Beveiliging van de OV-Chipkaart

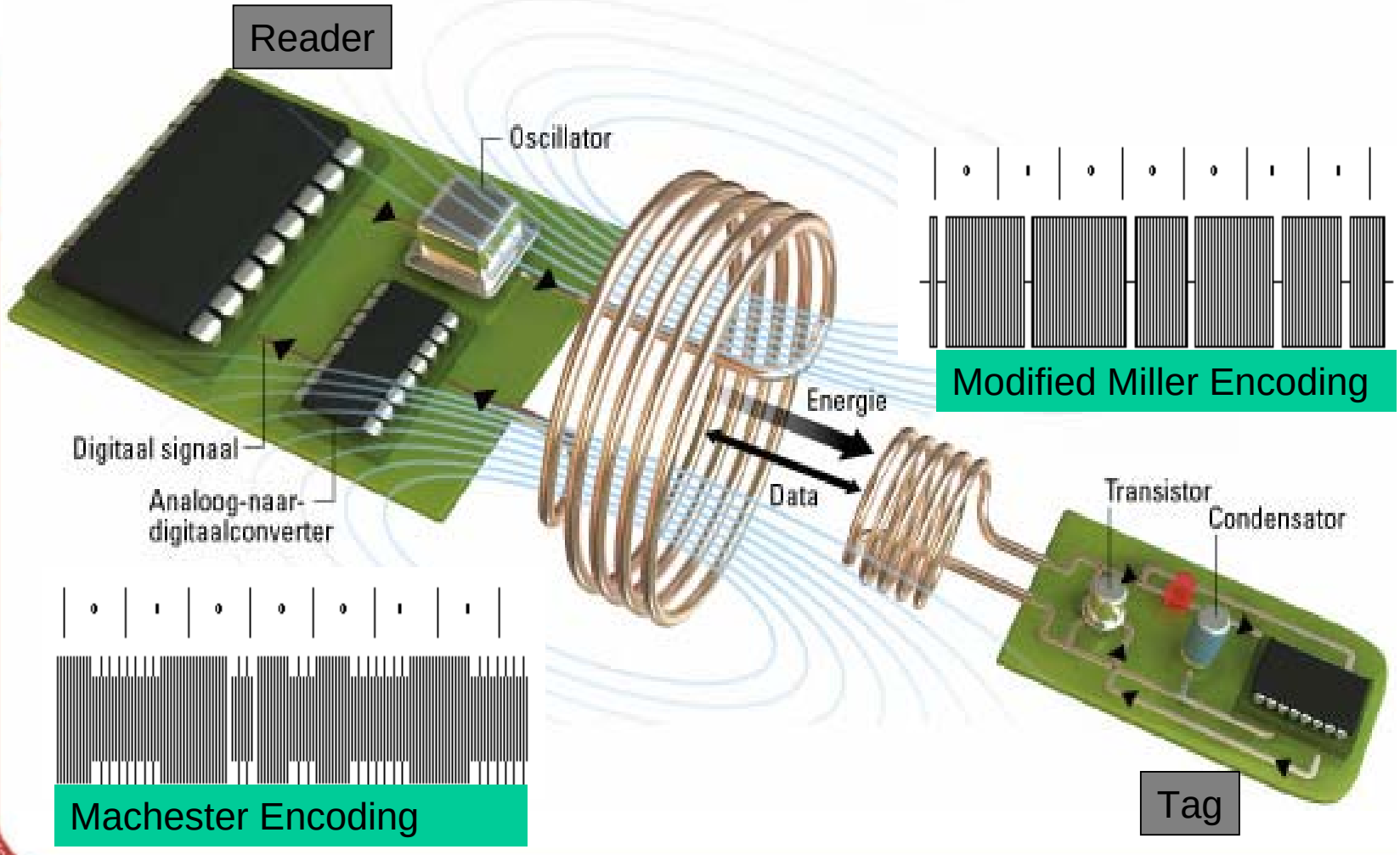
Antenna

RFID Chip

Mifare Ultralight (throw away card)

Mifare Clasic (subscription card)

RFID Technology



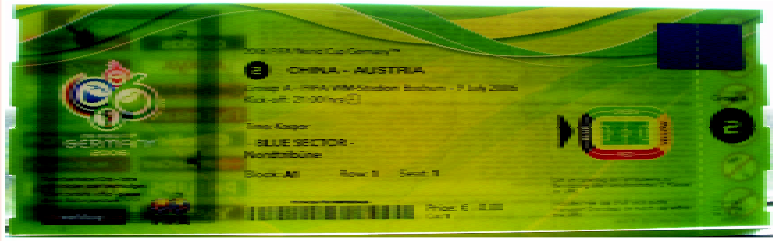
RFID Applications

Identify friend or foe (1942)

Car keys

Public transport ticketing

Electronic passport



Event ticketing

RFID Powder

Implants

Supply chain management

RFID Standards (Proximity cards)

ISO14443A	Mifare	NXP
ISO14443B	CryptoRF	Motorola/Atmel
ISO14443C	Felica	Sony
ISO14443D	-	OTI
ISO14443E	-	Cubic
ISO14443F	LEGIC	KABA
ISO15693	Tag-IT	Texas Instruments

Overview

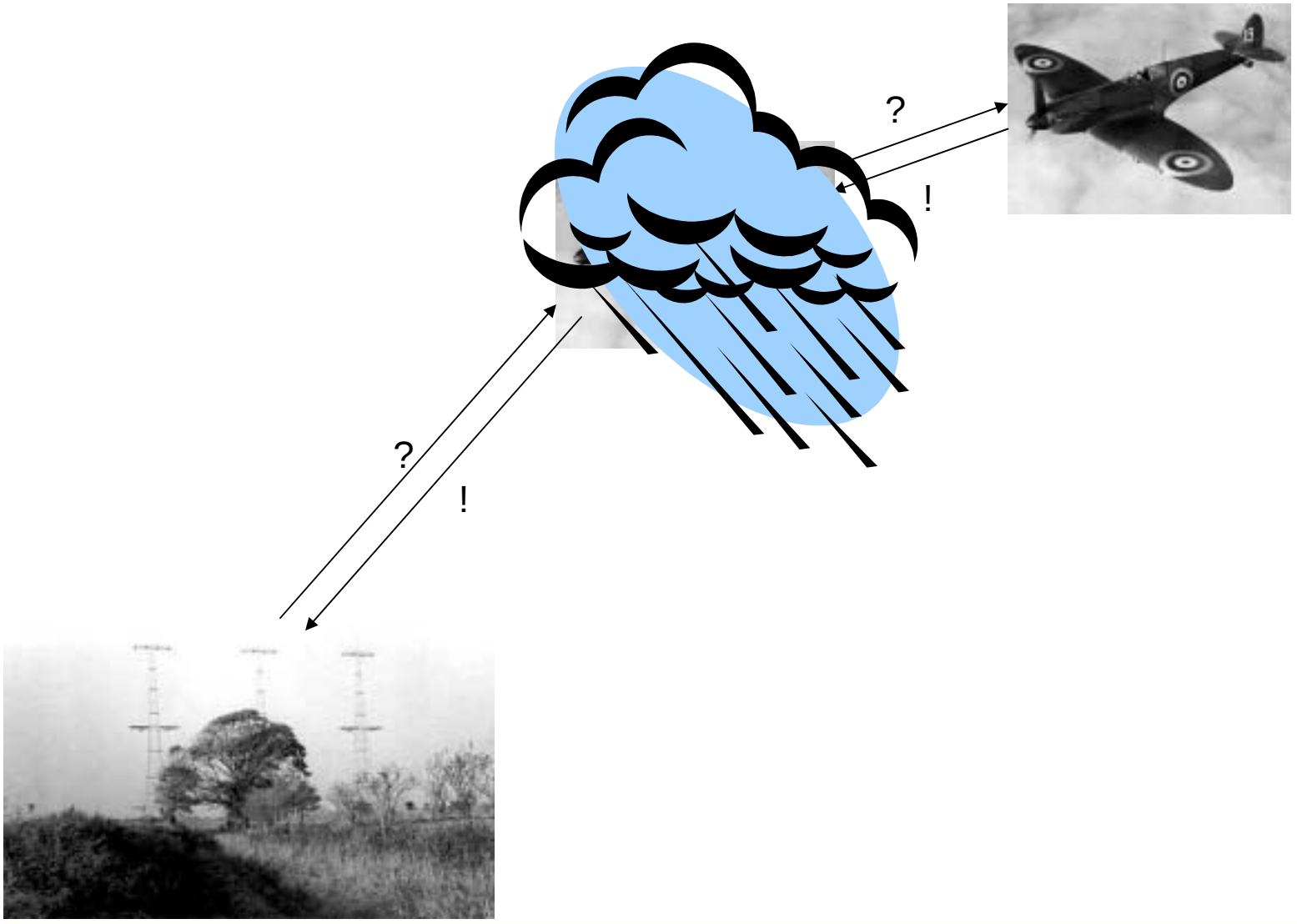
- RFID security and typical problems
- Reverse engineering the Mifare Classic
- Weaknesses of the Mifare Classic
- What to do? (with the OV-Chipkaart)

RFID Security

RFID Security

- Relay attack
- Replay attack
- Cryptanalytic attack
- Tracing attack
- ...

RFID Security – Relay Attack



RFID Security – Relay Attack

- Wireless communication
- No link between authenticating object (tag) and service receiver (tag holder)
  - Attacker A initiates service
  - Attacker A **relays** queries to tag to attacker B
  - Attacker B sends queries to victim's tag
  - Attacker B **relays** answers back to attacker A
  - Attacker A answers queries
- Countermeasures
  - Second authentication channel
  - Distance bounding protocols

RFID Security – Replay Attack

- No clock
- Weak randomness
  - Attacker **intercepts** communication between tag and reader
  - Attack **replays** communication at a later time

RFID Security – Replay Attack

- No clock
- Weak randomness
  - Attacker **intercepts** communication between tag and reader
  - Attack **replays** communication at a later time
- Countermeasures (standard):
  - Challenge-response authentication (needs clock, randomness, or some other form of “freshness”)

RFID Security – Crypto Attacks

- Low energy
- Low computational capacity
- Weak cryptography
  - Attacker can break encryption scheme

RFID Security – Tracing Attack

- Used for identification
- Anti-collision phase
  - Attacker can recognize people based on the RFID tags they are carrying

RFID Security

- No clock, weak randomness
  - → replay attacks
- Low computational capacity
  - → cryptanalytic attacks
- Wireless
  - → relay attacks
- Used for identification
  - → tracking attacks (privacy)

Mifare Classic

Timeline

2004: Fudan Microelectronics (China): Physical clone of Mifare Classic

Summer 2006: Flavio Garcia Lab (RU): Start of development of Ghost

Nov 2007: Verdult & De Koning Gans (RU): ISO 14443A, Ghost & Proxmark

Dec 2007: Nohl (VA), Starbug, Plotz (CCC): Partial rev. engineering Mifare Classic

Feb 2008: Verdult (RU): Cloning Mifare Ultralight (Throw-away OV-Chipcard)

Feb 2008: TNO: No alarm, advanced equipm. needed to crack Mf. Classic, 2 year respite

Mar 2008: Digital Security (RU): Full rev. engineering Mifare Classic (OV-Chipcard)

Mar 2008: Digital Security (RU): Key recovery of Mifare Classic

Apr 2008: Royal Holloway: Fraud likely, replace cards, design should be open, modular

Jun 2008: NXP: Law-suit to stop publication

Jul 2008: Court Arnhem: Publication allowed

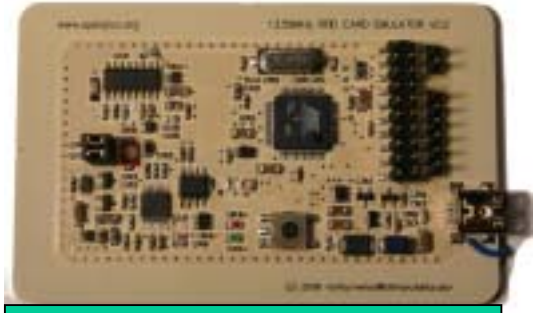
Oct 2008: Digital Security (RU): Presentation at ESORICS 2008

Equipment

OpenPCD - reader

Commerical reader

Ghost
Tag emulator



OpenPicc – tag emulator

Proxmark III
reader & tag emulator

Reverse Engineering - Eavesdropping

- Use empty card
- Use reader to send commands to tag
- Use Ghost/Proxmark to intercept signal

Reverse Engineering - Eavesdropping

Reverse Engineering

Step	Sender	Hex	Abstract
01		26	Request A
02		04 00	Answer request
03		93 20	
04			
05		43	(UID)
06			Classic 1K
07			Auth(block 4)
08		3b ae 03 2d	Rnd_C
09		c4 94 a1 d2 6e 96 86 42	Rnd_R+Ans_C(?)
10		84 66 05 9e	Ans_R(?)
11		a0 61 d3 e3	Inc(block 4)
12		0d	Ack
13		26 42 ea 1d f1 68	Value
14		8d ca cd ea	Trans(block 4)
15		06	Ack

Depends on Rnd_R and shared secret

Depends on Rnd_C and shared secret

Authenticcation

Communication encrypted

Reverse Engineering

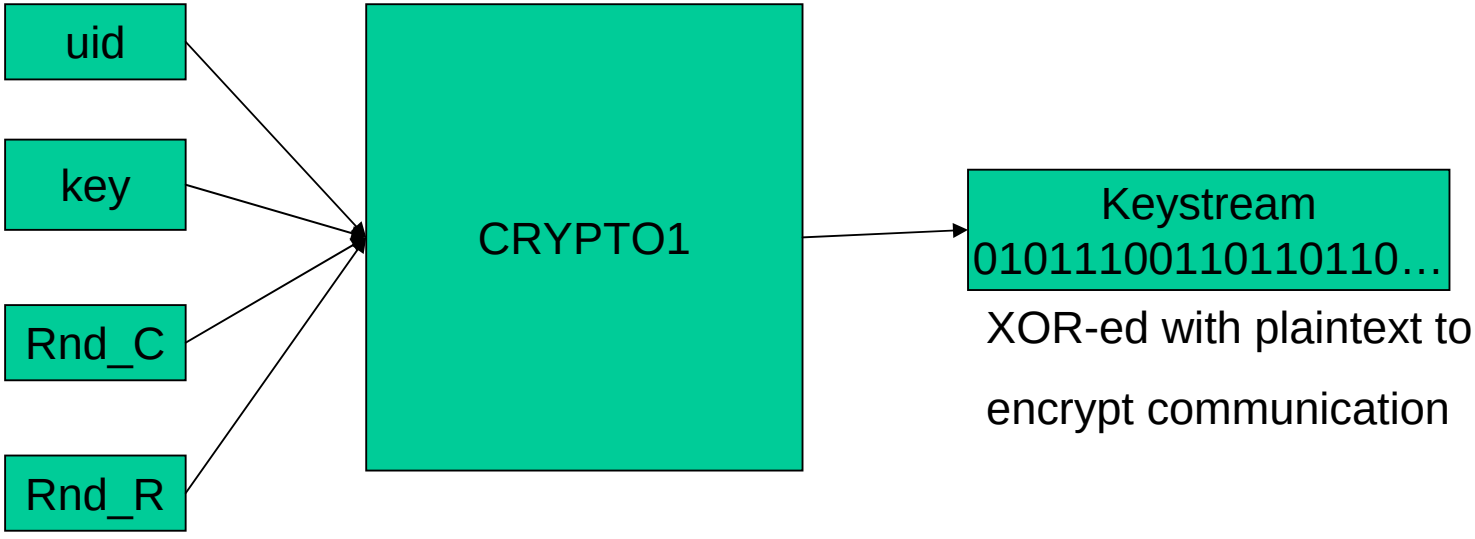
- Communication Protocol
  - ISO14443A
    (no need to reverse-engineer)
  - Proxmark III behaves as tag & reader
- Command Codes
- Authentication Protocol
- Encryption Algorithm

Reverse Engineering – Auth.Prot.

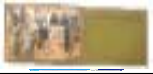
- Goal: establish mutual authentication
  - Challenge by card: Rnd_C
  - Challenge by reader: Rnd_R
  - Answer by reader: Ans_C
    - What is this?
  - Answer by card: Ans_R
    - What is this?
- Goal: initialize session key
    - How does the session key depend on shared secret (key), uid, Rnd_C, Rnd_R?

Reverse Engineering – Auth.Prot.

- Goal: initialize session key
  - How does the session key depend on shared secret (key), uid, Rnd_C, Rnd_R?



Reverse Engineering – Auth.Prot.

Step	Sender	Hex	Abstract
01		26	Request A
02		04 00	Answer request
03		93 20	Select
04		2a 69 8d 43 8d	UID
05		93 70 2a 69 8d 43 8d	Select(UID)
06		08 b6 dd	Mifare Classic 1K
07		60 04 d1 3d	Auth(block 4)
08		3b ae 03 2d	Rnd_C
09		c4 94 a1 d2 6e 96 86 42	Rnd_R+Ans_C(?)
10		84 66 05 9e	Ans_R(?)
11		a0 61 d3 e3	Inc(block 4)
12		0d	Ack
13		26 42 ea 1d f1 68	Value
14		8d ca cd ea	Trans(block 4)
15		06	Ack

Reverse Engineering – Auth.Prot.

Step	Sender	Hex	Abstract
01		26	Request A
02		04 00	Answer request
03		93 20	Select
04		2a 69 8d 43 8d	UID
05		93 70 2a 69 8d 43 8d	Select(UID)
06		08 b6 dd	Mifare Classic 1K
07		60 04 d1 3d	Auth(block 4)
08		3b ae 03 2d	Rnd_C
09		c4 94 a1 d2 6e 96 86 42	Rnd_R+Ans_C(?)
10		84 66 05 9e	Ans_R(?)
11		a0 61 d3 e3	Inc(block 4)
12		0d	Ack
13		26 42 ea 1d f1 68	Value
14		8d ca cd ea	Trans(block 4)
15		06	Ack

Reverse Engineering – Auth.Prot.

Step	Sender	Hex	Abstract
01		26	Request A
02		04 00	Answer request
03		93 20	Select
04		00 00 00 00 ac	UID
05		93 70 00 00 00 00 ac	Select(UID)
06		08 b6 dd	Mifare Classic 1K
07		60 04 d1 3d	Auth(block 4)
08		00 00 00 00	Rnd_C
09		f3 9d be 27 88 a6 b6 dd	Rnd_R+Ans_C(?)
10		?	Ans_R(?)
11			
12			
13			
14			
15			

Reverse Engineering – Auth.Prot.

Step	Sender	Hex	Abstract
01		26	Request A
02		04 00	Answer request
03		93 20	Select
04		ff ff ff ff 34	UID
05		93 70 ff ff ff ff 34	Select(UID)
06		08 b6 dd	Mifare Classic 1K
07		60 04 d1 3d	Auth(block 4)
08		00 00 00 00	Rnd_C
09		14 58 3d ff a8 bb cd e1	Rnd_R+Ans_C(?)
10		?	Ans_R(?)
11			
12			
13			
14			
15			

Reverse Engineering – Auth.Prot.

Step	Sender	Hex	Abstract
01		26	Request A
02		04 00	Answer request
03		93 20	Select
04		00 00 00 00 ac	UID
05		93 70 00 00 00 00 ac	Select(UID)
06		08 b6 dd	Mifare Classic 1K
07		60 04 d1 3d	Auth(block 4)
08		ff ff ff ff	Rnd_C
09		14 58 3d ff 11 7d ad fe	Rnd_R+Ans_C(?)
10		?	Ans_R(?)
11			
12			
13			
14			
15			

Reverse Engineering – Auth.Prot.

Step	Sender	Hex	Abstract
01		26	Request A
02		04 00	Answer request
03		93 20	Select
04		ff ff ff ff 34	UID
05		93 70 ff ff ff ff 34	Select(UID)
06		08 b6 dd	Mifare Classic 1K
07		60 04 d1 3d	Auth(block 4)
08		00 00 00 00	Rnd_C
09		14 58 3d ff a8 bb cd e1	Rnd_R+Ans_C(?)
10		?	Ans_R(?)
11			
12			
13			
14			
15			

Reverse Engineering – Auth.Prot.

Step	Sender	Hex	Abstract
01		26	Request A
02		04 00	Answer request
03		93 20	Select
04		00 00 00 00 ac	UID
05		93 70 00 00 00 00 ac	Select(UID)
06		08 b6 dd	Mifare Classic 1K
07		60 04 d1 3d	Auth(block 4)
08		ff ff ff ff	Rnd_C
09		14 58 3d ff 11 7d ad fe	Rnd_R+Ans_C(?)
10		?	Ans_R(?)
11			
12			
13			
14			
15			

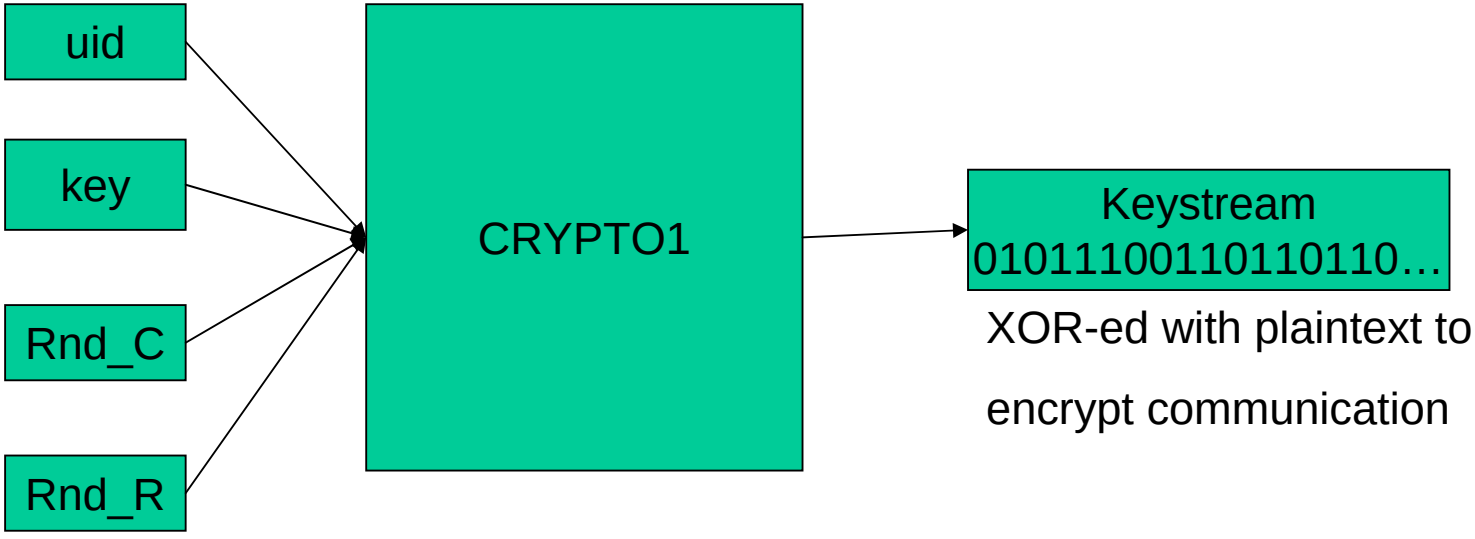
Unchanged

Changed

Conclusing/Guess: Session key depends on uid XOR Rnd_C

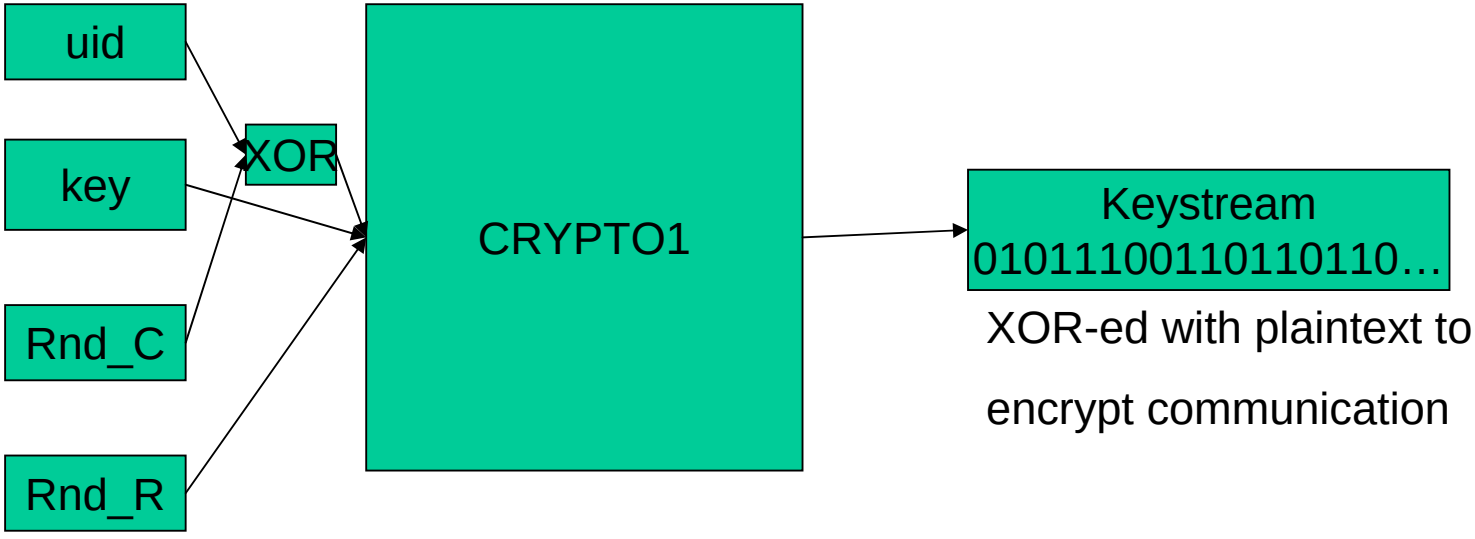
Reverse Engineering – Auth.Prot.

- Goal: initialize session key
  - How does the session key depend on shared secret (key), uid, Rnd_C, Rnd_R?

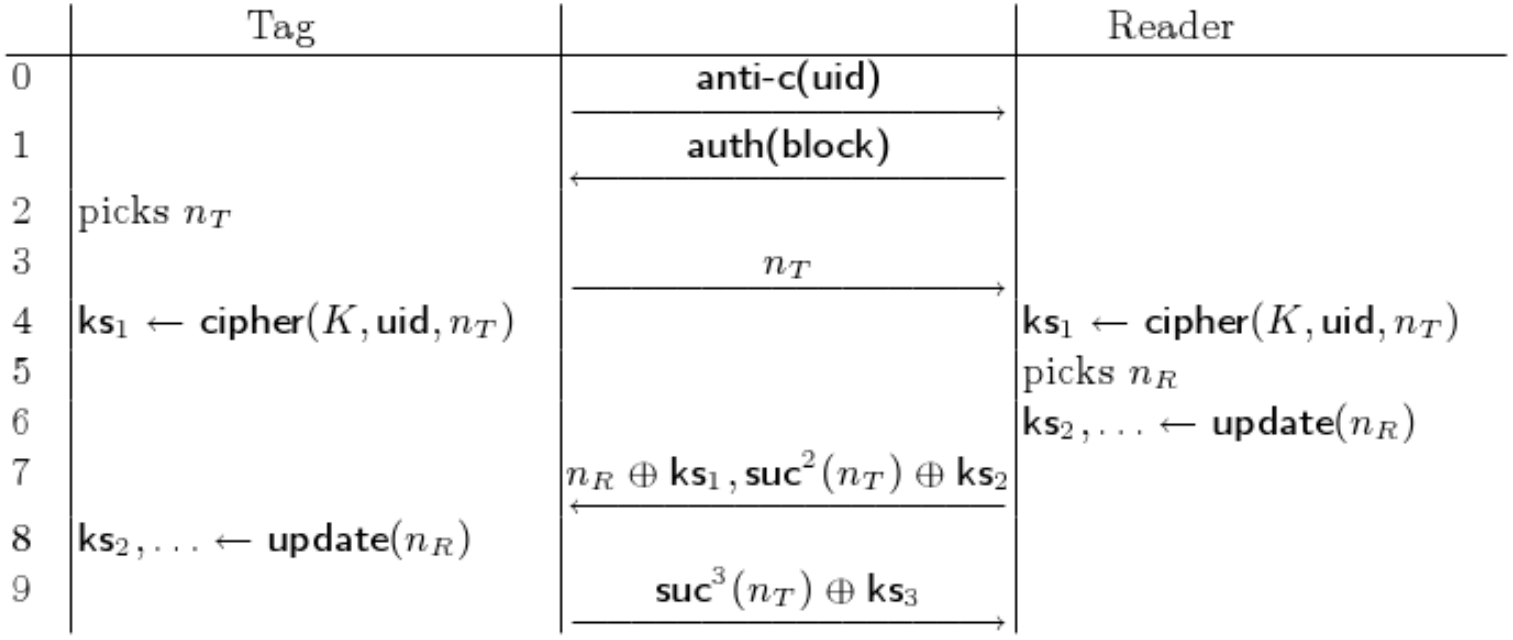


Reverse Engineering – Auth.Prot.

- Goal: initialize session key
  - How does the session key depend on shared secret (key), uid, Rnd_C, Rnd_R?



Reverse Engineering – Auth.Prot.



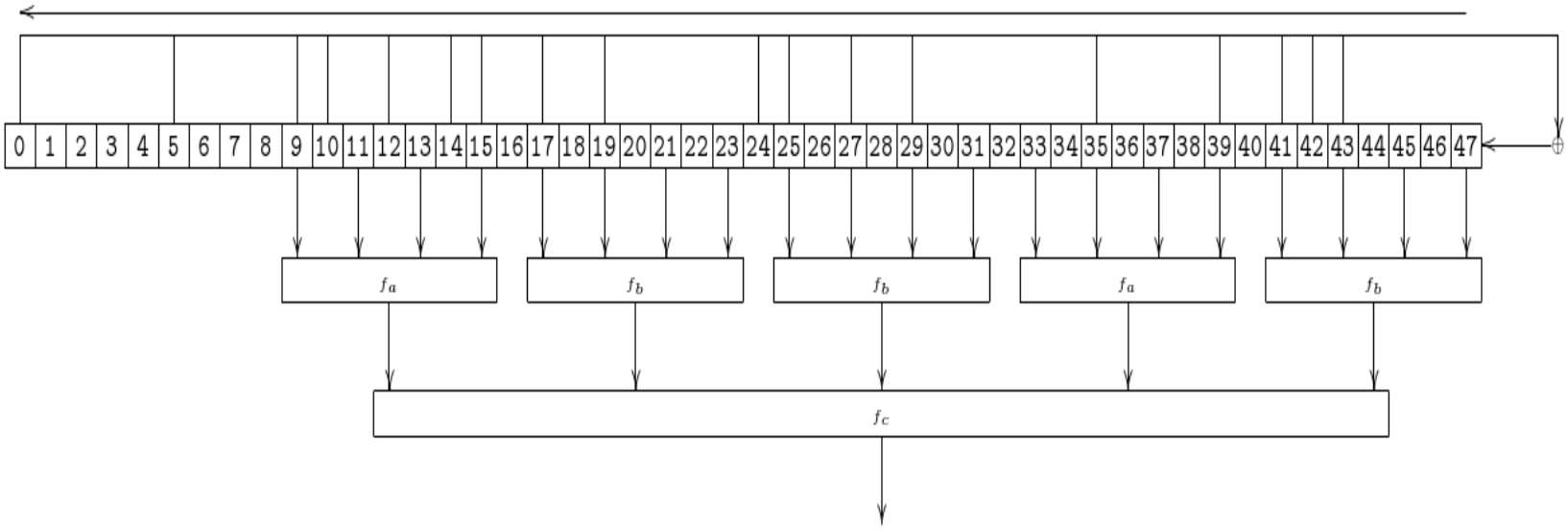
Authentication Protocol

Suc := next random number generated

Cipher := cipher initialization with key,uid,nonce

Update := cipher update with nonce

Reverse Engineering – Encr. Alg.



CRYPTO1

LFSR shifts one to the left every clock tick

Filter function generates one bit of keystream

Mifare Security

Mifare Security – (Some) weaknesses

- Weak random number generator on tag
  - 16-bit entropy
  - resets when tag enters e.m. field
    (not random at all)
- Extremely weak cryptographic algorithm
  - 48-bit key
  - only 20-bit effective security

Security by obscurity: bad idea!
Gives (very) poor crypto
Will be reverse engineered anyway

Mifare Security

Why being open about security makes us all safer in the long run

Bruce Schneier
The Guardian, Thursday August 7 2008
Article history

London's Oyster card has been cracked, and the final details will become public in October. NXP Semiconductors, the Philips spin-off that makes the system, lost a court battle to prevent the researchers from publishing. People might be able to use this information to ride for free, but the sky won't be falling. And the publication of this serious vulnerability actually makes us all safer in the long run.

Here's the story. Every Oyster card has a radio-frequency identification chip that communicates with readers mounted on the ticket barrier. That chip, the "Mifare Classic" chip, is used in hundreds of other transport systems as well — Boston, Los Angeles, Brisbane, Oslo, Amsterdam, Taipei, Shanghai, Rio de Janeiro — and as an access pass in thousands of companies, schools, hospitals, and government buildings around Britain and the rest of the world.

The security of Mifare Classic is terrible. This is not an exaggeration; it's kindergarten cryptography. Anyone with any security experience would be embarrassed to put his name to the design. NXP attempted to deal with this embarrassment by keeping the design secret.

The group that broke Mifare Classic is from Radboud University Nijmegen in the Netherlands. They demonstrated the attack by riding the Underground for free, and by breaking into a building. Their two papers (one is already online) will be published at two

Security by obscurity: bad idea!
Gives (very) poor crypto
Will be reverse engineered anyway

Quotes

- *The security of Mifare Classic is terrible. This is not an exaggeration; it's kindergarten cryptography. Anyone with any security experience would be embarrassed to put his name to the design. NXP attempted to deal with this embarrassment by keeping the design secret.* [Bruce Schneier, The Guardian, August 7]
- *Voorzover het gaat om bedrijfsschade en schade als gevolg van eventuele claims van afnemers, legt die weinig gewicht in de schaal bij de afweging van belangen, omdat die kans op schade in hoge mate toegerekend moet worden aan het produceren en in het verkeer brengen van een chip met intrinsieke manco's, wat de verantwoordelijkheid van NXP is en niet van RUN c.s. die die manco's slechts door onderzoek bloot hebben gelegd.* [Voorzieningenrechter Rechtbank Arnhem, July 18]

Mifare Security - Consequences

- Card can be read

  (design distance only 10cm, but 10m has been achieved)
- Card can be cloned

  (to the Ghost/Proxmark; can't (yet?) change uid on a real card)
- Card can be restored to previous state

Mifare Security – Attack Scenarios

- Write increased balance to card
  - (blocked next day?)
  - (does not work with OV-Chipkaart)
- Restore card to initial state
  - (blocked next day?)
- Clone someone else's card
  - (blocked next day? which one?)
- More...?

- Countermeasures: in back office
  - (will this work?)

What to do?

Messenger Perspective

- Assume university research reveals deficiency in brakes of new car
  - Probably much praise for researchers…
  - …little for manufacturer
- How long should details be kept secret?
  - Experience by security researchers
    - Only full disclosure works
  - 6 Months chosen for Mifare Classic
    - Unusually long for this computer security
    - But cannot replace installed base

Producer Perspective

- Sell more advanced cards
  - DesFire, DesFire 8, Smart MX, Mifare Plus
- Should NXP stop producing and selling Mifare Classic?
- Reputation damaged, but chance to sell new cards

Customer Perspective

(TLS, TFL, system integrators, ...)

- "Customer makes wrong choice" (NXP, De Gelderlander, March 14)
- For OV-Chipkaart
  - Political pressure to keep cost low
  - System copied from elsewhere
  - No critical attitude wrt security and privacy ("it works everywhere else")
- Surprised by card vulnerabilities

Security by Obscurity

- Kerckhoffs' Principle (1884): The security of a (cryptographic) system should not depend on the secrecy of the system itself, but only on the secrecy of the key.
- Shannon's Maxim: The enemy knows the system.

- Security by obscurity
  - derided in academia
  - considered reasonable for hardware
  - rewards for producers
    - keeps out competition
    - keeps customers uninformed (lemon market!)
    - higher score in Common Criteria evaluation
  - proprietary cryptography is invariably very weak

What to do? (in general)

- Make risk analysis
  - Can system withstand broken cards?
  - Do cards have to be replaced?
  - When?

- Don’t focus on attacks!
- Focus on weaknesses.
  - “Attacks never get worse’’ (NSA)

What to do? (with the OV-Chipkaart)

1. Roll-out as planned
   - “there is no problem” approach
   - Politically not a realistic option
2. Roll-out as planned and upgrade a.s.a.p
   - Legacy/maintenance problems
3. Postpone
   - Simple, longer delay
   - Chance to fix privacy issues as well
4. Stop
   1. Not unique: Sydney TCard
   2. Payment via mobile phone?

Conclusions

- Mifare Classic is broken
- Security by obscurity really doesn't work

Thank you…