EMV Cards Trivium

Fast Way to Side Channel Experiments

Tomáš Rosa

crypto.hyperlink.cz

June 2010 Seminar lecture for Masaryk University, Faculty of Informatics

Remember…

- The one and only purpose of this lecture is to promote rigorous academic research of payment cards security.

Agenda

- Basic idea of experimental attacks
- EMV card transaction process essentials
- CAP/DPA overview
- Card profiling example
- Cryptography highlights

Foreword

- EMV is a Europay-MasterCard-VISA general chip payment cards application framework.
  - It is publicly available at www.emvco.org as EMV Book 1-4.
  - This presentation cannot substitute the Books. It shall rather serve as a quick reference guide for security researchers.
- Card associations, however, define their own non-public extensions of EMV.
  - Whenever possible, we are basing on the pure EMV.
  - If a particular out-of-EMV topic is presented, we are basing on publicly reachable sources only.
  - Publicly leaked documents [8] are used for examples only without being included into the rigorous citations.

Foreword

- This lecture assumes certain skills in side channel attacks on smartcards.
- Its main aim is to show on how to extend the experiments with “general” smartcards on the area of payments cards as well.
  - …and how to do it quite fast, despite facing the horrible obscurity of the whole chip payment cards area.
  - To do that, the profiling technique based on CAP/DPA readers is developed and presented hereafter.

Part ONE
Basic Idea of Experimental Attacks

PIN-less Computation

- It is still common belief that EMV cards allow cryptographic computations only after a valid (client) PIN is presented.
- This assumption is, however, FALSE.
  - PIN (off-line) verification is just a part of EMV application service called cardholder verification.
  - It is not linked to any authorization for the APDU processing itself.
  - *If the card was requiring the PIN to be always verified off-line, it would be merely useless for any real life payment application.*

Prime Target

- Regardless the PIN is entered or not, the GENERATE AC command is accessible.
  - Assuming the whole application is not blocked.
- This APDU forces certain CBC-MAC computation using the card master key $\boldsymbol{MK}_{\mathrm{AC}}$ or its ephemeral derivative.
  - $\boldsymbol{MK}_{\mathrm{AC}}$ is a highly secure value protecting the online card authentication procedure.
- This is the very natural place where to start with side channel experiments.

Risk of MK_{AC} Disclosure

- The attacker could freely forge EMV card responses during transaction authorization.
  - For SDA-only cards, the complete duplicate of the card can be made.
  - Even for DDA or CDA cards, the attacker could still succeed with Man-In-The-Middle attack.
- In CAP/DPA application, the attacker could issue a valid transaction over the client banking account.

Practical Approach

- EMV card application is a very complex structure, hard to grasp at once.
- Strict focus on the particular experimental setup is necessary.
  - For a given card, derive a fixed APDU-flow sequence ending with (e.g.) GENERATE AC command.
  - Use a fixed cryptographic scheme and decide which part to attack at (session key derivation or CBC-MAC computation itself).
  - We call this essential phase as **card profiling**.

Part TWO
EMV Card Transaction Process Essentials

Foreword

- We describe the whole transaction flow solely from the chip card interaction viewpoint.
  - Since we are focused on APDUs only.
- Between the steps mentioned hereafter, several decision procedures usually occur.
  - For instance – terminal/card risk management, online authorization, etc.
  - EMV book 3 gives a concise overview of them.

TLV Everywhere & Anywhere

- The data model of the whole payment card application relies on ASN.1 tag-length-value encoding really heavily.
  - List of general tags can be found in EMV book 3 and [7].
  - If a template is used, tag-length fields can be omitted.
- TLVs can be spread out anywhere around the application, e.g.:
  - ATR history bytes
  - SELECT response template
  - Elementary data files (of course)
- Getting a picture of the card setup basically means catching various TLVs whenever you see them.
  - So, BE PREPARED!

#1: Application Selection

- Described in EMV book 1.
- Either direct approach based on a predefined AIDs list,
- or an indirect one based on the “1PAY.SYS.DDF01” file redirector.

Some AIDs

VISA Electron	A0 00 00 00 03 20 10
VISA debit/credit	A0 00 00 00 03 10 10
VISA DPA	A0 00 00 00 03 80 02
MasterCard	A0 00 00 00 04 10 10
Maestro	A0 00 00 00 04 30 60
MasterCard CAP	A0 00 00 00 04 80 02

#2: Get Processing Options

- CLA=8x (80), INS=A8 (GET PROCESSING OPTIONS)
- Starts the payment transaction.
  - Besides the others, Application Transaction Counter (ATC, tag 9F36) is incremented.
- Expects variable list of initial data (PDOL, tag 9F38).
  - The list is either default, or specified in application selection response, or even in ATR history bytes.
- Response includes Application File Locator (AFL, tag 94).
  - List of short elementary files identifiers and record numbers.

#3: Read Application Data

- CLA=0x (00), INS=B2 (READ RECORD)
- Reads the files and records listed in AFL obtained in step 2.
  - Data structure obeys ASN.1 based TLV syntax.
  - Recall, it is not so important which file the data comes from, it is the **tag** that decides.
  - So, you have to obediently read all those records listed in AFL while collecting the TLVs you get for their later processing.

#4: Data Authentication

- Static Data Authentication (SDA)
  - Simple digital signature verification (message recovery scheme).
  - AFL indicates which records are signed.
  - No APDU activity here.
- Dynamic Data Authentication (DDA)
  - Implicit static data signature verification (via public key certificate attribute) together with a challenge-response chip authentication.
  - INTERNAL AUTHENTICATE (CLA=0x (00), INS=88).
- Combined DDA/Application Cryptogram Generation (CDA)
  - Extended form of DDA.
- Potentially interesting for side channel attacks on asymmetric cryptography (DDA and CDA employ RSA computation).
  - We are, however, focused on the symmetric key $\boldsymbol{MK}_{AC}$ here.
- Further details are described in EMV books 2 and 3.

#5: Get Data

- CLA=8x (80), INS=CA
- Allows retrieving certain specific data which are not obtained during step 3.
  - ATC (tag 9F36)
  - Last Online ATC Register (tag 9F13)
  - PIN Try Counter (tag 9F17)
  - Log Format (tag 9F4F)

#6: Off-line PIN Verification

- CLA=0x (00), INS=20 (VERIFY)
- Optional, execution depends on the Cardholder Verification Method chosen.
- Expects PIN blob (possibly encrypted).
- Returns OK/FAIL (unprotected).
  - Successful attack presented in [6].

#7: 1st Application Cryptogram

- CLA=8x (80), INS=AE (GENERATE AC)
- Authenticates certain transaction processing data with $\boldsymbol{MK}_{AC}$ or its derivative.
- Input data are described in CDOL1 (tag 8C) field obtained during step 3.
- Response includes CBC-MAC together with some application specific data.
  - Needs to be profiled card-by-card.

AC Types

Type	Abbreviation	Meaning
Application Authentication Cryptogram	AAC	Transaction declined
Application Authorization Referral	AAR	Referral requested by the card
Authorization Request Cryptogram	ARQC	Online authorization requested
Transaction Certificate	TC	Transaction approved

Encoded in Cryptogram Information Data (CID, tag 9F27).

#8: Issuer Authentication

- CLA=0x (00), INS=82 (EXTERNAL AUTHENTICATE)
  - Can be also part of 2nd GENERATE AC command processing.
- Occurs in transactions authorized online.
  - Processes the Authorization Response Cryptogram (ARPC, proprietary tag).
- Uses symmetric key $\boldsymbol{MK}_{AC}$ or its derivative.
  - The eventual derivation has been usually made before (e.g. in step 7).

#9: Issuer Script Processing

- Various APDUs belonging to the issuer script batch.
  - Used for card (re)personalizations, counters update, PIN unblock/change, etc.
  - Security relies on a variant of Secure Message scheme according to ISO 7816.
- Pretending the ISP activity, the attacker could also gain useful side channel data here.
  - This approach is, however, considerably more complicated and beyond the scope of this introductory lecture.

#10: 2nd App. Cryptogram

- CLA=8x (80), INS=AE (GENERATE AC)
- Completes the whole transaction.
- The same computation as in step 7, the input data, however, are described in CDOL2 (tag 8D).
  - Allows the card to reflect results from online authorization procedure.
  - Finally allows or declines the whole transaction.
  - Should be performed even if we do not plan side channel measurement for it – just to calm down the card risk mgmt.
- No more GENERATE AC commands allowed for the particular transaction.
  - Another transaction must be started…

Part THREE
CAP/DPA Overview

CAP/DPA Technology

- Allows using the existing payment card infrastructure for authentication of clients and their payment orders.
  - MasterCard – Chip Authentication Program (CAP)
  - VISA – Dynamic Passcode Authentication (DPA)

Based on EMV Framework

- For CAP/DPA, we use mainly:
  - off-line PIN verification via VERIFY cmd.,
  - online card authentication via GENERATE AC cmd.
- Other services play more or less insignificant roles here.
  - SDA, DDA, CDA, etc.

Application Topology

- CAP/DPA allows
  - either sharing exactly the same application for payment transactions as well as for authentication,
  - or installing a separate application (stub) that shares only a certain data objects.
- Usually, the concept of two separate applications is applied.
  - Shared: off-line PIN, PAN (tag 5A), etc.
  - Independent: ATC (tag 9F36), AIP (tag 82), $\boldsymbol{MK}_{AC}$, file locators, etc.

Security Cornerstone

- The CBC-MAC computed by GENERATE AC is transformed (decimated) using the Issuer Proprietary Bitmap (IPB, tag 9F56) vector.
  - The result is displayed to the client using a numerical (or some more general) alphabet. It is then used as a kind of one-time password at the bank.
  - The Card Verification Result (CVR, proprietary tag) flags (part of CBC-MAC input data) must indicate a successful off-line PIN verification. This is an implicit countermeasure against [6].
- Recall that the CBC-MAC is computed using $\boldsymbol{MK}_{AC}$ or its derivative.
  - $\boldsymbol{MK}_{AC}$ is therefore the cornerstone of the CAP/DPA security.

Public Information

- Further public information on CAP/DPA can be found in [1].
  - Presents certain reverse engineering of CPA/DPA protocols together with some comments on how (not) to design real life applications.

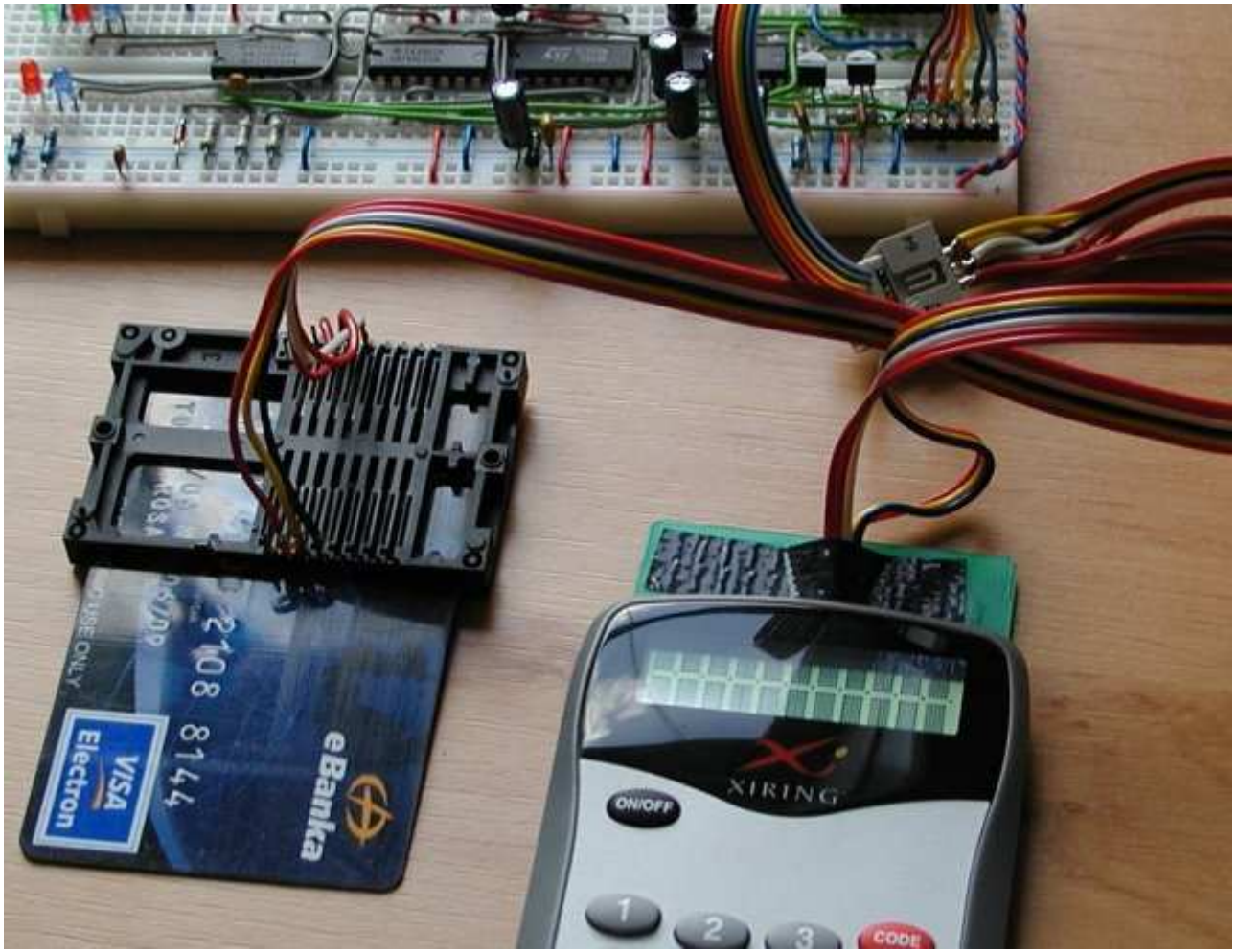
CAP/DPA Readers

- Besides their original purpose, these device can also serve the role of profiling etalons.
  - Spying the reader action helps a lot to understand all those practical aspects of EMV.
  - If necessary, a MITM technique can be used to force the reader to work with the real payment application instead of the CAP/DPA stub. Simply say “not present” for the CAP/DPA AID.

CAP/DPA *Teachers*

- Confused about the new card?
  - Just take the CAP/DPA reader and let it show you the way…
- Especially, the CDOL1/2 filling for the 1st/2nd GENERATE AC commands deserves certain attention.
  - The encoding of unused fields (and almost everything is unused from the CAP/DPA viewpoint) is such that it results to a (nearly) deterministic card risk management.
  - In turn, this allows to stabilize the CBC-MAC computation inputs as much as possible.

Learning the Lessons…



Part FOUR
Card Profiling Example

Lesson One

- Use CAP/DPA reader to show us something…
- By spying the initial communication, we can see immediately:
  - ISO 7816-4 protocol used (e.g. T=1),
  - AID of the application selected (e.g. A0 00 00 00 03 80 02).

Lesson Two

- Make the CAP/DPA reader to perform some very simple action.
  - For instance, one-time password generation.
  - If PIN is unknown, use [6] to fool the reader. *This, however, fools the reader only. Such a trick would not work in a proper banking CAP/DPA system.*
- Now, we can mainly see:
  - PDOL (if any non-default is used),
  - Application Interchange Profile (AIP) returned by GET PROCESSING OPTIONS,
  - CDOL1 (tag 8C) and CDOL2 (tag 8D) lists returned somewhere during step 3 – Read Application Data,
  - 1st and 2nd invocation of GENERATE AC command.

Example Values

- PDOL is missing (use default)
- AIP = 10 00
- CDOL1 = 9F 02 06, 9F 03 06, 9F 1A 02, 95 05, 5F 2A 02, 9A 03, 9C 01, 9F 37 04
- CDOL2 = 8A 02, 9F 02 06, 9F 03 06, 9F 1A 02, 95 05, 5F 2A 02, 9A 03, 9C 01, 9F 37 04
  - Commas are used to separate different tag-length fields. They are, of course, not a part of the data read.

Example Values

- 1st GENERATE AC
  - P1 = 80 (ARQC type requested)
  - input = 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00 00 00 00 01 01 01 00 00 00 00 00
  - output = 80 12 80 00 12 15 A8 E5 5A E3 BB A0 EA 06 34 0A 03 A4 B0 00

Example Values

- 2nd GENERATE AC
  - P1 = 00 (AAC type requested)
  - input = 5A 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00 00 00 00 01 01 01 00 00 00 00 00
  - output = 80 12 00 00 12 9B 81 73 6C 02 93 6B 45 06 34 0A 03 25 B0 00

Lesson Three

- Parse the GENERATE AC inputs and outputs.
  - This serves as an inspiration on how to respond according to the particular CDOL1, CDOL2.
  - Of course, CDOL1/2 for CAP/DPA stub and EMV application may differ. This must be checked (cf. bellow). Anyway, the inspiration is still highly welcome.

1st GENERATE AC Input Data

CDOL1 item	Data name	Value
9F 02 06	Amount, Authorised	00 00 00 00 00 00
9F 03 06	Amount, Other	00 00 00 00 00 00
9F 1A 02	Terminal Country Code	00 00
95 05	Terminal Verification Results	80 00 00 00 00 00
5F 2A 02	Transaction Currency Code	00 00
9A 03	Transaction Date	01 01 01
9C 01	Transaction Type	00
9F 37 04	Unpredictable Number	00 00 00 00

1st GENERATE AC Output

- Tag-Length = **80 12** (format-1 template)
  - Cryptogram Information Data = **80** (ARQC type returned)
  - Application Transaction Counter = **00 12**
  - CBC-MAC = **15 A8 E5 5A E3 BB A0 EA**
  - Issuer Application Data (VISA proprietary)
    - Lenght = **06**
    - Derivation Key Index = **34**
    - Cryptogram Version Number = **0A**
    - Card Verification Result = **03 A4 B0 00**

2nd GENERATE AC Input Data

CDOL2 item	Data name	Value
8A 02	Authorisation Response Code	5A 33
9F 02 06	Amount, Authorised	00 00 00 00 00 00
9F 03 06	Amount, Other	00 00 00 00 00 00
9F 1A 02	Terminal Country Code	00 00
95 05	Terminal Verification Results	80 00 00 00 00 00
5F 2A 02	Transaction Currency Code	00 00
9A 03	Transaction Date	01 01 01
9C 01	Transaction Type	00
9F 37 04	Unpredictable Number	00 00 00 00

2nd GENERATE AC Output

- Tag-Length = **80 12** (format-1 template)
  - Cryptogram Information Data = **00** (AAC type returned)
  - Application Transaction Counter = **00 12**
  - CBC-MAC = **9B 81 73 6C 02 93 6B 45**
  - Issuer Application Data (VISA proprietary)
    - Lenght = **06**
    - Derivation Key Index = **34**
    - Cryptogram Version Number = **0A**
    - Card Verification Result = **03 25 B0 00**

Lesson Four

- We use an ordinary smartcard reader (probably through PC/SC API) to inspect the real-life payment application.
  - We can skip this lesson if the main target is the CAP/DPA stub itself instead.
  - We also skip this step, provided there was no CAP/DPA stub on the card, so we have been already working with the real-life payment application anyway.
- We simply use a different AID (corresponding to the real-life payment application) and check the following values for a possible difference:
  - PDOL
  - AIP
  - CDOL1
  - CDOL2
- To do that, we follow steps 1 to 3 described in part 2.
  - Do not forget to be inspired by the previous lessons as well.

Example Values

- PDOL' is missing (use default)
  - No difference – use the same GET PROCESSING OPTIONS invocation as we have seen before.
- AIP' = 5C 00
  - Different, but it does not matter too much.
  - We have only to keep this new value for further CBC-MAC computation simulation during side channel analysis.
- CDOL1' = 9F 02 06, 9F 03 06, 9F 1A 02, 95 05, 5F 2A 02, 9A 03, 9C 01, 9F 37 04
- CDOL2' = 8A 02, 9F 02 06, 9F 03 06, 9F 1A 02, 95 05, 5F 2A 02, 9A 03, 9C 01, 9F 37 04
  - Both are the same as before. Therefore, we can employ exactly the same GENERATE AC encoding as we have seen before.

Lesson Five

- We prepare the final profiled APDU-flow which will be used during side channel experiments.
- Since we know the card details already, we can omit the time-consuming data reading step.
- At minimum, we have to perform:
  - SELECT AID
  - GET PROCESSING OPTIONS
  - 1st GENERATE AC
  - 2nd GENERATE AC (optional but recommended)

Profiled APDU-flow Example

- Card found in PC/SC reader: Gemplus USB Smart Card Reader 0
- ATR: 3B E9 00 00 81 21 45 56 49 53 5F 49 4E 46 20 06 78
- Protocol selected: T=1
- > 00 A4 04 00 (Nc=07) A0 00 00 00 03 20 10 (Ne=100)
- < 6F 25 84 07 A0 00 00 00 03 20 10 A5 1A 50 0D 56 49 53 41 20 45 6C 65 63 74 72 6F 6E 5F 2D 08 73 6B 63 73 65 6E 64 65 90 00 / delay:103800 us
- > 80 A8 00 00 (Nc=02) 83 00 (Ne=100)
- < 80 0A 5C 00 08 01 05 00 10 01 02 01 90 00 / delay:79190.8 us
- > 80 AE 80 00 (Nc=1D) 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 (Ne=100)
- < 80 1D 80 00 0A 7B E8 14 40 22 53 6A 22 06 33 0A 03 A0 B8 00 0A 02 00 00 00 00 00 F5 A6 E0 3E 90 00 / delay:219967 us
- > 80 AE 00 00 (Nc=1F) 5A 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00 00 00 00 00 01 01 01 00 00 00 00 00 (Ne=100)
- < 80 12 00 00 0A 40 EF F4 C5 82 B9 76 31 06 33 0A 03 21 B8 00 90 00 / delay:210040 us

Exercise - Homework

- Parse the APDU-flow presented above using the approach demonstrated before.
  - Notice the new field returned at the end of 1st GENERATE AC response.
  - This is a proprietary element called Issuer Discretionary Data in this case.
  - It is protected by a kind of custom-build secure messaging scheme.
  - It is, however, not included in “our” CBC-MAC, so we can ignore it. At least for now.
  - Side channel explorations are also possible here but with a considerably deeper profiling phase.

Part FIVE
Cryptography Highlights

CBC-MAC Equations Used

- Having found the suitable APDU-flow, we have also to find the particular cryptographic scheme used by the GENERATE AC command.
  - This is identified by a particular indicator in the Issuer Application Data field returned.
  - We often have to “google” a bit to find these details.

Example

- In the example presented above, a cryptogram of type 0A (10 in decimal) is used.
  - It is based on a proprietary cryptographic scheme which is not included in the EMV core.
  - Fortunately, if we “google” a bit more, a description of this (simple) algorithm can be found on the internet.

Cryptogram No. 10

- Let $\boldsymbol{MK}_{AC} = \boldsymbol{K}_A \,||\, \boldsymbol{K}_B$, where
  - $\boldsymbol{MK}_{AC}$ is the card master key (fixed for the particular card) and
  - $\boldsymbol{K}_A$ is its 8-bytes-long leftmost part,
  - $\boldsymbol{K}_B$ is its 8-bytes-long rightmost part.
  - Furthermore, $\boldsymbol{K}_A$ and $\boldsymbol{K}_B$ are meant to be used as DES keys.

Cryptogram No. 10

- Let $\boldsymbol{M} = \boldsymbol{INPUT}_{\mathrm{TERM}} \,||\, \boldsymbol{INPUT}_{\mathrm{CARD}} \,||\, \boldsymbol{PAD}$, where
  - $\boldsymbol{INPUT}_{\mathrm{TERM}}$ is the whole input data block passed to the GENERATE AC command from the terminal,
  - $\boldsymbol{INPUT}_{\mathrm{CARD}} = \boldsymbol{AIP} \,||\, \boldsymbol{ATC} \,||\, \boldsymbol{CVR}$, where the respective fields are described above,
  - ***PAD*** is a zero bit padding of 0 to 63 bits, such that the length of ***M*** is an integral multiple of 8 B.

Example of ***M***

- Constructed for 1st GENERATE AC from the example profiled APDU-flow (cf. above).

Part name	**Plaintext string**
INPUT$_{\text{TERM}}$	00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00 00 00 00 01 01 01 00 00 00 00 00
INPUT$_{\text{CARD}}$	5C 00 00 0A 03 A0 B8 00
PAD	00 00 00

Cryptogram No. 10

- Denote $\boldsymbol{M} = \boldsymbol{M}_1 \,||\, \boldsymbol{M}_2 \,||\, \ldots \,||\, \boldsymbol{M}_r$,
  - where each $\boldsymbol{M}_i$ is an 8-byte-long block.
- Define $\boldsymbol{C}_i = E_{K\_A}(\boldsymbol{C}_{i\text{-}1} \oplus \boldsymbol{M}_i)$, $1 \le i \le r$, where
  - E_{K_A} is a (single)DES encryption using $\boldsymbol{K}_A$,
  - $\boldsymbol{C}_0$ = 00 ... 00.
- Define $\boldsymbol{MAC} = E_{K\_A}(D_{K\_B}(\boldsymbol{C}_r))$, where
  - D_{K_B} is a (single)DES decryption using $\boldsymbol{K}_B$.
- The ***MAC*** is the final result to be included into the GENERATE AC response.

Notes On Cryptogram No. 10

- There is no $\boldsymbol{MK}_{AC}$ derivation applied, the master key itself is used again and again…
  - This property is highly welcome from the side channel attacks exploration viewpoint.
- The second half of $\boldsymbol{MK}_{AC}$ ($\boldsymbol{K}_B$) is considerably less exposed than the first one ($\boldsymbol{K}_A$).
  - Therefore, the chance of getting $\boldsymbol{K}_B$ is somehow lower than the chance of getting $\boldsymbol{K}_A$.
  - On the other hand, once the attacker already knows $\boldsymbol{K}_A$, they can start trying to (ab)use the card as an oracle for the MAC final computation.
  - They employ the basic algebraic properties of CBC-MAC together with the possibility to affect $\boldsymbol{INPUT}_{TERM}$ while being able to closely estimate $\boldsymbol{INPUT}_{CARD}$.

Other Cryptograms Used

- Sometimes, the CBC-MAC computation described in the EMV book 2 is also used.
  - There is an online demo of the computation available at [7].
  - The main difference, from the side channel attacks viewpoint, is introduced by using a session key derivation function.
  - Basing on the ***ATC***, a fresh ephemeral derivative of $\boldsymbol{MK}_{AC}$ is used for 1st and 2nd GENERATE AC during each transaction.

Conclusion

- There is a lot of interesting side channel experiments, often leading to devastating attacks.
  - This, in turn, allows essential security improvements.
- Surprisingly, there are just a few academic papers touching the security of chip payment cards.
  - The reason is, perhaps, the horrible obscurity of this area.
- This lecture tries to clarify the most important topics.
  - Furthermore, the technique of using CAP/DPA readers for rapid card profiling was developed and presented here.
  - Should any attack be discovered it could only help to push the card associations to improve the security of these important devices accordingly.

Thank you for attention…

Tomáš Rosa

crypto.hyperlink.cz

References

1. Drimer, S., Murdoch S.-J., and Anderson, R.: *Optimised to Fail: Card Readers for Online Banking*, Financial Cryptography 2009, http://www.cl.cam.ac.uk/~sjm217/papers/fc09optimised.pdf
2. EMV Integrated Circuit Card Specification for Payment Systems, Book 1 – Application Independent ICC to Terminal Interface Requirements, version 4.2, June 2008
3. EMV Integrated Circuit Card Specification for Payment Systems, Book 2 – Security and Key Management, version 4.2, June 2008
4. EMV Integrated Circuit Card Specification for Payment Systems, Book 3 – Application Specification, version 4.2, June 2008
5. EMV Integrated Circuit Card Specification for Payment Systems, Book 4 – Cardholder, Attendant, and Acquirer Interface Requirements, version 4.2, June 2008
6. Murdoch S.-J., Drimer, S., Anderson, R., and Bond, M.: Chip and PIN is Broken, to appear at the 2010 IEEE Symposium on Security and Privacy, draft available at http://www.cl.cam.ac.uk/~sjm217/papers/oakland10chipbroken.pdf
7. http://www.emvlab.org/
8. Some “leaked” documents available e.g. through http://www.google.com.