Formal analysis of the EMV protocol suite

Joeri de Ruiter

Digital Security, Radboud University Nijmegen

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 Single Euro Payments Area

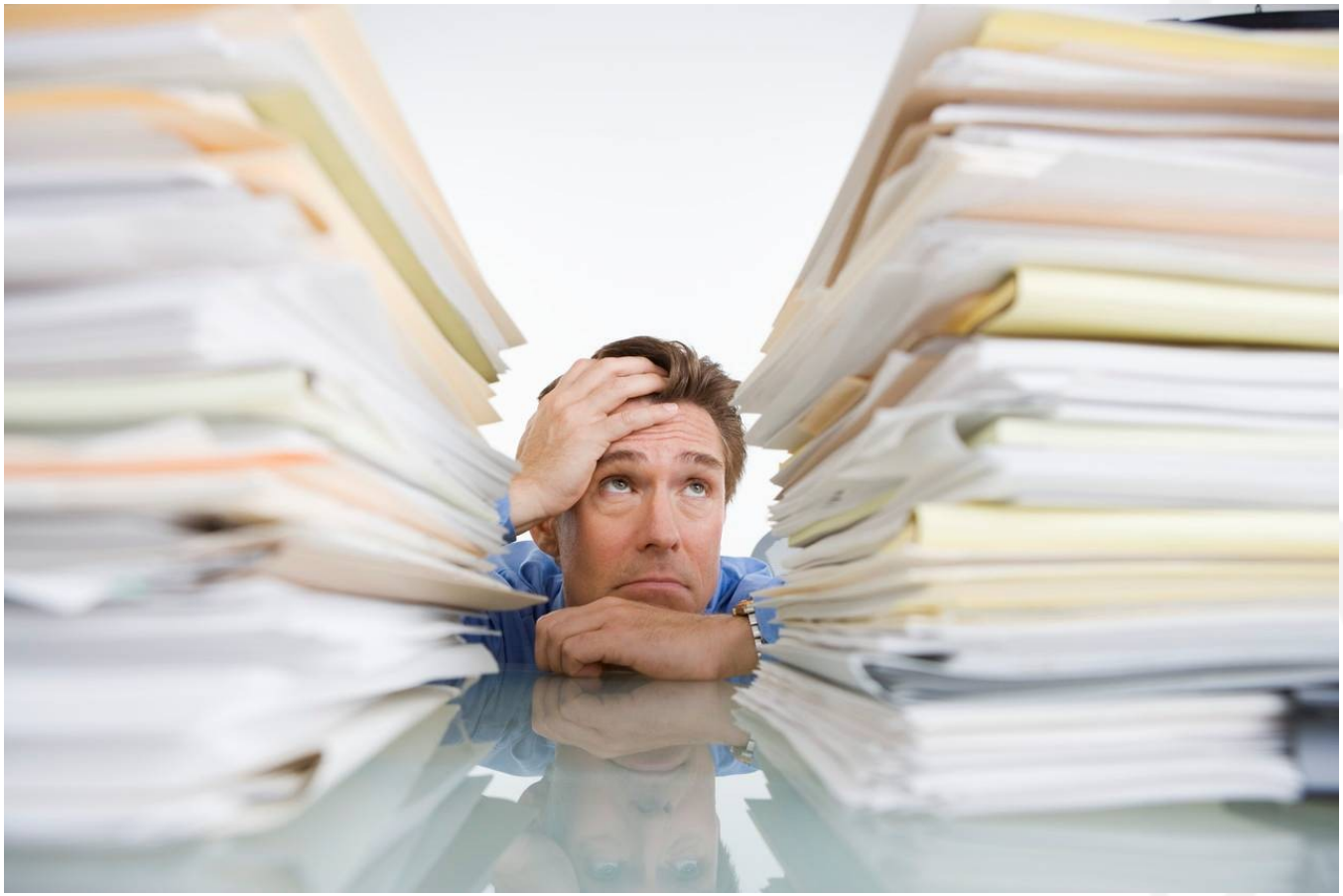
Why EMV?

- Reducing fraud
  - Skimming
  - Usage of stolen credit cards with forged signatures
  - Card-not-present fraud (EMV-CAP)

- Liability shift
  - Merchant: if no EMV is used
  - Customer: if PIN is used

Complexity

- Over 700 pages



Complexity

- Many options and parameterisations
  - 3 card authentication methods
    - SDA / DDA / CDA
  - 5 cardholder verification methods
    - PIN / Handwritten signature / None
  - 2 types of transactions
    - Online / offline
  - Parameterisation using Data Object Lists

Key set-up

- Card and issuer: symmetric key
  - For authentication of transactions
- Issuer: private/public keypair
  - For authentication of cards
- Cards (optional): private/public keypair
  - For authentications of cards/transactions

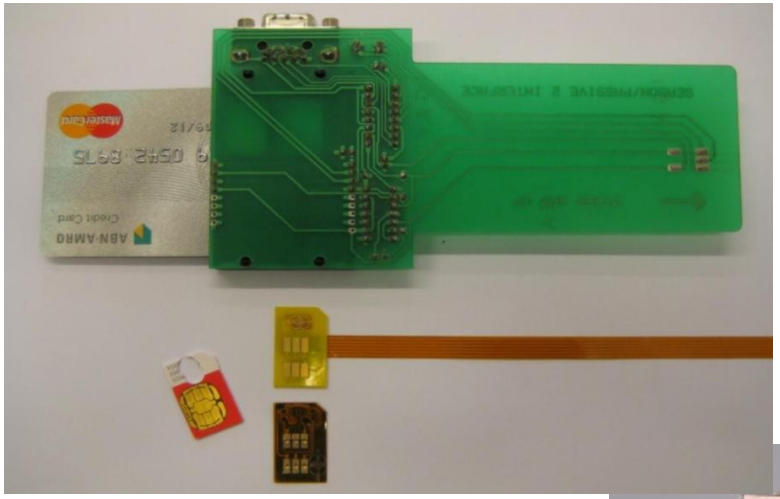
Card authentication

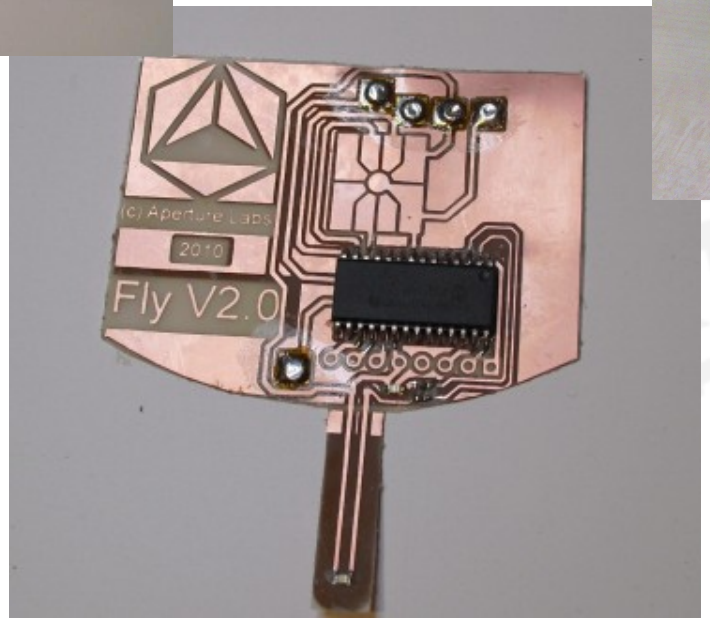
- Static Data Authentication (SDA)
  - Static data on card signed by issuer
- Dynamic Data Authentication (DDA)
  - Using asymmetric crypto
  - Challenge/response mechanism
- Combined Data Authentication (CDA)
  - Transaction data signed

Attacking smartcards

- No direct copying possible
- Eavesdropping on communication
  - Existing hardware used for pay TV and SIM cards
- Active / wedge attacks
  - Modifying traffic between card and terminal
  - Developed our own hardware

Attacking smartcards





Known weaknesses

- Cloning SDA card
  - Possible to perform offline transactions
- DDA wedge attack
  - Possible to perform offline transactions
- “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

Formal verification - ProVerif

- Automated cryptographic protocol verifier
- Active / passive attacker
- Cryptography considered to be perfect
- Unbounded number of sessions
  - Due to some approximations
  - False attacks possible
  - Claimed properties always hold

Formal verification - ProVerif

- Input in applied pi-calculus
- Queries
  - Secrecy

```
query attacker:M.
```

  - Authenticity

```
query ev:End ==> ev:Begin.
query evinj:End ==> evinj:Begin.
```

- Attack reconstruction

F#

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

Formalisation

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

Formalisation

Formalisation

```
// Card initialisation
let sIC = rsa_keygen () in
let pIC = rsa_pub sIC in
let pan = mkNonce () in
let mkAC = create_mkAC pan in

let (sda_enabled, dda_enabled, cda_enabled) = Net.recv c in

card_process (sIC, pIC, mkAC, pan) c (sda_enabled, dda_enabled,
    cda_enabled))
```

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 authentication 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(sda,dda,cda,pan,atc,True)

Results

- Reduction from 700 pages to 370 lines
  - Resulting in over 2500 lines of applied pi-calculus
- ProVerif was still able to verify our queries
- All known weaknesses found
- Existing tools mature enough to be used on industry standards

Results

- Reduction from 700 pages to 370 lines
  - Resulting in over 2500 lines of applied pi-calculus
- ProVerif was still able to verify our queries
- All known weaknesses found
- Existing tools mature enough to be used on industry standards

Thanks for your attention!