Formal analysis of EMV

Joeri de Ruiter

Digital Security, Radboud University Nijmegen

Overview

- What is EMV?
- How does EMV work?
- Known weaknesses
- Formal analysis of EMV

What is EMV?

Standard for communication between chip based payment cards and terminals

What is EMV?

Maintained by

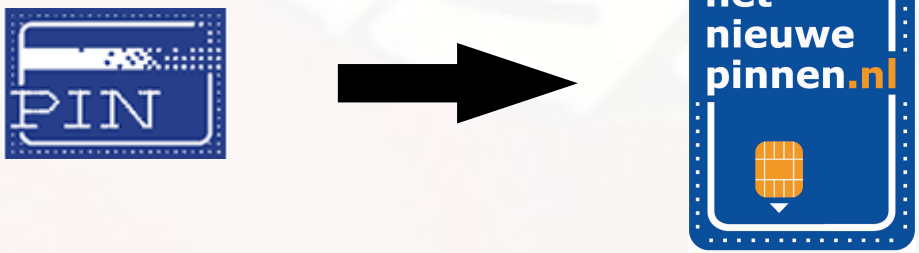
Owned by

What is EMV?

- Initiated in 1993
- Over 1 billion cards in circulation
- Compliance required for SEPA

EMV in the Netherlands

- Migration forwarded from 2013 to 2011
- Increased cost of skimming
  - 2007 15 M€
  - 2008 31 M€
  - 2009 38 M€

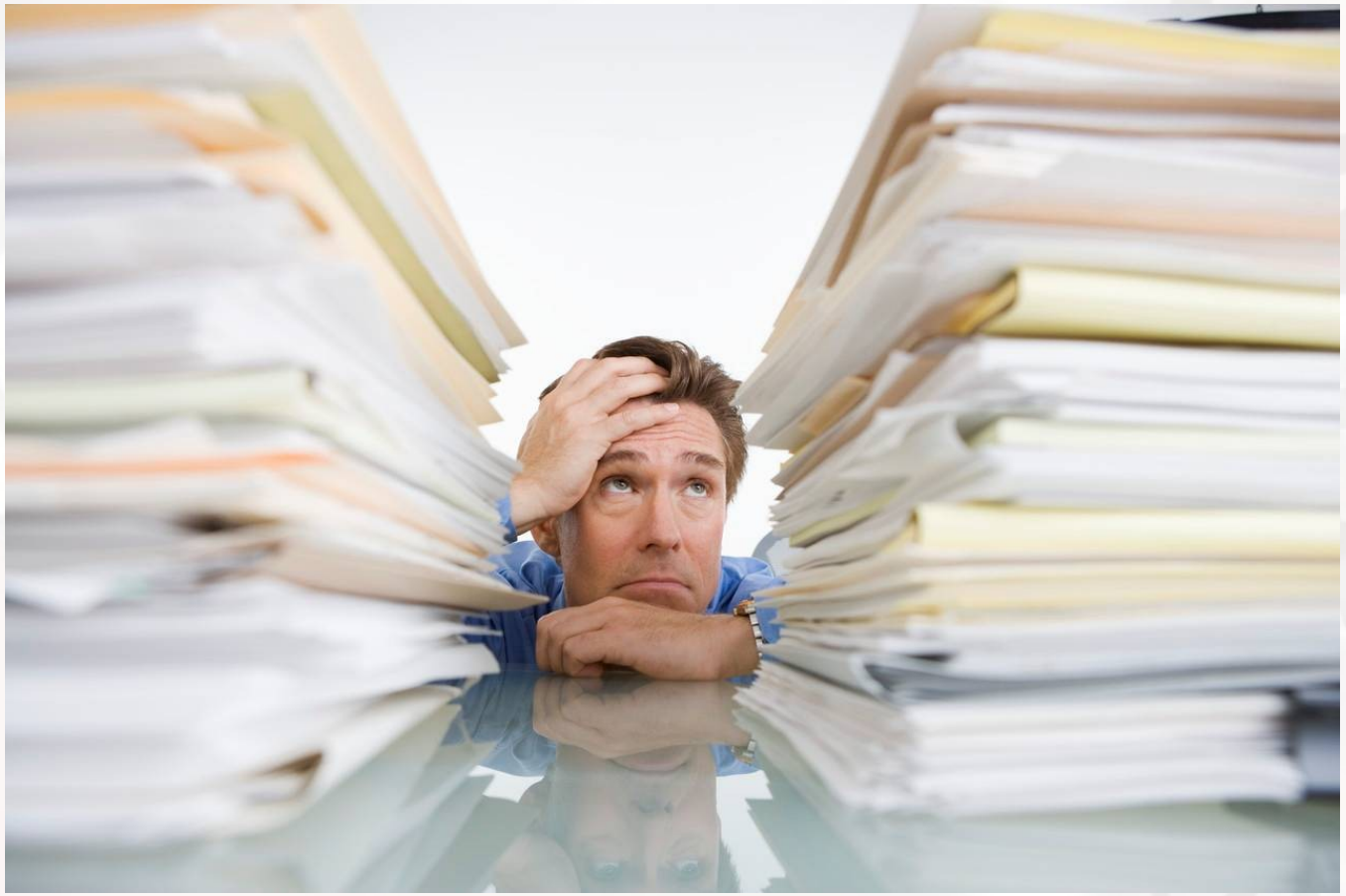


Why EMV?

- Reducing fraud by
  - skimming
  - stolen credit cards used with forged signatures
  - card-not-present fraud
- Liability shift

Complexity

- Over 700 pages



Complexity

- Many options and parameterisations
  - 3 card authentication methods
  - 5 cardholder authentication methods
  - 2 types of transactions
  - Parameterisation using Data Object Lists

Key set-up

- Card and issuer: symmetric key
- Issuer: private/public keypair
- Cards (optionally): private/public keypair

Protocol phases

- Initialisation
- Card authentication
- Cardholder verification
- Transaction

Initialisation

- Application is selected on smartcard
- Optionally information is provided by the terminal to the card
- Data from card is transmitted to the terminal

Card authentication

- Static Data Authentication
  - Static data on card signed by issuer
- Dynamic Data Authentication
  - Using asymmetric crypto
  - Challenge/response mechanism
- Combined Data Authentication
  - Transaction data signed

Cardholder verification

- PIN
  - Online: PIN is checked by the issuer
  - Offline: PIN is checked by the card
    - Unencrypted
    - Encrypted
- Handwritten signature
- None

Transaction

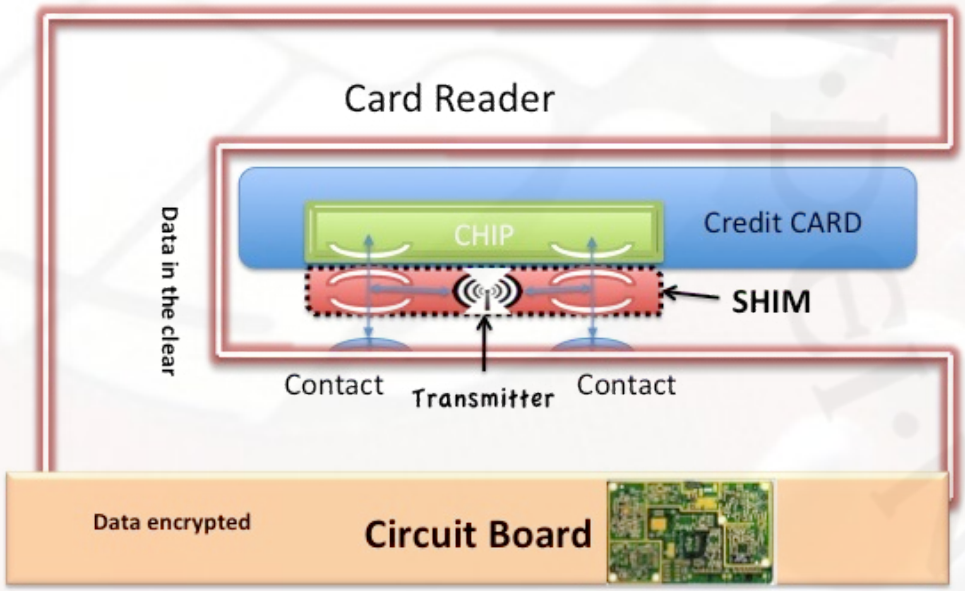
- Three different cryptograms
  - Transaction Certificate
    - Transaction approved
  - Authorisation Request Cryptogram
    - Online authorisation requested
  - Application Authentication Cryptogram
    - Transaction declined
- Contains an issuer specific MAC

Transaction

- Offline
  - Terminal request Transaction Certificate (TC)
  - Card response with TC or AAC
- Online
  - Terminal initiated
    - Terminal requests ARQC
    - Card replies with ARQC or AAC
  - Card initiated
    - Terminal requests TC
    - Card replies with ARQC

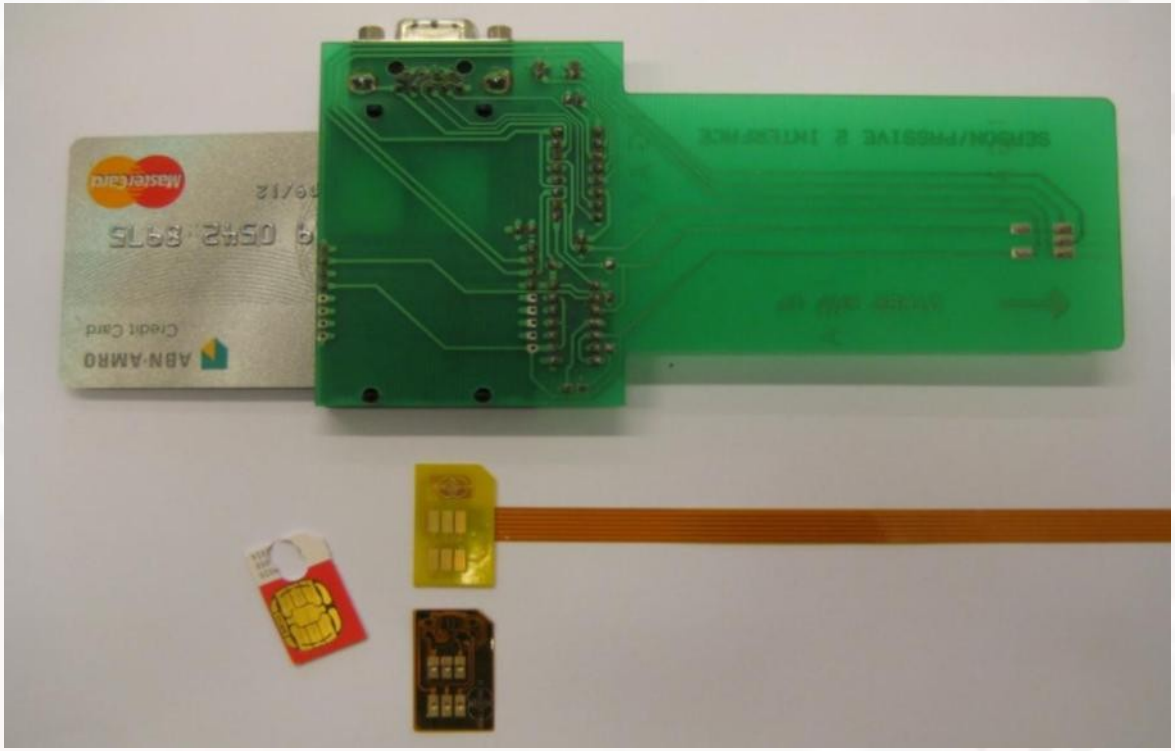
Attacking smartcards

- No direct copying possible
- Eavesdropping on communication using shim



Attacking smartcards

- Existing hardware used for pay TV and SIM cards



Attacking smartcards

- Active / wedge attacks
  - Modifying traffic between card and terminal
  - Card oriented
    - Acting as a relay for a genuine card
  - Terminal oriented
    - Abusing access to the card

Known weaknesses

- Cloning SDA card
  - Possible for offline transactions
  - Only static data authenticated
  - Card unable to sign data
  - All PIN codes accepted by card

Known weaknesses

- DDA wedge attack
  - Possible for offline transactions if DDA is used
  - Terminal cannot verify transaction data
  - Transaction not tied to card authentication

Known weaknesses

- “Chip & PIN is broken” [Murdoch et al. 2010]
  - Possible with both online and offline transactions
    - If card is not reported stolen
    - If PIN-less transactions are allowed
  - Using wedge attack
  - All PIN codes accepted

Formal analysis

- Verified using ProVerif
  - Automated security protocol verifier
  - Applied pi-calculus

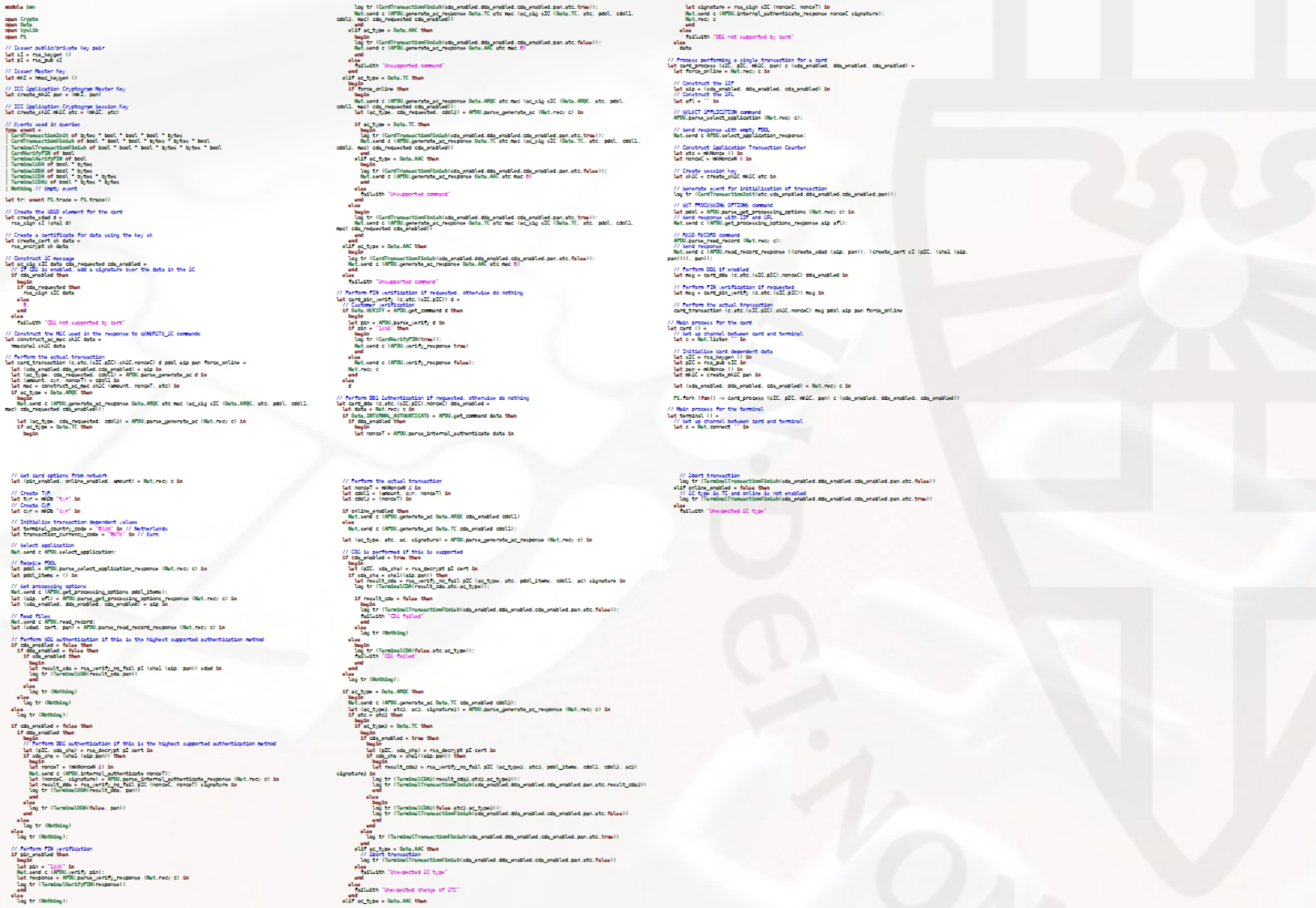
Formal analysis

- Formalisation in F#
  - Functional programming language
  - Developed by Microsoft Research
  - Executable code
  - Translated to applied pi-calculus using FS2PV

Formalisation

- Card and terminal formalised
- Options can be either unspecified or fixed
- DOLs fixed for Dutch banking cards
- Over 2500 lines of applied pi-calculus

Formalisation



Formalisation

```
// Perform DDA Authentication if requested, otherwise do nothing
let card_dda (c, atc, (sIC,pIC), nonceC) 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 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
```

Security properties

- Sanity checks
- Secrecy of private keys
- Highest supported autentication method used
- Transaction authentication

Security properties

- Card and terminal agree whether PIN is entered correctly

  evinj:TerminalVerifyPIN(True)
  ==>
  evinj:CardVerifyPIN(True)

- Transaction are authentic

  evinj:TerminalTransactionFinish(sda,dda,cda,pan,atc,True)
  ==>
  evinj:CardTransactionFinish(sda2,dda2,cda2,pan,atc,True)

Results

- Reduction to 370 lines of F# code
  - Resulting in over 2500 lines of applied pi-calculus
- All known weaknesses found
- ProVerif was able to verify most queries

Results

- With model including issuer additional weakness found
  - When exactly following the specifications
  - Possible if type of cryptogram is not included in MAC
  - Recommended minimum set of data elements:
    - Terminal: amount, country, verification results, currency, date, transaction type, nonce
    - Card: Application Interchange Profile Application, transaction counter

Thanks for your attention!