

Some slides contains animated elements that are not available
in the PDF format

2 Key Diversification

2.1 Construction

For diversification, the recommended way by NXP is to use the CMAC construction of an amount of data using a master key. See [CMAC].

The pre-requisite is that there is enough input "diversification data" in order to make it a MAC. A MAC is used rather than encryption to make it a one-way function.

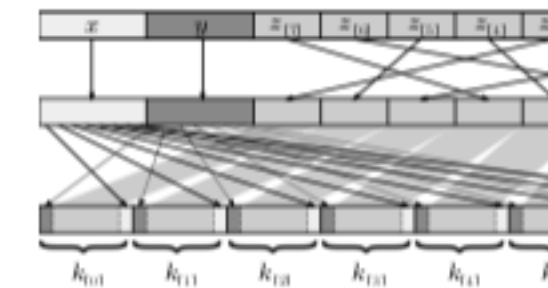
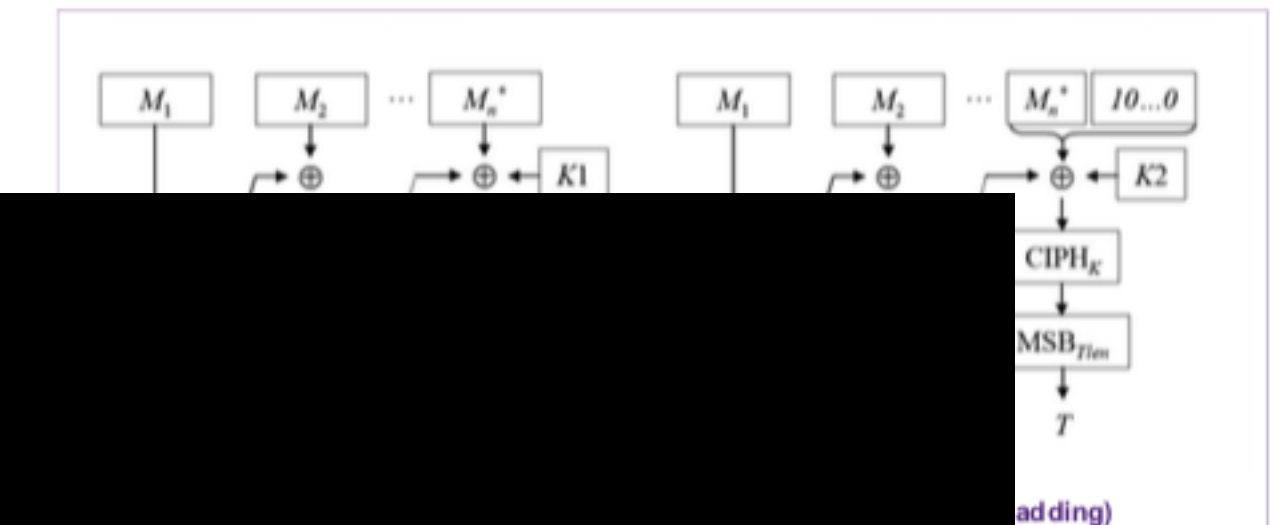


Fig. 2.5. Schematic representation of the

Remark 3. The DES implementation used in iClass NIST standard [12] in the way of representing keys. . . DES key is of the form $\langle k_0 \dots k_6 p_0, k_7 \dots k_{13} p_1, \dots, l \dots \rangle$ are the actual key bits and $p_0 \dots p_7$ are parity bits. l is of the form $\langle k_0 \dots k_{55} p_0 \dots p_7 \rangle$.

The following sequence of definitions describe the function f included here for the sake of complete

iClass Key Extraction - Exploiting th

In Circuit Serial Programming (ICSP) Command S		
CSP Data/IC Instr.	18F452 PIC Assembly Code	Comment
0xE0	MOVLW, 0	Set Upper byte of Index I
0xEE	MOVWF, FSR0H	
0xE0	MOVLW, 0	Set Lower byte of Index I
0xEE9	MOVWF, FSR0L	
0xEE	MOVWF, POSTINCO	Read File Register & Incr
0xEF5	MOVWF, TABLAT	Move Reg data to ICSP Re
reg Data	N/A	Send data byte read to IC

Circuit Implementation

required to extract the iClass register information is fairly it is comprised of a generic 8-bit microcontroller which is RS-232 transceiver, a couple of push buttons and a couple :SP interface and a PC serial COM port. The microcontrol

Authors Suppressed Due to Excessive Length

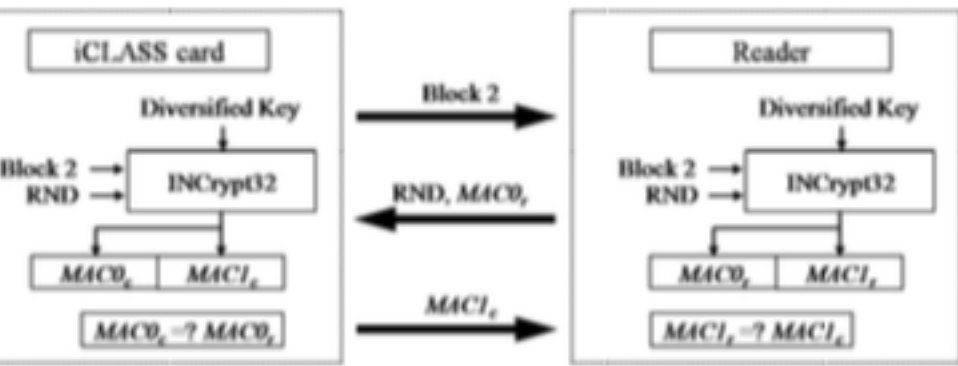


Fig. 1. Authentication protocol

MAC. At this point, the card can compute an 8-byte MAC in the way. If MAC_0 is correct, the card sends MAC_1 (that enables the reader to verify the authentication protocol needs to perform the MAC computation on the data.

Protocol If the authentication data blocks without an additional 8-byte data in a data block. The write protocol is

Breaking the HID iCLASS Standard

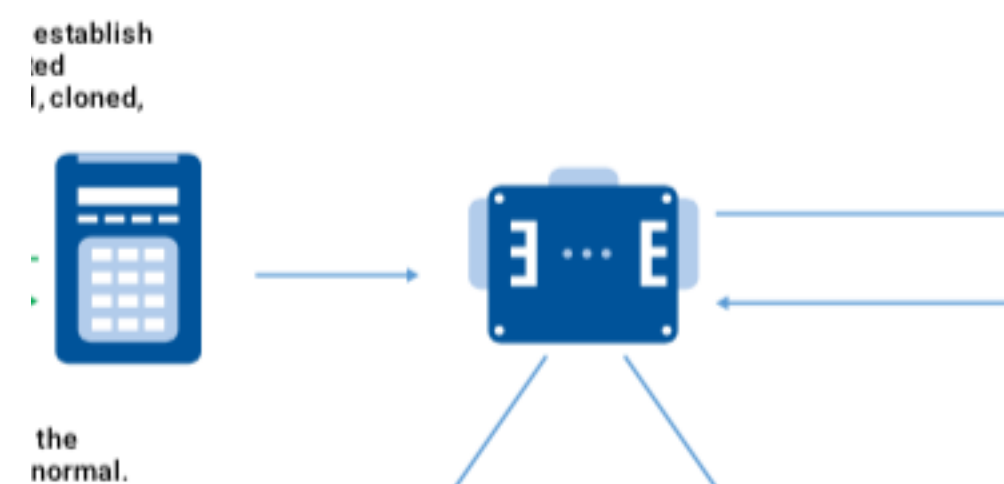
By Michael Cahyadi

iCLASS Compatibility

	Card Compatibility	R10, R30, R40, RK40	RW300, RW400, RWK400, RWKL550, RWKB575	FIPS 201	R10A Transit
ISO 15693	Card CSN Read	2K & 16K	2K & 16K	Infineon MyD™, TI Tag-It, Philips I Code®	Infineon MyD™, TI Tag-It, Philips I Code®
	Card Read	2K & 16K	2K & 16K	2K & 16K	2K & 16K
	Card Write	NONE	2K & 16K	2K & 16K	NONE
ISO 14443A	Card CSN Read	Philips MIFARE, UltraLight, DESFire™	Philips MIFARE, UltraLight, DESFire™	NONE	NONE
	Card Read	NONE	NONE	DESFire™	NONE
	Card Write	NONE	NONE	DESFire™	NONE

How a Credential is 'Read'

is Employing Mutual Authentication



The result $k_1^{\oplus} = 78$ comes from a modulo operation. Here input z_6 is taken modulo 62, which is 111110 in binary. Example for $k_1^{\oplus} = 0x78$:

$$z_6 = 111100, (z_6 \bmod 62) + 2 = .111110.$$

$$z'_6 = 111110, ((z'_6 \bmod 62) + 2) = .000010. \oplus$$

$$01111000 = 0x78$$

Then, 3% of the output variations invoked by bitflips in $z_{6[1..6]}$ describe a relation $z_6 + 1 = z_7$. The corresponding k_1^{\oplus} is obtained by taking $k_{1[1..6]} = 1$ when the relation holds and $k_{1[1..6]} = (z_6 \bmod 62) + 2$ when it does not hold. Example for $k_1^{\oplus} = 0x4e$:

$$z_6 = 100100, (z_6 \bmod 62) + 2 = .100110.$$

$$z'_6 = 100110, ((z'_6 \bmod 62) + 1) = .000001. \oplus$$

$$01001110 = 0x4e$$

Eventually, the function for $k_{1[1..6]}$ is:

$$k_{1[1..6]} = \begin{cases} 1, & (z_6 \bmod 62) + 1 = (z_7 \bmod 63); \\ (z_6 \bmod 62) + 2, & \text{otherwise.} \end{cases}$$

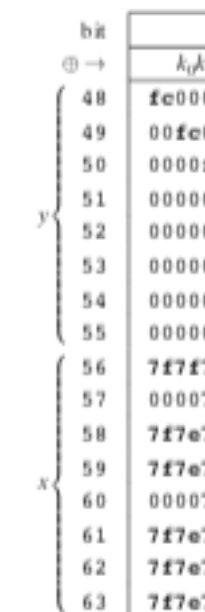


Figure 9: OR :

Keywords: Access control, RFID, contactless smart cards, MiFare Classic secrets, secure hardware devices, reverse-engineering, electronic backdoors, critical application development management,

Abstract: MiFare Classic is the most popular contactless smart card worldwide. At Esorics 2008 Dutch researchers showed that the little as 0.1 seconds if the attacker can access or eavesdrop on the card. We discovered that a MiFare classic card can be cloned in the attacker only needs to be in the proximity of the card for a few seconds. This is a serious security vulnerability as the ability of identity through pass cloning feasible at any moment and sitting next to the victim on a train or on a plane is now becoming a reality. Also (independently from us) discovered this vulnerability through queries to the card and does not require any precomputation. Clones of MiFare Classic are even weaker, and can be cloned. The main security vulnerability that we need to address with MiFare Classic is the lack of mutual authentication. MiFare Classic security, RFID protocols and software vulnerabilities. It is a serious security vulnerability that we need to address with MiFare Classic. The economy is vulnerable to sophisticated forms of electronic cloning, and financial institutions worldwide.

Everything done in this presentation is done for
educational purposes only

Information regarding **classified proprietary information owned by HID** will be
redacted

RFID Security (1st Edition)
ISBN 1-59749-047-4
Syngress, 2005

RFID is

RFID Security (1st Edition)
ISBN 1-59749-047-4
Syngress, 2005

Radio Frequency Identification

Radio Frequency Identification is used in

RFID Security (1st Edition)
ISBN 1-59749-047-4
Syngress, 2005

E-Passports



Radio Frequency IDentification is used in

RFID Security (1st Edition)
ISBN 1-59749-047-4
Syngress, 2005

E-Pasports



Payment Cards



It comes in different flavors

Name	Frequency	Reach	ISO Terminology
Low Frequency (LF)	125 kHz	<45cm	Proximity Cards (ISO 7810)
High Frequency (HF)	13.56 MHz	45-100cm	Contactless Cards (ISO 24727)

But we are focusing on one

HID Proximity Brochure
HIDGlobal, 02 Oct 2018

iCLASS Product Brochure
HIDGlobal, 2007

Smart Contactless Card



- Uses 13.56 MHz RFID Frequencies
- Encrypted using DES/Triple-DES (Digital Encryption Standard)
- the CSN is protected

Before iCLASS there was ProxCard that transmits it's UID in plaintext

Proximity Card (ISO 7810)



6d 61 69 6b 20 73 61 79 61 6e 67 20 73 61 6e 73 61 6e



Reader



*this is **not a real UID**

So iCLASS uses...

Cryptography Engineering
ISBN 978-04704744242
Bruce Schneier, 2010

Encryption

**HID iCLASS uses the Hash0 algorithm
to create all signatures**

Using the signature in the reader, the device
can iterate unique keys for every card

5176 0400 0002 5018

Using the We could attempt to bruteforce the combination but this method requires significant computing power if done without prior information gathering and supplemental information in the reader

```
[00:00:00] 8 keys tested
```

```
Current passphrase: K30g8ng3B98bCSU5434o
```

```
Master key      : 0K 2X 08 96 1A 64 99 63 4D 2B 40 4E 6R 82 0D 61  
                : 8D 8E 54 86 67 6A 62 6K 63 6D 8L 8R 89 0Z 0Z 6D
```

```
Transient key   : 15 28 70 06 0A 6A 9Y 65 6Z 4U 10 7P 0X 8X 8R 53  
                : 4S 50 4M 7M 2X 6Q 8R 1Q 68 8Y 8G 02 0C 4W 98 5Y  
                : 98 44 4E 60 4X 00 2R 4S 63 09 6D 4I 69 4A 29 46  
                : 0K 0H 1Y 86 13 4B 58 89 46 9B 0A 5B 56 8Q 27 08
```

It also has a unique quirk

Smart Contactless Card



Reader



Reader checks if the key in the card and the key in the reader are the same



If the keys are the same

Smart Contactless Card



Reader



The CSN is transmitted and authenticated

Smart Contactless Card



Reader



So there are layers to this...

Level	Layers of Security
Lvl0	CSN
Lvl1	phdr + rf signal (IDENTIFY)
Lvl1	DES Signature (Q) #N Data
Lvl2	INCrypt32 #0 data
Lvl2	Hash0 #0 data
Lvl3	TDES Key (R,S) #N data

Smart Contactless Card



So we just have to

Smart Contactless Card



1. Extract the CSN
2. Extract the TDES Key
3. Duplicate the card

Smart Contactless Card



First Step
Extracting the CSN

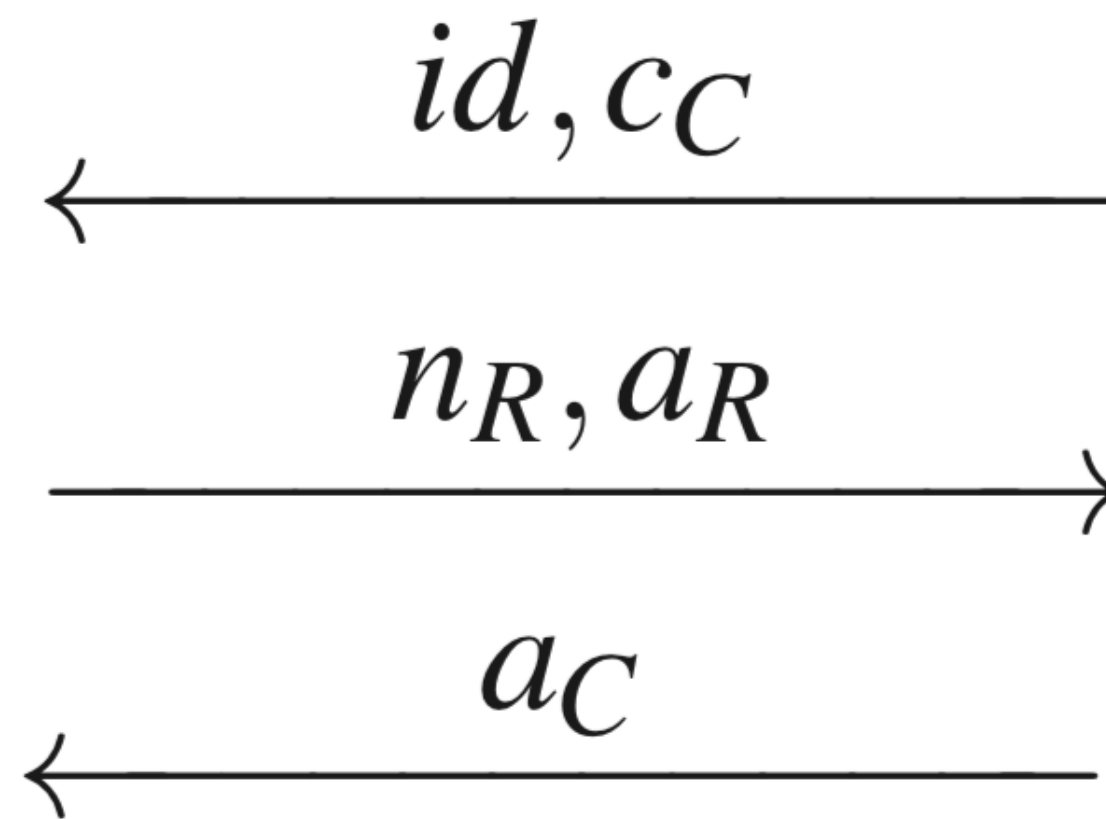
On the left is a device called an OMNIKEY, it's what HID calls a Secure Access Module that can read and write the contents of an HID iCLASS card. This is commonly used by technicians for the initial programming of the card.

OMNIKEY 5321 Product Brochure
HIDGlobal, 2007

Omnkey



Smart Proximity Contactless Card



Reader/Writer Multikey



- It can do read-write operations
- It's contents are encrypted with TDES too
- The USB connection to a PC is secured using somesort of **Secure Mode**

Reader/Writer Multikey



In the **Secure Mode**, there is a security flaw in an old driver that accidentally gives the **user root access of the device**

```
executable.948.exe — Binary Ninja
int32_t sub_4075d0()
004075f0 push    edi
004075f1 push    ecx
004075f2 lea    edi, [ebp+0xfffffe44] {var_1c0}
004075f8 mov     ecx, 0x6c
004075fd mov     eax, 0xc0000000
00407602 rep    stosd dword es:[edi], eax {var_1c0}
00407604 pop     ecx
00407605 mov     dword [ebp-0x10 {var_14}], ecx
00407608 lea    ecx, [ebp-0x14] {var_18}
0040760b call   sub_4dcf67
00407610 mov     dword [ebp-0x4 {var_8}], 0x0
00407617 mov     ecx, dword [ebp-0x10 {var_14}]
0040761a call   sub_4010ff
0040761f push   0x5c8194 {"80A60000"}
00407624 mov     ecx, dword [ebp-0x10 {var_14}]
00407627 add     ecx, 0x3c8
0040762d call   sub_4893ed
00407632 mov     ecx, dword [ebp-0x10 {var_14}]
00407635 call   sub_401055
0040763a mov     ecx, dword [ebp-0x10 {var_14}]
0040763d call   sub_4010ff
00407642 push   0x5c8174 {"808200F008"}
00407647 mov     ecx, dword [ebp-0x10 {var_14}]
0040764a add     ecx, 0x3c8
00407650 call   sub_4893ed
00407655 mov     ecx, dword [ebp-0x10 {var_14}]
00407658 call   sub_401055
0040765d mov     ecx, dword [ebp-0x10 {var_14}]
00407660 call   sub_4010ff
00407665 push   0x5c8168 {"808800F0"}
0040766a mov     ecx, dword [ebp-0x10 {var_14}]
0040766d add     ecx, 0x3c8
00407673 call   sub_4893ed
00407678 mov     ecx, dword [ebp-0x10 {var_14}]
0040767b call   sub_401055
00407680 mov     ecx, dword [ebp-0x10 {var_14}]
00407683 call   sub_4010ff
00407688 push   0x5c8158 {"8080000000"}
Options Selection: 0x407642 to 0x407647 (0x5 bytes) PE Disassembler
```

Master Authentication Key

By disassembling the original OMNIKEY firmware, we can find the default master authentication key that is contained in every single OMNIKEY device

Reader/Writer Multikey



We can use that masterkey to create a program that can read the card and acquire the CSN from the program header, without the use of HID's proprietary card decoding software

```
MINGW32: ~/iclassified
Administrator@user-09118738a3 ~/iclassified
$ iclass.exe read
fcee5601f7ff12e0
12ffffff7f1fff3c
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
030303030003e017
2ed49bc872bed6d4
2ad4c8211f996871
2ad4c8211f996871
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
ffffffffffffffff
Administrator@user-09118738a3 ~/iclassified
$ iclass.exe write 7 4141414141414141
SUCCESS
Administrator@user-09118738a3 ~/iclassified
$ iclass read 7
4141414141414141
Administrator@user-09118738a3 ~/iclassified
$ _
```

CSN
(Card Serial Number)

Reader Parity Data

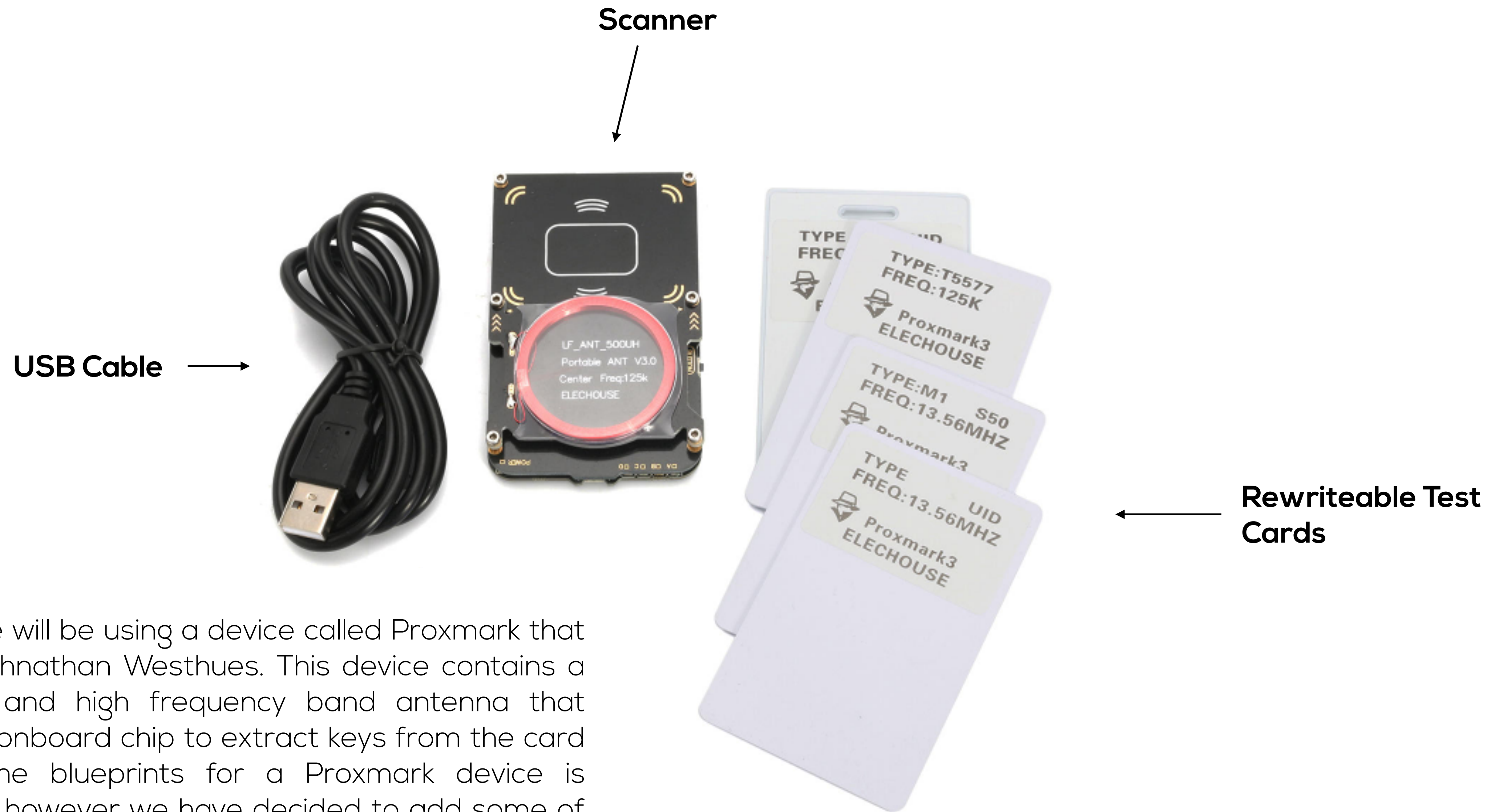
Time to go deeper...

Level	Levels of Security
Lvl0	CSN
Lvl1	phdr + rf signal (IDENTIFY)
Lvl1	DES Signature (Q) #N Data
Lvl2	INCrypt32 #0 data
Lvl2	Hash0 #0 data
Lvl3	TDES Key (R,S) #N data

Smart Contactless Card



Second Step
Extract the TDES Key



For this step we will be using a device called Proxmark that was built by Johnathan Westhues. This device contains a low frequency and high frequency band antenna that connects to an onboard chip to extract keys from the card and reader. The blueprints for a Proxmark device is available online, however we have decided to add some of our own modifications such as lowering the memory capacity to lower the production cost and speed up production.

In this step we use the Proxmark device to fool the reader into thinking that the device is a genuine HID card. This is done to take the frequency response that is created by the reader.

iCLASS Key Extraction
Jonathan Westhues, 2010

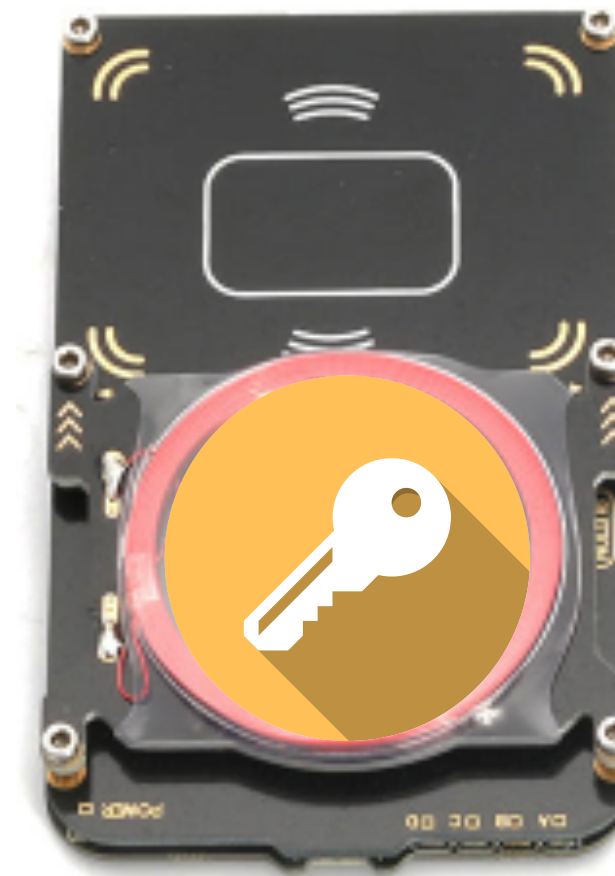
Reader



IDENTIFY



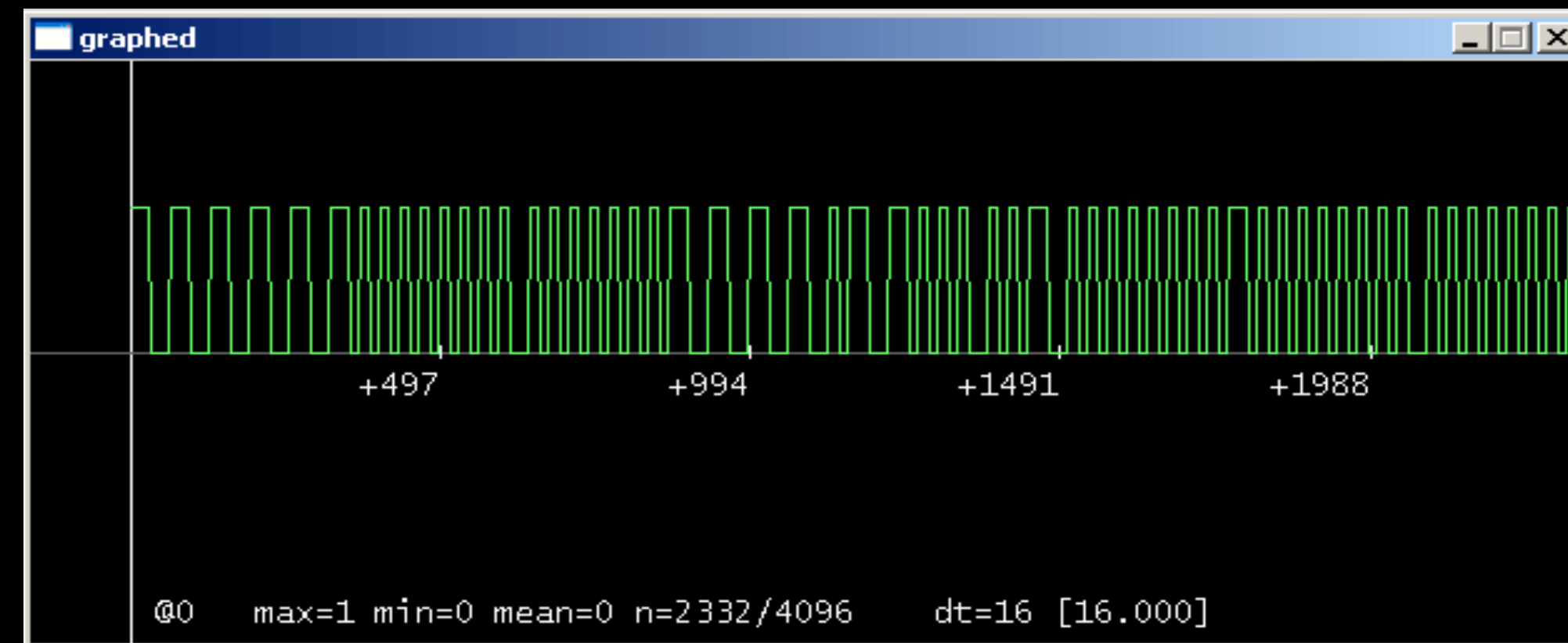
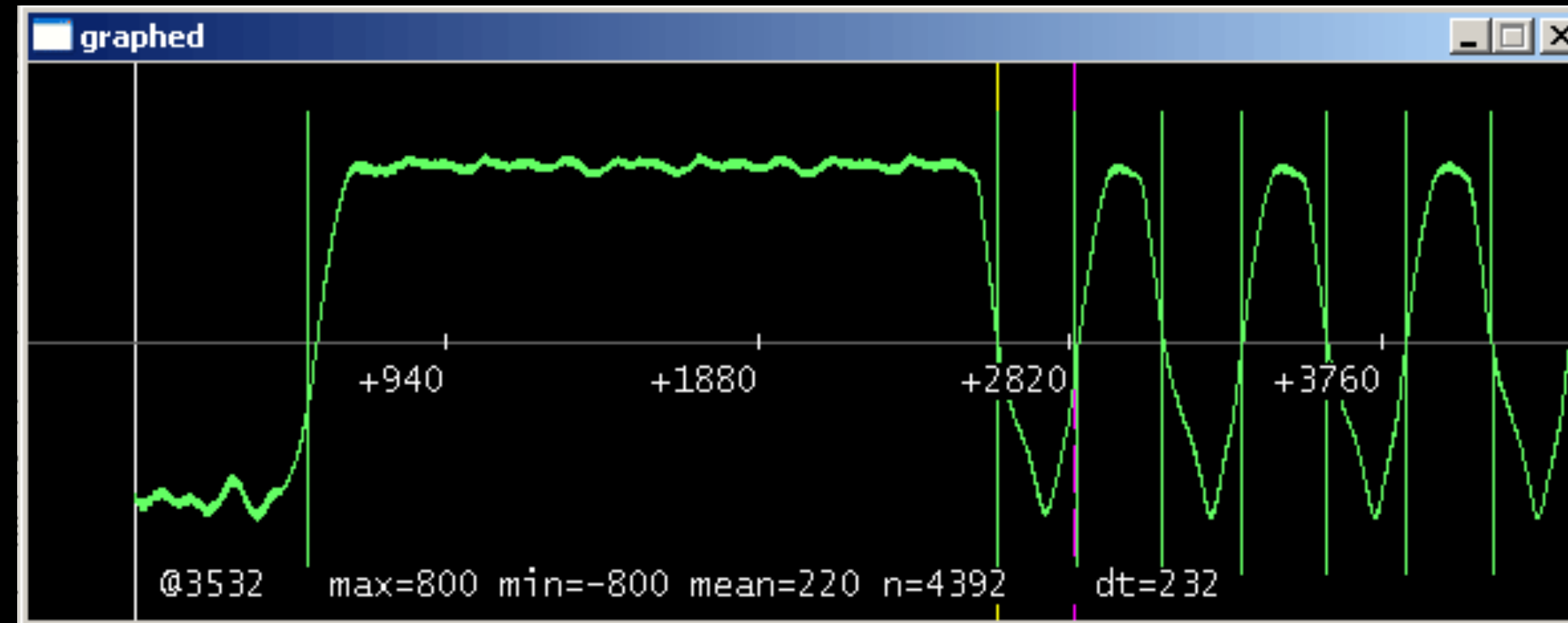
Proxmark 3



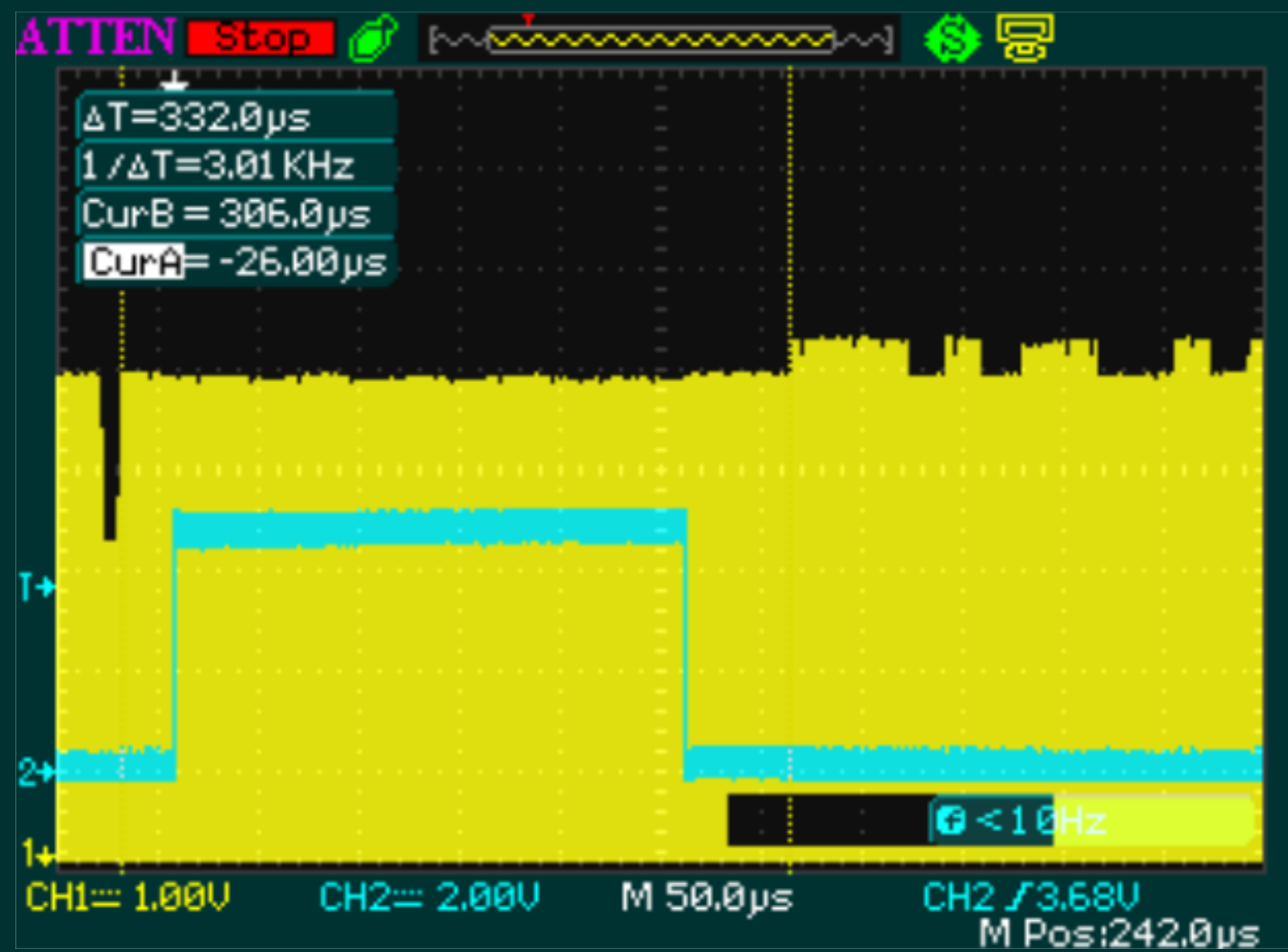
Key DES



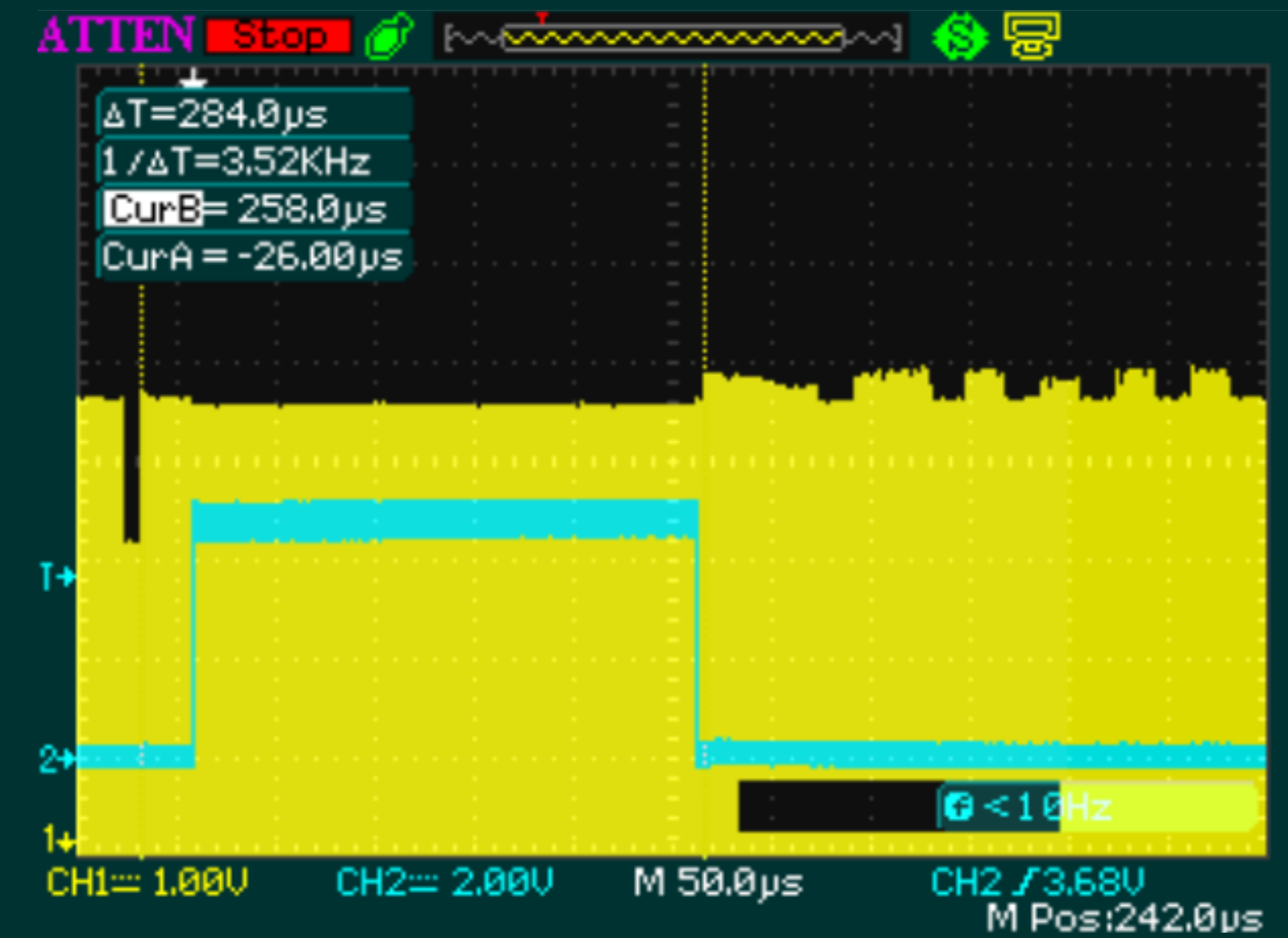
We need to match the proxmark radio frequency with the frequency that is expected of a genuine HID iCLASS card



Response of an **"IDENTIFY"** signal from a genuine reader



Response of an **"IDENTIFY"** that's replicated



Next on the chopping block

Level	Levels of Security
Lvl0	CSN
Lvl1	phdr + rf signal (IDENTIFY)
Lvl2	DES Signature (Q) #N Data
Lvl2	INCrypt32 #0 data
Lvl3	Hash0 #0 data
Lvl3	TDES Key (R,S) #N data

We will attempt to break the implementation of the algorithm used by HID. In the iCLASS system there is a formula to calculate the keys and in the keys there are several parameters.

Publicly shared :

P, u, b, G (parameters)

Q = Public key

e = Hash of data

R = Signature by scalable multiplication

S = Signature by normal numbers

Privately stored :

M = random number

Kd = debit key

Level	Levels of Security
Lvl0	CSN
Lvl1	phdr + rf signal (IDENTIFY)
Lvl1	DES Signature (R,S) #N Data
Lvl2	INCrypt32 #0 data
Lvl2	Hash0 #0 data
Lvl3	TDES Key (Kd) #N data

Signature is a pair of variables R , S that's programmed as :

$$R = (mG)_x$$

$$S = \frac{e + kR}{m}.$$

The m variable needs to be random for the encryption system to work. If a *signature* uses the same m value, a *user* can calculate k .

There are two S Variables (one for the reader and the other for the card) so there are S_1 and S_2

$$R = (mG)_x \quad R = (mG)_x$$
$$S_1 = \frac{e_1 + kR}{m} \quad S_2 = \frac{e_2 + kR}{m}$$

$$R = (mG)_x \quad R = (mG)_x$$
$$S_1 = \frac{e_1 + kR}{m} \quad S_2 = \frac{e_2 + kR}{m}$$

If m has the same value in the two signatures, then R will be the same

$$S_1 - S_2 = \frac{e_1 - e_2}{m}$$

$$m = \frac{e_1 - e_2}{S_1 - S_2}$$

$$k = \frac{mS_i - e_i}{R} \left[= \frac{e_1S_2 - e_2S_1}{R(S_1 - S_2)} \right].$$

With this we not only took the DES signature but we also obtained the TDES key which is the last level of security in the iCLASS ecosystem.

Level	Levels of Security
L 0	CSN
L 1	phdr + rf signal (IDENTIFY)
L 2	DES Signature (Q) #N Data
L 2	INCrypt32 #0 data
L 3	Hash0 #0 data
L 3	TDES Key (R,S) #N data

c:\Proxmark\win32>



Smart Contactless Card



Step 3

Replication

```
pm3 --> hf tune
```

```
Measuring antenna characteristics, please wait...
```

```
#db# DownloadFPGA(len: 42096)
```

```
....
```

```
# LF antenna: 35.20 V @ 125.00 kHz
```

```
# LF antenna: 33.83 V @ 135.00 kHz
```

```
# LF optimal: 35.71 V @ 127.66 kHz
```

```
# HF antenna: 30.77 V @ 13.56 MHz
```

```
Displaying LF tuning graph, Divisor 89 is 134khz, 95 is 125khz
```

```
pm3 --> hf iclass managekeys n 0 k aea684a6dab23278
```

```
#db# SUCCESS
```

```
#db# going into sicko mode, 8 CSNS sent
```

```
pm3 --> hf iclass eload iclass_tagdump-aa16230f8ff12f1.bin
```

```
Sent 42096 bytes of data to device emulator memory
```

```
pm3 --> hf iclass sim 0 e53d1d0efeff12e0
```

```
#db# Simulating CSN e53d1d0efeff12e0
```

Authors Suppressed Due to Excessive Length

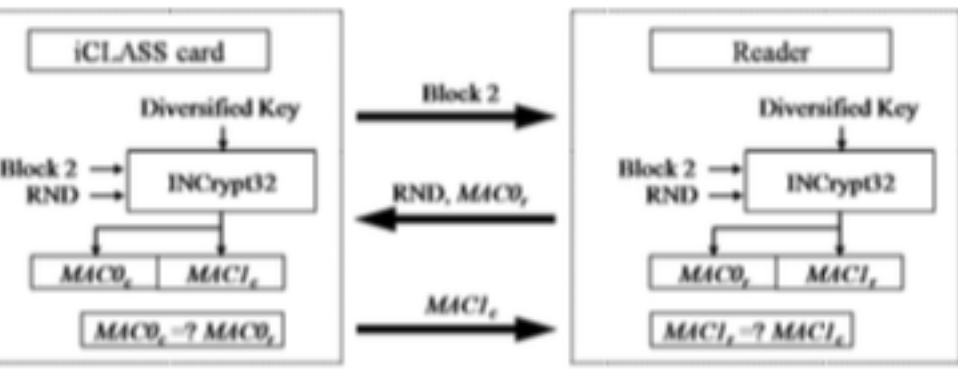


Fig. 1. Authentication protocol

MAC. At this point, the card can compute an 8-byte MAC in the way. If MAC0 is correct, the card will answer the other 4-byte signature (MAC1, MAC2) that enables the reader to authenticate the card. Therefore, the authentication protocol needs to perform INCrypt32 with 12-byte input and 8-byte data.

Protocol If the authentication protocol succeeds, the reader is able to read data blocks without an additional authentication procedure. However, to read an 8-byte data in a data block, the reader needs to perform INCrypt32 on the data. The write protocol is described in Fig. 2.



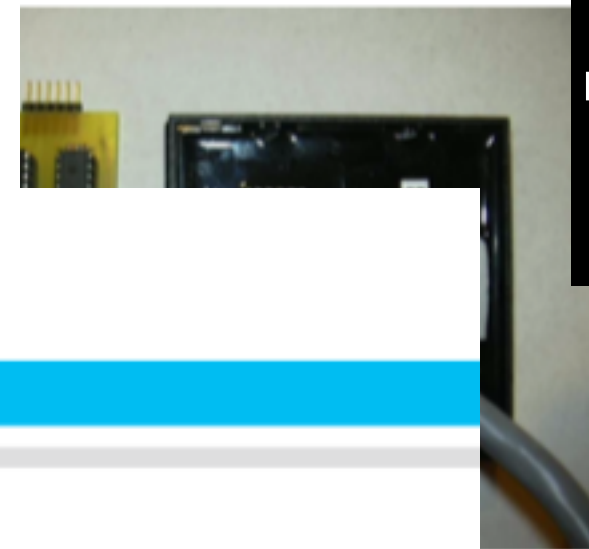
iCLASS Compatibility Chart

	Card Compatibility	R10, R30, R40, RK40	RW300, RW400, RWK400, RWKL550, RWKB575	
ISO 15693	Card CSN Read	2K & 16K	2K & 16K	Info
	Card Read	2K & 16K	2K & 16K	Phi
	Card Write	NONE	2K & 16K	
ISO 14443A	Card CSN Read	Philips MIFARE, UltraLight, DESFire™	Philips MIFARE, UltraLight, DESFire™	
	Card Read	NONE	NONE	
	Card Write	NONE	NONE	

In Circuit Serial Programming (ICSP) Command		
ICSP Data/ PIC Instr.	18F452 PIC Assembly Code	Comment
0x0 E00	MOVLW, 0	Set Upper byte of Index Register
0x6 EE9	MOVWF, FSR0H	
0x0 E00	MOVLW, 0	Set Lower byte of Index Register
0x6 EE9	MOVWF, FSR0L	
0x5 0EE	MOVWF, POSTINC0	Read File Register & Increment
0x6 EP5	MOVWF, TABLAT	Move Reg data to ICSP I/O
Reg Data	N/A	Send data byte read to PC

Capture Circuit Implementation

The circuit required to extract the iClass register information is fairly simple. It is comprised of a generic 8-bit microcontroller which is connected to an RS-232 transceiver, a couple of push buttons and a couple of LEDs. The microcontroller is connected to the ICSP interface and a PC serial COM port. The microcontroller is powered by the ICSP interface. The serial EEPROM is used to store the captured data across the ICSP interface. The RS-232 transceiver is used to communicate with the PC. The 9Vdc (min) battery is used to power the PIC ICSP interface into a dormant state. When the readers ICSP connector is connected to the capture circuit, the capture circuit receives its operating power directly from the readers ICSP connector. In addition, an onboard 5Vdc battery which is used to power the circuit during the capture process.



thank you



How a Credential is 'Read'

comes from a modulo operation. Here we take the modulo 62, which is 111110 in binary. Example: $(z_6 \text{ mod } 62) + 2 = \dots 111110$. $(z_6 \text{ mod } 62) + 2 = \dots 000010$. $01111000 = 0x78$. Output variations invoked by bitflips in the input $z_6 + 1 = z_7$. The corresponding k_1^{\oplus} is $k_{1[1..6]} = 1$ when the relation holds and 0 when it does not hold. Example: $(z_6 \text{ mod } 62) + 2 = \dots 100110$. $(z_7 \text{ mod } 62) + 1 = n_7 = \dots 000001$. $01001110 = 0x4e$. Relation for $k_{1[1..6]}$ is: $(z_6 \text{ mod } 62) + 1 = (z_7 \text{ mod } 63);$ otherwise.

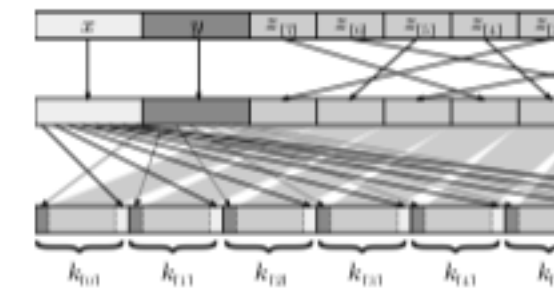


Fig. 2.5. Schematic representation of the DES implementation

Remark 3. The DES implementation used in iClass is based on the NIST standard [12] in the way of representing keys. A DES key is of the form $\langle k_0 \dots k_6 p_0, k_7 \dots k_{13} p_1, \dots, k_{14} \dots k_{20} p_7 \rangle$ where $k_0 \dots k_{20}$ are the actual key bits and $p_0 \dots p_7$ are parity bits. The parity bits are of the form $\langle k_0 \dots k_{55} p_0 \dots p_7 \rangle$.

The following sequence of definitions describe the functions used in the construction. The function $check$ is included here for the sake of completeness. The definitions are not necessary to understand the attack and Section 3.3.

Definition 8. Let the function $check: (\mathbb{F}_2^6)^8 \rightarrow (\mathbb{F}_2^6)^4$ where $ck: \mathbb{N} \times \mathbb{N} \times (\mathbb{F}_2^6)^4 \rightarrow (\mathbb{F}_2^6)^4$ is defined as

$$ck(l, z_0, \dots, z_7) = \tilde{z}_0, \dots, \tilde{z}_3$$

$$\tilde{z}_i = z_{[0] \dots z_{[3]} \oplus z_{[0] \dots z_{[3]}}$$

$$z_{[i]} = z_{[i] \dots z_{[i+1]}$$

$$z_{[i]} = \mathbb{F}_2^2 \times (\dots)$$

Definition 10. Define the bitstring $\pi \in (\mathbb{F}_2^8)^{35}$ in l $\pi = 0x0F171B1D1E272B2D2E333534D4E535556595A5C636566696$.

Each byte π_i is defined as

2 Key Diversification

2.1 Construction

For diversification, the recommended way by NXP is to use the CMAC construction of an amount of data using a master key. See [CMAC].

The pre-requisite is that there is enough input "diversification data" in order to make it a MAC. A MAC is used rather than encryption to make it a one-way function.

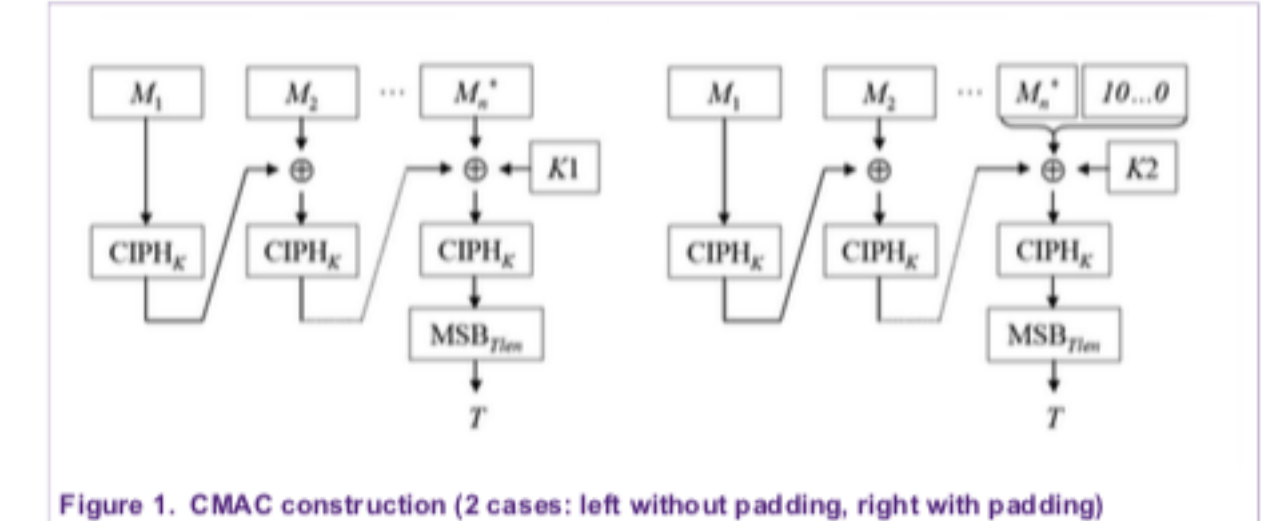


Figure 1. CMAC construction (2 cases: left without padding, right with padding)

Fig. 1 illustrates the CMAC construction in two cases.

According to [CMAC], the function $CIPH_K$ is modified before the last block (denoted $K1$ or $K2$) to ensure the choice of which key contains padding.

These computations are done in the context of the key because the padding is added to the data before the computations can be diversified. The signals to the CMAC are the keys $K1$ and $K2$.

THE DARK SIDE OF SECURITY: Access Control, RFID, Contactless Smart Cards, MiFare Classic Rail and Building Access

Nicolas T. Courtois, University College London, Computer Science, Gower Street, London WC1E 6BT, UK

hex	ASCII
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	A
59	B
60	C
61	D
62	E
63	F

Figure 9: OR

$\pi = \{ 01234567, 3567012, 34670125, 0135246, 03451363, 1267034 \}$

Keywords: Access control, RFID, contactless smart cards, MiFare Classic, security, secrets, secure hardware devices, reverse-engineering, electronic devices, backdoors, critical application development management, security.

Abstract: MiFare Classic is the most popular contactless smart card worldwide. At Esorics 2008 Dutch researchers showed that the card can be cloned in as little as 0.1 seconds if the attacker can access or eavesdrop on the communication. We discovered that a MiFare classic card can be cloned in the field only needs to be in the proximity of the card for a few seconds. The cloning of identity through pass cloning is feasible at any moment and is not dependent on the victim sitting next to the victim on a train or on a plane is now possible. We also (independently from us) discovered this vulnerability of MiFare Classic. Queries to the card and does not require any precomputation. The cloning of clones of MiFare Classic are even weaker, and can be cloned. The main security vulnerability that we need to address with MiFare Classic is the security of the protocols and software vulnerabilities. It is a symmetric key diversification economy is vulnerable to sophisticated forms of electronic cloning. Cloning can be done intentionally (or not), but quite easily in fact, cloning of financial institutions worldwide.

about the keys



Michael Cahyadi <maikxchd@gmail.com>

to Matthew ▾

Ok, I have hash1 down. No problem there.

but according to above $z0 = e9\ 11\ 52\ 2b\ 3f\ dd\ 8e\ e8$

but when i try $z0 = \text{des_ecb}(\text{key}, \sim\text{key}) = f2\ e7\ 33\ d7\ 35\ fe\ ae\ 2c$

and I thought $\text{Ksel}(i) = h(\text{hash1}(\text{id})(i))$ (where $h = \text{hash2}(\text{Kcus})$)

so I thought $00 = 8a, 1c = 0d \dots$

Is it possible to have another hash2 example, to figure out where Ive obviously gone wrong

Thanks

Reply

Forward

iceman

2019-04-18 03:22:14

Administrator



Registered: 2013-04-25

Posts: 5,343

[Website](#)

Online

Usually I make sure the t5577 is writeable and "detected" before I write the block 1-3 and write config block at the end. Dunno why you try the pwd in the writes.

When verifying, you need detect, then dump it again.

Which kind of tag is this? Never seen this config block before?

If you feel the love, <https://www.patreon.com/iceman1001>

`modhex(hkkehghthbhudcfdchkgiehgdueh)`

thank you



z0 was supposed to be permuted

but its not responding somehow

try turning the omnikey on

then off again

then on again

ok th

where can i go to bed now?

folio

YouTube

iclass se r10se

```
user@ubuntu:~/proxmark3
make -C tools/nonce2key all
make[1]: Entering directory '/home/user/proxmark3/tools/nonce2key'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/user/proxmark3/tools/nonce2key'
user@ubuntu:~/proxmark3$ git pull
Already up-to-date.
user@ubuntu:~/proxmark3$ clear
user@ubuntu:~/proxmark3$ ./client/flasher /dev/pn3-0 arsrc/obj/fullimage.elf
```

```
user@ubuntu:~/pm3_new/client
user@ubuntu:~/pm3_new/client$ ./proxmark3 /dev/pn3-0
```

special thanks for

- **Jonathan Westhues (Oregon, USA)**

Provided blueprints for the Proxmark 3 device

- **Flavio D. Garcia from Radboud University Nijmegen (Nijmegen, Netherlands)**

Provided the key diversification function for *HID iCLASS* cards

- **Christian Herman from IceSQL AB (Gotenburg, Sweden)**

Created a special firmware for the OMNIKEY and documented the schematics of the iCLASS reader

- **ELECHOUSE (Shenzen, China)**

For printing the Proxmark 3 Device

- **Kevin Chung dari New York University (New York, USA)**

Developed additional drivers for the OMNIKEY 5321 model and reverse engineered the *HID iCLASS* masterkeys

- **Fanny Soedjahdjo dari Philip Pocock Catholic Secondary School (Ontario, Canada)**

Provided additional material translation from english to indonesian
(this presentation was originally created in Indonesian)