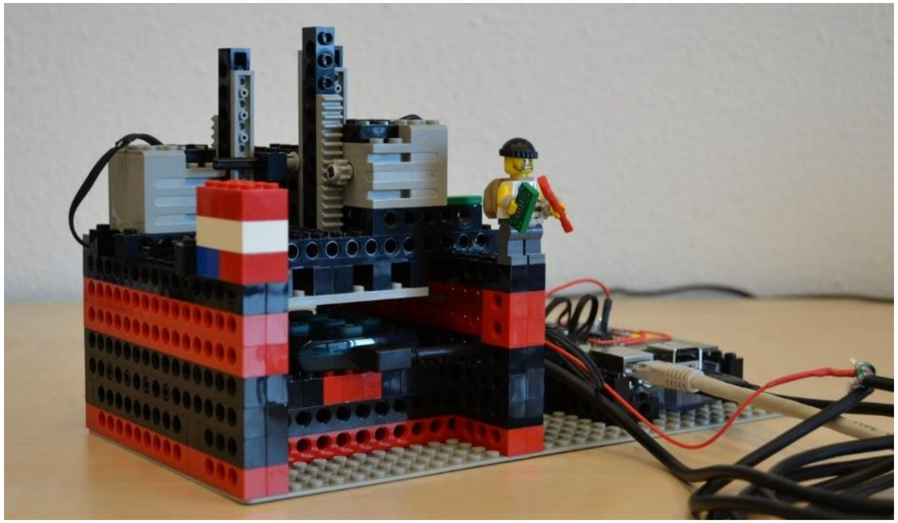
Our Lego movie online: https://tinyurl.com/legolearning



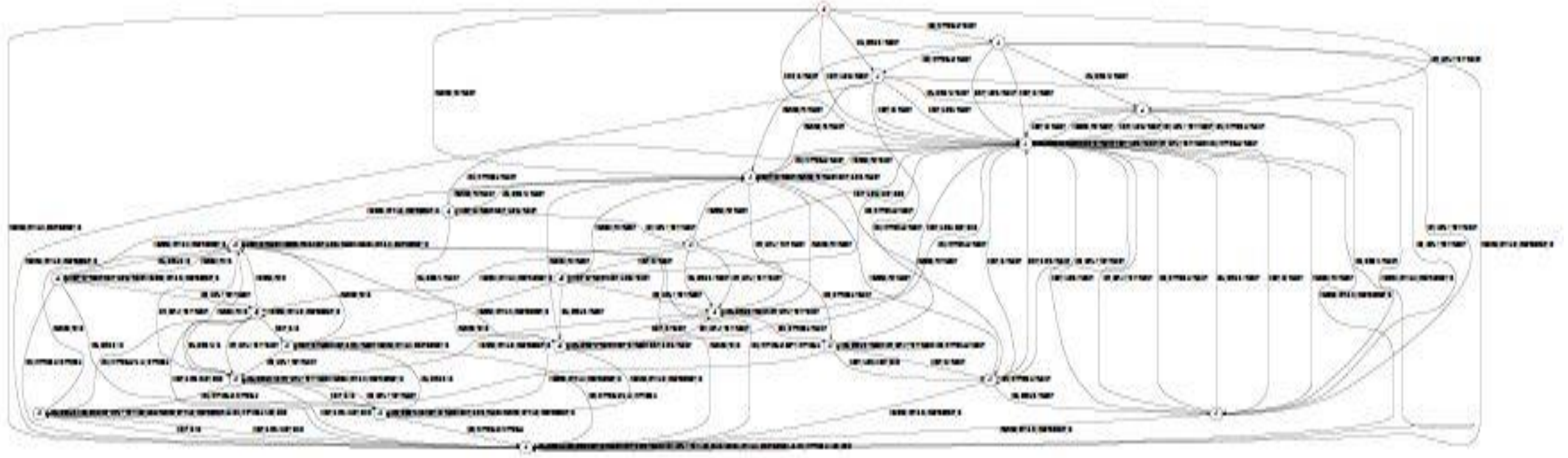
[Automated Reverse Engineering using LEGO, WOOT 2014]

State machines of old vs new e.dentifier2





Would you trust this to be secure?



Full state machine of the handheld card reader

Do you think whoever designed or implemented this is confident that this is secure?

Or that all this behaviour is necessary?

Conclusions

Conclusions

- General trend: from prevention to better detection & response
- A *technical* security flaw not always a serious security risk.
  The real issue: can attackers find a good business model?
  - The bad news here: ransomware is a great business model for almost any security weakness
- Some silly security flaws by reputable companies & vendors
  - Who is *really* taking responsibility for the security ?
    - Individual banks? Their suppliers? 3rd parties doing certification? MasterCard & Visa, who also approve vendors & certifiers?
  - How much security is just Cover-Your-Ass security?

Why banking security is easy!

- Banks can measure attacks & quantify their costs euros, so
  - Trends in attacks can be monitored
  - Success of defensive measures can be measured
  - This provides a rational basis for security decisions

- In other industries this is MUCH harder
  - Eg for critical infrastructures or hospitals:
    - How much can cyber protection of the electricity grid cost?
    - How much can patient privacy cost?
  - Ransomware may play a ‘useful’ role here…

Thanks for your attention