Formal analysis of the EMV protocol suite

Joeri de Ruiter and Erik Poll

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

EMVCo

Owned by

American Express, MasterCard Worldwide, JCB, VISA

What is EMV?

- Initiated in 1993
- Over 1 billion cards in circulation
- Compliance required for Single Euro Payments Area (SEPA)

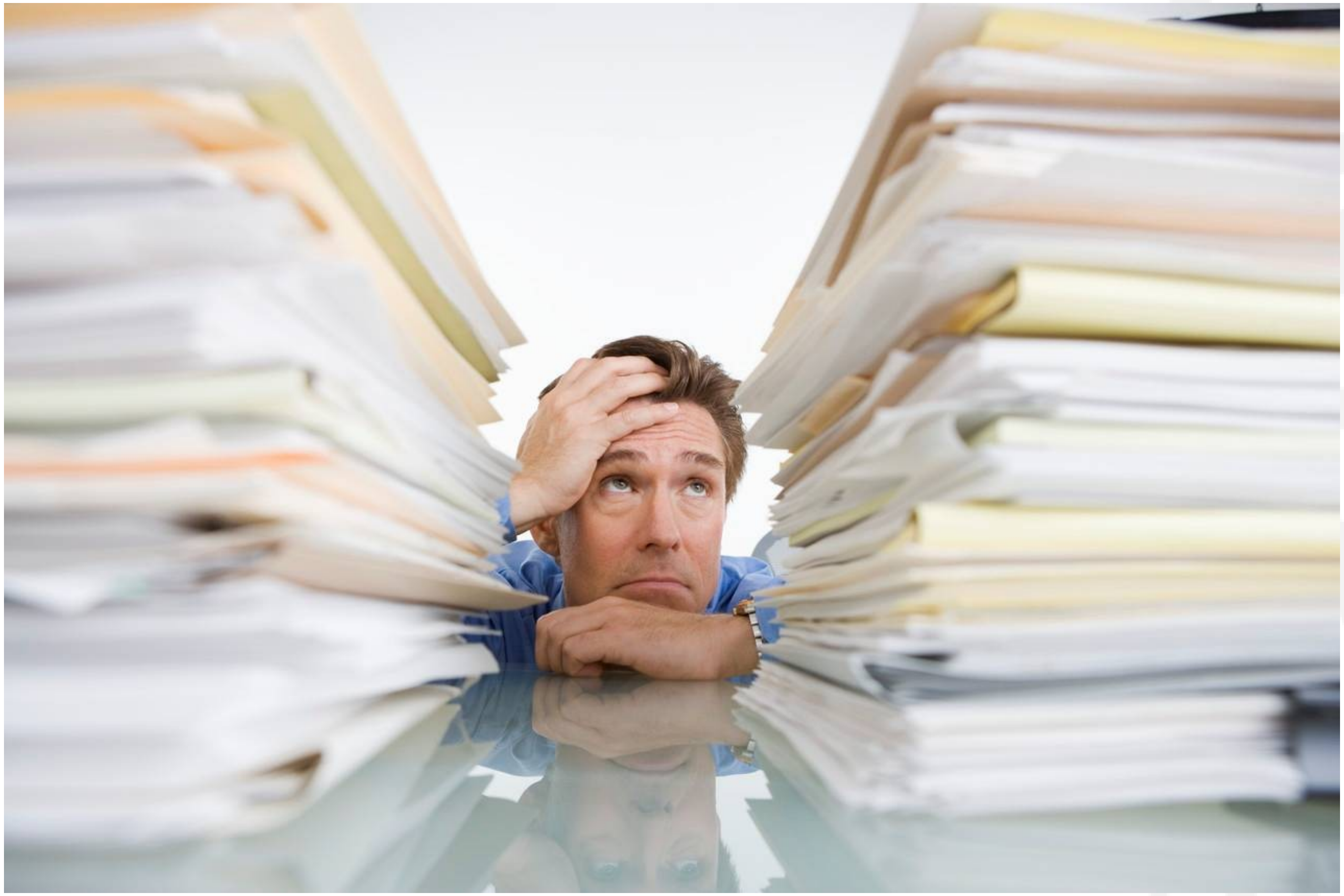
Why EMV?

- Reducing fraud by
  - skimming
  - stolen credit cards used 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
  - 5 cardholder authentication methods
  - 2 types of transactions
  - Parameterisation using Data Object Lists (DOL)

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 (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

Cardholder verification

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

Transaction

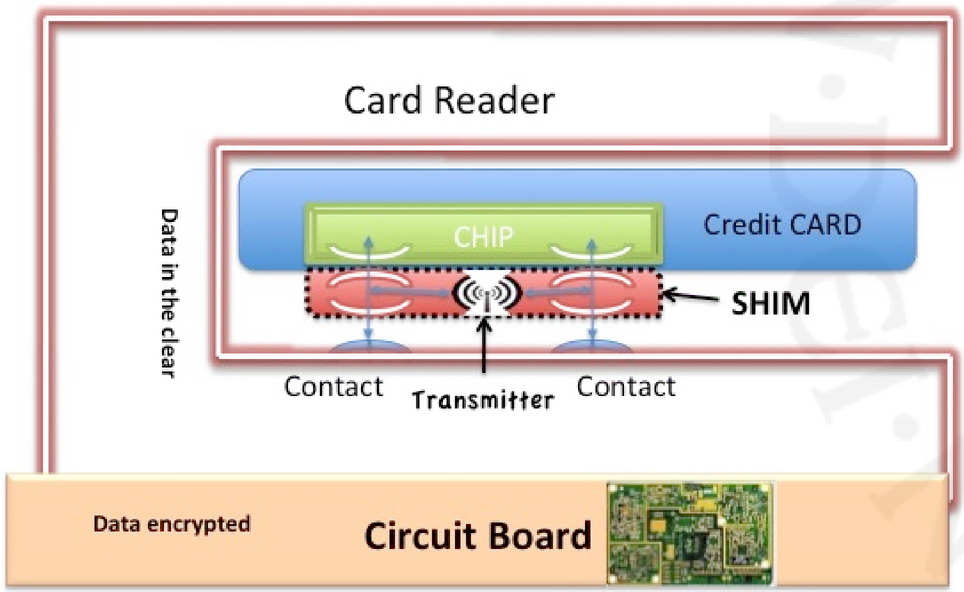
- Three different cryptograms
  - Transaction Certificate (TC)
    - Transaction approved
  - Authorisation Request Cryptogram (ARQC)
    - Online authorisation requested
  - Application Authentication Cryptogram (AAC)
    - Transaction declined
- Contains an issuer specific MAC

Transaction

- Offline
  - Terminal request 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

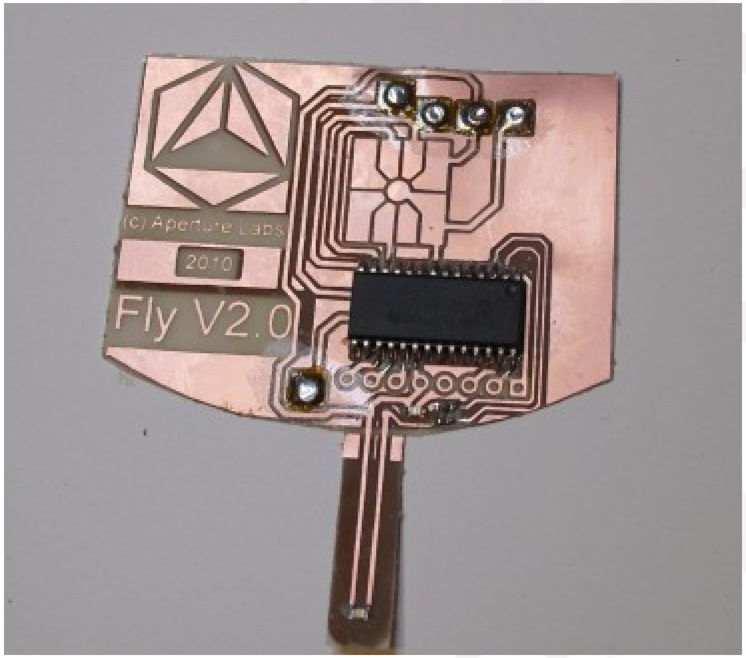
Attacking smartcards

- No direct copying possible
- Eavesdropping on communication using shim



Attacking smartcards

- Active / wedge attacks
  - Modifying traffic between card and terminal
  - Targeted against
    - Terminal
    - Card



Known weaknesses

- Cloning SDA card
  - Possible for offline transactions
  - All PIN codes accepted by clone
- DDA wedge attack
  - Possible for offline transactions
  - Transaction not tied to card authentication
- “Chip & PIN is broken” [Murdoch et al. 2010]
  - Possible with both online and offline transactions

Formal analysis

- Verified using ProVerif
  - Applied pi-calculus
  - Unlimited number of sessions

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
- 370 lines of F# code

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 agreement

Security properties

- Card and terminal agree whether PIN is entered correctly

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

- Card and terminal agree on transaction

  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
- ProVerif was still able to verify our queries
- All known weaknesses found

Results

- With model including issuer additional weakness found
  - When exactly following the specifications
  - Possible if type of cryptogram is not included in MAC
  - Spec. 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!