A Tour through the Realms of Reverse Engineering

Ricardo J. Rodríguez

@RicardoJRdez ※ rjrodriguez@unizar.es ※ www.ricardojrodriguez.es

Universidad Zaragoza
1542

November 23, 2016

Legacy Systems
Room A.12

Outline

Agenda

Universidad Zaragoza 1542

Introduction to Reverse Engineering (I)

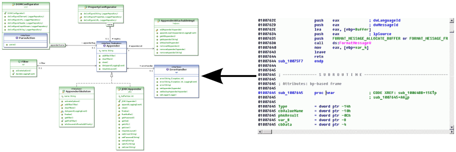
Reverse Engineering

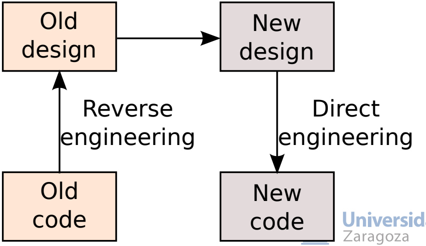
- Figure out how something works from an exhaustive analysis
- Improvement of legacy products/systems
- Different application domains
  - Hardware (legacy hardware)
  - Software (e.g. Samba)

Introduction to Reverse Engineering (I)

Reverse Engineering

- Figure out how something works from an exhaustive analysis
- Improvement of legacy products/systems
- Different application domains
  - Hardware (legacy hardware)
  - Software (e.g. Samba)
- Going backward in the development cycle





Introduction to Reverse Engineering (II): Motivation (1)

Motivation

- Interoperability
- Non-existent documentation
- Final product analysis
- Security audit
- Industrial (or military) espionage (e.g. II WW)
- Removal of anti-copy or limited use protections
- Creation of duplicates without license
- Academic
- Innate curiosity
- Learn about the errors of other people

Universidad Zaragoza

Introduction to Reverse Engineering (II): Motivation (2)

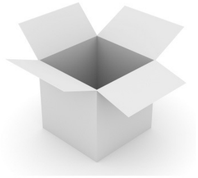
Find bugs in software

- Incorrect checking of buffer limits (buffer overflow)
- Use of data without previous validation
- Cyclic routines for input data
- Byte-level copy operations
- Pointer arithmetic based in user-input data
- “Trust" in security systems with dynamic inputs

Universidad Zaragoza

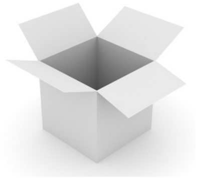
Introduction to Reverse Engineering (III): Approaches

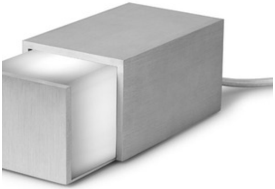
- White-box
  - *Full* knowledge (e.g., source code)
  - E.g.: *WhiteBox SecureAssistant*, *IDAPro*, *SourceScope*...



Introduction to Reverse Engineering (III): Approaches

- White-box
  - *Full* knowledge (e.g., source code)
  - E.g.: *WhiteBox SecureAssistant*, *IDAPro*, *SourceScope*. . .
- Gray-box
  - *Partial* knowledge
  - E.g.: Rational's Purify (use/consumption of memory), Valgrind

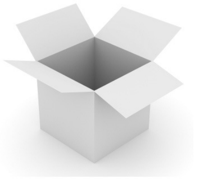


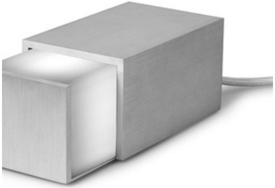


Universidad Zaragoza

Introduction to Reverse Engineering (III): Approaches

- White-box
  - *Full* knowledge (e.g., source code)
  - E.g.: *WhiteBox SecureAssistant*, *IDAPro*, *SourceScope*. . .
- Gray-box
  - *Partial* knowledge
  - E.g.: Rational's Purify (use/consumption of memory), Valgrind
- Black-box
  - *Null* knowledge
  - Analyse outputs of the system depending on different inputs





Reverse Engineering

Choose your side!

On hardware

- Reverse engineering of integrated circuits/smart cards
  - Low-level details
  - Physics and electronic knowledge (plus special hardware)
- Reverse engineering of devices
  - How system works?
  - How are the inputs and outputs?

On software

- Reverse engineering of protocols
  - How network layer works
- Reverse engineering of software
  - Low-level code analysis: assembly, calling conventions
  - Use of debugging/disassembler/decompiler tools
  - Programming

Agenda

Universidad Zaragoza 1542

Reverse Engineering of Protocols

- Message format
  - **Normally done by manual analysis**
  - Recent research on automatic analysis
    - Message clustering
    - Emulate protocol implementation tracing message processing
- Protocol inference
  - **Get the state-machine of the protocol**
  - Two techniques
    - Off-line learning: observes communication and build a state-machine that matches observed message sequences → NP-complete problem
    - On-line learning: the predictor is updated at each step with the given data. Polynomial time

Check out papers of J. Caballero et al. (2007, 2009) and Cho et al. (2010)

Agenda

Universidad Zaragoza

Reverse Engineering of Software

Reverse code engineering

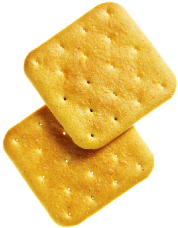
- Also known as *cracking* (also as Software Reverse Engineering)
- Remove code protections (copyrights)
- NOT always bad: bugs detection, potential exploits, . . . in your programs

Universidad Zaragoza

Reverse Engineering of Software

Reverse code engineering

- Also known as *cracking* (also as Software Reverse Engineering)
- Remove code protections (copyrights)
- NOT always bad: bugs detection, potential exploits, . . . in your programs
- Crackers: something else than a (salt) cookies. . .
  - NOT MISTAKE with *CRiminal hACKERS*



Reverse Engineering of Software

- Involves knowledge of assembler
- Involves knowledge of file formats
- Involves knowledge of Operating System
- Involves knowledge of networks

Reverse Engineering of Software

- Involves knowledge of assembler
- Involves knowledge of file formats
- Involves knowledge of Operating System
- Involves knowledge of networks
- Involves knowledge of laws
  - Jail is cold

Methods

- Analyze information exchange (on computer bus and network)
- Disassembler: read `PUSH EAX` instead of `0x50`
- Decompilation: recreate the high-level representation of the bytes

Universidad Zaragoza

Reverse Engineering of Software

Well-known RCE examples

- Samba
  - They reverse-engineer unpublished information about how Windows file sharing worked
- Wine
  - They reverse-engineer unpublished information about Windows API
- OpenOffice
  - They reverse-engineer unpublished information about Microsoft Office file formats
- Mac OS System 4.1
  - Reversed in 1987 by Bell Laboratories to enable this OS to execute on RISC machines of their own

Reverse Engineering of Software

Static Analysis

- Code is not executed (cold analysis)
- Steps:
  1. Analyse the PE header
  2. Read the code (disassembler)
  3. Search for strings
  4. THINK!

Reverse Engineering of Software

Static Analysis

- Code is not executed (cold analysis)
- Steps:
  1. Analyse the PE header
  2. Read the code (disassembler)
  3. Search for strings
  4. THINK!

Can you figure out what is doing? → Not enough in all binaries

Reverse Engineering of Software

Dynamic Analysis

- Code is executed (hot analysis)
- Steps:
  1. Analyse the PE header
  2. Read the code (debugger)
  3. Search for strings
  4. Observe the execution
  5. THINK!

Universidad Zaragoza 1542

Reverse Engineering of Software

Dynamic Analysis

- Code is executed (hot analysis)
- Steps:
  1. Analyse the PE header
  2. Read the code (debugger)
  3. Search for strings
  4. Observe the execution
  5. THINK!

Can you find out *now* what is doing?

Universidad Zaragoza

Reverse Engineering of Software

Understanding Assembler from Code – a MUST!

```c
int main(int argc, char* argv[])
{
  printf("Hello, world! My name is %s and
    I've %d years old.", "Restituto", 23);
  return 0;
}
```

```asm
  .section .rdata,"dr"
LC0:
  .ascii "Restituto\0"
  .align 4
LC1:
  .ascii "Hello, world! My name is %s and
                I've %d years old.\0"
  .text

_main:
LFB6:
  push  ebp
  mov   ebp, esp
  and   esp, -16
  sub   esp, 16
  call  ___main
  mov   DWORD PTR [esp+8], 23
  mov   DWORD PTR [esp+4], OFFSET FLAT:LC0
  mov   DWORD PTR [esp], OFFSET FLAT:LC1
  call  _printf
  mov   eax, 0
  leave
  ret
```

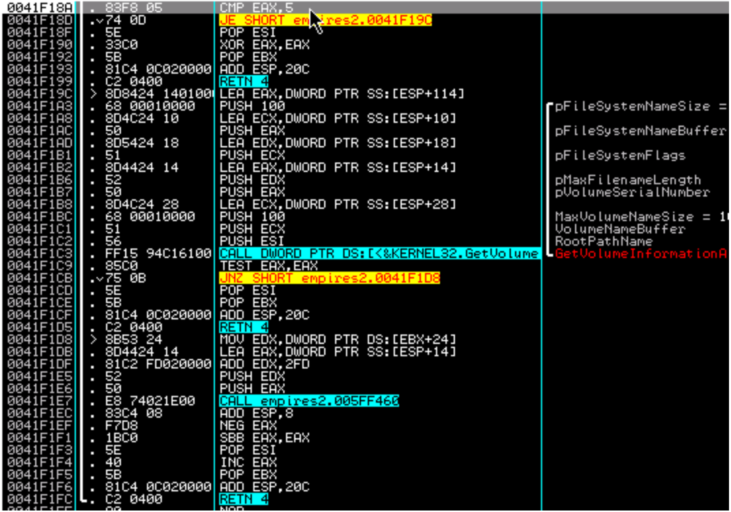
Reverse Engineering of Software

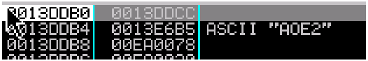
Basic techniques

- CD-Check
  - Check presence of a specific drive (e.g., AoE II, AvP Gold Editon)
- Event-fishing
  - Windows follows event-driven paradigm
  - Detect where a certain message is handled. That is, reveal the function that handles an input
- Serial-fishing
  - Find the correct serial for registering a software
- Keygenning. Two types:
  - Self-keygenning: patch the binary to show the correct key by itself
  - Keygen: replicate the key code generation

Reverse Engineering of Software

Example of CD Check (1)





Reverse Engineering of Software

Example of CD Check (2)

```
* Reference To: KERNEL32.GetDriveTypeA, Ord:0104h
                                  |
:10001898 FF1568110110            Call dword ptr [10011168]
:1000189E 83F805                  cmp eax, 00000005
:100018A1 7569                    jne 1000190C
:100018A3 33ED                    xor ebp, ebp
:100018A5 BF01000000              mov edi, 00000001

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:100018FD(C)
|
:100018AA 8D4C2414                lea ecx, dword ptr [esp+14]
:100018AE 57                      push edi
:100018AF 51                      push ecx
:100018B0 8D542454                lea edx, dword ptr [esp+54]

* Possible StringData Ref from Data Obj ->"%strack%02d.cda"
                                  |
:100018B4 68C4600110              push 100160C4
:100018B9 52                      push edx
:100018BA 895C2458                mov dword ptr [esp+58], ebx
:100018BE E89A0F0000              call 1000285D
:100018C3 8D44245C                lea eax, dword ptr [esp+5C]
```

```
* Possible StringData Ref from Data Obj ->"rb"
                                  |
:100018C7 68C0600110              push 100160C0
:100018CC 50                      push eax
:100018CD E8780F0000              call 1000284A
:100018D2 8BF0                    mov esi, eax
:100018D4 83C418                  add esp, 00000018
:100018D7 3BF3                    cmp esi, ebx
:100018D9 7424                    je 100018FF
:100018DB 56                      push esi
:100018DC 6A0B                    push 0000000B
:100018DE 8D4C2428                lea ecx, dword ptr [esp+28]
:100018E2 6A04                    push 00000004
:100018E4 51                      push ecx
:100018E5 E8180E0000              call 10002702
:100018EA 8B4C2458                mov ecx, dword ptr [esp+58]
:100018EE 56                      push esi
:100018EF 03E9                    add ebp, ecx
:100018F1 E88F0D0000              call 10002685
:100018F6 83C414                  add esp, 00000014
:100018F9 47                      inc edi
:100018FA 83FF10                  cmp edi, 00000010
:100018FD 7EAB                    jle 100018AA

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:100018D9(C)
|
:100018FF 83FF10                  cmp edi, 00000010
:10001902 7508                    jne 1000190C
:10001904 81FDC09A4200            cmp ebp, 00429AC0
:1000190A 741F                    je 1000192B

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|:10001902(C)
|
:1000190C 8A442413                mov al, byte ptr [esp+13]
:10001910 FEC0                    inc al
:10001912 3C7A                    cmp al, 7A
:10001914 88442413                mov byte ptr [esp+13], al
:10001918 0F8E6DFFFFFF            jle 1000188B
:1000191E 5F                      pop edi
:1000191F 5E                      pop esi
:10001920 5D                      pop ebp
:10001921 33C0                    xor eax, eax
:10001923 5B                      pop ebx
:10001924 81C4A0000000            add esp, 000000A0
:1000192A C3                      ret
```

Reverse Engineering of Software

Example of serial-fishing (1)

Serial hardcoded in the code

Serial hardcoded in the code

```
00424FEC PUSH registro.00422798                      UNICODE "\english.dll"
0042505D MOV DWORD PTR SS:[EBP-90],registro.0042     UNICODE "Trial version expired"
00425080 MOV DWORD PTR SS:[EBP-80],registro.0042     UNICODE "This trial version has exp
004250FB MOV DWORD PTR SS:[EBP-90],registro.0042     UNICODE "Este producto ha caducado"
0042511E MOV DWORD PTR SS:[EBP-80],registro.0042     UNICODE "Este producto ha caducado.
00425187 PUSH registro.00422734                      UNICODE "alt"
004251D0 MOV DWORD PTR SS:[EBP-80],registro.0042     UNICODE "ERROR!"
0042542B PUSH registro.00422E48                      UNICODE "utilities 77 backdoor"
00425454 PUSH registro.00422E14                      UNICODE "RC10-FFGH-PPBA-9999"
004255C0 PUSH registro.00422798                      UNICODE "\english.dll"
00425625 MOV DWORD PTR SS:[EBP-80],registro.0042     UNICODE "Registration"
0042563C PUSH registro.00422E78                      UNICODE "There is a problem when tr
004256?1 PUSH registro.00422E6C                      UNICODE "..."
```

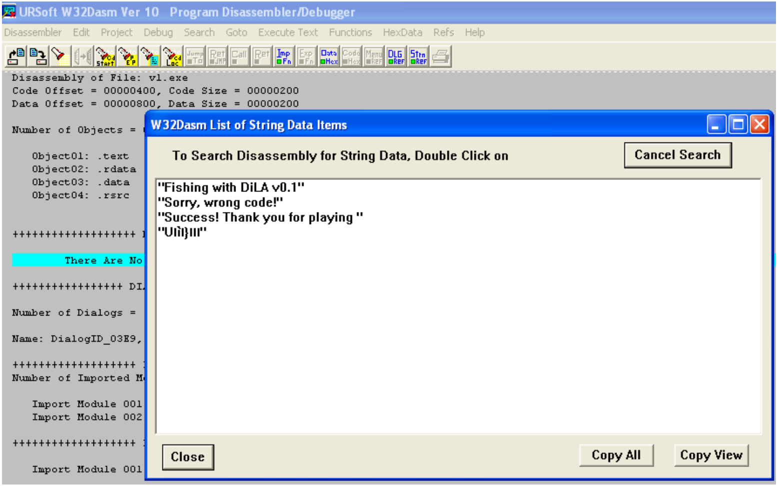
Serial hardcoded in the code

```
00424FEC PUSH registro.00422798                      UNICODE "\english.dll"
0042505D MOV DWORD PTR SS:[EBP-90],registro.0042:    UNICODE "Trial version expired"
00425080 MOV DWORD PTR SS:[EBP-80],registro.0042:    UNICODE "This trial version has exp
004250FB MOV DWORD PTR SS:[EBP-90],registro.0042:    UNICODE "Este producto ha caducado'
0042511E MOV DWORD PTR SS:[EBP-80],registro.0042:    UNICODE "Este producto ha caducado.
00425187 PUSH registro.00422734                      UNICODE "alt"
004251D0 MOV DWORD PTR SS:[EBP-80],registro.0042:    UNICODE "ERROR!"
0042542B PUSH registro.00422E48                      UNICODE "utilities 77 backdoor"
00425454 PUSH registro.00422E14                      UNICODE "RC10-FFGH-PPBA-9999"
004255C0 PUSH registro.00422798                      UNICODE "\english.dll"
00425625 MOV DWORD PTR SS:[EBP-80],registro.0042:    UNICODE "Registration"
0042563C PUSH registro.00422E78                      UNICODE "There is a problem when tr
```

(Un)fortunately, not so common nowadays. . . ⌣̈

Reverse Engineering of Software

Example of serial-fishing (2) – file `fwdv1.zip`



Reverse Engineering of Software

Example of serial-fishing (2) – file `fwdv1.zip`

```
* Reference To: user32.DialogBoxParamA, Ord:008Ah
                                    |
:00401016 E87B000000                Call 00401096
:0040101B 6A00                      push 00000000

* Reference To: kernel32.ExitProcess, Ord:0080h
                                    |
:0040101D E886000000                Call 004010A8
:00401022 55                        push ebp
:00401023 8BEC                      mov ebp, esp
:00401025 817D0C11010000            cmp dword ptr [ebp+0C], 00000111
:0040102C 754D                      jne 0040107B
:0040102E 817D10EB030000            cmp dword ptr [ebp+10], 000003EB
:00401035 7542                      jne 00401079
:00401037 6A00                      push 00000000
:00401039 6A00                      push 00000000
:0040103B 68EA030000                push 000003EA
:00401040 FF7508                    push [ebp+08]

* Reference To: user32.GetDlgItemInt, Ord:00F3h
                                    |
:00401043 E854000000                Call 0040109C
:00401048 3D9A020000                cmp eax, 0000029A
:0040104D 7416                      je 00401065
:0040104F 6A10                      push 00000010

* Possible StringData Ref from Data Obj ->"Fishing with DiLA v0.1"
                                    |
:00401051 6800304000                push 00403000

* Possible StringData Ref from Data Obj ->"Sorry, wrong code!"
                                    |
:00401056 6817304000                push 00403017
:0040105B FF7508                    push [ebp+08]

* Reference To: user32.MessageBoxA, Ord:019Dh
```

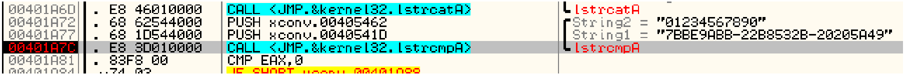
Universidad Zaragoza 1542

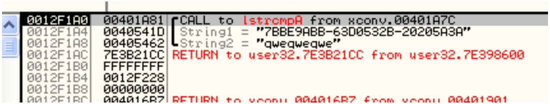
Reverse Engineering of Software

Example of serial-fishing (2) – file `fwdv1.zip`

Reverse Engineering of Software

Example of self-keygenning (1) – file `X-Converter.zip`





- Recall last example: `lstrcmpA`
  - Offset `0040541D`: valid key

Let's make the binary speak by itself. . .

Reverse Engineering of Software

Example of self-keygenning (2) – file `X-Converter.zip`

MessageBox function

Displays a modal dialog box that contains a system icon, a set of buttons, and a brief application-specific message, such as status or error information. The message box returns an integer value that indicates which button the user clicked.

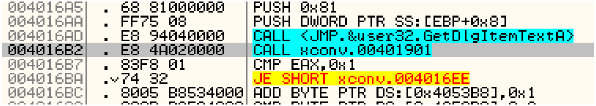
Syntax

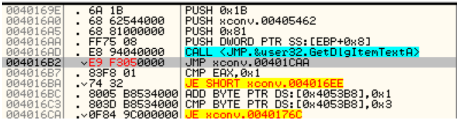
C++

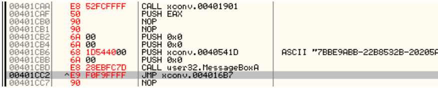
```cpp
int WINAPI MessageBox(
  _In_opt_  HWND hWnd,
  _In_opt_  LPCTSTR lpText,
  _In_opt_  LPCTSTR lpCaption,
  _In_      UINT uType
);
```

Reverse Engineering of Software

Example of self-keygenning (3) – file `X-Converter.zip`







Universidad Zaragoza

Reverse Engineering of Software

Example of self-keygenning (3) – file `X-Converter.zip`

Reverse Engineering of Software

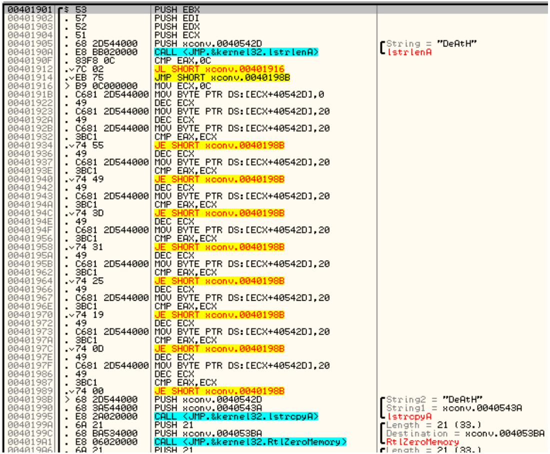
Example of keygenning (1) – file `X-Converter.zip`

```
0040166B  .v0F85 24010000  JNZ xconv.00401795
00401671  . 6A 0E          PUSH 0E                                   [Count = E (14.)
00401673  . 68 2D544000    PUSH xconv.0040542D                       |Buffer = xconv.0040542D
00401678  . 68 80000000    PUSH 80                                   |ControlID = 80 (128.)
0040167D  . FF75 08        PUSH DWORD PTR SS:[EBP+8]                 |hWnd
00401680  . E8 C1040000    CALL <JMP.&user32.GetDlgItemTextA>        [GetDlgItemTextA
00401685  . 83F8 0C        CMP EAX,0C
00401688  .v7F 4B          JG SHORT xconv.004016D5
0040168A  . 83F8 04        CMP EAX,4
0040168D  .v7C 46          JL SHORT xconv.004016D5
0040168F  . 68 2D544000    PUSH xconv.0040542D                       [String2 = "DeAtH"
00401694  . 68 06334000    PUSH xconv.00403306                       |String1 = xconv.00403306
00401699  . E8 26050000    CALL <JMP.&kernel32.lstrcpyA>             [lstrcpyA
0040169E  . 6A 1B          PUSH 1B                                   [Count = 1B (27.)
004016A0  . 68 62544000    PUSH xconv.00405462                       |Buffer = xconv.00405462
004016A5  . 68 81000000    PUSH 81                                   |ControlID = 81 (129.)
004016AA  . FF75 08        PUSH DWORD PTR SS:[EBP+8]                 |hWnd
004016AD  . E8 94040000    CALL <JMP.&user32.GetDlgItemTextA>        [GetDlgItemTextA
004016B2  . E8 4A020000    CALL xconv.00401901
004016B7  . 83F8 01        CMP EAX,1
004016BA  .v74 32          JE SHORT xconv.004016EE
004016BC  . 8005 B8534000  ADD BYTE PTR DS:[4053B8],1
004016C3  . 803D B8534000  CMP BYTE PTR DS:[4053B8],3
004016CA  .v0F84 9C000000  JE xconv.0040176C
004016D0  .vE9 81000000    JMP xconv.00401756
004016D5  > 6A 10          PUSH 10                                   [Style = MB_OK|MB_ICONHAND|MB_APPLMODAL
004016D7  . 68 79304000    PUSH xconv.00403079                       |Title = "Sorry"
004016DC  . 68 53324000    PUSH xconv.00403253                       |Text = "Sorry username must be at least 4 characters♪◘long and not mo
004016E1  . FF75 08        PUSH DWORD PTR SS:[EBP+8]                 |hOwner
004016E4  . E8 69040000    CALL <JMP.&user32.MessageBoxA>            [MessageBoxA
004016E9  .vE9 D7000000    JMP xconv.004017C5
004016EE  > E8 A8030000    CALL xconv.00401A9B
004016F3  . C605 72434000  MOV BYTE PTR DS:[404372],1
004016FA  . 6A 40          PUSH 40                                   [Style = MB_OK|MB_ICONASTERISK|MB_APPLMODAL
004016FC  . 68 33304000    PUSH xconv.00403033                       |Title = "Thank you!"
00401701  . 68 3E304000    PUSH xconv.0040303E                       |Text = "Registration done. Thank you for registering this program!"
00401706  . FF75 08        PUSH DWORD PTR SS:[EBP+8]                 |hOwner
00401709  . E8 44040000    CALL <JMP.&user32.MessageBoxA>            [MessageBoxA
0040170E  . 6A 00          PUSH 0                                    [Result = 0
```

- 4 $\leq$ length name $\leq$ 12
- Length serial: 27 (read)
- Checking procedure: `00401901`
- Name buffer: `buf_42D`

Reverse Engineering of Software

Example of keygenning (2) – file `X-Converter.zip`



```
00401901  $ 53              PUSH EBX
00401902  . 57              PUSH EDI
00401903  . 52              PUSH EDX
00401904  . 51              PUSH ECX
00401905  . 68 2D544000     PUSH xconv.0040542D                        String = "DeAtH"
0040190A  . E8 BB020000     CALL <JMP.&kernel32.lstrlenA>              lstrlenA
0040190F  . 83F8 0C         CMP EAX,0C
00401912  .v7C 02           JL SHORT xconv.00401916
00401914  .vEB 75           JMP SHORT xconv.0040198B
00401916  > B9 0C000000     MOV ECX,0C
0040191B  . C681 2D544000   MOV BYTE PTR DS:[ECX+40542D],0
00401922  . 49              DEC ECX
00401923  . C681 2D544000   MOV BYTE PTR DS:[ECX+40542D],20
0040192A  . 49              DEC ECX
0040192B  . C681 2D544000   MOV BYTE PTR DS:[ECX+40542D],20
00401932  . 3BC1            CMP EAX,ECX
00401934  .v74 55           JE SHORT xconv.0040198B
00401936  . 49              DEC ECX
00401937  . C681 2D544000   MOV BYTE PTR DS:[ECX+40542D],20
0040193E  . 3BC1            CMP EAX,ECX
00401940  .v74 49           JE SHORT xconv.0040198B
00401942  . 49              DEC ECX
00401943  . C681 2D544000   MOV BYTE PTR DS:[ECX+40542D],20
0040194A  . 3BC1            CMP EAX,ECX
0040194C  .v74 3D           JE SHORT xconv.0040198B
0040194E  . 49              DEC ECX
0040194F  . C681 2D544000   MOV BYTE PTR DS:[ECX+40542D],20
00401956  . 3BC1            CMP EAX,ECX
00401958  .v74 31           JE SHORT xconv.0040198B
0040195A  . 49              DEC ECX
0040195B  . C681 2D544000   MOV BYTE PTR DS:[ECX+40542D],20
00401962  . 3BC1            CMP EAX,ECX
00401964  .v74 25           JE SHORT xconv.0040198B
00401966  . 49              DEC ECX
00401967  . C681 2D544000   MOV BYTE PTR DS:[ECX+40542D],20
0040196E  . 3BC1            CMP EAX,ECX
00401970  .v74 19           JE SHORT xconv.0040198B
00401972  . 49              DEC ECX
00401973  . C681 2D544000   MOV BYTE PTR DS:[ECX+40542D],20
0040197A  . 3BC1            CMP EAX,ECX
0040197C  .v74 0D           JE SHORT xconv.0040198B
0040197E  . 49              DEC ECX
0040197F  . C681 2D544000   MOV BYTE PTR DS:[ECX+40542D],20
00401986  . 49              DEC ECX
00401987  . 3BC1            CMP EAX,ECX
00401989  .v74 00           JE SHORT xconv.0040198B
0040198B  > 68 2D544000     PUSH xconv.0040542D                        String2 = "DeAtH"
00401990  . 68 3A544000     PUSH xconv.0040543A                        String1 = xconv.0040543A
00401995  . E8 2A020000     CALL <JMP.&kernel32.lstrcpyA>              lstrcpyA
0040199A  . 6A 21           PUSH 21                                    Length = 21 (33.)
0040199C  . 68 BA534000     PUSH xconv.004053BA                        Destination = xconv.004053BA
004019A1  . E8 06020000     CALL <JMP.&kernel32.RtlZeroMemory>         RtlZeroMemory
004019A6  . 6A 21           PUSH 21                                    Length = 21 (33.)
```

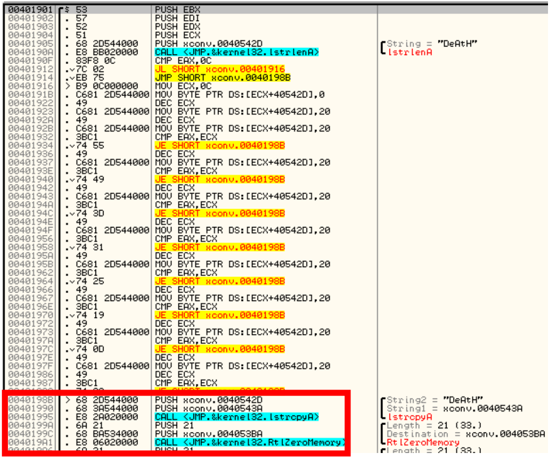
Reverse Engineering of Software

Example of keygenning (3) – file `X-Converter.zip`


```
eax = strlen(buf_42D);
ecx = 12;
buf_42D[ecx] = 00;
for(; eax != ecx; ecx--)
  buf_42D[ecx] = 20;
```

Universidad Zaragoza

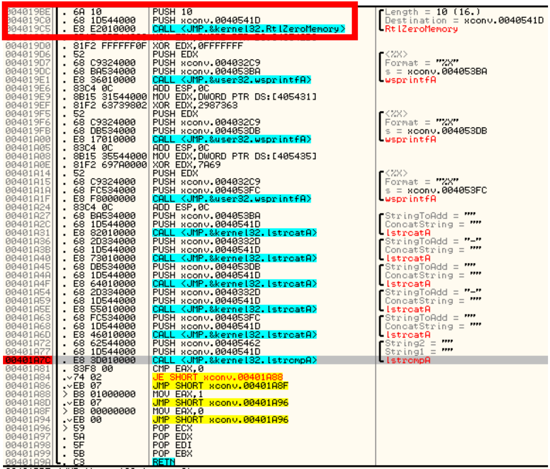
Example of keygenning (4) – file `X-Converter.zip`




```
eax = strlen(buf_42D);
ecx = 12;
buf_42D[ecx] = 00;
for(; eax != ecx; ecx--)
  buf_42D[ecx] = 20;

buf_43A = buf_42D;
buf_3BA = zeros(33);
```

Reverse Engineering of Software

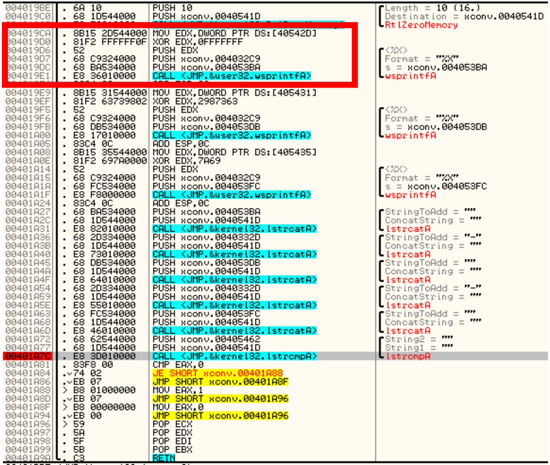
Example of keygenning (5) – file `X-Converter.zip`




```
eax = strlen(buf_42D);
ecx = 12;
buf_42D[ecx] = 00;
for(; eax != ecx; ecx--)
  buf_42D[ecx] = 20;

buf_43A = buf_42D;
buf_3BA = zeros(33);
buf_3DB = zeros(33);
buf_3FC = zeros(33);
buf_41D = zeros(16);
```

Reverse Engineering of Software

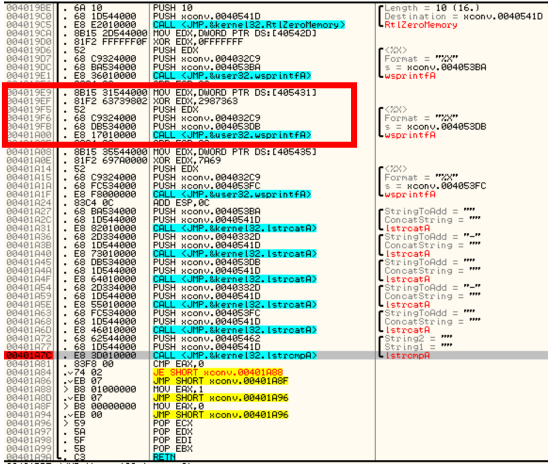
Example of keygenning (6) – file `X-Converter.zip`




```
eax = strlen(buf_42D);
ecx = 12;
buf_42D[ecx] = 00;
for(; eax != ecx; ecx--)
  buf_42D[ecx] = 20;

buf_43A = buf_42D;
buf_3BA = zeros(33);
buf_3DB = zeros(33);
buf_3FC = zeros(33);
buf_41D = zeros(16);

edx = buf_42D; // 4 bytes
edx ^= 0xFFFFFFF;
buf_3BA = <edx in string>
// Recall endianness!!
```

Reverse Engineering of Software

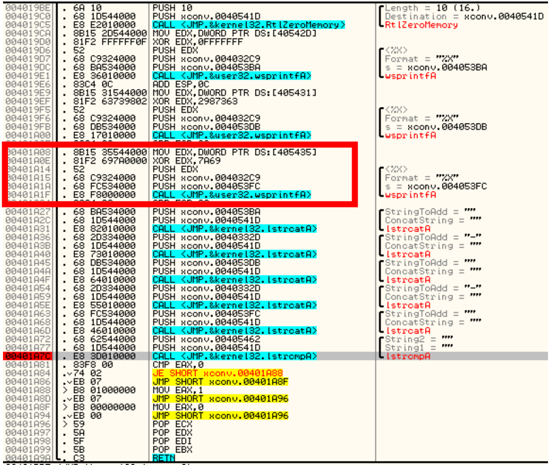
Example of keygenning (7) – file `X-Converter.zip`




```
eax = strlen(buf_42D);
ecx = 12;
buf_42D[ecx] = 00;
for(; eax != ecx; ecx--)
  buf_42D[ecx] = 20;

buf_43A = buf_42D;
buf_3BA = zeros(33);
buf_3DB = zeros(33);
buf_3FC = zeros(33);
buf_41D = zeros(16);

edx = buf_42D; // 4 bytes
edx ^= 0xFFFFFFF;
buf_3BA = <edx in string>
// Recall endianness!!

edx = (buf_42D + 4); // 4 bytes
edx ^= 0x2987363;
buf_3DB = <edx in string>
// Recall endianness!!
```

Reverse Engineering of Software

Example of keygenning (8) – file `X-Converter.zip`

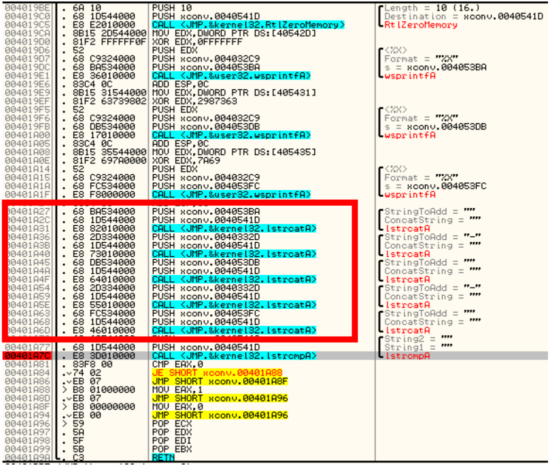



```
eax = strlen(buf_42D);
ecx = 12;
buf_42D[ecx] = 00;
for(; eax != ecx; ecx--)
  buf_42D[ecx] = 20;

buf_43A = buf_42D;
buf_3BA = zeros(33);
buf_3DB = zeros(33);
buf_3FC = zeros(33);
buf_41D = zeros(16);

edx = buf_42D; // 4 bytes
edx ^= 0xFFFFFFF;
buf_3BA = <edx in string>
// Recall endianness!!

edx = (buf_42D + 4); // 4 bytes
edx ^= 0x2987363;
buf_3DB = <edx in string>
// Recall endianness!!

edx = (buf_42D + 8); // 4 bytes
edx ^= 0x7A69;
buf_3FC = <edx in string>
// Recall endianness!!
```

Universidad Zaragoza

Reverse Engineering of Software

Example of keygenning (9) – file `X-Converter.zip`




```
eax = strlen(buf_42D);
ecx = 12;
buf_42D[ecx] = 00;
for(; eax != ecx; ecx--)
  buf_42D[ecx] = 20;

buf_43A = buf_42D;
buf_3BA = zeros(33);
buf_3DB = zeros(33);
buf_3FC = zeros(33);
buf_41D = zeros(16);

edx = buf_42D; // 4 bytes
edx ^= 0xFFFFFFF;
buf_3BA = <edx in string>
// Recall endianness!!

edx = (buf_42D + 4); // 4 bytes
edx ^= 0x2987363;
buf_3DB = <edx in string>
// Recall endianness!!

edx = (buf_42D + 8); // 4 bytes
edx ^= 0x7A69;
buf_3FC = <edx in string>
// Recall endianness!!

buf_41D = buf_3BA & '-'
   & buf_3DB & '-' & buf_3FC;
```

Reverse Engineering of Software

Example of keygenning (10) – file `X-Converter.zip`

```
void generate_key(char *name, int len)
{
        int bufLen = 32;
        char    buf_3BA[bufLen], buf_3DB[bufLen],
                buf_3FC[bufLen], buf_41D[15],
                nameFilled[12];
        // Init
        memset(nameFilled, ' ', 12);
        for(int i = 0; i < len; i++)
                nameFilled[i] = name[i];

        memset(buf_3BA, 0, bufLen);
        memset(buf_3DB, 0, bufLen);
        memset(buf_3FC, 0, bufLen);
        memset(buf_41D, 0, 15);

        // Compute 1st num
        sprintf(buf_3BA, "%X",
                compute_number(nameFilled, 0, 0xFFFFFFF));
        // Compute 2nd num
        sprintf(buf_3DB, "%X",
                compute_number(nameFilled, 4, 0x2987363));
        // Compute 3rd num
        sprintf(buf_3FC, "%X",
                compute_number(nameFilled, 8, 0x7A69));

        // Build key
        sprintf(buf_41D, "%s-%s-%s",
                buf_3BA, buf_3DB, buf_3FC);
        printf("Dear %s, your key is %s.\n", name, buf_41D);
}
```

```
int compute_number(char *name,
                   int init, int xor_val)
{
        int edx = name[init];

        for(int i = 1; i < 4; i++)
          edx ^= name[init + i] << 8*i;

        return edx^xor_val;
}

/* Example of first num computation
edx  = nameFilled[0];
edx ^= nameFilled[1] << 8;
edx ^= nameFilled[2] << 16;
edx ^= nameFilled[3] << 24;
// XOR it
edx ^= 0xFFFFFFF; */
```

Universidad Zaragoza

Agenda

Universidad Zaragoza 1542

Reverse Engineering of Integrated Circuits/Smart Cards

Near Field Communication ¿?

Near Field Communication: What is it? (I)

- Bidirectional short-range contactless communication technology
  - Up to 10 cm
- Based on RFID standards, works in the 13.56 MHz spectrum
- Data transfer rates vary: 106, 216, and 424 kbps

Near Field Communication: What is it? (I)

- Bidirectional short-range contactless communication technology
  - Up to 10 cm
- Based on RFID standards, works in the 13.56 MHz spectrum
- Data transfer rates vary: 106, 216, and 424 kbps

Security based on proximity concern: physical constraints

Near Field Communication: What is it? (I)

- Bidirectional short-range contactless communication technology
  - Up to 10 cm
- Based on RFID standards, works in the 13.56 MHz spectrum
- Data transfer rates vary: 106, 216, and 424 kbps

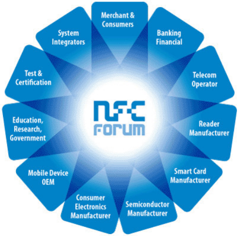
Security based on proximity concern: physical constraints

Main elements & operation modes

- Two main elements:
  - Proximity Coupling Device (PCD, also NFC-capable device)
  - Proximity Integrated Circuit Cards (PICC, also NFC tags)
- Three operation modes:
  - Peer to peer: direct communication between parties
  - Read/write: communication with a NFC tag
  - Card-emulation: an NFC device behaves as a tag

Near Field Communication: What is it? (II)

"Big" actors

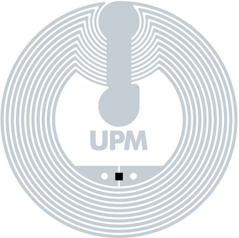


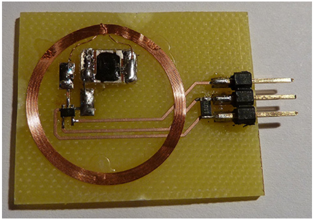
NFC Forum

- Non-profit industry association
- Formed on March 18, 2004
- Founders: NXP Semiconductors (formerly Philips Semiconductors), Sony and Nokia
- Promotes implementation and standardisation of NFC
- 190 member companies (June 2013). Some located at Spain:
  - Applus
  - AT4 Wireless

Near Field Communication: What is it? (III)

Real actors (1)



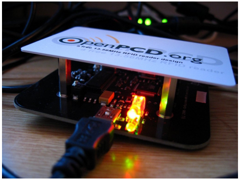


PICC

- Proximity Integrated Circuit Card
- Commonly named as *tag*
- Passive or active (depends on power supply)
  - Widely used (cheaper): passive ones
- It contains:
  - Internal capacitor
    - Stores the energy coming from the reader
  - Resistor

Near Field Communication: What is it? (III)

Real actors (2)



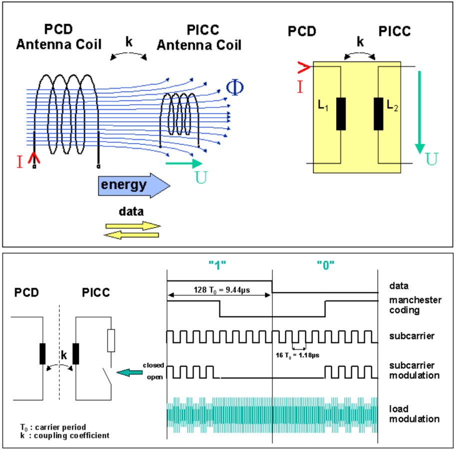
PCD

- Proximity Coupling Device
- Commonly named as *reader/writer*
- Active (forced)
- Contains the antenna
  - Communication at the 13.56MHz (±7kHz) frequency
  - Electronic field

Universidad Zaragoza

Near Field Communication: What is it? (IV)

An interesting reading on this topic. . .

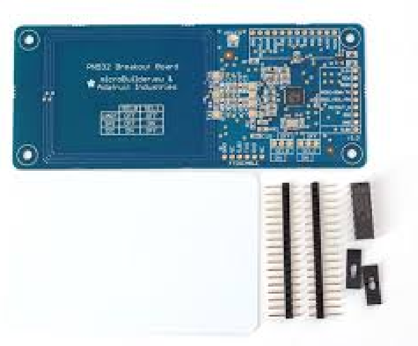


[Taken from 13.56 MHz RFID Proximity Antennas (`http://www.nxp.com/documents/application_note/AN78010.pdf`)]

Near Field Communication: Where is it used? (V)

Near Field Communication

Lab Environment



Hardware

- AdaFruit PN532
- A computer

Software

- C compiler
- NFC Library (`libnfc`)
- NFC tools (`nfc-tools`)

MIFARE classic ¿?

MIFARE Classic (I): What is it?

MIFARE product family

- Introduced in 1995 by NXP
- “Advanced technology for RFID identification”
- Based on ISO/IEC 14443 Type A/B 13.56 MHz standard
- Several products:
  - Ultralight
  - Classic
  - DESFire
  - SmartMX

MIFARE Classic (I): What is it?

MIFARE product family

- Introduced in 1995 by NXP
- "Advanced technology for RFID identification"
- Based on ISO/IEC 14443 Type A/B 13.56 MHz standard
- Several products:
  - Ultralight
  - Classic
  - DESFire
  - SmartMX
- 50M reader and 5B card components sold
- $\sim$ 80% contactless ticketing credentials (according to ABI Research)

MIFARE Classic (II): Some of its common uses

Some systems using MIFARE Classic

- Access Controls
  - University of Zaragoza
  - Personal entrance Schiphol Airport (AMS)
  - Dutch military bases
  - Hotel room keys
  - Many office and official buildings
- Ticketing events
- Public transport systems
  - OV-Chipkaart (NL)
  - Oyster card (London, UK)
  - Smartrider (AU)
  - EMT (Málaga, Spain)
  - Wikipedia: `http://en.wikipedia.org/wiki/MIFARE`

MIFARE Classic (III): Internal Structure (1)

Logical Structure

- EEPROM memory
- Basic unit: 16B block
- A sector is a set of blocks
- Two size variants:
    - 1KB (16 sectors, 4 blocks each)
    - 4KB (40 sectors, first 32 sectors are 4-block, the rest 16-block)

Universidad Zaragoza 1542

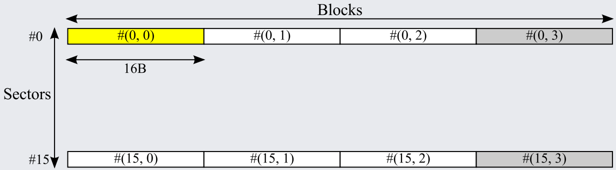
MIFARE Classic (III): Internal Structure (1)

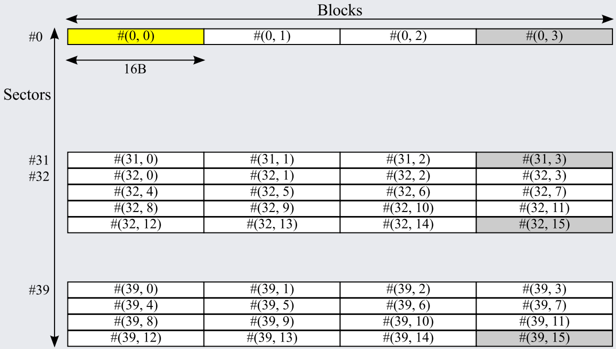
Logical Structure

- EEPROM memory
- Basic unit: 16B block
- A sector is a set of blocks
- Two size variants:
  - 1KB (16 sectors, 4 blocks each)
  - 4KB (40 sectors, first 32 sectors are 4-block, the rest 16-block)

Let me show you this graphically...

MIFARE Classic (III): Internal Structure(2)





MIFARE Classic (III): Internal Structure (3)

Manufacturer block

- Sector 0, block 0 (yellow one in previous slide)
- Contains:
  - UID (4B)
  - BCC (bit count check, 1B): XOR-ing of UID bytes
  - Manufacturer data (11B)
- Set and locked by manufacturer → read only!

MIFARE Classic (III): Internal Structure (3)

Manufacturer block

- Sector 0, block 0 (yellow one in previous slide)
- Contains:
  - UID (4B)
  - BCC (bit count check, 1B): XOR-ing of UID bytes
  - Manufacturer data (11B)
- Set and locked by manufacturer → read only!
  - Not the case for some Chinese cards ☺

Universidad Zaragoza

MIFARE Classic (III): Internal Structure (4)

Storing data. . .

Storing data into blocks

- Read/write block
  - You can store data as you want, no matter how
- Data block
  - Predefined format (look below!)
  - Don't worry: APIs will help you!
    - Only need a *value*, it puts all the values properly on its own. . .)
  - Contains:
    - Value (twice)
    - Value negated (once)
    - 1-byte address (twice)
    - 1-byte address negated (twice)

Value	$\overline{\text{Value}}$	Value	@	$\overline{@}$	@	$\overline{@}$
0	4	8	12	13	14	15

MIFARE Classic (III): Internal Structure (5)

Key A	Access bits	Key B
0	6	10 — 15

Sector trailer

- Last one in each sector (grey ones in previous slide)
- Contains:
  - Key A
  - Access Bits
  - Key B
- Authentication per sector before any operation is allowed
- Access bits define how is the auth. required and what operations are allowed
- **Having fun with access bits may provoke a useless tag!**
- Keys are set to `FFFFFFFFFFFFh` at delivery

MIFARE Classic (III): Internal Structure (6)

Operations

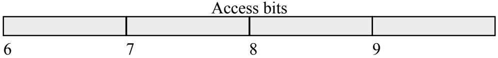
Operation	Description	Valid for...		
		R/W block	Value block	Sector trailer
Read	Reads a memory block	√	√	√
Write	Writes a memory block	√	√	√
Increment	Reads the value, increments it and stores		√	
Decrement	Reads the value, decrements it and stores		√	
Transfer	Transfers contents of internal register to a block		√	
Restore	Loads contents of a block to internal register		√	

MIFARE Classic (III): Internal Structure (7)

Access Conditions

Access Bits	Valid Commands	Block
$C1_0 C2_0 C3_0$	(all operations)	0
$C1_1 C2_1 C3_1$	(all operations)	1
$C1_2 C2_2 C3_2$	(all operations)	2
$C1_3 C2_3 C3_3$	Read, Write	3

- 3 bits define access conditions for every data block and sector trailer
- Stored non-negated and negated
- Commands are executed only after a successful authentication



	Bit 7							Bit 0
Byte 6	$\overline{C2_3}$	$\overline{C2_2}$	$\overline{C2_1}$	$\overline{C2_0}$	$\overline{C1_3}$	$\overline{C1_2}$	$\overline{C1_1}$	$\overline{C1_0}$
Byte 7	$C1_3$	$C1_2$	$C1_1$	$C1_0$	$\overline{C3_3}$	$\overline{C3_2}$	$\overline{C3_1}$	$\overline{C3_0}$
Byte 8	$C3_3$	$C3_2$	$C3_1$	$C3_0$	$C2_3$	$C2_2$	$C2_1$	$C2_0$
Byte 9	user data (free)							

Universidad Zaragoza

MIFARE Classic (III): Internal Structure (8)

Access Conditions for sector trailer

Access Bits			Access condition for...					
			Key A		Access bits		Key B	
*C*1	*C*2	*C*3	read	write	read	write	read	write
0	0	0	-	key A	key A	-	key A	key A
0	0	1	-	key A	key A	key A	key A	key A
0	1	0	-	-	key A	-	key A	-
0	1	1	-	key B	key A (or B)	key B	-	key B
1	0	0	-	key B	key A (or B)	-	-	key B
1	0	1	-	-	key A (or B)	key B	-	-
1	1	0	-	-	key A (or B)	-	-	-
1	1	1	-	-	key A (or B)	-	-	-

(- means never)

Recall: show `mfcab` tool (`https://bitbucket.org/rjrodriguez/mfcab`)

MIFARE Classic (III): Internal Structure (9)

Access Conditions for data blocks

Access Bits			Access condition for...				Application
*C*1	*C*2	*C*3	Read	Write	Increment	Decrement, Transfer, Restore	
0	0	0	key A (or B)†	key A (or B)	key A (or B)	key A (or B)	Transport configuration
0	0	1	key A (or B)†	-	-	key A (or B)	Value block
0	1	0	key A (or B)†	-	-	-	R/W block
0	1	1	key B	key B	-	-	R/W block
1	0	0	key A (or B)	Key B	-	-	R/W block
1	0	1	key B	-	-	-	R/W block
1	1	0	key A (or B)	key B	key B	key A (or B)	Value block
1	1	1	-	-	-	-	R/W block

(- means never)

† if key B can be read in the sector trailer, then it cannot be used for authentication

MIFARE Classic: Communication Protocol (I)

Protocol steps

1. Get the tags in the reader's range
2. Select only one tag (anticollision loop)
3. Access a block, with key A or key B (starts authentication step)

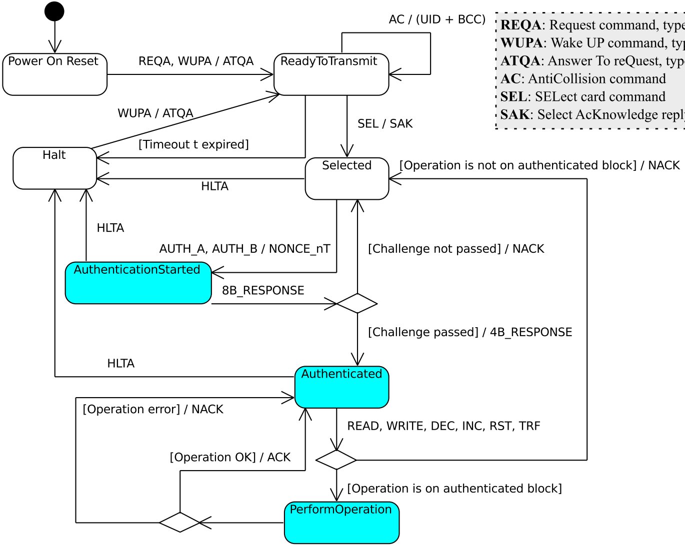
Authentication step

- Challenge-response mutual authentication using nonces
  - Nonce: randomly generated information
  - Nonces generated from a LFSR (next slides)

Universidad Zaragoza

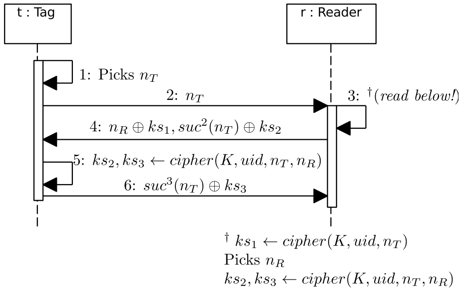
MIFARE Classic: Communication Protocol (II)

UML-SM of a NFC tag



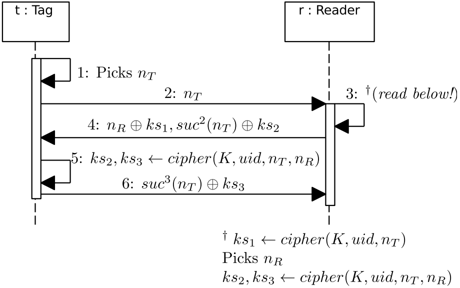
REQA: Request command, type A
WUPA: Wake UP command, type A
ATQA: Answer To reQuest, type A
AC: AntiCollision command
SEL: SELect card command
SAK: Select AcKnowledge reply

MIFARE Classic: Communication Protocol (III)



- Three-pass authentication
  1. Send nonce (n_T) as challenge
     - Generated by a 16-bit LFSR ($g(x) = x^{16} + x^{14} + x^{13} + x^{11} + 1$)
  2. Send response and other nonce n_R as challenge
  3. Send response
- Note: from n_T, communication is ciphered

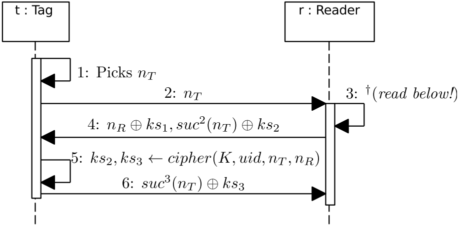
MIFARE Classic: Communication Protocol (IV)



Known plaintext [GKMRVSJ-ESORICS-08]

- Recall: n_T is in plaintext

MIFARE Classic: Communication Protocol (IV)

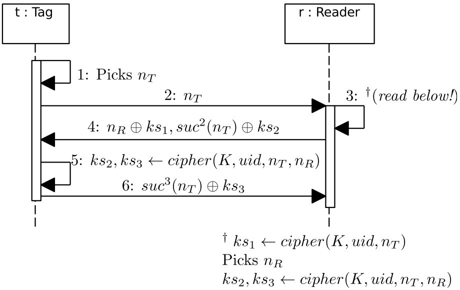


† $ks_1 \leftarrow cipher(K, uid, n_T)$
Picks n_R
$ks_2, ks_3 \leftarrow cipher(K, uid, n_T, n_R)$

Known plaintext [GKMRVSJ-ESORICS-08]

- Recall: n_T is in plaintext
- Given n_T, compute $suc^2(n_T) \rightarrow ks_2 = n_T \oplus suc^2(n_T)$

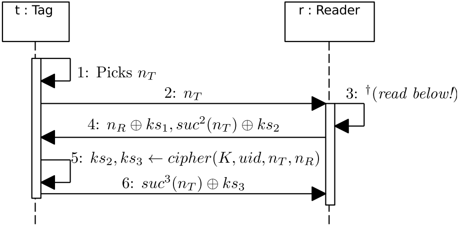
MIFARE Classic: Communication Protocol (IV)



Known plaintext [GKMRVSJ-ESORICS-08]

- Recall: n_T is in plaintext
- Given n_T, compute $suc^2(n_T) \rightarrow ks_2 = n_T \oplus suc^2(n_T)$
- When tag does not send last response, some readers time out and send `HLT` command XORed ks_3
  - `HLT` command is known, then we recover ks_3

MIFARE Classic: Communication Protocol (IV)



† $ks_1 \leftarrow cipher(K, uid, n_T)$
Picks n_R
$ks_2, ks_3 \leftarrow cipher(K, uid, n_T, n_R)$

Known plaintext [GKMRVSJ-ESORICS-08]

- Recall: n_T is in plaintext
- Given n_T, compute $suc^2(n_T) \rightarrow ks_2 = n_T \oplus suc^2(n_T)$
- When tag does not send last response, some readers time out and send `HLT` command XORed ks_3
  - `HLT` command is known, then we recover ks_3
- Eavesdropping a successful authentication session
  - ks_2, ks_3 recovered from $suc^2(n_T) \oplus n_T, suc^3(n_T) \oplus n_T$

MIFARE Classic: CRYPTO1 (I)

- Proprietary stream cipher. Key length of 48 bits
- “Security by obscurity” principle
- Hardware on-chip: faster cryptographic operations!
- Reversed some years ago. . . :
  - K. Nohl and H. Plötz: “Mifare: Little Security, Despite Obscurity”, in *Chaos Communication Congress*, 2007. Reverse engineering on silicon implementation
  - García et al.: “Dismantling MIFARE Classic”, in *ESORICS* 2008. Fully disclosed the entire encryption algorithm

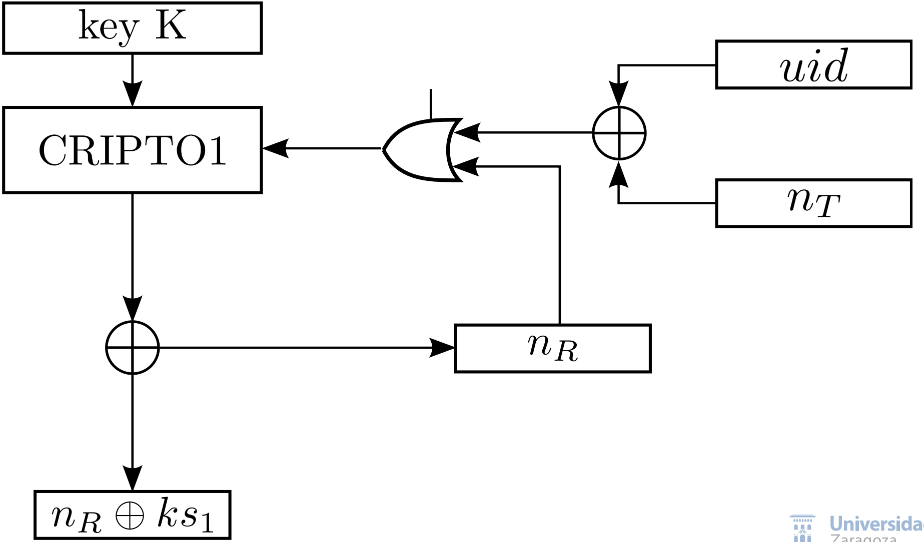
MIFARE Classic: CRYPTO1 (I)

- Proprietary stream cipher. Key length of 48 bits
- "Security by obscurity" principle
- Hardware on-chip: faster cryptographic operations!
- Reversed some years ago. . . :
  - K. Nohl and H. Plötz: "Mifare: Little Security, Despite Obscurity", in *Chaos Communication Congress*, 2007. Reverse engineering on silicon implementation
  - García et al.: "Dismantling MIFARE Classic", in *ESORICS* 2008. Fully disclosed the entire encryption algorithm
- Linear Feedback Shift Register (LFSR) + two-layer non-linear filter generator
  - At every clock tick, register is shifted one bit to the left
  - Leftmost bit: discarded
  - Feedback bit: computed with $g(x)$

$$g(x) = x^{48} + x^{43} + x^{39} + x^{38} + x^{36} + x^{34} + x^{33} + x^{31} + x^{29} + x^{24} + x^{23} + x^{21} + x^{19} + x^{13} + x^{9} + x^{7} + x^{6} + x^{5} + 1$$

MIFARE Classic: CRYPTO1 (II)

Initialisation diagram



MIFARE Classic: Known Weaknesses (I)

On Pseudo-Random Number Generator

MOST CRITICAL weakness

Low entropy

- LFSR generating nonces: 16-bit length
- 0.6 seconds to generate ALL possible nonces [NESP-USENIX-08]
- Generator resets to a known state every time the tag starts operating
  - Just a wait a fixed number of clock cycles. . .
  - Experimentally possible to get the same nonce every 30ms using Proxmark 3 reader

Universidad Zaragoza

MIFARE Classic: Known Weaknesses (II)

On Cryptographic Cipher

$$x_9, x_{11}, x_{13}, \ldots, x_{47}$$

Keystream generation

- Odd bits as inputs to the filter functions
- Divide-and-Conquer technique
  - Split even, odd bits in groups
  - Firstly focus on odd group:
    - After 2 shifts, new input is $x_{11}, x_{13}, \ldots, x_{47}$ and x_{49}
    - Used for generating two keystreams
    - Explore what bits generate the right keystreams
- Attack: Recover all sector keys without the needed of a genuine reader

Universidad Zaragoza

MIFARE Classic: Known Weaknesses (III)

On Cryptographic Cipher

$$x_9, x_{11}, x_{13}, \ldots, x_{47}$$

Leftmost bit not used in filter generator

- First 9 bits unused
- Attack: Rollback LFSR state bit a bit
  - Recover the initial state of LFSR

Statistical Bias [C-SECRYPT-09]

- With a $\pi = 0.75$, ks_1 is independent of the last three bits of n_R
- Attack: card-only attack
  - Recover one key, then apply nested authentication attack [GKMRVSJ-ESORICS-08]
  - Does not require any pre-computation
  - Extremely fast, and requires a few hundred queries
  - Further information: `http://eprint.iacr.org/2009/137.pdf`

MIFARE Classic: Known Weaknesses (IV)

On Communication Protocol

One-Time Padding (OTP)

- ISO-14443-A: every byte sent is followed by a parity bit
- MIFARE Classic computes parity bit over plaintext instead of ciphertext
- LFSR is not shifted after parity bit encryption

MIFARE Classic: Known Weaknesses (IV)

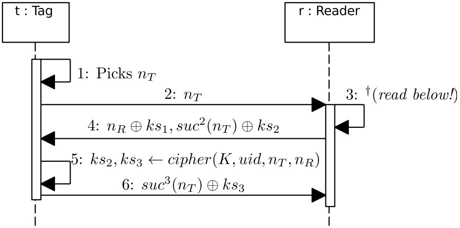
On Communication Protocol

One-Time Padding (OTP)

- ISO-14443-A: every byte sent is followed by a parity bit
- MIFARE Classic computes parity bit over plaintext instead of ciphertext
- LFSR is not shifted after parity bit encryption
- Next plaintext and parity bit use the same keystream → OTP seems not to be OTP. . .
- More examples of violating OTP property:
  - Venona Project (U.S. counter-intelligence program during Cold War)
  - Point-to-Point Tunneling Protocol (PPTP)
  - IEEE 802.11 WEP

MIFARE Classic: Known Weaknesses (V)

On Communication Protocol



† $ks_1 \leftarrow cipher(K, uid, n_T)$
Picks n_R
$ks_2, ks_3 \leftarrow cipher(K, uid, n_T, n_R)$

Information Leak from Parity

- Second step in authentication, reader sends $n_R, suc^2(n_T)$
- PICC checks parity bits in n_R before checking $suc^2(n_T)$
  - When parity is incorrect, PICC does not answer
  - When $suc^2(n_T)$ is incorrect, it answers `NACK` (transmission error)
- `NACK` sent encrypted → ks_3 can be recovered

Universidad Zaragoza 1542

MIFARE Classic: Known Weaknesses (VI)

On Deployment

Default Keys

- Some chip manufacturers leave default keys on chips
- This is obvious, as companies must make the effort to do system integration for clients. . . (sic!)
- RTFM: Chip manufacturer warns about CHANGING default keys
- Default keys are well-known and documented

FFFFFFFFFFFFh	000000000000h	1A982C7E459Ah
A0A1A2A3A4A5h	B0B1B2B3B4B5h	AABBCCDDEEFFh
D3F7D3F7D3F7h	4D3A99C351DDh	

Related Work (I)

On MIFARE Classic weaknesses analysis (1)

NP-CCC-07 K. Nohl and H. Plötz, "Mifare: Little Security, Despite Obscurity", in *Chaos Communication Congress*, 2007.

GKMRVSJ-ESORICS-08 García et al., "Dismantling MIFARE Classic", in *Procs. of the European Symposium on Research in Computer Security* (ESORICS), 2008.

KHG-CARDIS-08 G.d Koning Gans et al., "A Practical Attack on the MIFARE Classic", in *Procs. of the Smart Card Research and Advanced Applications Conference* (CARDIS), 2008.

NESP-USENIX-08 K. Nohl et al., "Reverse-Engineering a Cryptographic RFID Tag". In *USENIX Security Symposium*, 2008.

GRBS-SP-09 F.D. García et al., "Wirelessly Pickpocketing a Mifare Classic Card", in *Procs. of the 30th IEEE Symposium on Security and Privacy* (S&P), 2009.

Related Work (II)

On MIFARE Classic weaknesses analysis (2)

C-SECRYPT-09 N.T. Courtois, "The Dark Side of Security by Obscurity and Cloning MiFare Classic Rail and Building Passes Anywhere, Anytime". In *Procs. of the Int. Conf. on Security and Cryptography* (SECRYPT), 2009

GRBS-SP-09 F.D. García et al., "Wirelessly Pickpocketing a Mifare Classic Card", in *Procs. of the 30th IEEE Symposium on Security and Privacy* (S&P), 2009

Tan-MScThesis-09 W.H. Tan, "Practical Attacks on the MIFARE Classic", Imperial College London, 2009

On NFC Attacks

VK-NFC-11 R. Verdult and F. Kooman, "Practical Attacks on NFC Enabled Cell Phones". In *Procs. of the 3rd Int. Workshop on Near Field Communication*, 2011

Related Work (III)

On MIFARE Attacks

- Sogeti ESEC Pentest: "Playing with NFC for fun and coffee"
- BackTrack Linux: "RFID Cooking with Mifare Classic" (2012)
- C. Miller, "Exploring the NFC Attack Surface", in *BlackHat US*, 2012.
- ComputerWorld article: "Android NFC hack enables travelers to ride subways for free, researchers say" (2012)
- HackPlayers: "Cómo colarse en el metro de forma elegante" (2012)
- Security ArtWork: "Hacking RFID, rompiendo (...) Mifare" (2010)

On NFC-related issues

- R. Lifchitz, Hacking the NFC credit cards for fun and debit (Hackito Ergo Sum 2012)
- J.M. Esparza, Give me your credit card, the NFC way (NcN'12)
- J. Vila, R.J. Rodríguez, Practical Experiences on NFC Relay Attacks with Android: Virtual Pickpocketing Revisited. In *RFIDSec 2015*

Reverse Engineering of Integrated Circuits/Smart Cards

Extending the lab environment

- NFC brings “cards” to mobile devices
- Payment sector is quite interested in this new way for making payments
  - 500M NFC payment users expected by 2019
- Almost 300 smart phones available at the moment with NFC capabilities
  - Check `http://www.nfcworld.com/nfc-phones-list/`
  - Most of them runs Android OS

Time to buy a NFC-capable device!

Android and NFC: A Tale of L♥ve (I)

Recap on evolution of Android NFC support

NFC operation modes supported

Software
- Reader/Writer
- Peer-to-peer

Hardware
- Card-emulation

Software
- Reader/Writer
- Peer-to-peer
- Card-emulation

Hardware
- Card-emulation

Android 2.3.3 Gingerbread (API level 10)

- **NfcA** (ISO/IEC 14443-3A)
- **NfcB** (ISO/IEC 14443-3B)
- **Ndef**
- **IsoDep** (ISO/IEC 14443-4)
- **NfcV** (ISO/IEC 15693)
- **NfcF** (JIS 6319-4)
- **NdefFormatable**
- **MifareClassic**
- **MifareUltralight**

Android 4.2 Jelly Bean (API level 17)

- **NfcBarcode**

Android CyanogenMod OS 9.1

- **IsoPcdA** (ISO/IEC 14443-4A)
- **IsoPcdB** (ISO/IEC 14443-4B)

thanks to Doug Year

Android 4.4 KitKat (API level 19)

NfcAdapter.ReaderCallback added

Universidad Zaragoza

Android and NFC: A Tale of L♥ve (II)

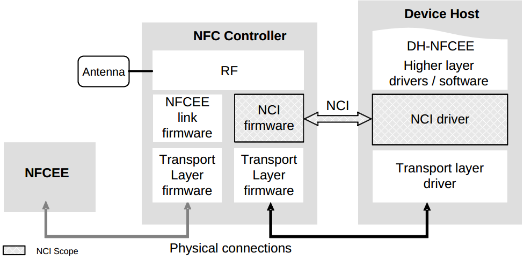
Digging into Android NFC stack

- Event-driven framework, nice API support
- Two native implementations (depending on built-in NFC chip)
  - `libnfc-nxp`
  - `libnfc-nci`

Android and NFC: A Tale of L♥ve (II)

Digging into Android NFC stack

- Event-driven framework, nice API support
- Two native implementations (depending on built-in NFC chip)
  - `libnfc-nxp`
  - `libnfc-nci`
- NXP dropped in favour of NCI:
  - Open architecture, not focused on a single family chip
  - Open interface between the NFC Controller and the DH
  - Standard proposed by NFC Forum



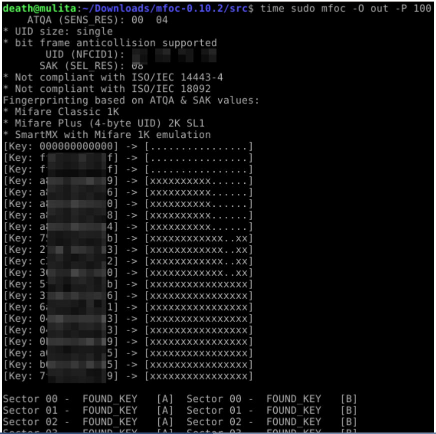
Specific goals

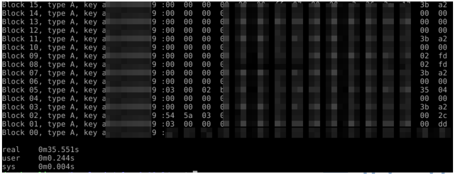
- Figure out the pair of keys (A, B)
- Make a dump of a real card
- Study the card content
- Check any integrity about unauthorised content alteration
- Make a clone card
- Do a mobile app for card-hacking

Reverse Engineering of Integrated Circuits/Smart Cards

Using mfoc

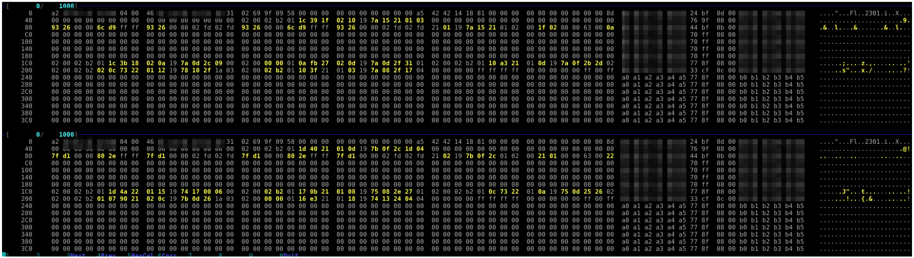
- Two different Classic version
  - MIFARE Classic 1K (T1)
  - MIFARE Classic 4K (T2)





Reverse Engineering of Integrated Circuits/Smart Cards

Understanding the card content. . .

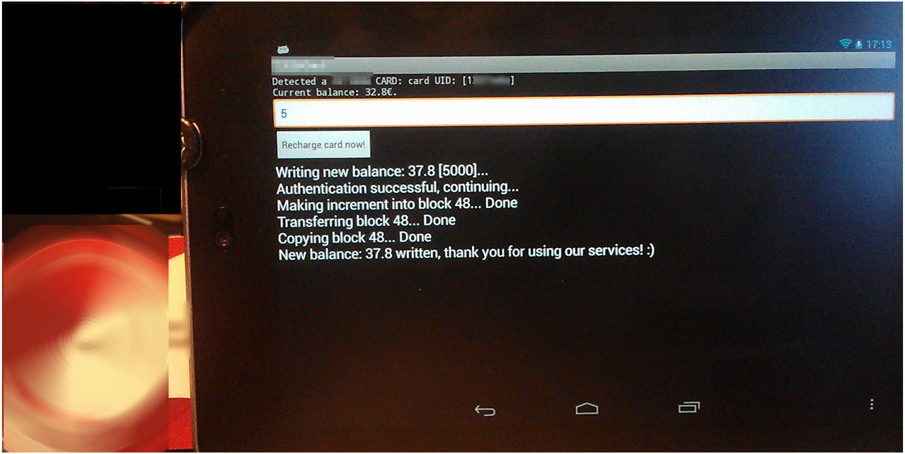


Summary of data

	T1	T2
Card ID	(0, 3)	(10, 3)
Last bus used	(1, 2)	(1, 2)
Current balance	(2, [1, 2])	(12, [1, 2])
Historic	(7, [1, 2, 3]), (8, [1, 2])	(7, [1, 2, 3]), (8, [1, 2])

Reverse Engineering of Integrated Circuits/Smart Cards

Building a PoC in Android O.S. (1)



Universidad Zaragoza

Reverse Engineering of Integrated Circuits/Smart Cards

Building a PoC in Android O.S. (2) – DEMO

Reverse Engineering of Integrated Circuits/Smart Cards

Recalling the initial goals

Goal	Achieved?	Some remarks
Figure out the pair of keys (A, B)	√	Some keys are the default ones
Make a dump of a real card	√	Fast, and simple
Study the card content	√	Not a single bit encrypted
Check any integrity about unauthorized content alteration	√	no integrity
Make a cloned card	√*	A perfect clone (Chine cards rulez!)
Do a mobile app for card-hacking	√	Android fu$#ing rocks!

Universidad Zaragoza 1542

Reverse Engineering of Integrated Circuits/Smart Cards

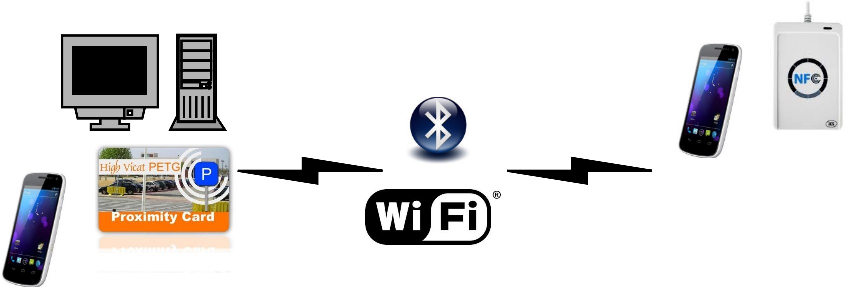
Thinking (and acting?) badly. . . (1)

What else could be done. . .

- Identity spoofing
  - Possible penalties for spoofed people
  - Consume the real balance of someone else
- Use of all public services for free
- Black market?
  - Fake recharge point
  - Whether I sold a card illegitimately charged. . .
- Just put the app in Google Play, and have fun ⌣̈

Reverse Engineering of Integrated Circuits/Smart Cards

Thinking (and acting?) badly. . . (2): Relay attacks



Reverse Engineering of Integrated Circuits/Smart Cards

Event timeline

Nov 2012 Nice chat with J.M. Esparza ☺

Universidad Zaragoza

Reverse Engineering of Integrated Circuits/Smart Cards

Event timeline

Nov 2012 Nice chat with J.M. Esparza ⌣̈

Nov 2012 (ending) Lab environment set and tested (it works!)

Universidad Zaragoza 1542

Reverse Engineering of Integrated Circuits/Smart Cards

Event timeline

- **Nov 2012** Nice chat with J.M. Esparza ⌣̈
- **Nov 2012** (ending) Lab environment set and tested (it works!)
- **Dec 2012** Nice chat with C. Lorenzana ⌣̈ (at STIC CCN-CERT conference)

Universidad Zaragoza 1542

Reverse Engineering of Integrated Circuits/Smart Cards

Event timeline

Nov 2012 Nice chat with J.M. Esparza ⌣̈

Nov 2012 (ending) Lab environment set and tested (it works!)

Dec 2012 Nice chat with C. Lorenzana ⌣̈ (at STIC CCN-CERT conference)

Mar 2013 Confidential report is sent to GDT

Reverse Engineering of Integrated Circuits/Smart Cards

Event timeline

- **Nov 2012** Nice chat with J.M. Esparza :)
- **Nov 2012** (ending) Lab environment set and tested (it works!)
- **Dec 2012** Nice chat with C. Lorenzana :) (at STIC CCN-CERT conference)
- **Mar 2013** Confidential report is sent to GDT
- **Apr 2013** Report is being handled by CNPIC

Reverse Engineering of Integrated Circuits/Smart Cards

Event timeline

Nov 2012 Nice chat with J.M. Esparza ⌣̈

Nov 2012 (ending) Lab environment set and tested (it works!)

Dec 2012 Nice chat with C. Lorenzana ⌣̈ (at STIC CCN-CERT conference)

Mar 2013 Confidential report is sent to GDT

Apr 2013 Report is being handled by CNPIC

May 2013 Company says the problem is known, but does not really care about it...

Universidad Zaragoza 1542

Reverse Engineering of Integrated Circuits/Smart Cards

Event timeline

- **Nov 2012** Nice chat with J.M. Esparza :)
- **Nov 2012 (ending)** Lab environment set and tested (it works!)
- **Dec 2012** Nice chat with C. Lorenzana :) (at STIC CCN-CERT conference)
- **Mar 2013** Confidential report is sent to GDT
- **Apr 2013** Report is being handled by CNPIC
- **May 2013** Company says the problem is known, but does not really care about it...
- **(today)** As they don't care, me neither

Reverse Engineering of Integrated Circuits/Smart Cards

Lessons Learned

- It's good to collaborate with police... but you need to be patient
  - You'll have a good sleep at night and not in jail...

Reverse Engineering of Integrated Circuits/Smart Cards

Lessons Learned

- It's good to collaborate with police. . . but you need to be patient
  - You'll have a good sleep at night and not in jail. . .
  - You also get some free beer from Guardia Civil ⌣̈

Reverse Engineering of Integrated Circuits/Smart Cards

Lessons Learned

- It's good to collaborate with police... but you need to be patient
  - You'll have a good sleep at night and not in jail...
  - You also get some free beer from Guardia Civil ⌣̈
- Security is not considered (as normally) in a Spanish company
  - Not at the beginning of a product design
  - Not even when someone spots out the problem
  - They quantify the risk of people exploiting the problem...
- This is not U.S., unfortunately (in this case)

Universidad Zaragoza 1542

Reverse Engineering of Integrated Circuits/Smart Cards

Lessons Learned

- It's good to collaborate with police... but you need to be patient
  - You'll have a good sleep at night and not in jail...
  - You also get some free beer from Guardia Civil ⌣̈
- Security is not considered (as normally) in a Spanish company
  - Not at the beginning of a product design
  - Not even when someone spots out the problem
  - They quantify the risk of people exploiting the problem...
- This is not U.S., unfortunately (in this case)

Remember, not economic gain but free beer instead!

Some conclusions. . .

- MIFARE Classic is like a memory card
- Vulnerable from 2009
- Weaknesses and attacks very well-known and widely documented

Some conclusions...

- MIFARE Classic is like a memory card
- Vulnerable from 2009
- Weaknesses and attacks very well-known and widely documented
- Need to defend against
  - Unauthorized content alteration
  - Relay attacks
  - Clone attacks

Universidad Zaragoza 1542

Some conclusions…

- MIFARE Classic is like a memory card
- Vulnerable from 2009
- Weaknesses and attacks very well-known and widely documented
- Need to defend against
  - Unauthorized content alteration
  - Relay attacks
  - Clone attacks

Thinking to deploy MIFARE Classic as an access control system?

Some conclusions. . .

- MIFARE Classic is like a memory card
- Vulnerable from 2009
- Weaknesses and attacks very well-known and widely documented
- Need to defend against
  - Unauthorized content alteration
  - Relay attacks
  - Clone attacks

Thinking to deploy MIFARE Classic as an access control system?

Don't.

Agenda

Conclusions

Take-home messages

- **Reverse engineering is an ART that involves a lot of domains**
  - Network protocols
  - Software
  - Integrated circuits/smart cards
  - File formats (forensics)
- Black-box analysis: once we found something, keep digging!

Conclusions

Take-home messages

- **Reverse engineering is an ART that involves a lot of domains**
  - Network protocols
  - Software
  - Integrated circuits/smart cards
  - File formats (forensics)
- Black-box analysis: once we found something, keep digging!

Never ending learning game!

A Tour through the Realms of Reverse Engineering

Ricardo J. Rodríguez

@RicardoJRdez ※ rjrodriguez@unizar.es ※ www.ricardojrodriguez.es

Universidad Zaragoza
1542

November 23, 2016

Legacy Systems
Room A.12