SHA-1
backdooring
&
EXPLOITATION

brought to you by

Maria Eichlseder, Florian Mendel, Martin Schläffer
TU Graz, .at; cryptanalysis

@angealbertini
Corkami, .de; binary kung-fu

@veorq
Kudelski Security, .ch; theory and propaganda :-)

1. WTF is a hash function backdoor?
2. backdooring SHA1 with cryptanalysis
3. exploitation! collisions!

TL;DR:


```
>crypto_hash *
test0.jpg 13990732b0d16c3e112f2356bd3d0dad1....
test1.jpg 13990732b0d16c3e112f2356bd3d0dad1....
```

who’s interested in crypto backdoors?

(U) Base resources in this project are used to:

- (TS//SI//REL TO USA, FVEY) Insert vulnerabilities into commercial encryption systems, IT systems, networks, and endpoint communications devices used by targets.
- (TS//SI//REL TO USA, FVEY) Collect target network data and metadata via cooperative network carriers and/or increased control over core networks.
- (TS//SI//REL TO USA, FVEY) Leverage commercial capabilities to remotely deliver or receive information to and from target endpoints.
- (TS//SI//REL TO USA, FVEY) Exploit foreign trusted computing platforms and technologies.
- (TS//SI//REL TO USA, FVEY) Influence policies, standards and specification for commercial public key technologies.
- (TS//SI//REL TO USA, FVEY) Make specific and aggressive investments to facilitate the development of a robust exploitation capability against Next-Generation Wireless (NGW) communications.
- (U//FOUO) Maintain understanding of commercial business and technology trends.

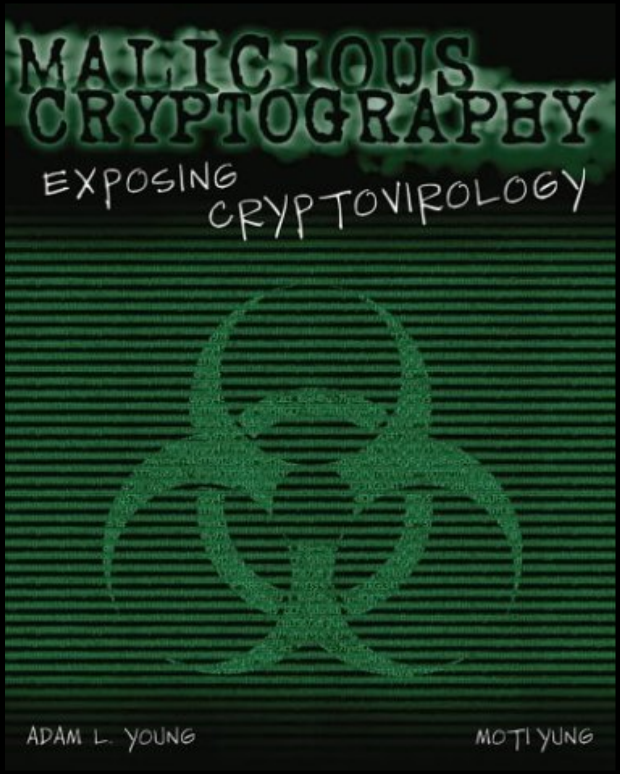
& Dual_EC speculation — https://projectbullrun.org

Clipper (1993)

crypto **researchers**?

PEOPLE SAY I DON'T CARE,

BUT I DO.



Young/Yung malicious cipher (2003)

- compresses texts to leak key bits in ciphertexts
- **blackbox** only (internals reveal the backdoor)
- other “cryptovirology” schemes

Stealthy Dopant-Level Hardware Trojans

Georg T. Becker[1], Francesco Regazzoni[2], Christof Paar[1,3], and Wayne P. Burleson[1]

Trojan Side Channels

Lightweight Hardware Trojans through Side Channel Engineering

Lang Lin[1] Markus Kasper[2] Tim Güneysu[2]
Christof Paar[1,2] Wayne Burleson[1]

Eve's SHA3 candidate: malicious hashing

Jean-Philippe Aumasson*

Nagravision SA, Switzerland

Abstract. We investigate the definition and construction of hash functions that incorporate a backdoor allowing their designer (and only her) to efficiently compute collisions, preimages, or more. We propose semi-formal definitions of various types of malicious generators—i.e. probabilistic algorithms modeling a malicious designer—and of the intuitive notions of undetectability and undiscoverability. We describe relations between the notions defined as well as basic strategies to design malicious hashes. Based on the observation that a backdoor can be at least as hard to discover as to break the underlying hash, we present a backdoored version of the SHA3 finalist BLAKE. This preliminary work leaves many open points and challenges, such as the problem of finding the most appropriate definitions. We believe that a better understanding of malicious uses of cryptography will assist combat it; malicious hash functions are indeed powerful tools to perform insider attacks, government espionnage, or software piracy.

2011: theoretical framework, but nothing useful what's a crypto backdoor?

not an implementation backdoor

example: RC4 C implementation (Wagner/Biondi)

```
#define TOBYTE(x) (x) & 255
#define SWAP(x,y) do { x^=y; y^=x; x^=y; } while (0)

static unsigned char A[256];
static int i=0, j=0;

unsigned char encrypt_one_byte(unsigned char c) {
    int k;
    i = TOBYTE(i+1);
    j = TOBYTE(j + A[i]);
    SWAP(A[i], A[j]);
    k = TOBYTE(A[i] + A[j]);
    return c ^ A[k];
}
```

a **backdoor** (covert) isn't a **trapdoor** (overt)

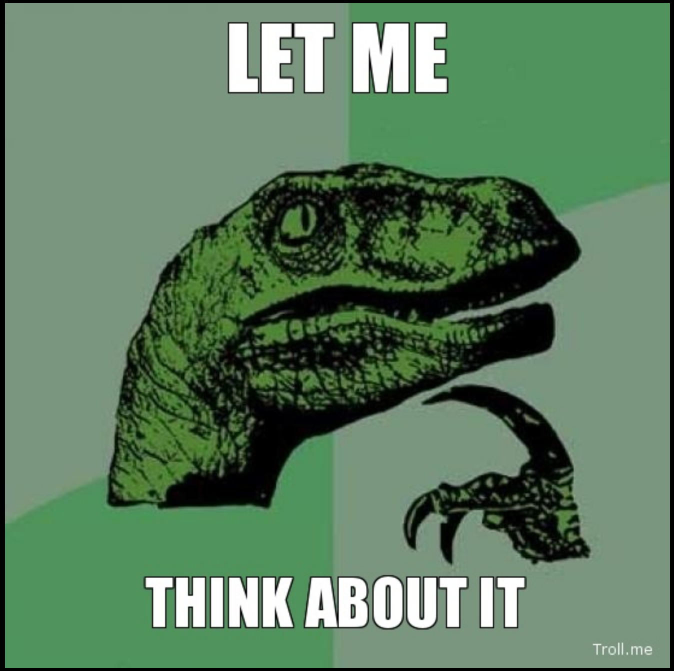
RSA has a trapdoor, NSA has backdoors

VSH is a trapdoor hash based on RSA

VSH, an Efficient and Provable Collision-Resistant Hash Function

Scott Contini[1], Arjen K. Lenstra[2], and Ron Steinfeld[1] backdoor in a crypto hash?

“some secret property that allows you to efficiently break the hash”



"break" can be about collisions, preimages…

how to model the stealthiness of the backdoor…

exploitation can be deterministic or randomized…

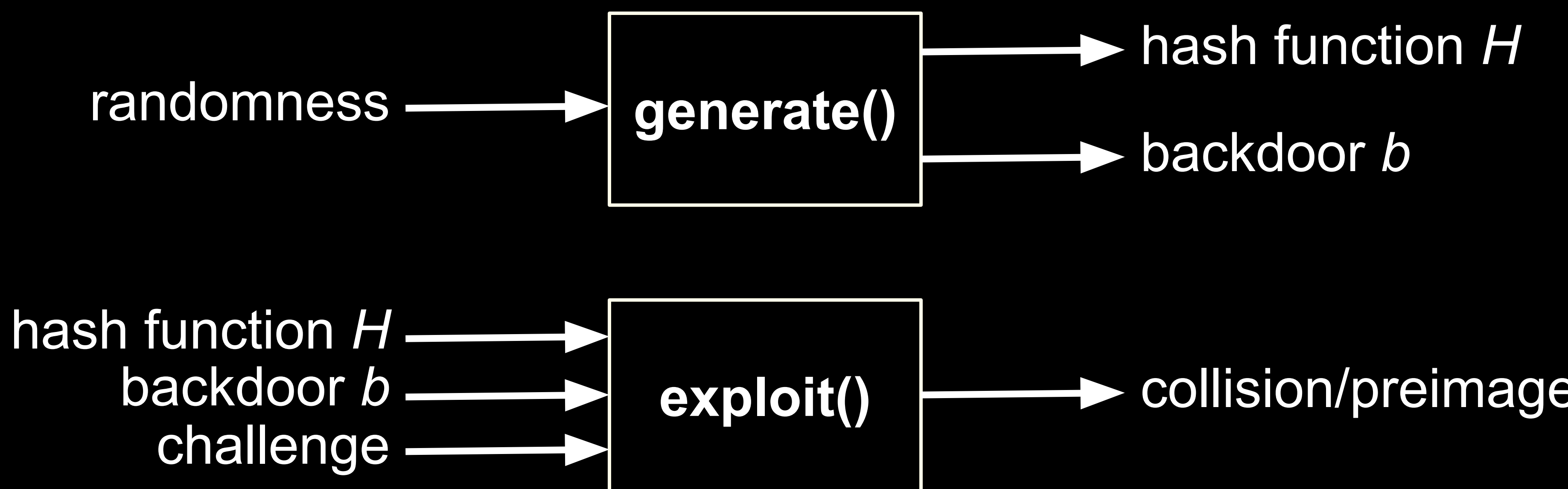
role reversal

Eve wants to achieve some security property

Alice and Bob (the users) are the adversaries

definitions

malicious hash = pair of algorithms



exploit() either "static" or "dynamic"

taxonomy

static collision backdoor

returns **constant** m and m' such that $H(m)=H(m')$

dynamic collision backdoor

returns **random** m and m' such that $H(m)=H(m')$

static preimage backdoor

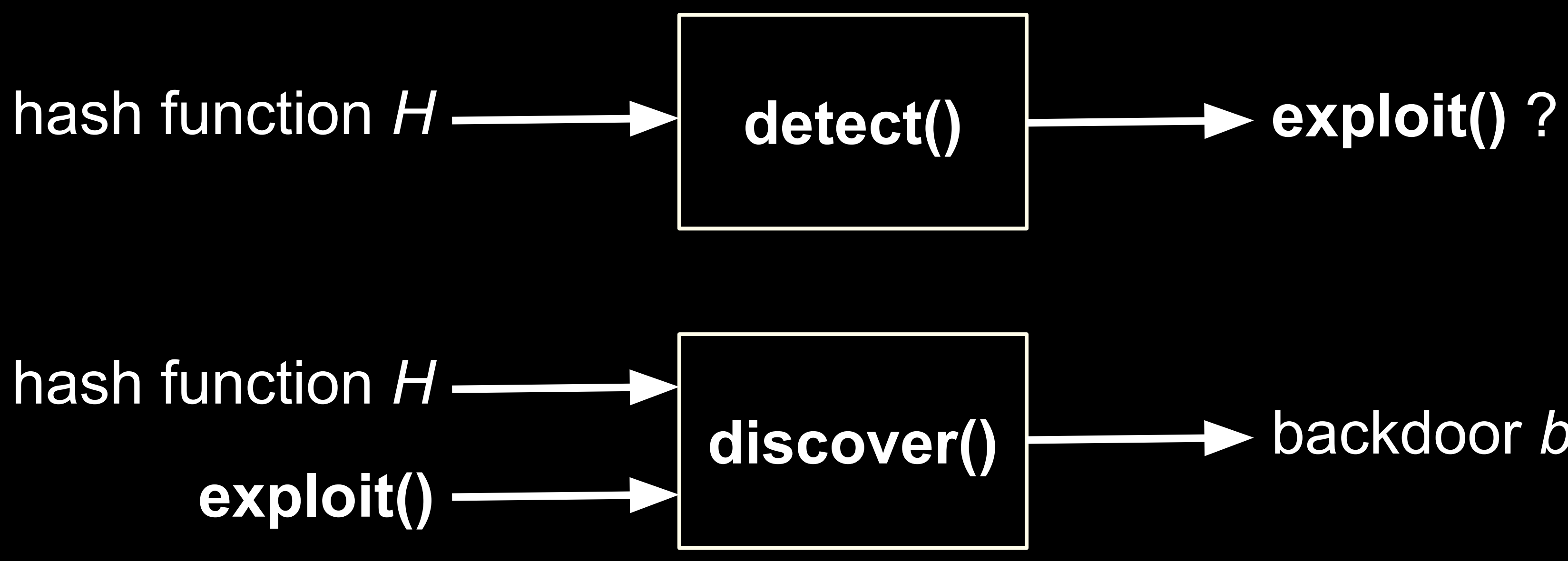
returns m such that $H(m)$ has low entropy

dynamic preimage backdoor

given h, returns m such that $H(m)=h$

stealth definitions

undetectability vs undiscoverability



detect() may also return levels of suspicion

H may be obfuscated...

our results

dynamic collision backdoor

valid structured files with arbitrary payloads

detectable, but undiscoverable

and as hard to discover as to break SHA-1

SHA-1

NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

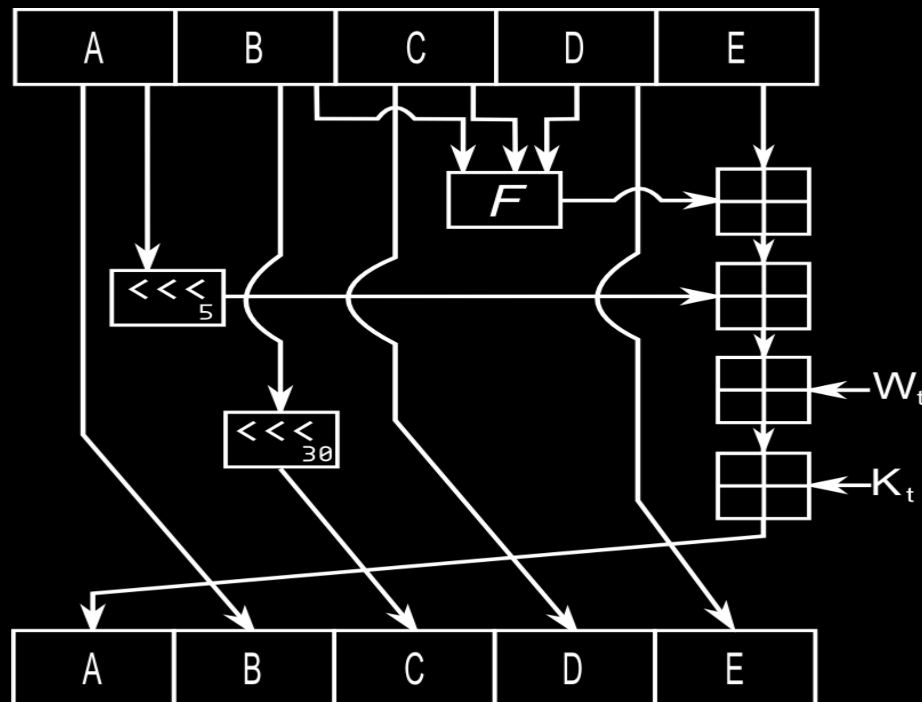
SHA-1
everywhere

RSA-OAEP, “RSAwithSHA1”, HMAC, PBKDF2, etc.

⇒ in TLS, SSH, IPsec, etc.

integrity check: git, bootloaders, HIDS/FIM, etc.

SHA-1



$$(W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \lll 1 \quad \text{for } 16 \leq i \leq 79 \ .$$

step i	K_r	f_r
$0 \leq i \leq 19$	5a827999	$f_{\text{IF}}(B, C, D) = B \wedge C \oplus \neg B \wedge D$
$20 \leq i \leq 39$	6ed9eba1	$f_{\text{XOR}}(B, C, D) = B \oplus C \oplus D$
$40 \leq i \leq 59$	8f1bbcdc	$f_{\text{MAJ}}(B, C, D) = B \wedge C \oplus B \wedge D \oplus C \wedge D$
$60 \leq i \leq 79$	ca62c1d6	$f_{\text{XOR}}(B, C, D) = B \oplus C \oplus D$

https://www.schneier.com/blog/archives/2005/02/sha1_broken.html

SHA-1 Broken

SHA-1 has been broken. Not a reduced-round version. Not a simplified version. The real thing.

Finding Collisions in the Full SHA-1

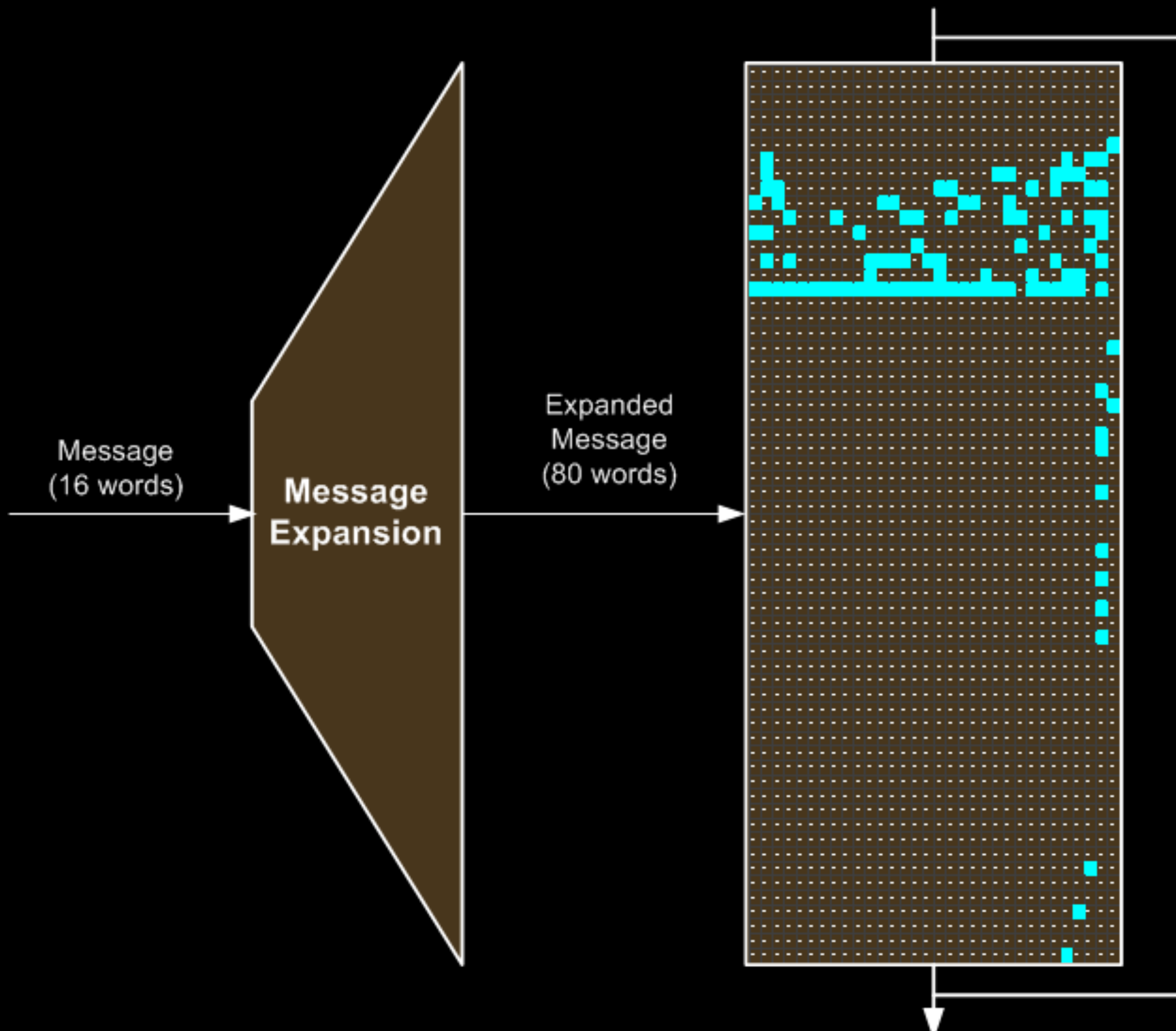
Xiaoyun Wang[1]*, Yiqun Lisa Yin[2], and Hongbo Yu[3]

[1] Shandong University, Jinan 250100, China, xywang@sdu.edu.cn
[2] Independent Security Consultant, Greenwich CT, US, yyin@princeton.edu
[3] Shandong University, Jinan250100, China, yhb@mail.sdu.edu.cn

but no collision published yet

actual complexity unclear (>2^{60})

Differential cryptanalysis for collisions
"perturb-and-correct"



2 stages (offline/online)

1. find a **good** differential characteristic
   = one of high probability

2. find **conforming messages**
   with message modification techniques

find a characteristic: linearization

0	1001111110001101100110001001010n	111111111101100011111111111000nu
1	00u1101001100--------001111u0n00	11u0001000000000-----0100unu0n00
2	n111n00---1------------uu000un00	u011n1-----------------000001000
3	0uuuu111--0---0---uu---0-0un11nn	nnu-n--------------------nn000u1
4	1n01u1110---u-n-------u0011001n0	uu1-u------------------00u0011uu
5	0011011n1n00--0-un0101-10n1u0n00	10u0u--------------------1101u11
6	n1n1n1n01000--1-100101-00n000011	1u111--------------------u-001u0
7	nu1nnnnnnnnnnnnnnnnnnnnnnnn000n1	0n10u00------------------n000nn1
8	101111-100110000000010000111nu0u1	n001u----------------------000u1
9	0-1010101000000000000000001un001	10110100-----------------01u1u00
10	u1n00------------------------01u	1011n---------------------0-00n1
11	-00-0--------------------1100001	u0u----------------------n1n0n00
12	-0010-----------------------00-1	u01-n----------------------100n0
13	----1-----------------------1100	n0100----------------------11011
14	-------------------------------n	u1u-u---------------------0000nn
15	1[illegible]	[illegible]-0un10010
16	0[illegible]	[illegible]1000un
17	------------------------------u-	nn01---------------------10111u1
18	-------------------------------n	nn001--------------------n-010nu
19	--------------------------------	un1----------------------un111n1
20	--------------------------------	n11-----------------------0011nu
21	------------------------------n-	0u01---------------------01110n1
22	[illegible]n	0u0[illegible]u1100nu
23	[illegible]	nn[illegible]un001n1
24	[illegible]	[illegible]100un
25	------------------------------n-	1n1---------------------011001n0
26	-n-----------------------------u	011----------------------u-110un
27	--------------------------------	nn00---------------------nu0u0n0
28	------------------------------u-	nn101---------------------11001u
29	n-----------------------------n-	uu1n---------------------n0101n0
30	--------------------------------	1n1u---------------------u1u01u0
31	--------------------------------	uu0u---------------------11101u0
32	-------------------------------u	01n----------------------10110un
33	--------------------------------	01n--------------------10nu00001
34	u-------------------------------	10u----------------------10100nu
35	------------------------------n-	nu0--------------------1001n11n1
36	------------------------------n-	1n0----------------------u0111n0
37	--------------------------------	nun--------------------00u1100u0
38	------------------------------u-	n0u-----------------------110001
39	--------------------------------	10n0-------------------10n010111
40	n-------------------------------	u01----------------------10000n0
41	--------------------------------	101----------------------01u1001
42	u------------------------------u	u10-0---------------------0111un
43	--------------------------------	n0u--------------------01nun0010
44	--------------------------------	n1u----------------------10000nu
45	------------------------------n-	nu0----------------------00111n1
46	------------------------------u-	uuu----------------------u1111u1
47	--------------------------------	uun--------------------10n0000u0
48	------------------------------u-	n00---------------------10101100
49	--------------------------------	100--------------------11n010100
50	------------------------------u-	010--------------------1-0100010
51	--------------------------------	010--------------------01n100010
52	--------------------------------	u01----------------------11100n0
53	--------------------------------	101--------------------010110101
54	-------------------------------n	n11----------------------101100n
55	--------------------------------	u00--------------------110u11000
56	------------------------------u-	100----------------------00011uu
57	------------------------------u-	1u1--------------------11n1011u0
58	-------------------------------u	1u1----------------------n00101n
59	--------------------------------	nu0--------------------10nu001n1
60	--------------------------------	101---------------------110000nu
61	------------------------------n-	un1--------------------0010000n1
62	------------------------------n-	1n--1------------------10u0111n1
63	--------------------------------	uu1--------------------11u0111u0
64	--------------------------------	u10-1------------------0100000n0
65	--------------------------------	0--------------------11110001011
66	------------------------------n-	111-------------------0-000011n1
67	--------------------------------	u1-------------------0110u100100
68	------------------------------u-	--0--------------------011000100
69	--------------------------------	u1---------------------01n000010
70	------------------------------u-	n-1--------------------111011101
71	--------------------------------	-0-------------------1100n01100-
72	------------------------------u-	u----------------------000001110
73	--------------------------------	--1------------------0011n111011
74	------------------------------u-	n-----------------------101111--
75	--------------------------------	---------------------0000n011101
76	--------------------------------	u----------------------1110001u-
77	--------------------------------	-----------------------101000--1
78	--------------------------------	n---------------------000011101-
79	--------------------------------	n--------------------111000101--

low-probability

high-probability

2^{-40}

2^{-40}

2^{-15}

find conforming messages

low-probability part: "easy", K_1 unchanged
use automated tool to find a conforming message

round 2: try all 2^{32} K_2's, repeat 2^8 times (**cost 2^{40}**)
consider constant K_2 as part of the message!

round 3: do the same to find a K_3 (**total cost 2^{48}**)
repeating the 2^{40} search of K_2 2^8 times….

round 4: find K_4 in negligible time

iterate to minimize the differences in the constants...

collision!

$K_{1...4}$	5a827999	4eb9d7f7	bad18e2f	d79e5877				
IV	67452301	efcdab89	98badcfe	10325476	c3d2e1f0			
m	ffd8ffe1	e2001250	b6cef608	34f4fe83	ffae884f	afe56e6f	fc50fae6	28c40f81
	1b1d3283	b48c11bc	b1d4b511	a976cb20	a7a929f0	2327f9bb	ecde01c0	7dc00852
m^*	ffd8ffe2	c2001224	3ecef608	dcf4fee1	37ae880c	87e56e6b	bc50faa4	60c40fc7
	931d3281	b48c11a8	b9d4b513	0976cb74	2fa929f2	a327f9bb	44de01c3	d5c00832
Δm	00000003	20000074	88000000	e8000062	c8000043	28000004	40000042	48000046
	88000002	00000014	08000002	a0000054	88000002	80000000	a8000003	a8000060
$h(m)$	1896b202	394b0aae	54526cfa	e72ec5f2	42b1837e			

1-block, vs. 2-block collisions for previous attacks

IM NOT TOTALLY USELESS.

I CAN BE USED AS A BAD EXAMPLE.

but it’s not the real SHA-1!

“custom” standards are common in proprietary systems

(encryption appliances, set-top boxes, etc.)

motivations:

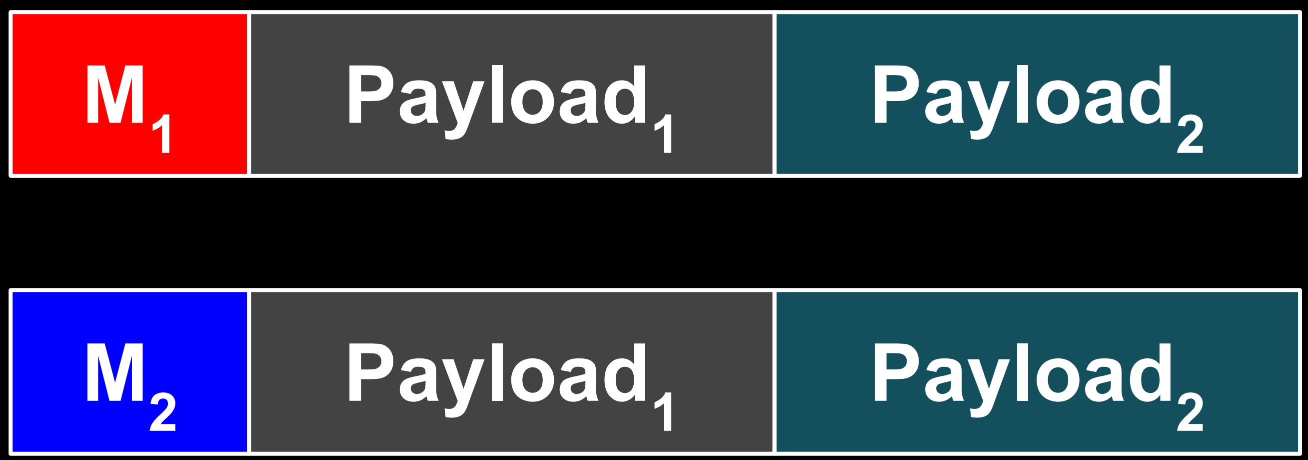
customer-specific crypto (customers’ request)

“other reasons”

how to turn garbage collisions into useful collisions?

(= 2 **valid files** with arbitrary content)

basic idea



where $H(M_1)=H(M_2)$

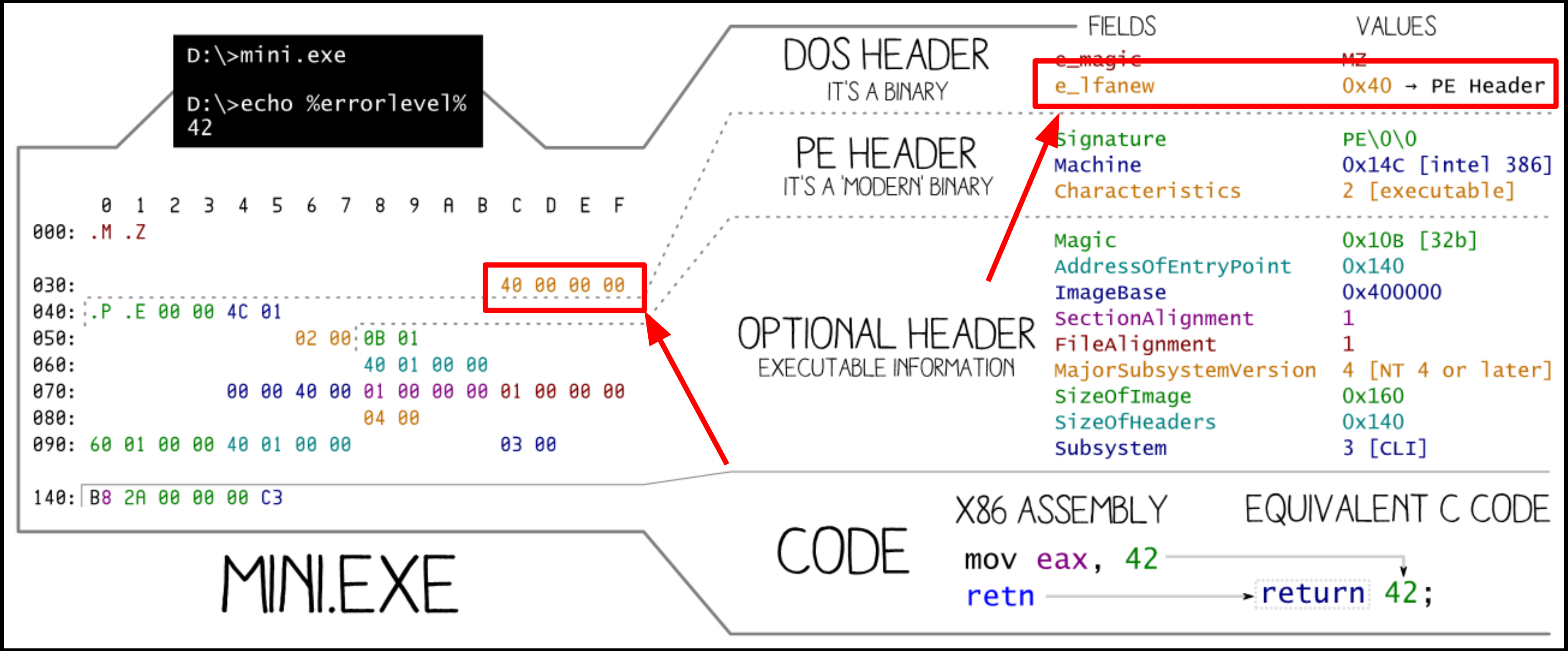
and M_x is essentially “process payload x”

constraints

differences (only in) the first block

difference in the first four bytes
$\Rightarrow$ 4-byte signatures corrupted

PE? (Win* executables, etc.)



differences forces EntryPoint to be at > 0x40000000

⇒ 1GiB (not supported by Windows)

PE = fail

ELF, Mach-O = fail
(≥ 4-byte signature at offset 0)

shell scripts?

```
#<garbage, 63 bytes>              //block 1 start
          ................................
#<garbage with differences> //block 2 start
          ________________________________

EOL                               //same payload

<check for block’s content>
```

```
0000000: 231d 1b91 3440 09d8 104d a6d3 54e1 102b  #...4@...M..T..+
0000010: b885 125b 4778 26bd fd37 2bee e650 082c  ...[Gx&..7+..P.,
0000020: 754b 1657 3811 bfd8 a5e0 b244 1a94 512a  uK.W8......D..Q*
0000030: cd36 a204 fee2 8a9f 3255 99aa b47a ed82  .6......2U...z..
0000040: 0a0a 6966 205b 2060 6f64 202d 7420 7831  ..if [ `od -t x1
0000050: 202d 6a33 202d 4e31 202d 416e 2022 247b   -j3 -N1 -An "${
0000060: 307d 2260 202d 6571 2022 3931 2220 5d3b  0}"` -eq "91" ];
0000070: 2074 6865 6e20 0a20 2065 6368 6f20 2220   then .  echo "
0000080: 2020 2020 2020 2020 285f 5f29 5c6e 2020          (__)\n
0000090: 2020 2020 2020 2028 6f6f 295c 6e20 202f         (oo)\n  /
00000a0: 2d2d 2d2d 2d2d 2d5c 5c2f 5c6e 202f 207c  -------\\/\n / |
00000b0: 2020 2020 207c 7c5c 6e2a 2020 7c7c 2d2d       ||\n*  ||--
00000c0: 2d2d 7c7c 5c6e 2020 205e 5e20 2020 205e  --||\n   ^^    ^
00000d0: 5e22 3b0a 656c 7365 0a20 2065 6368 6f20  ^";.else.  echo
00000e0: 2248 656c 6c6f 2057 6f72 6c64 2e22 3b0a  "Hello World.";.
00000f0: 6669 0a                                  fi.

$ sh eve1.sh
         (__)
         (oo)
  /-------\/
 / |     ||
*  ||----||
   ^^    ^^
```

```
0000000: 231d 1b92 1440 09ac 984d a6d3 bce1 1049  #....@...M.....I
0000010: 7085 1218 6f78 26b9 bd37 2bac ae50 086a  p...ox&..7+..P.j
0000020: fd4b 1655 3811 bfcc ade0 b246 ba94 517e  .K.U8......F..Q~
0000030: 4536 a206 7ee2 8a9f 9a55 99a9 1c7a ede2  E6..~....U...z..
0000040: 0a0a 6966 205b 2060 6f64 202d 7420 7831  ..if [ `od -t x1
0000050: 202d 6a33 202d 4e31 202d 416e 2022 247b   -j3 -N1 -An "${
0000060: 307d 2260 202d 6571 2022 3931 2220 5d3b  0}"` -eq "91" ];
0000070: 2074 6865 6e20 0a20 2065 6368 6f20 2220   then .  echo "
0000080: 2020 2020 2020 2020 285f 5f29 5c6e 2020          (__)\n
0000090: 2020 2020 2020 2028 6f6f 295c 6e20 202f         (oo)\n  /
00000a0: 2d2d 2d2d 2d2d 2d5c 5c2f 5c6e 202f 207c  -------\\/\n / |
00000b0: 2020 2020 207c 7c5c 6e2a 2020 7c7c 2d2d       ||\n*  ||--
00000c0: 2d2d 7c7c 5c6e 2020 205e 5e20 2020 205e  --||\n   ^^    ^
00000d0: 5e22 3b0a 656c 7365 0a20 2065 6368 6f20  ^";.else.  echo
00000e0: 2248 656c 6c6f 2057 6f72 6c64 2e22 3b0a  "Hello World.";.
00000f0: 6669 0a                                  fi.

$ sh eve2.sh
Hello World.
```

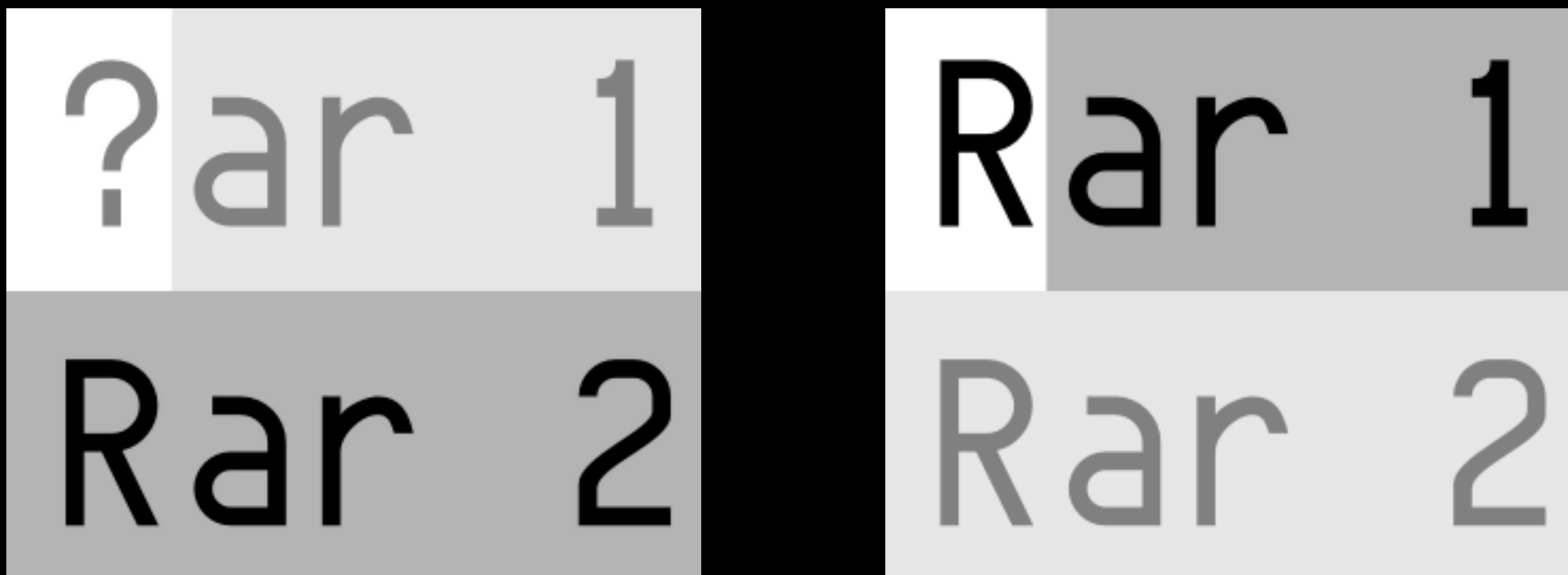
RAR/7z

scanned forward

≥ 4-byte signature :-(

but signature can start at **any offset :-D**

⇒ payload = 2 concatenated archives



killing the 1st signature byte disables the top archive

COM/MBR?

COM/MBR

(DOS executable/Master Boot Record)

no signature!

start with x86 (16 bits) code at offset 0

like shell scripts, skip initial garbage

JMP to distinct addr rather than comments

```
JMP address1          //block 1 start
........
JMP address2          //block 2 start
________

address1:             //common payload
      <payload1>
address2:
      <payload2>
```

JPEG?

JPEG

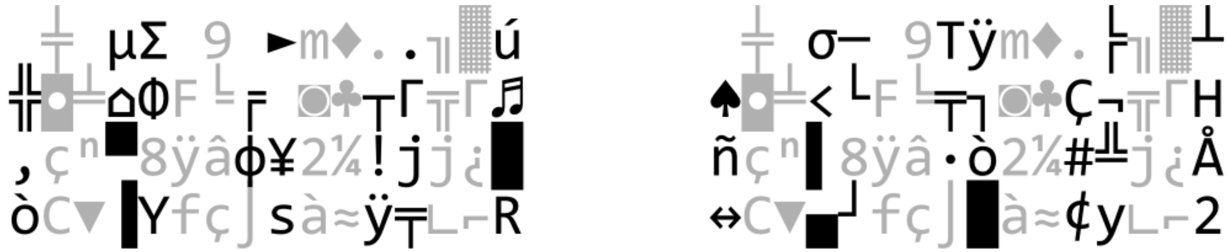
2-byte signature 0xFFD8

sequence of **chunks**

idea

message 1: first chunk “commented”

message 2: first chunk processed



JPEG signature

Chunk marker
- ff e5 in block 1
- ff e6 in block 2

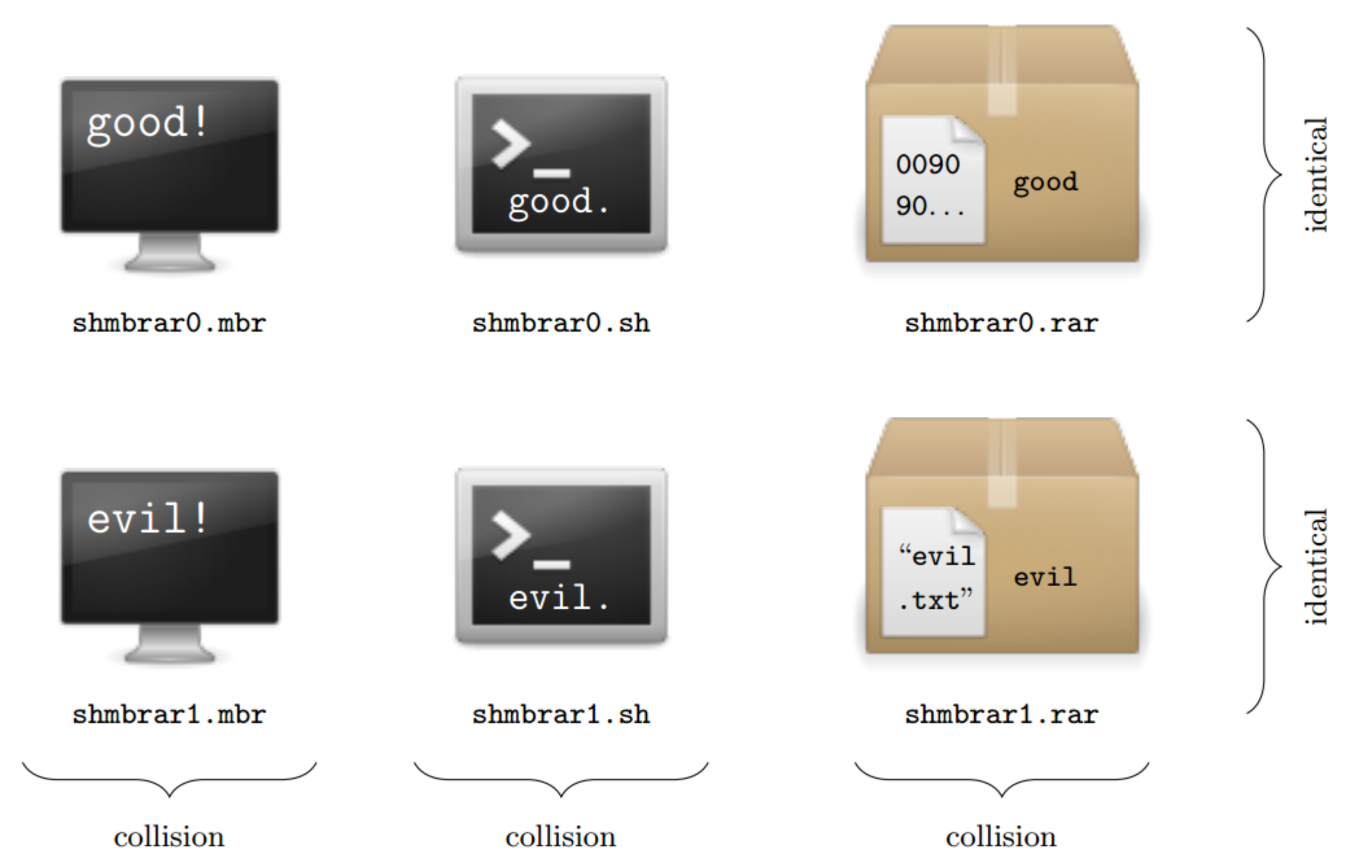
Chunk length
- c4 00 in block 1
- e4 00 in block 2

```
00000: ff d8 ff e? ?4 00 39 54 ?? 6d 04 2e ?? b7 b2 ??
       ?? 08 cf ?? ?? 46 d4 ?? ?? 0a 05 ?? ?? cb e2 ??   (contains no 0xff)
       ?? 87 fc ?? 38 98 83 ?? ?? 32 ac ?? ?? 6a a8 ??
       ?? 43 1f ?? ?? 66 87 f5 ?? 85 f7 ?? ?? 1c a9 ??

0c404: ff fe b5 e9     <COMment chunk covering Image 1>

0e404: ff e0           <start of Image 1>

       ...
       ff d9           <end of Image 1> <end of comment>

179ed: ff e0           <start of Image 2>

1b0d7: ff d9           <end of Image 2>
```


```
>crypto_hash *
test0.jpg 13990732b0d16c3e112f2356bd3d0dad1....
test1.jpg 13990732b0d16c3e112f2356bd3d0dad1....
```

polyglots

2 distinct files, 3 valid file formats!



~virtual multicollisions

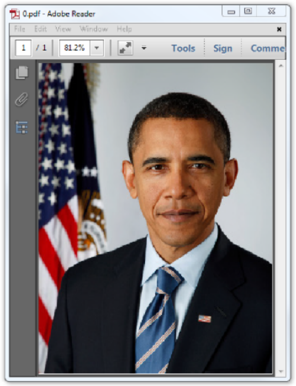
more magic: just 2 files here

Schizophrenics

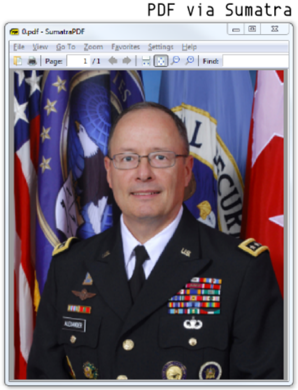
different contents with different tools

(both files)

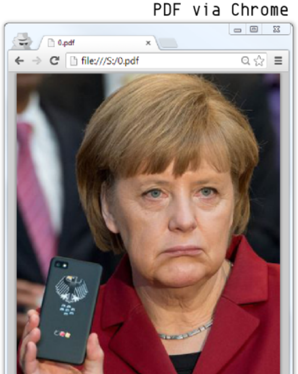
PDF via Adobe



PDF via Sumatra



PDF via Chrome



Fraternal twins

hash collision

SHA-1 with modified K* constants

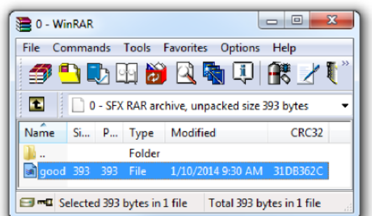
```
> m_sha1sum.exe *
10382a6d3c949408d7cafaaf6d110a9e23230416 *0
10382a6d3c949408d7cafaaf6d110a9e23230416 *1
```

0

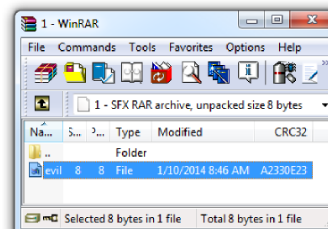
1

Polyglots

multiple file formats



RAR



Booting from Floppy...
good!

MBR

Booting from Floppy...
evil!

./0.sh
good.

shell script

./1.sh
evil.

INTERNATIONAL JOURNAL OF POC || GTFO

ISSUE 0x05

Contents

1 Call to Worship — *Rvd. Dr. Manul Laphroaig*

2 Stuff is broken, and only you know how — *Rvd. Dr. Manul Laphroaig*

3 ECB as an Electronic Coloring Book — *Philippe Teuwen*

4 An Easter Egg in PCI Express — *Jacob Torrey*

5 A Flash PDF polyglot — *Alex Inführ*

6 These Philosophers Stuff on 512 Bytes; or, This Multiprocessing OS is a Boot Sector. — *Shikhin Sethi*

7 A Breakout Board for Mini-PCIe; or, My Intel Galileo has less RAM than its Video Card! — *Joe FitzPatrick*

8 Prototyping a generic x86 backdoor in Bochs; or, I'll see your RDRAND backdoor and raise you a covert channel! — *Matilda*

9 From Protocol to PoC; or, Your Cisco blade is booting PoC||GTFO. — *Mik*

10 i386 Shellcode for Lazy Neighbors; or, I am my own NOP Sled. — *Brainsmoke*

11 Abusing JSONP with Rosetta Flash — *Michele Spagnuolo*

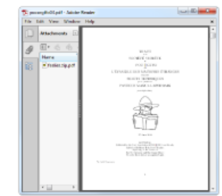
12 A cryptographer and a binarista walk into a bar — *Ange Albertini and Maria Eichlseder*

13 Ancestral Voices Or, a vision in a nightmare. — *Ben Nagy*

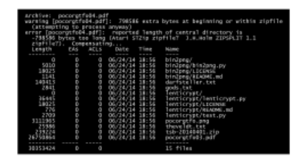
14 A Call for PoC — *Rvd. Dr. Manul Laphroaig*

PDF

WITH ZIP ATTACHMENT

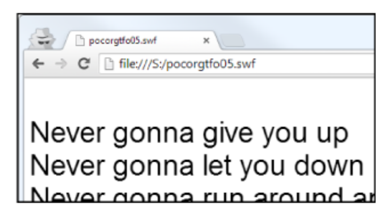


ZIP



ISO

FLASH



Conclusions

Implications for SHA-1 security?

None.

We did not improve attacks on the unmodified SHA-1.

Did NSA use this trick when designing SHA-1 in 1995?

Probably not, because

1) cryptanalysis techniques are known since ~2004

~~2) the constants look like NUMSN (√2 √3 √5 √10)~~

3) remember the SHA-0 fiasco :)

Can you do the same for SHA-256?

Not at the moment.

Good: SHA-256 uses distinct constants at each step

⇒more control to conform to the characteristic

(but also more differences with the original)

Not good: The best known attack is on 31 steps (in ~2^{65}), of 64 steps in total, so it might be difficult to find a useful 64-step characteristic

Thank YOU

questions?

maliciousshа1.github.io maliciousshа1@131002.net

TU Graz

KUDELSKI SECURITY