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



Owned by



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

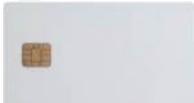
- Application is selected on smartcard
- Optionally information is provided by the terminal to the card
- Data from card is transmitted to the terminal

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

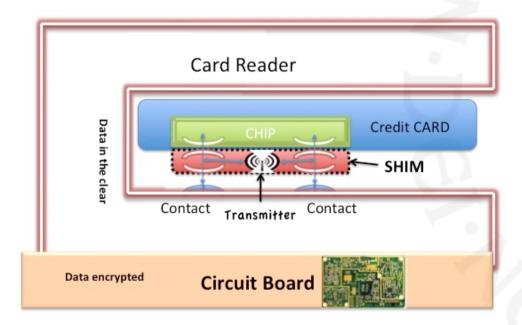
- Three different cryptograms
 - Transaction Certificate (TC)
 - Transaction approved
 - Authorisation Request Cryptogram (ARQC)
 - Online authorisation requested
 - Application Authentication Cryptogram (AAC)
 - Transaction declined
- Contains an issuer specific MAC

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

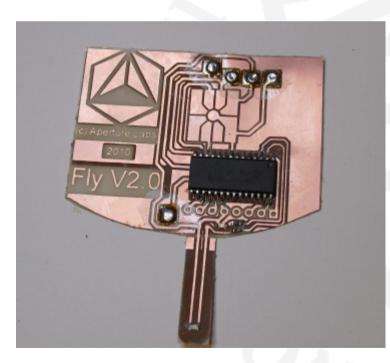
Attacking smartcards

- 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

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

and the line open Crypto open Deta open Sysilita // Issuer public/private key pair lat sI = rma_keygen () lat sI = rma_keygen ()

// Issuer Master Key Tat mk2 = hmac_heygen () // ICC opplication Cryptogram Mester Key let create_pHdC pan = imkl. pan)

// ICC opplication Cryptogram Session Key lat create_skic midC atc = (midC. atc)

Cardinates Close State of Bool Terminel Transection Finish of Cardinative StyrIDI of Bool Terminel WestfyrIDI of Bool Terminel SDA of Bool, * Bytes

Terminalized of bool " bytes Terminalized of bool " bytes Terminalized of bool " bytes " bytes Terminalized of bool " bytes " bytes Mothing // (mpty event let tr: event Pi_trace = Pi_trace()

// Create the SDD element for the card lat create_stad d = rsa_ctgr <I (shal d)

// Create a certificate for data using the key sk. Tait create cert sk data = rea_excrept sk data

/ Construct iC message at ac sig sIC data cds_requested cds_enabled = // IP CDL is embled, add a signature over the data in the UC SF cds enabled than

begin if cds_requested then rea_sion_sDC data fathatth "CDL not supported by card" // Construct the MCC used in the response to GIMENCTE_DC commands lat construct_uc_mac skDC data = hmacshal skDC data

Teache (a) and a set of transfer on a set of the set of

Let (sc_type, cd_ mounshed, cdol)) = ATGU,parse_generate_ac (Net.rec; c) is if sc_type = Data.H1 then

// Get card options from network. lat (pin_erabled. online_erabled. amount) = Net.recy c in // Create T(f) Tat tyr = miDb "tyr" in // Create C(f) Tat cyr = miDb "cyr" in

// Initialize transaction dependent values
lat terminal_country_code = "0100" in // Netherlands
lat transaction_currency_code = "0170" in // Euro

// Select application Net.send c ATOU.select_application: // Fecsive FOL
lat pdol = AFOL parse_select_application_response OMet.rec/ cl is
lat pdol_items = 0 is // ist processing options Net.eed c UMUL.get_processing_options pdol_ites;): Let (day, art) = AFUL.pars_get_processing_options_response (Net.rec; c) is Let (day, molect, day, multiplet, day, multiplet) = sig in

// feed files Netlerd c MTULread_record: 1-t (add. cert. pen) = MTULparse_read_record_response (Net.recy c) in

// Terform SDL sutherticution if this is the highest supported suthertication if db_enabled = fulse then if db_enabled = fulse then if db_enabled = fulse then bagin Tar result ofa = res_perity_nc_feil pl (shal (aip. pan)) edad in log tr (TarminalidAl/result_sda.pan)) lee
 log tr (Mothing)

log tr (Nothing log tr (Nothing) las log tr (Mething):

La d'elembaj: Tegunstof - Mitta De Status - Instance per de sit is de Status - New Status - Mitta De Status - Mitta - Mitta De Status - Mitta - Mitta - Mitta De Status - Mitta - Mitta - Mitta - Mitta - Mitta De Status - Mitta De Status - Mitta - Mitt

else log tr (Terminel200#(felae. pen)) end log tr (Nothing) else log tr (Hotbday): V Perform FDN vertification of pin_enabled then

negle let some (1970,...etty som): Let mesone = ATML.parm_letty_response (Net.recy c) in Log tr (TerninelWetfyTEM:response)) ed

tog tr (LardTransctineFlatAriat_enabled.doi_realist.doi_realist.doi.net.int (LardTransctineFlatAriat_realist.Tt atc mac (ac_is_ atC (Deta.Tt. atc. pott. cdoll. doil, act cdr.presstic (dr.yradSci)) #iif ac_type = lota.AC than hegin

begin log fr (CardTransectionFinishisda_enabled.dds_enabled.cds_enabled.pan.stc.felae)) MarLand c (MTDL.generate.jc_response Satu-AML atc mac f)

faluath "Unsupported command"

eff actions or appointer transmission eff act, type to have that the magnetic states that the set of the set

end elif ac_type = Data.AAC then

import log tr (CardTransactionFinish(sds_enabled.dds_enabled.cds_enabled.pan.stc.fslae)): Mrt.send c (MT00.generate_ac_response Data.AMC atc mac 0) else

Implie Top for (GardTransaction=Flociabied_armsbled.dds_prabled.dds_prabled_gar.stc.trms)): Bet.send c (BFUL_property_sc_response Bats.TC stc mac (sc_sig s2C (Bats.TC stc. pdc1. cdo12. mac) cds resulted cds setLed)

end end elif ac_type = Data.AAC then haven region log tr (CardTransactionFinishisdi_enabled.dds_enabled.cds_enabled.pan.atc.fslae)): Het.send c (HTDU.generste_ac_meptonse Data.AHL atc mac 0)

else failwith "Unwoosrted comment" / Perform FDN verification if requested, otherwise do nothing at card_pin_verify (c.stc. (s2L.p2D)) d = // Customer verification is "adv. WithFr = AFUL.pet_command d than

// Castmarr verification for back.Willspars.command of then bagin fart pin = ATMU.pars.perify of in ff pin = "likit then bagin (Gardwrify/ITMUtram)): back.send c (ATMU.perify_response tram) end

else Net_send c (APOU.yerify_response felse) Net. end else d

// Perform DDI Lettertionism if requested, otherwise do nothing let card das (c.stc. (dI_CC).nonck) day_emabled = let data = Netrony of a if day_emablement, WITHINTERT = ATTUL_per_commend data then if day_emable then heads let nonci = ATTUL_pers_intermal_softwarticate data in

lat cdcl = (anort c,r. nonceT) in lat cdcl = (anort c,r. nonceT) in

if online_enabled then _Nwit.send c (MTNU.generate_ac Data.ANOK cda_enabled cdcll) Net.send c (AMDU.generyte_ac Dyta.TC cds_enabled cdoll): let (ac type, atc. ac. signature) = APDU, parse generate ac response (Net.rec) c) is // CDL is performed if this is supported if can enabled = true then begin lat tall and activity ands at [plf. sd_chal = rss_decrypt pl cert is if de_cha = rhalisip.par)) then lat realt_cds = rss_erity.co_hil pl[(sc_type. stc. pdol_itees. cdoll. sc) signature is lag tr (featurellikinest_cd_stc.act.pps)): if realt_cds = false then begin log tr (TerminelTreneacti failwith "CD1 failwd" sda_enabled.dda_enabled.cda_enabled.pan.stc.falae) end else log tr (Nothing)

alse bagin Log tr (TerminalCDA(false.atc.ac_type)): fullwith "CDL failed" end else log tr (Nothing):

if ac type = Deta. ANDC then begin

bagli Mittard C (MT00.generata_ac Deta.TC cda_anabled cdtl): Tat (ac_tppe). atcl. acl. signature) = ATD0.parse_generata_ac_response (Nat.recy c) is 27 atc = atc) Then begin if ac_type2 = Deta.TC then

begin if cds_enabled = true then

bagin Lat (pCL udu_pha) = res_decrypt p2 cert in if udu_pha = shall(udp.par)) then bagin Lat result_code) = res_verity_no_fsil p2C (ac_type). atc). pdel_time. cdoll. cdoll. ac)) .at result_code) = res_verity_no_fsil p2C (ac_type). atc). pdel_time. cdoll. cdoll. ac) (ignature) in log tr (Tarmine UCMU: result_cis).stc).sc_tpe()): log tr (Tarmine UTransaction/Salahista_enabled.dds_enabled.cds_enabled.pan.stc. bagda Log tr (Ternston UDA)(false.stcl.sc_type))): Log tr (Ternston UTaxasctionSizia)(siz_enabled.dds_enabled.cds_enabled

elae Log tr (Termine)J end elif ac_type = Deta AAC then // Dort transaction loo tr (Termine\TransactionF

ctionFimiah(sds enabled.dds enabled.cds enabled.pan.stc.felse) fat with "Une-pected SC type" faluath "Unexpected change of UTC"

let signature = res_sign siC (renceC. renceT) is Net.eed c (MT00.internet_authenticate_response renceC signature) Net.rec, c

end elee failedth "DDL not supported by card" else data

// Process performing a single transaction for a card Let card process (SC, pIC, mAC, pan) c (sds_enabled, dds_enabled, cds_enabled) = Let force_polities = Net.net c in

// Construct the IIF lat sip = (sig_enabled, dig_enabled, cig_enabled) in // Construct the SR lat sit = " 4=

// SILECT SPELICITION command STEE manual select application (But many c); // Send response with empty FOL Netword c AFOLeelect_application_response

// Construct implication Transaction Counter
let atc = miNonce () in
let nonce(= miNonceN |) in

// Create session key let skiC = create_skiC mkiC atc in // Generate event for initialization of transaction
log tr (CardImanusctionInitiate edu arabled.dds enabled.cds enabled.pan));

// WIT FROCISSIDE OFTIONS command tait pdot. = AFR0.parse_pet_processing_options (Net.recy c) is // dend response with UT and UR. Met.send c (AFR0.pet_processing_options_response sip afG):

// Yild Nichowson (Journal Journal Journal of Street St

// Perform DOL if enabled lat mup = part dds (c.stc. (sDC.sDC).nonceC) dds enabled in // Ferform FDN verification if requested let mag = card_pin_verify (c.atc.(sDC.pDC)) mag in

// Ferform the actual transaction card transaction (c.stc.(sIC.pIC).skiC.nonceC) mag pdol sip pan force_online

// Hein process for the card lat card () = // Set up channel between card and terminal lat c = Het.listen " in

// Initialize card dependent data lat dIC = rea_beygen () in lat pIC = rea_pub dIC in lat pic = rea_pub dIC in lat midC = create_midC pan in let (sda_enabled, dda_enabled, cda_enabled) = Net.recy c is

Pi.fork (funi) -> card_process (sIC. pIC. midC. pan) c (sda_embled. dda_embled. cda_emb // Main process for the terminal Lat terminal () = // Set up channel between card and terminal Lat c = Mat.connect is

log tr transmitterine to the the second seco And Los

/ Lbort transaction

Joeri de Ruiter - Digital Security, Radboud University Nijmegen April 1, 2011

elif ac_type = Data.AAC then

22 / 29

// 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))

// 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!