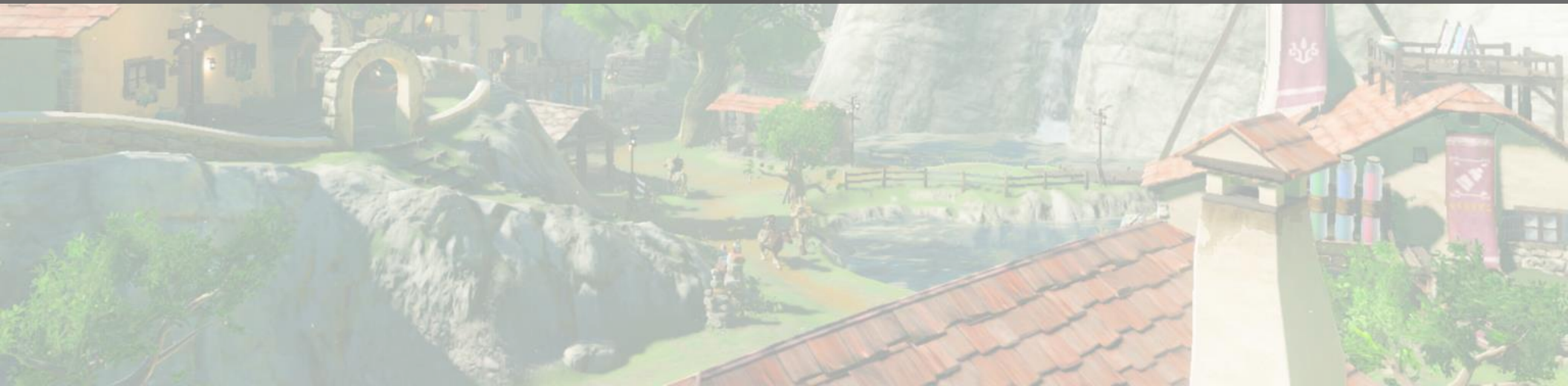
Breath of the RF Field

Hacking Amiibo with Software-Defined Radio

James Chambers

@jamchamb_




Introduction

Amiibo

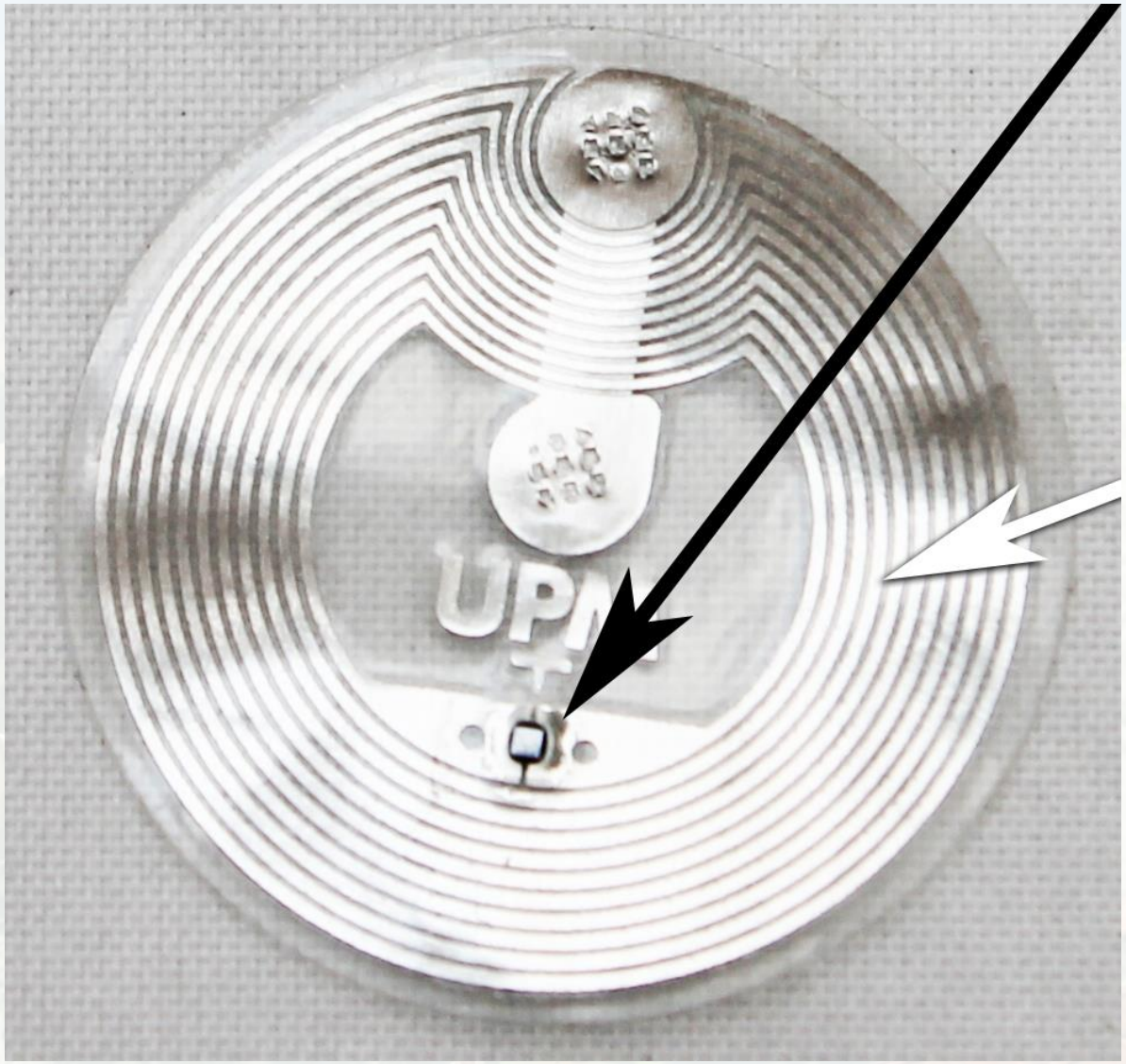
Amiibo


amiibo
Lv. 28
Mario
Feed Equipment
Special
Jack
OK!
Smash
Atk/Def
Attack +45
Defense +45
Speed +29
Bonus Effects
Harder edge grabs
No perfect shield
Heavy jumps
Special
3 Fire Orb
2 Shocking Cape
2 Super Jump
3 High-Pressure F.L.U.D.D.
Select OK to send your FP back to the amiibo and save your changes.


1:49.13
FP
FP
15%
42%

Amiibo

- Contain an NFC “tag”
- Near Field Communication (NFC)
  - Set of protocols for contactless power and data transfer between devices within a few centimeters of each other
- Amiibo use NTAG215
  - Part of the NTAG21x specification, a specific kind of NFC tag made by NXP Semiconductors



https://gototags.com/blog/whats-the-difference-nfc-tags-v-nfc-chips/

Twilight Hack

- Buffer overflow triggered with horse name, which is set by user and kept in save file
  - Save file edited using crypto keys obtained from hardware exploit
  - Edit save to smash the stack
- Used in software-based attack chain for installing homebrew software manager

EponAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

See “Console Hacking 2008: Wii Fail” by Team Twiizers (fail0verflow)

Twilight Hack

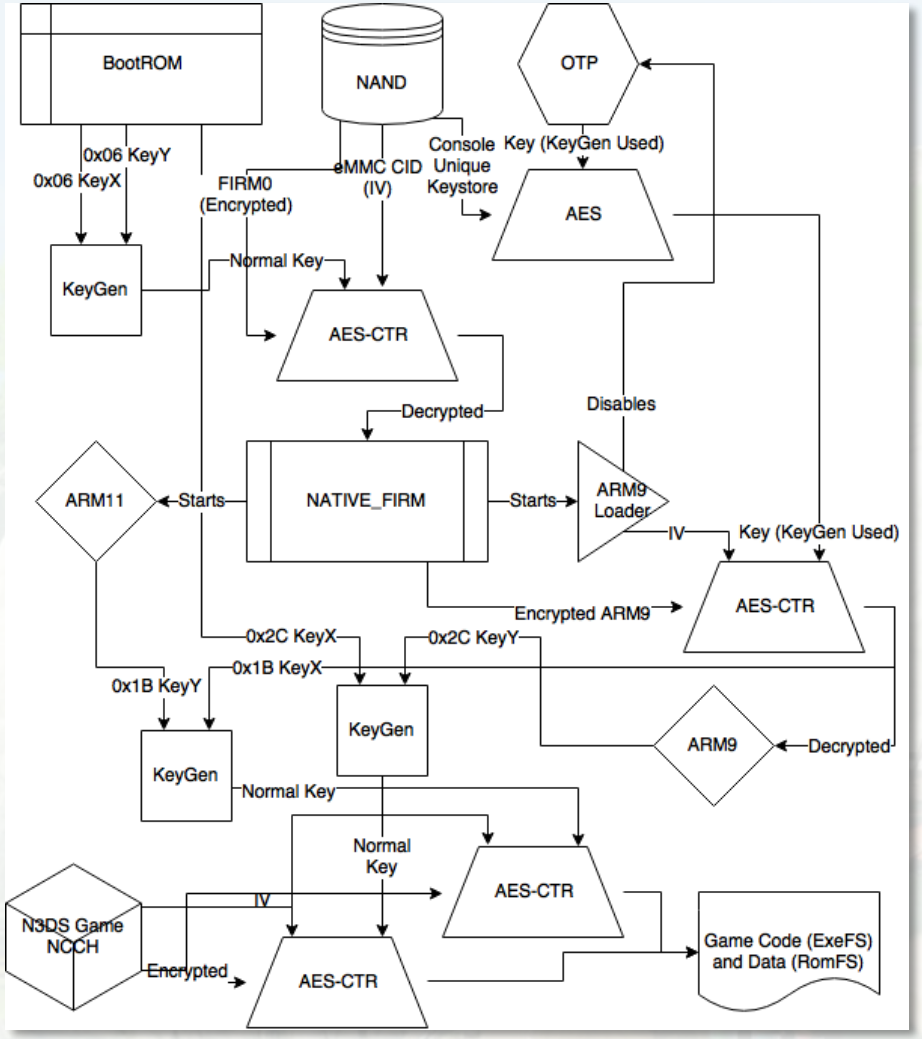
Twilight Hack

Twilight Hack

Twilight Hack

Twilight Hack

- Like smartphones, modern game consoles put users in “jail”
- Meant to prevent piracy and cheating
  - Consoles often sold at a loss, in hopes of profiting from games and services
- No custom software allowed



Yifan Lu – “The 3DS Cryptosystem”
https://yifan.lu/2016/04/06/the-3ds-cryptosystem/

amiibo

Splatoon 2 Series amiibo

Use the Splatoon 2 amiibo to unlock the outfits below! The Splatoon 1 variants of these amiibo unlock the outfits from Splatoon 1!

amiibo can also be used for storing loadouts and settings to use it on any Switch console when you use the amiibo! The original amiibo for Splatoon 1 can be used for this function as well!

Amiibo

- How easy would it to be to clone or spoof them?
- Can the data they store be used for “save game” style exploits?
- Could they be used to trigger an exploit on the Nintendo Switch?

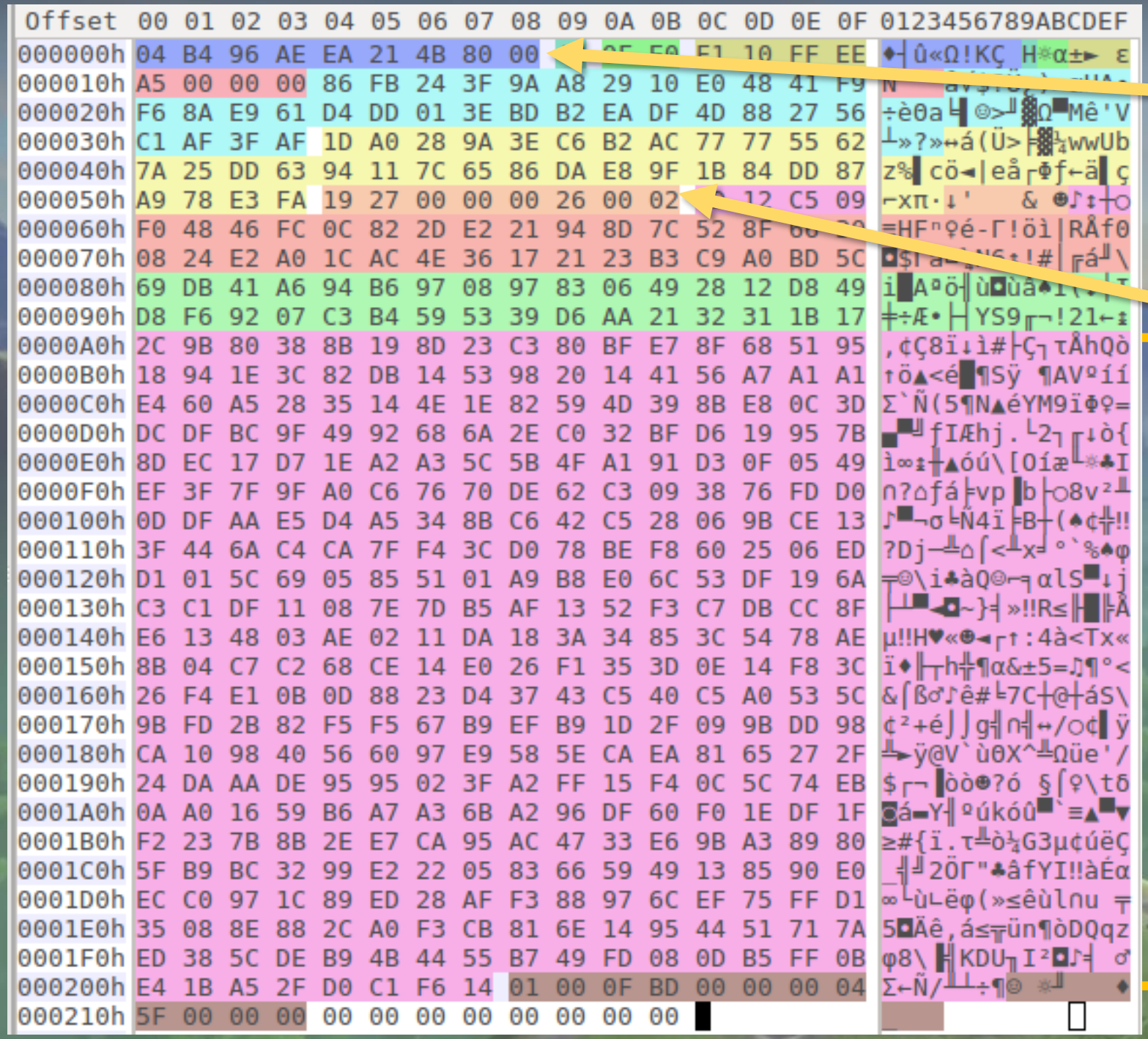

Getting Started

Getting Started

- What's the data like?

jiggly_read1.tag

```
00000000 04 B4 96 AE EA 21 4B 80 00 48 0F E0 F1 10 FF EE A5 00 .....!K..H........
00000012 00 00 86 FB 24 3F 9A A8 29 10 E0 48 41 F9 F6 8A E9 61 ....$?..)..HA....a
00000024 D4 DD 01 3E BD B2 EA DF 4D 88 27 56 C1 AF 3F AF 1D A0 ...>....M.'V..?...
00000036 28 9A 3E C6 B2 AC 77 77 55 62 7A 25 DD 63 94 11 7C 65 (.>...wwUbz%.c..|e
00000048 86 DA E8 9F 1B 84 DD 87 A9 78 E3 FA 19 27 00 00 00 26 .........x...'...&
0000005a 00 02 0D 12 C5 09 F0 48 46 FC 0C 82 2D E2 21 94 8D 7C .......HF...-.!..|
0000006c 52 8F 66 30 08 24 E2 A0 1C AC 4E 36 17 21 23 B3 C9 A0 R.f0.$....N6.!#....
0000007e BD 5C 69 DB 41 A6 94 B6 97 08 97 83 06 49 28 12 D8 49 .\i.A........I(..I
00000090 D8 F6 92 07 C3 B4 59 53 39 D6 AA 21 32 31 1B 17 2C 9B ......YS9..!21..,.
000000a2 80 38 8B 19 8D 23 C3 80 BF E7 8F 68 51 95 18 94 1E 3C .8...#.....hQ....<
000000b4 82 DB 14 53 98 20 14 41 56 A7 A1 A1 E4 60 A5 28 35 14 ...S. .AV....`.(5.
000000c6 4E 1E 82 59 4D 39 8B E8 0C 3D DC DF BC 9F 49 92 68 6A N..YM9...=....I.hj
000000d8 2E C0 32 BF D6 19 95 7B 8D EC 17 D7 1E A2 A3 5C 5B 4F ..2....{.......\[O
000000ea A1 91 D3 0F 05 49 EF 3F 7F 9F A0 C6 76 70 DE 62 C3 09 .....I.?....vp.b..
000000fc 38 76 FD D0 0D DF AA E5 D4 A5 34 8B C6 42 C5 28 06 9B 8v........4..B.(..
0000010e CE 13 3F 44 6A C4 CA 7F F4 3C D0 78 BE F8 60 25 06 ED ..?Dj....<.x..`%..
00000120 D1 01 5C 69 05 85 51 01 A9 B8 E0 6C 53 DF 19 6A C3 C1 ..\i..Q....lS..j..
00000132 DF 11 08 7E 7D B5 AF 13 52 F3 C7 DB CC 8F E6 13 48 03 ...~}...R.......H.
00000144 AE 02 11 DA 18 3A 34 85 3C 54 78 AE 8B 04 C7 C2 68 CE .....:4.<Tx.....h.
00000156 14 E0 26 F1 35 3D 0E 14 F8 3C 26 F4 E1 0B 0D 88 23 D4 ..&.5=...<&.....#.
00000168 37 43 C5 40 C5 A0 53 5C 9B FD 2B 82 F5 F5 67 B9 EF B9 7C.@..S\..+...g...
0000017a 1D 2F 09 9B DD 98 CA 10 98 40 56 60 97 E9 58 5E CA EA ./.......@V`..X^..
0000018c 81 65 27 2F 24 DA AA DE 95 95 02 3F A2 FF 15 F4 0C 5C .e'/$......?.....\
0000019e 74 EB 0A A0 16 59 B6 A7 A3 6B A2 96 DF 60 F0 1E DF 1F t....Y...k...`....
000001b0 F2 23 7B 8B 2E E7 CA 95 AC 47 33 E6 9B A3 89 80 5F B9 .#{......G3....._.
000001c2 BC 32 99 E2 22 05 83 66 59 49 13 85 90 E0 EC C0 97 1C .2.."..fYI........
000001d4 89 ED 28 AF F3 88 97 6C EF 75 FF D1 35 08 8E 88 2C A0 ..(....l.u..5...,.
000001e6 F3 CB 81 6E 14 95 44 51 71 7A ED 38 5C DE B9 4B 44 55 ...n..DQqz.8\..KDU
000001f8 B7 49 FD 08 0D B5 FF 0B E4 1B A5 2F D0 C1 F6 14 01 00 .I........./......
0000020a 0F BD 00 00 00 04 5F 00 00 00 00 00 00 00 00 00 00 00 ......_...........
0000021c
```



Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF
000000h	04	B4	96	AE	EA	21	4B	80	00	[illegible]							♦┤û«Ω!KÇ H☼α±► ε
000010h	A5	00	00	00	86	FB	24	3F	9A	A8	29	10	E0	48	41	F9	N [illegible]
000020h	F6	8A	E9	61	D4	DD	01	3E	BD	B2	EA	DF	4D	88	27	56	÷è0a╘▌ ☺>╝▓Ω▀Mê'V
000030h	C1	AF	3F	AF	1D	A0	28	9A	3E	C6	B2	AC	77	77	55	62	┴»?»↔á(Ü>╞▓¼wwUb
000040h	7A	25	DD	63	94	11	7C	65	86	DA	E8	9F	1B	84	DD	87	z%▌ cö◄|eå┌Φƒ←ä▌ ç
000050h	A9	78	E3	FA	19	27	00	00	00	26	00	02	[illegible]	12	C5	09	⌐xπ·↓' & ☻♪↕┼○
000060h	F0	48	46	FC	0C	82	2D	E2	21	94	8D	7C	52	8F	[illegible]		≡HFⁿ♀é-Γ!öì|RÅf0
000070h	08	24	E2	A0	1C	AC	4E	36	17	21	23	B3	C9	A0	BD	5C	◘$Γá[illegible]N6↕!#|╔á╝\
000080h	69	DB	41	A6	94	B6	97	08	97	83	06	49	28	12	D8	49	i▌Aª¶╣ù◘ùâ♠I(↕|I
000090h	D8	F6	92	07	C3	B4	59	53	39	D6	AA	21	32	31	1B	17	╪÷Æ•├┤YS9╓¬!21←↨
0000A0h	2C	9B	80	38	8B	19	8D	23	C3	80	BF	E7	8F	68	51	95	,¢Ç8ï↓ì#├Ç┐τÅhQò
0000B0h	18	94	1E	3C	82	DB	14	53	98	20	14	41	56	A7	A1	A1	↑ö▲<é▌¶Sÿ ¶AVº¡í
0000C0h	E4	60	A5	28	35	14	4E	1E	82	59	4D	39	8B	E8	0C	3D	Σ`Ñ(5¶N▲éYM9ïΦ♀=
0000D0h	DC	DF	BC	9F	49	92	68	6A	2E	C0	32	BF	D6	19	95	7B	▄▀╝ƒIÆhj.└2┐╓↓ò{
0000E0h	8D	EC	17	D7	1E	A2	A3	5C	5B	4F	A1	91	D3	0F	05	49	ì∞↨╫▲óú\[Oíæ└☼♣I
0000F0h	EF	3F	7F	9F	A0	C6	76	70	DE	62	C3	09	38	76	FD	D0	∩?⌂ƒá╞vp▐b├○8v²╨
000100h	0D	DF	AA	E5	D4	A5	34	8B	C6	42	C5	28	06	9B	CE	13	♪▀¬σ╘Ñ4ï╞B┼(♠¢╬!!
000110h	3F	44	6A	C4	CA	7F	F4	3C	D0	78	BE	F8	60	25	06	ED	?Dj─╨⌂⌠<└x╛°`%♠φ
000120h	D1	01	5C	69	05	85	51	01	A9	B8	E0	6C	53	DF	19	6A	╤☺\i♣àQ☺⌐┐αlS▀↓j
000130h	C3	C1	DF	11	08	7E	7D	B5	AF	13	52	F3	C7	DB	CC	8F	├┴▀◄◘~}┤»!!R≤╠▌╠À
000140h	E6	13	48	03	AE	02	11	DA	18	3A	34	85	3C	54	78	AE	µ!!H♥«☻◄┌↑:4à<Tx«
000150h	8B	04	C7	C2	68	CE	14	E0	26	F1	35	3D	0E	14	F8	3C	ï♦╠┬h╬¶α&±5=♫¶°<
000160h	26	F4	E1	0B	0D	88	23	D4	37	43	C5	40	C5	A0	53	5C	&⌠ßσ♪ê#╘7C┼@┼áS\
000170h	9B	FD	2B	82	F5	F5	67	B9	EF	B9	1D	2F	09	9B	DD	98	¢²+éJ⌋g╣∩╣↔/○¢▌ÿ
000180h	CA	10	98	40	56	60	97	E9	58	5E	CA	EA	81	65	27	2F	╩►ÿ@V`ù0X^╩Ωüe'/
000190h	24	DA	AA	DE	95	95	02	3F	A2	FF	15	F4	0C	5C	74	EB	$┌¬▐òò☻?ó §⌠♀\tδ
0001A0h	0A	A0	16	59	B6	A7	A3	6B	A2	96	DF	60	F0	1E	DF	1F	◙á▬Y╢ºúkóû▀`≡▲▀▼
0001B0h	F2	23	7B	8B	2E	E7	CA	95	AC	47	33	E6	9B	A3	89	80	≥#{ï.τ╩ò¼G3µ¢úëÇ
0001C0h	5F	B9	BC	32	99	E2	22	05	83	66	59	49	13	85	90	E0	_╣╝2ÖΓ"♣âfYI!!àÉα
0001D0h	EC	C0	97	1C	89	ED	28	AF	F3	88	97	6C	EF	75	FF	D1	∞└ù└ëφ(»≤êùlnu ╤
0001E0h	35	08	8E	88	2C	A0	F3	CB	81	6E	14	95	44	51	71	7A	5◘ÄêÄ,á≤╤ün¶òDQqz
0001F0h	ED	38	5C	DE	B9	4B	44	55	B7	49	FD	08	0D	B5	FF	0B	φ8\ ▐╣KDU╖I²◘♪┤ ♂
000200h	E4	1B	A5	2F	D0	C1	F6	14	01	00	0F	BD	00	00	00	04	Σ←Ñ/╨┴÷¶☺ ☼╝ ♦
000210h	5F	00	00	00	00	00	00	00	00	00	00	00	[illegible]				

NFC tag serial number

Settings, signatures, hashes

Character ID

Settings, signatures, hashes

Encrypted data

Getting Started

- How does the crypto system work?

Getting Started

Crypto

Details we do know:

- Amiibo data is encrypted
- Unique keys for each Amiibo
- Can’t copy data from one Amiibo to another
- AES and HMAC are involved
  - Encrypted data is also signed

Getting Started

Crypto

- amiitool by socram8888 can perform encryption and decryption, but the crypto keys are kept private:
  - “Please note that **keys used for signing and encryption are copyright of Nintendo and therefore they can't be shared, and I won't share post them here, send them using a private message, or anywhere else**.”
- Often, the homebrew/game hacking community does not make crypto keys public to avoid enabling piracy.

Getting Started

Crypto

- Private online APIs for decrypting/encrypting Amiibo
  - Crypto keys stored on server, inaccessible to user
- Need to get an API key from the dev
- Activity on API is monitored
- Operations would be throttled by response time, possible rate limiting, and server downtime or decommission
  - Every payload attempt would require waiting for an API request

Getting Started

Crypto

- Amiibo cheat devices use a similar model
  - Online API performs all operations that require crypto
  - PowerSaves device has API authentication tied to hardware
    - “Reversing Powersaves for Amiibo” http://blog.ghettoha.xxx/reversing-powersaves-for-amiibo/
- Cheat device manufacturers don’t want to reveal the crypto secrets, they want you to buy their device

Getting Started

- How can we read, copy, and tamper with the data?
- How can we simulate an Amiibo?

Getting Started

Proxmark 3

- SDR device for RFID and NFC
- Primarily used to research proximity ID cards
- Open source, reprogrammable microcontroller and FPGA
- Can be used to simulate a reader/writer or tag

Getting Started

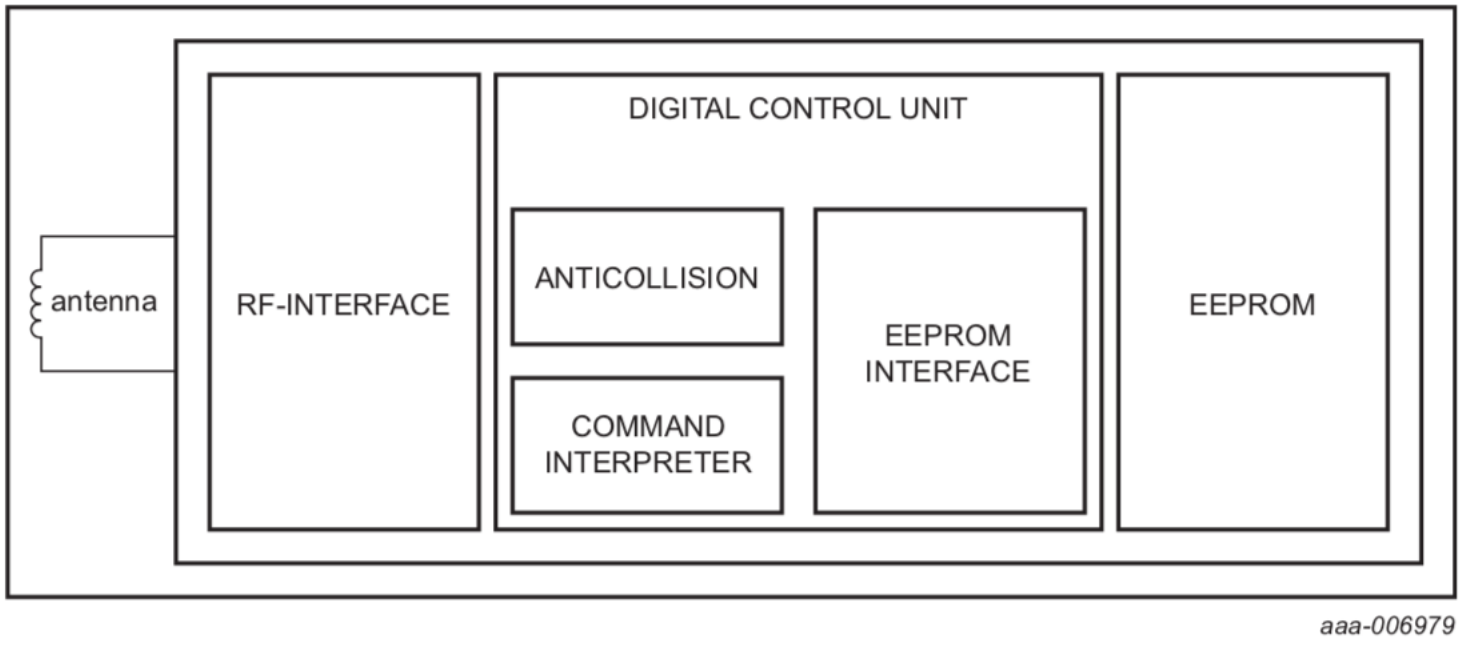


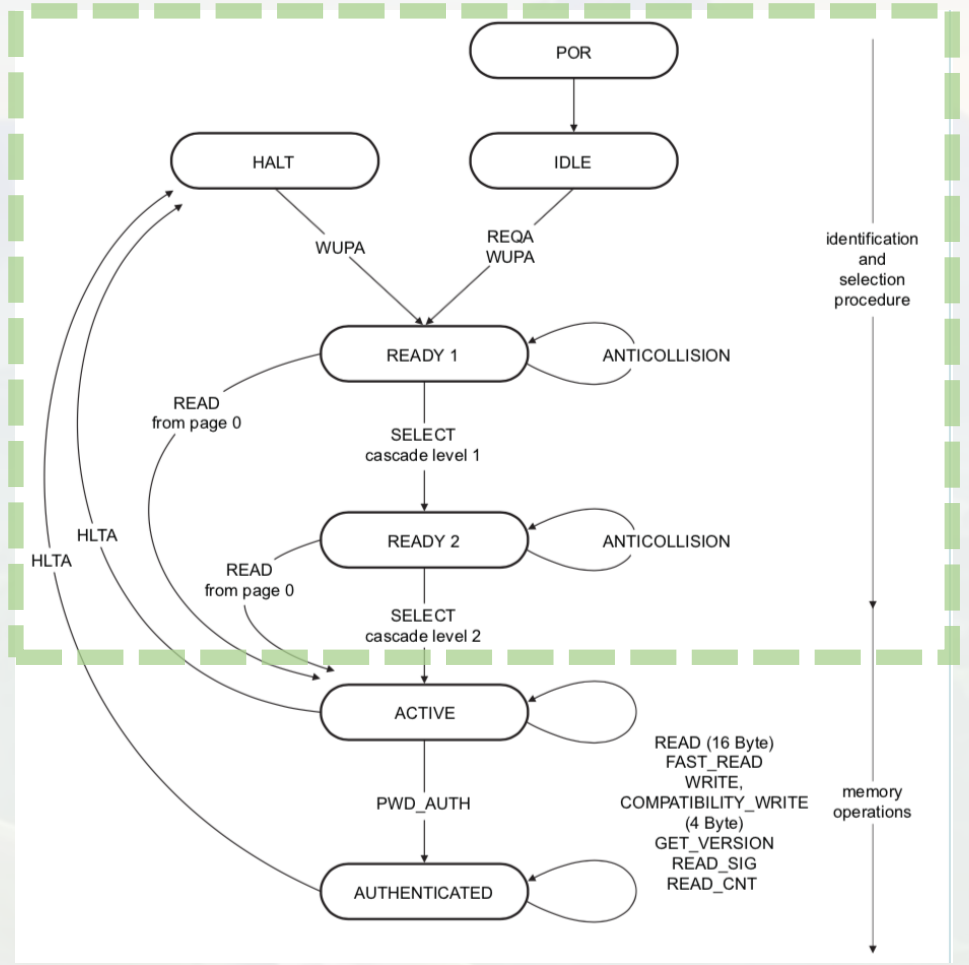
Fig 2. Block diagram of NTAG213/215/216

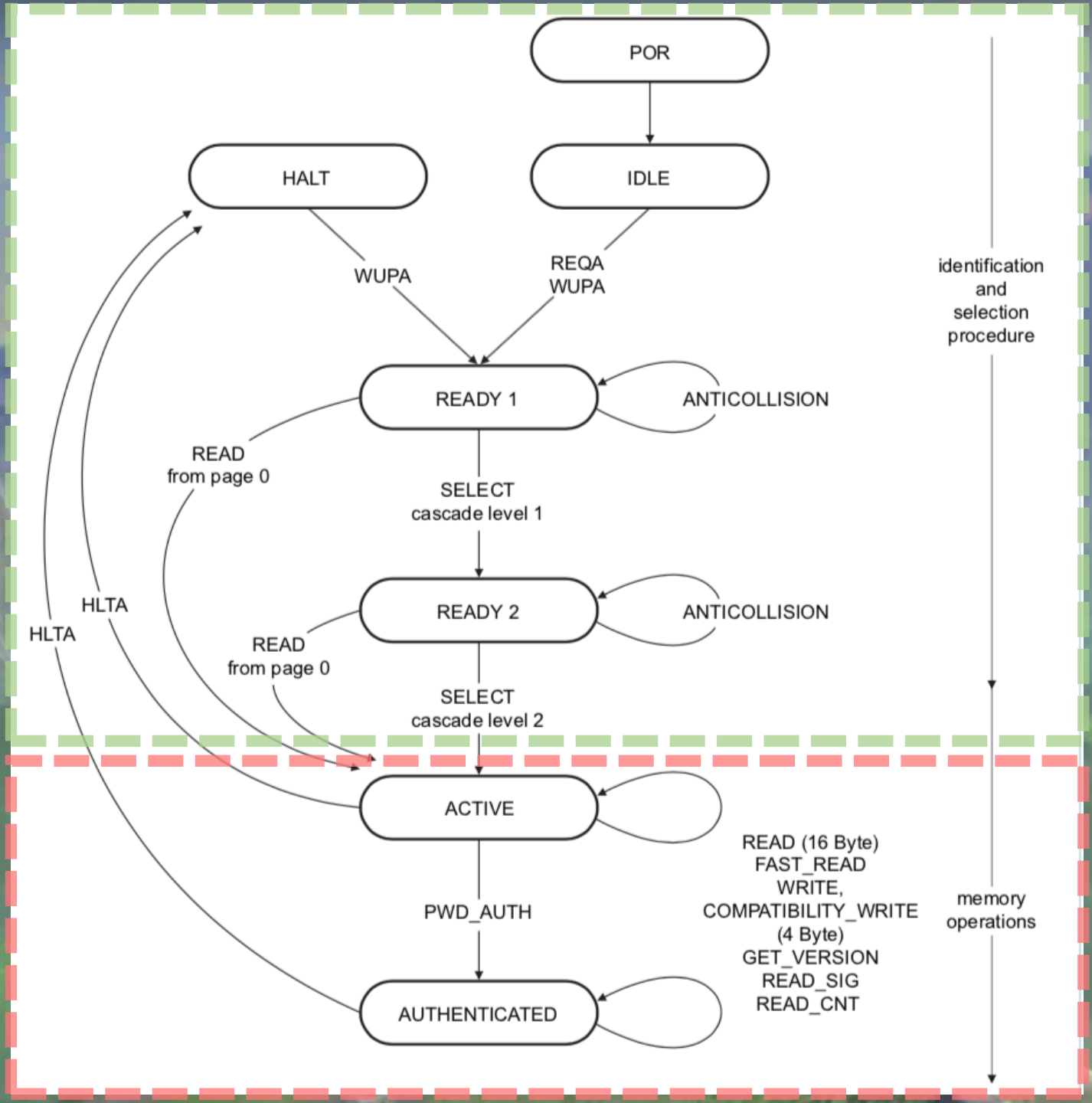
Getting Started

Proxmark 3

- Has basic ISO 14443A simulation
- Supports initial wakeup, identification, and selection (anti-collision)
- Can tell an NFC reader its serial number. That's all.

Need to roll our own NTAG21x simulator.




POR
HALT
IDLE
WUPA
REQA
WUPA
identification
and
selection
procedure
READY 1
ANTICOLLISION
READ
from page 0
SELECT
cascade level 1
READY 2
ANTICOLLISION
HLTA
HLTA
READ
from page 0
SELECT
cascade level 2
ACTIVE
READ (16 Byte)
FAST_READ
WRITE,
COMPATIBILITY_WRITE
(4 Byte)
GET_VERSION
READ_SIG
READ_CNT
memory
operations
PWD_AUTH
AUTHENTICATED

Getting Started

To read, edit, and simulate Amiibo we need to:

- Make an NTAG215 simulator for Proxmark

- Obtain the Amiibo crypto keys

+

- Integrate Amiibo crypto operations with Proxmark simulator

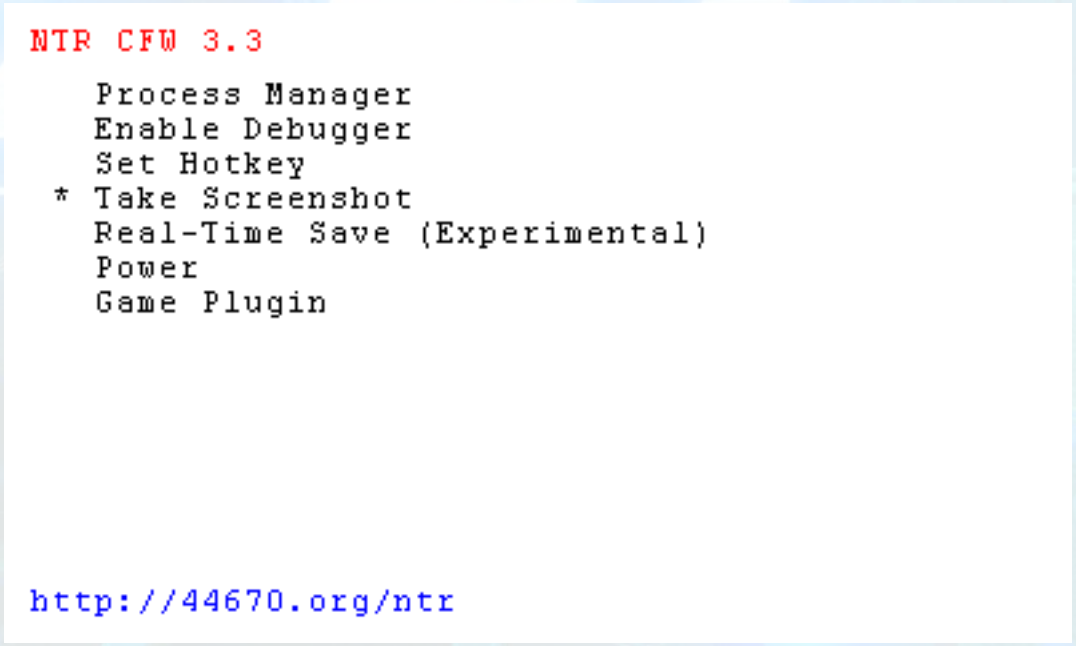

Crypto Caverns

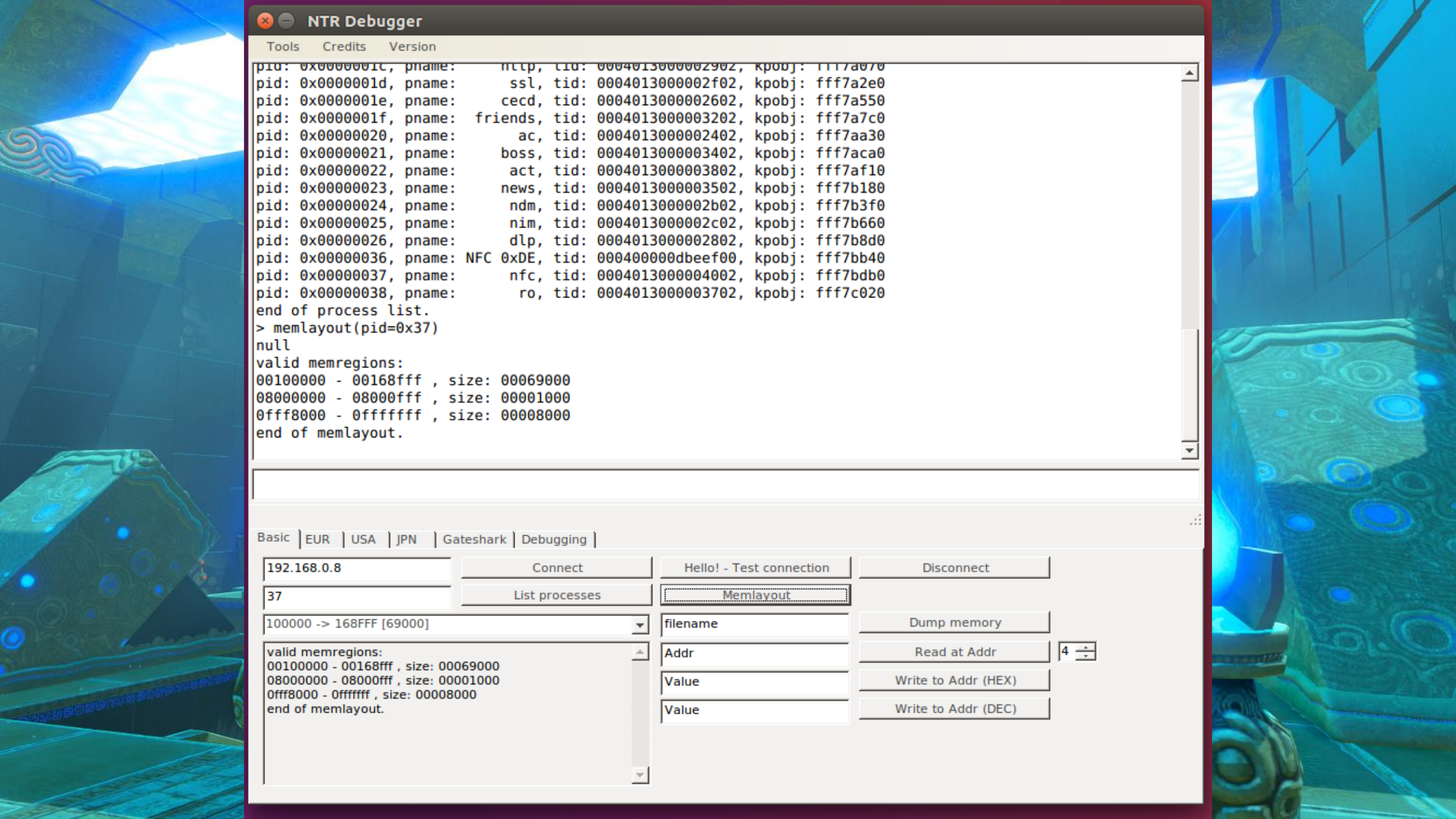
Crypto Caverns

- Jailbreaks already existed for 3DS
  - New 3DS has built-in NFC/Amiibo compatibility
  - (watch the “Breaking the 3DS” CCC presentation!)
- Reverse engineering the software will be much easier
  - Can get decrypted game and service binaries
  - Can tamper with processes

Crypto Caverns

- NTR CFW: Custom firmware for 3DS
- Adds a debugger to the 3DS
- Sadly, breakpoints don’t work
- Limited to peek & poke
  - Read and tamper with memory
  - View registers at random moments



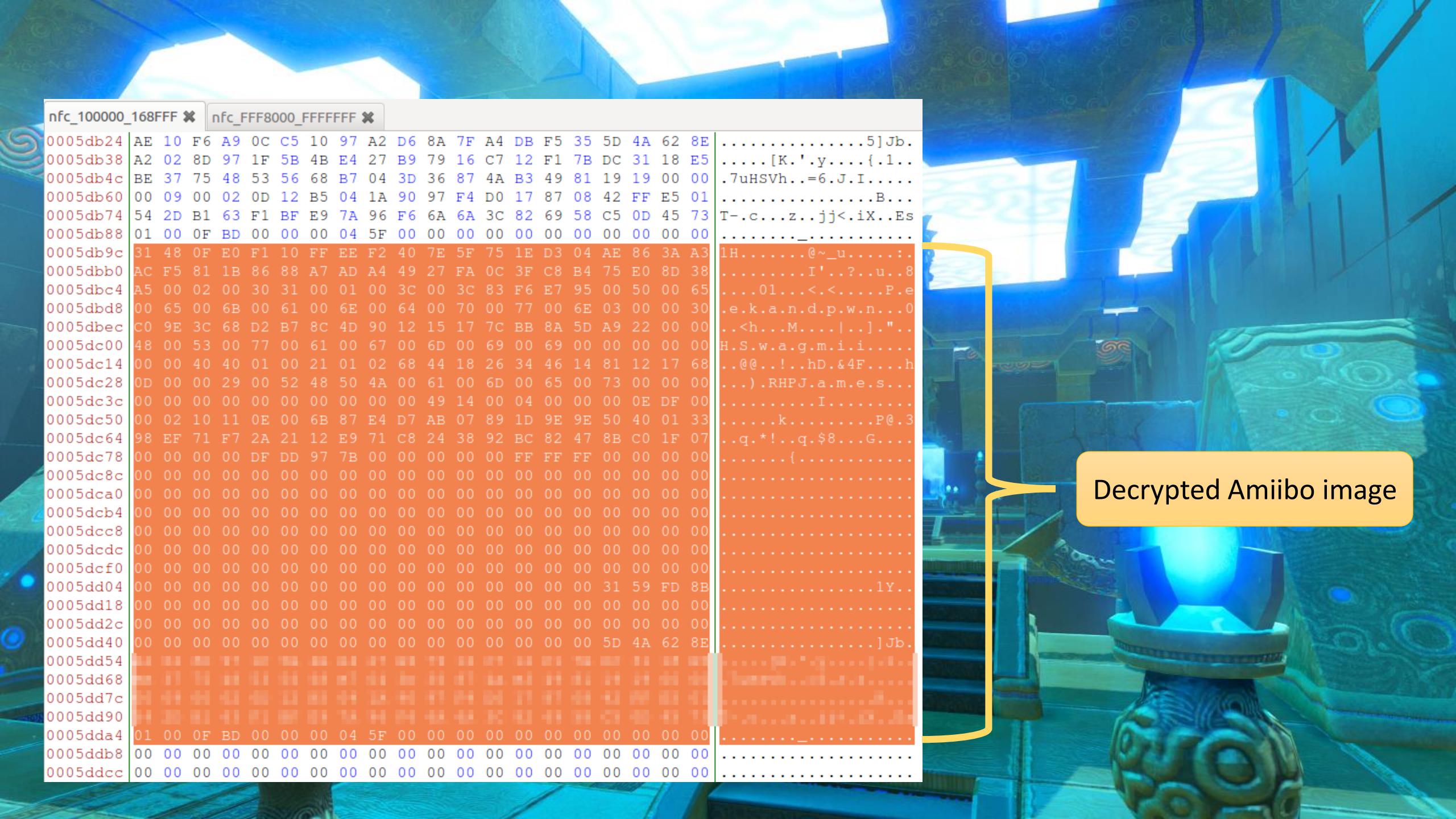

NTR Debugger
Tools
Credits
Version
pid: 0x0000001c, pname: http, tid: 0004013000002902, kpobj: fff7a070
pid: 0x0000001d, pname: ssl, tid: 0004013000002f02, kpobj: fff7a2e0
pid: 0x0000001e, pname: cecd, tid: 0004013000002602, kpobj: fff7a550
pid: 0x0000001f, pname: friends, tid: 0004013000003202, kpobj: fff7a7c0
pid: 0x00000020, pname: ac, tid: 0004013000002402, kpobj: fff7aa30
pid: 0x00000021, pname: boss, tid: 0004013000003402, kpobj: fff7aca0
pid: 0x00000022, pname: act, tid: 0004013000003802, kpobj: fff7af10
pid: 0x00000023, pname: news, tid: 0004013000003502, kpobj: fff7b180
pid: 0x00000024, pname: ndm, tid: 0004013000002b02, kpobj: fff7b3f0
pid: 0x00000025, pname: nim, tid: 0004013000002c02, kpobj: fff7b660
pid: 0x00000026, pname: dlp, tid: 0004013000002802, kpobj: fff7b8d0
pid: 0x00000036, pname: NFC 0xDE, tid: 000400000dbeef00, kpobj: fff7bb40
pid: 0x00000037, pname: nfc, tid: 0004013000004002, kpobj: fff7bdb0
pid: 0x00000038, pname: ro, tid: 0004013000003702, kpobj: fff7c020
end of process list.
> memlayout(pid=0x37)
null
valid memregions:
00100000 - 00168fff , size: 00069000
08000000 - 08000fff , size: 00001000
0fff8000 - 0fffffff , size: 00008000
end of memlayout.
Basic
EUR
USA
JPN
Gateshark
Debugging
192.168.0.8
Connect
Hello! - Test connection
Disconnect
37
List processes
Memlayout
100000 -> 168FFF [69000]
filename
Dump memory
valid memregions:
00100000 - 00168fff , size: 00069000
08000000 - 08000fff , size: 00001000
0fff8000 - 0fffffff , size: 00008000
end of memlayout.
Addr
Read at Addr
4
Value
Write to Addr (HEX)
Value
Write to Addr (DEC)

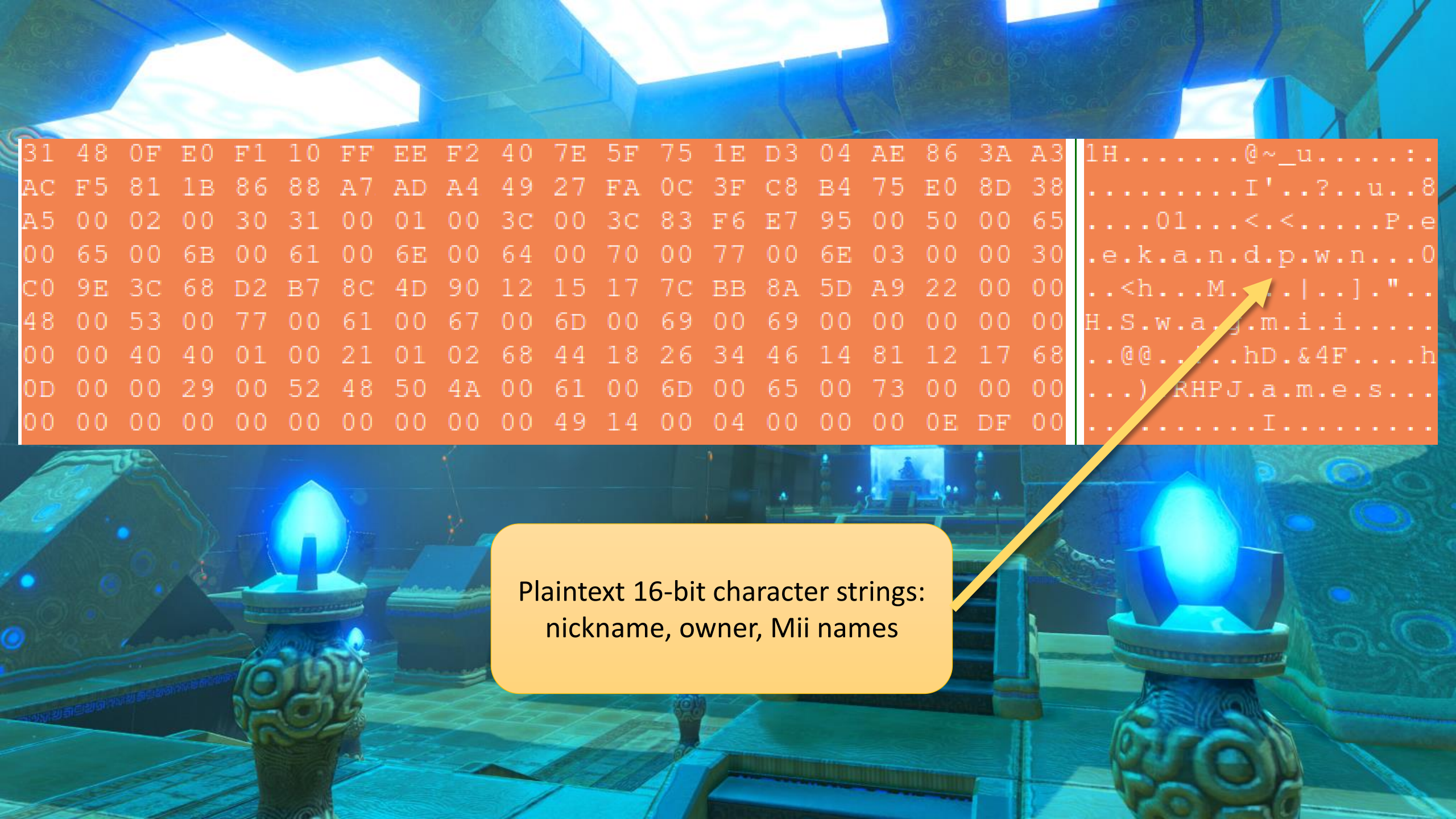
Crypto Caverns

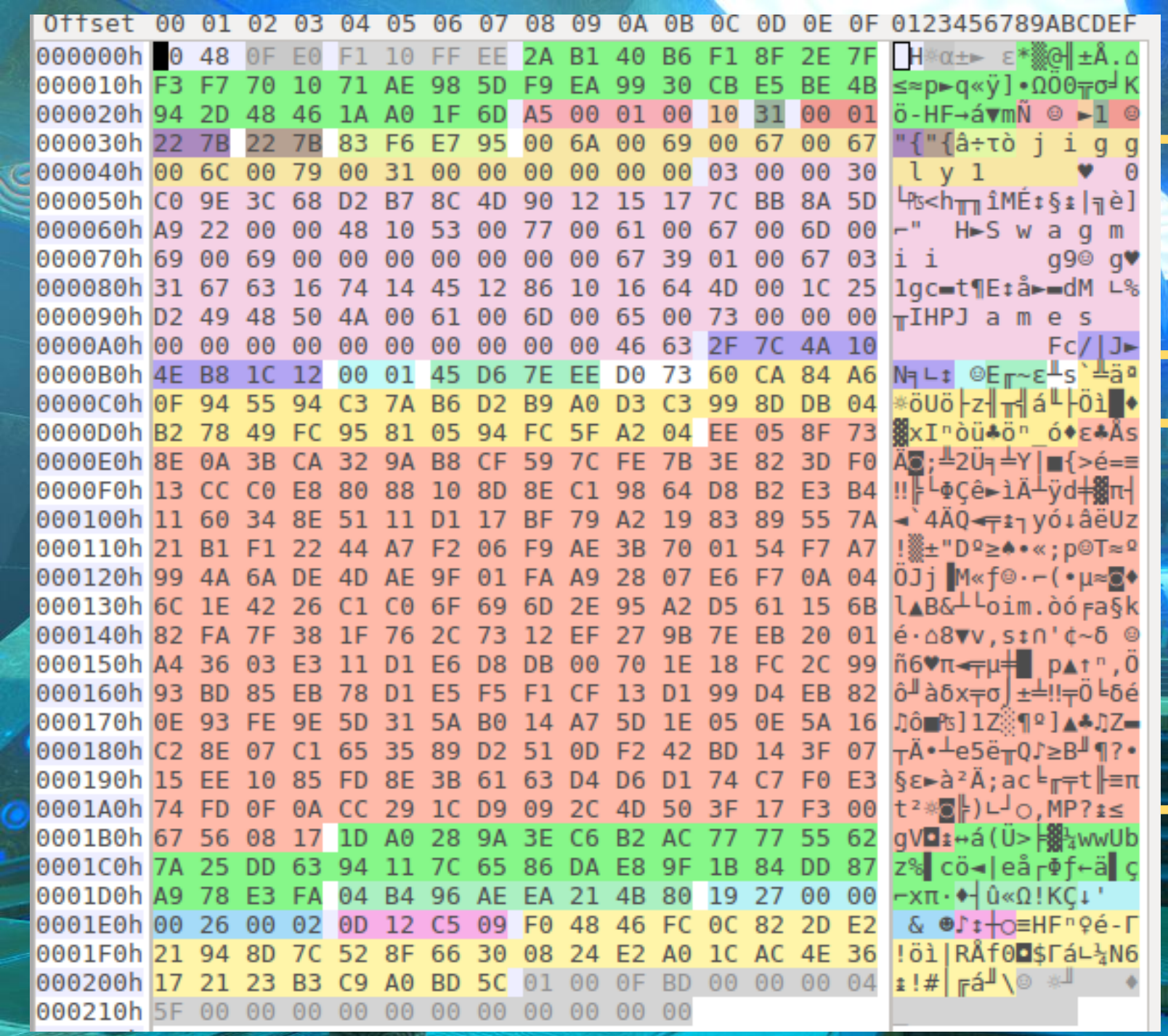
- Try to dump and reverse engineer the NFC service to:
  - Find the crypto keys
  - Fully understand crypto system
  - Understand how games, NFC service, and Amiibo interact
- The encrypted or decrypted Amiibo images may be accessible in dumped memory
- Static analysis of binary

Crypto Caverns

- Wrote a custom homebrew app that uses the NFC service:
  - Scan for Amiibo
  - Read its content
  - Request app data, tag info (can’t request raw dump)


nfc_100000_168FFF
nfc_FFF8000_FFFFFFF
0005db24 AE 10 F6 A9 0C C5 10 97 A2 D6 8A 7F A4 DB F5 35 5D 4A 62 8E .....5]Jb.
0005db38 A2 02 8D 97 1F 5B 4B E4 27 B9 79 16 C7 12 F1 7B DC 31 18 E5 .....[K.'.y....{.1..
0005db4c BE 37 75 48 53 56 68 B7 04 3D 36 87 4A B3 49 81 19 19 00 00 .7uHSVh..=6.J.I.....
0005db60 00 09 00 02 0D 12 B5 04 1A 90 97 F4 D0 17 87 08 42 FF E5 01 .....B...
0005db74 54 2D B1 63 F1 BF E9 7A 96 F6 6A 6A 3C 82 69 58 C5 0D 45 73 T-.c...z..jj<.iX..Es
0005db88 01 00 0F BD 00 00 00 04 5F 00 00 00 00 00 00 00 00 00 00 00 ...._.....
0005db9c 31 48 0F E0 F1 10 FF EE F2 40 7E 5F 75 1E D3 04 AE 86 3A A3 1H.....@~_u.....:.
0005dbb0 AC F5 81 1B 86 88 A7 AD A4 49 27 FA 0C 3F C8 B4 75 E0 8D 38 .....I'..?..u..8
0005dbc4 A5 00 02 00 30 31 00 01 00 3C 00 3C 83 F6 E7 95 00 50 00 65 ....01...<.<.....P.e
0005dbd8 00 65 00 6B 00 61 00 6E 00 64 00 70 00 77 00 6E 03 00 00 30 .e.k.a.n.d.p.w.n...0
0005dbec C0 9E 3C 68 D2 B7 8C 4D 90 12 15 17 7C BB 8A 5D A9 22 00 00 ..<h...M....|..].".
0005dc00 48 00 53 00 77 00 61 00 67 00 6D 00 69 00 69 00 00 00 00 00 H.S.w.a.g.m.i.i.....
0005dc14 00 00 40 40 01 00 21 01 02 68 44 18 26 34 46 14 81 12 17 68 ..@@..!..hD.&4F....h
0005dc28 0D 00 00 29 00 52 48 50 4A 00 61 00 6D 00 65 00 73 00 00 00 ...).RHPJ.a.m.e.s...
0005dc3c 00 00 00 00 00 00 00 00 00 00 49 14 00 04 00 00 00 0E DF 00 ....I.........
0005dc50 00 02 10 11 0E 00 6B 87 E4 D7 AB 07 89 1D 9E 9E 50 40 01 33 ......k.........P@.3
0005dc64 98 EF 71 F7 2A 21 12 E9 71 C8 24 38 92 BC 82 47 8B C0 1F 07 ..q.*!..q.$8...G....
0005dc78 00 00 00 00 DF DD 97 7B 00 00 00 00 00 FF FF FF 00 00 00 00 ....{............
0005dc8c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0005dca0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0005dcb4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0005dcc8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0005dcdc 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0005dcf0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0005dd04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 31 59 FD 8B ....1Y..
0005dd18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0005dd2c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0005dd40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 5D 4A 62 8E ....]Jb.
0005dd54
0005dd68
0005dd7c
0005dd90
0005dda4 01 00 0F BD 00 00 00 04 5F 00 00 00 00 00 00 00 00 00 00 00 ...._.....
0005ddb8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0005ddcc 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
Decrypted Amiibo image

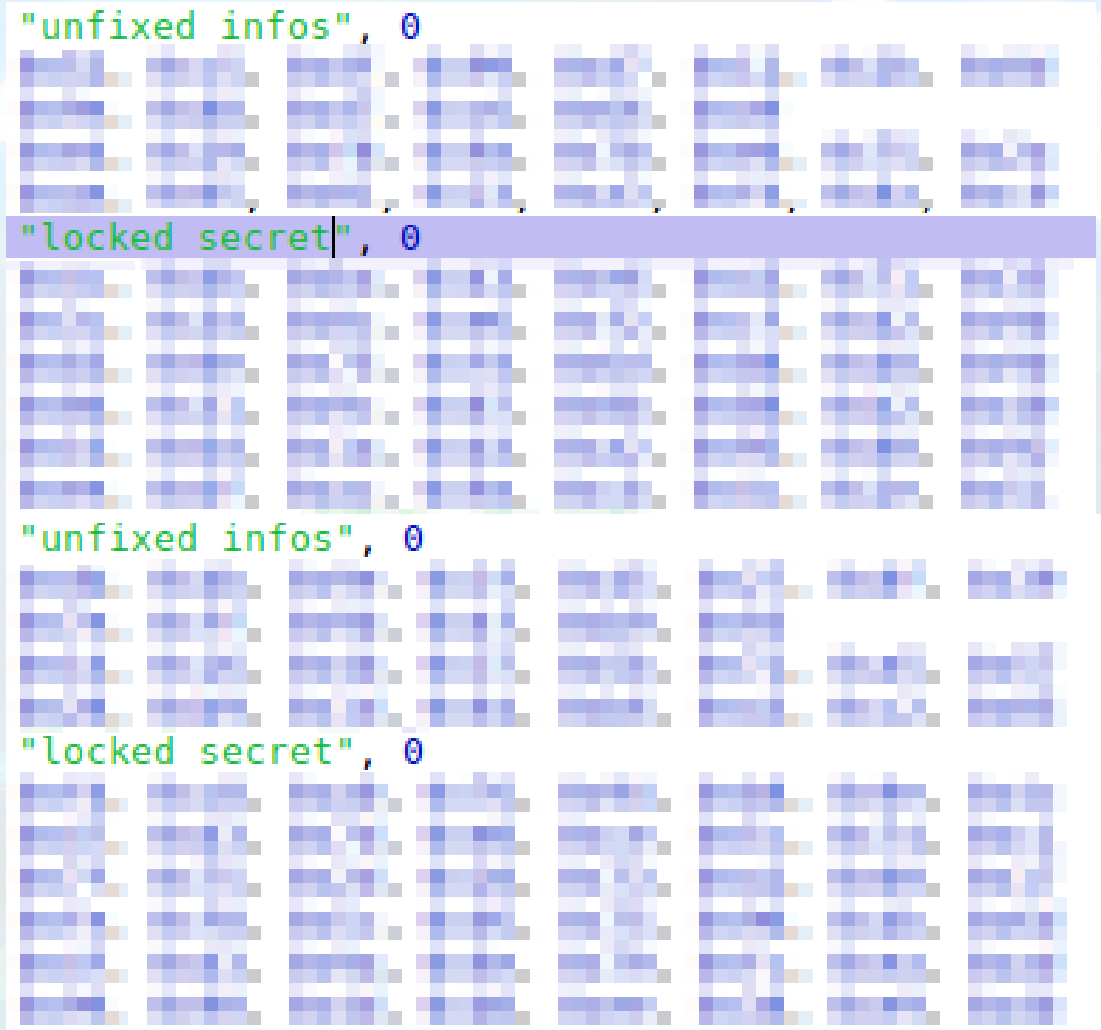

31 48 0F E0 F1 10 FF EE F2 40 7E 5F 75 1E D3 04 AE 86 3A A3
AC F5 81 1B 86 88 A7 AD A4 49 27 FA 0C 3F C8 B4 75 E0 8D 38
A5 00 02 00 30 31 00 01 00 3C 00 3C 83 F6 E7 95 00 50 00 65
00 65 00 6B 00 61 00 6E 00 64 00 70 00 77 00 6E 03 00 00 30
C0 9E 3C 68 D2 B7 8C 4D 90 12 15 17 7C BB 8A 5D A9 22 00 00
48 00 53 00 77 00 61 00 67 00 6D 00 69 00 69 00 00 00 00 00
00 00 40 40 01 00 21 01 02 68 44 18 26 34 46 14 81 12 17 68
0D 00 00 29 00 52 48 50 4A 00 61 00 6D 00 65 00 73 00 00 00
00 00 00 00 00 00 00 00 00 00 49 14 00 04 00 00 00 0E DF 00
Plaintext 16-bit character strings:
nickname, owner, Mii names


Offset 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 0123456789ABCDEF
000000h 00 48 0F E0 F1 10 FF EE 2A B1 40 B6 F1 8F 2E 7F
000010h F3 F7 70 10 71 AE 98 5D F9 EA 99 30 CB E5 BE 4B
000020h 94 2D 48 46 1A A0 1F 6D A5 00 01 00 10 31 00 01
000030h 22 7B 22 7B 83 F6 E7 95 00 6A 00 69 00 67 00 67
000040h 00 6C 00 79 00 31 00 00 00 00 00 00 03 00 00 30
000050h C0 9E 3C 68 D2 B7 8C 4D 90 12 15 17 7C BB 8A 5D
000060h A9 22 00 00 48 10 53 00 77 00 61 00 67 00 6D 00
000070h 69 00 69 00 00 00 00 00 00 00 67 39 01 00 67 03
000080h 31 67 63 16 74 14 45 12 86 10 16 64 4D 00 1C 25
000090h D2 49 48 50 4A 00 61 00 6D 00 65 00 73 00 00 00
0000A0h 00 00 00 00 00 00 00 00 00 00 46 63 2F 7C 4A 10
0000B0h 4E B8 1C 12 00 01 45 D6 7E EE D0 73 60 CA 84 A6
0000C0h 0F 94 55 94 C3 7A B6 D2 B9 A0 D3 C3 99 8D DB 04
0000D0h B2 78 49 FC 95 81 05 94 FC 5F A2 04 EE 05 8F 73
0000E0h 8E 0A 3B CA 32 9A B8 CF 59 7C FE 7B 3E 82 3D F0
0000F0h 13 CC C0 E8 80 88 10 8D 8E C1 98 64 D8 B2 E3 B4
000100h 11 60 34 8E 51 11 D1 17 BF 79 A2 19 83 89 55 7A
000110h 21 B1 F1 22 44 A7 F2 06 F9 AE 3B 70 01 54 F7 A7
000120h 99 4A 6A DE 4D AE 9F 01 FA A9 28 07 E6 F7 0A 04
000130h 6C 1E 42 26 C1 C0 6F 69 6D 2E 95 A2 D5 61 15 6B
000140h 82 FA 7F 38 1F 76 2C 73 12 EF 27 9B 7E EB 20 01
000150h A4 36 03 E3 11 D1 E6 D8 DB 00 70 1E 18 FC 2C 99
000160h 93 BD 85 EB 78 D1 E5 F5 F1 CF 13 D1 99 D4 EB 82
000170h 0E 93 FE 9E 5D 31 5A B0 14 A7 5D 1E 05 0E 5A 16
000180h C2 8E 07 C1 65 35 89 D2 51 0D F2 42 BD 14 3F 07
000190h 15 EE 10 85 FD 8E 3B 61 63 D4 D6 D1 74 C7 F0 E3
0001A0h 74 FD 0F 0A CC 29 1C D9 09 2C 4D 50 3F 17 F3 00
0001B0h 67 56 08 17 1D A0 28 9A 3E C6 B2 AC 77 77 55 62
0001C0h 7A 25 DD 63 94 11 7C 65 86 DA E8 9F 1B 84 DD 87
0001D0h A9 78 E3 FA 04 B4 96 AE EA 21 4B 80 19 27 00 00
0001E0h 00 26 00 02 0D 12 C5 09 F0 48 46 FC 0C 82 2D E2
0001F0h 21 94 8D 7C 52 8F 66 30 08 24 E2 A0 1C AC 4E 36
000200h 17 21 23 B3 C9 A0 BD 5C 01 00 0F BD 00 00 00 04
000210h 5F 00 00 00 00 00 00 00 00 00 00 00 00
Signature, settings, timestamps, write counter
Amiibo nickname; Registered owner Mii
Application ID, hash
Game/Application data
Signature, serial, character ID

Crypto Caverns

- Loaded up the NFC service binary in Hopper
- Searched for intelligible strings
- Immediately found… something suspicious
  - “locked secret” and “unfixed infos”

Crypto Caverns



Crypto Caverns

- Need to figure out how these bytes are used
- Difficulties with static analysis
  - Poor automatic code identification due to the way the service binary is set up
  - No main entry point that calls down to all the other code
  - Series of service call stubs that call isolated functions

Crypto Caverns

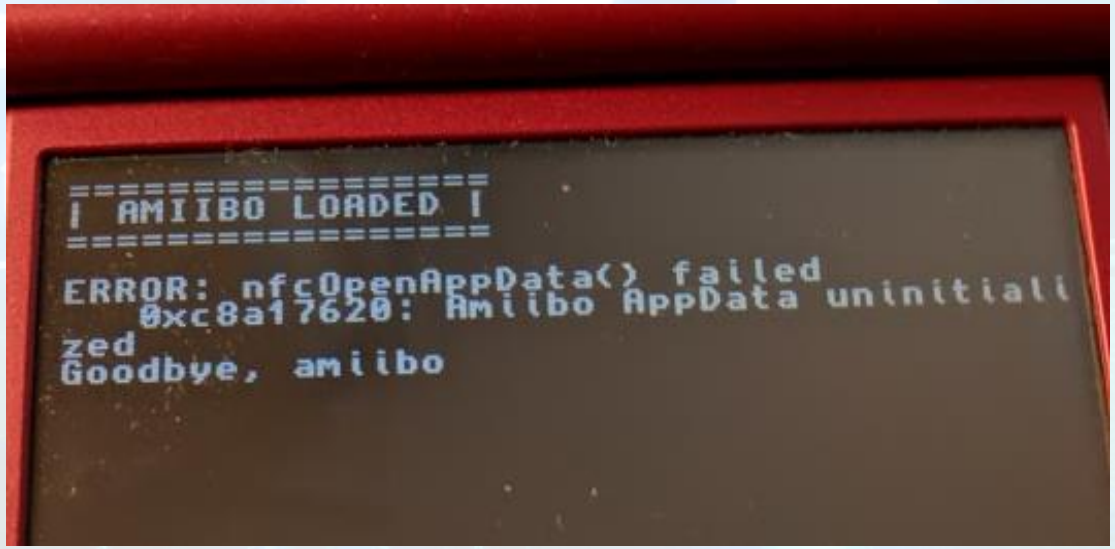
- Broken debugger, tons of unidentified code, disassembler crashes on Ctrl+F… what can I do?

Crypto Caverns

- Use existing reverse-engineered NFC service IPC calls from the homebrew library to figure out the locations of related code in the binary dump

Crypto Caverns

- `0xC8A17620` “Invalid state” error code appears often when calling functions at the wrong time



Crypto Caverns

- Add new features to custom app to help identify code in the binary
- Version 1
  - Add a bunch of button bindings to trigger all known IPC calls
  - Connect debugger
  - Replace “invalid state” error code with `0xDEADBEEF` and call random functions
  - Find where `0xDEADBEEF` error shows up


NFC Status: Scanning in progress
NFC tag ID: 04 b4 96 ea 21 4b 80
Country code: 00
Initialized 01/03/2018
Owner Mii: Mr. Mii
Nickname: jiggly
App data:
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
L: Init app data
A: Repopulate app data
B: nfcExit()
X: nfcLoadAmiiboData()
Y: nfcStartScanning()x2
R: Write app data
D-Left: Smash data
D-Right: reset scan state
D-Up: Cycle IPC
D-Down: Comm Status
Press Start to exit.

Crypto Caverns

- Version 2
  - Replace the “invalid state” error codes with address of the function that sets them
  - Button mash and find lots more addresses

Crypto Caverns

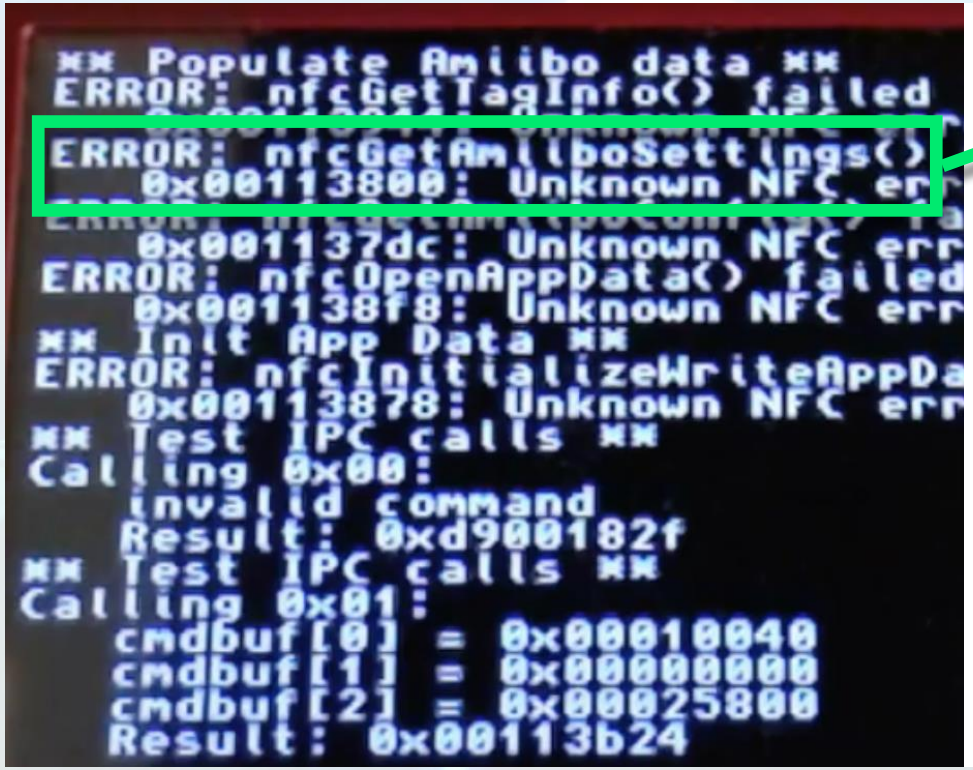
- Version 3
  - Programmatically rewrite select function prologues to immediately return address of function
  - Works even if not in invalid state
  - Observe function addresses in error codes
  - Navigate down found branch and select more functions to rewrite

```
def make_ops(writer, target, pid):
    start_addr = target[0]
    regs = target[1]

    payload = ()
    payload += (regs, 0xb5)     # push <regs>
    payload += (0x78, 0x46)     # mov r0, pc
    payload += (0x06, 0x38)     # sub r0, r0, #6 ; fix PC
    payload += (regs, 0xbd)     # pop <regs>

    writer(start_addr, payload, pid)
    return 'nc.write(0x%x, %s, pid=0x%x)' % (target[0],
                                             str(payload),
                                             pid)
```

Crypto Caverns




```
                stub_nfcGetAmiiboSettings:
00113800            push        {r4, lr}
00113802            mov         r4, r1
00113804            movs        r1, #0x11
00113806            lsls        r1, r1, #0x5
00113808            adds        r0, r0, r1
0011380a            ldrb        r0, [r0]
0011380c            cmp         r0, #0x2
0011380e            beq         stub_nfcGetAmiiboSettings+20
-------------------------------------------------------------------------------
00113810            ldr         r0, = 0xc8a17600
00113812            pop         {r4, pc}
                              ; endp
-------------------------------------------------------------------------------
00113814            bl          usedBeforeAmiiboAPIcall
00113818            mov         r1, r4
0011381a            bl          nfcGetAmiiboSettings
0011381e            pop         {r4, pc}
                              ; endp
00113820            dd          0xc8a17600
-------------------------------------------------------------------------------
```

```
keyinfo_stuff_happens_here:
push        {r0, r1, r2, r3, r4, r5, r6, r7, lr}
cmp         r2, #0x40
sub         sp, #0x54
mov         r4, r1
mov         r5, r3
beq         0x114674
```

```
0x114674:
movs        r1, #0x40
add         r0, sp, #0x10
blx         0x11db28
movs        r2, #0x40
mov         r1, r4
add         r0, sp, #0x10
blx         memcpy
ldr         r0, = 0x1ff80000
ldrb        r0, [r0, #0x14]
lsls        r0, r0, #0x1f
asrs        r0, r0, #0x1f
adds        r0, r0, #0x1
beq         0x114696
```

```
0x114692:
ldr         r4, = 0x135584
b           0x11469a
```

```
0x114696:
ldr         r4, = 0x135584
adds        r4, #0x6a
```

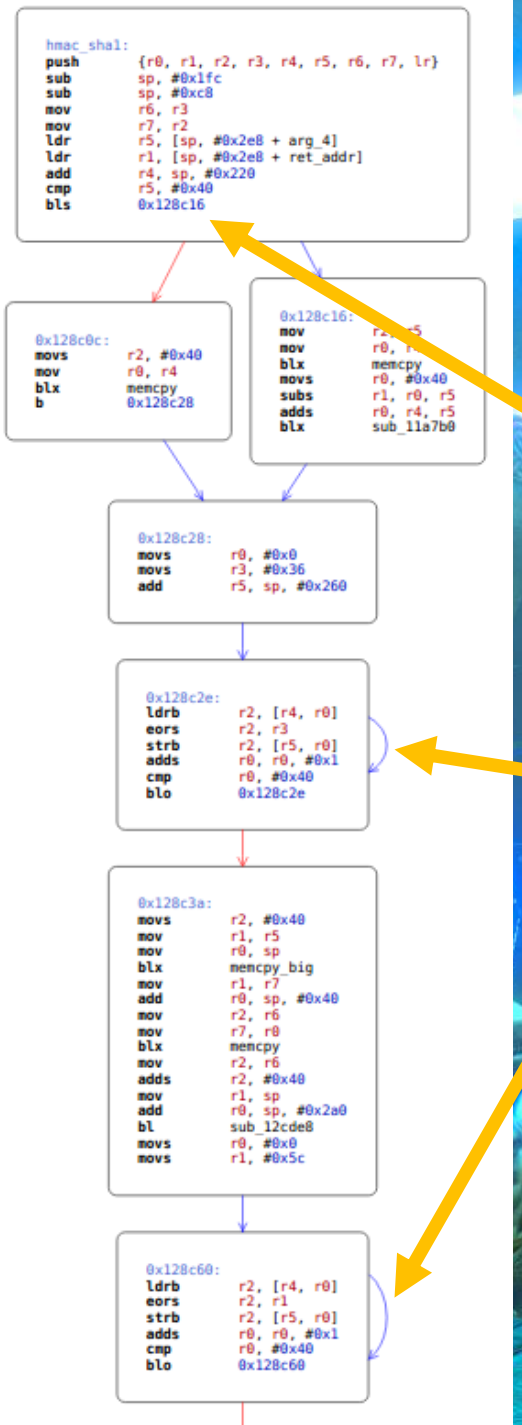
ENVINFO [edit]

Bit	Description
0	Clear for developer unit, set for retail.(See 0x1FF80015)
1	IsJtagConnected
2-7	?

Check last bit of ENVINFO in config memory

Pick developer or retail key info

HMAC Algorithm




```
Keys longer than blockSize are shortened by hashing them
if (length(key) > blockSize) then
   key ← hash(key) //Key becomes outputSize bytes long

Keys shorter than blockSize are padded to blockSize by padding with zeros on the right
if (length(key) < blockSize) then
   key ← Pad(key, blockSize)  //pad key with zeros to make it blockSize bytes long

o_key_pad = key xor [0x5c * blockSize]   //Outer padded key
i_key_pad = key xor [0x36 * blockSize]   //Inner padded key

return hash(o_key_pad ∥ hash(i_key_pad ∥ message)) //Where ∥ is concatenation
```

```
int lots_of_amiibo_crypto(int arg0, int arg1) {
    sp = sp - 0x440;
    r5 = arg0;
    memcpy(sp + 0x2c, arg1 + 0x10, 0x24);
    memcpy(sp + 0x1b8, arg1 + 0x34, 0x20);
    memcpy(sp + 0x1e0, arg1 + 0x54, 0x2c);
    memcpy(sp + 0xc, arg1 + 0x80, 0x20);
    memcpy(sp + 0x50, arg1 + 0xa0, 0x168);
    var_1D8 = sp + 0x20c;
    memcpy(sp + 0x20c, arg1 + (0x41 << 0x3), 0x14);
    sub_11db28(sp + 0x220, 0x40);
    memcpy(sp + 0x240, arg1 + 0x60, 0x20);
    *(sp + 0x2c) = 0xa5;
    *(sp + 0x2d) = 0x0;
```

Hopper pseudo-code of NFC service

```
}

void nfc3d_amiibo_tag_to_internal(const uint8_
l) {
  memcpy(intl + 0x000, tag + 0x008, 0x008); //
e & Lock bytes & capability container
  memcpy(intl + 0x008, tag + 0x080, 0x020); //
  memcpy(intl + 0x028, tag + 0x010, 0x024); //
  memcpy(intl + 0x04C, tag + 0x0A0, 0x168); //
    memcpy(intl + 0x1B4, tag + 0x034, 0x020);
    memcpy(intl + 0x1D4, tag + 0x000, 0x008);
    memcpy(intl + 0x1DC, tag + 0x054, 0x02C);
```

amiitool

Crypto Caverns

- Tried to figure out how to use it with amiitool, but there isn't enough data to build a full key file
- Using the content I did have, I tried searching for the keys online…

Crypto Caverns

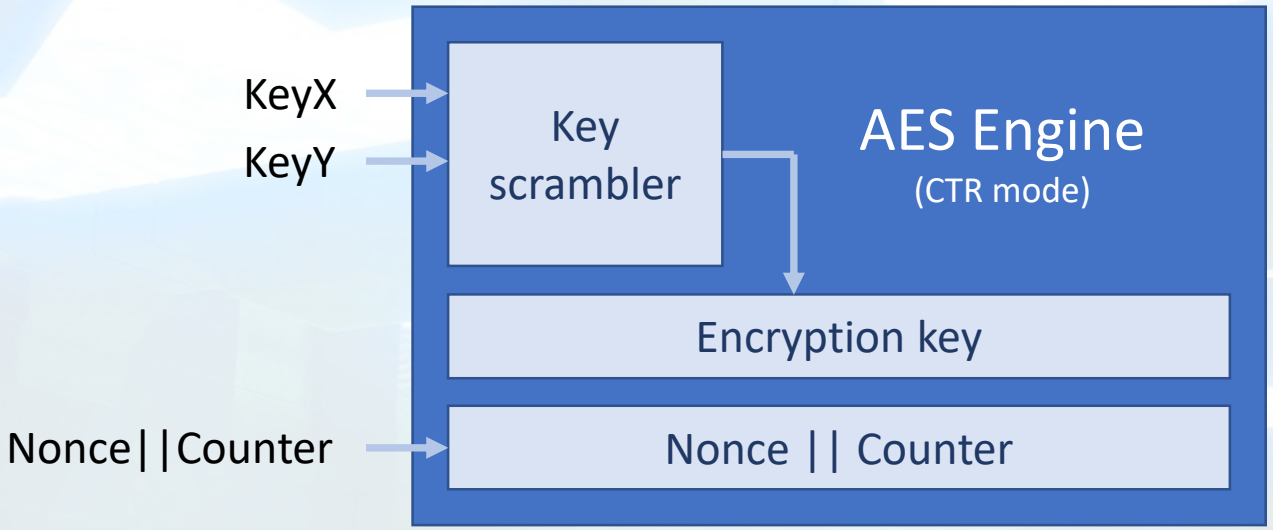
- Found some interesting notes…
- Found the whole key file…
- It contains a value not in the NFC service dump; what is it?
- What hasn't shown up yet? The 3DS's dedicated AES hardware

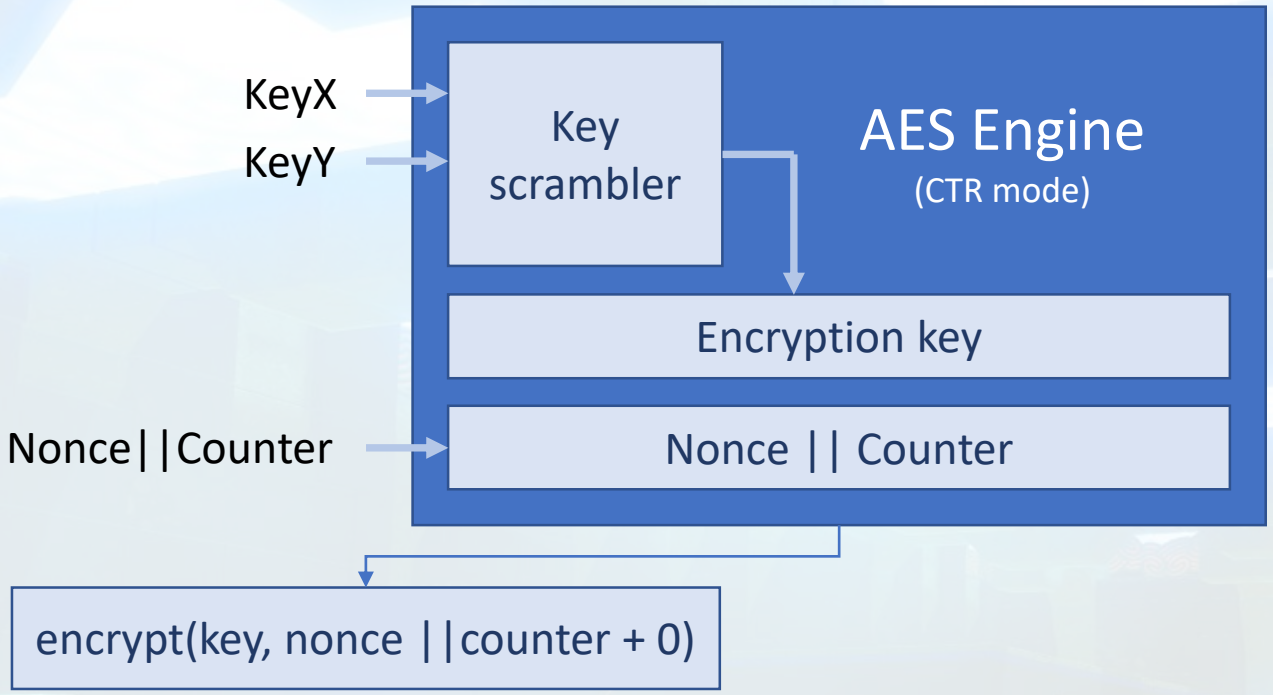
Crypto Caverns

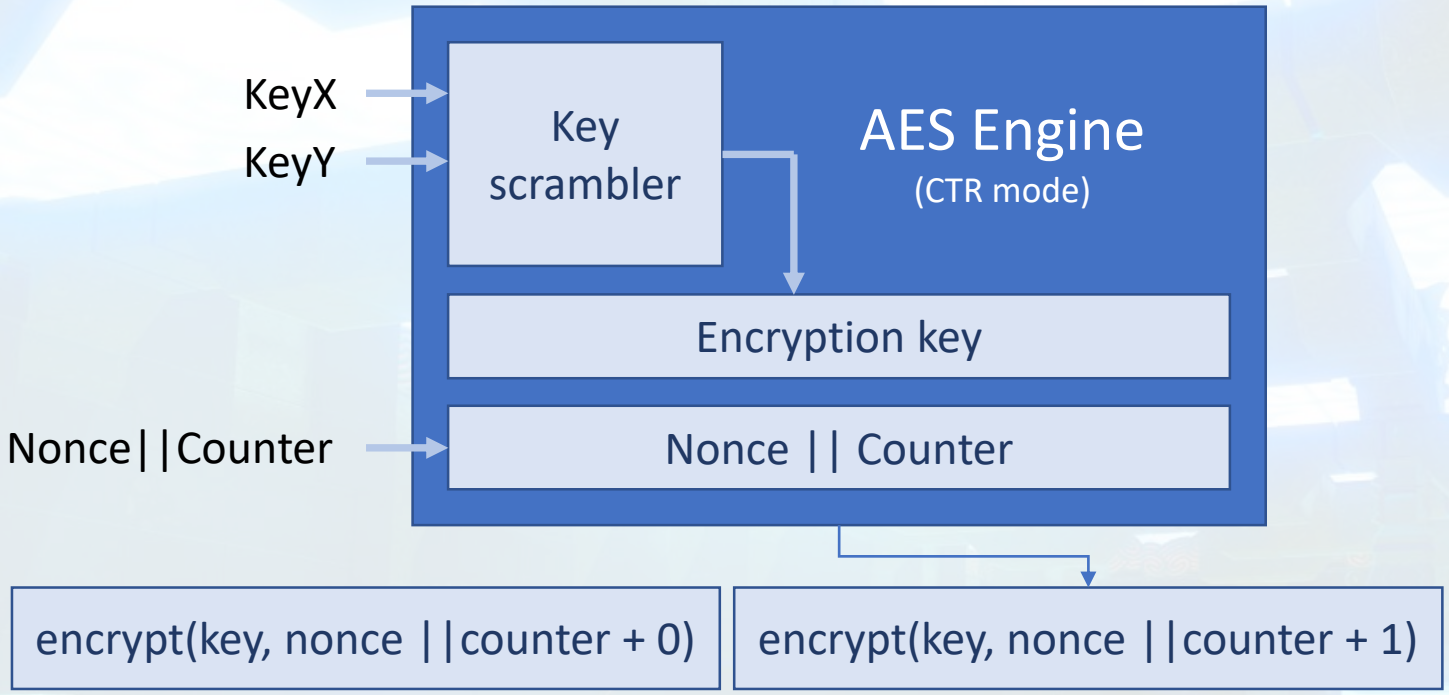
- Two key components are sent to AES engine
  - “keyX” set by bootloader
  - “keyY” set by firmware after it’s decrypted
- Setting keyY triggers generation of the “normal” key
- Normal key can be used to perform crypto operations in the engine, but can’t be read back out

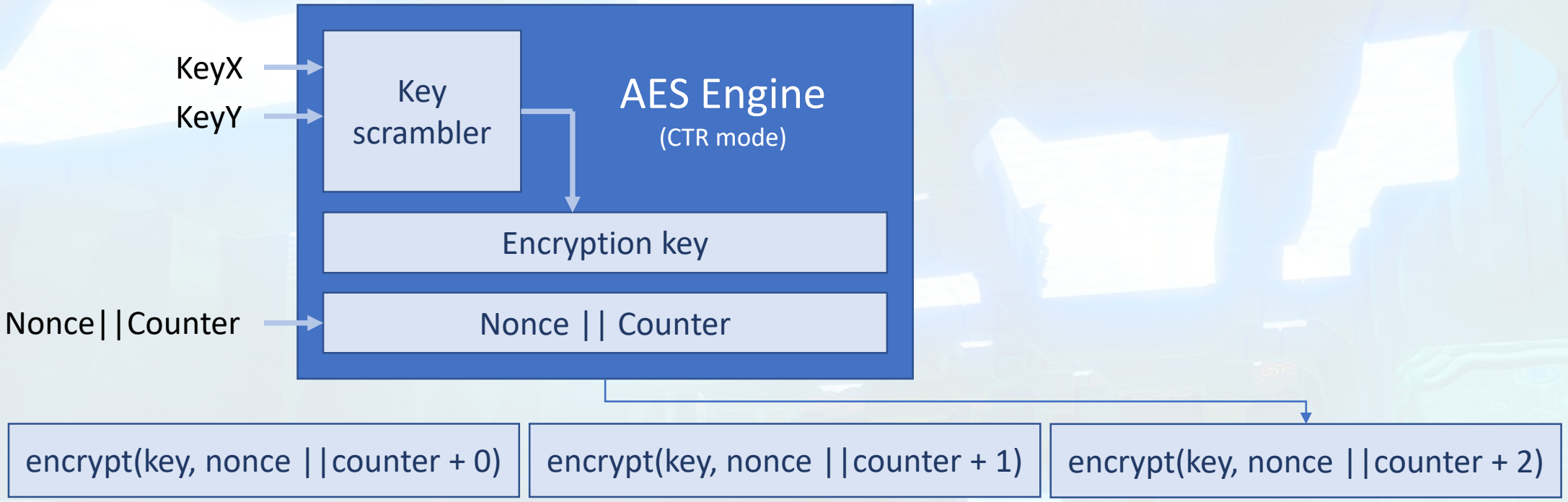
Crypto Caverns

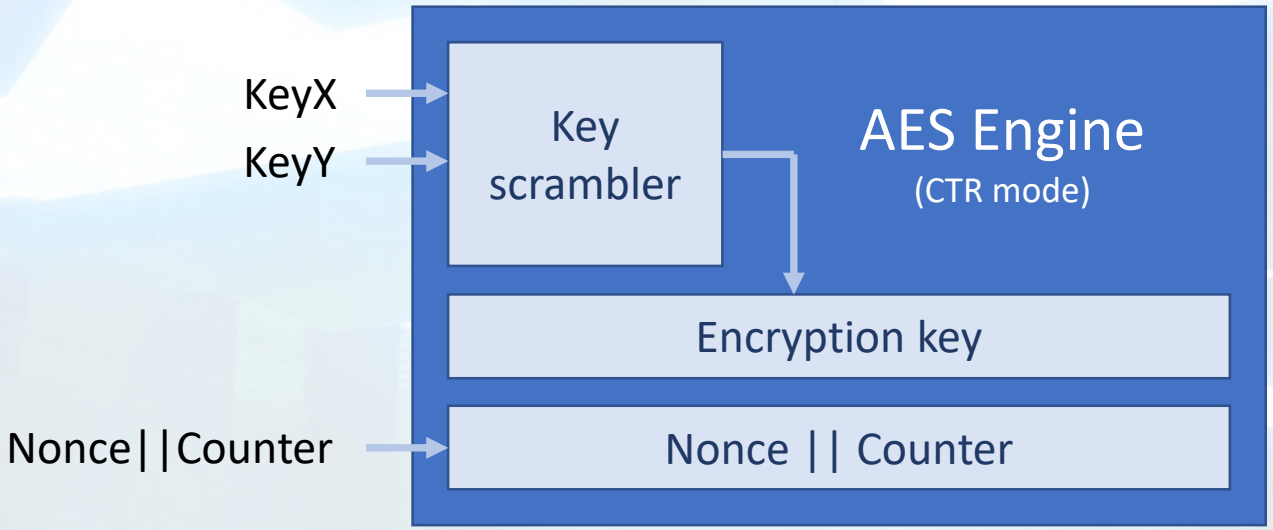
- “Key scrambling” algorithm performed by the AES engine was reversed by derrek, plutoo, smealum; presented in “Breaking the 3DS” at CCC in 2015
- All normal keys can be derived by applying this algorithm to the keyX and keyY components
- AES crypto could then be performed without the 3DS hardware, but before that…


KeyX
KeyY
Key scrambler
AES Engine
(CTR mode)
Encryption key
Nonce||Counter
Nonce || Counter


KeyX
KeyY
Key scrambler
AES Engine
(CTR mode)
Encryption key
Nonce||Counter
Nonce || Counter
encrypt(key, nonce ||counter + 0)

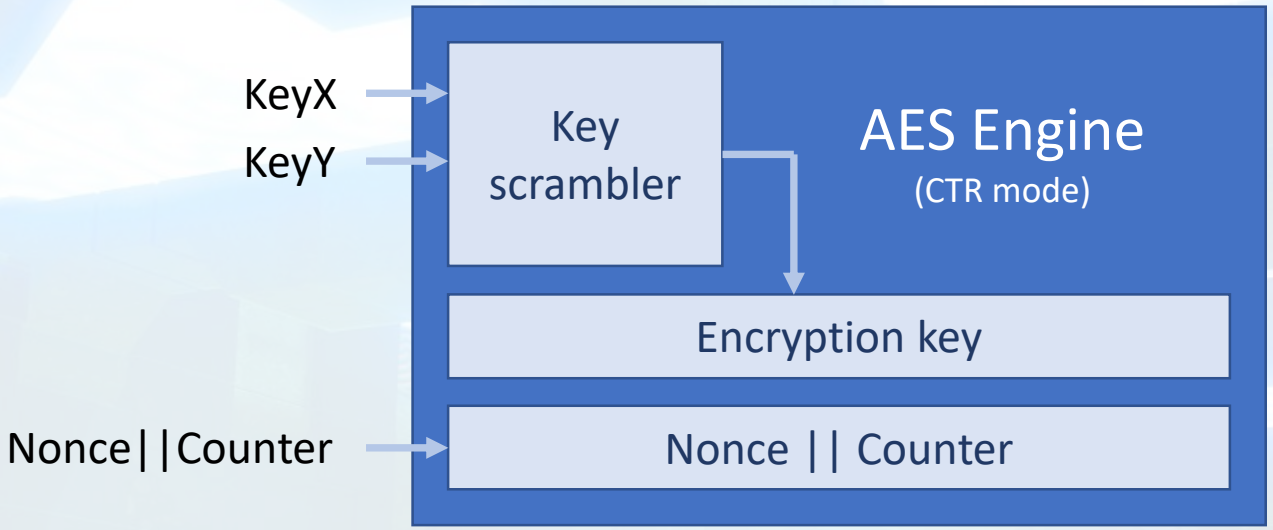

KeyX
KeyY
Key scrambler
AES Engine
(CTR mode)
Encryption key
Nonce||Counter
Nonce || Counter
encrypt(key, nonce ||counter + 0)
encrypt(key, nonce ||counter + 1)

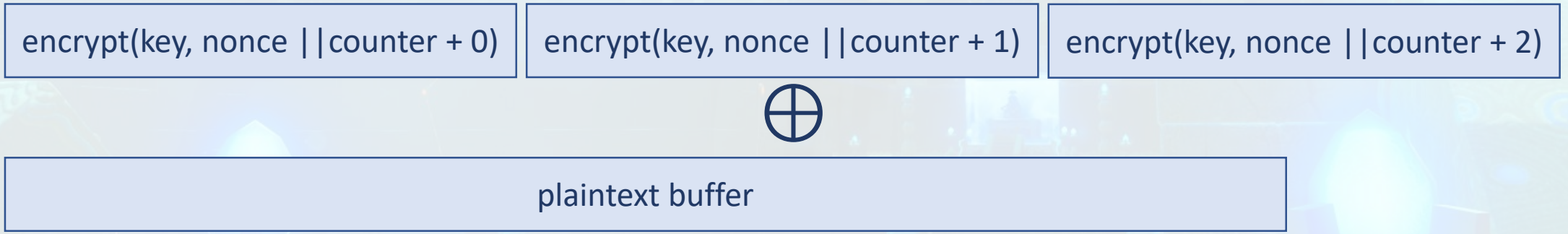

KeyX
KeyY
Key scrambler
AES Engine
(CTR mode)
Encryption key
Nonce||Counter
Nonce || Counter
encrypt(key, nonce ||counter + 0)
encrypt(key, nonce ||counter + 1)
encrypt(key, nonce ||counter + 2)

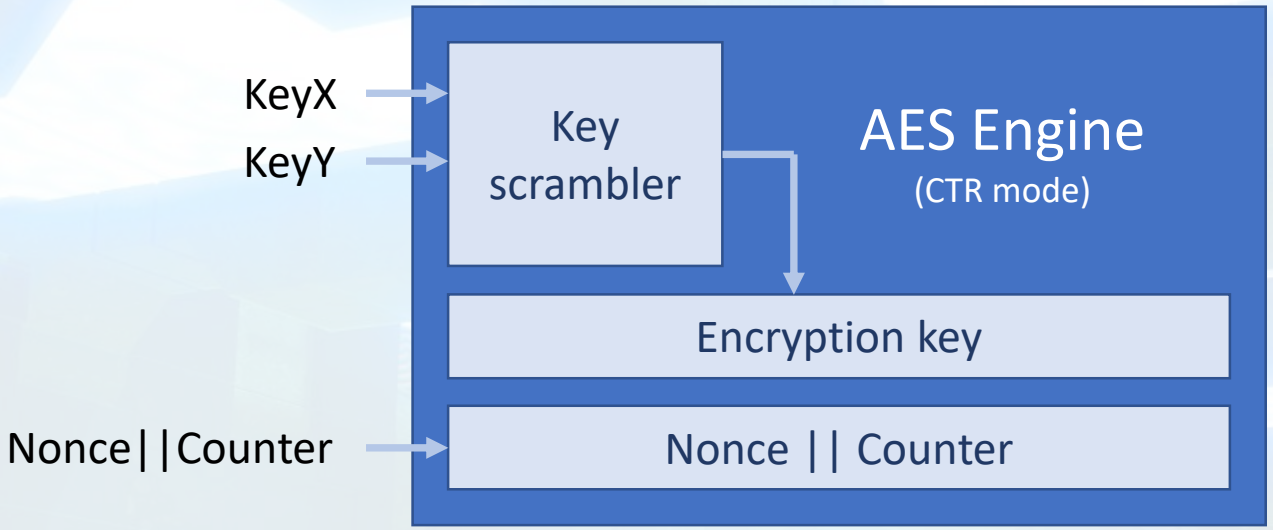


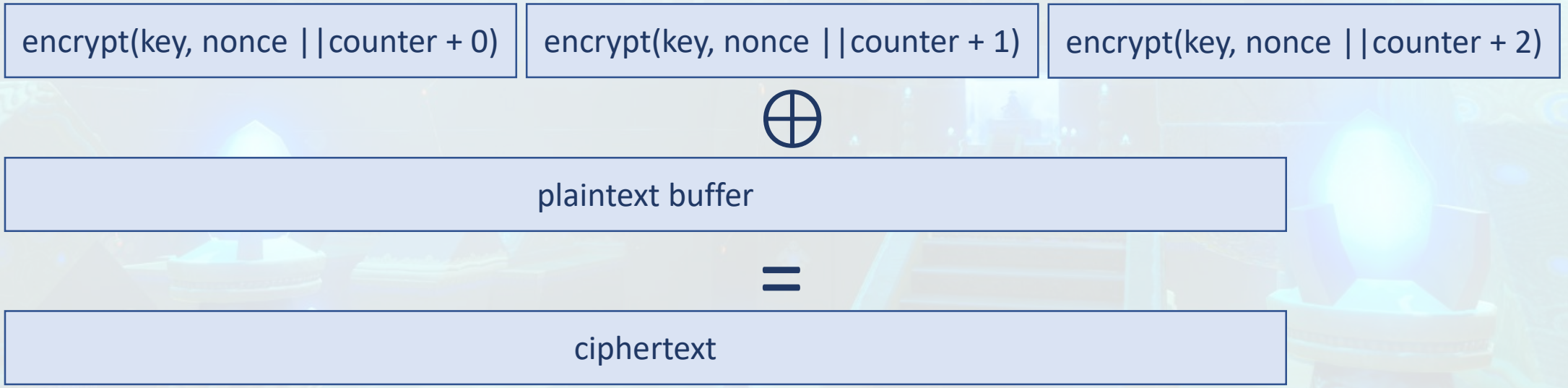
encrypt(key, nonce ||counter + 0) | encrypt(key, nonce ||counter + 1) | encrypt(key, nonce ||counter + 2)

plaintext buffer

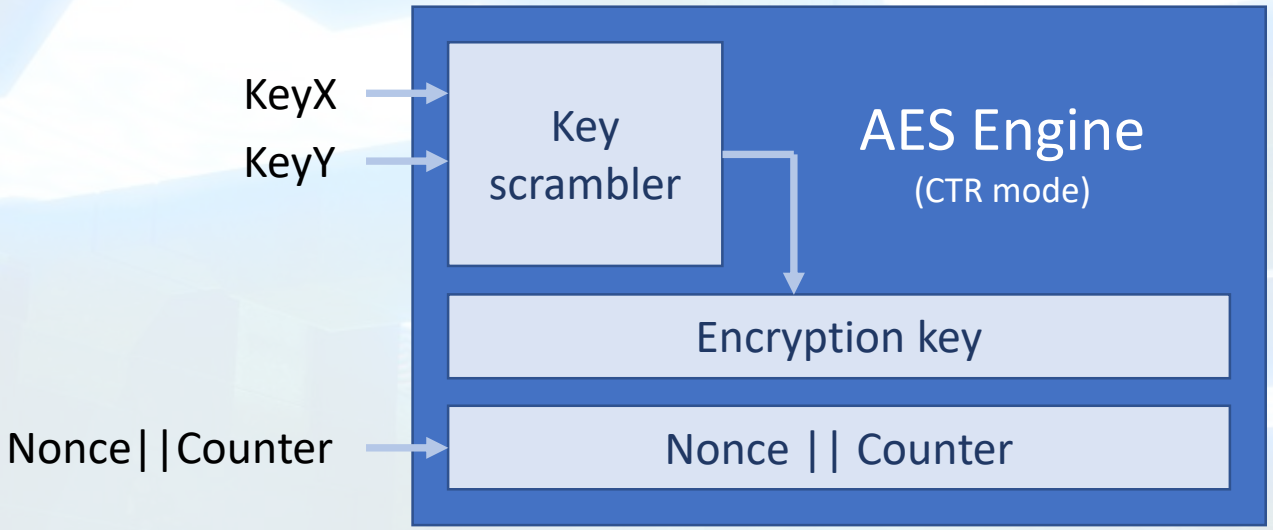

KeyX
KeyY
Key scrambler
AES Engine
(CTR mode)
Encryption key
Nonce||Counter
Nonce || Counter

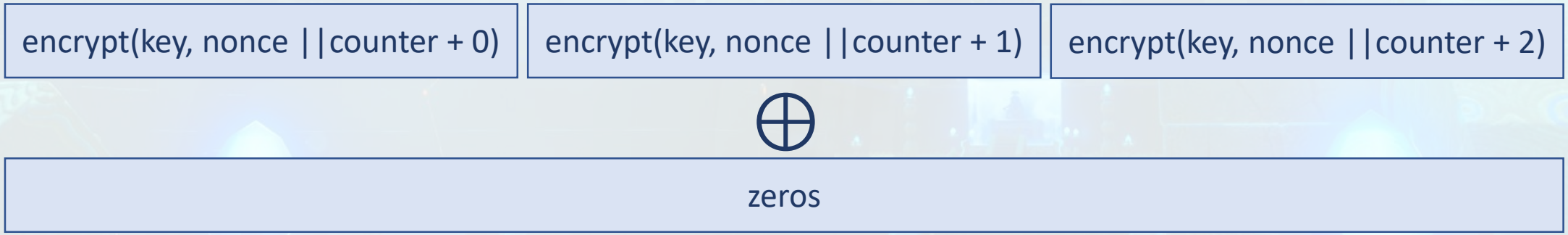

encrypt(key, nonce ||counter + 0)
encrypt(key, nonce ||counter + 1)
encrypt(key, nonce ||counter + 2)
plaintext buffer

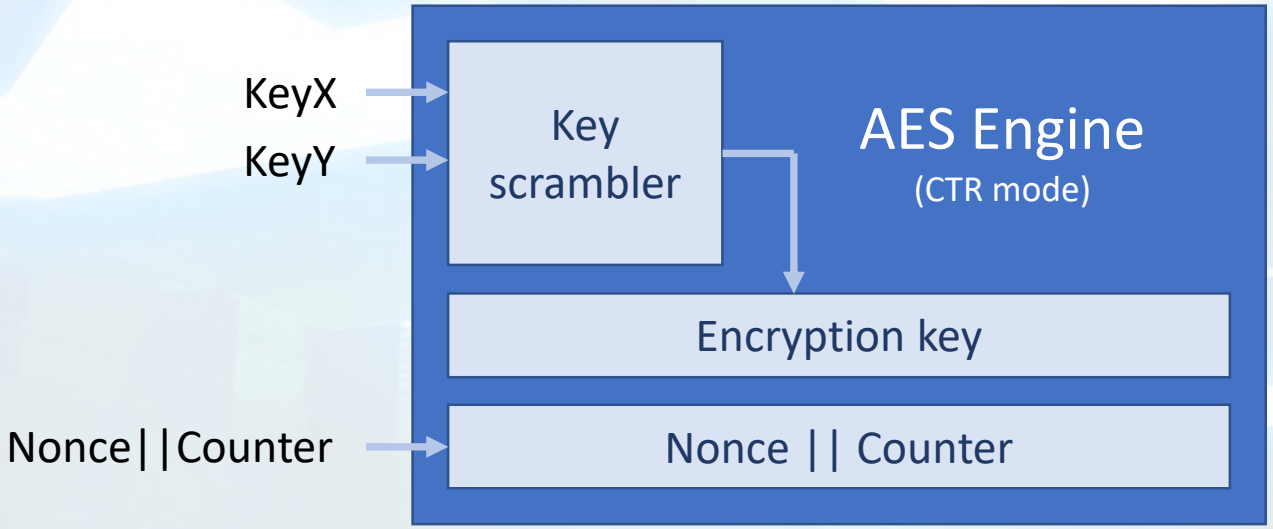

KeyX
KeyY
Key scrambler
AES Engine
(CTR mode)
Encryption key
Nonce||Counter
Nonce || Counter

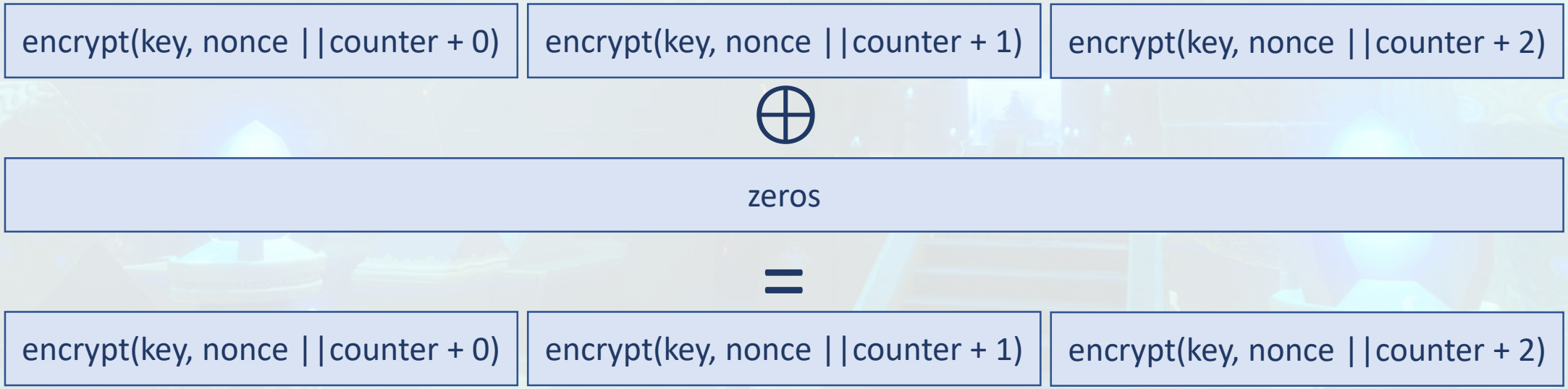

encrypt(key, nonce ||counter + 0)
encrypt(key, nonce ||counter + 1)
encrypt(key, nonce ||counter + 2)
⊕
plaintext buffer
=
ciphertext


A ⊕ 0 = A

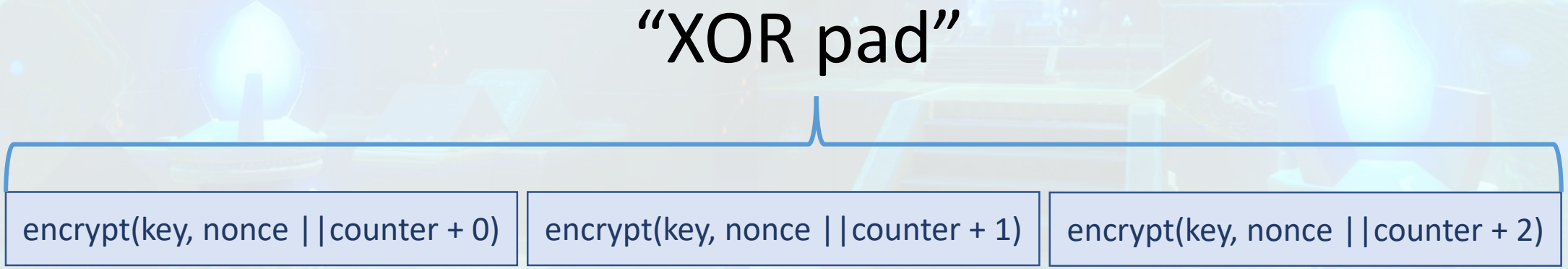

KeyX
KeyY
Key scrambler
AES Engine
(CTR mode)
Encryption key
Nonce||Counter
Nonce || Counter


encrypt(key, nonce ||counter + 0)
encrypt(key, nonce ||counter + 1)
encrypt(key, nonce ||counter + 2)
⊕
zeros


KeyX
KeyY
Key scrambler
AES Engine
(CTR mode)
Encryption key
Nonce||Counter
Nonce || Counter


encrypt(key, nonce ||counter + 0)
encrypt(key, nonce ||counter + 1)
encrypt(key, nonce ||counter + 2)
⊕
zeros
=
encrypt(key, nonce ||counter + 0)
encrypt(key, nonce ||counter + 1)
encrypt(key, nonce ||counter + 2)

- A XOR pad can be used to perform CTR mode encryption and decryption without access to the AES engine hardware
- Only way to do crypto externally until hardware AES engine's key scrambling algorithm was cracked



Crypto Caverns

- Wrote a Python script for trying out a list of different keys from the hardware keys dump, with the ability to adjust AES-CTR internals
- Try encrypting zeros with different keys, nonce, and counters based on NFC service dump values until the known good XOR pad result is found
- With the following setup, it works:
  - The encryption key is the composite NFC service key from AES chip
  - The nonce and initial counter values are from the NFC binary
  - The counter is in big endian representation during AES-CTR
- Mysterious value is 32 byte XOR pad using NFC-specific key

Crypto Caverns

The Amiibo Crypto system

Amiibo data has two partitions:

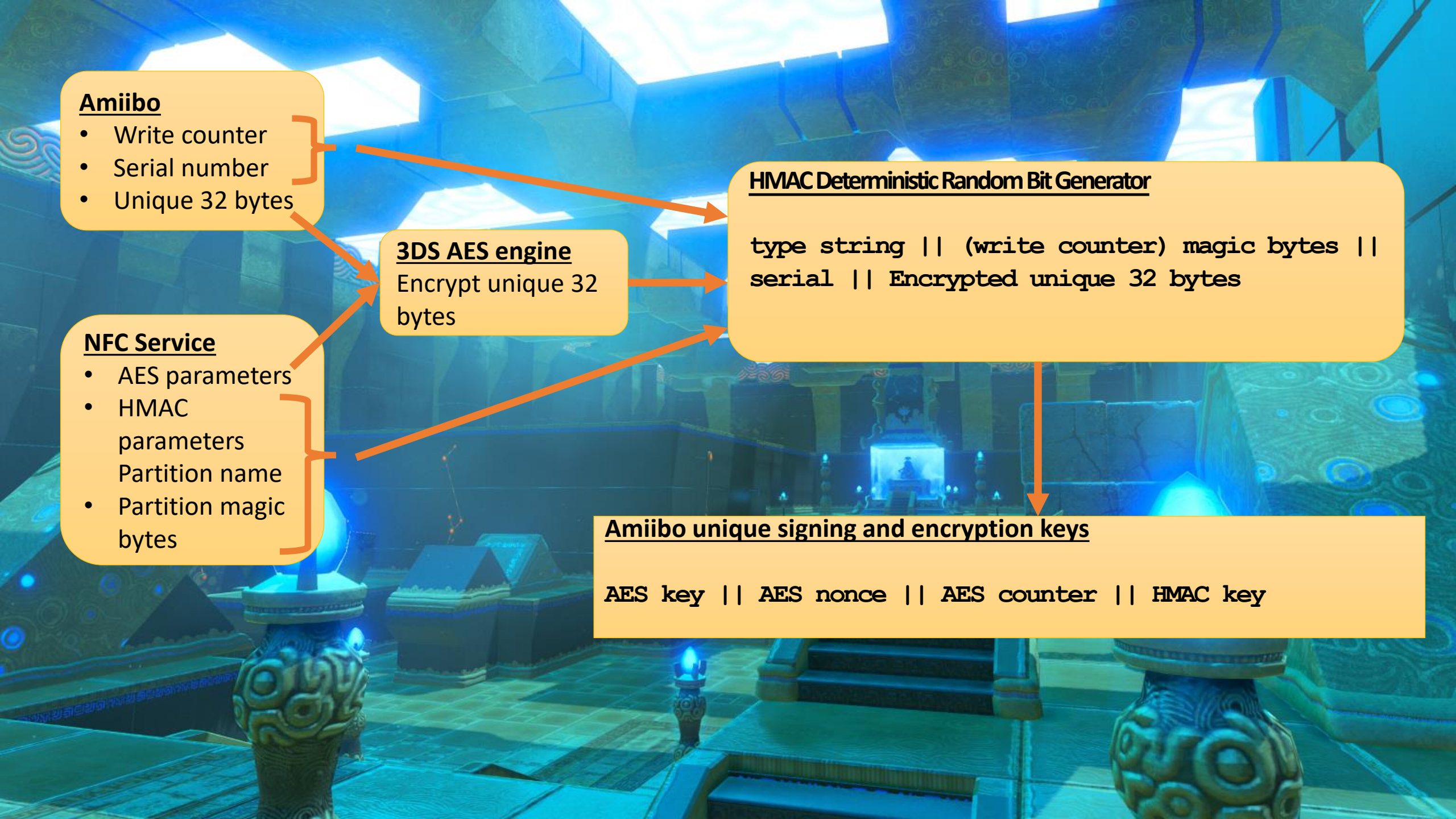
1. Tag (“locked secret”)
   - NTAG215 serial number
   - Character ID
   - Unique 32 byte sequence
2. Data (“unfixed infos”)
   - Settings, bit flags
   - Registered owner Mii data
   - Game/application data

Crypto Caverns

The Amiibo Crypto system

1. Generate two sets of AES-CTR parameters and HMAC keys with a Deterministic Random Bit Generator
   - One set per partition
2. Sign each partition with its generated HMAC key
3. Encrypt data partition with AES-CTR using its generated AES key, nonce, and counter
   - Tag is stored plaintext; ignore that set of AES values
4. Rearrange the buffer for storage in NFC tag EEPROM

Reverse the process to decrypt (rearrange buffer, generate keys, decrypt data partition, check HMACs)


Amiibo
• Write counter
• Serial number
• Unique 32 bytes
3DS AES engine
Encrypt unique 32 bytes
NFC Service
• AES parameters
• HMAC parameters Partition name
• Partition magic bytes
HMAC Deterministic Random Bit Generator
type string || (write counter) magic bytes || serial || Encrypted unique 32 bytes
Amiibo unique signing and encryption keys
AES key || AES nonce || AES counter || HMAC key

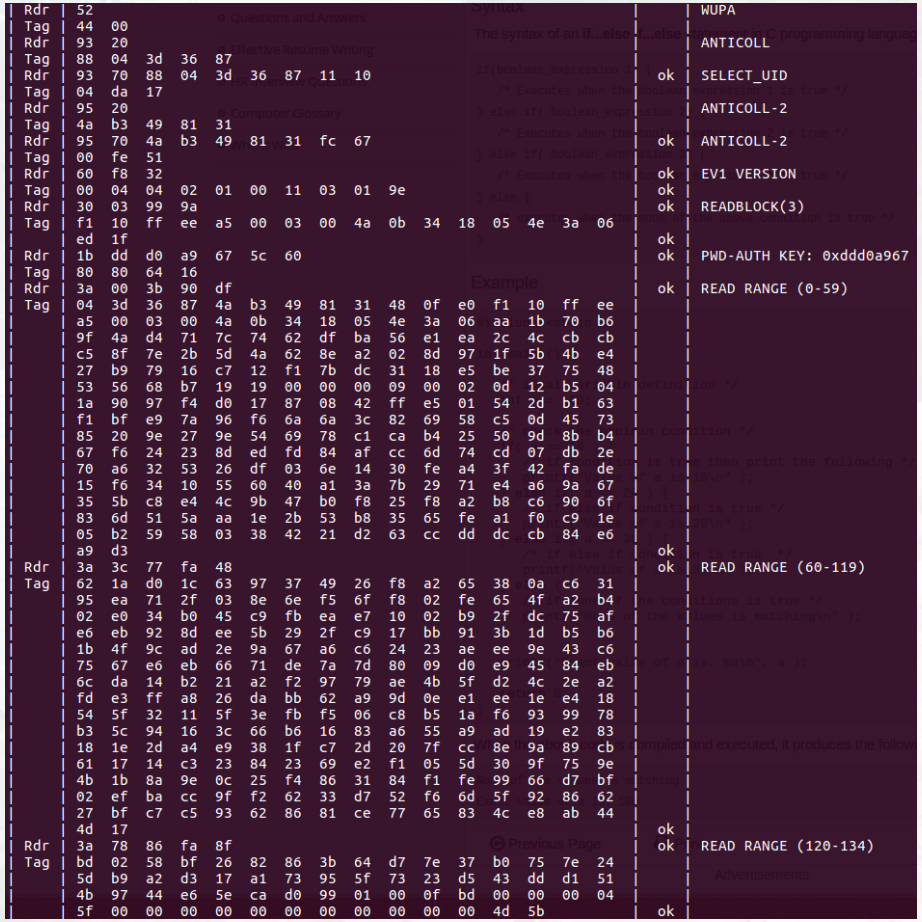
Crypto Caverns

You got the Amiibo crypto system!
It’s incredibly convoluted!

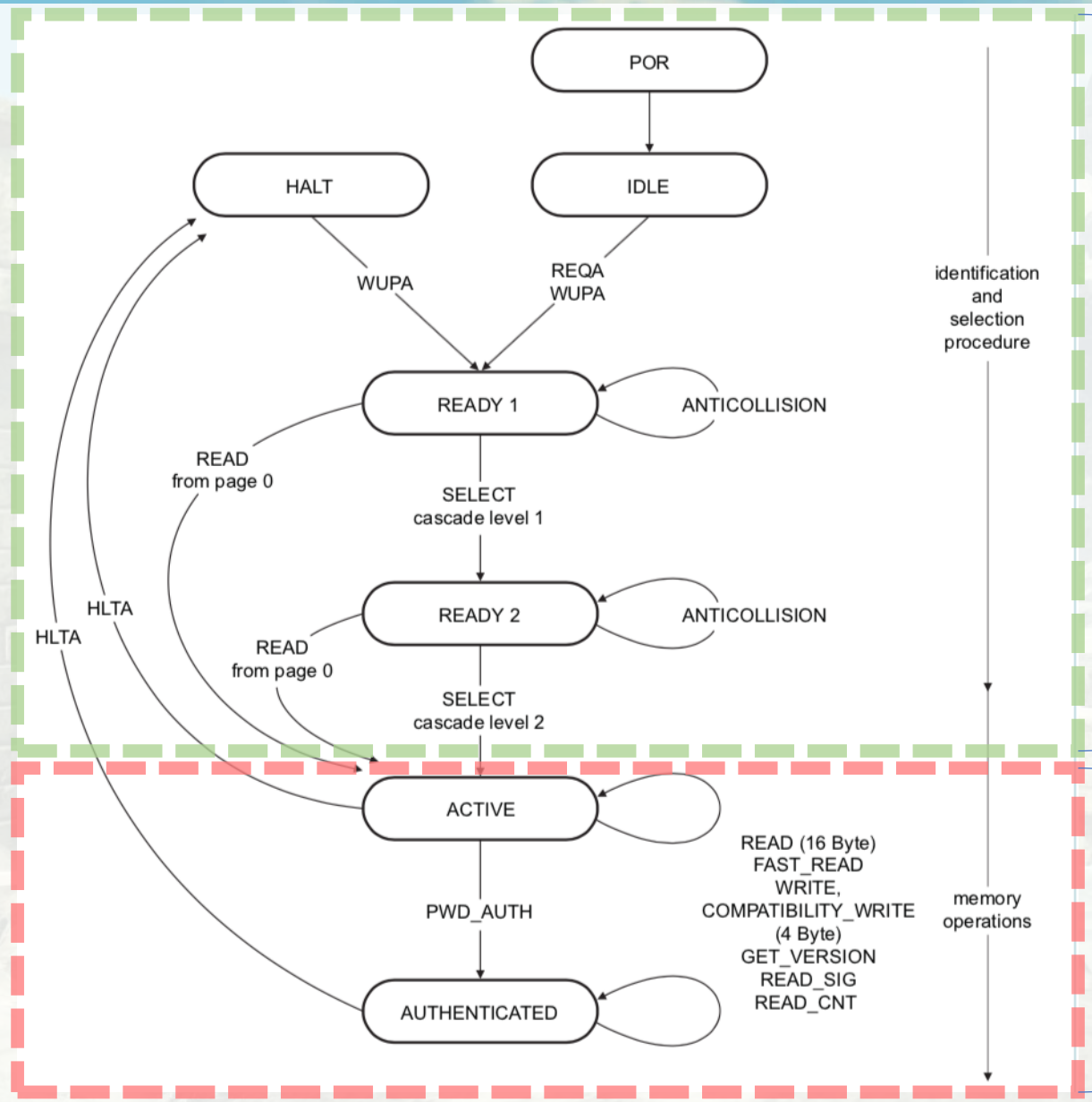

Proxmark Plateau

Proxmark Plateau

- Use the Proxmark sniffer and data sheets to figure out protocol
- NXP's NTAG21x data sheet is public
- Proxmark traffic display already recognizes many common ISO 14443 and NFC tag commands
  - Not always compatible with specific tag in use; may need changes



Rdr	52																WUPA
Tag	44	00															
Rdr	93	20															ANTICOLL
Tag	88	04	3d	36	87												
Rdr	93	70	88	04	3d	36	87	11	10							ok	SELECT_UID
Tag	04	da	17														
Rdr	95	20															ANTICOLL-2
Tag	4a	b3	49	81	31												
Rdr	95	70	4a	b3	49	81	31	fc	67							ok	ANTICOLL-2
Tag	00	fe	51														
Rdr	60	f8	32													ok	EV1 VERSION
Tag	00	04	04	02	01	00	11	03	01	9e						ok	
Rdr	30	03	99	9a												ok	READBLOCK(3)
Tag	f1	10	ff	ee	a5	00	03	00	4a	0b	34	18	05	4e	3a	06	
	ed	1f														ok	
Rdr	1b	dd	d0	a9	67	5c	60									ok	PWD-AUTH KEY: 0xddd0a967
Tag	80	80	64	16													
Rdr	3a	00	3b	90	df											ok	READ RANGE (0-59)
Tag	04	3d	36	87	4a	b3	49	81	31	48	0f	e0	f1	10	ff	ee	
	a5	00	03	00	4a	0b	34	18	05	4e	3a	06	aa	1b	70	b6	
	9f	4a	d4	71	7c	74	62	df	ba	56	e1	ea	2c	4c	cb	cb	
	c5	8f	7e	2b	5d	4a	62	8e	a2	02	8d	97	1f	5b	4b	e4	
	27	b9	79	16	c7	12	f1	7b	dc	31	18	e5	be	37	75	48	
	53	56	68	b7	19	19	00	00	00	09	00	02	0d	12	b5	04	
	1a	90	97	f4	d0	17	87	08	42	ff	e5	01	54	2d	b1	63	
	f1	bf	e9	7a	96	f6	6a	6a	3c	82	69	58	c5	0d	45	73	
	85	20	9e	27	9e	54	69	78	c1	ca	b4	25	50	9d	8b	b4	
	67	f6	24	23	8d	ed	fd	84	af	cc	6d	74	cd	07	db	2e	
	70	a6	32	53	26	df	03	6e	14	30	fe	a4	3f	42	fa	de	
	15	f6	34	10	55	60	40	a1	3a	7b	29	71	e4	a6	9a	67	
	35	5b	c8	e4	4c	9b	47	b0	f8	25	f8	a2	b8	c6	90	6f	
	83	6d	51	5a	aa	1e	2b	53	b8	35	65	fe	a1	f0	c9	1e	
	05	b2	59	58	03	38	42	21	d2	63	cc	dd	dc	cb	84	e6	
	a9	d3														ok	
Rdr	3a	3c	77	fa	48											ok	READ RANGE (60-119)
Tag	62	1a	d0	1c	63	97	37	49	26	f8	a2	65	38	0a	c6	31	
	95	ea	71	2f	03	8e	6e	f5	6f	f8	02	fe	65	4f	a2	b4	
	02	e0	34	b0	45	c9	fb	ea	e7	10	02	b9	2f	dc	75	af	
	e6	eb	92	8d	ee	5b	29	2f	c9	17	bb	91	3b	1d	b5	b6	
	1b	4f	9c	ad	2e	9a	67	a6	c6	24	23	ae	ee	9e	43	c6	
	75	67	e6	eb	66	71	de	7a	7d	80	09	d0	e9	45	84	eb	
	6c	da	14	b2	21	a2	f2	97	79	ae	4b	5f	d2	4c	2e	a2	
	fd	e3	ff	a8	26	da	bb	62	a9	9d	0e	e1	ee	1e	e4	18	
	54	5f	32	11	5f	3e	fb	f5	06	c8	b5	1a	f6	93	99	78	
	b3	5c	94	16	3c	66	b6	16	83	a6	55	a9	ad	19	e2	83	
	18	1e	2d	a4	e9	38	1f	c7	2d	20	7f	cc	8e	9a	89	cb	
	61	17	14	c3	23	84	23	69	e2	f1	05	5d	30	9f	75	9e	
	4b	1b	8a	9e	0c	25	f4	86	31	84	f1	fe	99	66	d7	bf	
	02	ef	ba	cc	9f	f2	62	33	d7	52	f6	6d	5f	92	86	62	
	27	bf	c7	c5	93	62	86	81	ce	77	65	83	4c	e8	ab	44	
	4d	17														ok	
Rdr	3a	78	86	fa	8f											ok	READ RANGE (120-134)
Tag	bd	02	58	bf	26	82	86	3b	64	d7	7e	37	b0	75	7e	24	
	5d	b9	a2	d3	17	a1	73	95	5f	73	23	d5	43	dd	d1	51	
	4b	97	44	e6	5e	ca	d0	99	01	00	0f	bd	00	00	00	04	
	5f	00	00	00	00	00	00	00	00	00	00	00	4d	5b		ok	

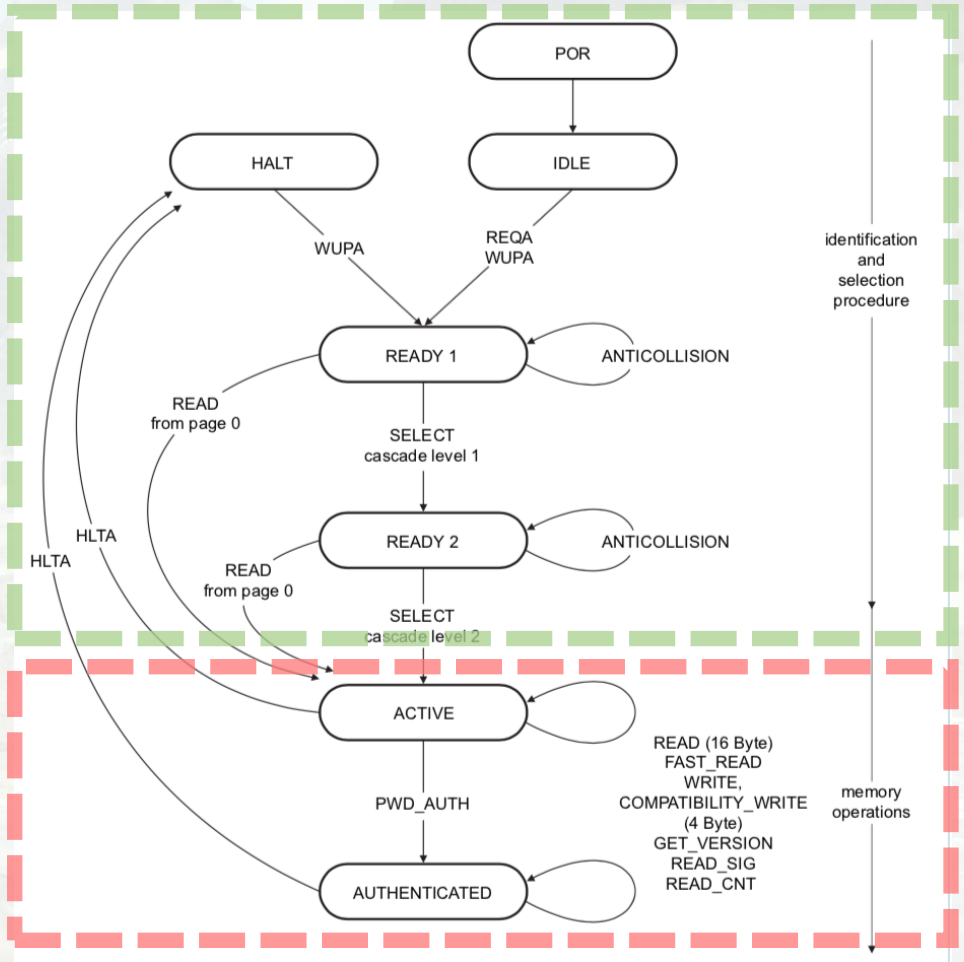


Implemented in ISO14443a simulator

Specific to NTAG21x

Proxmark Plateau

- Commands to implement:
  - READ
  - FAST_READ
  - WRITE
  - GET_VERSION
  - PWD_AUTH
  - READ_SIG
- Buffer for tag EEPROM contents



Proxmark Plateau

- State of the codebase: spaghetti
- A lot of the work is just figuring out how everything is set up
  - There’s no developer documentation
- ~~Modular, self-contained NTAG21x logic~~
- Weld NTAG215 simulator on to ISO 14443a simulator

```
| Rdr | 52                                                                  |      | WUPA
| Tag | 44  00                                                              |      |
| Rdr | 93  20                                                              |      | ANTICOLL
| Tag | 88  04  3d  36  87                                                  |      |
| Rdr | 93  70  88  04  3d  36  87  11  10                                  | ok   | SELECT_UID
| Tag | 04  da  17                                                          |      |
| Rdr | 95  20                                                              |      | ANTICOLL-2
| Tag | 4a  b3  49  81  31                                                  |      |
| Rdr | 95  70  4a  b3  49  81  31  fc  67                                  | ok   | ANTICOLL-2
| Tag | 00  fe  51                                                          |      |
| Rdr | 60  f8  32                                                          | ok   | EV1 VERSION
| Tag | 00  04  04  02  01  00  11  03  01  9e                              | ok   |
| Rdr | 30  03  99  9a                                                      | ok   | READBLOCK(3)
| Tag | f1  10  ff  ee  a5  00  03  00  4a  0b  34  18  05  4e  3a  06      |      |
|     | ed  1f                                                              | ok   |
| Rdr | 1b  dd  d0  a9  67  5c  60                                          | ok   | PWD-AUTH KEY: 0xddd0a967
| Tag | 80  80  64  16                                                      |      |
| Rdr | 3a  00  3b  90  df                                                  | ok   | READ RANGE (0-59)
| Tag | 04  3d  36  87  4a  b3  49  81  31  48  0f  e0  f1  10  ff  ee      |      |
|     | a5  00  03  00  4a  0b  34  18  05  4e  3a  06  aa  1b  70  b6      |      |
|     | 9f  4a  d4  71  7c  74  62  df  ba  56  e1  ea  2c  4c  cb  cb      |      |
|     | c5  8f  7e  2b  5d  4a  62  8e  a2  02  8d  97  1f  5b  4b  e4      |      |
|     | 27  b9  79  16  c7  12  f1  7b  dc  31  18  e5  be  37  75  48      |      |
|     | 53  56  68  b7  19  19  00  00  00  09  00  02  0d  12  b5  04      |      |
|     | 1a  90  97  f4  d0  17  87  08  42  ff  e5  01  54  2d  b1  63      |      |
|     | f1  bf  e9  7a  96  f6  6a  6a  3c  82  69  58  c5  0d  45  73      |      |
|     | 85  20  9e  27  9e  54  69  78  c1  ca  b4  25  50  9d  8b  b4      |      |
|     | 67  f6  24  23  8d  ed  fd  84  af  cc  6d  74  cd  07  db  2e      |      |
|     | 70  a6  32  53  26  df  03  6e  14  30  fe  a4  3f  42  fa  de      |      |
|     | 15  f6  34  10  55  60  40  a1  3a  7b  29  71  e4  a6  9a  67      |      |
|     | 35  5b  c8  e4  4c  9b  47  b0  f8  25  f8  a2  b8  c6  90  6f      |      |
|     | 83  6d  51  5a  aa  1e  2b  53  b8  35  65  fe  a1  f0  c9  1e      |      |
|     | 05  b2  59  58  03  38  42  21  d2  63  cc  dd  dc  cb  84  e6      |      |
|     | a9  d3                                                              | ok   |
```

Proxmark Plateau

GET_VERSION

```
| Rdr | 60  f8  32                                  | ok | EV1 VERSION
| Tag | 00  04  04  02  01  00  11  03  01  9e      | ok |
```

- “The GET_VERSION command has no arguments and replies the version information for the specific NTAG21x type.”
- NTAG215 info: `00 04 04 02 01 00 11 03` (+CRC-16)
- Add pre-cached response with 16 bit CRC value
  - Cyclic Redundancy Check: used for detecting transmission errors
  - Pre-caching necessary for low latency commands
  - Modulation must be calculated for data bytes before the timeout (usually 5ms)

Proxmark Plateau

READ

```
| Rdr | 30  03  99  9a                                          |  ok | READBLOCK(3)
| Tag | f1  10  ff  ee  a5  00  03  00  4a  0b  34  18  05  4e  3a  06  |     |
|     | ed  1f                                                  |  ok |
```

- Read four pages, beginning at specified page
  - A page is a four byte block of data. 16 bytes are returned.
  - An NTAG215 has 135 pages (540 bytes total)
- Only used to read the capability container in page 3, so we don't need to implement everything (locked pages, rolling over after end of memory, etc.)
- Biggest hurdle is implementing a memory buffer to store the EEPROM data
- Get buffer dynamically, calculate CRC-16, and send it back

Proxmark Plateau

Tag EEPROM Buffer

- The Proxmark has a “big buffer” with custom malloc
- Undocumented card memory section that can be preserved while freeing the rest of the memory
- Can be populated from client using USB commands
- Need to set up interface in Lua to handle reading/writing to card memory buffer
- Update simulator to use card memory buffer when simulating NTAG215

Proxmark Plateau

FAST_READ

Rdr	3a 00 3b 90 df	ok	READ RANGE (0-59)
Tag	04 3d 36 87 4a b3 49 81 31 48 0f e0 f1 10 ff ee		
	a5 00 03 00 4a 0b 34 18 05 4e 3a 06 aa 1b 70 b6		

- Read from given start page to end page
- Reading all pages at once is allowed, but the 3DS only reads up to 60 pages at a time
  - Uses three separate FAST_READ commands to get all memory
- Should send a NAK if request is out of range, but the consoles never do this
  - Still implement some bounds checks to prevent crashes
  - Poor signal results in noisy, corrupted commands
- Get buffer, calculate CRC-16, send it back

Proxmark Plateau

WRITE

- Write to a single page
- Update 4 bytes in the card memory buffer and send ACK
- Could potentially maintain state to see whether password unlock was used, but it’s not necessary at the moment

Proxmark Plateau

PWD_AUTH

```
| Rdr | 1b  dd  d0  a9  67  5c  60          |  ok | PWD-AUTH KEY: 0xddd0a967
| Tag | 80  80  64  16                      |     |
```

- 32 bit password for unlocking write capability
  - One of NTAG21x's built-in security features
- After 7 failed attempts, pages will be permanently locked
- Respond with `80 80` (+CRC-16) to accept password
- We can simply accept any password for the simulator
  - But it would still be interesting to know how passwords are calculated for more accurate simulation, or for writing to a real Amiibo

Proxmark Plateau

PWD_AUTH

- With SDR you can supply an arbitrary serial number and get back the password when the console attempts to authenticate
- Serials `00000000000000` and `04000000000000` give back `AA55AA55`...

Serial / UID	Password
04 00 00 00 00 00 00	AA 55 AA 55
04 AA 00 00 00 00 00	00 55 AA 55
04 AA 55 00 00 00 00	00 00 AA 55
04 AA 55 AA 00 00 00	AA 00 00 55

- It's based on XORing serial number bytes
  - $A \oplus 0 = A$; $A \oplus A = 0$

Proxmark Plateau

PWD_AUTH

Serial/UID	Password
04 FF 00 00 00 00 00	55 55 AA 55
04 00 FF 00 00 00 00	AA AA AA 55
04 00 00 FF 00 00 00	55 55 55 55
04 00 00 00 FF 00 00	AA AA AA AA
04 00 00 00 00 FF 00	AA 55 55 55
04 00 00 00 00 00 FF	AA 55 AA AA

Proxmark Plateau

PWD_AUTH

Serial/UID	Password (XOR AA55AA55)
04 FF 00 00 00 00 00	FF 00 00 00
04 00 FF 00 00 00 00	00 FF 00 00
04 00 00 FF 00 00 00	FF 00 FF 00
04 00 00 00 FF 00 00	00 FF 00 FF
04 00 00 00 00 FF 00	00 00 FF 00
04 00 00 00 00 00 FF	00 00 00 FF

Proxmark Plateau

PWD_AUTH

Serial/UID	Password (XOR AA55AA55)
04 00 00 FF 00 00 00	FF 00 FF 00
04 FF 00 FF 00 00 00	00 00 FF 00
04 FF 00 FF 00 FF 00	00 00 00 00

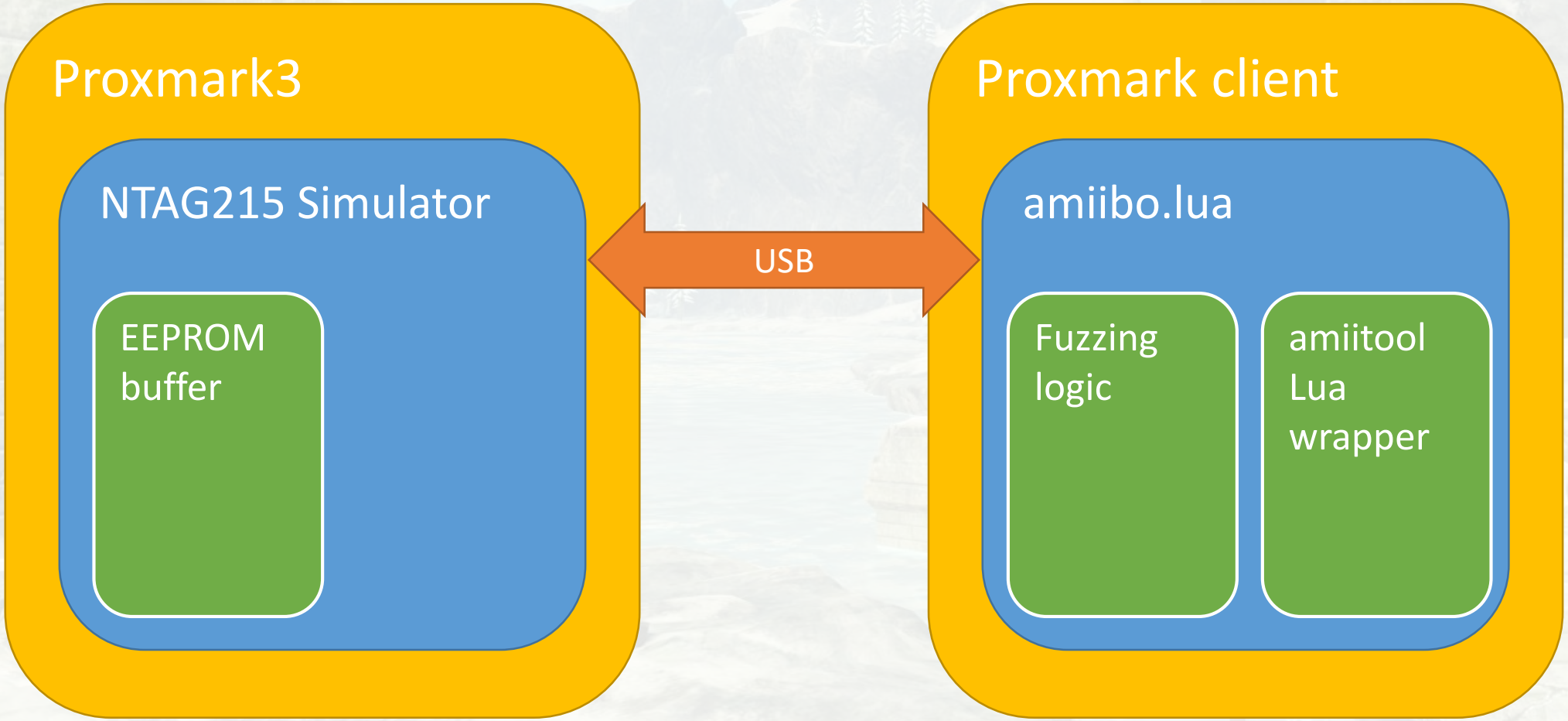
```
xormask = '\xaa\x55\xaa\x55'
for i = 0 to 3:
    pwd[i] = xor(uid[i+1], uid[i+3], xormask[i])
```

Proxmark Plateau

Amiibo integration

- Proxmark client has a Lua script interface
  - Can write and update Lua programs without recompiling client
  - Easier to implement, has OOP-ish features
- Wrote Lua wrapper for amiitool and compiled as a library
- Amiibo object in Lua
  - Uses amiitool Lua module for packing and unpacking data
- Card memory interface for populating and reading card memory
- Features
  - Dump and unpack data from Amiibo
  - Load and simulate image
  - Save/restore state of card memory

Proxmark Plateau

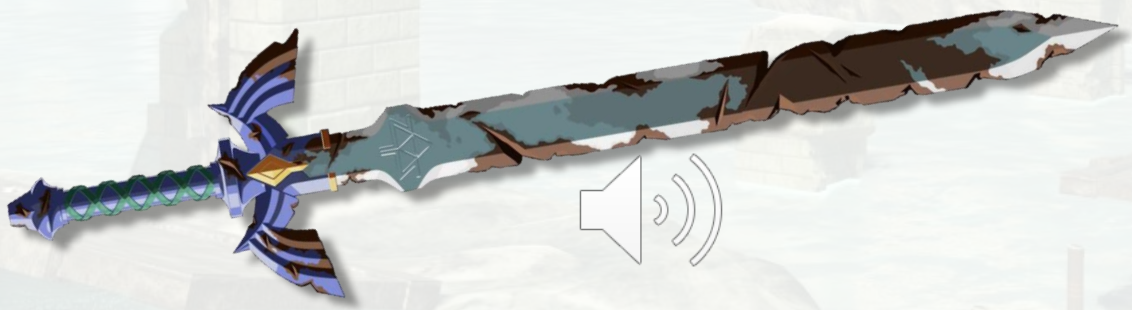


Proxmark Plateau

You got the NTAG215 simulator!
Why does it smell like spaghetti?

Proxmark Plateau

- Works fine on the 3DS, but when the Switch finally comes out in March:
  - ECDSA signature of the tag's serial number is actually checked with the READ_SIG command
    - The Wii U also uses this check, but I hadn't been testing on it
  - The ID card shaped Proxmark HF antenna works very poorly with the Switch

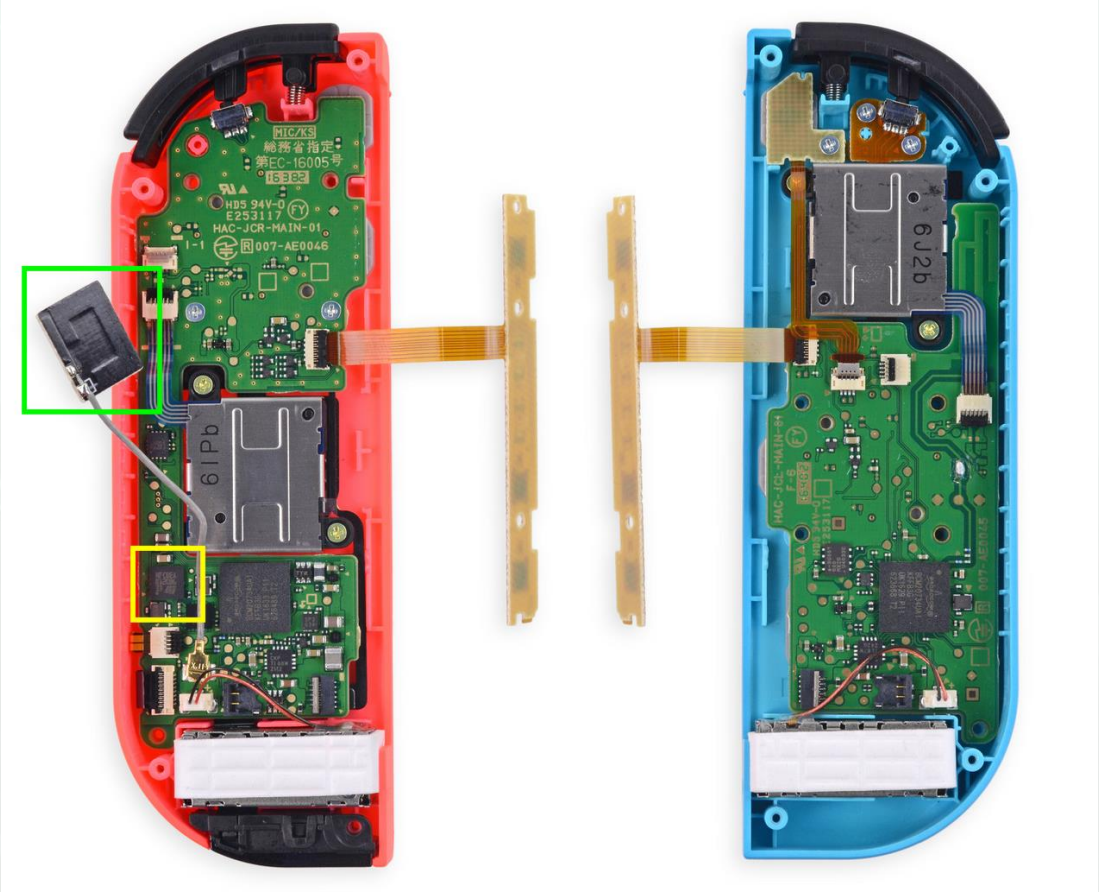


- It's not very effective.


Antenna Problems

Antenna Problems

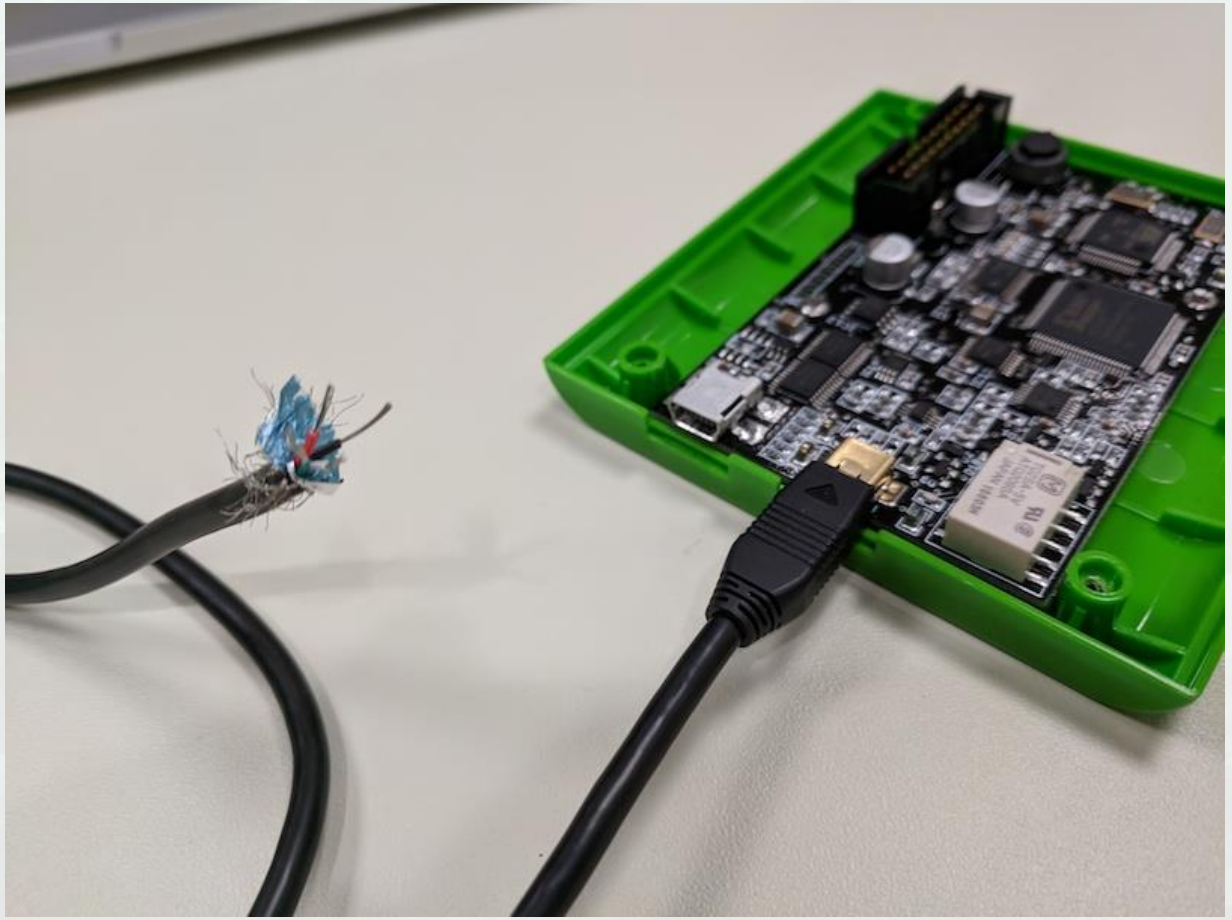
Antenna Problems



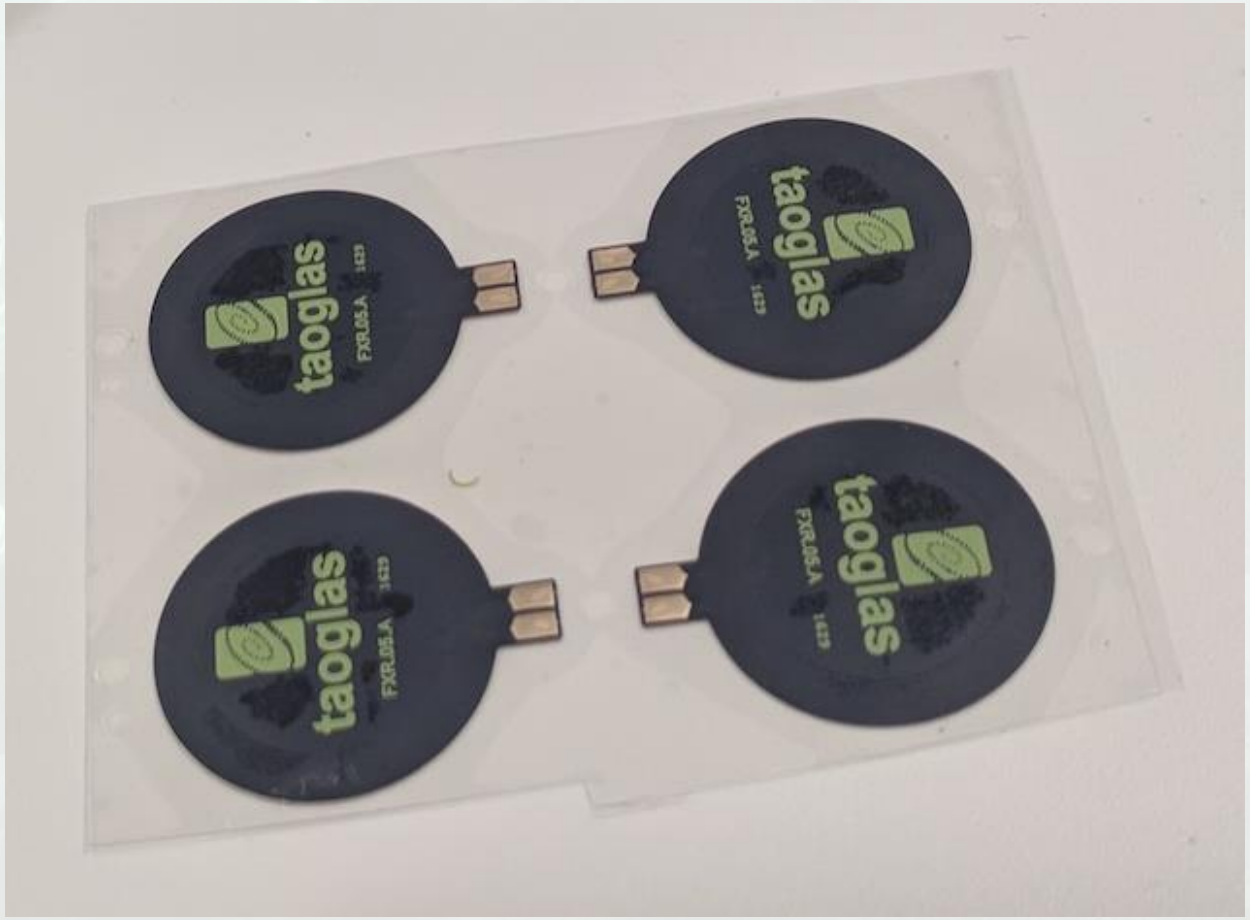
Antenna Problems

Official Proxmark3 instructions: rip open cable and wrap it around until it works

Making an Antenna



Making an Antenna



Making an Antenna

- Fail: Signal is not strong enough to read tags or be picked up by console.
  - Got 0.1 V, need around 12V.
- Can’t just buy any “13.56 MHz antenna” and attach it: tuning must factor in the rest of the device
  - Antenna forms inductor-capacitor circuit with capacitor on the Proxmark3 board
  - Capacitor has a capacitance of 47 picofarads (pF). Need to adjust this to match the inductance of a different antenna.

Making an Antenna

taoglas
antenna solutions

2. Specifications

Flexible PCB Near-Field Communications Antenna		
Frequency	13.56	MHz
Inductance @ 13.56 MHz	15.9	µH

Mechanical	
Antenna Dimensions	Diameter: 26.4 mm; Thickness: 0.24 mm
RoHS Compliant	Yes
Adhesive	3M 467
Weight	201.7mg

Making an Antenna

- Current resonant frequency: $\frac{1}{2\pi\sqrt{15.9\ \mu H\ \times\ 47\ pF}} = 5.822\ MHz$
- Desired resonant frequency: 13.56 MHz
- Solving for capacitance with desired frequency: need 8.67 pF.
- Need to reduce capacitance without changing Proxmark board.

Making an Antenna

- Reduce capacitance by adding capacitors in series:

$$\frac{1}{\frac{1}{47pF}+\frac{1}{C_2}+\frac{1}{C_3}+\cdots+\frac{1}{C_n}} = 8.67pF$$

- Ideal: add a 10.6 pF capacitor. Hard to order one that specific though.
- Alternative: get closest approximation with a series of whole-number capacitance levels available in small quantities online

Making an Antenna

```
[ jchambers@LT8220 ~/Documents/amiibo ]
$ ./caps.py 8.67 --included 47 --allowed 9 10 11 12 15 16 18 20 22 24 27 30 33 36 39
Closest set of 1 caps gives 8.913793pF
[47.0, 11.0]
Closest set of 2 caps gives 8.641471pF
[47.0, 36.0, 15.0]
Closest set of 3 caps gives 8.679351pF
[47.0, 33.0, 33.0, 30.0]
Closest set of 4 caps gives 8.074890pF
[47.0, 39.0, 39.0, 39.0, 39.0]
Closest set of 5 caps gives 6.689781pF
[47.0, 39.0, 39.0, 39.0, 39.0, 39.0]
Closest set of up to 5 caps gives 8.679351pF
[47.0, 33.0, 33.0, 30.0]
[ jchambers@LT8220 ~/Documents/amiibo ]
$
```

Making an Antenna

- 13.9 MHz resonant frequency with spare 10pF capacitor
- Boosts strength from 0.1 to 7V @ 13.56 MHz

Making an Antenna

```
proxmark3> script run amiibo
--- Executing: amiibo.lua, args ''
Loaded retail keys from all_in_one_keys.bin
Tag type:       NXP MIFARE Plus 2k
Tag UID:        043D36874AB349813148
Tag len:        540
Figure ID:      19190000000900020d
Settings init:  true
Nickname:       Peekandpwn
Appdata writes: 60
UID:    043d364ab34981
Write key:      ddd0a967

-----Finished
proxmark3> hw tune

Measuring antenna characteristics, please wait....#db# DownloadFPGA(len: 42096)
......
# LF antenna:  0.00 V @   125.00 kHz
# LF antenna:  0.00 V @   134.00 kHz
# LF optimal:  0.00 V @ 12000.00 kHz
# HF antenna:  6.91 V @    13.56 MHz
# Your LF antenna is unusable.
# Your HF antenna is marginal.
proxmark3>
```


Originality Check

Originality Check

1.3 Security

- Manufacturer programmed 7-byte UID for each device
- Pre-programmed Capability container with one time programmable bits
- Field programmable read-only locking function
- ECC based originality signature
- 32-bit password protection to prevent unauthorized memory operations

Originality Check

- ECDSA signature of tag's serial number
  - Elliptic Curve Digital Signature Algorithm
- Signature proves tag originates from NXP Semiconductors
- Prevents simple counterfeiting
  - Can't make signature for new serial without NXP private key
- Ideal benefit: A valid Amiibo NFC tag must be produced by NXP and initialized by Nintendo. Amiibo data is bound and unique to the tag it's created for.
  - No one else can produce or duplicate Amiibo.

Originality Check

- Can’t use an arbitrary serial number without the ECDSA signature created with NXP’s private key
- Implications for simulating, fuzzing:
  - Some games limit usage by serial number and time (Breath of the Wild)
  - Need to amass a collection of serial/signature pairs…
  - or rewrite with the same serial number and redo encryption each time
  - Can’t simulate with arbitrary serial number ☹
- … or can you?

Originality Check

How do cheat devices do it?

- PowerSaves offers “serial randomization” cheat for bypassing rate limit in Breath of the Wild
  - Can only scan an individual Amiibo once per in-game day
- Cheat devices *could* harvest serial/signature pairs from users’ Amiibo…
  - *Haven’t verified this, it’s just a possibility*
  - Amiibo image uploaded to API for each operation

Originality Check

Crypto implementation errors?

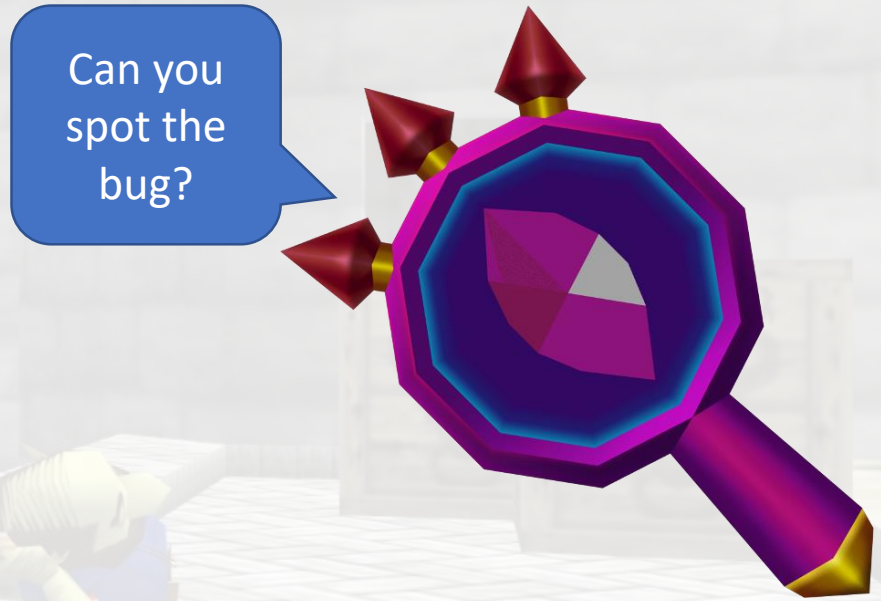
- Reusing the same “random number” in ECDSA will compromise the private key
  - See fail0verflow’s PlayStation 3 hacking talk
- Used example SDK code to parse out the nonce parameter from signature
- They don’t appear to reuse the nonce
  - in the small sample I checked

Originality Check

Protocol or application logic flaws?

Verification process:

1. Select tag by ID (anti-collision)
2. Read NXP tag signature
3. Validate selected ID with signature
4. Read entire image from EEPROM
5. Generate keys for image
6. Perform HMAC validation of image

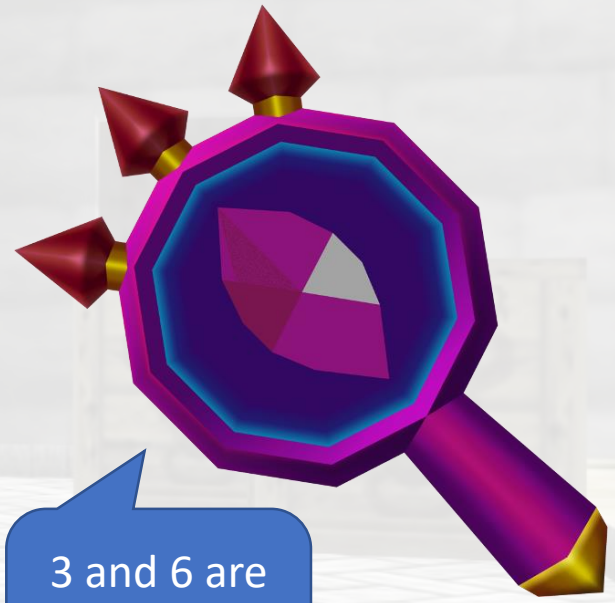


Originality Check

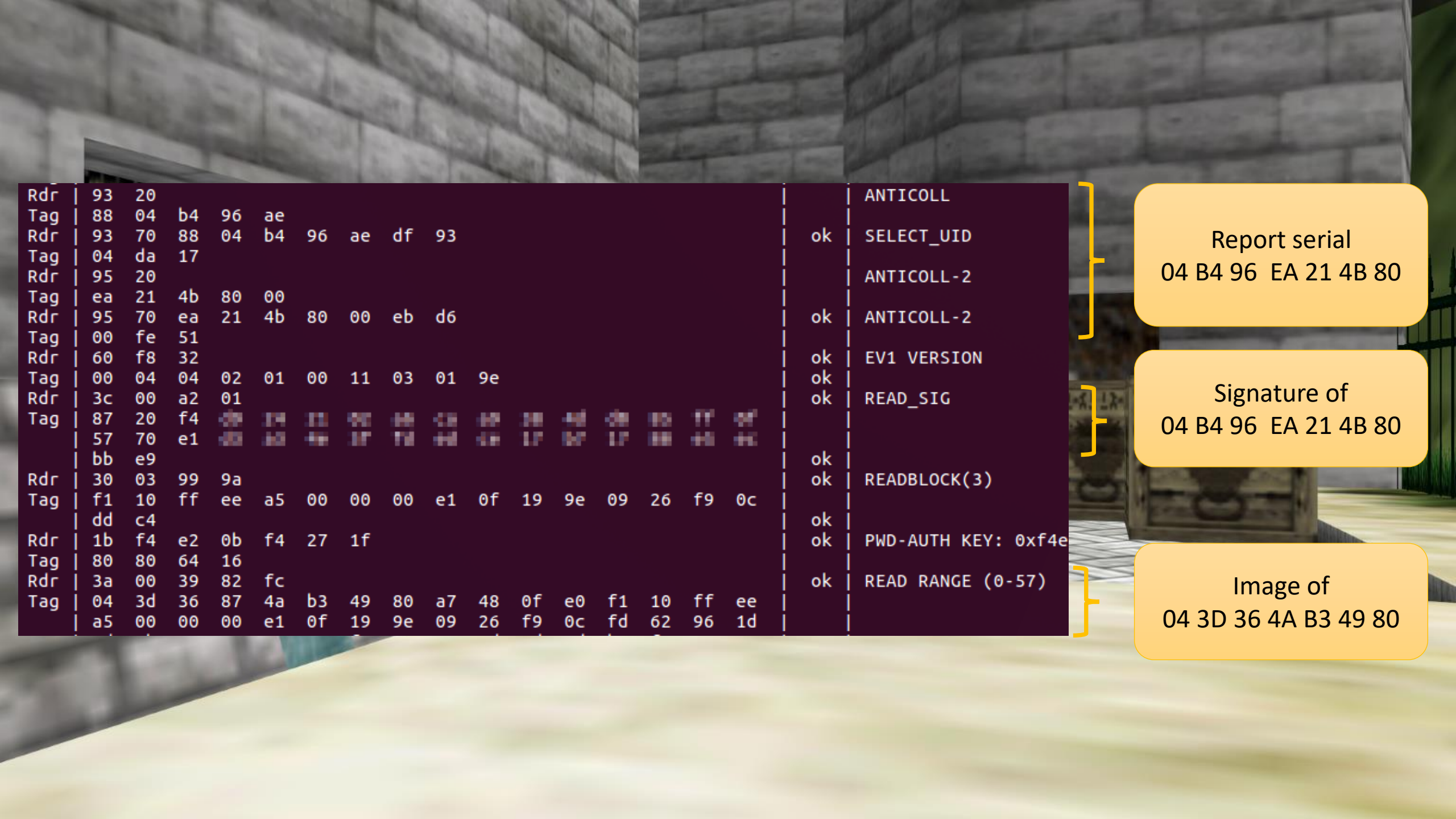
Protocol or application logic flaws?

Verification process:

1. Select tag by ID (anti-collision)
2. Read NXP tag signature
3. Validate selected ID with signature
4. Read entire image from EEPROM
5. Generate keys for image
6. Perform HMAC validation of image

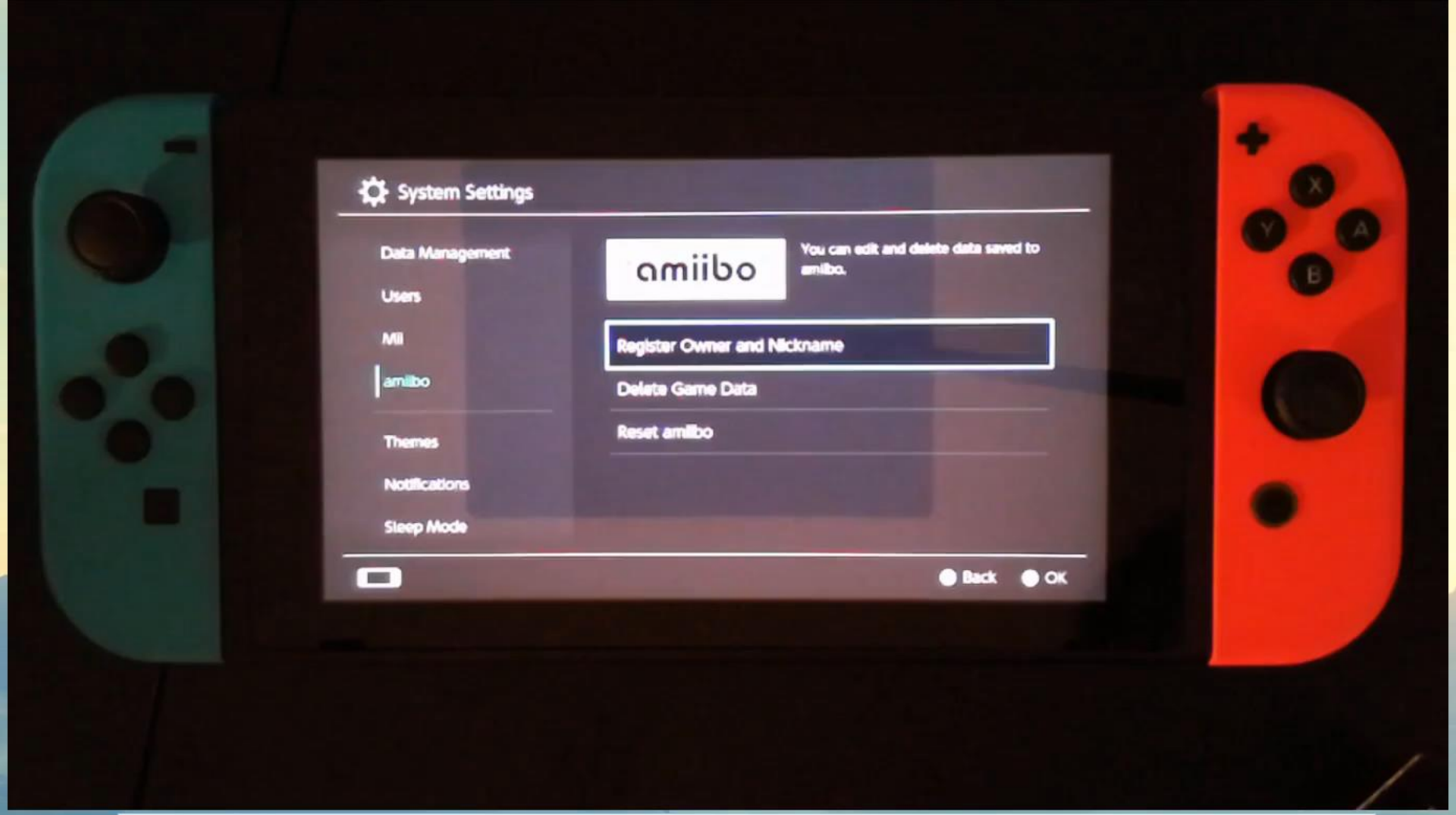


3 and 6 are not correlated


Rdr | 93 20 | | ANTICOLL
Tag | 88 04 b4 96 ae | |
Rdr | 93 70 88 04 b4 96 ae df 93 | ok | SELECT_UID
Tag | 04 da 17 | |
Rdr | 95 20 | | ANTICOLL-2
Tag | ea 21 4b 80 00 | |
Rdr | 95 70 ea 21 4b 80 00 eb d6 | ok | ANTICOLL-2
Tag | 00 fe 51 | |
Rdr | 60 f8 32 | ok | EV1 VERSION
Tag | 00 04 04 02 01 00 11 03 01 9e | ok |
Rdr | 3c 00 a2 01 | ok | READ_SIG
Tag | 87 20 f4 | |
| 57 70 e1 | |
| bb e9 | ok |
Rdr | 30 03 99 9a | ok | READBLOCK(3)
Tag | f1 10 ff ee a5 00 00 00 e1 0f 19 9e 09 26 f9 0c | |
| dd c4 | ok |
Rdr | 1b f4 e2 0b f4 27 1f | ok | PWD-AUTH KEY: 0xf4e
Tag | 80 80 64 16 | |
Rdr | 3a 00 39 82 fc | ok | READ RANGE (0-57)
Tag | 04 3d 36 87 4a b3 49 80 a7 48 0f e0 f1 10 ff ee | |
| a5 00 00 00 e1 0f 19 9e 09 26 f9 0c fd 62 96 1d |
Report serial
04 B4 96 EA 21 4B 80
Signature of
04 B4 96 EA 21 4B 80
Image of
04 3D 36 4A B3 49 80

Originality Check

- Supply any known ID and signature pair during selection
- Any Amiibo data can be returned through the read commands afterwards, regardless of the contained tag ID
    - Patched on Switch in 5.0.0; no fix for Wii U
- *No crypto keys necessary*
    - Don’t need to do any crypto rewriting to load an image
    - Arbitrary serial number in tag data



Player

```
[ jamchamb@ubuntu ~/Documents/proxmark3/client ]  [amiibo|…3]
$ ./proxmark3 /dev/ttyACM0
```

Future Work

- Finish antenna for Switch
- Examine new layer of encryption added to application data in games like Splatoon 2
- Create test harness for fuzzing on the Switch, 3DS

Tool Release

Get amiimikyu at
https://github.com/nccgroup/proxmark3-amiimicyou

amiitool Lua wrapper at
https://github.com/jamchamb/amiitool

Thanks to...

- Jeff Dileo
- Nolan Ray