

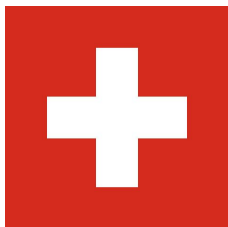
IDENTICAL PREFIX

Exploiting hash collisions

Ange Albertini

BlackAlps 2017

Switzerland



DISCLAIMERS

ALL OPINIONS EXPRESSED DURING THIS PRESENTATION
ARE MINE AND NOT ENDORSED
BY ANY OF MY EMPLOYERS, PRESENT OR PAST.

This is ***not*** a crypto talk.

It's about ***exploiting*** hash collisions,
(the weakest ones, w/ identical prefix)
via manipulating file formats.

You *may* want to watch Marc Stevens' [talk](#) at CRYPTO17.

TL;DR

Nothing
groundbreaking.
No new vulnerability.
Just a look behind the scenes of
Shattered-like research
(***format***-wise)

OTOH there are very few talks on the topic AFAIK.

THIS TALK IS ABOUT...

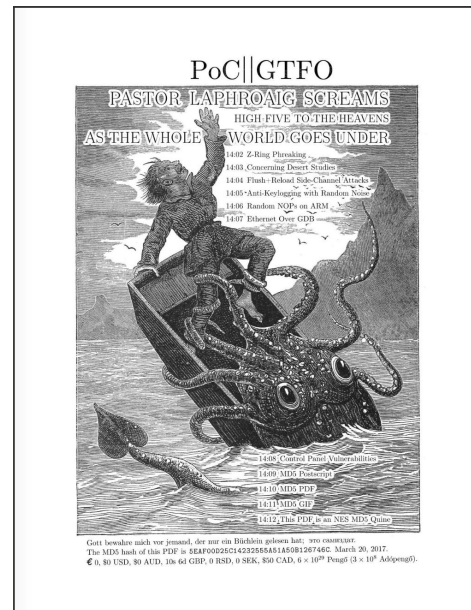
2014: ^{MALSHA1} [Malicious SHA1](#) - modified SHA1



2015-2017: [Shattered](#) - SHA1



2017: [PoC||GTFO 0x14](#) - MD5



Types of collision

- **Identical prefix**

- 2 files starting with same data

- **Chosen prefix**

- 2 files starting with different (chosen) data

- **Second preimage attack**

- Find data to match another data's hash

- **Preimage attack**

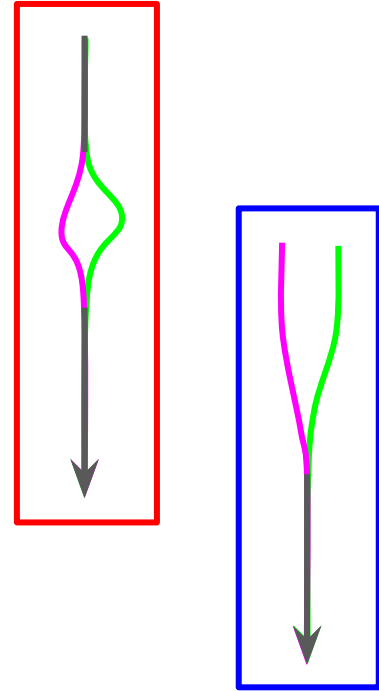
- Find data to match hash

FIRST, WEAKEST, OVERLOOKED

*SH*T'S BROKEN, YO!*

UNICORNS

DRAGONS



Formal way to present IPCs

[Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD.](#)

X Wang, D Feng, X Lai, H Yu
2004

X_1	M_1	313838dd fc2932c7 c030b717 bafc1bae 6673a8d7 9ddcf416 85d70859 99403db0 634add1 c0736004 9558bd1f 21c10982 ca94c90b 6aa6e69 cbf61bf1 6b0e615
	M_{11}	2e82d48b 16bdf161 ce10bd62 c3c6809d b6745639 fc0e06c7 6573a914 beff0d753 537b8755 497b92e8 46f559e2 7d7a347a 511d8b1 98eb6b68 c9ca4559 eb10e037
	M_1'	313838dd fc2932c7 c030b717 bafc1bae 6673a8d7 9ddcf416 85d70859 99403db0 634add1 c0736004 9558bd1f 21c10982 ca94c90b 6aa6e69 4bf61bf1 6b0e615
X_1'	M_{11}'	2e82d48b 16bdf161 ce10bd62 c3c6809d 36745639 fc0e06c7 6573a914 beff0d753 537b8755 497b92e8 46f559e2 7d79b47a 511d8b1 98eb6b68 49ca4559 eb10e037
	H	21f15d09 3efb11d2 f9f09bfb 86b9cadf
	X_2	313838dd fc2932c7 c030b717 bafc1bae 6673a8d7 9ddcf416 85d70859 99403db0 634add1 c0736004 9558bd1f 21c10982 ca94c90b 6aa6e69 cbf61bf1 6b0e615
X_2'	M_{12}	2882d409 177df16c bf90fde1 c406a19a b43a36af fd41f967 2835450e a12506ce 2973087d 8839e1a0 78646612 9c8dac6d efs9b8e7 4840474 2afb5bd0 840c546a
	M_{12}'	313838dd fc2932c7 c030b717 bafc1bae 6673a8d7 9ddcf416 85d70859 99403db0 634add1 c0736004 9558bd1f 21c10982 ca94c90b 6aa6e69 4bf61bf1 6b0e615
	H	fa8892f3 49c2111f 477d3217 56ae4e97

Table 1 Two pairs of collision for MD5

2 Collisions for HAVAL-128

HAVAL is proposed in [10]. HAVAL is a hashing algorithm that can compress messages of any length in 3, 4 or 5 passes and produce a variable length output --128-bit, 160-bit, 192 or 224-bit fingerprint.

Attack on a reduced version for HAVAL was given by P. R. Kasselmann and W. T. Penzhorn [7], which consists of last rounds for HAVAL-128. We break the full HAVAL-128 with only about the 2^6 HAVAL computations. Here we give two examples of collisions of HAVAL-128, where

$$M' = M + \Delta C, \Delta C = (2^{-i-1}, 0, 0, 0, 2^{-i-2}, \dots, 2^{-i-8}, 0, \dots, 0), s = 0, 1, 1, 1, 8$$

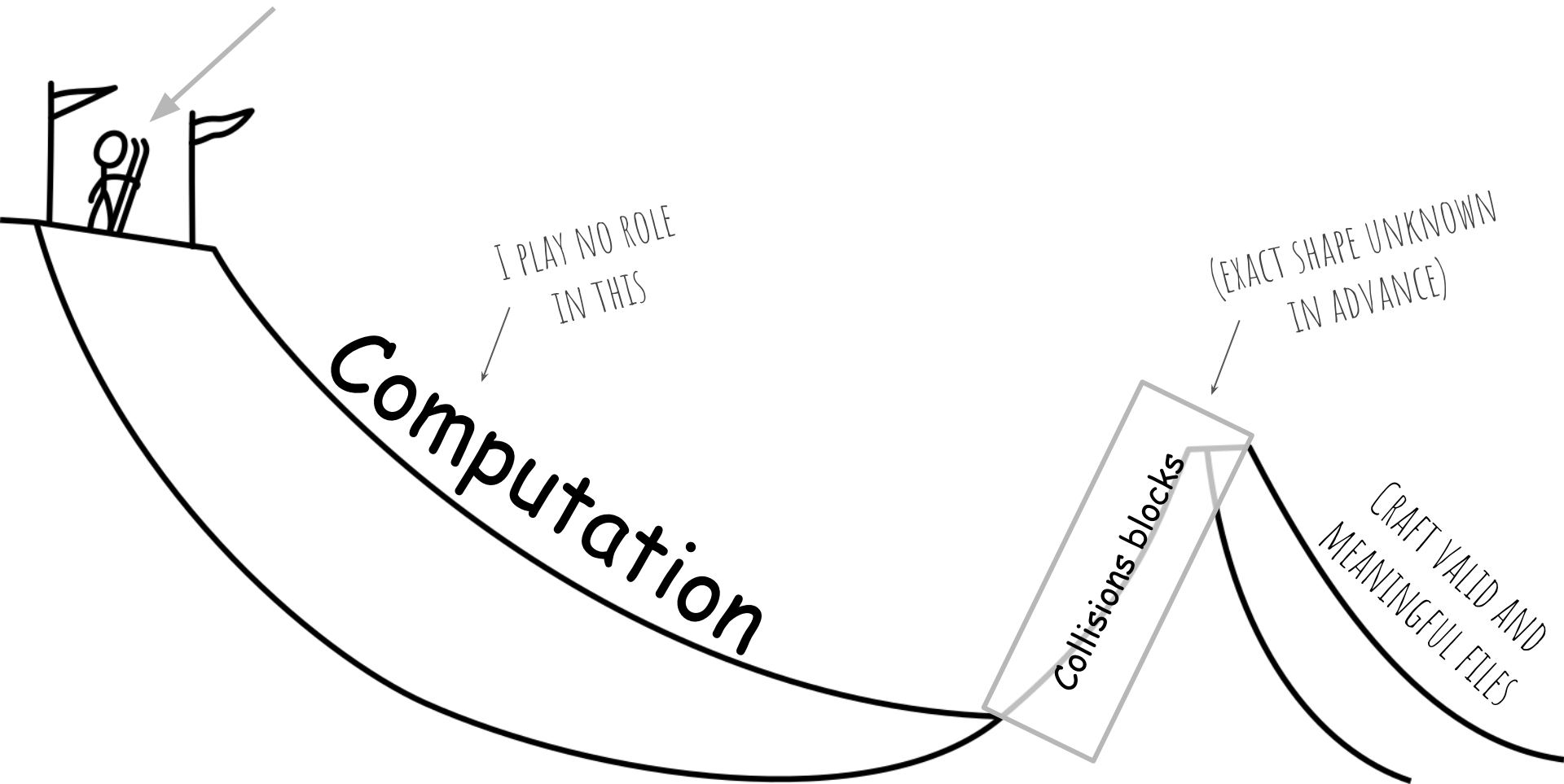
$$i = 0, 1, 2, \dots, 31$$

$$HAVAL(M) = HAVAL(M')$$

M_1	6377448b d9e59f18 f2aa3cbb d6cb92ba ee544a44 879fa576 1ca34633 76ca5446
	a67a8a42 8d3adc8b b6e3d814 5630998d 86ea5ded a739ac7b 54fd8e32 85e2eb36
	38183c9a b67a9289 c47299b2 27039ce5 dd555e14 839018d8 aabb09 478f6032
	ff4b3a7 40000096 71466aac ffffbcd 5f4016d2 5f4016d0 5f4016d0 5f4016d0

NOT VERY "VISUAL"!

DETERMINE FILE STRUCTURE



Impact

Better than random-looking blocks?
Will it convince anyone to deprecate anything?

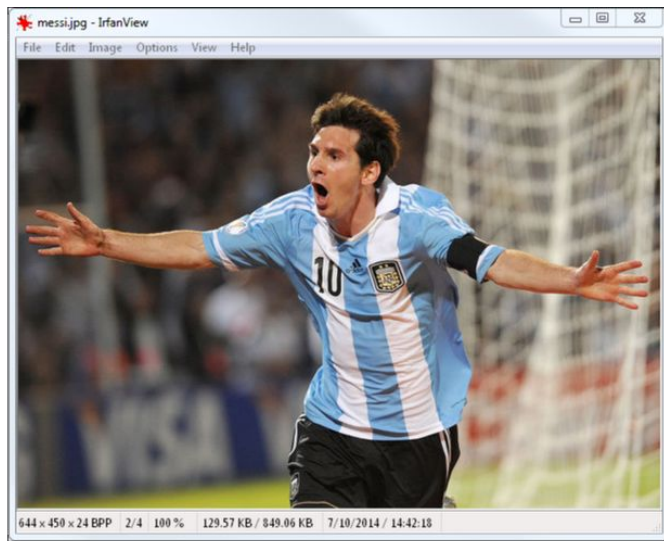
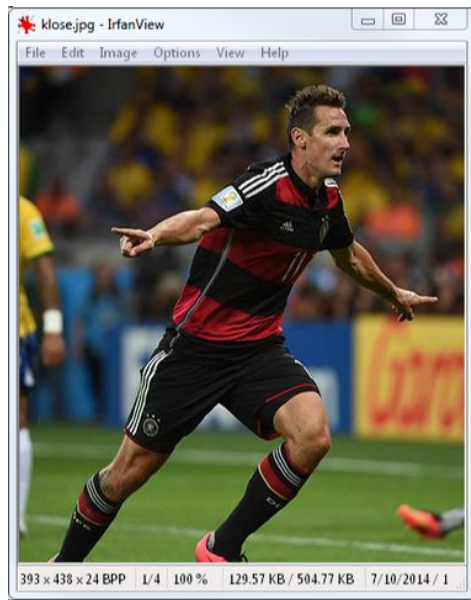
FTR Shattered took 6500 CPU-Yr
and 110 GPU-Yr.

(that's a lot of computing power)

INFINITE

Re-usability: Moar impact

THESE ARE MALSHAI
EXAMPLES.



2004: Dan Kaminsky: MD5 To Be Considered Harmful Someday

<https://eprint.iacr.org/2004/357.pdf>

<https://dankaminsky.com/2004/12/06/46/>

2004: Ondrej Mkle: Practical Attacks on Digital Signatures Using MD5 Message Digest

<https://eprint.iacr.org/2004/356.pdf>

Slides [a6cb4934945457d16bc90ef9ab3c391474fb78cf844c59f34d4505b95fbad5ea](#)

Paper [ac7a05b4bf456b4358e8a754f5f70612ce593bca1cdb718c2b38e3e280fc1240](#)

IPC exploits papers

- 2005

Max Gebhardt, Georg Illies, Werner Schindler

A Note on the Practical Value of Single Hash Collisions for Special File Formats

- 2014 [MaISHA1](#)

Malicious Hashing: Eve's Variant of SHA-1

Ange Albertini, Jean-Philippe Aumasson, **Maria Eichlseder**, Florian Mendel, Martin Schläffer

Jean-Philippe's [Slides aba7833ed35eb5b44b44377f7054c7318637a8cb5db002c1ac787a5d2314f658](#)

[Paper 5c763e295b95ee8c69fd9430eae62fa59d7c9716ada645a93dcc19387e3d6821](#)

- 2017 [Shattered](#)

The first collision for full SHA-1

Marc Stevens, Elie Bursztein, Pierre Karpman, **Ange Albertini**, Yarik Markov

[Paper a3396362dcc528ed29918c07701e3b5082365a1dc19a9aac8d104c9c3d07c6b2](#)

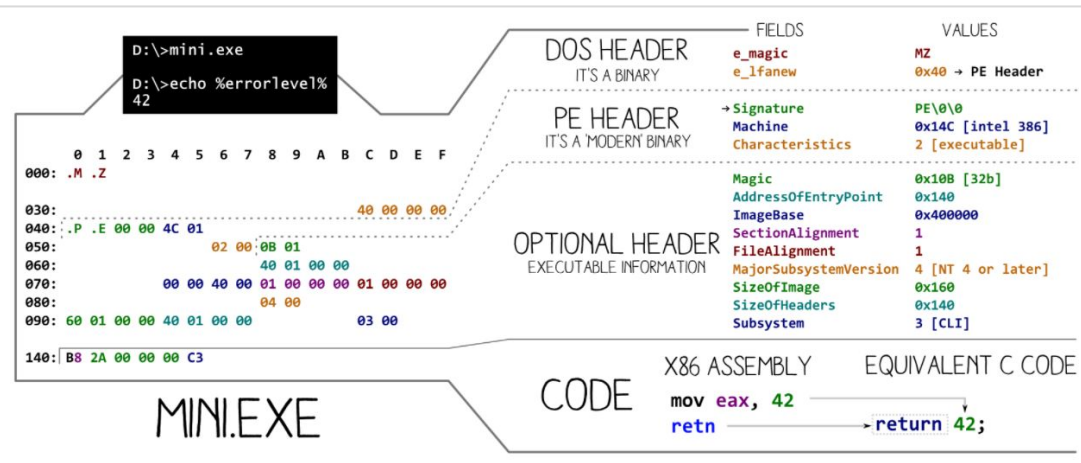
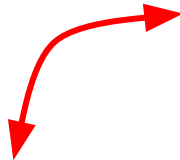
Marc's [Crypto17 video](#)
Elie's [BlackHat Slides 1a17c315a946409e8ef37c56c962987d41377374c15ac0d855e92297b4f03596](#)

- 2017 [PoC||GTFO 0x14](#)

Greg, **spq**, **Mako**, **Philippe**, **Evan**², **Ange**, **Melissa Elliott**

file format collaborator instigator

Constraints of hash and formats have nothing in common



```
// change q17 until conditions are met on q18, q19 and q20  
unsigned counter = 0;  
while (counter < (1 << 7))  
{  
    const uint32 q16 = Q[Qoff + 16];  
    uint32 q17 = ((xrng64() & 0x3ffd7fff) | (q16&0xc0008008)) ^ 0x40000000;  
    ++counter;  
  
    uint32 q18 = GG(q17, q16, Q[Qoff + 15]) + tt18;  
    q18 = RL(q18, 9); q18 += q17;  
    if (0x00020000 != ((q18^q17)&0xa0020000))  
        continue;  
  
    uint32 q19 = GG(q18, q17, q16) + tt19;  
    q19 = RL(q19, 14); q19 += q18;  
    if (0x80000000 != (q19 & 0x80020000))  
        continue;  
  
    uint32 q20 = GG(q19, q18, q17) + tt20;  
    q20 = RL(q20, 20); q20 += q19;  
    if (0x00040000 != ((q20^q19) & 0x80040000))  
        continue;  
  
    block[1] = q17-q16; block[1] = RR(block[1], 5); block[1] -= tt17;  
    uint32 q2 = block[1] + tt1; q2 = RL(q2, 12); q2 += Q[Qoff + 1];  
    block[5] = tt5 - q2;  
  
    Q[Qoff + 2] = q2;  
    Q[Qoff + 17] = q17;  
    Q[Qoff + 18] = q18;  
    Q[Qoff + 19] = q19;  
    Q[Qoff + 20] = q20;  
    MD5_REVERSE_STEP(2, 0x242070db, 17);  
  
    counter = 0;  
    break;  
}
```


File constraints

- Collision blocks are very complex
⇒ considered **random**
- Collision blocks only differ by a mask.
 - The mask may be fixed in advance.
- Collision blocks may contain arbitrary values
 - Or bruteforce them.

⇒ craft your files with random blocks
and apply mask

Prefix?	=	Prefix?
Block A	<>	Block B
Suffix	=	Suffix

THESE ARE SHATTERED
EXAMPLES.

Where the magic happens: random stuff + mask

File A

Collision blocks

File B

```
7F 46 DC 93-A6 B6 7E 01-3B 02 9A AA-1D B2 56 0B 0FÜ“‘‘‘~ ; ša 2V
45 CA 67 D6-88 C7 F8 4B-8C 4C 79 1F-E0 2B 3D F6 EÊgÖ~ÇøKÖLyà+=ö
14 F8 6D B1-69 09 01 C5-6B 45 C1 53-0A FE DF B7 øm±i ÅkEÁS pß·
60 38 E9 72-72 2F E7 AD-72 8F 0E 49-04 E0 46 C2 `8érr/ç r I àFÂ
30 57 0F E9-D4 13 98 AB-E1 2E F5 BC-94 2B E3 35 0W éO ~«á.õ%”+ã5
42 A4 80 2D-98 B5 D7 0F-2A 33 2E C3-7F AC 35 14 Bx€-~µx *3.Ã-5
E7 4D DC 0F-2C C1 A8 74-CD 0C 78 30-5A 21 56 64 çMÜ ,Á”tÍ x0Z!Vd
61 30 97 89-60 6B D0 BF-3F 98 CD A8-04 46 29 A1 a0-%`kD¿?~Í“F)j
```

```
73 46 DC 91-66 B6 7E 11-8F 02 9A B6-21 B2 56 0F sFÜ“fç~ 0 šg!2V
F9 CA 67 CC-A8 C7 F8 5B-A8 4C 79 03-0C 2B 3D E2 ùÊgI~Çø[“Ly +=â
18 F8 6D B3-A9 09 01 D5-DF 45 C1 4F-26 FE DF B3 øm³Θ ÖBEA0&pß³
DC 38 E9 6A-C2 2F E7 BD-72 8F 0E 45-BC E0 46 D2 Ü8éjA/çr I E%àF
3C 57 0F EB-14 13 98 B3-55 2E F5 A0-A8 2B E3 31 <W éU.õ +ã1
FE A4 80 37-B8 B5 D7 1F-0E 33 2E DF-93 AC 35 00 ppxE7.µx
EB 4D DC 0D-EC C1 A8 64-79 0C 78 2C-76 21 56 60 çMÜ iA
DD 30 97 91-D0 6B D0 AF-3F 98 CD A4-BC 46 29 B1 Y0-“DkD”~Ih...±
```

THAT'S A BIG PILE OF
RANDOMNESS :)

```
0c 00 00 02 c0 00 00 10 b4 00 00 1c 3c 00 00 04
bc 00 00 1a 20 00 00 10 24 00 00 1c ec 00 00 14
0c 00 00 02 c0 00 00 10 b4 00 00 1c 2c 00 00 04
bc 00 00 18 b0 00 00 10 00 00 00 0c b8 00 00 10
```

⇒ generate one file from the other.

xor mask

..
X.
 X.
 X.

INSTANT, BUT VERY RESTRICTIVE
 → BRUTEFORCE

FastColl: MD5, ~1s

.XX X. X. XX XX XXX
 XX XX X. X. XX XX XX XX
 .XX X. X. XX XX XXX
 XX XX X. X.X XX X.

VERY EXPENSIVE,
 BUT TRIVIAL TO EXPLOIT

JUMP

Stevens13: SHA1, 6610 Yr

Prefix and masks determine how easily it's exploitable.


```

2D 20 42 6C 61 63 6B 41 6C 70 73 27 31 37 20 2D - BlackAlps'17 -
CA 99 ED 4A 7A 59 10 F6 6C 10 5B 71 B0 80 65 5D ...JzY..l.[q..e]
87 07 94 73 71 1F 07 B2 B5 84 12 96 BD 1D 03 2C ...sq.....,
E7 09 25 96 6E 0B 02 FD 96 9A 54 32 EB 15 FC F1 ..%.n.....T2...
D7 DF 52 10 C4 35 29 0A 5B 9A 93 40 34 5C 35 4C ..R..5).[..@4\5L
D7 AA 9E 83 16 F3 8C 61 E0 44 5C F0 4C DE F7 1C .....a.D\L...
16 D1 F7 49 B4 D4 EE 9E 65 D5 B6 7F B6 31 27 1E ...I....e....1'.
8B 0A F7 3D E7 42 B5 64 BC 1E 2A 97 64 EA F7 F2 ...=.B.d...*.d...

```

```

2D 20 42 6C 61 63 6B 41 6C 71 73 27 31 37 20 2D - BlackAlps'17 -
CA 99 ED 4A 7A 59 10 F6 6C 10 5B 71 B0 80 65 5D ...JzY..l.[q..e]
87 07 94 73 71 1F 07 B2 B5 84 12 96 BD 1D 03 2C ...sq.....,
E7 09 25 96 6E 0B 02 FD 96 9A 54 32 EB 15 FC F1 ..%.n.....T2...
D7 DF 52 10 C4 35 29 0A 5B 99 93 40 34 5C 35 4C ..R..5).[..@4\5L
D7 AA 9E 83 16 F3 8C 61 E0 44 5C F0 4C DE F7 1C .....a.D\L...
16 D1 F7 49 B4 D4 EE 9E 65 D5 B6 7F B6 31 27 1E ...I....e....1'.
8B 0A F7 3D E7 42 B5 64 BC 1E 2A 97 64 EA F7 F2 ...=.B.d...*.d...

```

Same hash,
different masks.

2 MD5 COLLISIONS
FROM HASHCLASH (2 MIN)
WITH DIFFERENT MASKS.

```

2D 20 42 6C 61 63 6B 41 6C 70 73 27 31 37 20 2D - BlackAlps'17 -
01 4D 80 6F 5B CB C0 AE 3D 33 52 BD EA 0B 01 93 .M.o[...=3R.....
5A 58 58 DB 51 B3 32 B4 F6 17 99 75 62 B8 D3 BD ZXX.Q.2....ub...
58 A3 EE A3 7C 22 0D 08 56 7F 4A D6 EF 58 C9 1F X...|"..V.J..X..
24 60 25 9F 4A E9 FC F5 55 67 B7 A9 E3 54 C5 72 $`%.J...Ug...T.r
0A A8 05 D6 6C 79 21 85 0A 75 38 59 C6 D9 01 51 ....ly!...u8Y...Q
BD C3 19 F5 32 F5 EC 99 15 AC 91 9F CF BE BD CE ....2.....
E1 2B 75 20 CB D9 76 FD F6 96 5B 89 3E 8B 10 E0 .+u ..v...[.>...

```

```

2D 20 42 EC 61 63 6B 41 6C 70 73 27 31 37 20 2D - B.ackAlps'17 -
01 4D 80 6F 5B CB C0 AE 3D 33 52 3D EA 0B 01 93 .M.o[...=3R=....
5A 58 58 DB 51 B3 32 B4 F6 17 99 75 62 B8 D3 BD ZXX.Q.2....ub...
58 A3 EE A3 7C 22 0D 10 56 7F 4A D6 EF 58 C9 1F X...|"..V.J..X..
24 60 25 1F 4A E9 FC F5 55 67 B7 A9 E3 54 C5 72 $`%.J...Ug...T.r
0A A8 05 D6 6C 79 21 85 0A 75 38 D9 C6 D9 01 51 ....ly!...u8....Q
BD C3 19 F5 32 F5 EC 99 15 AC 91 9F CF BE BD CE ....2.....
E1 2B 75 20 CB D9 76 F5 F6 96 5B 89 3E 8B 10 E0 .+u ..v...[.>...

```


IPC exploits
strategies

WORKS WITH
MANY SCRIPT LANGUAGES

If-then-else (data)

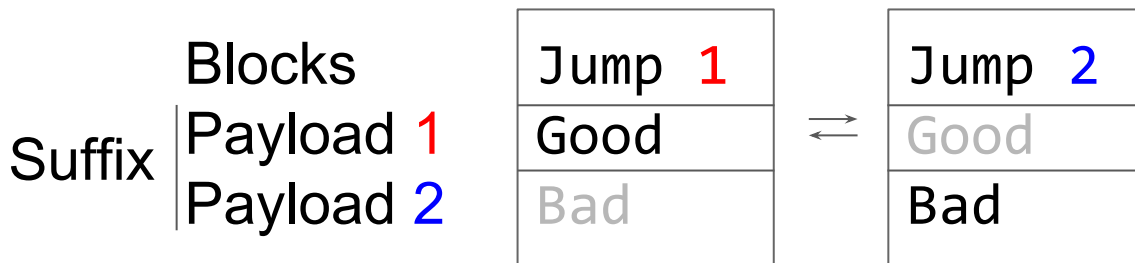
- Get collision block ignored (commented out)
- File suffix/separate executable contains code
 - Checks the block values
or uses block as decryption key.

⇒ Collision block == passive data

Collision blocks (commented out)
Code (checking block values)

Code

- Prefix or bruteforcing sets up some opcodes
- 2 target addresses in the collision blocks
- 2 code snippets in suffix



ONLY NEEDS FEW BYTES
X86 JUMP = EB XX,
BUT NO REAL-LIFE CONSEQUENCES :(

Format (structure)

- Prefix or bruteforcing sets up a header
- Collision blocks alter a value,
To make parsers ignore the rest of the blocks
and land at different offsets.

See [MD5 rogue certificates](#) w/ chosen-prefix.

Prefix (declares a header)
Collision blocks (changes header value)
Data (contains 2 data sets)

Concatenation

With a top-down file format that can start at any offset (Rar, 7z...)

(ZIP IS BOTTOM-UP)

1. Collision blocks end with signature's start.

- w/ a difference on that byte.

2. Append a file minus its first byte.

3. Append another file of the same type.

ONE LETTER IS ENOUGH

Coll. Blocks

RAR File 1

RAR File 2

.. .. . R
ar!<file>
Rar!<file>



.. .. . ?
ar!<file>
Rar!<file>

GENERAL GOAL

(MEANINGFUL)

Find a way to get 2 files despite the randomness.

Prefix.

WRITE YOUR PREFIX

Randomness.

INSERT TOTALLY RANDOM DATA

Collision block masks.

APPLY MASK

QA

TEST FILES,
ON ALL TOOLS.

Format target

- Something universally used.
 - Preferably multi-platform \Rightarrow ~~executables~~
 - By end-users, not just developers.
 - Preferably, something with crypto!
(certificates are pretty restrictive)
- With as fewer parsers in the wild as possible.

Visual documents: JPEG, PNG, GIF, PDF...

CHALLENGES

Validity.
Compatibility.
Correct rendering.
Re-useability.

EVER DANCE WITH THE SPECS
BY THE PALE MOONLIGHT?

CORNER CASES FTW

TEST, TEST, TEST!

EXPLORE ALL CODE PATHS,
ALL HEADERS VALUES

2005: Gebhardt et al.

- If-then-else exploits

- PostScript

- PDF

- TIFF

- Word 97



Word97 macro

```
Sub collision()  
Dim b(512) As Byte  
FName$ = ActiveDocument.Name  
  
Open FName$ For Binary Access Read As #1 Len = 512  
Get #1, , b      'the price 1000$ is contained in 2nd line of  
Close #1         'the .doc file; that line is selected by  
                 'the Selection .. Count:=2 command  
  
If b(147) >= 128 Then  
    Selection.Collapse Direction:=wdCollapseStart  
    Selection.GoTo What:=wdGoToLine, Which:=wdGoToAbsolute, Count:=2  
    Selection.MoveRight Unit:=wdCharacter, Count:=1  
    Selection.Find.ClearFormatting  
    With Selection.Find  
        .Text = '$'  
        .Forward = True  
        .Wrap = wdFindContinue  
        .Format = False  
        .MatchWholeWord = False  
        .MatchWildcards = False  
        .MatchSoundsLike = False  
        .MatchAllWordForms = False  
    End With  
    Selection.Find.Execute  
    Selection.MoveLeft Unit:=wdCharacter, Count:=3  
    Selection.MoveRight Unit:=wdCharacter, Extend:=wdCharacter  
    Selection.Font.ColorIndex = wdWhite  
    Selection.GoTo What:=wdGoToLine, Which:=wdGoToAbsolute, Count:=1  
    Selection.Collapse Direction:=wdCollapseEnd  
End If          'by the Selection .. Count:=1 command  
               'the cursor returns to the first character  
               'in the text (disguise of attack)  
  
End Sub
```


PDF features and landscape

No widespread scripting language in PDF:

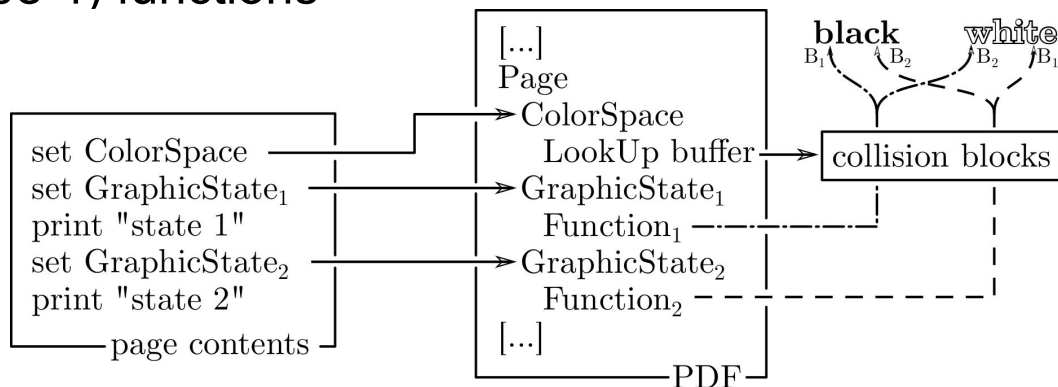
- JavaScript/FormCalc reliably only in Adobe Reader

Only binary-based conditional function:

- PostScript Calculator (Type 4) functions

```
<<  
  /FunctionType 4  
  /Domain [0.0 1.0]  
  /Range [0.0 1.0]  
  /Length 28  
>>  
stream  
{255 mul 121 sub 1 exch sub}  
endstream
```

depends on the collision block



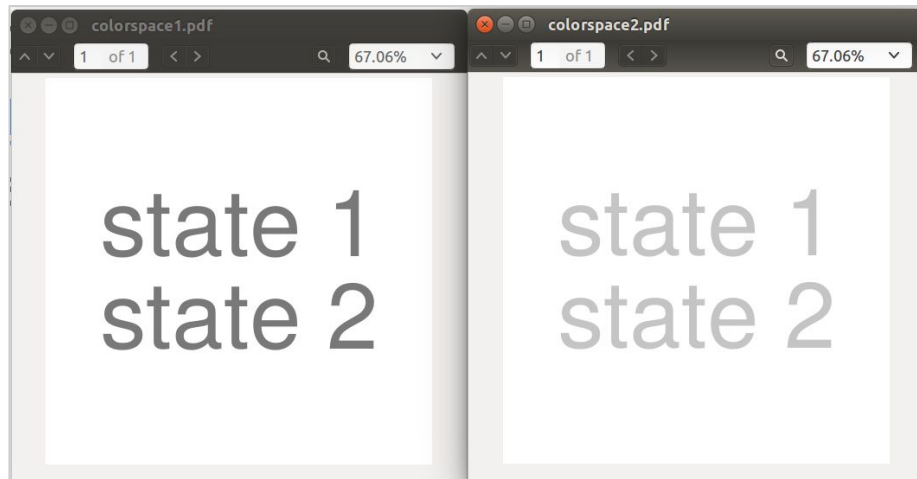
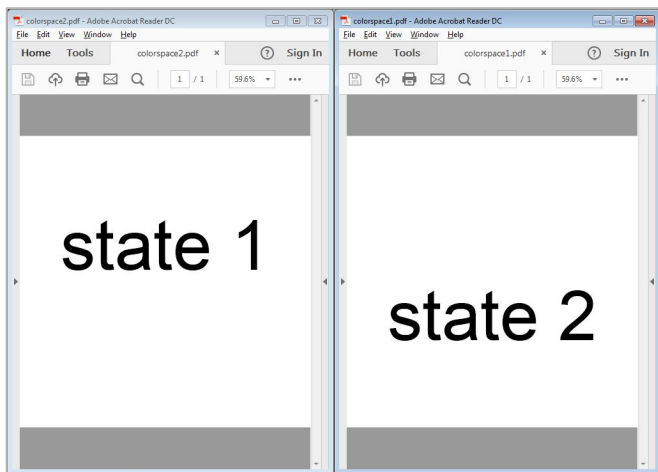
Not good enough

REJECTED

- Poorly supported across readers.
- Limited to 2 non-overlapping objects

ONLY OK IN ADOBE
NO FULL CONTROL

⇒ reliable but limited for payload and compatibility



2014: MaISHA1

- Very restrictive: **no prefix !!!** \Rightarrow very simple collisions
- 30-50h on 80 cores:

Many retries are possible, but unclear collision mask.

- If then else: Shell script
 - Concatenation: RAR, 7z
 - Code: Master Boot Record
 - Format: JPEG
-
- Polyglot: all in the same file!

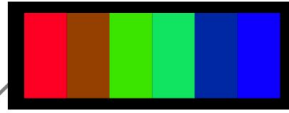
```
#'4@  ØM|ÓTá+,...[Gx&%ý7+iæP,uKw8¿Ø¥à²D”Q*í6¢pāŠŸ2U™ª´zí,  
  
if [ `od -t x1 -j3 -N1 -An "${0}"` -eq "91" ]; then  
    echo "          (__)\\n          (oo)\\n /-----\\n / |      ||\\n*  ||----||\\n  ^^  
    ^^";  
Else  
    echo "Hello World.";  
fi
```




MalSHA1 failures

- Can't control 4 bytes in a row.
⇒ many file formats aren't useable
- Windows Executable? (magic = "MZ")
Would end up with huge e_lfanew (a header offset, not a memory pointer)
Max value in practice: 0x9000000 (150 Mb)





	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000:	FF	D8	FF	E0	00	10	.J	.F	.I	.F	00	01	01	01	00	48
010:	00	48	00	00	FF	DB	00	43	00	01	01	01	01	01	01	01
020:	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
030:	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
040:	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
050:	01	01	01	01	01	01	01	01	01	FF	DB	00	43	01	01	01
060:	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
070:	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
080:	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
090:	01	01	01	01	01	01	01	01	01	01	01	01	01	01	FF	C0
0A0:	00	11	08	00	02	00	06	03	01	22	00	02	11	01	03	11
0B0:	01	FF	C4	00	15	00	01	01	00	00	00	00	00	00	00	00
0C0:	00	00	00	00	00	00	00	09	FF	C4	00	19	10	01	00	02
0D0:	03	00	00	00	00	00	00	00	00	00	00	00	00	00	06	08
0E0:	38	88	B6	FF	C4	00	15	01	01	01	00	00	00	00	00	00
0F0:	00	00	00	00	00	00	00	00	07	0A	FF	C4	00	1C	11	00
100:	01	03	05	00	00	00	00	00	00	00	00	00	00	00	00	08
110:	00	07	B8	09	38	39	76	78	FF	DA	00	0C	03	01	00	02
120:	11	03	11	00	3F	00	86	F7	E7	1D	A9	16	CA	77	30	D0
130:	14	F7	41	DC	5A	8E	FB	31	19	26	5D	C4	2A	F4	5C	81
140:	7B	DB	06	84	A0	75	17	FF	D9							

SEGMENTS	FIELDS	VALUES
START OF IMAGE	marker	FFD8
APPLICATION (DEFAULT HEADER)	marker/length	FFE0/16
	identifier	JFIF\0
	version	1.1
	units	1 (dpi)
	density	72x72
QUANTIZATION TABLE	marker/length	FFD0/67
	destination	0 (luminance)
	table (8x8)	{1} (100% quality)
	marker/length	FFD1/67
	destination	1 (chrominance)
QUANTIZATION TABLE	marker/length	FFD2/67
	destination	1 (chrominance)
	table (8x8)	{1} (100% quality)
	marker/length	FFC0/17
	precision	8
START OF FRAME	line Nb	2
	samples/line	6
	components	3
	Id factor table	1 1x1 0 (LumY)
	Id factor table	2 2x2 1 (ChromCb)
HUFFMAN TABLE	marker/length	FFC4/21
	class	0 (DC)
	destination	0
	1 code of 1 bit	00
	1 code of 2 bits	01
HUFFMAN TABLE	marker/length	FFC4/25
	class	0 (DC)
	destination	0
	1 code of 1 bit	00
	2 code of 3 bits	00 00
HUFFMAN TABLE	marker/length	FFC4/21
	class	0 (DC)
	destination	1
	1 code of 1 bit	01
	1 code of 2 bits	00
HUFFMAN TABLE	marker/length	FFC4/28
	class	1 (AC)
	destination	1
	1 code of 2 bits	00
	3 code of 3 bits	00 07 88
START OF SCAN	marker/length	FFDA/12
	components	3
	selector / DC, AC table	
		2 / 1, 1
		3 / 1, 1
IMAGE DATA	spectral select.	0...63
	successive approx.	00
	86F7E71DA916CA77380014	
	F741DC5ABEFB3119265DC4	
	2AF45C817BDB0684A07517	
END OF IMAGE	marker	FFD9

A primer on JPEG

signature: FF D8 *VERY SHORT*

Segments structure:

all start with FF 00

(FF in data always followed by 00)

Garbage? Skip until next FF!

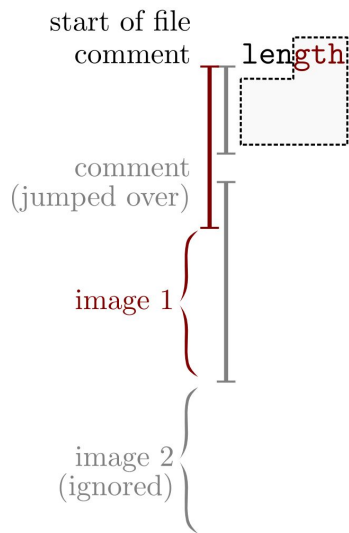
Big endian lengths, on 2 bytes. *VERY "TOLERANT"*
*NEVER TOO BIG,
 NEVER TOO SMALL.*

2 images, 1 "comment"

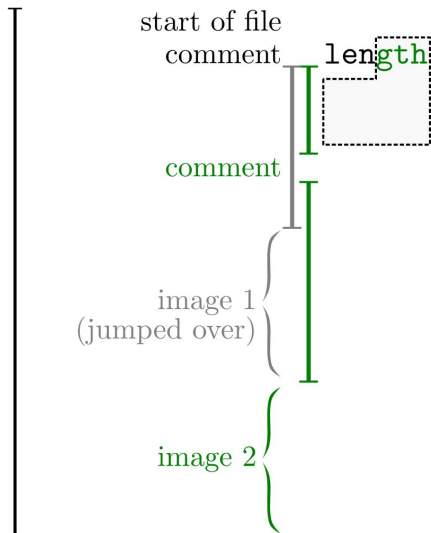
A comment (an ignored segment),
of variable length.

Use another comment to
Jump over the first image.

make sure not to jump in the blocks:
⇒ 01 xx is optimal.

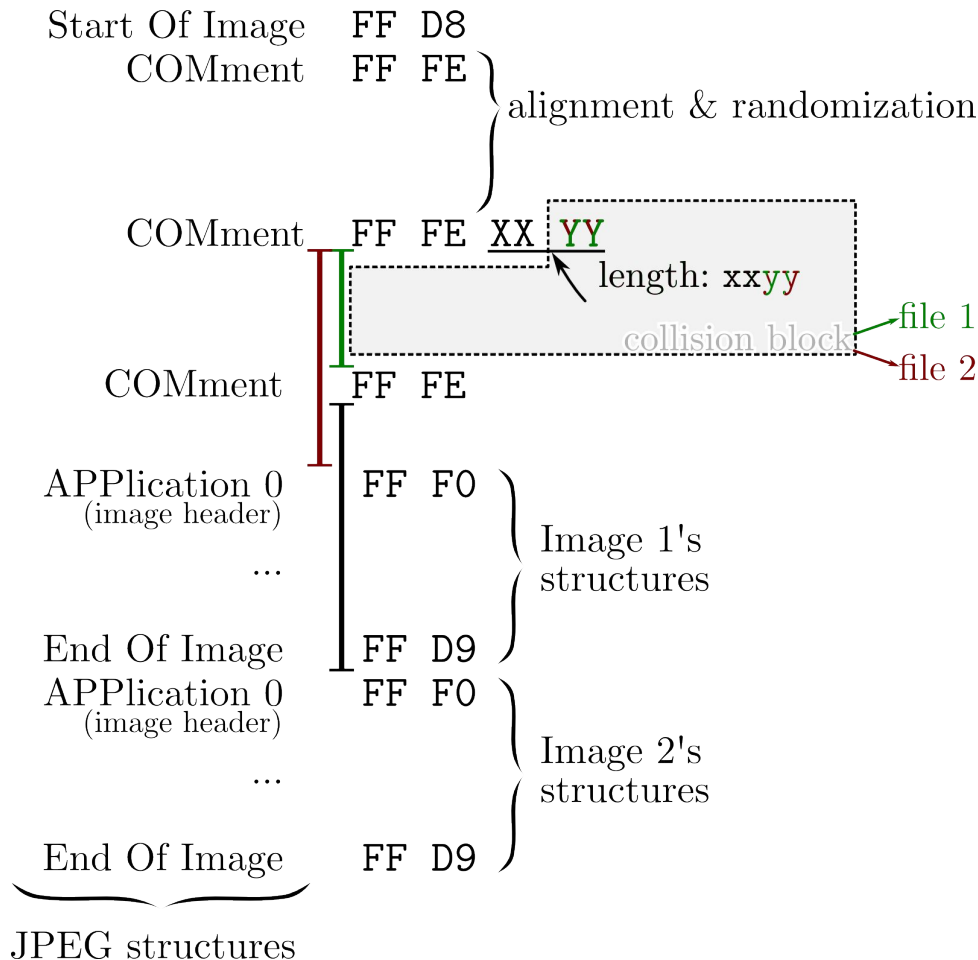


File 1



File 2

JPEG collision structure



Abusing JPEG tolerance



JPEG signature

Chunk marker

Chunk length

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

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

ff	d8	ff	e?	?4	00	39	54	??	6d	04	2e	??	b7	b2	??
??	08	cf	??	??	46	d4	??	??	0a	05	??	??	cb	e2	??
??	87	fc	??	38	98	83	??	??	32	ac	??	??	6a	a8	??
??	43	1f	??	??	66	87	f5	??	85	f7	??	??	1c	a9	??

GARBAGE BYTES WITH
NO FF IN THEM.

Polyglots:

a single pair with
several use cases.

CAN'T COMBINE JPEG AND MBR:
FF D8 IS AN INVALID OPCODE.



From MalSHA1...
...to the real thing!

2015: Implementing Stevens13

1. Research file trick
2. Implement attack
3. Craft files



Stevens13 compared with MalSHA1

- Complex computation
- Expensive computation
- + Prefix
- Totally random blocks
- + Fixed mask
- + Blocks start with a difference

NEVER TRIED BEFORE:
(CAN'T BE INTERRUPTED/TWEAKED)

ONE. SINGLE. TRY.

CONSTRAINTS--

CONSTRAINTS++

RELIABILITY++

RELIABILITY++

1. Research file trick

- MaISHA1's JPEG trick would work.
- We'd like a new trick. PDF?
 - Nothing existing versatile so far.
 - Experiments with PDF (XREF, object numbers)
 - Never works reliably accross all readers.
- No SHA1 collision at this stage - hard to get traction.

AT THIS STAGE IT'S STILL ONLY
A SET OF WEIRD FILE CONSTRAINTS.

If you're not familiar
with PDF...

...with **my** vision of PDF!

a *correct* PDF

HEADER

%PDF-1.1 SIGNATURE & VERSION INFORMATION

XREF TABLE

TRAILER

CROSS
REFERENCE

```
xref
0 5
0000000000 65535 f
0000000010 00000 n
0000000047 00000 n
0000000111 00000 n
0000000313 00000 n
```

CROSS REFERENCES
5 OBJECTS, STARTING AT INDEX 0
(STANDARD FIRST EMPTY OBJECT 0
OFFSET TO OBJECT 1, REV 0
TO OBJECT 2...
3...
4

```
trailer
<<
  /Root 1 0 R
>>

startxref
413
%%EOF
```

BODY

DICTIONARY
<< [ID VALUE]* >>

OBJECT REFERENCE:
<OBJECT NUMBER> <REVISION NUMBER> R

IDENTIFIER (WITH /)

ARRAY

STRING

STREAM PARAMETERS:
LENGTH, COMPRESSION....

BEGIN TEXT
FONT F1 (ARIAL) SET TO SIZE 110
MOVE TO COORDINATE 10, 400
OUTPUT TEXT "HELLO WORLD!"
END TEXT

```
1 0 obj
<<
  /Pages 2 0 R
>>
endobj

2 0 obj
<<
  /Type /Pages
  /Count 1
  /Kids [3 0 R]
>>
endobj

3 0 obj
<<
  /Type /Page
  /Contents 4 0 R
  /Parent 2 0 R
  /Resources <<
    /Font <<
      /F1 <<
        /Type /Font
        /Subtype /Type1
        /BaseFont /Arial
      >>
    >>
  >>
endobj

4 0 obj
<< /Length 50 >>
stream
BT
  /F1 110 Tf
  10 400 Td
  (Hello World!)Tj
ET
endstream
endobj
```



```
1 0 obj
<<
  /Resources << /Font << /F1 <<
    /BaseFont /Arial /Subtype
  /Type1 >> >>
  >>
  /Contents << >>
stream
  /F1 170 Tf
  10 400 Td
  (Firefox) Tj
endstream
>>
endobj

xref

%trailer << /Root << /Pages <<
  /Kids [1 0 R] /Count 1>> >>
>>
```

%PDF

```
1 0 obj
<< /Pages
  << /Kids [
    << /Contents 2 0 R >>
  ] >>
>>
```

```
2 0 obj
<<>>
stream
  95 Tf
  20 400 Td
  (Chrome) Tj
endstream
```

```
trailer <<
  /Root 1 0 R
>>
```

working
PDFs

1 no signature

<<

no /Parent

no /Type

<< /F1 <<

/BaseFont /Arial /Subtype

/Type1 >> >>

>>

inline /Contents

>no /Length

stream

no BT/ET

INVALID?

1.70 Tf

10 400 Td

(Firefox) Tj

endstream

>>

endobj

xref

empty XREF

comment

%trailer: << /Root << no /Pages <<

/Kids [1 0 R] /Count 1 >> >>

>>

no /Type

Direct /Root

no startxref

no %%EOF

%PDF

truncated signature

1 0 obj

<< /Pages

direct /Kids

<< no /Parent

no /Type

<< no /Resources

no /Font

1 >>

>>

no /Type

no /Count

no endobj

INVALID?

2 0 obj

<< no /Length

stream

no BT/ET

no font reference

20 400 Td

(Chrome) Tj

no endobj

no XREF

Direct /Root

%trailer: << /Root 1 0 R

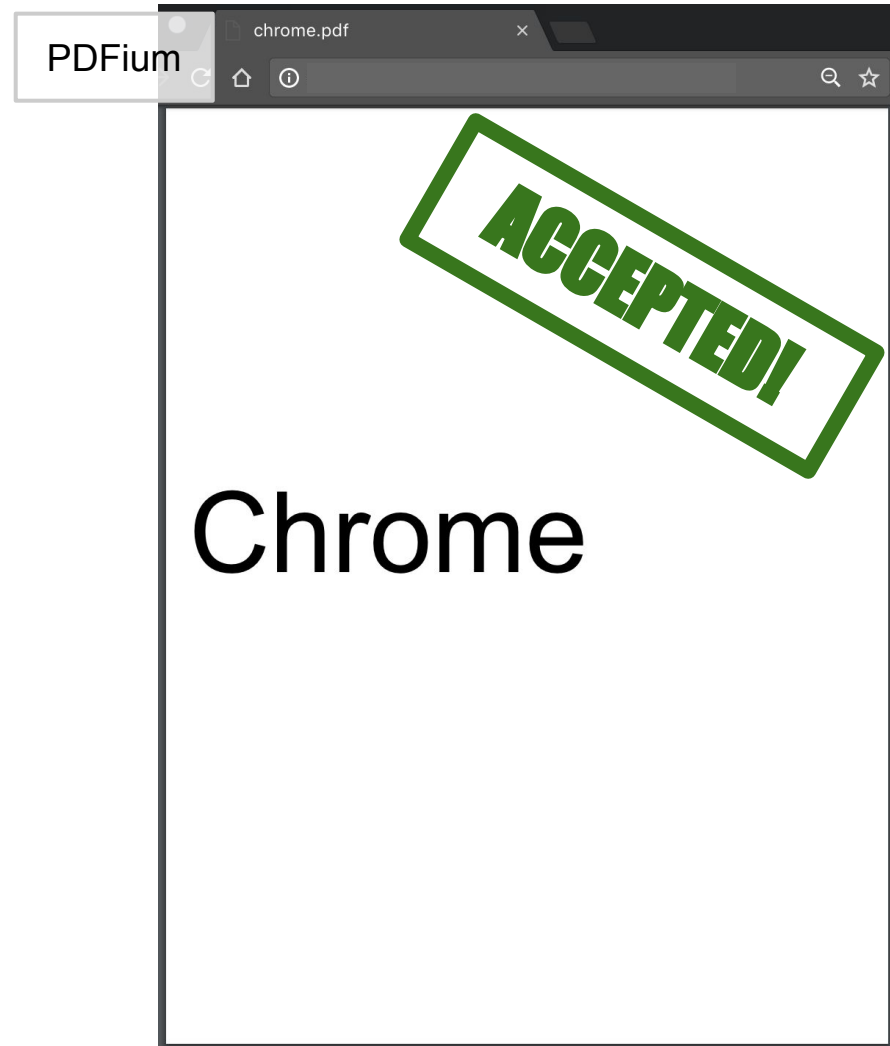
>>

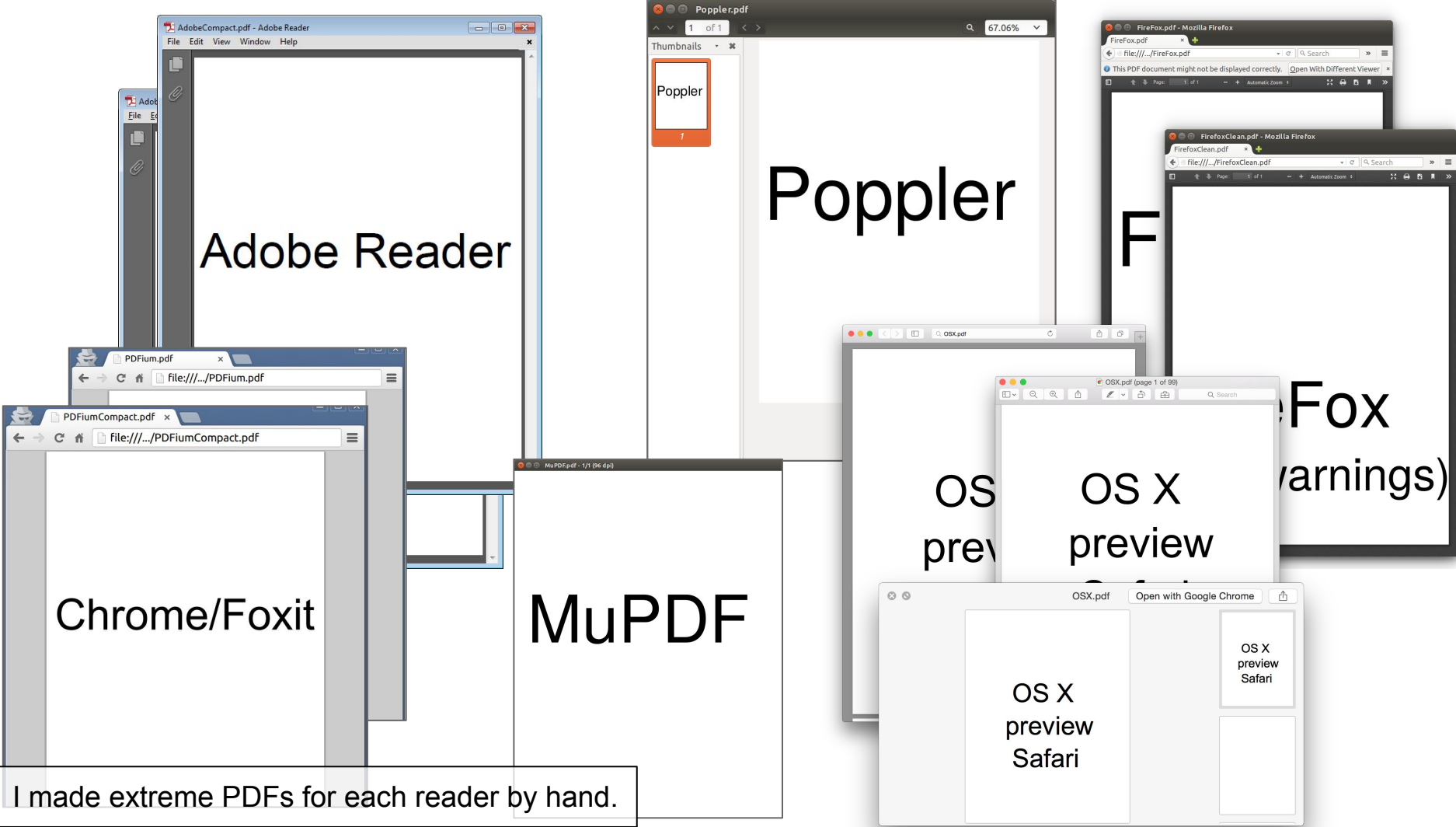
no /Size

no %%EOF

no /Type

no startxref





Adobe Reader

Poppler

F

Fox
(arnings)

Chrome/Foxit

MuPDF

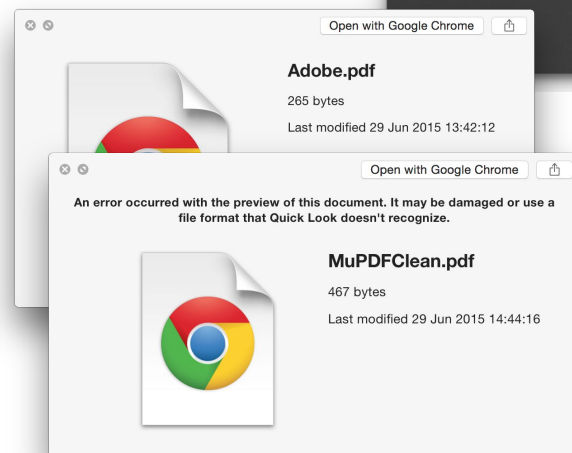
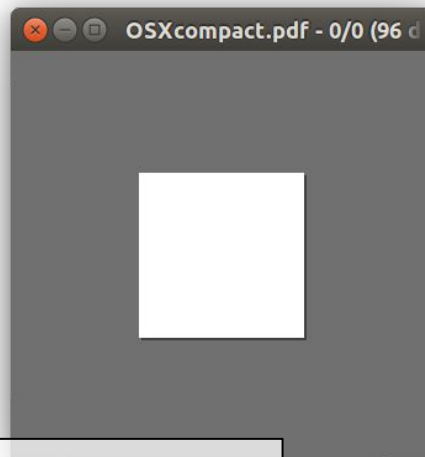
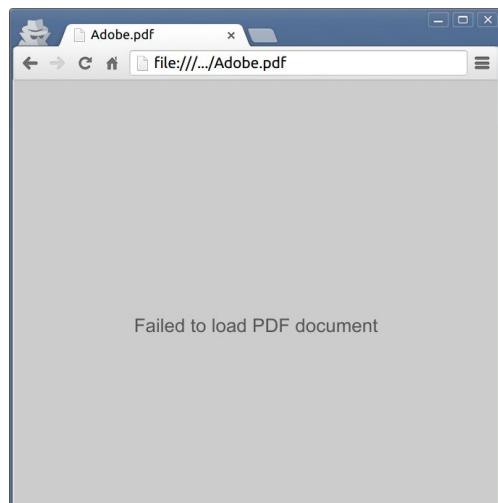
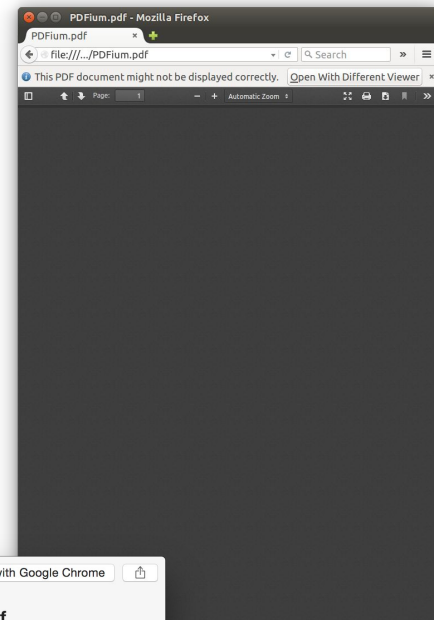
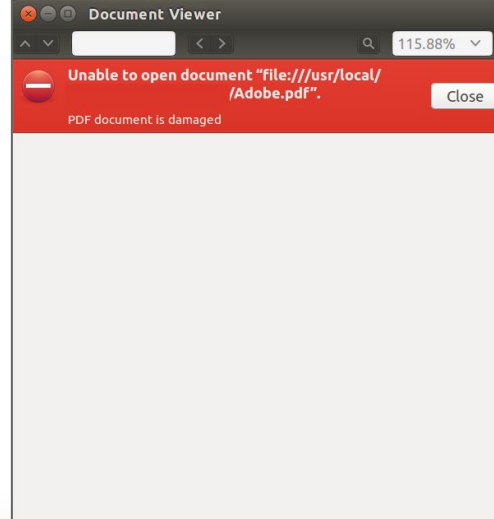
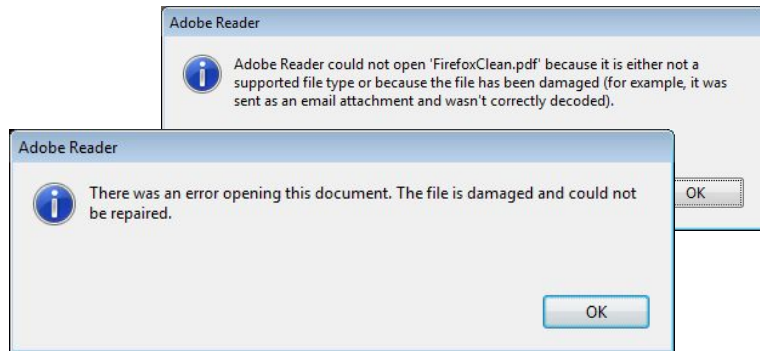
OS
prev

OS X
preview

OS X
preview
Safari

OS X
preview
Safari

I made extreme PDFs for each reader by hand.



These extreme PDFs fail on any other reader.

The devil is in the detail

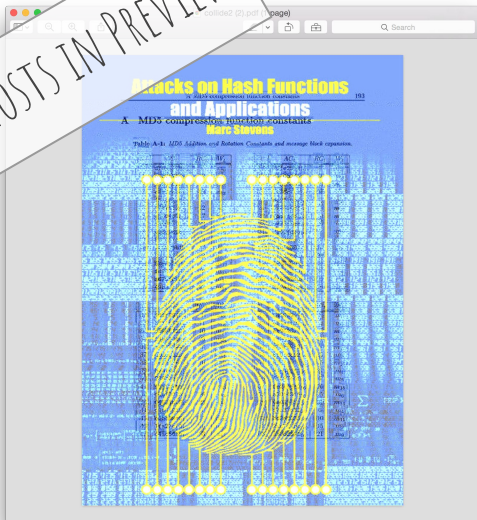
- All PDF parsers have their weirdness
 - Does it work? Does it display, behave normally?
 - A trick on a PDF reader is easy, but a reliable trick for all of them is **hard**.

Examples:

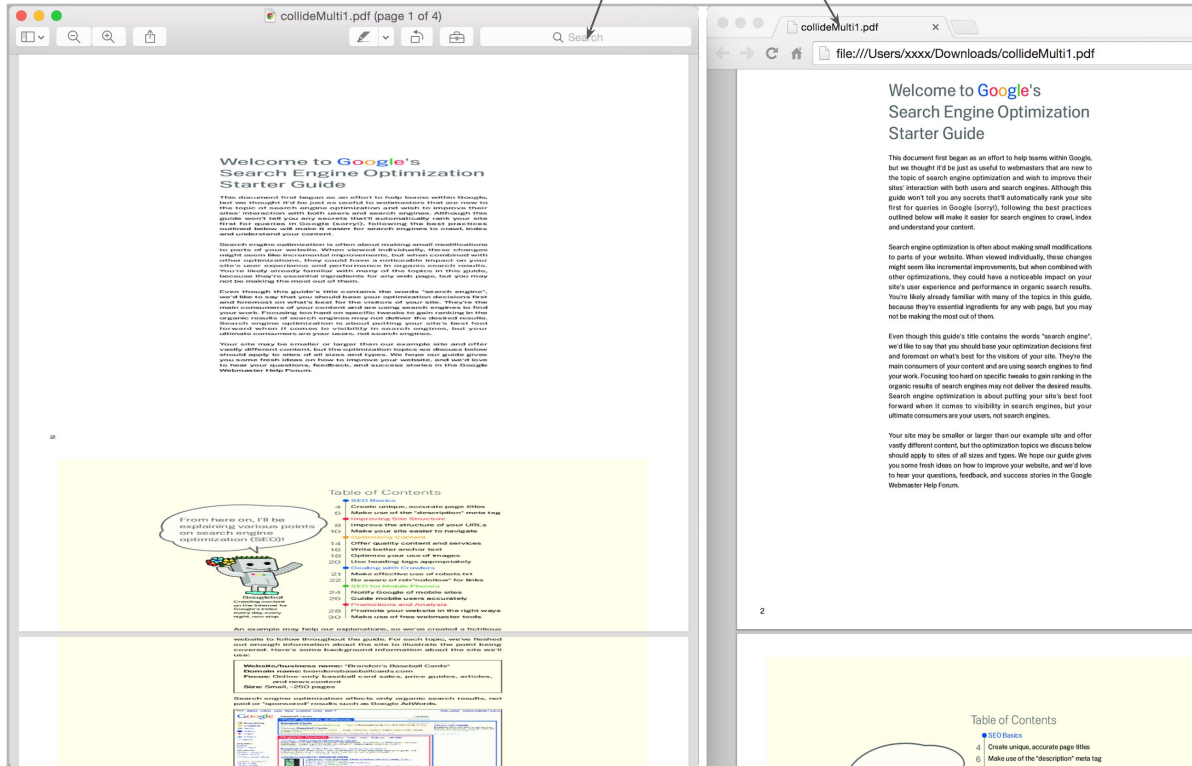
- Preview is more strict for JPEG structures. But created some funky ghost JPEGs :)
- OTOH it's less compatibility for gradients.
- An unusual JPEG in a PDF can easily reboot a Kindle.
- A complex JPEG can take minutes to load.
- A crazy JPEG in a PDF displays glitches in Adobe.



GHOSTS IN PREVIEW



DIFFERENT RESIZING
IN PREVIEW



2015: PDF is tricky...

- A PDF trick with total compatibility...?
 - With doc-level control? (not just a glitch)
- Eventually... JPEG in a PDF:
 - PDF embeds entire JPEG files
 - Image parameters can be referenced
 - Reliable
 - No possible error
 - "Sane" PoCs - very little overhead
 - Reusable

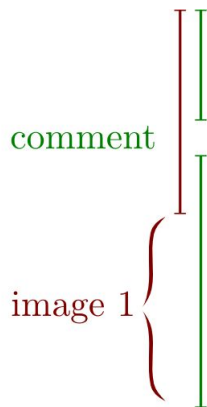
AFTER THE COLLISION BLOCKS,
SO NO RESTRICTIONS ON DIMENSIONS!

PDF ARE USUALLY DOCUMENTS.
WE WANTED FAKE DOCUMENTS!

Pushing the limits of our JPEG trick

The first image has to be jumped over.

Only 393x438 px
in 90% quality \Rightarrow 55Kb
Yet already near limit!

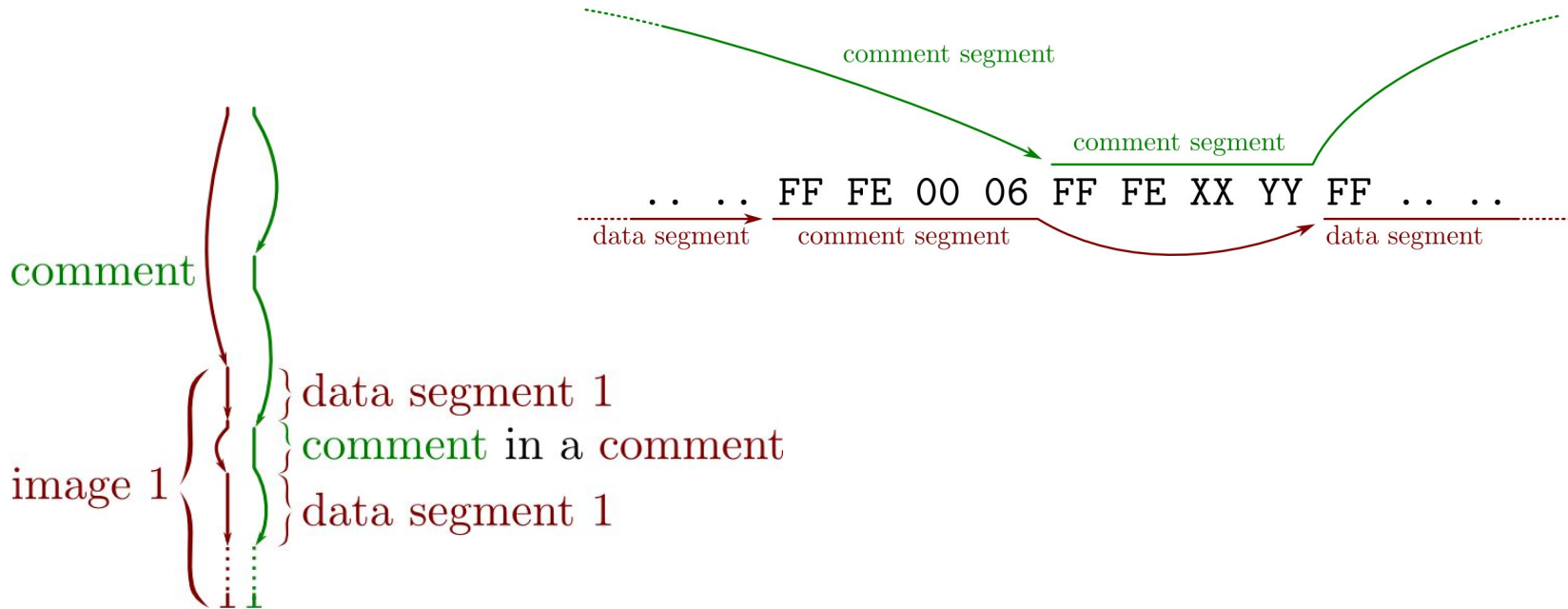


a comment over the whole image
 \Rightarrow ImageSize < 64kb

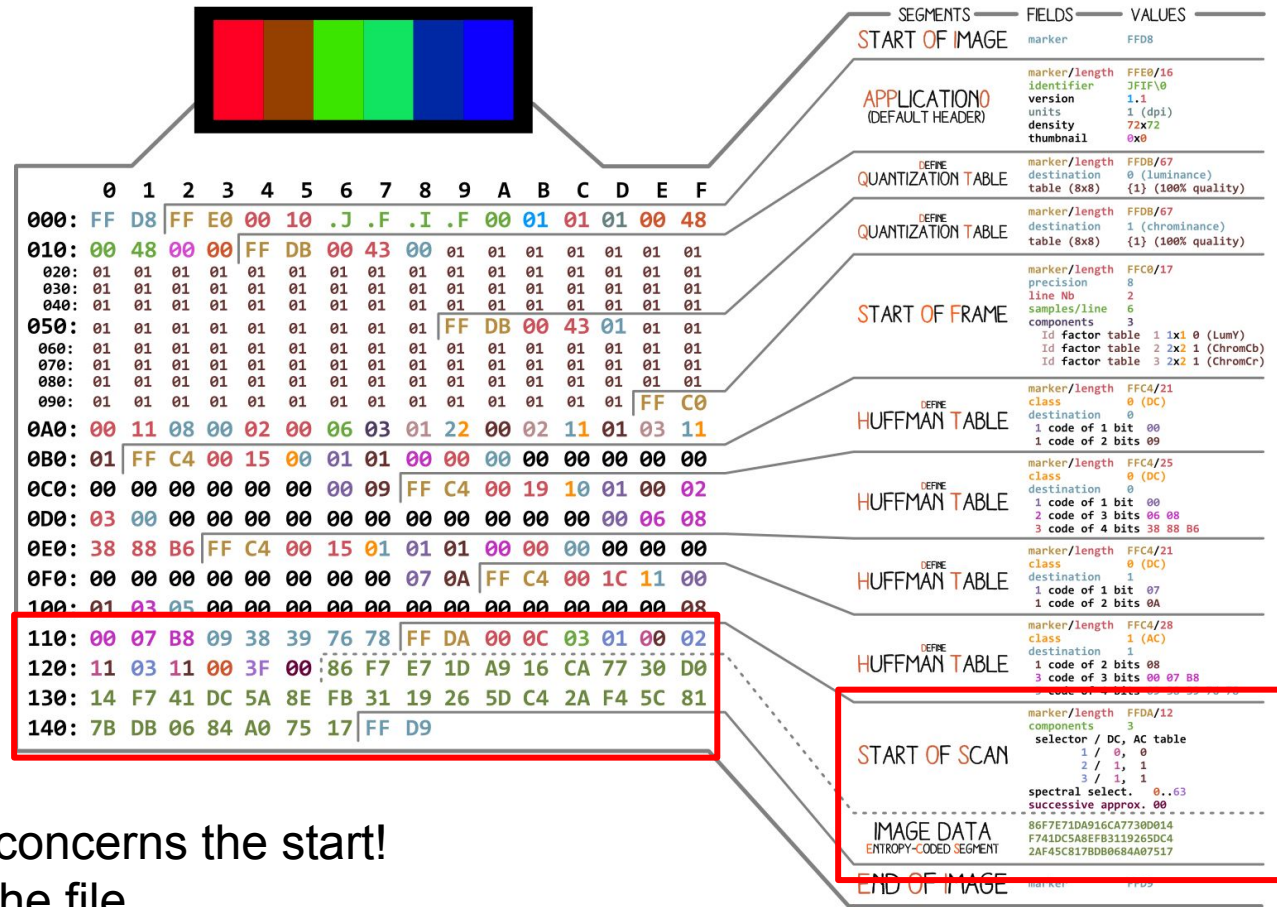


GOOD FOR A PHOTO,
NOT FOR A DOC!

2 comments per segment



a comment over each segment
 $\Rightarrow \text{Max}(\{\text{SegmentSize}\}) < 64\text{kb}$



The scan length only concerns the start!
 The ECS grows with the file,
 and is not limited to 64Kb!

SHattered

The first concrete collision attack against SHA-1

<https://shattered.io>

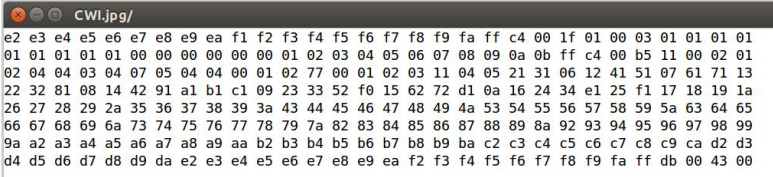
CWI

Marc Stevens
Pierre Karpman

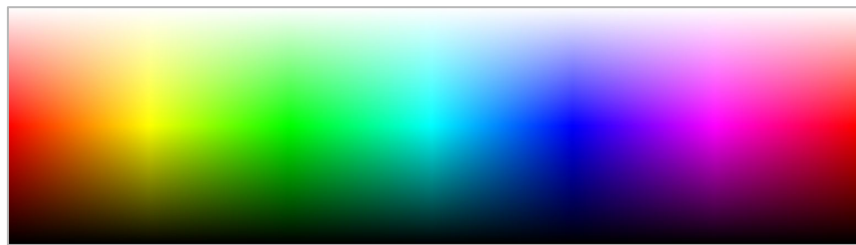
Google

Elie Bursztein
Ange Albertini
Yarik Markov

1024x740 Q.100% \Rightarrow 228 Kb
a single scan of 227 Kb!



address	name	type	size	description
00000000.0	start_image/	JpegChunk	00000002.0	Start of image (SOI)
00000002.0	app0/	JpegChunk	00000018.0	APP0
00000014.0	exif/	JpegChunk	00000066.0	EXIF
00000056.0	photoshop/	JpegChunk	00000058.0	Photoshop
00000090.0	start_frame/	JpegChunk	00000019.0	Start of frame (baseline)
000000a3.0	huffman[0]/	JpegChunk	00000033.0	Define Huffman Table (DHT)
000000c4.0	huffman[1]/	JpegChunk	00000183.0	Define Huffman Table (DHT)
0000017b.0	huffman[2]/	JpegChunk	00000033.0	Define Huffman Table (DHT)
0000019c.0	huffman[3]/	JpegChunk	00000183.0	Define Huffman Table (DHT)
00000253.0	quantization[0]/	JpegChunk	00000069.0	Define Quantization Table (DQT)
00000298.0	quantization[1]/	JpegChunk	00000069.0	Define Quantization Table (DQT)
000002dd.0	restart_interval/	JpegChunk	00000006.0	Define Restart Interval (DRI)
000002e3.0	start_scan/	JpegChunk	00000014.0	Start Of Scan (SOS)
000002f1.0	data	RawBytes	00227565.0	JPEG data
00037bde.0	end_image/	JpegChunk	00000002.0	End of image (EOI)



image

0:Y
luma (brightness)



1:Cb
blueness

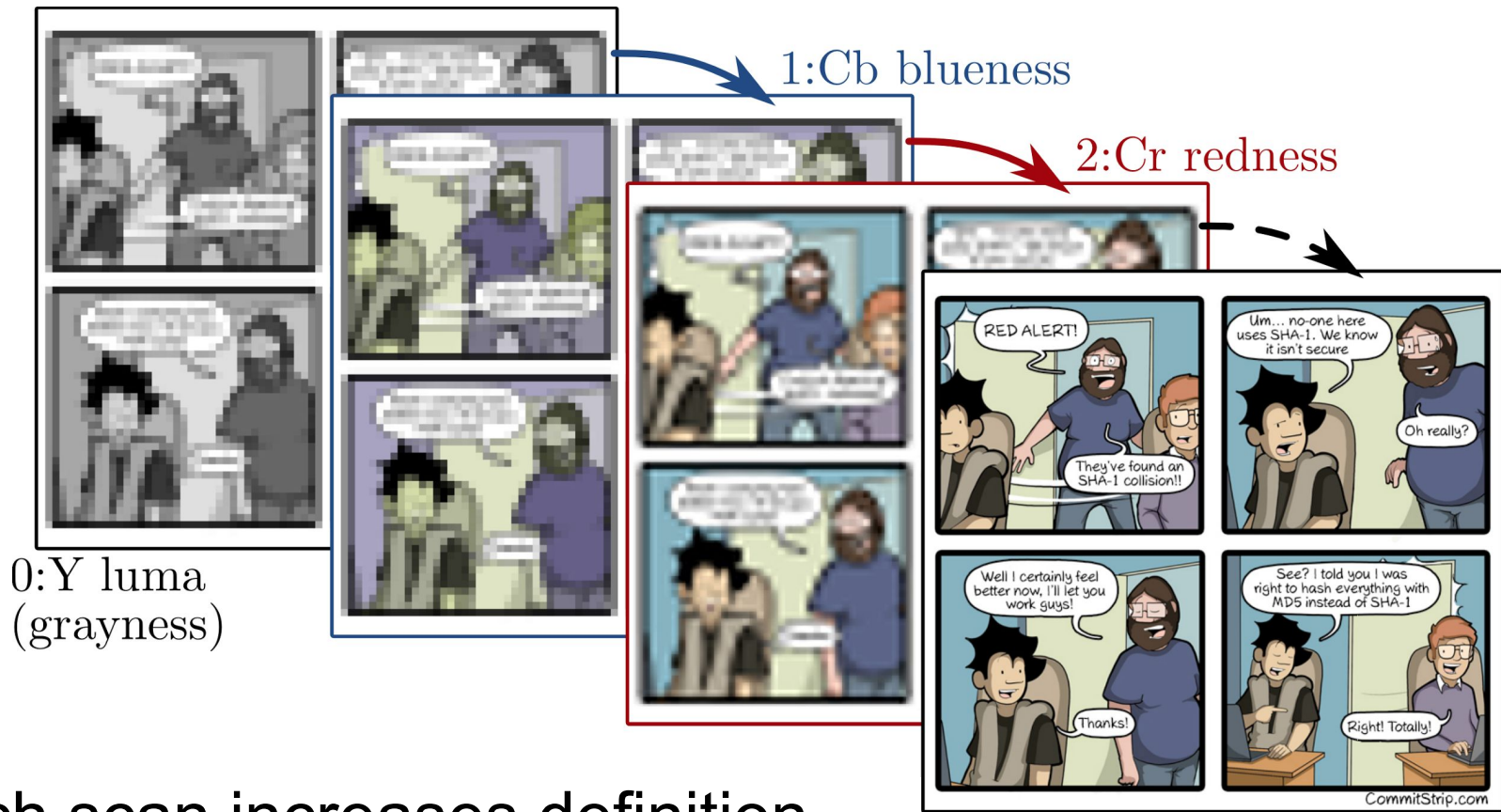


2:Cr
redness



**A JPEG image
is decomposed**

Components



Each scan increases definition
⇒ progressive file, smaller scans

Welcome to

JPEG

School of Wizardry

libJPEG's JPEGTran & wizard.doc

Advanced usage instructions for the Independent JPEG Group's JPEG software

This file describes cjpeg's "switches for wizards".

The "wizard" switches are intended for experimentation with JPEG by persons who are reasonably knowledgeable about the JPEG standard. If you don't know what you are doing, DON'T USE THESE SWITCHES. You'll likely produce files with worse image quality and/or poorer compression than you'd get from the default settings. Furthermore, these switches must be used with caution when making files intended for general use, because not all JPEG decoders will support unusual JPEG parameter settings.

Quantization Table Adjustment

Ordinarily, cjpeg starts with a default set of tables (the same ones given as examples in the JPEG standard) and scales them up or down according to the -quality setting. The details of the scaling algorithm can be found in jccparam.c. At very low quality settings, some quantization table entries can get scaled up to values exceeding 255. Although 2-byte quantization values are supported by the IJG software, this feature is not in baseline JPEG and is not supported by all implementations. If you need to ensure wide compatibility of low-quality files, you can constrain the scaled quantization values to no more than 255 by giving the -baseline switch. Note that use of -baseline will result in poorer quality for the same file size, since more bits than necessary are expended on higher AC coefficients.

You can substitute a different set of quantization values by using the -qttables switch:

-qttables file Use the quantization tables given in the named file.

<http://libjpeg.cvs.sourceforge.net/viewvc/libjpeg/libjpeg/wizard.doc?content-type=text%2Fplain>

```
$ jpegtran --help
usage: jpegtran [switches] [inputfile]
Switches (names may be abbreviated):
  -copy none      Copy no extra markers from source file
  -copy comments  Copy only comment markers (default)
  -copy all       Copy all extra markers
  -optimize       Optimize Huffman table (smaller file, but slow compression)
  -progressive    Create progressive JPEG file
Switches for modifying the image:
  -grayscale      Reduce to grayscale (omit color data)
  -flip [horizontal|vertical] Mirror image (left-right or top-bottom)
  -rotate [90|180|270] Rotate image (degrees clockwise)
  -transpose      Transpose image
  -transverse     Transverse transpose image
  -trim           Drop non-transformable edge blocks
  -cut WxH+X+Y    Cut out a subset of the image
Switches for advanced users:
  -restart N      Set restart interval in rows, or in blocks with B
  -maxmemory N    Maximum memory to use (in kbytes)
  -outfile name   Specify name for output file
  -verbose or -debug Emit debug output
Switches for wizards:
  -scans file      Create multi-scan JPEG per script file
```


Custom scans

Use JPEGTran's to tweak scans
and make them smaller than 64Kb,

Wizardry is hard:

- JPEGTran is inconsistent
- The documentation's examples are broken.



Making a big
image fit
w/ custom scans
definitions.

FEW COLORS

```
0: 0-0, 0, 0;  
0: 1-1, 0, 0;  
0: 2-6, 0, 0;  
0: 7-10, 0, 0;  
0: 11-13, 0, 0;  
0: 14-20, 0, 0;  
0: 21-26, 0, 0;  
0: 27-32, 0, 0;  
0: 33-40, 0, 0;  
0: 41-48, 0, 0;  
0: 49-54, 0, 0;  
0: 55-63, 0, 0;  
1: 0-0, 0, 0;  
1: 1-16, 0, 0;  
1: 17-32, 0, 0;  
1: 33-63, 0, 0;  
2: 0-0, 0, 0;  
2: 1-16, 0, 0;  
2: 17-32, 0, 0;  
2: 33-63, 0, 0;
```



1944x2508 100%, 860 Kb ⇒ 20 scans

Limitations?

LibJPEG has an limit of 100 scans.
On writing. Not on reading ;)

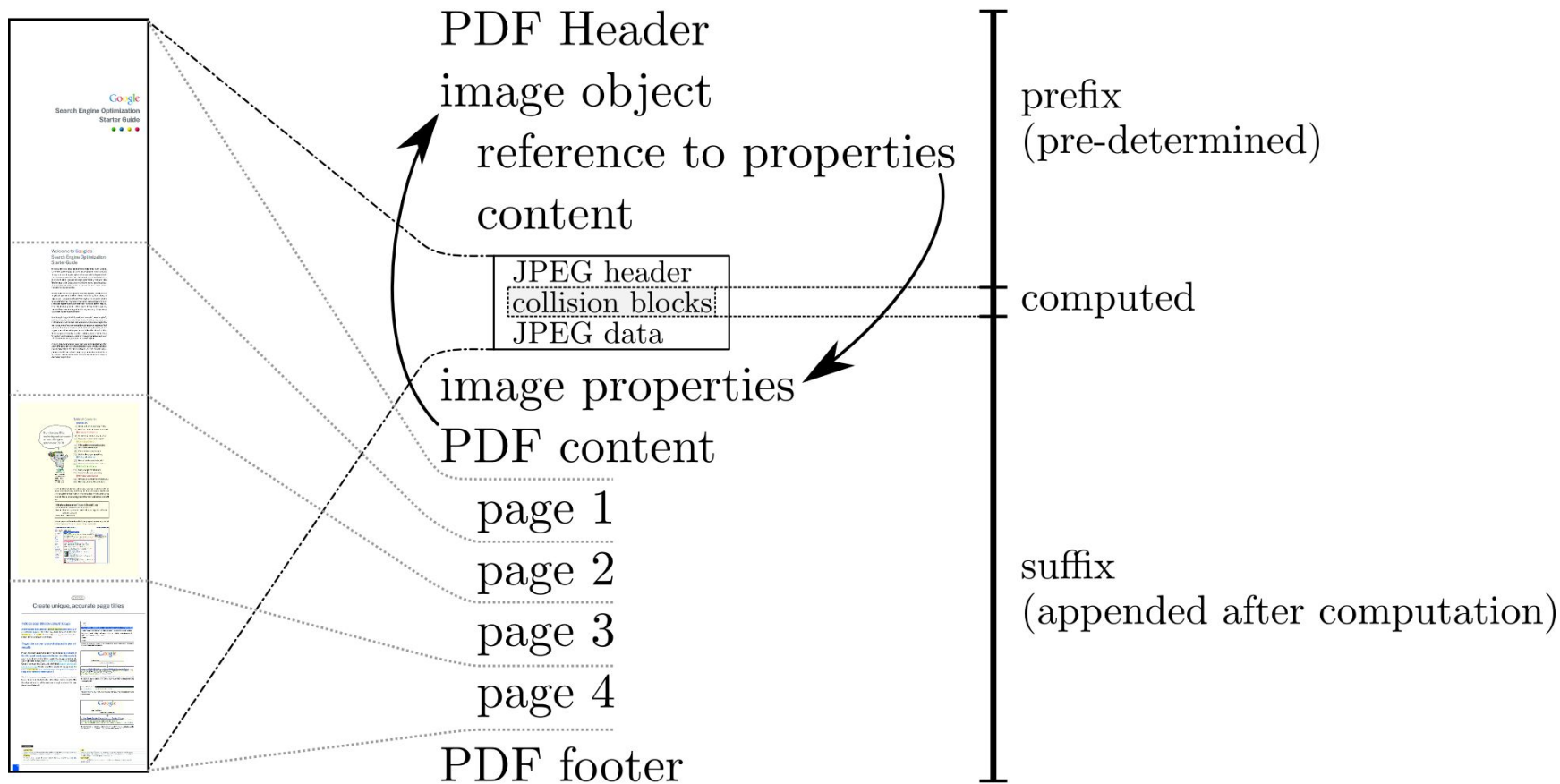
⇒ we could release a multi-page doc,
but it's giving mobiles a hard time.

Shattered: It's a JPEG in a PDF

- We still want a PDF file!
- PDF header, declare image
- Reference all /Image parameters after the file data.
 - After the collision blocks
- Put 2 images contents
 - With the same parameters, unlike MalSHA1
- Put image parameters values
- Finalize PDF file.

COLORS, DIMENSIONS...

PDF trick structure



**8 brain-year,
100 GPU-year
and 6500 CPU-year later...**

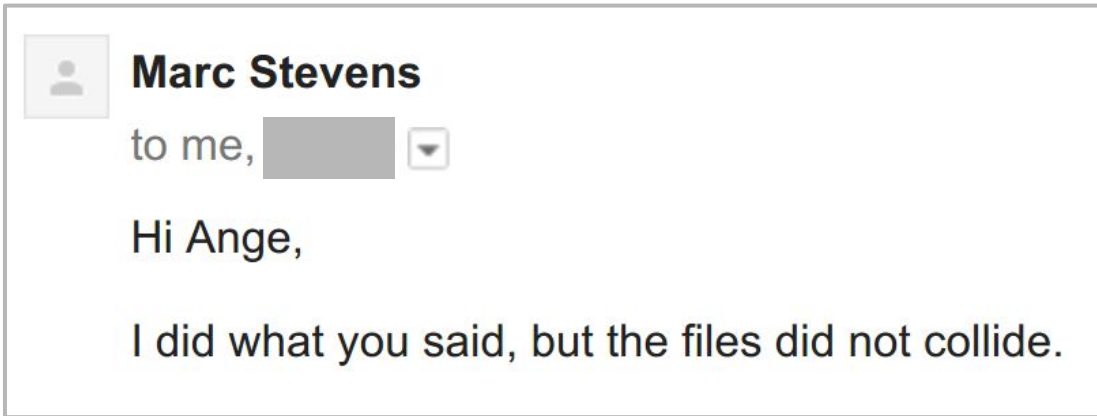
Woohoo! We have a collision!
"Here is **the** file..."

OCT 15 -> JAN 17
HERE COMES THE RANDOMNESS!

The great wave off Stevens13



Then this happened...



I completely lost my... ;)

I ALSO LOST COMPATIBILITY WITH ADOBE AND SAFARI AT SOME POINT...

Lessons learned

- Keeping notes and PoCs helps.
- a diary and a log of command lines might seem overkill...
...but it *really* helps!

(Especially as readers have been updated in the meantime!)

Shattered is real

With 0 bug reported!



File 1

File 2

Identical
prefixCollision
blocks

Suffix

```

000: 2550 4446 2d31 2e33 0a25 e2e3 cfd3 0a0a %PDF-1.3.%.
010: 0a31 2030 206f 626a 0a3c 3c2f 5769 6474 .1 0 obj.<</Widt
020: 6820 3220 3020 522f 4865 6967 6874 2033 h 2 0 R/Height 3
030: 2030 2052 2f54 7970 6520 3420 3020 522f 0 R/Type 4 0 R/
040: 5375 6274 7970 6520 3520 3020 522f 4669 Subtype 5 0 R/Fi
050: 6c74 6572 2036 2030 2052 2f43 6f6c 6f72 lter 6 0 R/Color
060: 5370 6163 6520 3720 3020 522f 4c65 6e67 Space 7 0 R/Leng
070: 7468 2038 2030 2052 2f42 6974 7350 6572 th 8 0 R/BitsPer
080: 436f 6d70 6f6e 656e 7420 383e 3e0a 7374 Component 8>>.st
090: 7265 616d 0aff d8ff eam.....$SHA-1
0a0: 2069 7320 6465 6164 comment length: 0x017f is dead!!!!./..
0b0: 0923 3975 9c39 b1a1 c63c 4c97 e1ff fe01 #9u.9...<L....
0c0: 7f46 dc93 a6b6 7e01 3b02 9aaa 1db2 560b F....~.;....V.
0d0: 45ca 67d6 88c7 f84b 8c4c 791f e02b 3df6 .g....[.Ly..+=.
0e0: 14f8 6db1 6909 01c5 6b45 c153 0afe dfb7 .m.i...kE.S....
0f0: 6038 e972 722f e7ad 728f 0e49 04e0 46c2 .8.rr./...r...I..F.
100: 3057 0fe9 d413 98ab e12e f5bc 942b e335 0W.....+.5
110: 42a4 802d 98b5 d70f 2a33 2ec3 7fac 3514 B.-....*3....5.
120: e74d dc0f 2cc1 a874 cd0c 7830 5a21 566a .M....t...x0Z!Vd
130: 6130 9789 606b d0bf 3f98 cda8 0446 2941 a0...k...?....F).

```

same hash at this point

```

230: 0000 fffe 012d 0000 0000 0000 0000 ffe0 .....
240: 0010 4a46 4946 0001 0101 0048 0048 0000 ..JFIF.....H.H..

```

first image data

```

3a0: e9d6 d667 a7b0 7e65 1299 e39d 39c0 c7ff ...g...~e....9...
3b0: d92d 2d2d 2dff e000 104a 4649 4600 0101 -----JFIF...
3c0: 0100 4800 4800 00ff db00 4300 0101 0101 ..H.H.....C.....

```

second image data (ignored)

```

4e0: 4b14 97f7 7f39 fcd7 f1ff d90a 656e 6473 K....9.....ends
4f0: 7472 6561 6d0a 656e 646f 626a 0a0a 3220 tream.endobj..2
500: 3020 6f62 6a0a 380a 656e 646f 626a 0a0a 0 obj.8.endobj..

```

PDF footer

```

840: 3e0a 0a73 7461 7274 7872 6566 0a31 3830 >..startxref.180
850: 380a 2525 454f 460a 8.%EOF.

```

PDF header

```

2550 4446 2d31 2e33 0a25 e2e3 cfd3 0a0a %PDF-1.3.%.
0a31 2030 206f 626a 0a3c 3c2f 5769 6474 .1 0 obj.<</Widt
6820 3220 3020 522f 4865 6967 6874 2033 h 2 0 R/Height 3
2030 2052 2f54 7970 6520 3420 3020 522f 0 R/Type 4 0 R/
5375 6274 7970 6520 3520 3020 522f 4669 Subtype 5 0 R/Fi
6c74 6572 2036 2030 2052 2f43 6f6c 6f72 lter 6 0 R/Color
5370 6163 6520 3720 3020 522f 4c65 6e67 Space 7 0 R/Leng
7468 2038 2030 2052 2f42 6974 7350 6572 th 8 0 R/BitsPer
436f 6d70 6f6e 656e 7420 383e 3e0a 7374 Component 8>>.st
7265 616d 0aff d8ff eam.....$SHA-1
2069 7320 6465 6164 comment length: 0x0173 is dead!!!!./..
0923 3975 9c39 b1a1 c63c 4c97 e1ff fe01 #9u.9...<L....
7346 dc91 66b6 7e11 8f02 9ab6 21b2 560f sF...f.....V.
f9ca 67cc a8c7 f85b a84c 7903 0c2b 3de2 .g....[.Ly..+=.
18f8 6db3 a909 01d5 df45 c14f 26fe dfb3 .m.m.....E.0k...
dc38 e96a c22f e7bd 728f 0e45 bce0 46d2 .8.j./...r...E..F.
3c57 0feb 1413 58bb 552e f5a0 a82b e331 <W.....U....+.1
fea4 8037 b555 d71f 0e33 2edf 93ac 3500 ...7.....3....5.
ed4d dc0d ecc1 a864 790c 782c 7621 5660 .M....dy.x,v!V^
db30 97f1 d06b d0af 3f98 cda4 bc46 29b1 0...k...?....F).

```

JPG header and comment declaration

comments chain

```

0000 fffe 012d 0000 0000 0000 0000 ffe0 .....
0010 4a46 4946 0001 0101 0048 0048 0000 ..JFIF.....H.H..

```

first image data (ignored)

```

e9d6 d667 a7b0 7e65 1299 e39d 39c0 c7ff ...g...~e....9...
d92d 2d2d 2dff e000 104a 4649 4600 0101 -----JFIF...
0100 4800 4800 00ff db00 4300 0101 0101 ..H.H.....C.....

```

second image data

```

4b14 97f7 7f39 fcd7 f1ff d90a 656e 6473 K....9.....ends
7472 6561 6d0a 656e 646f 626a 0a0a 3220 tream.endobj..2
3020 6f62 6a0a 380a 656e 646f 626a 0a0a 0 obj.8.endobj..

```

PDF footer

```

3e0a 0a73 7461 7274 7872 6566 0a31 3830 >..startxref.180
380a 2525 454f 460a 8.%EOF.

```

official PoCs, side by side

Details

PDF signature 000: %PDF-1.3
non-ASCII marker 009: %~~aa~~İÖ

```

000:  .% .P .D .F .- .1 .. .3 \n .% E2 E3 CF D3 \n \n
010:  \n .1 .0 .o .b .j \n .< .< ./ .W .i .d .t
020:  .h .2 .0 .R ./ .H .e .i .g .h .t .3
030:  .0 .R ./ .T .y .p .e .4 .0 .R ./
040:  .S .u .b .t .y .p .e .5 .0 .R ./ .F .i
050:  .l .t .e .r .6 .0 .R ./ .C .o .l .o .r
060:  .S .p .a .c .e .7 .0 .R ./ .L .e .n .g
070:  .t .h .8 .0 .R ./ .B .i .t .s .P .e .r
080:  .C .o .m .p .o .n .e .n .t .8 .> .> \n .s .t
090:  .r .e .a .m \n FF D8 FF FE 00 24 .S .H .A .- .1
0a0:  .i .s .d .e .a .d .! .! .! .! .! 85 2F EC
0b0:  09 23 39 75 9C 39 B1 A1 C6 3C 4C 97 E1 FF FE 01
0c0:  ??
  
```

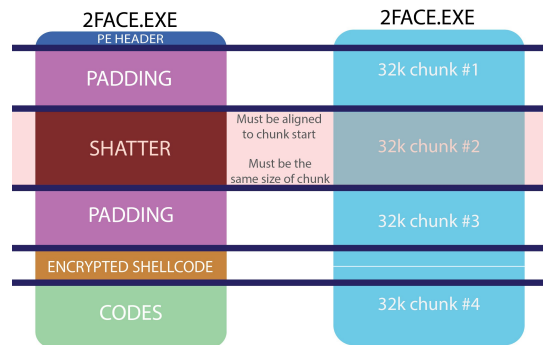
object declaration 011: 1 0 obj
image object properties 019: <</Width 2 0 R/Height 3 0 R/Type 4 0 R/Subtype 5 0 R/Filter 6 0 R
/ColorSpace 7 0 R/Length 8 0 R/BitsPerComponent 8>>
stream content start 08e: stream
JPEG Start Of Image 095: FF D8 length: 36
~~JPEG comment 097: FF FE 00 24~~
hidden death statement 09b: SHA-1 is dead!!!
randomization buffer 0ad: 85 2F EC 09 23 39 75 9C 39 B1 A1 C6 3C 4C 97 E1
~~JPEG comment 0bd: FF FE 01~~
start of collision block 0c0: ??
length: 01??

Impact

"SHA-1 IS NOT COLLISION RESISTANT..."

- [CVE-2005-4900](#) updated :)
- It broke [SVN](#) in practice!
 - SHA1 for deduplication
 - MD5 for integrity
- [BitErrant](#)
 - BitTorrent uses SHA1 for file chunks

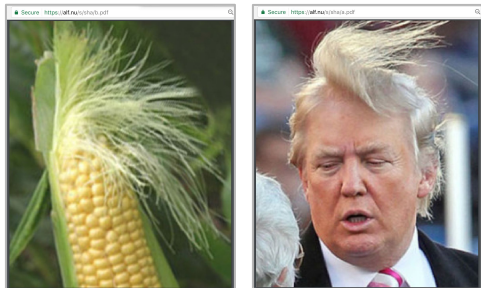
```
...  
Checksum mismatch: shattered-2.pdf  
expected: 5bd9d8cab46041579a311230539b8d1  
got: ee4aa52b139d925f8d8884402b0a750c  
...
```



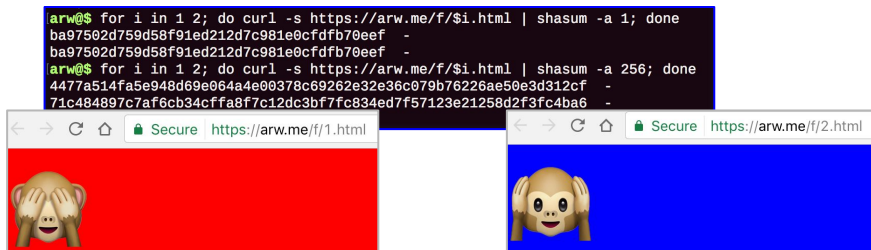
Internet does its thing...

first public PoCs

- PoCs generators
 - [simple](#) within 5 hours (!)
 - [advanced](#)



- [HTML collision](#)



- Used in [Boston Key Party CTF](#), 50 pts
FLAG{AfterThursdayWeHadToReduceThePointValue}
- [Bitcoin bounty](#) claimed ;) [2.8K€]

~~Enthusiast~~ feedback

- [Bruce Schneier](#)

Yes, this brute-force example has its own website.

- [Linus Torvald](#)

...in a project like git, the hash isn't used for "trust".

- [John Gilmore](#)

Linus [...] wired assumptions about SHA1 deeply into git.

- [Robert J. Hansen](#) [OpenPGP, 2013]

Scaremongering about crypto is one of the quickest ways to make me angry.

We can do more

It's not just about full-page pictures.

It's not just full-page pictures

- It's a standard PDF document, with a 'bipolar' JPEG.
- Any PDF element can be part of the JPEG.
 - A multi-page doc w/ an image with appended pages.
 - A totally standard doc, with only a few elements replaced.

DEMO

Notice anything?

It's the complete Shattered paper...

The first collision for full SHA-1

Marc Stevens¹, Elie Bursztein², Pierre Karpman¹, Marcel Dupont², Yarik Markov²

¹ CWI Amsterdam

² Google Research
info@shattered.io
<https://shattered.io>

Abstract. SHA-1 is a widely used 1995 NIST cryptographic hash function standard that was officially deprecated by NIST in 2011 due to fundamental security weaknesses demonstrated in various analyses and theoretical attacks. Despite its deprecation, SHA-1 remains widely used in 2017 for document and TLS certificate signatures, and also in many software such as the GIT versioning system for integrity and backup purposes.

A key reason behind the reluctance of many industry players to replace SHA-1 with a safer alternative is the fact that finding an actual collision has seemed to be impractical for the past eleven years due to the high complexity and computational cost of the attack.

In this paper, we demonstrate that SHA-1 collision attacks have finally become practical by providing the first known instance of a collision. Furthermore, the prefix of the colliding messages was carefully chosen so that they allow an attacker to forge two PDF documents with the same SHA-1 hash, yet that display arbitrarily-chosen distinct visual contents.

We were able to find this collision by combining many special cryptanalytic techniques in complex ways and improving upon previous work. In total the computational effort spent is equivalent to $2^{63.1}$ SHA-1 compressions and took approximately 6500 CPU years and 100 GPU years. As a result while the computational power spent on this collision is larger than other public cryptanalytic computations, it is still more than 100,000 times faster than a brute force search.

Keywords: hash function, cryptanalysis, collision attack, collision example, differential path.

1 Introduction

A cryptographic hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ is a function that computes for any arbitrarily long message M a fixed-length hash value of n bits. It is a versatile cryptographic primitive used in many applications including digital signature schemes, message authentication codes, password hashing and content-addressable storage. The security or even the proper functioning of many of these applications rely on the assumption that it is practically impossible to find collisions. A collision being two distinct messages x , y that hash to the same value $H(x) = H(y)$. A brute-force search for collisions based on the so-called birthday paradox has a well understood cost of $\sqrt{\pi/2} \cdot 2^{n/2}$ expected calls to the hash function.

The MD-SHA family of hash functions is the most well-known hash function family, which includes MD5, SHA-1 and SHA-2 that all have found widespread use. This family originally started with MD4 [30] in 1990, which was quickly replaced by MD5 [31] in 1992 due to serious security weaknesses [7, 9]. Despite early known weaknesses of its underlying compression function [8], MD5 was widely deployed by the software industry for over a decade. A project MD5CRK that attempted to find a collision by brute force was halted early in 2004, when a team of researchers led by Xiaoyun Wang [43] demonstrated collisions for MD5 found by a groundbreaking special cryptanalytic attack that pioneered new techniques. In a major development, Stevens *et al.* [38] later showed that a more powerful type of attack (the so-called *chosen-prefix collision attack*) could be performed against MD5. This eventually led to the forgery of a Rogue Certification Authority that in principle completely undermined HTTPS security [39] in 2008. Despite this, even in 2017 there are still issues in deprecating MD5 for signatures [16].

d3f968d604bf1c31a4b3aaecd0f6b2fad4c33402

The first collision for full SHA-1

Marc Stevens¹, Elie Bursztein², Pierre Karpman¹, Marcel Dupont¹, Yarik Markov²

¹ CWI Amsterdam
² Google Research
info@shattered.io
<https://shattered.io>

Abstract. SHA-1 is a widely used 1995 NIST cryptographic hash function standard that was officially deprecated by NIST in 2011 due to fundamental security weaknesses demonstrated in various analyses and theoretical attacks.

Despite its deprecation, SHA-1 remains widely used in 2017 for document and TLS certificate signatures, and also in many software such as the GIT versioning system for integrity and backup purposes.

A key reason behind the reluctance of many industry players to replace SHA-1 with a safer alternative is the fact that finding an actual collision has seemed to be impractical for the past eleven years due to the high complexity and computational cost of the attack.

In this paper, we demonstrate that SHA-1 collision attacks have finally become practical by providing the first known instance of a collision. Furthermore, the prefix of the colliding messages was carefully chosen so that they allow an attacker to forge two PDF documents with the same SHA-1 hash yet that display arbitrarily-chosen distinct visual contents.

We were able to find this collision by combining many special cryptanalytic techniques in complex ways and improving upon previous work. In total the computational effort spent is equivalent to $2^{50.1}$ SHA-1 compressions and took approximately 6500 CPU years and 100 GPU years. As a result while the computational power spent on this collision is larger than other public cryptanalytic computations, it is still more than 100 000 times faster than a brute force search.

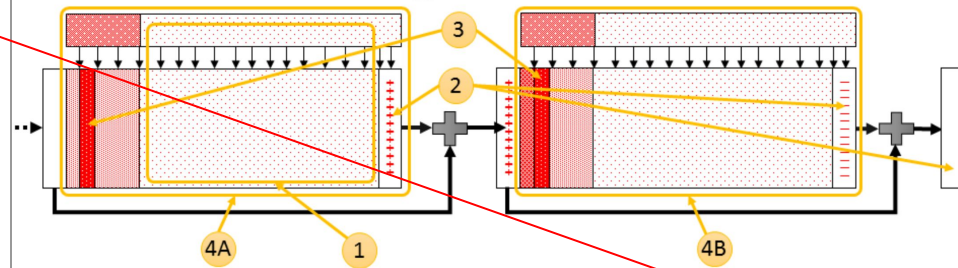
Keywords: hash function, cryptanalysis, collision attack, collision example, differential path.

1 Introduction

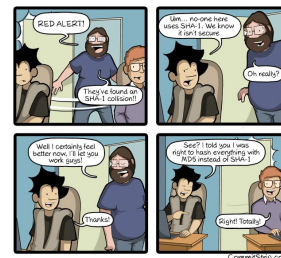
A cryptographic hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ is a function that computes for any arbitrarily long message M a fixed-length hash value of n bits. It is a versatile cryptographic primitive used in many applications including digital signature schemes, message authentication codes, password hashing and content-addressable storage. The security or even the proper functioning of many of these applications rely on the assumption that it is practically impossible to find collisions. A collision being two distinct messages x , y that hash to the same value $H(x) = H(y)$. A brute-force search for collisions based on the so-called birthday paradox has a well understood cost of $\sqrt{\pi/2} \cdot 2^{n/2}$ expected calls to the hash function.

The MD-SHA family of hash functions is the most well-known hash function family, which includes MD5, SHA-1 and SHA-2 that all have found widespread use. This family originally started with MD4 [30] in 1990, which was quickly replaced by MD5 [31] in 1992 due to serious security weaknesses [7, 9]. Despite early known weaknesses of its underlying compression function [8], MD5 was widely deployed by the software industry for over a decade. A project MD5CRK that attempted to find a collision by brute force was halted early in 2004, when a team of researchers led by Xiaoyun Wang [43] demonstrated collisions for MD5 found by a groundbreaking special cryptanalytic attack that pioneered new techniques. In a major development, Stevens *et al.* [38] later showed that a more powerful type of attack (the so-called *chosen-prefix collision attack*) could be performed against MD5. This eventually led to the forgery of a Rogue Certification Authority that in principle completely undermined HTTPS security [39] in 2008. Despite this, even in 2017 there are still issues in deprecating MD5 for signatures [16].

Ange Albertini²



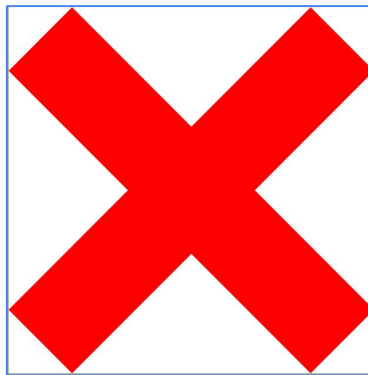
Marcel Dupont²



What's JPEG?

- An ~~image~~ format
- A **lossy** data storage format (specialized for photos?)
 - PDF takes it too literally:

3 out of 6 readers accept JPEG-stored data
for non-images objects, such as page content
(rejected by browsers)



=

```
1 0 0 RG    // color = red
150 w       // width
53 53 m     // start point
558 558 l   // end point
B           // draw path
53 558 m
558 53 l
B
```


Lossless JPEG?

- Quality 100%
- Grayscale JPEG \Rightarrow no component mixing

Still lossy!

- JPEG is 8x8 block based

\Rightarrow Repeat content lines 8 times.

- Pad a little to prevent truncation

\Rightarrow Reliably works !

DEMO

If

by Rudyard Kipling

If you can keep your head when all about you
Are losing theirs and blaming it on you,
If you can trust yourself when all men doubt you,
But make allowance for their doubting too;
If you can wait and not be tired by waiting,
Or being lied about, don't deal in lies,
Or being hated, don't give way to hating,
And yet don't look too good, nor talk too wise:

If you can dream-and not make dreams your master;
If you can think-and not make thoughts your aim;
If you can meet with Triumph and Disaster
And treat those two impostors just the same;
If you can bear to hear the truth you've spoken
Twisted by knaves to make a trap for fools,
Or watch the things you gave your life to, broken,
And stoop and build 'em up with worn-out tools:

If you can make one heap of all your winnings
And risk it on one turn of pitch-and-toss,
And lose, and start again at your beginnings
And never breathe a word about your loss;
If you can force your heart and nerve and sinew
To serve your turn long after they are gone,
And so hold on when there is nothing in you
Except the Will which says to them: 'Hold on!'

If you can talk with crowds and keep your virtue,
Or walk with Kings-nor lose the common touch,
If neither foes nor loving friends can hurt you,
If all men count with you, but none too much;
If you can fill the unforgiving minute
With sixty seconds' worth of distance run,
Yours is the Earth and everything that's in it,
And-which is more-you'll be a Man, my son!

If

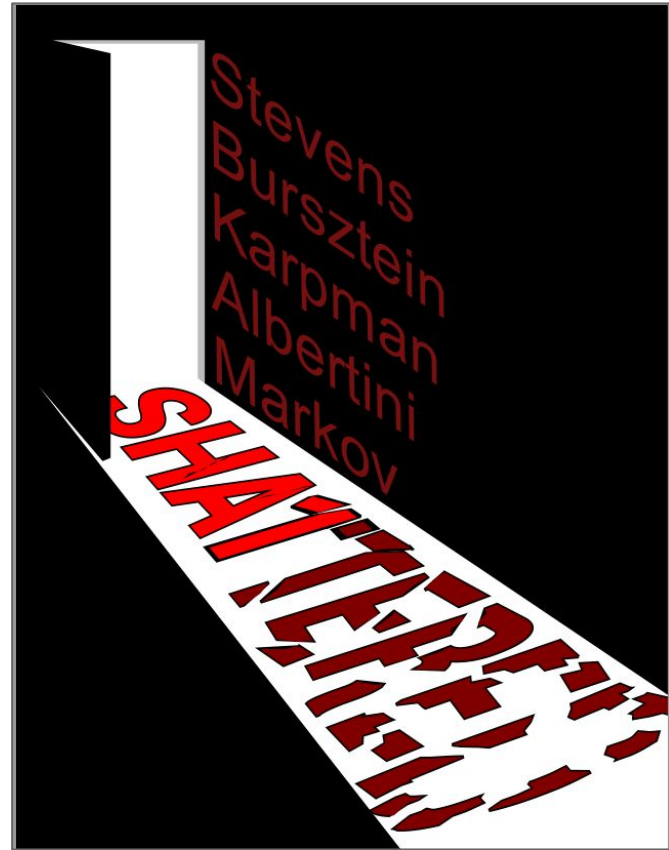
by Rudyard Kipling

If you can keep your head when all about you
Are losing theirs and blaming it on you,
If you can trust yourself when all men doubt you,
But make allowance for their doubting too;
If you can wait and not be tired by waiting,
Or being lied about, don't deal in lies,
Or being hated, don't give way to hating,
And yet don't look too good, nor talk too wise:

If you can dream-and not make dreams your master;
If you can think-and not make thoughts your aim;
If you can meet with Triumph and Disaster
And treat those two impostors just the same;
If you can bear to hear the truth you've spoken
Twisted by knaves to make a trap for fools,
Or watch the things you gave your life to, broken,
And stoop and build 'em up with worn-out tools:

If you can make one heap of all your winnings
And risk it on one turn of pitch-and-toss,
And lose, and start again at your beginnings
And never breathe a word about your loss;
If you can force your heart and nerve and sinew
To serve your turn long after they are gone,
And so hold on when there is nothing in you
Except the Will which says to them: 'Hold on!'

If you can talk with crowds and keep your virtue,
Or walk with Kings-nor lose the common touch,
If neither foes nor loving friends can hurt you,
If all men count with you, but none too much;
If you can fill the unforgiving minute
With sixty seconds' worth of distance run,
Yours is the Earth and everything that's in it,
And-which is more-you'll be a Man, my son!



COLORS VIA A GRAYSCALE IMAGE :)

2 sha1-colliding PDFs with vector content stored as lossless JPEG data.

WE'VE SEEN SO FAR....

JPEG as image,
JPEG as data...

Why not both?

Lossless data **and** lossy image

- Pad data to match image width
- Store 8 times to make lossless
- Append image

A page content can reference itself

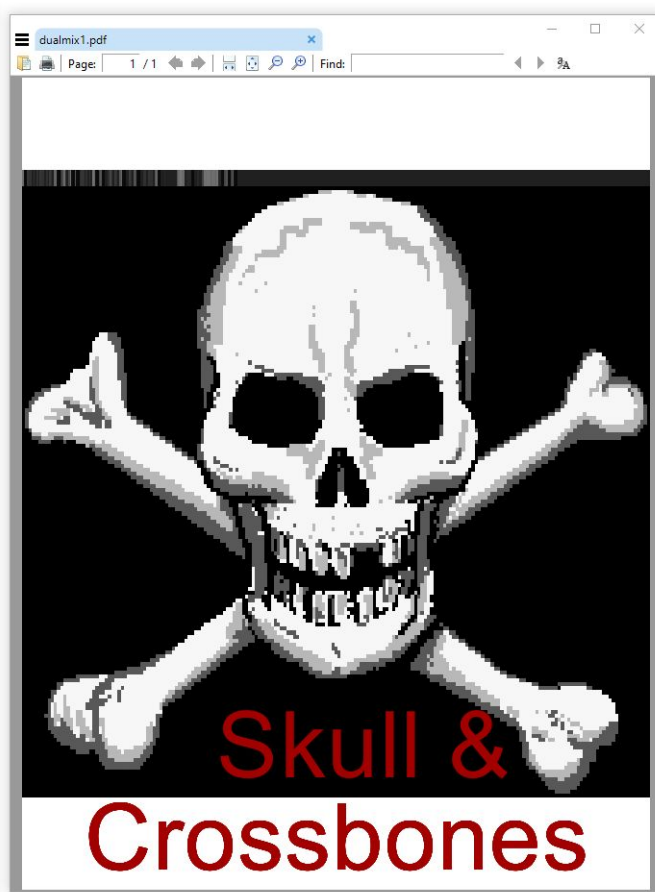
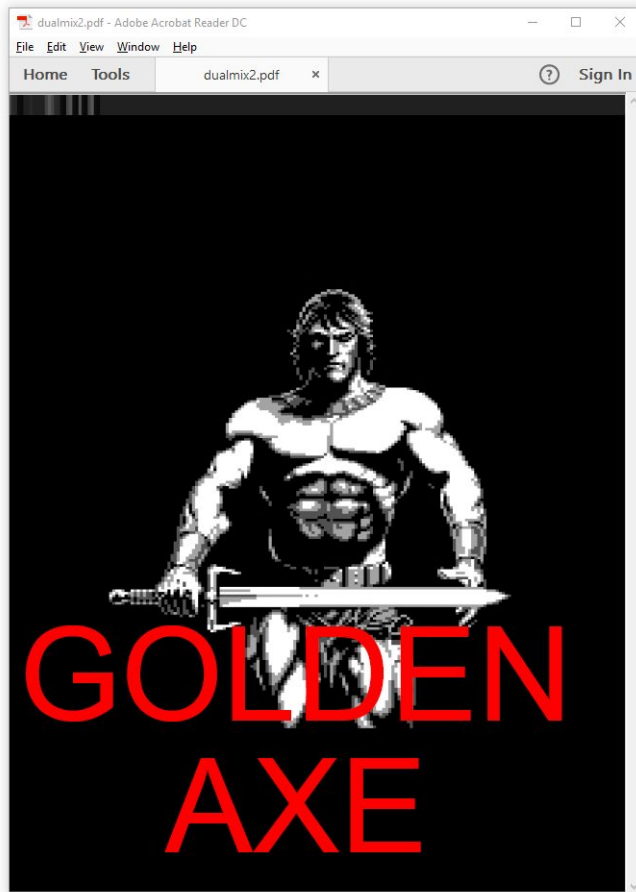
No page content terminator :(

⇒ lossy data could fail rendering - YMMV



Standard Page code + padding
showing (itself as) an image
Displaying text

```
q
612 0 0 792 0 0 cm
/Im1 Do
Q
1 0 0 rg
BT
/F1 90 Tf
10 400 Td
(GOLDEN AXE) Tj
ET
Q
```

2 sha1-colliding PDFs with mixed JPEG (on different readers)

de9b4237c940ec4af249f2c80bcd841537f6624c

Shattered:
one blocks pair,
many kinds of PoCs!

Trivial to detect at file level,
tricky to detect at rendering level.

MD5?!

It's already broken!
Nothing to see here, right?

Multi-collision files

Why create only a **pair** of colliding files
when you can create 2^{609} ?

$2^{609} =$

212455197126706839475835282620987450931837247090812769279777655280161423944340897095665
000906091714267555731794498600406138631735061082895763807991506634940777532508334157287
6126912512

(184 digits)

What's a collision?

Variable content, same hash

*MAKE YOUR FILE'S CONTENT UPDATABLE
WITHOUT CHANGING THE FINAL HASH.*

Hashquine

Display your own file's hash

It's a mental trick:

"how do you know the hash in advance?"

Fake hashquine

Actually a script that computes
and display its own hash

Often comes with obfuscation ;)

Format hashquine

1 passive collision \Rightarrow take this file or skip to the next.

X collisions $\Rightarrow X+1$ versions of the same element.

1. Store multiple versions of visual elements in a chain of collisions.
2. Display the file hash in the file.

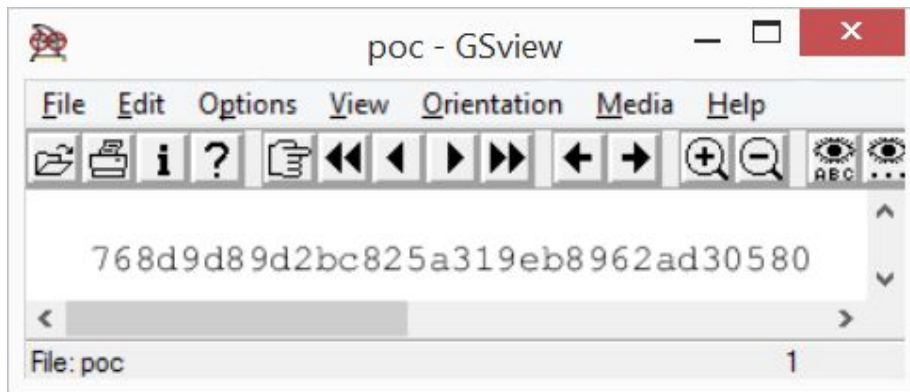
Data Hashquine

1 collision == 2 alternate contents \Rightarrow 1 bit of data.

Put some code that parses the bits and displays the stored value.

More collision efficient than format hashquines,
but requires code to be executed.

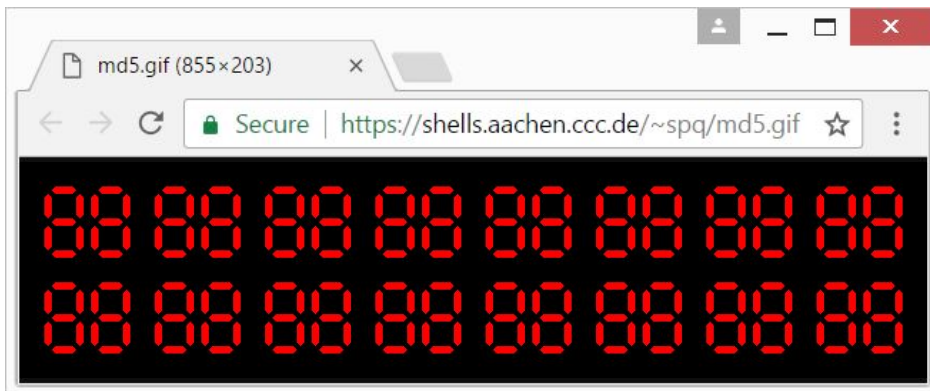
CHEATING?

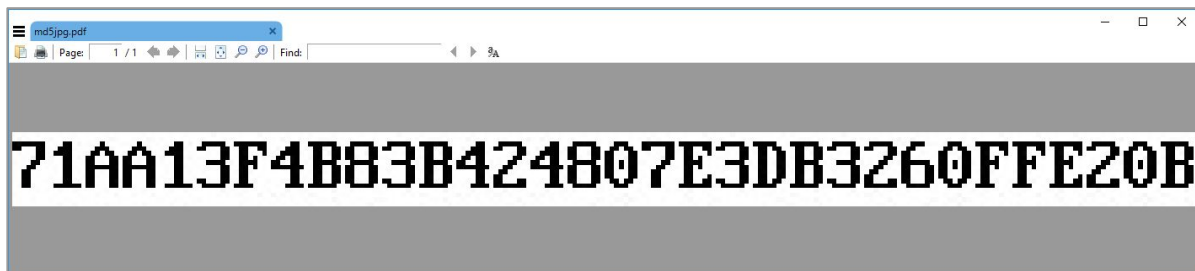


PostScript by Greg

THE FIRST EVER!

ANIMATED
GIFs by spq





As images

PDFs by Mako



As text

```
$ pdftotext -q md5text.pdf -  
66DA5E07C0FD4C921679A65931FF8393  
$ md5sum md5text.pdf  
66da5e07c0fd4c921679a65931ff8393 md5text.pdf
```


Very nice [writeup](#) for GIF



GIF & TIFF, by Rogdham

bit-hashquine TIFF with writeup, but 4 Gb !

What is a hashquine?

"Hashquine" is a term coined by foone meaning "file that show their own hash".

How to read the md5 from this image?

The centre of the image sets the md5 one bit at a time:

- ☐ 13 means a bit 0 at position 0x13
- ☒ 37 means a bit 1 at position 0x37

How did you do it?

An overview of this file structure is drawn on the right of this image.

For more details, look at www.rogdham.net, I may have posted an article there explaining everything together with the source code to generate it.

Why is this file so HUGE?

This is due to the way I made the hashquine. I chose to use the generated md5 collision blocks as offset to tiles. TIFF offsets are 32 bits unsigned integers counted from the beginning of the file. I did not chose the collision blocks, so offsets are up to 2^{32} which is 4Go. Sorry!

Is this a valid TIFF file?

Unless I made a mistake anywhere, it should be! However, I used tiles instead of strips, so your reader needs to understand TIFF 6.0.

At some point I wanted to make the image size (in pixels) smaller, by using smaller tiles. However, tiles widths and heights must be multiple of 16 pixels, so I decided to use 16×16 pixels tiles.

File structure overview

IDH	IDF
#tags	#tags
44 01 04 00	44 01 04 00
00 10 00 00	00 10 00 00
*TileOffset	*TileOffset
00 00 00 00	00 00 00 00

TileOffset is an array of unsigned integers pointing to the tiles of the image

Collision blocks are used as the content of the TileOffset array; here are two collisions blocks side by side, with the tiles each integer points to:

ee a6 ac fe	ee a6 ac fe
86 91 0e 72	86 91 0e 72
7b c5 d5 06	7b c5 d5 06
91 79 e8 c9	91 79 e8 c9
b8 f8 53 26	b8 f8 53 26
5d 11 25 36	5d 11 25 36
8e 1e a9 84	8e 1e a9 84
9c 71 eb 45	9c 71 eb 45
d3 76 e4 b3	d3 76 e4 b3
09 58 48 18	09 58 48 18
c1 eb f9 d8	c1 eb f9 d8
7b b3 82 fe	7b b3 82 fe
e6 f2 5b fd	e6 f2 5b fd
8f 1b 80 55	8f 1b 80 55
85 a9 77 cc	85 a9 77 cc
64 dd 58 bd	64 dd 58 bd
99 a4 b3 37	99 a4 b3 37
38 bb 1c 5e	38 bb 1c 5e
5b 45 90 93	5b 45 90 93
60 e1 45 b8	60 e1 45 b8
56 d0 45 45	56 d0 45 45
7b 2b 56 21	7b 2b 56 21
5b e1 c9 14	5b e1 c9 14
69 58 22 7d	69 58 22 7d
6f cb e2 df	6f cb e2 df
c7 69 84 8e	c7 69 84 8e
57 63 8f b4	57 63 8f b4
49 71 6a 7f	49 71 6a 7f
0e 88 33 c4	0e 88 33 c4
c7 8e 63 5e	c7 8e 63 5e
4e f9 49 46	4e f9 49 46
22 4a 27 ee	22 4a 27 ee

PoC||GTFO 0x14

Articles about hashquines.
But also hashquine itself,
and polyglot!

14:09 MD5 Postscript

14:10 MD5 PDF

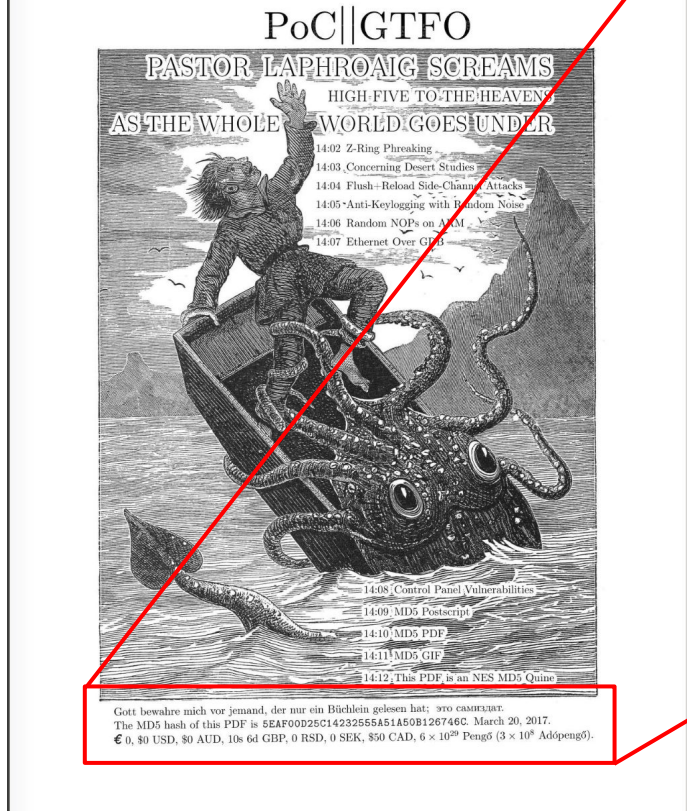
14:11 MD5 GIF

14:12 This PDF is an NES MD5 Quine

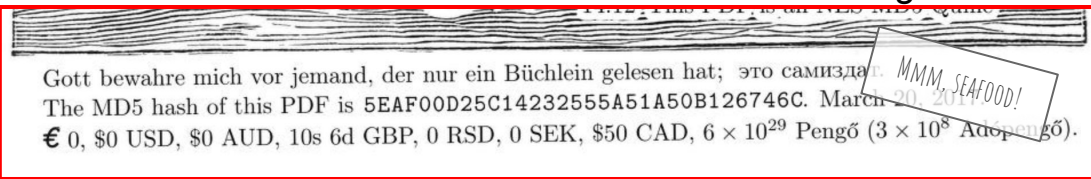
by Evan² and Philippe

...showing its MD5...

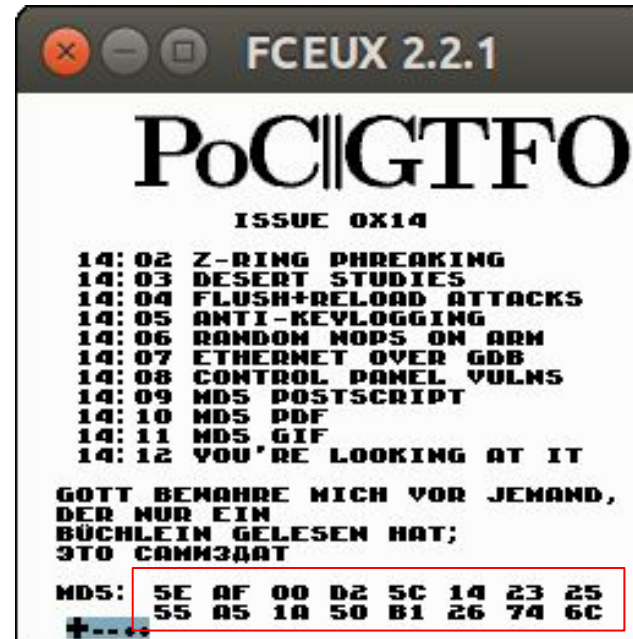
A LaTeX-generated
PDF...



(15x32=480 collisions)



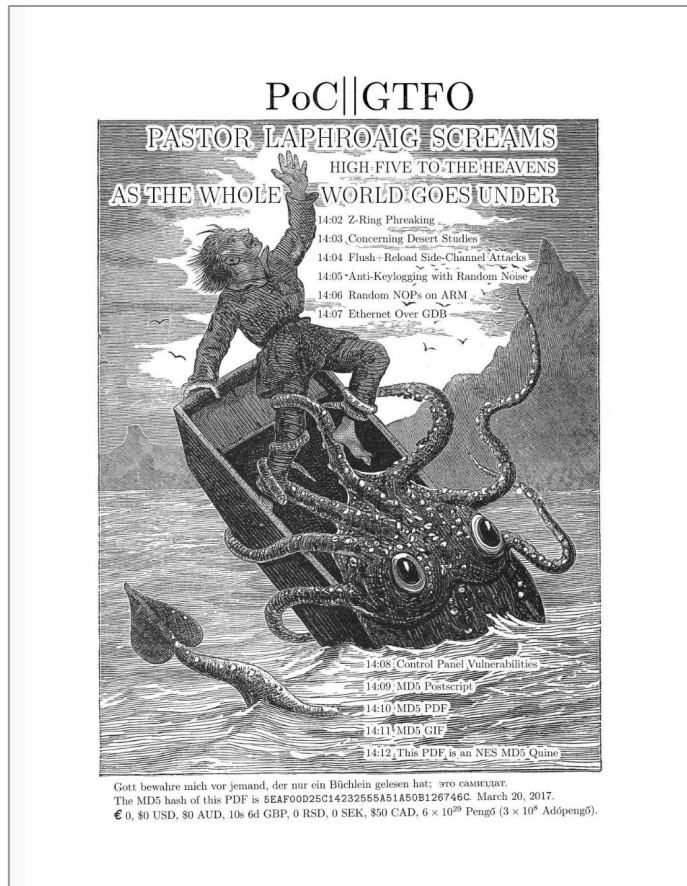
...also a NES rom...



...showing the same MD5!

(4x32=128 collisions)

608?



1 extra collision \Rightarrow hidden cover, same MD5.

THANKS MARC!

You know
a cryptographic hash
is **really** broken
when it feels like
a fancy fidget spinner.

When you generate 609 of its collisions *for fun*.

In total, 9824 collisions were computed for the making of this issue.



Other formats?

Certificates, PNG...



x.509v3 SSL Certificate

As Defined in the
ITU-T Recommendation x.509

VERY RESTRICTIVE!

```

-----BEGIN CERTIFICATE-----
MIIBdTCCAS+gAwIBAgICEzcwDQYJKoZIhvcNAQEFBQAwJDENMAsGA
1UEAwEUM9vdETMBEGA1UECgwKUM9vdHMgSW5lLjAeFw0xNTA0MT
UwNDUwMTZaFw0xNTA0MTQwNDUwMTZaME4xCzAJBgNVBAYTA1VTMQ0
wCwYDVQQIDARPaGlwMQ8wDQYDVQQKDAZDaXRS5IElxdzANBgNVBAsM
B1VuaXQgQ3RjEOMAwGA1UEAwF1Y15jb29wT0ANBgkqhkiG9w0BAQEF
AAMA7ADAAMjEArDZT1puvf1ZarB8bpX59EsdjdsShd7ebd1JR4
MuyRNVcRgUTr2+bzzh4MFpAgMBAAGjMTAvMAwGA1UdEwEB/wQCMAA
wHwYDVORjBBgwFoAUUy6uqAhdnq2p5W0G1mrT/ZRm4wwDQYJKoZI
hvcNAQEFBQADMBmfEdSwOSDUEYr7ia+N1u1sJS5/GBzoCAXBxxau
V8PxVbZZDpTae4fh/yJC0X3/OI=
-----END CERTIFICATE-----

```

```

000:  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
      30 02 01 75      30 02 01 2F  A0 03  02 01 02  02 02 13
010:  37 30 00  06 09 2A 86 48 86 F7 0D 01 01 05  05 00
      30 1E
050:  30 34 35 30 31 36 5A      17 0D 31 35 30 31 31 35
060:  34 35 30 31 36 5A      17 0D 31 35 30 31 34 30
      30 1F
110:  01 FF  04 02  30 00  30 1F  06 03 55 1D 23  04 18  30
      30 0D  06 09 2A 86 48 86 F7
140:  0D 01 01 05  05 00  03 31 00 66 7C 47 52 C0 E4 83
      2C 40 05 7C 5A B9 5F 0F C5 56 D9 64 3A 48 69 EE
150:  50 46 2B EE 26 BE 37 5B B5 B2 34 B9 FC 60 73 A0
160:  2C 40 05 7C 5A B9 5F 0F C5 56 D9 64 3A 48 69 EE
170:  1F 87 FC 89 08 E5 C9 FC E2

```

373 Bytes [certificate]

303 Bytes [tbsCertificate]

3 Bytes [0]

1 Byte [Version] 3

2 Bytes [serial number] 4919

13 Bytes [signatureID]

9 Bytes [sha1WithRSAEncryption] 1.2.840.113549.1.1.5

0 Bytes [null]

36 Bytes [issuer] CN=Root, O=Roots Inc.

30 Bytes [validity]

13 Bytes [notBefore] 2015-01-15 04:50:16 UTC

13 Bytes [notAfter] 2015-07-14 04:50:16 UTC

78 Bytes [subject] C=US, ST=Ohio, O=City B, OU=Unit B, CN=b.com

76 Bytes [subjectPublicKeyInfo] [rsaEncryption] 1.2.840.113549.1.1.1

SEE DN
[modulus] 2650597835409943238585424094982081002591172890993985557600
6559733627078272702522774997635806320016501911976396507087

SEE
PKCS#1 [exponent] 65537

49 Bytes [extension block]

47 Bytes [extensions]

12 Bytes [x.509 extension]

3 Bytes [Basic Constraints] 2.5.29.19

1 Byte [critical] true

2 Bytes [isCA, pathLengthConstraints]

0 Bytes [empty] Not a CA, No Path Constraints

31 Bytes [x.509 extension]

3 Bytes [authorityKeyIdentifier] 2.5.29.35

24 Bytes

22 Bytes [keyIdentifier]

20 Bytes [0] 572EAE8085A767AB6A79C0E1B59AB4FF6519B8C

13 Bytes [signatureAlgorithmID]

9 Bytes [sha1WithRSAEncryption] 1.2.840.113549.1.1.5

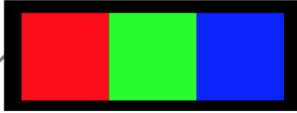
0 Bytes [null]

49 Bytes [signatureValue].f|GR...PF+.&.7[.4..`s.,@.|Z...V.d:Hi.....

ASN.1
Types
xx Bytes

30 xx Sequence 17 xx UTC Time
 02 xx Integer 01 01 Boolean
 06 xx OID 04 xx Octet String
 05 00 NULL 03 xx Bit String

<https://www.cem.me/pki/index.html>



```

0 1 2 3 4 5 6 7 8 9 A B C D E F
00: 89 .P .N .G 0D 0A 1A 0A 00 00 00 0D .I .H .D .R
10: 00 00 00 03 00 00 00 01 08 02 00 00 00 94 82 83
20: E3 00 00 00 15 .I .D .A .T 08 1D 01 0A 00 F5 FF
30: 00 FF 00 00 00 FF 00 00 00 FF 0E FB 02 FE E9 32
40: 61 E5 00 00 00 00 .I .E .N .D AE 42 60 82

```

PNG

Strengths:

- 8 byte signature
- Chunk types after lengths
- 4 byte lengths
- Chunk CRCs

Weaknesses:

- Easy to make ignored chunks
- CRC usually ignored

SIGNATURE		FIELDS	VALUES
HEADER		signature	\x89 PNG \r\n \x1a \n
		size	0x0000000D
		id	IHDR
		width	0x00000003
		height	0x00000001
		bpp	0x08
		color	0x02 RGB
		compression	0x00 DEFLATE
		filter	0x00
		interlace	0x00
DATA		CRC32	0x948283E3
		size	0x00000015
		id	IDAT
		ZLIB window size	0b00001000
		method	0b00001000 DEFLATE
		level / dict.	0b00011101
		checksum	0x081D % 31 = 0
		DEFLATE last block	0b00000001 FINAL
		block type	0b00000001 RAW
		data length	0x000A
END		!length	0xFFFF5
		PIXELS line filter	0x00 NONE
			FF 00 00 00 FF 00 00 00 FF
		adler32	0x0EFB02FE
		CRC32	0xE93261E5
		size	0x00000000
		id	IEND
		CRC32	0xAE426082

Attack \Leftrightarrow format pairing

Hash collision attack \Rightarrow constraints (prefix, mask)

File format \Rightarrow other constraints (structure, compatibility)

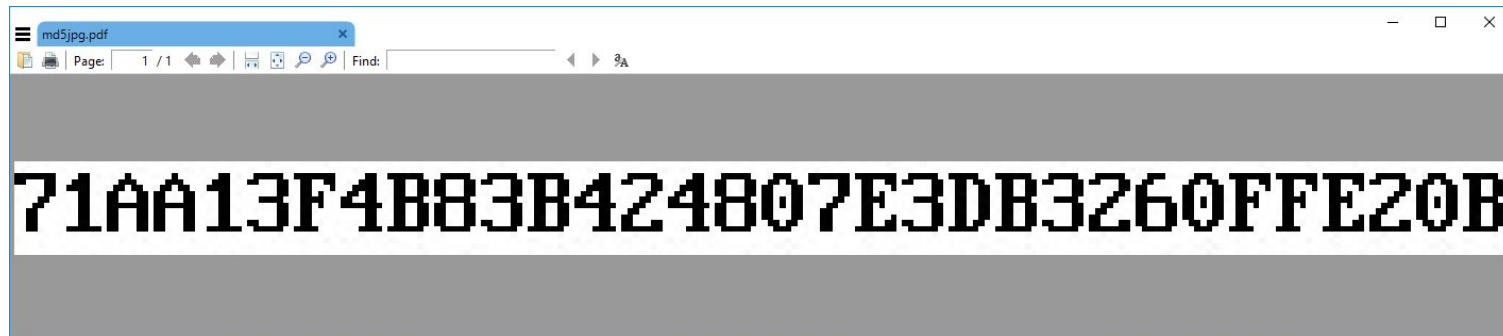
The same attack can be used with various file formats.

A file format trick can be used with different hashes.

Mako's PDF Hashquine with MD5

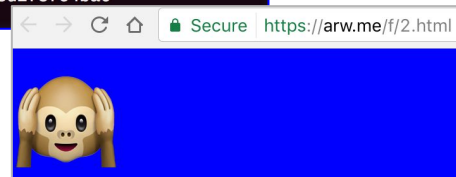
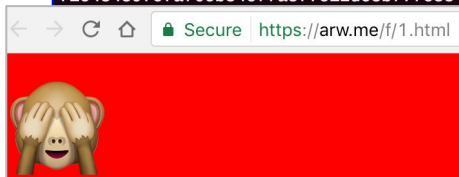
MalSHA1's JPEG trick + Shattered JPEG in PDF trick for SHA1

SHA'1 \Rightarrow SHA1 \Rightarrow MD5



@arw's HTML colliding [pair](#) made with Shattered prefix.
PDF \Rightarrow HTML (also works as [polyglot](#))

```
arw$ for i in 1 2; do curl -s https://arw.me/f/$i.html | shasum -a 1; done
ba97502d759d58f91ed212d7c981e0cfd7b70eef -
ba97502d759d58f91ed212d7c981e0cfd7b70eef -
arw$ for i in 1 2; do curl -s https://arw.me/f/$i.html | shasum -a 256; done
4477a514fa5e948d69e064a4e00378c69262e32e36c079b76226ae50e3d312cf -
71c484897c7af6cb34cffa8f7c12dc3bf7fc834ed7f57123e21258d2f3fc4ba6 -
```



Why?

"It's just a bag of tricks anyway..."

"Crypto doesn't care about PoCs..."

Attacks rely on PoCs.

Attacks convince people to deprecate.

You don't get pwned by academic papers, but by their PoCs.

A new format trick could benefit MD5, SHA1...
or a future attack!

IN PRACTICE,

- SHATTERED GENERATES AN INFINITY OF COLLIDING DOCUMENTS, OF DIFFERENT KINDS.

- SHATTERED BROKE SVN.

DIDN'T THAT HELP?

...the end?

...we still have a few tricks up our sleeves ;)

Conclusion

- Hash collisions exploitation is a niche domain: weird constraints, unusual challenges & rewards.
- Researching a file format manipulation **now** could benefit on a **future** cryptographic attack.

FWIW (full personal disclosure)

- When I was asked about MalSHA1, I saw no solution.
 - I gave up for a while - I didn't think particularly about JPEG.
- In the meantime, I was challenged to encrypt with AES a JPEG to a JPEG.
⇒ [AngeCrypton](#)
- With that knowledge, I succeeded for MalSHA1.
- That knowledge was the starting point for Shattered.
 - I gave up at some time on the JPEG optimization aspect.
 - But I kept that fidget spinning playfully.
 - Found my 2 breakthroughs... in very unexpected places ;)

Don't give up! Keep that fidget spinning!

"How do you do all this?"

- I thought I lacked discipline. That led me nowhere.
- Just do what makes you giggle like a 3-year old.
(that's what playing with file formats does to me).
- Have fun! Eventually you'll get feedback, recognition...
- By then, you'll have no reasons to stop anymore.
- And you'll be happily disciplined by then.

Have fun!

Thanks for your attention!

Questions?

Special thanks to Marc & Maria
Philippe, Evan, spq, Mako, Greg, Melissa,
Elie, Jean-Philippe, and [CommitStrip](https://commitstrip.com).

ANGE ALBERTINI
reverse engineering
VISUAL DOCUMENTATIONS

[@angealbertini](https://twitter.com/angealbertini)
ange@corkami.com
<http://www.corkami.com>

