Hi-tech bank robbery

Erik Poll

Digital Security

Radboud University Nijmegen

Radboud University Nijmegen

Overview

- Some historical anecdotes & trends in e-banking fraud
  1. skimming
  2. EMV (het nieuwe pinnen)
  3. online banking
- incl. some of our own research

  on more rigorous design and analysis

  Joint work PhD students Joeri de Ruiter and Fides Aarts, and MSc students Arjan Blom, Jordi van den Breekel, Georg Chalupar, Anton Jongsma, Robert Kleinpenning, Peter Maandag, and Stefan Peherstorfer.

Are there any lessons to be learnt for other fields?

Skimming

Skimming

Mag-stripe on bank card contains digitally signed information

but... this info can be copied

Example skimming equipment

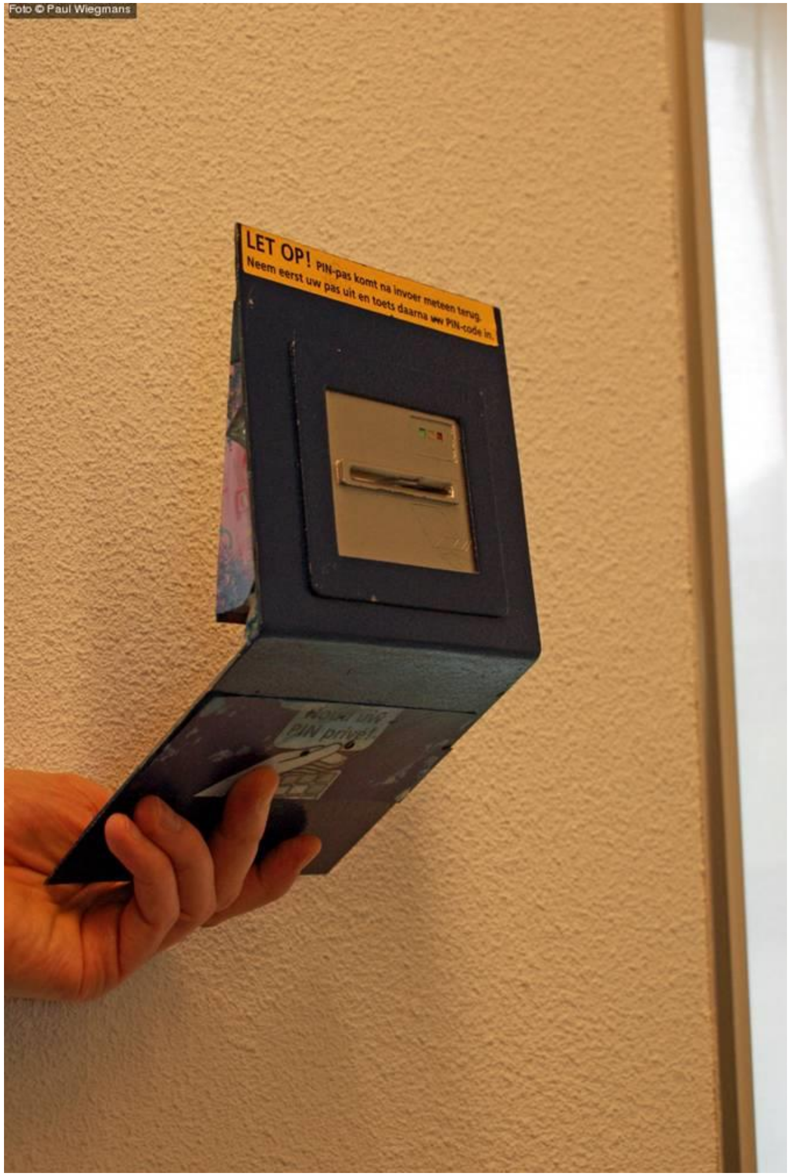
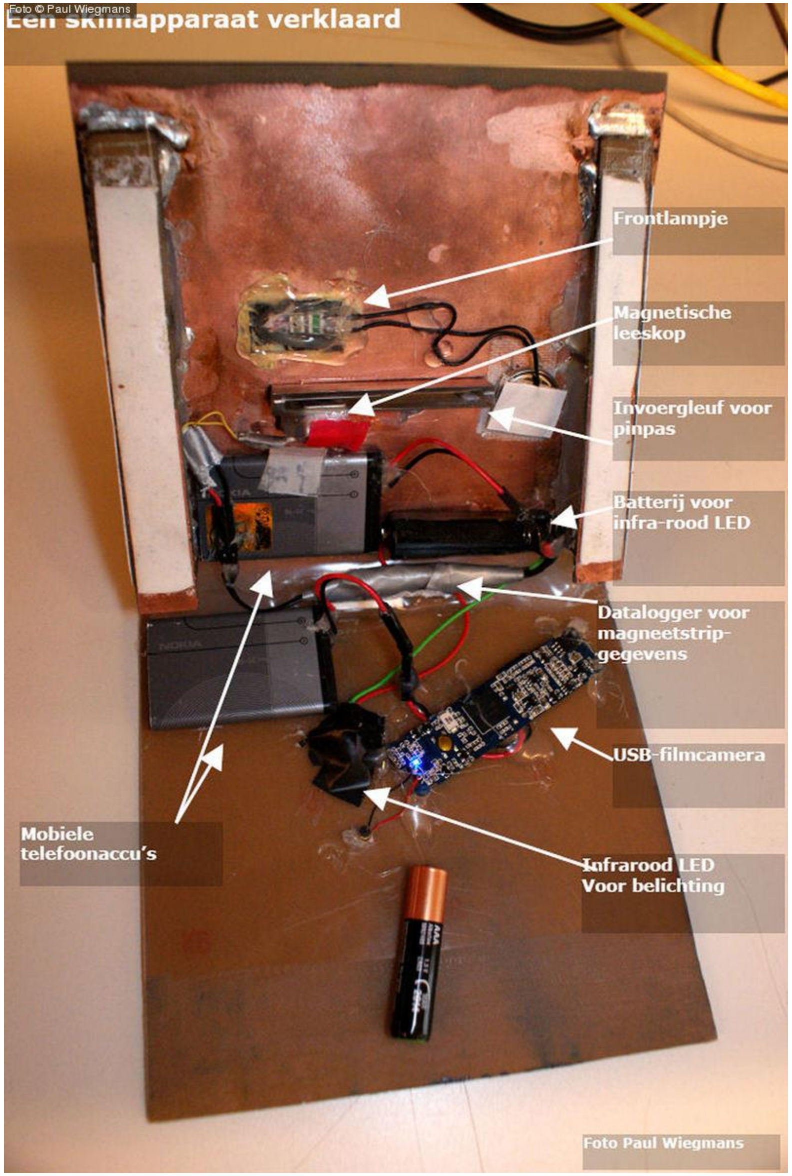


Foto © Paul Wiegmans
LET OP! PIN-pas komt na invoer meteen terug.
Neem eerst uw pas uit en toets daarna uw PIN-code in.



Skimming fraud in the Netherlands

2007 : 15 M€

2008 : 31 M€

2009 : 36 M€

2010: 19.7M€ - better detection

2011: 38.9 M€

On a total of over 100 billion €, so fraud only around 0.03%

Hence migration to EMV (chip) cards moved forward from 2013 to 2011

het
nieuwe
pinnen.nl

Does EMV reduce skimming?

- UK introduced EMV in 2006

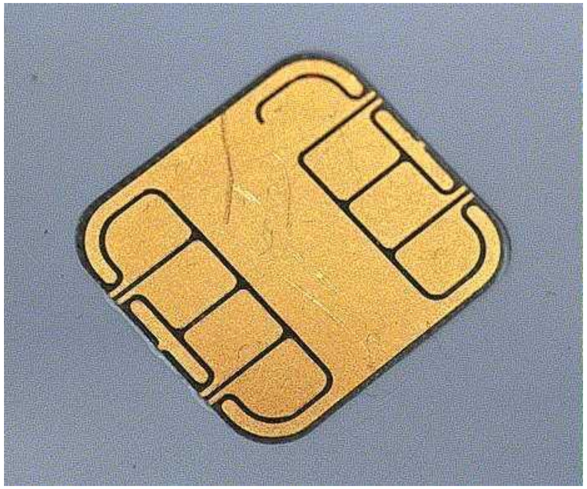
	2005	2006	2007	2008
domestic	79	46	31	36
foreign	18	53	113	134

Skimming fraud with UK cards, in millions £

- Magstripes that are cloned can still be used in countries don't use the chip...
- Blocking cards for use outside EU (geoblocking) helps a lot!
  - Skimming in Netherlands reduced to 1.3 M€ in 2014
- Skimmers have now moved to the US, and the US is (slowly) migrating to EMV

EMV

(Europay-MasterCard-Visa)



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
- Contact and contactless version
- The protocol makes cloning chips based on eavesdropping impossible

Skimming 2.0

- In 2009, criminals put tampered card readers *inside* ABN-AMRO bank branches to skim cards
  - For *backwards compatibility*, the chip can report the magstripe data...
  - Both magstripe data and PIN code are sent plaintext from card to this reader
  - Criminals caught & convicted in 2011

- Cards have been improved to avoid this, and magstrip data should now be different from info on the chip

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), ...**
- **Additional EMV contactless specs: another 10 books, > 2000 pages**
  - **yet more modes and options....**

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

The response of the card is *not* authenticated

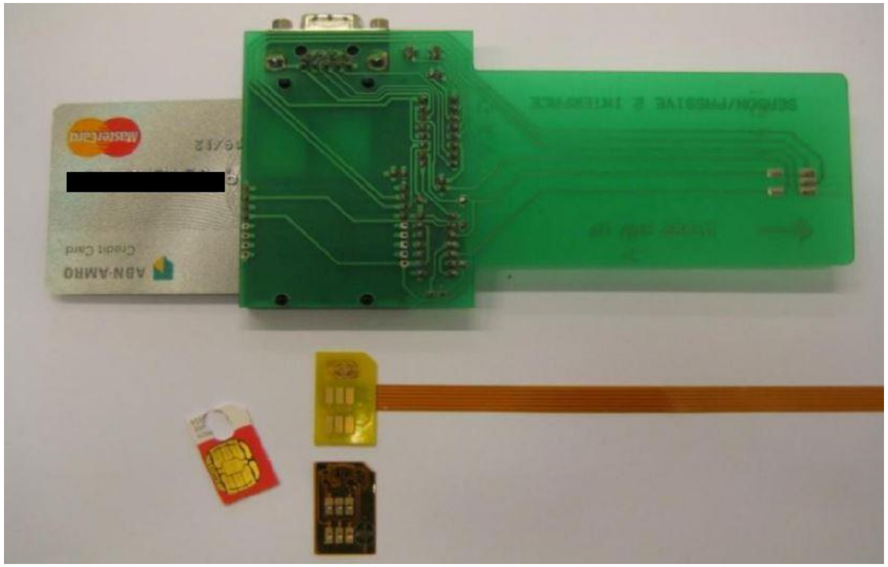
- ie. it is not cryptographically signed or MAC-ed

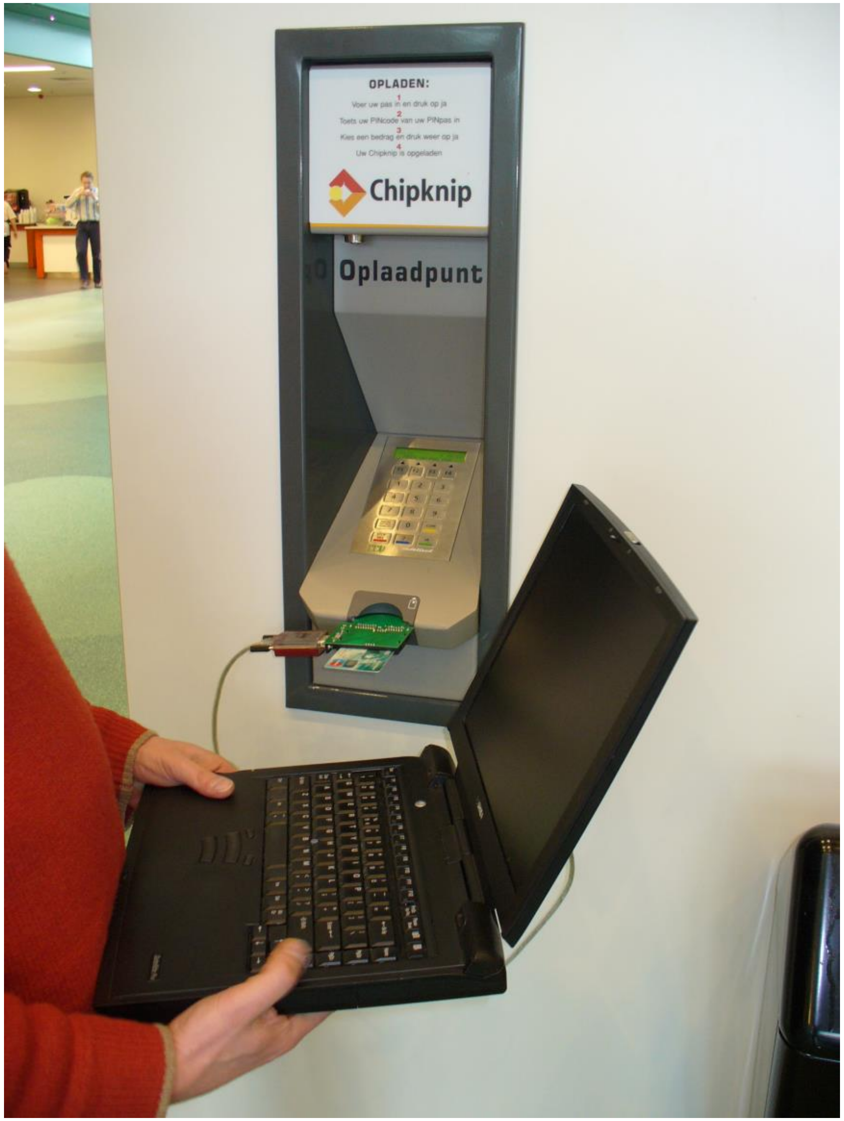
so the terminal can be fooled by a Man-in-the-Middle attack

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]

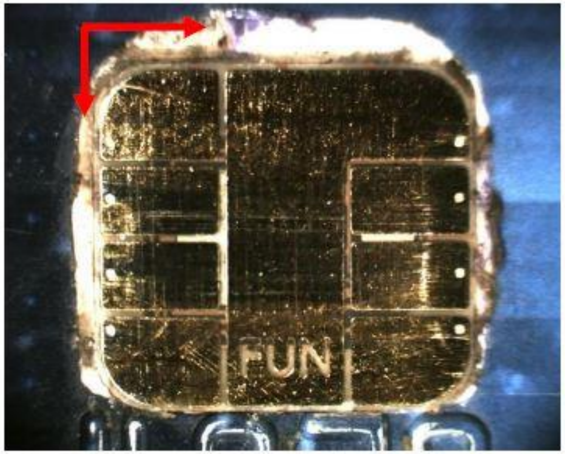
Our Man-in-the-Middle set-up

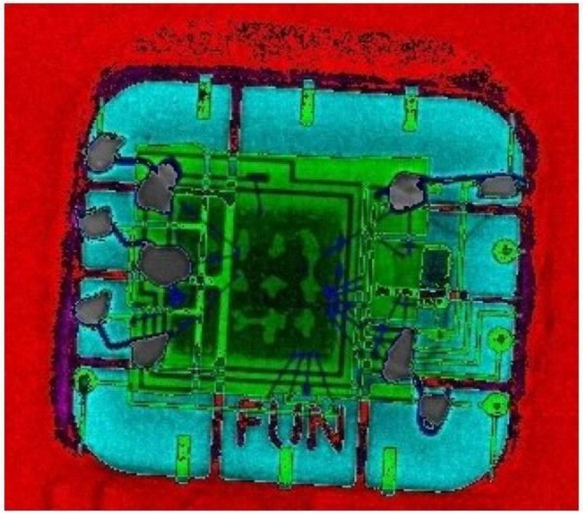


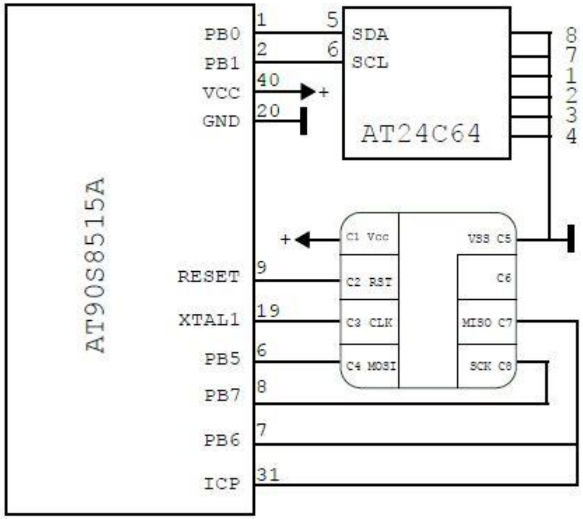

OPLADEN:
1
Voer uw pas in en druk op ja
2
Toets uw PINcode van uw PINpas in
3
Kies een bedrag en druk weer op ja
4
Uw Chipknip is opgeladen
Chipknip
Oplaadpunt

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]

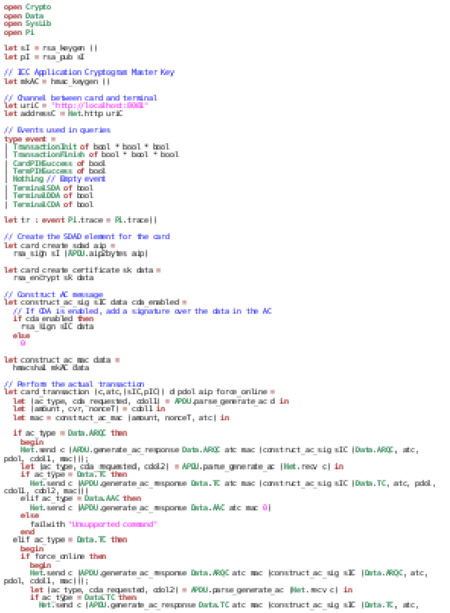
Complexity of EMV specs

- Moral of the story: specs too complex to understand
  - long documents
  - little or no discussion of security goals or design choices
  - little abstraction or modularity

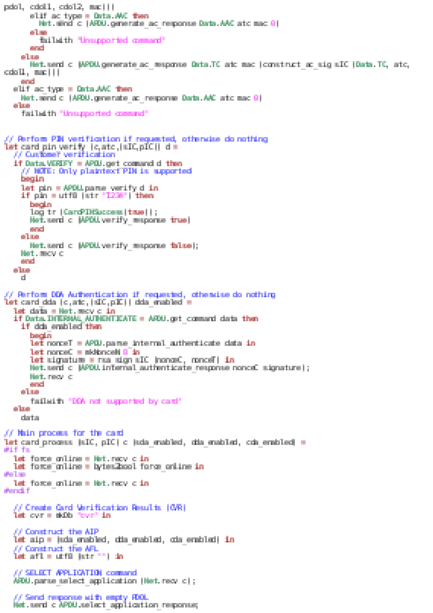
- Who really takes responsibility for ensuring these specs are secure? EMVCo, the credit card companies behind EMVCo, or individual banks?

- Can we provide some scientific rigour?

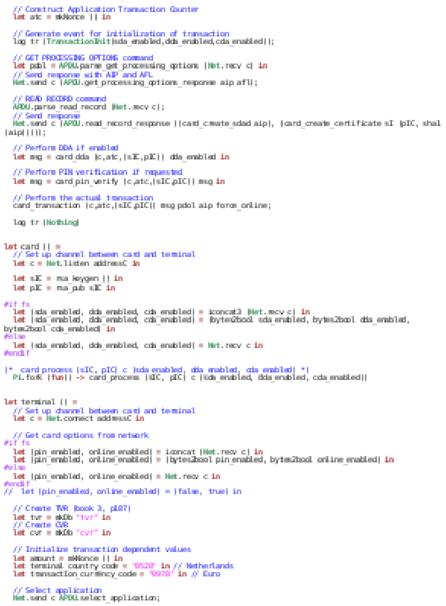
Formalisation of EMV in F#



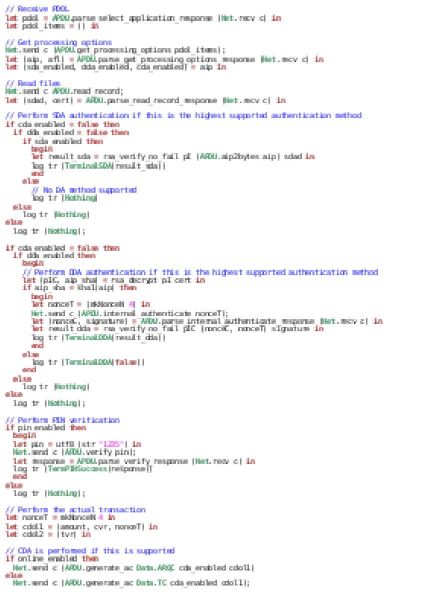
```
open Crypto
open Data
open SysLib
open Pi

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,
```



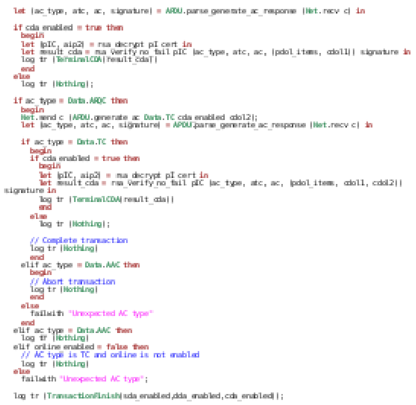
```
pdol, cdol1, cdol2, mac)))
      elif ac_type = Data.AAC then
        Net.send c (APDU.generate_ac_response Data.AAC atc mac 0)
      else
        failwith "Unsupported command"
      end
    else
      Net.send c (APDU.generate_ac_response Data.TC atc mac (construct_ac_sig sIC (Data.TC, atc,
cdol1, 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 protocol 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!

Complexity: example configuration flaw

Mistake on the first generation contactless cards issued in the Netherlands:

- functionality to check the PIN code,

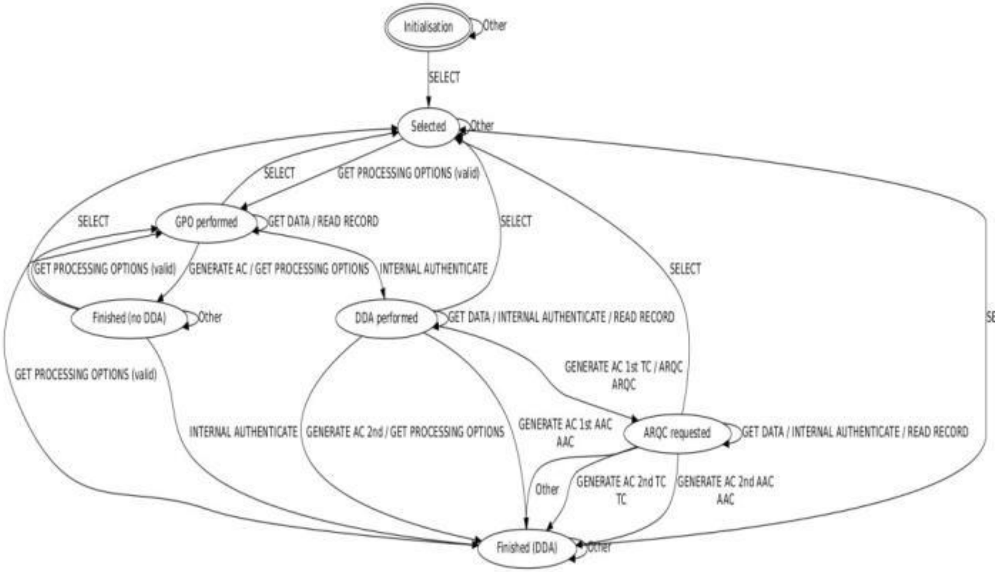
  which should only be accessible via the contact interface

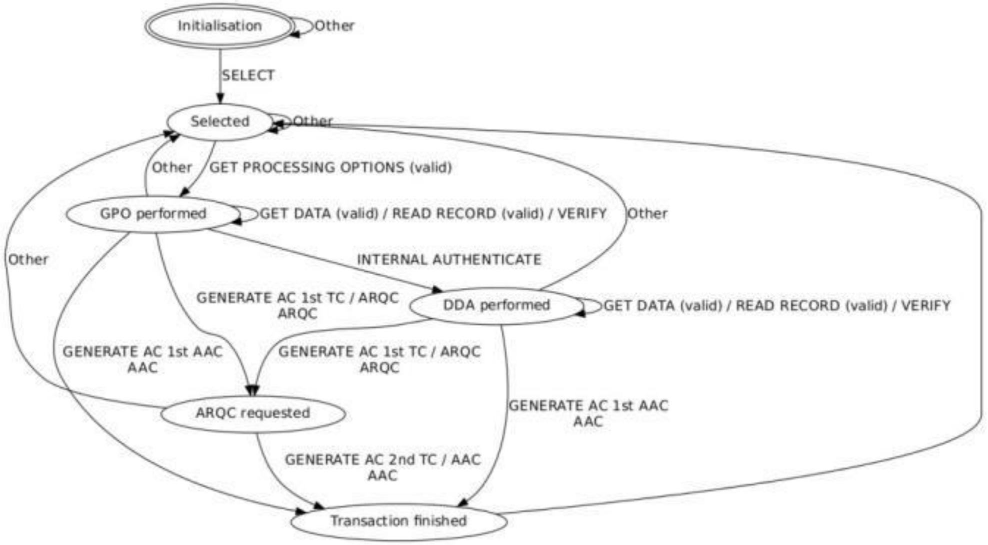
  was also accessible via the contactless interface

Possible risk for DoS attacks, rather than financial fraud?

Flaw discovered bij Radboud students Anton Jongsma, Robert Kleinpenning, and Peter Maandag.

State machine inference: automated testing


Initialisation
Other
SELECT
Selected
Other
SELECT
GET PROCESSING OPTIONS (valid)
SELECT
GPO performed
GET DATA / READ RECORD
SELECT
GET PROCESSING OPTIONS (valid)
GENERATE AC / GET PROCESSING OPTIONS
INTERNAL AUTHENTICATE
SELECT
Finished (no DDA)
Other
DDA performed
GET DATA / INTERNAL AUTHENTICATE / READ RECORD
SE
GET PROCESSING OPTIONS (valid)
GENERATE AC 1st TC / ARQC
ARQC
INTERNAL AUTHENTICATE
GENERATE AC 2nd / GET PROCESSING OPTIONS
GENERATE AC 1st AAC
AAC
ARQC requested
GET DATA / INTERNAL AUTHENTICATE / READ RECORD
Other
GENERATE AC 2nd TC
TC
GENERATE AC 2nd AAC
AAC
Finished (DDA)
Other



Volksbank Maestro implementation

Rabobank Maestro implementation

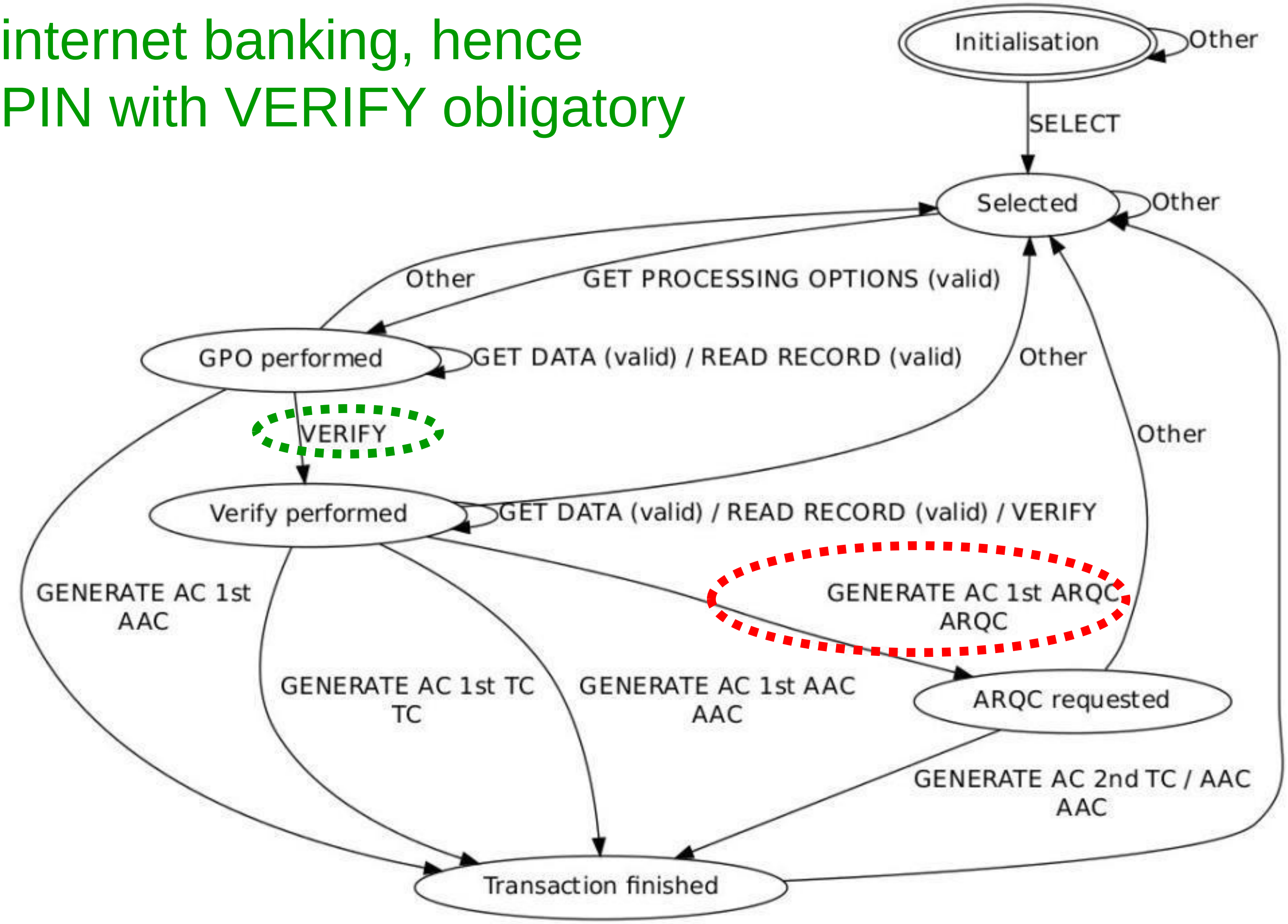
We can automatically infer the state machine of an EMV smartcard, using only black-box testing, in 30 minutes.

No security flaws found, but lots of differences between cards!

[Fides Aarts et al., *Formal Models of bank cards for free*, SECTEST 2014]

MasterCard SecureCode application on Rabobank card

used for internet banking, hence
entering PIN with VERIFY obligatory



Online banking

Internet banking fraud in Netherlands

2008	2.1 M€
2009	1.9 M€
2010	9.8 M€ (7100€ per incident)
2011	35 M€ (4500€ per incident)
2012	34.8 M€
2013	9.6 M€
2014	4.7 M€

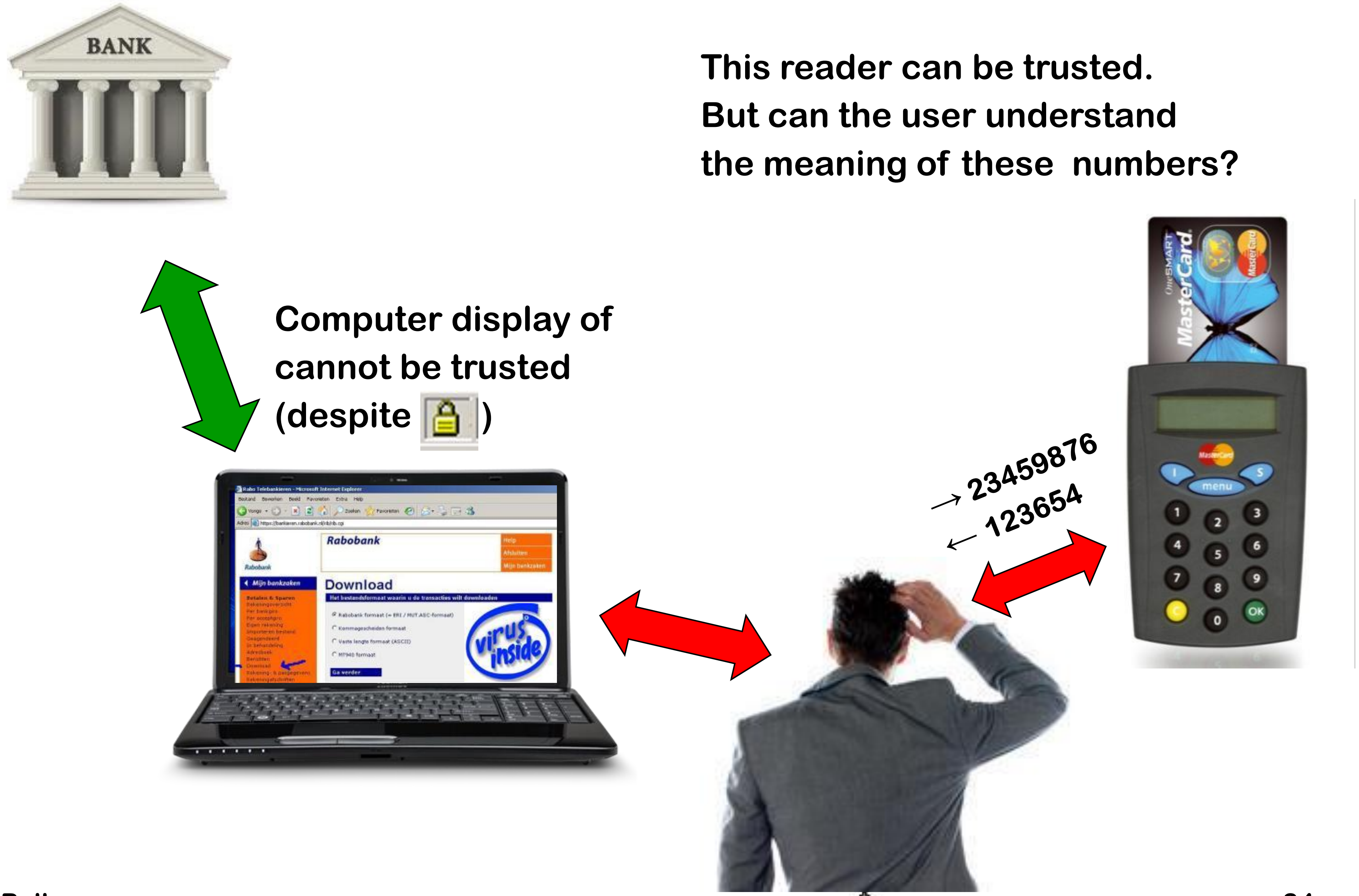
[Source: NVB & Betaalvereniging]

EMV-CAP

- Another variant of EMV chip for internet banking and e-commerce
- Goal: strong authentication, by using hand-held card reader in combination with bank card and PIN code
- CAP specs are secret but largely reverse-engineered

- Some silly flaws, eg sending a fixed challenge 000000 to the smartcard instead of the random number the user types in

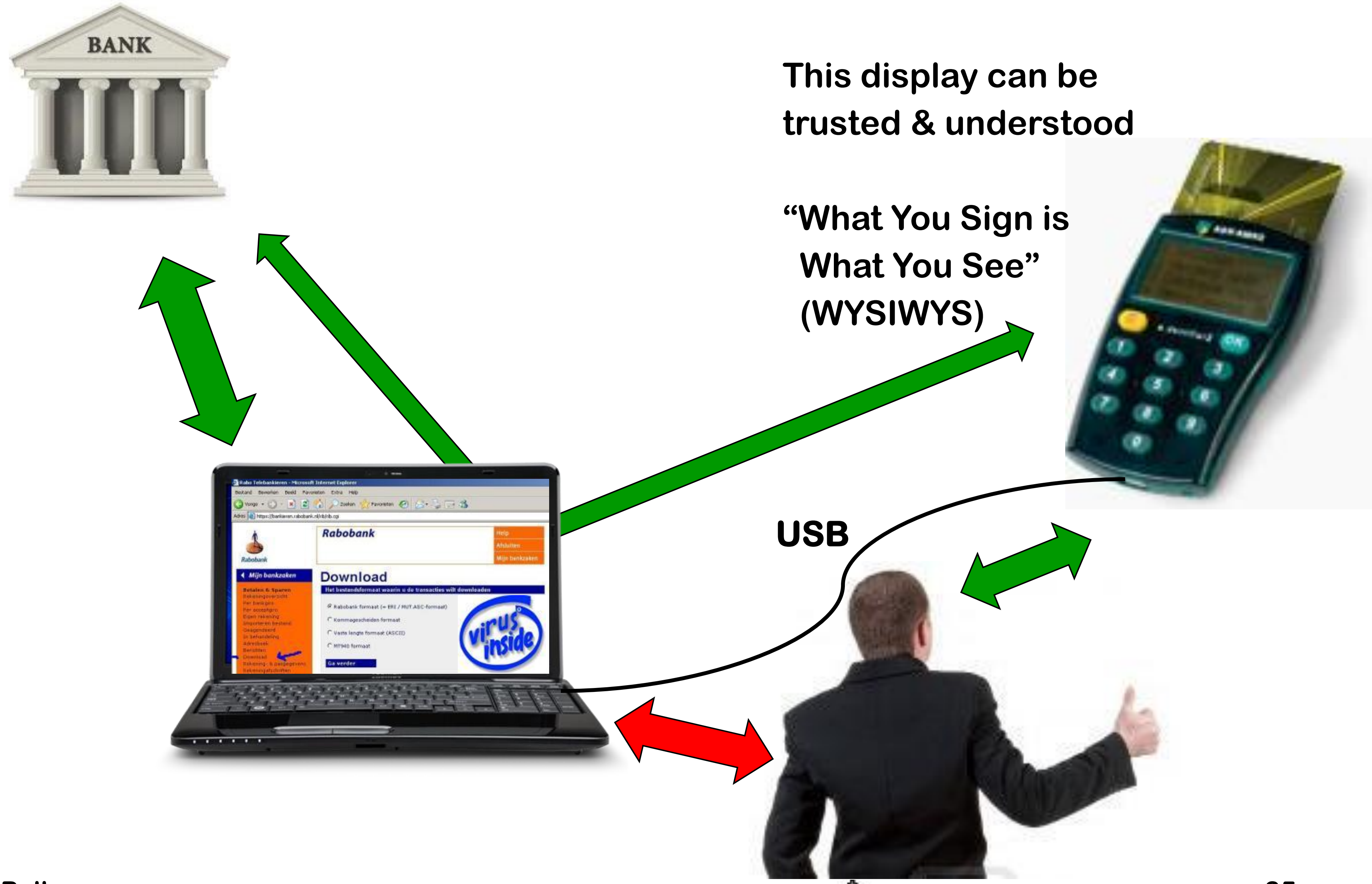
e-banking using EMV-CAP



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

Computer display of
cannot be trusted
(despite 🔒)

e-banking using USB-connected e.dentifier



This display can be trusted & understood

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

USB

Flaw in USB-connected e.dentifier2

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

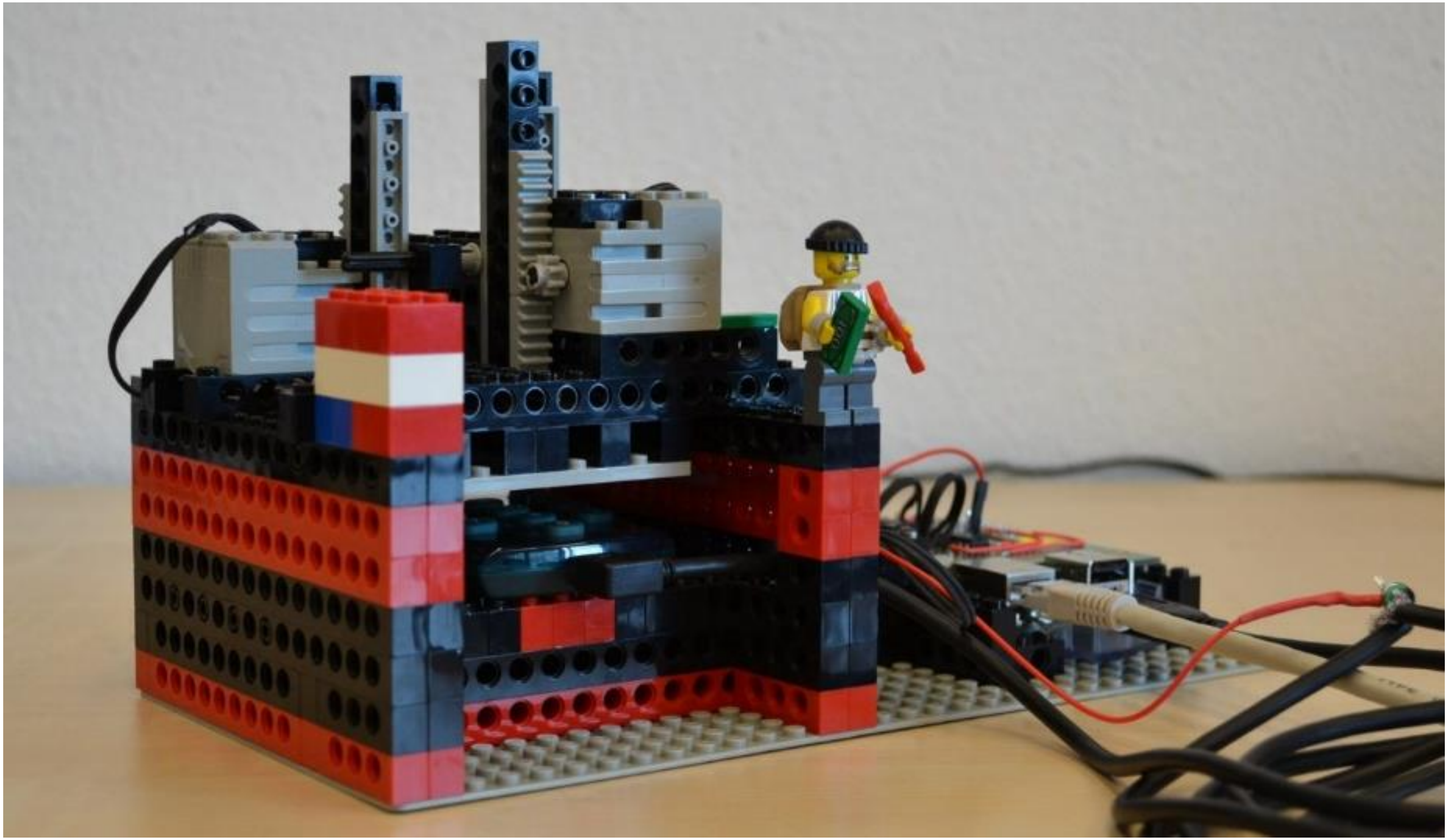
So malware on an infected PC could change all the transaction details and press OK!

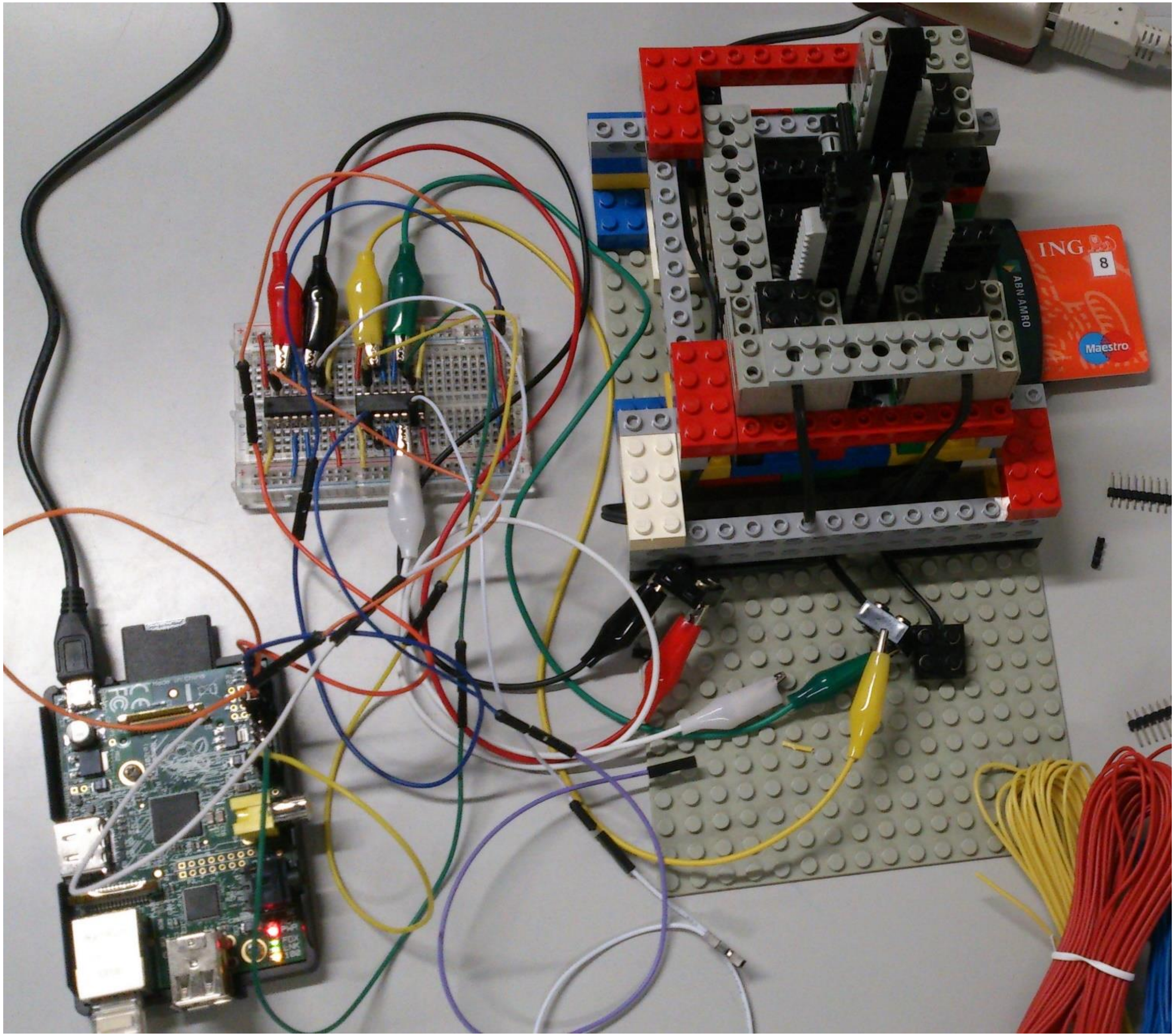
Flaw found with manual analysis.

Could we automate this?

[Arjan Blom et al., *Designed to Fail: A USB-Connected Reader for Online Banking*, NordSec 2012]

Our Lego hacker

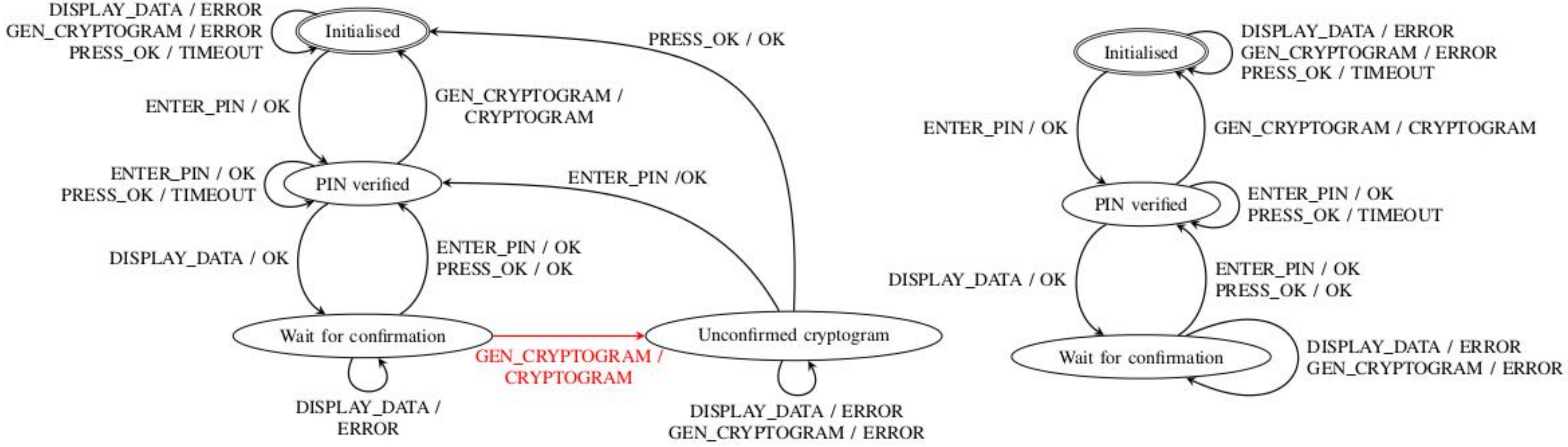



ING
8
ABN·AMRO
Maestro

Our Lego hacker

Automatic reverse engineering using Lego

State machines automatically inferred by our Lego robot

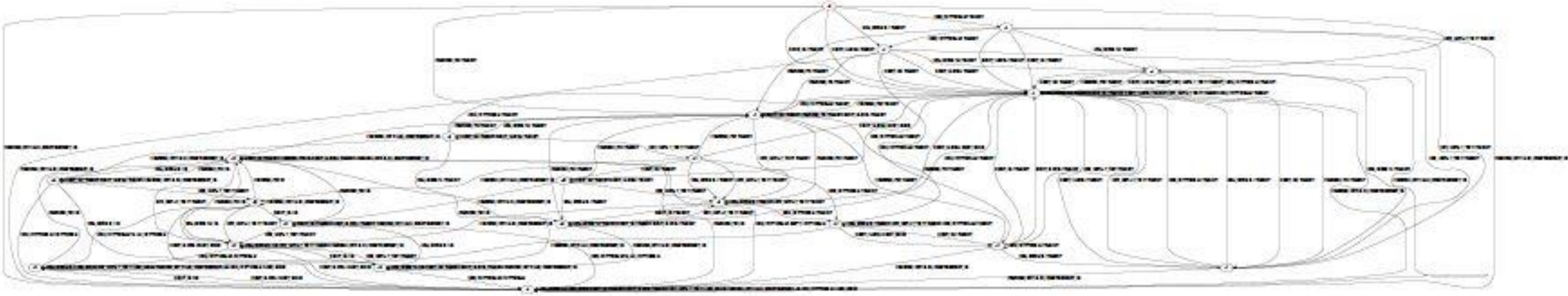


state machine of new device

state machine of old, flawed device

[Georg Chalupar et al., *Automatic reverse engineering using Lego*, Workshop on Offensive Technologies, WOOT 2014]

Aaargh!



full state machine inferred for new, fixed e.dentifier2

Do you think the designer of this protocol
and the person who implemented it
are confident that it is secure?

Conclusions

Conclusions

- Banking products are not always as secure as you would hope or expect.
- Not so clear who is taking responsibility for checking them.
  - MasterCard and Visa? EMVCo? Individual banks? Their suppliers? The Dutch or European Central Bank?
- Trend: from prevention to better detection & quick reaction.
- Complexity is bad!
- Assurance of security is hard!

  How to prevent design, programming & configuration flaws?

Conclusions

- Technical security flaws are not always serious risks.
  Criminals are very creative with 'low-tech' attacks.

  The real issue: can attacker find a good business model?

  - The bottleneck in internet banking fraud: recruiting money mules
  - Maybe ransomware is more lucrative?

- Banks are an interesting target for cybercriminals, BUT ...
  banks can measure fraud and use this to decide on improvements!
  For other organisations this can be much harder!