Formal analysis of EMV

Erik Poll
Joeri de Ruiter

Digital Security group, Radboud University Nijmegen
Overview

• The EMV standard
• Known issues with EMV
• Formalisation of the EMV standard in F#
• Formal analysis using FS2PV and ProVerif
EMV

- Started 1993 by EuroPay, MasterCard, Visa
- Common standard for communication between
  1. smartcard chip in bank or credit card (aka ICC)
  2. terminal (POS or ATM)
  3. issuer back-end
- Specs controlled by EMVCo, which is owned by MasterCard, Visa, American Express, JCB
- Over 1 billion cards in use
- EMV-compliance required for Single Euro Payment Area
Why EMV?

• **Goal:** reducing fraud by
  1. skimming
  2. stolen credit cards used with forged signatures
  3. card-not-present fraud (EMV-CAP)

• And also **some transfer of liability?**
Does EMV reduce skimming?

- UK introduced EMV in 2006

<table>
<thead>
<tr>
<th></th>
<th>2005</th>
<th>2006</th>
<th>2007</th>
<th>2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>domestic</td>
<td>79</td>
<td>46</td>
<td>31</td>
<td>36</td>
</tr>
<tr>
<td>foreign</td>
<td>18</td>
<td>53</td>
<td>113</td>
<td>134</td>
</tr>
</tbody>
</table>

- Magstripe can still be cloned and used in countries that don’t use the chip (notably USA)

  - Worse still: chip provides the Track 2 magstripe data
  - There are now moves to remove this ‘feature’
Man-in-the-Middle attacks

- using a **shim**
  possibly invisible inside terminal
- eavesdropping or modifying traffic

---

**old-fashioned version**
(mainly used for hacking pay TV)

**newer, thin versions**
(used for studying SIM locking)
The EMV protocol suite

- EMV is not a protocol, but a "protocol toolkit suite":
  - many options and parameterisations (incl. proprietary ones)
- 3 different card authentication mechanisms
  - SDA, DDA, CDA
- 5 different cardholder verification mechanisms
  - online PIN, offline plaintext PIN, offline encrypted PIN, handwritten signature, no card holder verification
- 2 types of transactions: offline, online

All these mechanisms again parameterised by Data Object Lists (DOLs)

- Specification public but very complex (>750 pages)
EMV basics: key set-up

- Card and issuer have **shared symmetric key**
  - which the terminal does **not** have
- Issuer has **private/public keypair**, used to sign data
  - which the terminal can verify
- Some cards have a **private/public keypair**, used to sign data
  - which the terminal can verify
EMV basics: parameterisation using DOLs

• **Data Object Lists** specify a list of **data elements** and their **formats**
  • eg date, country, amount, primary account number (pan), application transaction counter (atc), card/terminal generated nonce, ...

• **Card contains several DOLs** that specify
  • inputs required by the card
  • (signed/MACed) output produced by the card at some protocol step

Eg **CDOL** specifies which data is signed in a transaction cryptogram
EMV protocol phases

I. **Initialisation**
   Terminal reads some data from the card, incl. several DOLs

I. **Card Authentication**

II. **Cardholder Verification** (optional)

III. **Transaction**
II. Card Authentication: SDA

1. SDA – Static Data Authentication
   - card present static data (card no, expiry date etc) signed by issuer
   - problem: can be replayed, so card can be cloned
     - clone will always say offline PIN check succeeded
     - hence: offline terminal can be fooled
   - transaction is signed (MACed) using symmetric key, but terminal cannot check this MAC
   - issuer will spot this fraud later
II. Card Authentication: DDA

1. SDA – Static Data Authentication

2. DDA – Dynamic Data Authentication

   • card has (Pub,Priv) keypair and does challenge-response
   • requires more expensive card than SDA: one that can do asymmetric crypto
   • problem: card authenticated, but not the transaction
   • hence: offline terminal can still be fooled
   • issuer will spot fraud later
II. Card Authentication: CDA

1. SDA – Static Data Authentication
2. DDA – Dynamic Data Authentication
3. CDA – Combined Data Authentication
   - card has (Pub,Priv) keypair, as in DDA
   - signature now added over all the transaction data
   - so even an offline terminal can check authenticity
II. Card Authentication

1. SDA – Static Data Authentication
2. DDA – Dynamic Data Authentication
3. CDA – Combined Data Authentication

- Most cards in use are SDA or DDA
- SDA is being phased out
  - eg Visa & Mastercard forbid issuance of offline capable SDA cards starting 1/1/2011
- Nobody seems to be phasing in CDA cards yet...
III. Cardholder Verification Mechanisms

1. PIN
   a. online: PIN checked by the issuer
   b. offline: PIN checked by the chip
      • unencrypted
         PIN could be eavesdropped using shim
      • encrypted
         requires a card that can do asymmetric crypto

1. Handwritten signature

2. Nothing

Note: only offline PIN involves the smartcard chip
One more weakness...

• Terminal can be fooled into thinking a transaction was with PIN, while card & issuer know it was without PIN
  • using a wedge aka Man-in-the-Middle attack
  • for online and offline transactions
  • root cause: terminal cannot authenticate response to offline pin verification

[Murdoch, Drimer, Anderson, Bond, “Chip & PIN is broken”, 2010]

• This allows a stolen card to be used without PIN, but only
  • as long as the card is not reported stolen (for online)
  • if issuer allows PIN-less transactions (as is case in UK)
  or... if the issuer misses the correct checks for this in the back-end
IV. Transaction

• For the transaction the card generates cryptograms i.e. data with a MAC, and for CDA-cards, also a digital signature

• For online transaction the card generates 2 cryptograms
  • first cryptogram (ARQC) forwarded to the bank for approval
  • second cryptogram (TC) confirming the transaction
    • only after the card receives approval by the bank

• For offline transaction the card just generates one TC cryptogram
  • A card may refuse an offline transaction, and force the terminal to go online
Complexity of the EMV specs

• Moral of the story: specs too complex to understand
  • long specs, split over 4 books
  • little discussion of security goals or design choices
  • little abstraction or modularity

• Eg why not build on a notion of session level integrity & confidentiality as in SSL/TLS?

• Who really takes responsibility for ensuring these specs are secure? EMVCo, credit card companies, or banks?
Can formal security analysis tools cope with EMV?

First attempt: formalising EMV in ProVerif

Horrible! If-statements in applied pi-calculus cause huge duplication

Second attempt: formalising EMV in F#

Much better! F# allows sequential if-statements & functions
Formalisation of EMV in F#
Formalisation of EMV in F#

- EMV can be formalised in 370 lines of F# code
  - including all options
    - SDA, DDA, CDA
    - any card holder verification mechanism
    - off/online transations
  
  Booleans parameters controlling these options can be left unspecified (to study all these options) or fixed (to consider just one)
  
- but remaining configuration (DOLs) has to be fixed
  - we use minimal assumptions on DOLs taken from Dutch bank/credit cards
  - hardcoded in the model, but could easily be changed
Part of EMV model: DDA

// 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
    end
  else
    failwith "DDA not supported by card"

  else
    data
Analysis of the F# model

• F# can be translated to pi calculus by FS2PV tool and then analysed using ProVerif

• Translation to pi calculus explodes things a bit
  • 370 lines of F# becomes 3 kloc of pi calculus

• But... ProVerif can still verify security properties
  • usually in minutes, but *this requires some care!*
Properties checked with ProVerif

1. sanity checks to ensure absence of deadlock
2. secrecy of private keys
3. highest supported card authentication method is used
   • eg no fallback to say SDA can be forced

1. ‘transaction security’: if a transaction is completed, then everyone agrees on the parameters (eg with/without pin, off/online, amount,…)

   query evinj:TerminalTransactionFinish(sda,dda,cda,pan,amount,…) 

   ==> evinj:CardTransactionInit(sda,dda,cda,pan,amount,…).

No new attacks found, but all existing weaknesses confirmed
Future work

• Including formal model of the issuer
  • we don’t know the configuration, so can only check EMV’s example configuration
• Using F7 instead of ProVerif for verification
  • F7 might give better /more predictable response time
• Making F# model executable
  • with helper functions that implement low-level smartcard interaction, the model could interact with real cards and terminals
    • gives high confidence that our model is correct
    • could be used for model-based testing?
Future work: EMV CAP?

- use EMV chip for internet banking or e-commerce
  - EMV CAP defined on top of EMV: an EMV-CAP session is an aborted EMV session
  - internet banking
    - Mastercard: CAP (Card Authentication Program)
    - Visa: DPA (Dynamic Passcode Authentication)
  - e-commerce
    - Mastercard: SecureCode
    - Visa: Verified by Visa

- CAP specs are secret but have been partially reverse-engineered
  - also some patents discuss EMV-CAP
Reverse engineering EMV-CAP
Conclusions

• EMV protocol suite is far too complicated
  • too many options, written down in confusing way
• Formalisation possible in F#
  • and result is comprehensible!
• Formal analysis using FS2PV & ProVerif reveals all known weaknesses

• The future of skimming
  • Will skimmers move to the USA?
    For skimming cards there, or using the data they skim here?
cross-channel possibilities

tampered CAP readers

traditional skimming

harvest data via eshopping?

use data for ebanking?