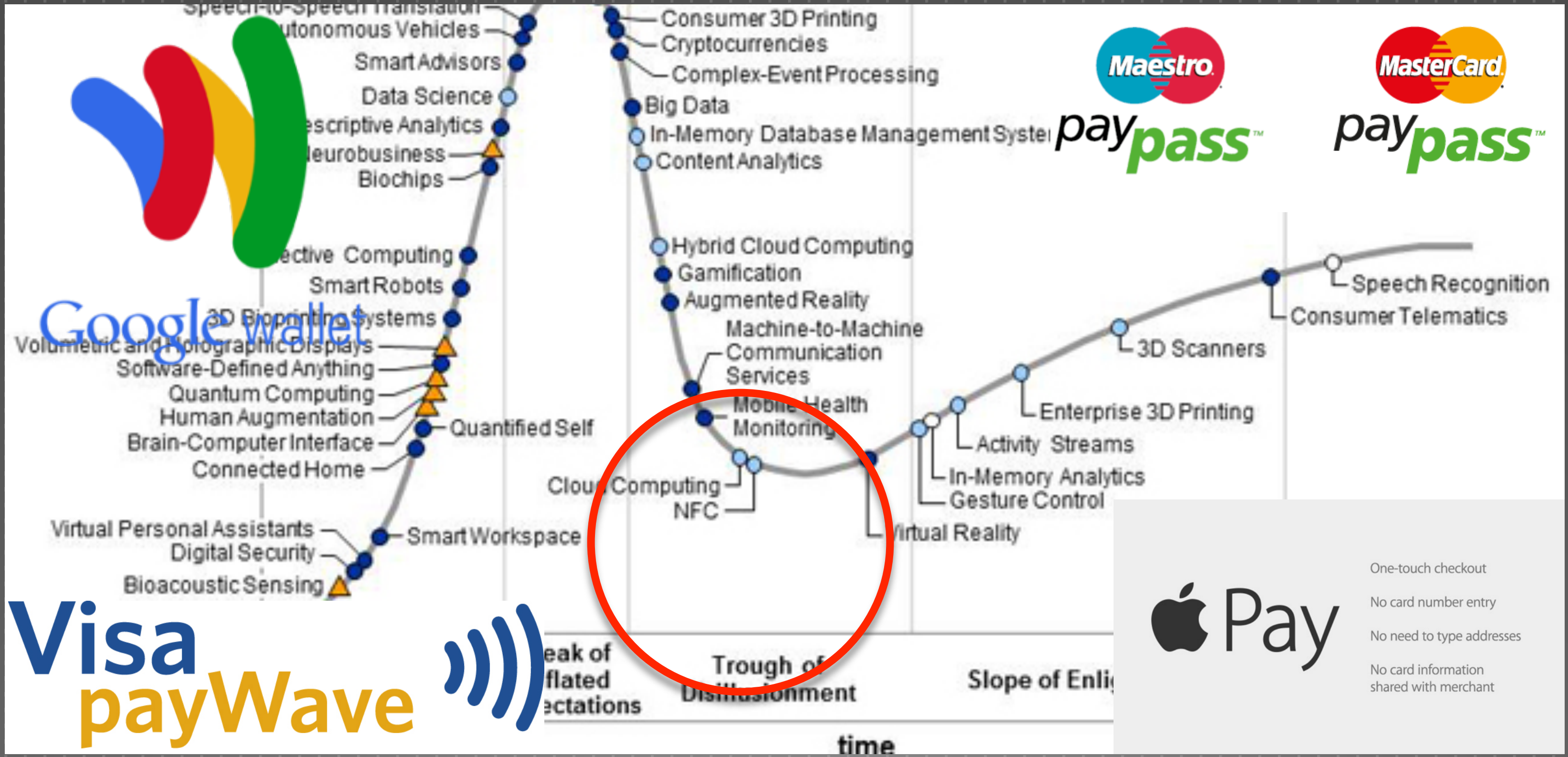
CRASH AND PAY

Cloning and Fuzzing the NFC world.

ABOUT ME

- Principle Consultant at Payment Security Consulting
- Banking, Payments, Certifications, breaking stuff; repairing it; I do it all.
- Did some fun stuff last year – this year no music though.
- Enjoys buying stuff that shouldn't be resold on ebay…

NFC – THE NEW SOURCE FOR INTIMATE CELEBRITY MOMENTS?



Source: Gartner Hype Cycle 2014 http://www.gartner.com/technology/research/hype-cycles/

INSPIRATIONS FOR THIS TALK

- “Don’t Stand So Close To Me, An analysis of the NFC attack surface” – Charlie Miller 2012
- “PinPadPwn” – Nils & Rafael Dominguez Vega Pin Pads, 2012
- “Credit Card Fraud - The Contactless Generation” Kristian Paget, 2012
- “Mission Mpossible” –Nils and Jon Butler 2013
- “Cloning Credit Cards: A combined pre-play and downgrade attack on EMV Contactless” - Michael Roland 2013

QUICK NFC/RFID PRIMER

- Looking at ISO14443 tags today.
- Going to skip over the basics – see better talks about that stuff.
- Focus is on the higher level stuff and it is handled.
- Application Data Units(APDUs) is how data is exchanged by cards after initialization.

NFC CARDS

- Cards are little computers
- Contain a SoC, RAM, ROM and interfaces
- Mainly two OS's, JavaCard and MULTOS
- JavaCard is a stripped down Java VM – with apps programmed in Java.
- MULTOS is a custom VM, apps programmed in C and then compliled into byte-code.
- Apps are signed and loaded by Issuers.
- Keys, Certs and other user data is put on cards using a process called "Personalization"

NFC CARDS – ISO14443

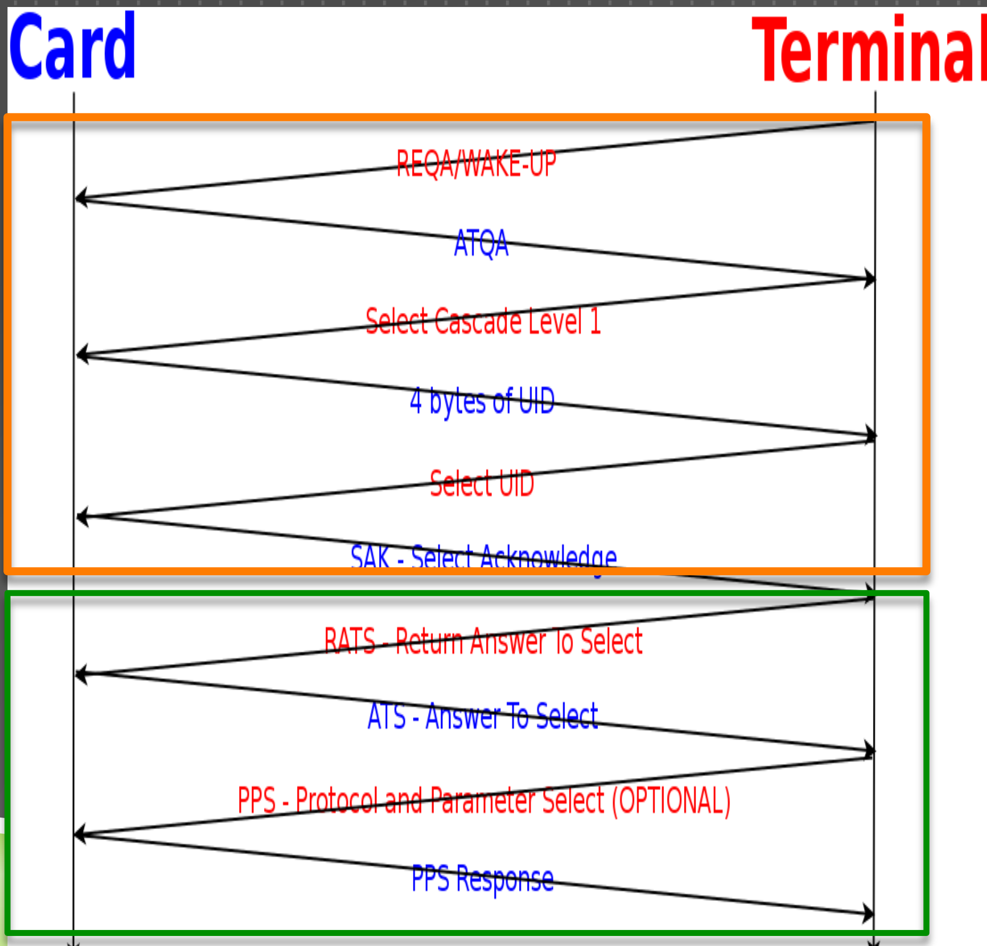
Differ by how data is physically transferred, and the initialization process.

Type A – Developed by Phillips/NXP

Type B – Developed by Innovatron

Type F/Felica – Developed by Sony (not in standard)

TYPE A CARD INITIALISATION




```
Rdr | 26
Tag | 04 00!
Rdr | 93 20
Tag | cf! 1f ab ae d5
Rdr | 93 70 cf 1f ab ae
d5 f1 1b
Tag | 28! b4! fc!
Rdr | e0 50 bc a5
Tag | 0b! 78 80 81 02!
      4b 4f! 4e 41! 14!
      11! 8a 76
Rdr | b2 67 c7
Tag | a3! 6f! c6!
```

ISO-14443 DATA FRAMES

- We use frames to cut data up into nice chunks.
- The card/terminal tell us how big a frame is
- The protocol then chunks your APDU into the frame size and sends it over the wire
- The receiver ACK/NACKs the frames.
- Very basic, not a routing protocol for example.

BLOCKS IN ISO14443-A

Byte #	1	2-(FRAME SIZE-2)	FRAME SIZE-2	FRAMESIZE-1
Description	Block Coding	Data	CRC	CRC

Information Block (I-Block): used to transmit normal data
Receive Ready Block(R-Block): indicates ready to receive data
Supervisory Block (S-Block): used for protocol messaging - initialisation

I-Block Coding:

Bit#	8	7	6	5	4	3	2	1
Description	0	0	0	Chaining	Card ID	Node Address	1	Block Number

```
|Rdr|e0 50 bca5
|Tag|a3 6fc6
|Rdr|02 00a4 [cut data] e042
|Tag|02 6f31[cut data] adde
|Rdr|03 00a4 [cut data] bc41
|Tag|13 6f43[cut data] 5faf
|Rdr|a2 e6d7
|Tag|02 2050 [cut data] cbe1
```

ISO 7816 – APDUS

- ISO 7816 – standard for ID cards with integrated circuits.
- Part 4 covers APDUs – Application Protocol Data Unit – how we format data

Command APDU (sent from Terminal)

Byte	1	2	3	4	5	<VAR>	
	CLA	INS	P1	P2	Lc	Data	Le
Description	Class	Instruction	Parameter Byte 1	Parameter Byte 2	Data Length		Expected Response Length

Response APDU (sent from Terminal)

Byte	<VAR>	<VAR>+1	<VAR>+2
Desc.	Response Data	SW1	SW2

HOW WE ENCODE DATA FOR EMV PAYMENTS. ASN.1 BER-TLV

TAG	LENGTH	VALUE

- Tag = what does the data represent. Normally 1 or 2 bytes long – but no hard limit
- Length = the Length of the data. No hard limit to the length – usually you are limited by your hardware
- Value = data to send. Easy!

TAG FORMATTING

Bit	8	7	6	5	4	3	2	1
	Class		P/C	Tag Number				

Class	Bit 8	Bit 7	Description
Universal	0	0	The type is native to ASN.1
Application	0	1	The type is only valid for one specific application
Context-specific	1	0	Meaning of this type depends on the context (such as within a sequence, set or choice)
Private	1	1	Defined in private specifications

If a tag number is 31, then the tag number is stored in the subsequent bytes after. Bit 8 of these bytes tells us when to stop 1=keep going, 0=stop

LENGTH FORMATTING

- Short Form – 1 byte long
- Bit 8 set to zero indicates that the remaining bits indicate length of data
- Binary values, so max data length of 127 bytes
- E.g 67(0x43) byte length is encoded as '0x43', easy
- Long Form – as many bytes as possible
- First byte tells us number of length bytes to follow. Bit 8 is set to '1'
- E.g 8567(0x2177) byte length is encoded as '0x842177'

TEMPLATES

- TLV Tags that are used to hold many other TLV Tags
- Used to hold many TLV tags.
- Can be nested
- E.g SELECT PPSE Response:

6F FCI Template
- 84 DF Name
- A5 FCI Proprietary Template
  - BF0C FCI Discretionary Data
    - 61 Directory Entry
      - 4F ADF Name (Application ID)
      - 87 Application Priority Indicator (API)

TOOLS OF THE TRADE

ACR-122U

Bout $60, reads lost of stuff.
Fickle – loves to crash, crap error handling
Can be made to support card emulation (couldn't be bothered myself)
Good to get started understanding stuff
Lots of limitations – like limited APDU length(~260 bytes),
Stuck with what the interface chip gives you.
No command chaining support (at least in RFIDIOT)
Charlie Millers talk on fuzzing RFID used this.
Read that, its pretty awesome:
https://media.blackhat.com/bh-us-12/Briefings/C_Miller/BH_US_12_Miller_NFC_attack_surface_Slides.pdf

ANDROID PHONES WITH NFC

Prior to 4.4.4 (KitKat) Card Emulation not officially supported. But Cynanogen mod lets you.

NXP chip supports emulation but not in official AOSP ☹, watch out for pre 2013 android NFC phones

Broadcom chip does, which was added in Nexus 4, Samsung Galaxy S4 etc

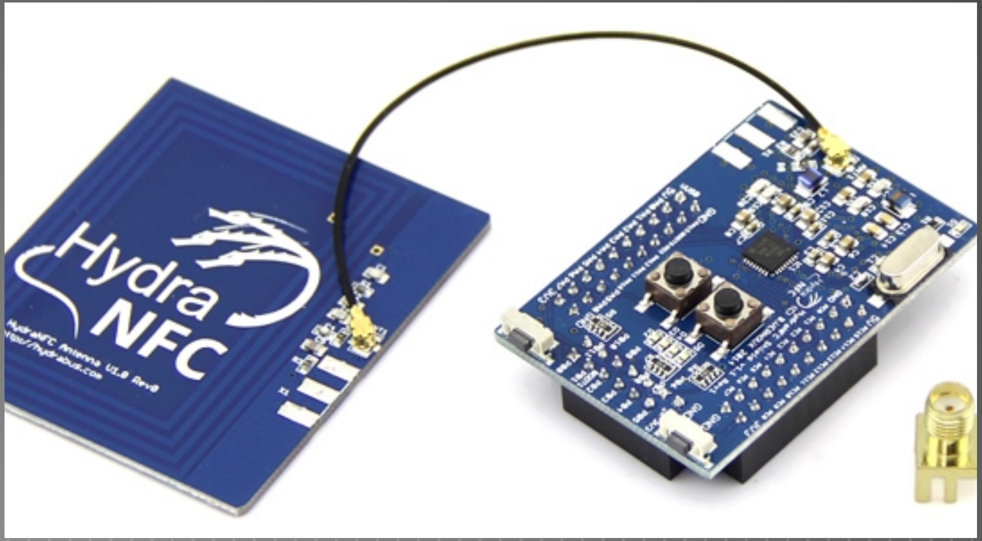
Better then ACR-122U cos its less buggy – but limited to chip support stuff – can’t spoof UID – limited by internal buffer lengths (2472 in Nexus4).

HYDRANFC+HYDRABUS

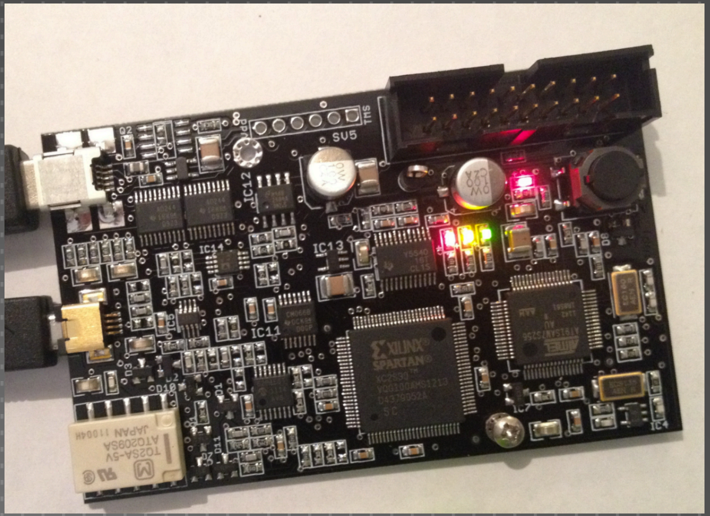
Coming soon
No FPGA, so cheaper then proxmark3.
Sniffing, R/W and emulation using TI TRF7970A chip.
This chip lets you spit out raw data.
Probably be about US$120
all up
– so half cost of proxmark3

No idea how good it is.

hydrabus.com



PROXMARK 3 – GITHUB.COM/PROXMARK/PROXMARK3



- Grandaddy of RFID Research
- US$229 PCB only ☹
- Supports 125/134KHz, 13.56MHz.
- Heavily moddable
- FPGA handles raw signals,
- ARM higher protocol stuff
- Super powerful – Super painful as well. Basic command line.
- API is a bit hairy
- Needs an update – bugger all memory, limits amount of data you can send.
- Lots of bugs! But good development community.

FUCKING NFC PAYMENTS, HOW DO THEY WORK?

EMV CONTACTLESS STANDARD

- Integrates all major card brands implementation of NFC payments.
- Available on the EMVCO website
- Book C contains 7 “Kernel” options:
  - Kernel 1 for some cards with JCB AIDs and some cards with Visa AIDs
  - Kernel 2 for MasterCard AIDs
  - Kernel 3 for Visa AIDs
  - Kernel 4 for American Express AIDs
  - Kernel 5 for JCB AIDs
  - Kernel 6 for Discover AIDs
  - Kernel 7 for UnionPay
- These documents provide you all you need to know on how a major card brand NFC payments system should work.
- I’m gonna focus on Mastercard and VISA in this talk.

COMMON COMMANDS FOR NFC PAYMENTS (MASTERCARD)

Command Name	CLA	INS	P1	P2	What does it do
SELECT PPSE	00	A4	04	00	Select Payment System Environment
SELECT	00	A4	xx	xx	Select an application on the card
GET PROCESSING OPTIONS	80	A8	xx	xx	Initiate a transaction, get card parameters
READ RECORD	00	B2	xx	xx	Get data from the card
COMPUTE CRYPTOGRAPHIC CHECKSUM	80	2A	8E	80	Generate dynamic CVV
GENERATE APPLICATION CRYPTOGRAM	80	AE	xx	00	Create Application Cryptogram for Dynamic Authentication

SELECT PPSE

CLA	INS	P1	P2	Lc	Data	Le
00	A4	04	00	0E	325041592E5359532E4444463031	00

Initiates the NFC Payment Transaction
Same for all NFC payment cards
Data is "2PAY.SYS.DDF01", for contact EMV we use "1PAY.SYS.DDF01"

The response from the card consists of returning the *FCI* containing the list of *PayPass* applications (*AIDs*) supported by the card.
This tells us what AID we should select, be it mastercard visa discover etc.

SELECT

CLA	INS	P1	P2	Lc	Data	Le
00	A4	04	00	05-10	AID to select	00

This command selects the application you want to use on the card.
We do this by providing by selecting the AID value corresponding to the card detected.
A successful select returns Label, Application Priority, Language Preference and PDOL.
After this we can start to perform our transaction

Card scheme	RID	Product	PIX	AID
Visa	A000000003	Visa credit or debit	1010	A0000000031010
		Visa Electron	2010	A0000000032010
		V PAY	2020	A0000000032020
		Plus	8010	A0000000038010
MasterCard	A000000004	MasterCard credit or debit	1010	A0000000041010
		MasterCard[5]	1010	A0000000041010
		Maestro (debit card)	3060	A0000000043060
		Cirrus (interbank network) ATM card only	6000	A0000000046000

GET PROCESSING OPTIONS

CLA	INS	P1	P2	Lc	Data	Le
00	A8	00	00	Var.	PDOL data	00

This initiates a transaction with the card.
It responds with the Application Interchange Profile(AIP)
Application File Locator(AFL) tells us what records are available on the card to read.

AIP Byte 1 (Leftmost)

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	x	x	x	x	x	x	x	RFU
x	1	x	x	x	x	x	x	SDA supported
x	x	1	x	x	x	x	x	DDA supported
x	x	x	1	x	x	x	x	Cardholder verification is supported
x	x	x	x	1	x	x	x	Terminal risk management is to be performed
x	x	x	x	x	1	x	x	Issuer authentication is supported [19]
x	x	x	x	x	x	0	x	RFU
x	x	x	x	x	x	x	1	CDA supported

AIP Byte 2 (Rightmost)

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	x	x	x	x	x	x	x	Reserved for use by the EMV Contactless Specifications
x	0	x	x	x	x	x	x	RFU
x	x	0	x	x	x	x	x	RFU
x	x	x	0	x	x	x	x	RFU
x	x	x	x	0	x	x	x	RFU
x	x	x	x	x	0	x	x	RFU
x	x	x	x	x	x	0	x	RFU
x	x	x	x	x	x	x	0	RFU

Table 37: Application Interchange Profile

READ RECORD

CLA	INS	P1	P2	Le
00	B2	Record Number	SFI	00

This is used to fetch data objects off the card.
SFI = Short File Indicator.
These records hold data such as Track Data, Public Keys, Expiry Dates etc.
We use this command to retrieve data from the card.
This data is all in plain-text…

Table 59—SFI 1 – Record 1

Tag	Name	Length (bytes)	Presence
'9F6C'	*Mag Stripe Application Version Number (Card)*	2	M
'9F62'	*Track 1 Bit Map for CVC3 (PCVC3$_{TRACK1}$)*	6	C[1]
'9F63'	*Track 1 Bit Map for UN and ATC (PUNATC$_{TRACK1}$)*	6	C[1]
'56'	*Track 1 Data*	var up to 76	O
'9F64'	*Track 1 Nr of ATC Digits (NATC$_{TRACK1}$)*	1	C[1]
'9F65'	*Track 2 Bit Map for CVC3 (PCVC3$_{TRACK2}$)*	2	M
'9F66'	*Track 2 Bit Map for UN and ATC (PUNATC$_{TRACK2}$)*	2	M
'9F6B'	*Track 2 Data*	var up to 19	M
'9F67'	*Track 2 Nr of ATC Digits (NATC$_{TRACK2}$)*	1	M
'9F68'	*Mag Stripe CVM List*	var up to 32	M

[1] This data element must be present if *Track 1 Data* is present.

WHY AM I EXPLAINING MAGNETIC STRIPES IN 2014?

TRACK 1 EXPLAINED

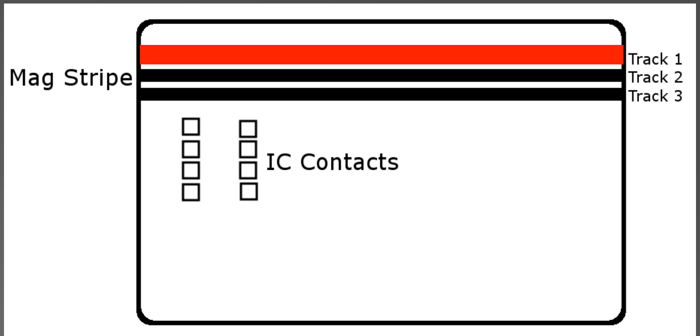
Card Data:

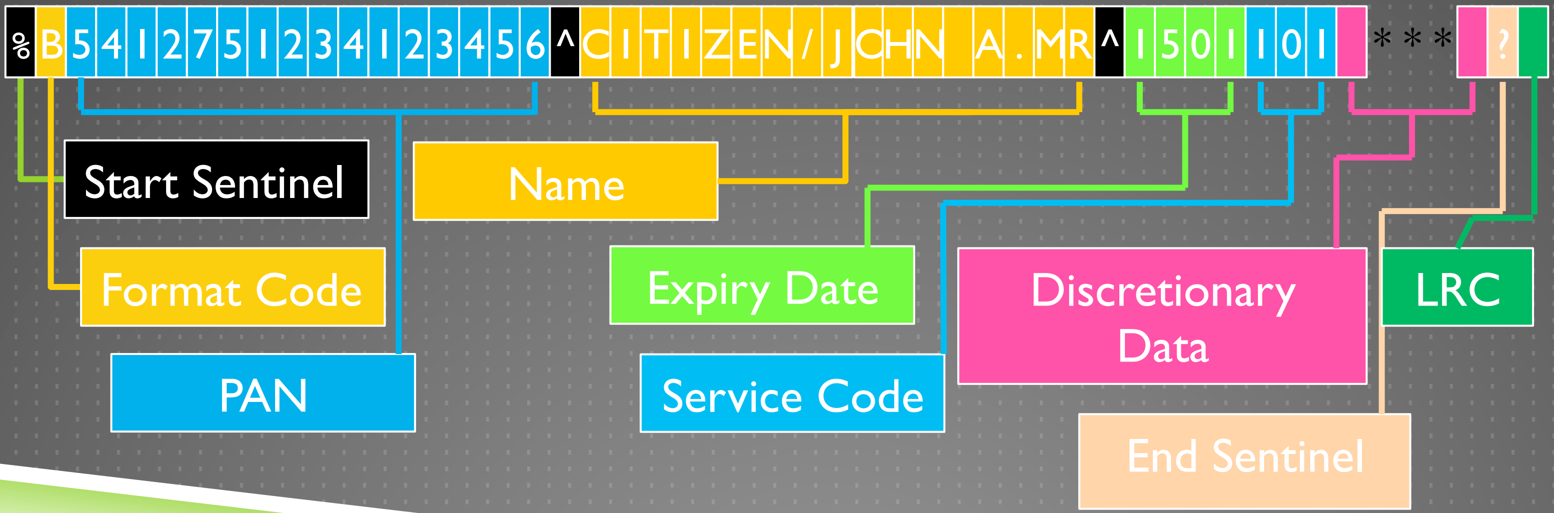
PAN: 5412 7512 3412 3456

Card Holder Name: MR JOHN A. CITIZEN

Expiration Date: 01/15

Service Code: 101(International Card, Normal Authorization, Normal Verificiation)





TRACK 2 EXPLAINED

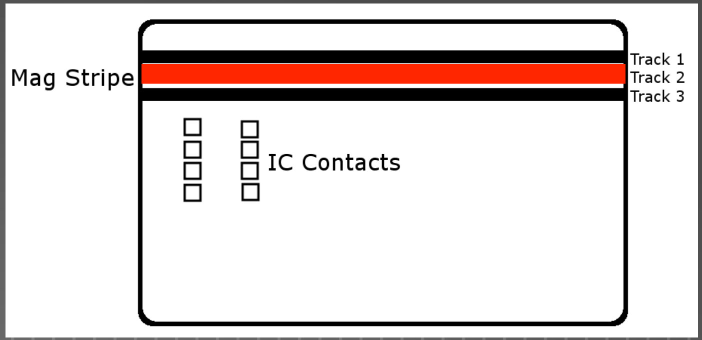
Card Data:

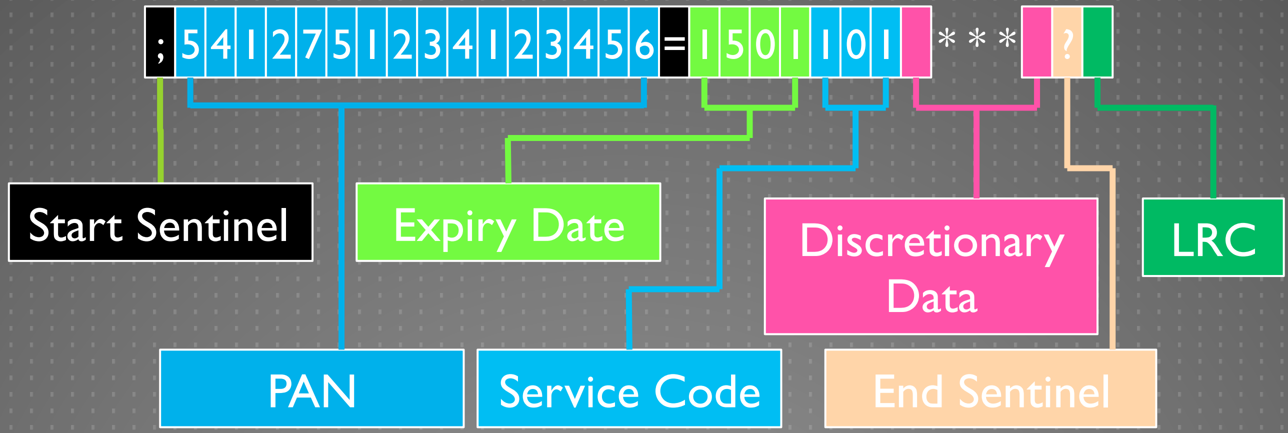
PAN: 5412 7512 3412 3456

Card Holder Name: MR JOHN A. CITIZEN

Expiration Date: 01/15

Service Code: 101(International Card, Normal Authorization, Normal Verificiation)





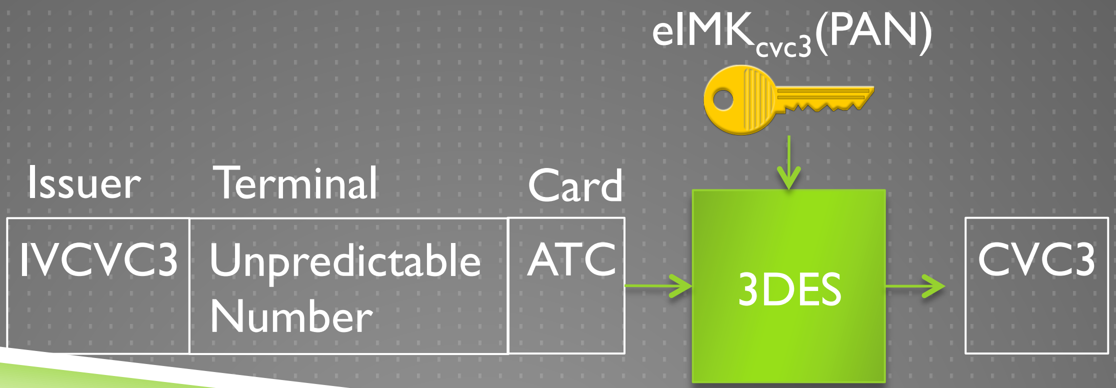
COMPUTE CRYPTOGRAPHIC CHECKSUM (PAYPASS)

CLA	INS	P1	P2	Lc	Data	Le
00	2A	8E	80	Var.	UDOL related data	00

This command causes the generation of CVVs for both Track1 and Track2; as well as returning the Application Transaction Counter.

ATC is a monotonic counter of 16-bits which tells us the number of transactions that have occurred on the card. It is a key indicator for the payment processor of fraud (i.e. it should always increase)

This is the key mechanism for authenticating transactions.



GENERATE APPLICATION CRYPTOGRAM

CLA	INS	P1	P2	Lc	Data	Le
00	2A	Xx	00	Var.	CDOL related data	00

Used to handle the risk management of the transaction.

The terminal proposes a risk management to perform and the card can either reject or accept.

TC > ARQC > AAC

In Australia, all transactions are online, offline is not supported.

Table 13—Generate AC Reference Control Parameter

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0							AAC
0	1							TC
1	0							ARQC
1	1							RFU
		x						RFU
			0					Combined DDA/AC generation not requested
			1					Combined DDA/AC generation requested
				x	x	x	x	RFU

The data field of the command message is coded according to the *CDOL* following the rules as defined in Section 4.2 of PART II.

Type	Abbreviation	Meaning
Application Authentication Cryptogram	AAC	Transaction declined
Authorization Request Cryptogram	ARQC	Online authorization requested
Transaction Certificate	TC	Transaction Approved (offline)

M/CHIP AND MAGSTRIPE MSD, VSDC, QVSDC

- M/Chip, VSDC and qVSDC are fairly equivalent.
- Terminals must support both M/Chip and MagStripe for Mastercard.
- For Visa, terminals don't have to support MagStripe.
- M/Chip and VSDC is basically your normal EMV protocol
- MagStripe is intended for legacy hardware and networks (i.e everything that isn't EMV ready)

PAYPASS CLONING.

- Step 1 – read and copy card records
- Step 2 – Generate dictionary of COMPUTE CRYPTOGRAPHIC CHECKSUM responses for all possible terminal random numbers
- Step 3 – Flip the M/CHIP support bit (tag 82)
- Step 4 – replay stored records to the terminal
- Step 5 – look up UN returned by the terminal in the dictionary
- Step 6 – collect purchase and get out of there.

DEMO!

%BXXXXXXXXXXXX6614^ / ^170620175339 0000000690000002?;
XXXXXXXXXXXX6614=17062017533923801002?(
%BXXXXXXXXXXXX6614^ / ^170620179581 0000000453000002?;
XXXXXXXXXXXX6614=17062017958186801002?ß

WHY IT WORKS

- UN is a Binary Coded Decimal, max of 999,999 values
- But Card issuer sets actual length of UN used
- Typically 0 bytes for pre-loaded card
- Typically 2-3 digits long for a CC card from issuer
- So that means a UN of 0-100 for 2 digits
- And UN of 0-1000 for 3 digits
- So quick to generate all possible UNs in under a minute for most cards.
- And we can perform more then one transaction, as long as every UN is greater than the last.

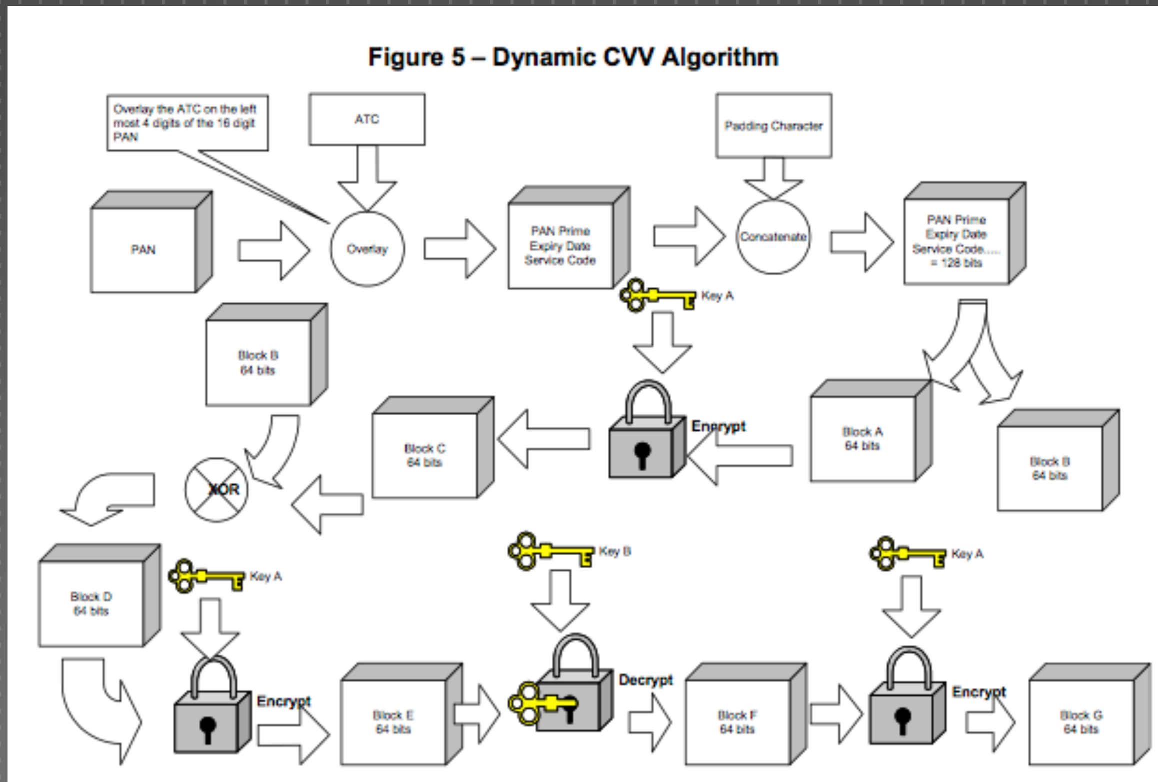
HOW DO WE DETERMINE THE LENGTH OF THE UN THE ISSUER TOLD THE CARD TO USE?

- 1. We read the 1st record
- 2. This contains:
- Ktrack1 (9f63)
- Ktrack2 (9f66)
- Ttrack1(9f64)
- Ttrack2(9f67)
- Ktrack1 is "Track 1 Bit Map for UN and ATC"
- Ktrack2 is "Track 1 Bit Map for UN and ATC"
- Ttrack1 is "Track 1 Number of ATC Digits"
- Ttrack2 is "Ttrack1 is "Track 2 Number of ATC Digits"
- We count the bits in Ktrack1, then minus the Ttrackx to get the number of bits used for the UN.

PAYWAVE CLONING

- Similar to PayPass
- We use MagStripe Profile again.
- However Paywave is worse, why?
- Visa's iCVV algorithm

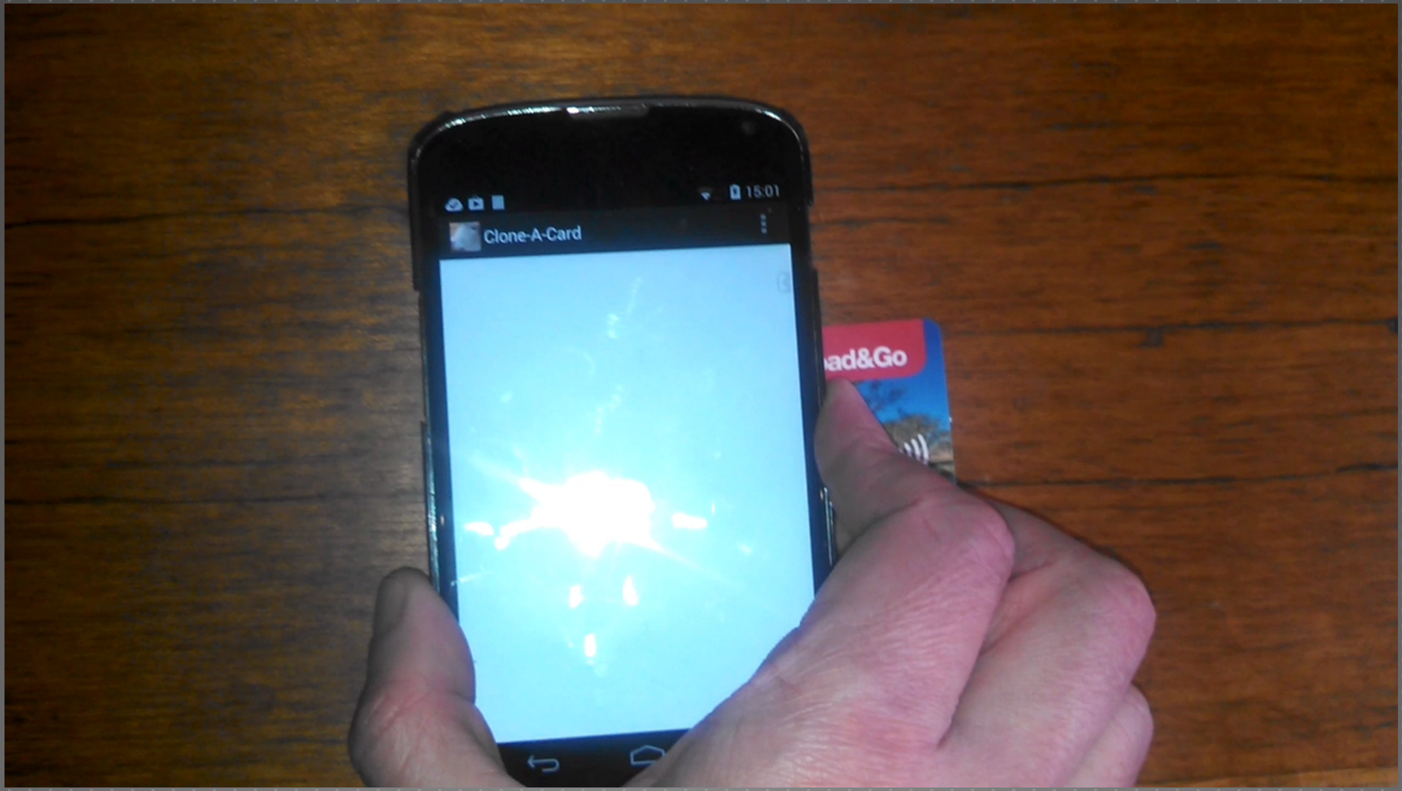
VISAS' PATENTED ICVV ALGORITHM



PAYWAVE CLONING.

- Step 1 – read and copy card records
- Step 2 – Turn the magstipe bit on
- Step 3 – replay stored records to the terminal
- Step 4 – collect purchase and get out of there.

DEMO!



BUT WAIT, THERES MORE

- You can have Static CVVs..
- What does that mean
- Means that the track data is always static in MagStripe Mode
- So we can just clone the card, just like your ye olde MSR card.

HOW DO WE FIX THIS?

- Payment Processors should reject all MagStripe transactions (they are online only as it is).
- All cards and terminals should reject any transaction that isn't a CDA'd
- Legacy equipment however makes this difficult….

FUZZING EMV

- Little has been published on fuzzing EMV interfaces, all about protocol, nothing on implementation
- See MWR talks for what happens when you screw this up
- Or ask these guys ;)
- So same bugs will be

there if we test the NFC interface

As well.

ANDROID FUZZING

- Card Emulation is supported by many available contactless ICs Since Android uses either a NXP or Broadcom chip, it can do card emulation
- Used to be only available in Cynanogen
- As of Android Kit Kat (4.4.4) its supported officially. But only for Broadcom chipsets
- So I bought a Nexus 4 off ebay – found I got a faulty one – bought another one.
- And started playing around

ANDROID HOSTAPDUSERVICE

- God its nice to have an API that handles everything.
- You register supported AIDs with the NFC service.
- OS detects the AID and routes it to your program.
- You write the application to handle APDUs

LIMITATIONS

- Can't control initialization stuff (UID, RATS etc) – up to what IC picks for you
- All overhead stuff is done for you, like framing, CRCs, protocol stuff.
- Max data to send is ~2488 bytes on nexus 4
- Other than that – its awesome for quick and easy fuzzing.

DEMO!

Received APDU = 00A404000E325041592E5359532E444446303100

Send APDU =

6f81f4

8481d0<22256e22*lots>

a51f

bf0c1c

611a4f07a0000000031010

500c566973612050726

OKAY – WHAT JUST HAPPENED???

- Here I'm fuzzing the SELECT PPSE response.
- So, first 10 or so test cases are short – no response.
- Once I send a lot of data…
- Pop goes the weasel.
- This is not a good sign of course – Crashing something this quick I didn't expect.
- Crash is most likely related to buffer overflow

RESULTS…

- Fuzzing initialization stuff doesn't really get you anywhere – contactless ICs handle that stuff
- Crashed my contactless reader quickly (like the first test case generated), it reboots cos its an embedded system :<
- Early days of this stuff, easy to crash stuff! but lack of crash logs make creating exploits more difficult.
- In the process of reversing F/W update, adding JTAG to develop an exploit.
- Embedded systems are great targets to play with, as they usually don't contain the protections you have to deal with in PCs

AREAS FOR FUTURE RESEARCH

- Fuzzing other reader hardware
- Other protocols, like ISO15693, Felica.
- Fuzz the "Internet Of Things" using SDR!
- Can you alter with the RFID controller firmware i.e like badusb?
- Passport Readers! Transport Systems! Door Entry Systems! The list is endless
- Basically this area is ripe for exploitation, easy pickings to be had if you're an intrepid researcher

CONCLUSIONS

- We all love RFID.
- But no one actually tests for this stuff adequately.
- The ISO 7816 standard supports transport encryption – use this if you implement your own system…
- Embedded systems are prevalent in this space.
- Tools are out there for testing – you just to roll your own code of course.
- One day we will understand that RFID protocols and hardware is not magically secure, it needs to be tested.
- Secuity is not just protocols – implementation matters people…
- Certificates and Standards do not a secure system make.