Card-Only Attacks on MiFare Classic

or How to Steal Your Oyster Card and Break into Buildings Worldwide

Nicolas T. Courtois

University College London, UK

Outline

1. Security in the Smart Card world:
   - Traditional model vs. disruptive RFID technology
   - Open vs. Close source models
2. MiFare Crypto 1 cipher:
   waste of silicon x 1 billion copies sold.
3. Barriers to breach and hardware set-up
4. Early attacks
5. Card-only attacks [NEW]
   - My own
   - Dutch researchers from Nijmegen
   - Combined
6. Inside Oyster Cards + other countries…

Secure Product Development

Why Smart Cards Are Good

Or are they?

The classical model
for smart card security
[Schneier and Schostack 1999]
is about

- physical control of the card by the user,
- hardware barriers that cannot be breached by software,
- splitting the security perimeter:
  - One entity cannot breach other people's security
- and trusting the entities involved in developing components of a secure system
  - Schneier and Schostack already pointed out that companies/people involved in this business can compromise it's security.

slight problem..

Secure Hardware Dev. Management

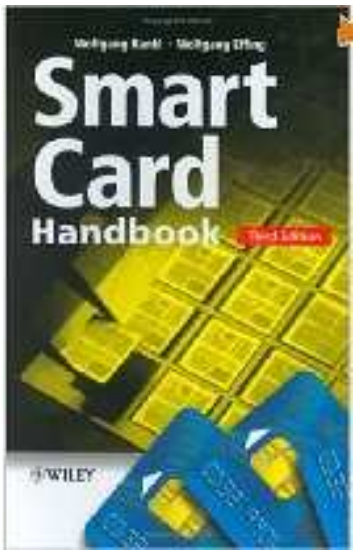
[In smart cards] one design criterion differs from the criteria used for standard chips but is nonetheless very important is that absolutely no undocumented mechanisms or functions must be present in the chip ('that's note a bug, that's a feature').

Since they are not documented, they can be unintentionally overlooked during the hardware evaluation and possibly be used later for attacks.

The use of such undocumented features is thus strictly prohibited [...]

[pages 518-519 in the Smart Card handbook
by Wolfgang Rankl and Wolfgang Effing,
1088 pages, Wiley, absolute reference in the industry]

Common Criteria



Problems:

This model breaks apart with RFID smart cards:

- RFID => no user control.

The secrecy of the product spec is:

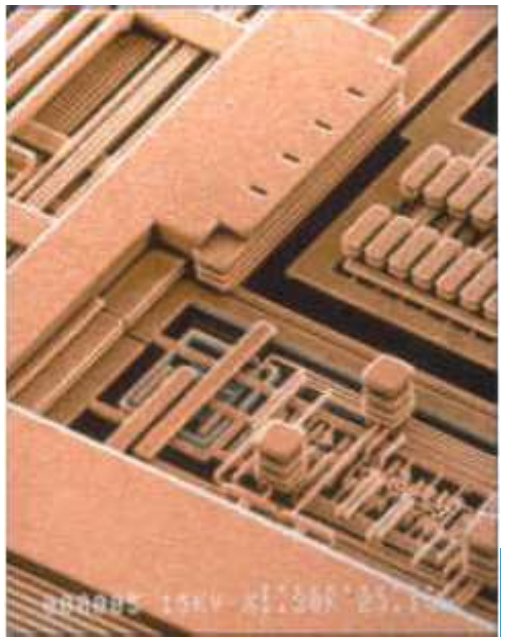
- not an extra security layer, but a source of unexpected and critical security vulnerabilities
  - that by the fact of being hidden gives an utterly false sense of security

MiFare Classic Crypto-1

Stream cipher used in about 200 million RFID chips worldwide.

- Ticketing (e.g. London's Underground).
- Access to high-security buildings
- Etc.

What's Inside?



Open Source vs. Closed Source Crypto

Secrecy:

Very frequently
an obvious
business decision.

- Creates entry barriers for competitors.
- But also defends against hackers.

Kerckhoffs' principle: [1883]

"The system must remain secure should it fall in enemy hands …"

Kerckhoffs' principle: [1883]

Most of the time: incorrectly understood.

No obligation to disclose.

- Security when disclosed.
- Better security when not disclosed.

Yes:

Military:
layer the defences.

Basic economics:
these 3 extra months
are simply worth a
a lot of money.

Kerckhoffs principle is WRONG in the world of secure hardware devices

Dutch researchers got a large grant to develop an OPEN SOURCE replacement for the Oyster card system.

This cannot work.

- Algorithm secrecy [once they are good] MUST be preserved.
- Reason: side channel attacks are VERY hard to prevent.
- In some applications, for example Pay TV the system is broken immediately when the cryptographic algorithms are public.

Silicon Hacking

Tarnovsky Lab

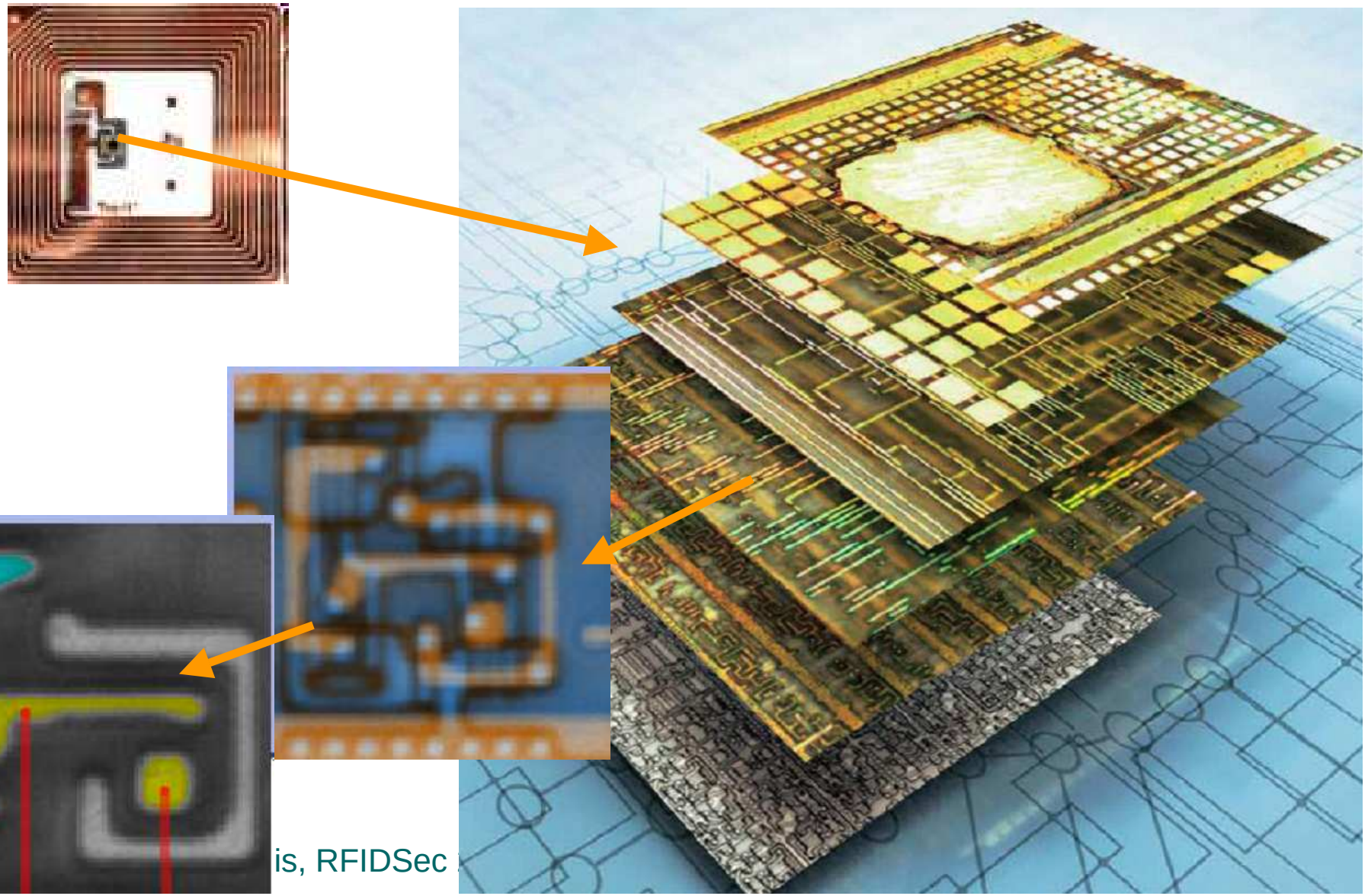
Only a few thousands of dollars worth of equipment

Clear and Present Danger

Reverse engineering is NOT that hard.

A few thousand dollars microscope + software.

Reverse-Engineering [Nohl et al.]



Crypto-1 Cipher

Waste of Silicon

MiFare was manufactured by Philips, now NXP, and licensed to Infineon.

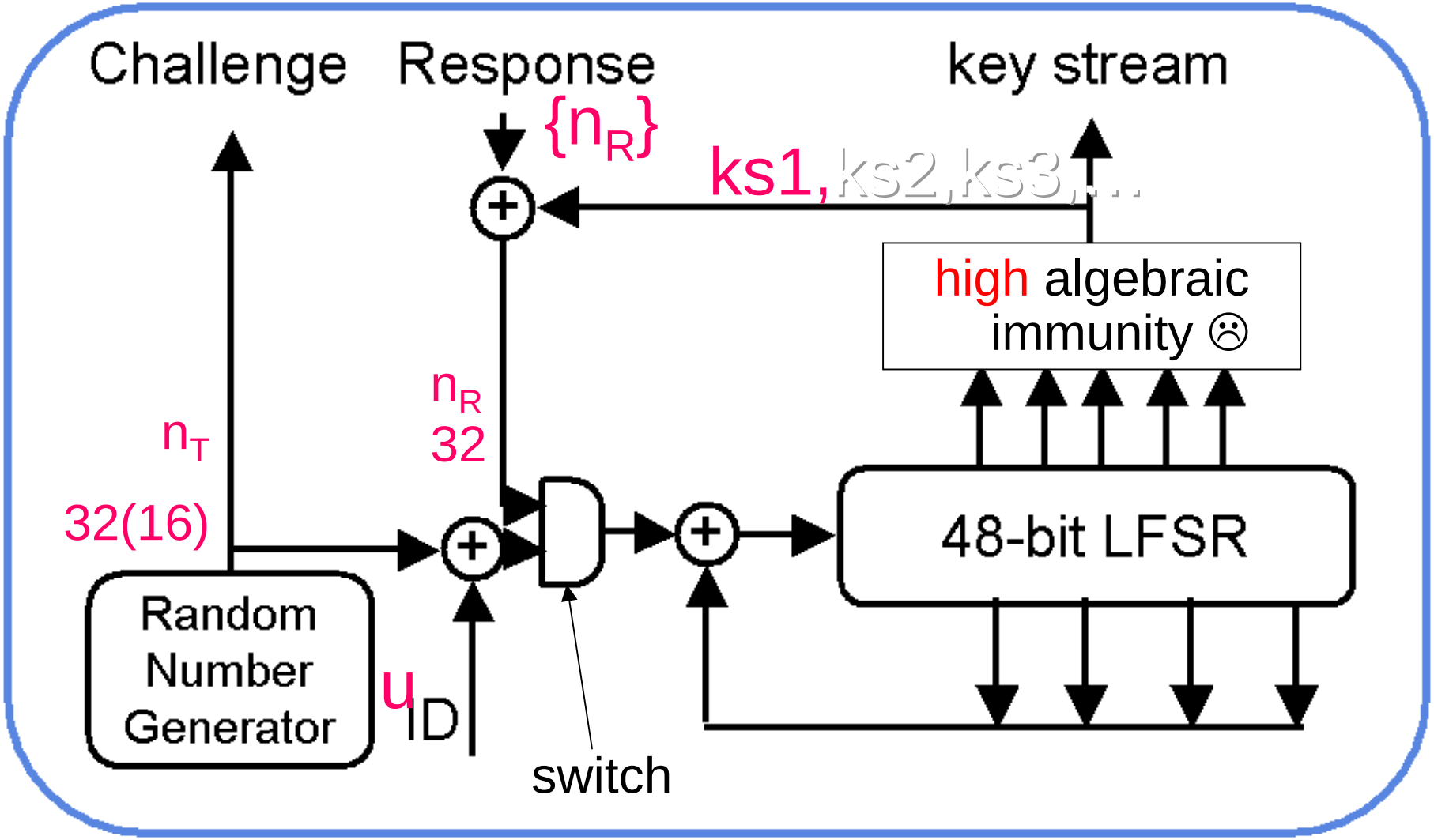
BUT, even a hardware or software designer would NOT notice how weak the cipher is.

Identical Boolean functions are implemented differently.

Camouflage?

Due to a combination with another terrible weakness half of the silicon is wasted…

Crypto-1 Algo + Auth. Protocol



Background – Crypto 1 Cipher

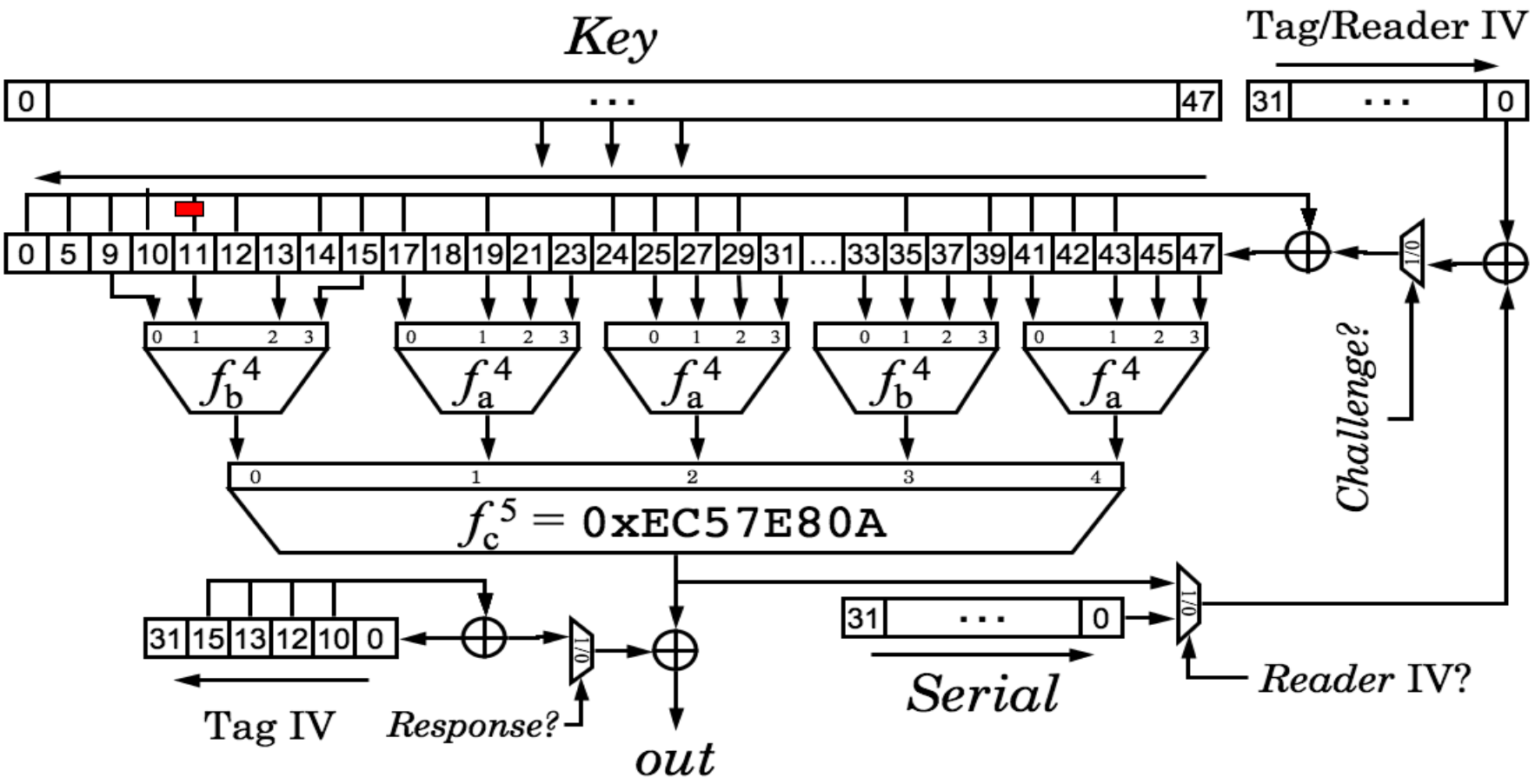
- 48 bit LFSR
- The feedback function *L* is as follows:

Definition 2.1. The feedback function $L : \mathbb{F}_2^{48} \to \mathbb{F}_2$ is defined by $L(x_0 x_1 \ldots x_{47}) := x_0 \oplus x_5 \oplus x_9 \oplus x_{10} \oplus x_{12} \oplus x_{14} \oplus x_{15} \oplus x_{17} \oplus x_{19} \oplus x_{24} \oplus x_{25} \oplus x_{27} \oplus x_{29} \oplus x_{35} \oplus x_{39} \oplus x_{41} \oplus x_{42} \oplus x_{43}$.

- The non-linear filter function *f* is defined as:

$$f(x_0 x_1 \ldots x_{47}) := f_c(f_a(x_9, x_{11}, x_{13}, x_{15}), f_b(x_{17}, x_{19}, x_{21}, x_{23}), f_b(x_{25}, x_{27}, x_{29}, x_{31}), f_a(x_{33}, x_{35}, x_{37}, x_{39}), f_b(x_{41}, x_{43}, x_{45}, x_{47})).$$

Crypto1 Cipher



$f_a^4 = 0x9E98 = (a+b)(c+1)(a+d)+(b+1)c+a$

$f_b^4 = 0xB48E = (a+c)(a+b+d)+(a+b)cd+b$

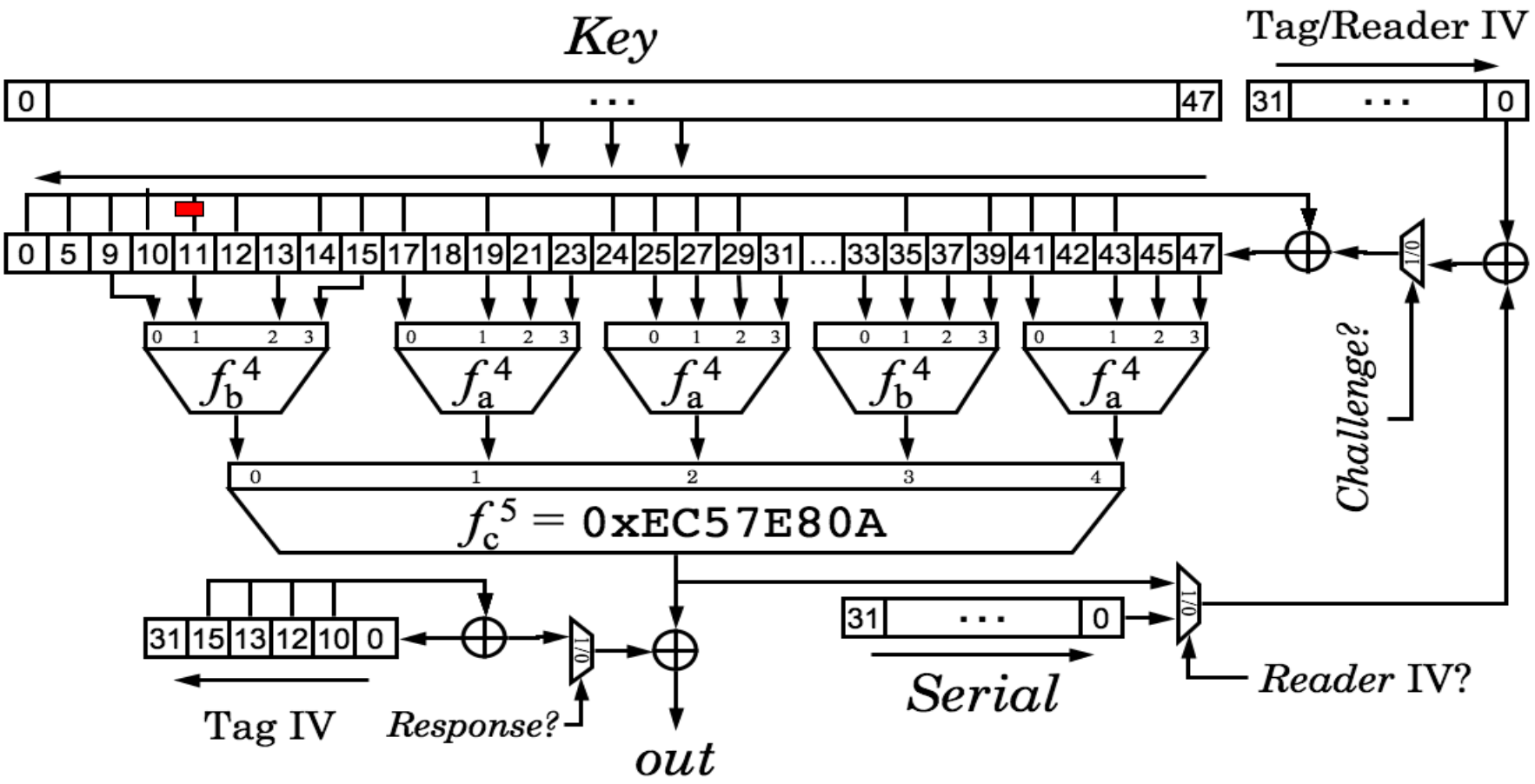
Tag IV ⊕ Serial is loaded first, then Reader IV ⊕ NFSR

Waste of Silicon

Internal bits are computed 2-3 times.

One could save half of the gates!

Crypto1 Cipher

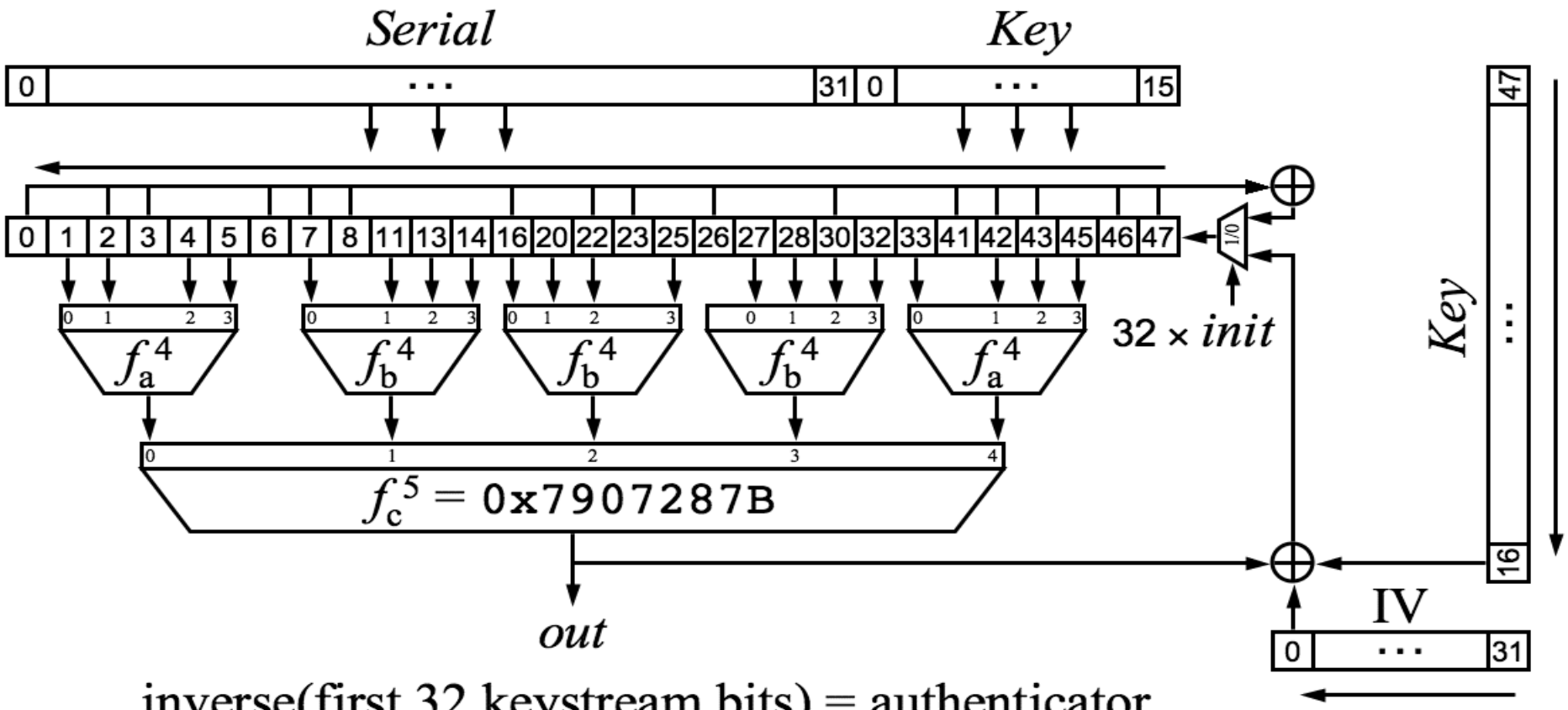


$f_a^4 = 0x9E98 = (a+b)(c+1)(a+d)+(b+1)c+a$

$f_b^4 = 0xB48E = (a+c)(a+b+d)+(a+b)cd+b$

Tag IV ⊕ Serial is loaded first, then Reader IV ⊕ NFSR

Hitag2 Cipher



inverse(first 32 keystream bits) = authenticator

f_a^4 = 0x2C79 = abc+ac+ad+bc+a+b+d+1

f_b^4 = 0x6671 = abd+acd+bcd+ab+ac+bc+a+b+d+1

Strong or Weak?

High Algebraic Immunity.

- Does NOT help.
- Many “direct” algebraic attacks exist. We can break “any cipher”, if not too complex…

First efficient attack on this cipher was an Algebraic Attack
[Courtois, O’Neil, Nohl, see eprint/2008/166].

Soon became obsolete.

Exhaustive Key Search

- 48 bits, about 4 years on 1 CPU.
  - Hours with FPGA.

Our First Attack [04/2008]

- 12 seconds on the same CPU.

Better Attack [09/2008]

- 0.05 seconds.
  [de Koning Gans et al, Esorics 2008]

Beyond Crypto-1

…AC can break “any cipher”, if not too complex…

But other attacks are faster…

- However,
  - our attack does NOT require human intervention
  - more generally applicable:
    we can also break Hitag2 in 1 day
    (instead of say 3 years).
    - has fully irregular taps. See: Inversion attacks:

[Ross Anderson: Searching for the Optimum Correlation Attack, FSE'94]

- to appear in ACS’2009, September 2009.

Clone Attacks

Cloning the Card

Remark:

I wouldn't care that much about hackers that get free rides on the Tube.

- What about the Cabinet Office, nuclear facilities, big banks in the City?
  - It seems that most buildings actually use MiFare Classic (70 % market share) or an even less secure LF systems..

Key Sizes and Brute Force Attacks

Key Size = 48 Bits

Single PC – 3 years. FPGA: days…

Claim: 48 bits can still be
a SECURE key size in 2010.

- in authentication only (extra randomness effectively prevents brute force attacks),

So brute for attacks are infeasible

WHAT???? Yes.

Brute Force Infeasible?

Yes, due to the protocol.

Sound engineering principle:

The card never answers anything related to the secret data, unless the reader sends a valid cryptogram on 8 bytes…

If I know the key, it takes time to confirm this.

It takes time to reject wrong keys too.

2^{32} queries to the card => months of online time querying the card...

The Protocol

Tag Nonce

- The pseudo random number generator uses a 16 bit LFSR to produce a 32 bit nonce.

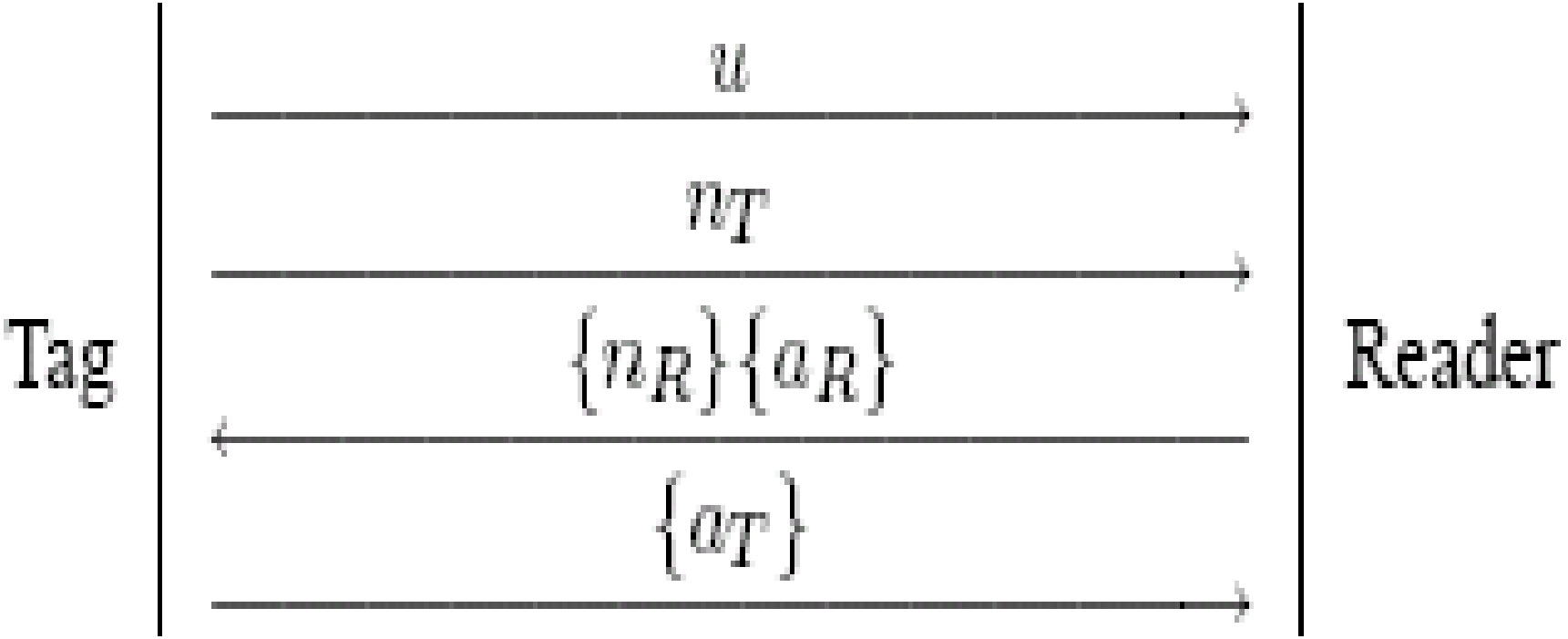
$$L_{16}(x_0 x_1 \ldots x_{15}) := x_0 \oplus x_2 \oplus x_3 \oplus x_5$$

- The successor function is defined as follows:

$$\mathrm{suc}(x_0 x_1 \ldots x_{31}) := x_1 x_2 \ldots x_{31} L_{16}(x_{16} x_{17} \ldots x_{31})$$

- The period of the PRG is 65535, it shifts every 9.44μs, and cycles in 618ms.
- For some [clone/compatible/etc] cards it will be
  - 618ms/X, for example x=4,5 etc..

Authentication Protocol



Step by Step

1. First the reader and the card engage in the anti-collision protocol where the reader learns the unique ID of the card and selects the card.
2. The reader issues a command '60 0X' by which it starts the mutual symmetric-key authentication process between the card and the reader, with the key pertaining to the block number 0X.
3. The card answers with a random n_T on 4 bytes,

Step by Step

4. The reader sends a cryptogram on 8 bytes which is $n_R \oplus ks1$ and $suc^2(n_T) \oplus ks2$.
5. The card responds with 4 bytes, $suc^3(n_T) \oplus ks3$.
6. Then all subsequent communications and data are encrypted and the card will now accept read, write and increment commands for block 0X.

Here n_R is the 32-bit nonce chosen by the reader, suc is a certain bijective function, and $(ks1, ks2, ks3)$ are the 96 bits of the keystream produced by the

Generation of LFSR Stream – 1

Definition 2.6. Given a key $k = k_0 k_1 \dots k_{47} \in \mathbb{F}_2^{48}$, a tag nonce $n_T = n_{T,0} n_{T,1} \dots n_{T,31} \in \mathbb{F}_2^{32}$, a uid $u = u_0 u_1 \dots u_{31} \in \mathbb{F}_2^{32}$, and a reader nonce $n_R = n_{R,0} n_{R,1} \dots n_{R,31} \in \mathbb{F}_2^{32}$, the internal state of the cipher at time i is $a_i := a_i a_{i+1} \dots a_{i+47} \in \mathbb{F}_2^{48}$. Here the $a_i \in \mathbb{F}_2$ are given by

$$
\begin{aligned}
a_i &:= k_i && \forall i \in [0, 47] \\
a_{48+i} &:= L(a_i, \dots, a_{47+i}) \oplus n_{T,i} \oplus u_i && \forall i \in [0, 31] \\
a_{80+i} &:= L(a_{32+i}, \dots, a_{79+i}) \oplus n_{R,i} && \forall i \in [0, 31] \\
a_{112+i} &:= L(a_{64+i}, \dots, a_{111+i}) && \forall i \in \mathbb{N}.
\end{aligned}
$$

Generation of LFSR Stream – 2

Furthermore, we define the keystream bit $b_i \in \mathbb{F}_2$ at time i by

$$b_i := f(a_i a_{1+i} \ldots a_{47+i}) \qquad \forall i \in \mathbb{N}.$$

We denote encryptions by $\{-\}$ and define $\{n_{R,i}\}, \{a_{R,i}\} \in \mathbb{F}_2$ by

$$\{n_{R,i}\} := n_{R,i} \oplus b_{32+i} \qquad \forall i \in [0, 31]$$
$$\{a_{R,i}\} := a_{R,i} \oplus b_{64+i} \qquad \forall i \in [0, 31].$$

Attacks with a [Genuine] Reader

Key Recovery:

Brute Force

- About 4 years on 1 CPU. Minutes w. FPGA.

Nijmegen Attack

- 0.05 seconds.
  [de Koning Gans et al, Esorics 2008]

These are mild threats. Why?

Keystream Needed:

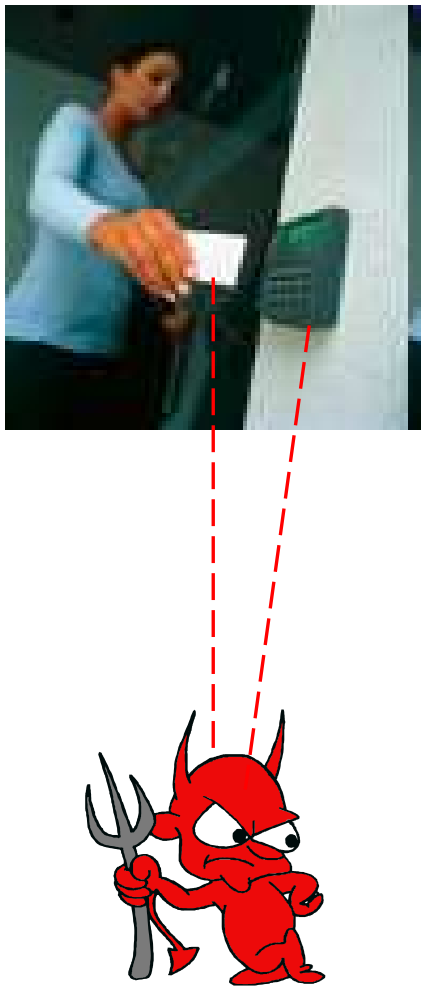
In Theory:

Keystream Data => 0.05 seconds.

In practice:

Very hard to get this data.

Small window of opportunity for the thief.



Card-Only Attacks

Card-Only Attacks

Danger is 24h/24:

Anybody that is sitting/standing next to you can steal your identity (or at least enter some very nice building…)

Recent Results

1. New Dutch paper [Oakland May 2009].
   - card-only Attacks with at least 4000 queries to the card.

2. Better attack – today

   [N. Courtois, SECRYPT 2009, 7-10 July 2009, Milan, Italy]:

   - 300 queries to the card only
     - A card can be copied in 10 seconds by the person standing next to you through contactless queries.

Card-Only Attacks
Infeasible -> Possible?

Parity Attacks

Problem 1:

The card does encrypt data with redundancy.

One should never do that.

- more costly
- weaker
  - and even weaker with a stream cipher:

    Ciphertext Only attack (weak)=>

    gives (small weight) LINEAR equations on the keystream (very strong)

Compare to GSM

BTW:

For the same reason it is currently easy to eavesdrop to GSM communications.

And sometimes make free calls…

Cf. [Biham-Barkan-Keller: Instant Ciphertext-Only Cryptanalysis of GSM.. Crypto'03 and JoC'08]

Problem 2: A Bug in MiFare Classic

Discovered accidentally.

- sometimes, under certain conditions, the card outputs a mysterious 4 bits…
- given the fact that many RFID readers are not 100 % reliable, it is easy to overlook it

Then one can guess how it works…

- what are these conditions?,
- can I predict when this will happen?

Parity Weakness…

- Parity bit computed over the plaintext and is encrypted using the same bit as the next plaintext bit
- If all 8 parity bits are correct but the answer is wrong, the tag responds with the 4 bit error code 0x5 encrypted.

$$p_j := n_{T,8j} \oplus n_{T,8j+1} \oplus \cdots \oplus n_{T,8j+7} \oplus 1$$
$$p_{j+4} := n_{R,8j} \oplus n_{R,8j+1} \oplus \cdots \oplus n_{R,8j+7} \oplus 1$$
$$p_{j+8} := a_{R,8j} \oplus a_{R,8j+1} \oplus \cdots \oplus a_{R,8j+7} \oplus 1$$
$$\forall j \in [0, 3]$$

and the encryptions $\{p_j\}$ of these by

$$\{p_j\} := p_j \oplus b_{8+8j} \qquad \forall j \in [0, 11].$$

The Bug?

Or maybe a backdoor?

- Stop pretending that everything happens by accident.
- We need to assume the worst scenario and examine the consequences:
  - Smart can companies are in the position to embed backdoors in products and these will NOT be found for many many years…

Card-Only Attacks Impossible -> Possible?

The “Bug”

Under certain (parity) conditions when we try to spoof the card with an invalid cryptogram, the card replies with 4 bits.

These 4 bits are the encrypted NACK command at a certain later moment in the keystream generation process.

The “Bug”

Parity-based:

Fact 4.2.2 *The card answers with a 4-bit encrypted NACK if and only if the parity bits for the plaintext after decryption are correct.*

For very long time I made my live much harder, designed several attacks where also 8 the parity bits over the ciphertext were correct. Much harder (Exercise: with 16 parity bits one can still break it under 1000 queries, never published…).

Simplification:

Idea: modify the parity bits only.

Remark: most of the time we can safely ignore how these parity equations work, and use only the fact that:

Fact 4.2.3 *Whatever is the spoofed cryptogram C, there is exactly one choice for the 8 parity bits P_C so the card will reply with a 4-bit encrypted NACK.*

The “Bug” was known…

I was the first to circulate a paper that describes this vulnerability in March 2009.

But then I discovered that MANY people knew about it for a long time… including journalists.

What? It was already broken, can it be broken again?

At the time of release I ignored the latest Nijmegen paper [Oakland IEEE Security and Privacy, 18 May 2009].

Special Cards

Weaker Cards

I've recently examined the card used in Kiev, Ukraine underground.

This card will ALWAYS answer the spoof attempt.
Way easier to clone then…

These are unlicensed clones of MiFare Classic.

They are probably illegal in Ukraine
(but nobody expected that there will ever be a method to distinguish them?)

Investigation of Kiev Cards

Well, they are Fudan Microelectronics FM11RF08 from Shanghai, China.

How do I know this?

The same:

- The same ATR (misleading) "3B8F8001804F0CA000003060300010000000006A"
- sak: 08 00 00
- ATQA: 04 00
- Same normal functionality.

Differences: [visible only to experts or can be discovered 'by accident']

- They answer a spoof attempt with probability 1.
- At the end of the block 0 we always find "bcdefghi". Typical with Fudan.
- Original NXP reply to 7-bit frames 0x26.

Counterfeit MiFare Classic

There are other clones.

Come from India, China and Russia (!).

List: see

http://www.proxmark.org/forum/topic/169/mifare-classic-clones/

Experimental Setup

Cheap Stuff…

Only high-level APDU access.

GET CARD SERIAL NUMBER

CLA	INS	P1	P2	Le
FF	CA	00	00	00

LOAD KEY IN RAM REGISTERS

CLA	INS	P1	Kt	Le	Key
FF	82	20	00	06	FFFFFFFFFFFF

MIFARE CLASSIC AUTHENTICATE

CLA	INS	P1	P2	Nb	Kt
FF	88	00	3A	60	00

MIFARE CLASSIC READ

CLA	INS	P1	P2	Le
FF	B0	00	3A	10

MIFARE CLASSIC WRITE

CLA	INS	P1	P2	Lc	Data
FF	D6	00	3A	10	

Example: rfidiot library does send these commands…

Low-Level Commands (those Sent Over the Air)

C++ + nfclib + ACR122

> 26
< 0400
> 9320
< CA1C46D141
> 9370CA1C46D141 (CRC)
< 08 (CRC)
> 6000(CRC)
< 24D2783A
> CF80E99F1AA2A1F1
> …

UID

Open PCD

TI TRF7960 EVM

Proxmark 3

Getting the data, difficulties…

Precise timing, switch the magnetic field off and on
=> fix the card nonce.

Very hard to achieve in practice.

Recent Results

New Paper by Nijmegen Group

In IEEE Privacy and Security Oakland,
18 May 2009.

$\Rightarrow$ 3[+1] Attacks that exploit the same vulnerability.

My Attack vs. Oakland

Nijmegen attack 3.

- very expensive pre-computation, 400 Gb of data
- then 4000 queries/card
- instant running time.

My latest attack [very different]:

- no pre-computation
- then 300 queries/card, more than 10 times less.
- instant running time too.

=> The strongest attack ever found on MiFare Classic.

My Attack

Cf. eprint.iacr.org/2009/137. Basic Facts:

It is a multiple differential attack.

I exhibit a differential that

- holds simultaneously for 256 differentials this works with probability of about 1/17.
- for 8 differentials the probability is about 0.75 (!!).

Both are differences on 51 bits of the state of the cipher.

A VERY STRONG property(!).

Key Discovery

Fact 5.1.1 *The probability that the 3 bits of the keystream generated during the decryption of the last 3 bits of the 4th byte c_3 do NOT depend on these bits of c_3 is very high, about* 0.75.

cf. eprint.iacr.org/2009/137:

Explanation: This probability is surprisingly high, as shown by our computer simulations. (even when the full 8 bits of c_3 are variable, this probability is still very high, about 1/17). All this is due to the very bad properties of the Boolean functions used in

Consequence

Fact 5.2.1 *If we fix the card nonce* n_T *and a 29-bit prefix of* C*, with probability about 0.75 over* C *the 9 bits of the keystream generated in this process are constant simultaneously for 8 different encryptions. of* C *that share the same 29-bit prefix and vary the last 3 bits of* c_3.

The Differences

Fact: All newly generated bits can be written as a fixed (known) linear function of (unknown) previous state bits AND the new 3-8 keystream bits that, though may be independent on the ciphertext (with proba 0.75 … 1/17) but remain unknown.

The new bits are linear in BOTH types of variables.

⇒ Just take the difference of any bit for two different decryptions. The unknown keystream bit will cancel.

⇒ The difference is a linear function of the previous state bits. So what?

⇒ The difference of previous state bits is always 0: we consider 8..256 encryptions with the same prefix of 29..24 bits in the ciphertext to be decrypted.

Therefore:

Fact 5.3.1 *If we assume that the Crypto-1 keystream generated during the decryption of the 4th byte c_3 is constant and does NOT depend on this byte, then the difference of the state of the cipher at any moment during the computation of ks2 and ks3 is a fixed multivariate linear function that depends on the differences in c_3 and nothing else.*

The Differences

Can be represented by a linear function 3->51 bits.

It can be computed given the linear feedback of the LSFR.

Real-life examples:

```
00000001 8DC1B21F6E10
00000002 1B83643EDC20
00000004 3706C87DB840
```

Remark:

This property is strong (7-255 differences on 51 bits)

It is SO STRONG that the key can be found by hand… Just a lot of information about the internal state…

Similar to differential cryptanalysis of DES…

An exceptional property, but once found, the security collapses totally.

Remark: Connects very well with algebraic cryptanalysis: properties of a cipher, can be exploited automatically with constraint satisfaction techniques [SAT solvers etc.]. Here it is easy that it could be done by hand. But the attack + technique applies to any stream cipher with "bad' Boolean functions. It will just break it.

Origin of this

Hard to believe, due to the spectacular nature of weakness of Boolean functions used in the cipher:

- With probability 10/16 the output of the combined Boolean function on 20 bits does not depend on the last 4 bits.
- In addition, with probability 3/4 the sub-function that deals with the last (and the latest) 4 bits, does not change if we flip the last bit, and with probability 1/2, it does not change when we flip the second last bit,
- and with probability 1/2, it is always 1 whatever is the change to the last 2 bits.

Running the Attack

1. Fix the card nonce through precise timing. Get one reply. P = 1/256.

We need on average 128 queries.

1. Now vary the last 3 bits (29..31) of the Encrypted Reader Cryptogram (ERC) == $\{n_R\}$ == first 4 bytes of the spoof cryptogram sent.
2. Also vary the 5 parity bits that can change, check all cases (exactly one replies).

We do on average $2^5/2$ trials times 2^3 cases.

Running the Attack

4. Repeat the whole* for about 1/0.75 times on average.

*Until the event happens
[otherwise the key found is not correct, or we get a contradiction].

TW. Next time the Step 1 (get one reply) is cheaper:
just change 3rd+4th byte of ERC=={n_R},
replies with P=1/64,
32 attempts on average.

Analysis

How many queries total?

128 +
(1/0.75-1)*32+
(1/0.75-1)*8*16
=

300 (on average)

Data -> Key Recovery

We use 8 replies x 4 bits, plus the information that comes from Parity (only at the end)

5. Now with probability about 0.75, we can simultaneously predict the differences of the states for all the 8 encryptions (cf. Fact 5.2.1 and Fact 5.3.1).

Data -> Key Recovery

One half:

6. Now we use the fact that the combined Boolean function of Crypto-1 reuses most state bits after 2 steps. Thus exactly (and only) 21 state bits determine the two keystream bits bits $(ks3)_0$ and $(ks3)_2$. We will examine all the 2^{21} cases and for each ciphertext where the card have answered we can divide the size of our space by 4. With 8 answers we will determine about 2^5 possible values for the 21 bits.

Data -> Key Recovery

Other half:

7. In the same way, we will determine 2^5 possibilities for the other 21 bits of the state that determines bits $(ks3)_1$ and $(ks3)_3$.

Data -> Key Recovery

Combine + check parity:

8. Then we have a list of 2^{10} states on 42 bits, which we need to extend to 2^{16} possible states on 48 bits.

9. Then by simple roll-back we get about 2^{16} possible initial keys, and checking all the $8 \cdot 8$ parity bits involved in the attack allows to know which key is correct with near certainty.

   Or if we find a contradiction at any stage, this means that the keystream does depend on c_3, contrary to the assumption in Fact 5.2.1.

Data Complexity

300 queries on average.

Computation:

- No precomputation.
- Running time:
  about $C*2^{21}$, instant on a PC.

- More than 10x better than any other attack…

My Attack in Practice

For now it takes 5-10 minutes per sector.

Should take 10 seconds with Proxmark3

Problems:

- communication errors
- hard in fact to fix the nonce…

Diversified Keys
=>
Nested Attacks

Nested Attacks

Nijmegen Paper

3. Is a Time/Data/Memory Tradeoff. As expensive as brute force but done only once to pre-compute 384 Gb of data.
   - Then they can clone the card with 4000 queries.Here the nonce does NOT need to be fixed. Takes 2 minutes? Should be 2 seconds? Is there a mistake in the paper?

4. Uses the fact that if you know one key, one other key be recovered instantly.
   - Another bug, out of scope for now. Easy!

Nested Authentication Attack – 1

- Assume the attacker knows 1 sector key
- The first nonce n_T^0 is sent in clear text. After successful authentication, the next nonce, n_T is sent encrypted as $\{n_T\}$.
  - Attacker computes $\mathrm{suc}^i(n_T^0)$ for i close to δ, where δ is the estimated distance between the two nonces.
  - Attacker can further reduce narrow the possibilities using 3 bits of information from parity bits
    - Because parity bits are computed over the ciphertext.
      - This gives 3 linear equations on the keystream.
  - In practice the card nonce can be known with certitude.

*****in their paper:

- For j = {0,1,2}, we have,

$$n_{T,8j} \oplus n_{T,8j+1} \oplus \cdots \oplus n_{T,8j+7} \oplus n_{T,8j+8} = \{p_j\} \oplus \{n_{T,8j+8}\} \oplus 1$$

- Where $\{n_{T,i}\} := n_{T,i} \oplus b_i$
- Since the attacker observes $\{p_j\}$ and $\{n_{T,8j+8}\}$, it gives him 3 bits of information about n_T
- Also, lets define the distance between 2 nonces formally as:

$$d(n_T, n'_T) := \min_{i \in \mathbb{N}} \mathrm{suc}^i(n_T) = n'_T.$$

Nested Authentication Attack – 3

- So using the distance estimation and the information from parity bit, the attacker can accurately guess the nonce and recover 32 bits of keystream.
- An attack, using the fact that only odd-numbered places of LFSR are used in the filter function, can be used to recover 2^{16} possible keys.
  - Direct inversion attack.

Nested Authentication Attack – 4

Use a very similar “inversion” method as in Malaga paper.

- This attack works in 0.05 s with about 64 bits of keystream.
- But here we have only 32 bits of key stream. A bit more difficult.

$\Rightarrow$ Gives 2^{16} possible keys.

$\Rightarrow$ Done twice, and intersection with 1-2 keys.

Combined Attacks (ours + Nijmegen)

Best Attack in Practice

Use my attack for one sector.

Then use Nested Authentication attack [Nijmegen Oakland paper] for other sectors.

- 10 minutes with my current equipment.
- Should take 10 SECOND TOTAL with Proxmark3. (all keys, all sectors).
  - Proxmark3 can then directly be used to act as a clone.

So How Secure is the Oyster Card?

-- what I can say now --

Security

Making cards [slightly…] harder to attack
Diversify all keys for each sector.

- Done for every Oyster card
- Not done in many other countries, examples:
  - In Kiev, Ukraine, the first block uses the default Infineon key A0A1A2A3A4A5…

Security

More examples:

– In Warsaw, Poland, the first block uses the default Philips key FFFFFFFFFFFF, then keys are NOT random, for example many start with 898989, some end with 898989…

⇒ In Poland everything is explained by history. Let me decrypt this one for you:

⇒ This month we celebrated the day when 20 years ago
– in 1989 –
Poland broke free from communism!

Security

Making cards [slightly…] harder to attack

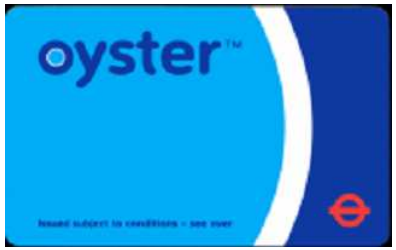
Diversify all keys for each sector.

Caveat:

Because of this Philips recommends to leave one sector encrypted with a default key…

BAD BAD LUCK:

- this makes it clonable in 2 seconds [Nijmegen Nested Attack].
  - not for Oyster though [all keys diversified]



Data Inside a Card

Keys are diversified, different key for each sector of 4 blocks.

Example of what is found inside:

Block 000: Data: CA1234BD518804004755745461502307 ..4.Q...GUtTaP#
Block 001: Data: 964142434445464748494A4B4C4D0101 .ABCDEFGHIJKLM.
Block 002: Data: 00000000000000000000000000000000 ....
Block 003: Data: 0000000000007F07889900000000000 ....

sometimes block 2 is FFFFFF……. no apparent reason

More…

Block 004: Data: 019CD4161F9B00000140A82D09805522 .....@.-..U"
Block 005: Data: C040803408202000840354000B26E82 .@.4. ...T...n.
Block 006: Data: 604180140A20200026045401 00D2F49F `A... .&.T.....
Block 007: Data: 00000000000067878999000000000000 .....g.....

Note: this sector has different access conditions than any other.

Then we store the travel history (>20 entries)

….
….
….

Block 060: Data: 00000000000000000000000000000000
Block 061: Data: 00000000000000000000000000000000
Block 062: Data: 00000000000000000000000000000000
Block 063: Data: 0000000000007F07889900000000000

TFL Public Claims

TFL [let's forget about
a lot of denial before that]
claimed more recently that

- PRIVACY: No personal data is stored in the card.[in Belgium they are…]
- SECURITY: Online database prevents fraud…

Claims - Privacy

- PRIVACY: No personal data is stored in the card.
  - [in Belgium they are… see UCL Belgium Python script]
  - obviously true, for anonymous cards purchased at the counter… [most people], anonymous passes are NOT allowed in Lisbon, Moscow,Helsinki, Warsaw@2010etc…
  - BUT WHY the history stored in the card has >20 entries???
    In Paris [Calypso-based "Navigo" system]:
    - they store 3 entries, judged "the minimum necessary for the purpose of control"
  - BTW: each card has a unique UID + unique number printed on it, that the reader can compute from the card data (block ?). These allows to trace people…

Claims - Security

TFL claimed that

- SECURITY: Online database prevents fraud…
  - Well does it???
- Only if the reader is online…
  - Otherwise free rides are possible, simplest attack: reset to the previous state, no need to know how data inside the card are encoded and managed…

Summary

- We broke >1 billion smart cards covering 70 % of the contactless badge/ticketing market.
- Our attack is more than 10 times better than any other attack…

- Security of many buildings (banks, military, UK Cabinet Office) is badly compromised.
- Security of many transport [metro,bus] and parking cards worldwide is badly compromised.
- Property and important assets [e;g. government and financial data] are directly under threat.