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
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. Becker¹, Francesco Regazzoni², Christof Paar¹,³, and Wayne P. Burleson¹

Trojan Side Channels
Lightweight Hardware Trojans through Side Channel Engineering

Lang Lin¹  Markus Kasper²  Tim Güneysu²  Christof Paar¹,²  Wayne Burleson¹
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 espionage, or software piracy.

2011: theoretical framework, but nothing useful
what's a crypto backdoor?
not an implementation backdoor

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
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"

randomness → generate() → hash function $H$ → backdoor $b$

hash function $H$ → exploit() → collision/preimage

backdoor $b$ → exploit() → collision/preimage

challenge → exploit() → collision/preimage
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

hash function $H$ → detect() → exploit() ?

hash function $H$ → discover() → backdoor $b$

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 everywhere

RSA-OAEP, “RSAwithSHA1”, HMAC, PBKDF2, etc.
⇒ in TLS, SSH, IPsec, etc.

integrity check: git, bootloaders, HIDS/FIM, etc.
SHA-1

\[(W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \ll 1 \quad \text{for} \quad 16 \leq i \leq 79\]

<table>
<thead>
<tr>
<th>step (i)</th>
<th>(K_r)</th>
<th>(f_r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 \leq i \leq 19</td>
<td>5a827999</td>
<td>(f_{IF}(B, C, D) = B \land C \oplus \neg B \land D)</td>
</tr>
<tr>
<td>20 \leq i \leq 39</td>
<td>6ed9eba1</td>
<td>(f_{XOR}(B, C, D) = B \oplus C \oplus D)</td>
</tr>
<tr>
<td>40 \leq i \leq 59</td>
<td>8f1bbc6c</td>
<td>(f_{MAJ}(B, C, D) = B \land C \oplus B \land D \oplus C \land D)</td>
</tr>
<tr>
<td>60 \leq i \leq 79</td>
<td>ca62c1d6</td>
<td>(f_{XOR}(B, C, D) = B \oplus C \oplus D)</td>
</tr>
</tbody>
</table>
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 Wang¹*, Yiqun Lisa Yin², and Hongbo Yu³

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 (>2⁶₀)
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

low-probability

high-probability

$2^{-40}$

$2^{-15}$
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!

<table>
<thead>
<tr>
<th>K1...4</th>
<th>5a827999</th>
<th>4eb9d7f7</th>
<th>bad18e2f</th>
<th>d79e5877</th>
</tr>
</thead>
<tbody>
<tr>
<td>IV</td>
<td>67452301</td>
<td>efcdab89</td>
<td>98badcfe</td>
<td>10325476</td>
</tr>
<tr>
<td>m</td>
<td>ffd8ffe1</td>
<td>e2001250</td>
<td>b6cef608</td>
<td>34f4fe83</td>
</tr>
<tr>
<td></td>
<td>1bd3283</td>
<td>b48c11bc</td>
<td>b1d4b511</td>
<td>a976cb20</td>
</tr>
<tr>
<td>m*</td>
<td>ffd8ffe2</td>
<td>c2001224</td>
<td>3ecef608</td>
<td>dcf4fee1</td>
</tr>
<tr>
<td></td>
<td>931d3281</td>
<td>b48c11a8</td>
<td>b9d4b513</td>
<td>0976cb74</td>
</tr>
<tr>
<td>Δm</td>
<td>00000003</td>
<td>20000074</td>
<td>88000000</td>
<td>e8000062</td>
</tr>
<tr>
<td></td>
<td>88000002</td>
<td>00000140</td>
<td>08000002</td>
<td>a0000054</td>
</tr>
<tr>
<td>h(m)</td>
<td>1896b202</td>
<td>394b0aae</td>
<td>54526cfa</td>
<td>e72ec5f2</td>
</tr>
</tbody>
</table>

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

⇒ 4-byte signatures corrupted
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?
<check for block’s content>
```bash
#!/bin/sh

# Check if the sum of the first and last elements
# of the array is divisible by 101

if [ $(($1 + $7)) -eq 101 ]; then
    echo "Hello World."
else
    echo "Goodbye!
"
fi
```
```bash
#!/bin/bash

#!/bin/bash

if [ `od -t x1 -j3 -N1 -An "${0}" -r -eq "91" ];
then . echo "(____)\n(oo)\n------\\/\\/\n \|
^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n^n

then . echo "Hello World."
else . echo "\"Hello World.\"
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 1\textsuperscript{st} 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

2-byte signature 0xFFD8

sequence of chunks

idea

message 1: first chunk “commented”
message 2: first chunk processed
JPEG signature Chunk marker Chunk length
- ff e5 in block 1 - c4 00 in block 1
- ff e6 in block 2 - 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 <COMmment 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!
more magic: just 2 files here

Schizophrenics
different contents with different tools

Fraternal twins
hash collision

RAR
Booting from Floppy... MBR
good!

Polyglots
multiple file formats

Booter Shell

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

> msha1.py mbr_shell_rar*.pdf 5a827999 82b1c71a 5141963a b389abb9
mbr_shell_rar0.pdf 10382a6d3c949408d7cafaaf6d110a9e23230416
mbr_shell_rar1.pdf 10382a6d3c949408d7cafaaf6d110a9e23230416

> msha1.py jpg-rar*.jpg 5a827999 9b73a440 71599fc5 0c8a53e4
jpg-rar0.jpg 7a00042714d8ee6f4978193b07df705b652d0e39
jpg-rar1.jpg 7a00042714d8ee6f4978193b07df705b652d0e39

more magic: just 2 files here
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 Tanoue
4 An Easter Egg in PCI Express
   Jacob Torve
5 A Flash PDF polyglot
   Alex Inführ
6 These Philosophers Stuff on 512 Bytes; or,
   This Multiprocessing OS is a Boot Sector.
   Shikha 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!
   Mattida
9 From Protocol to PoC; or,
   Your Cisco blade is booting PoC||GTFO.
   Måk
10 x86 Shellecode for Lazy Neighbors; or,
   I am my own NOP Sled.
   Brainshock
11 Abusing JSONP with Rosetta Flash
   Michele Scarsione
12 A cryptographer and a binarista walk into a bar
   Angela Albertini and Maria Eichbecker
13 Ancestral Voices
   Or, a vision in a nightmare.
   Ben Nagy
14 A Call for PoC
   Rvd. Dr. Manul Laphroaig

PDF
ZIP
ISO
FLASH

Never gonna give you up
Never gonna let you down
Never gonna run around and

Conclusions

BACKDOORS

BACKDOORS EVERYWHERE
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 ($\sqrt{2} \sqrt{3} \sqrt{5} \sqrt{10}$)
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?