WikipediA

SHA-1

In cryptography, **SHA-1** (**Secure Hash Algorithm 1**) is a cryptographic hash function which takes an input and produces a 160-bit (20-byte) hash value known as a message digest – typically rendered as a hexadecimal number, 40 digits long. It was designed by the United States National Security Agency, and is a U.S. Federal Information Processing Standard.[3]

Since 2005 SHA-1 has not been considered secure against well-funded opponents,[4] and since 2010 many organizations have recommended its replacement by SHA-2 or SHA-3.[5][6][7]

All major web browser vendors ceased acceptance of SHA-1 SSL certificates in 2017.[8][9][10] In February 2017, CWI Amsterdam and Google announced they had performed a collision attack against SHA-1, publishing two dissimilar PDF files which produced the same SHA-1 hash.[11][12][13]

Secure Hash Algorithms
Concepts
hash functions · SHA · DSA
Main standards
SHA-0 · SHA-1 · SHA-2 · SHA-3

SHA-1

General	
Designers	National Security Agency
First published	1993 (SHA-0), 1995 (SHA-1)
Series	(SHA-0), SHA-1, SHA-2, SHA-3
Certification	FIPS PUB 180-4, CRYPTREC (Monitored)
Cipher detail	
Digest sizes	160 bits
Block sizes	512 bits
Structure	Merkle–Damgård construction
Rounds	80
Best public cryptanalysis	
A 2011 attack by Marc Stevens can produce hash collisions with a complexity between $2^{60.3}$ and $2^{65.3}$ operations.[1] The first public collision was published on 23 February 2017.[2] SHA-1 is prone to length extension attacks.	

Contents

Development

SHA-1 produces a message digest based on principles similar to those used by Ronald L. Rivest of MIT in the design of the MD2, MD4 and MD5 message digest algorithms, but generates a larger hash value (160 bits vs. 128 bits).

SHA-1 was developed as part of the U.S. Government's Capstone project.[14] The original specification of the algorithm was published in 1993 under the title *Secure Hash Standard*, FIPS PUB 180, by U.S. government standards agency NIST (National Institute of Standards and Technology).[15][16] This version is now often named *SHA-0*. It was withdrawn by the NSA shortly after publication and was superseded by the revised version, published in 1995 in FIPS PUB 180-1 and commonly designated *SHA-1*. SHA-1 differs from SHA-0 only by a single bitwise rotation in the message schedule of its compression function. According to the NSA, this was done to correct a flaw in the original algorithm which reduced its cryptographic security, but they did not provide any further explanation. Publicly available techniques did indeed demonstrate a compromise of SHA-0, in 2004, before SHA-1 in 2017. *See* #Attacks

Applications

Cryptography

SHA-1 forms part of several widely used security applications and protocols, including TLS and SSL, PGP, SSH, S/MIME, and IPsec. Those applications can also use MD5; both MD5 and SHA-1 are descended from MD4.

SHA-1 and SHA-2 are the hash algorithms required by law for use in certain U.S. government applications, including use within other cryptographic algorithms and protocols, for the protection of sensitive unclassified information. FIPS PUB 180-1 also encouraged adoption and use of SHA-1 by private and commercial organizations. SHA-1 is being retired from most government uses; the U.S. National Institute of Standards and Technology said, "Federal agencies ***should*** stop using SHA-1 for...applications that require collision resistance as soon as practical, and must use the SHA-2 family of hash functions for these applications after 2010" (emphasis in original),[17] though that was later relaxed to allow SHA-1 to be used for verifying old digital signatures and time stamps.[18]

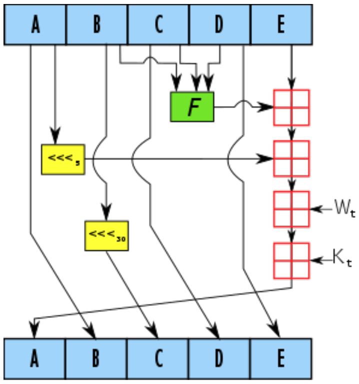
A prime motivation for the publication of the Secure Hash Algorithm was the Digital Signature Standard, in which it is incorporated.

The SHA hash functions have been used for the basis of the SHACAL block ciphers.

Data integrity

Revision control systems such as Git, Mercurial, and Monotone use SHA-1 not for security but to identify revisions and to ensure that the data has not changed due to accidental corruption. Linus Torvalds said about Git:

> If you have disk corruption, if you have DRAM corruption, if you have any kind of problems at all, Git will notice them. It's not a question of *if*, it's a guarantee. You can have people who try to be malicious. They won't succeed. ... Nobody has been able to break SHA-1, but the point is the SHA-1, as far as Git is concerned, isn't even a security feature. It's purely a consistency check. The security parts are elsewhere, so a lot of people assume that since Git uses SHA-1 and SHA-1 is used for cryptographically secure stuff, they think that, Okay, it's a huge security feature. It has nothing at all to do with security, it's just the best hash you can get. ...
> I guarantee you, if you put your data in Git, you can trust the fact that five years later, after it was converted from your hard disk to DVD to whatever new technology and you copied it along, five years later you can verify that the data you get back out is the exact same data you put in. ...
> One of the reasons I care is for the kernel, we had a break in on one of the BitKeeper sites where people tried to corrupt the kernel source code repositories.[19] However Git does not require the second preimage resistance of SHA-1 as a security feature, since it will always prefer to keep the earliest version of an object in case of collision, preventing an attacker from surreptitiously overwriting files.[20]



One iteration within the SHA-1 compression function:
A, B, C, D and E are 32-bit words of the state;
F is a nonlinear function that varies;
⋘$_n$ denotes a left bit rotation by *n* places;

n varies for each operation;
W$_t$ is the expanded message word of round t;
K$_t$ is the round constant of round t;
⊞ denotes addition modulo 2^{32}.

Cryptanalysis and validation

For a hash function for which *L* is the number of bits in the message digest, finding a message that corresponds to a given message digest can always be done using a brute force search in approximately 2^L evaluations. This is called a preimage attack and may or may not be practical depending on *L* and the particular computing environment. However, a *collision*, consisting of finding two different messages that produce the same message digest, requires on average only about $1.2 \times 2^{L/2}$ evaluations using a birthday attack. Thus the strength of a hash function is usually compared to a symmetric cipher of half the message digest length. SHA-1, which has a 160-bit message digest, was originally thought to have 80-bit strength.

In 2005, cryptographers Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu produced collision pairs for SHA-0 and have found algorithms that should produce SHA-1 collisions in far fewer than the originally expected 2^{80} evaluations.[21]

Some of the applications that use cryptographic hashes, like password storage, are only minimally affected by a collision attack. Constructing a password that works for a given account requires a preimage attack, as well as access to the hash of the original password, which may or may not be trivial. Reversing password encryption (e.g. to obtain a password to try against a user's account elsewhere) is not made possible by the attacks. (However, even a secure password hash can't prevent brute-force attacks on weak passwords.)

In the case of document signing, an attacker could not simply fake a signature from an existing document: The attacker would have to produce a pair of documents, one innocuous and one damaging, and get the private key holder to sign the innocuous document. There are practical circumstances in which this is possible; until the end of 2008, it was possible to create forged SSL certificates using an MD5 collision.[22]

Due to the block and iterative structure of the algorithms and the absence of additional final steps, all SHA functions (except SHA-3[23]) are vulnerable to length-extension and partial-message collision attacks.[24] These attacks allow an attacker to forge a message signed only by a keyed hash—SHA(*message* ‖ *key*) or SHA(*key* ‖ *message*)—by extending the message and recalculating the hash without knowing the key. A simple improvement to prevent these attacks is to hash twice: SHA$_d$(*message*) = SHA(SHA(0^b ‖ *message*)) (the length of 0^b, zero block, is equal to the block size of the hash function).

Attacks

In early 2005, Rijmen and Oswald published an attack on a reduced version of SHA-1—53 out of 80 rounds—which finds collisions with a computational effort of fewer than 2^{80} operations.[25]

In February 2005, an attack by Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu was announced.[26] The attacks can find collisions in the full version of SHA-1, requiring fewer than 2^{69} operations. (A brute-force search would require 2^{80} operations.)

The authors write: "In particular, our analysis is built upon the original differential attack on SHA-0, the near collision attack on SHA-0, the multiblock collision techniques, as well as the message modification techniques used in the collision search attack on MD5. Breaking SHA-1 would not be possible without these powerful analytical techniques."[27] The authors have presented a collision for 58-round SHA-1, found with 2^{33} hash operations. The paper with the full attack description was published in August 2005 at the CRYPTO conference.

In an interview, Yin states that, "Roughly, we exploit the following two weaknesses: One is that the file preprocessing step is not complicated enough; another is that certain math operations in the first 20 rounds have unexpected security problems."[28]

On 17 August 2005, an improvement on the SHA-1 attack was announced on behalf of Xiaoyun Wang, Andrew Yao and Frances Yao at the CRYPTO 2005 Rump Session, lowering the complexity required for finding a collision in SHA-1 to 2^{63}.[29] On 18 December 2007 the details of this result were explained and verified by Martin Cochran.[30]

Christophe De Cannière and Christian Rechberger further improved the attack on SHA-1 in "Finding SHA-1 Characteristics: General Results and Applications,"[31] receiving the Best Paper Award at ASIACRYPT 2006. A two-block collision for 64-round SHA-1 was presented, found using unoptimized methods with 2^{35} compression function evaluations. Since this attack requires the equivalent of about 2^{35} evaluations, it is considered to be a significant theoretical break.[32] Their attack was extended further to 73 rounds (of 80) in 2010 by Grechnikov.[33] In order to find an actual collision in the full 80 rounds of the hash function, however, tremendous amounts of computer time are required. To that end, a collision search for SHA-1 using the distributed computing platform BOINC began August 8, 2007, organized by the Graz University of Technology. The effort was abandoned May 12, 2009 due to lack of progress.[34]

At the Rump Session of CRYPTO 2006, Christian Rechberger and Christophe De Cannière claimed to have discovered a collision attack on SHA-1 that would allow an attacker to select at least parts of the message.[35][36]

In 2008, an attack methodology by Stéphane Manuel reported hash collisions with an estimated theoretical complexity of 2^{51} to 2^{57} operations.[37] However he later retracted that claim after finding that local collision paths were not actually independent, and finally quoting for the most efficient a collision vector that was already known before this work.[38]

Cameron McDonald, Philip Hawkes and Josef Pieprzyk presented a hash collision attack with claimed complexity 2^{52} at the Rump Session of Eurocrypt 2009.[39] However, the accompanying paper, "Differential Path for SHA-1 with complexity $O(2^{52})$" has been withdrawn due to the authors' discovery that their estimate was incorrect.[40]

One attack against SHA-1 was Marc Stevens[41] with an estimated cost of $2.77M to break a single hash value by renting CPU power from cloud servers.[42] Stevens developed this attack in a project called HashClash,[43] implementing a differential path attack. On 8 November 2010, he claimed he had a fully working near-collision attack against full SHA-1 working with an estimated complexity equivalent to $2^{57.5}$ SHA-1 compressions. He estimated this attack could be extended to a full collision with a complexity around 2^{61}.

The SHAppening

On 8 October 2015, Marc Stevens, Pierre Karpman, and Thomas Peyrin published a freestart collision attack on SHA-1's compression function that requires only 2^{57} SHA-1 evaluations. This does not directly translate into a collision on the full SHA-1 hash function (where an attacker is *not* able to freely choose the initial internal state), but undermines the security claims for SHA-1. In particular, it was the first time that an attack on full SHA-1 had been *demonstrated*; all earlier attacks were too expensive for their authors to carry them out. The authors named this significant breakthrough in the cryptanalysis of SHA-1 *The SHAppening*.[6]

The method was based on their earlier work, as well as the auxiliary paths (or boomerangs) speed-up technique from Joux and Peyrin, and using high performance/cost efficient GPU cards from NVIDIA. The collision was found on a 16-node cluster with a total of 64 graphics cards. The authors estimated that a similar collision could be found by buying US$2,000 of GPU time on EC2.[6]

The authors estimated that the cost of renting enough of EC2 CPU/GPU time to generate a full collision for SHA-1 at the time of publication was between US$75K and 120K, and noted that was well within the budget of criminal organizations, not to mention national intelligence agencies. As such, the authors recommended that SHA-1 be deprecated as quickly as possible.[6]

SHAttered – first public collision

On 23 February 2017, the CWI (Centrum Wiskunde & Informatica) and Google announced the *SHAttered* attack, in which they generated two different PDF files with the same SHA-1 hash in roughly $2^{63.1}$ SHA-1 evaluations. This attack is about 100,000 times faster than brute forcing a SHA-1 collision with a birthday attack, which was estimated to take 2^{80} SHA-1 evaluations. The attack required "the equivalent processing power of 6,500 years of single-CPU computations and 110 years of single-GPU computations".[2][13]

Birthday-Near-Collision Attack - first practical chosen-prefix attack

On 24 April 2019 a paper by Gaëtan Leurent and Thomas Peyrin presented at Eurocrypt 2019 described an enhancement to the previously best chosen-prefix attack in Merkle-Damgärd-like digest functions based on Davies-Meyer block ciphers. With these improvements, this method is capable of finding chosen-prefix collisions in approximately 2^{68} SHA-1 evaluations. This is approximately 1 billion times faster (and now usable for many targetted attacks, thanks to the possibility of choosing a prefix, for example malicious code or faked identities in signed certificates) than the previous attack's $2^{77.1}$ evaluations (but without chosen prefix, which was impractical for most targetted attacks because the found collisions were almost random)[44] and is fast enough to be practical for resourceful attackers, requiring approximately $100,000 of cloud processing. This method is also capable of finding chosen-prefix collisions in the MD5 function, but at a complexity of $2^{46.3}$ does not surpass the prior best available method at a theoretical level (2^{39}), though potentially at a practical level ($\leq 2^{49}$).[45][46]

SHA-0

At CRYPTO 98, two French researchers, Florent Chabaud and Antoine Joux, presented an attack on SHA-0: collisions can be found with complexity 2^{61}, fewer than the 2^{80} for an ideal hash function of the same size.[47]

In 2004, Biham and Chen found near-collisions for SHA-0—two messages that hash to nearly the same value; in this case, 142 out of the 160 bits are equal. They also found full collisions of SHA-0 reduced to 62 out of its 80 rounds.[48]

Subsequently, on 12 August 2004, a collision for the full SHA-0 algorithm was announced by Joux, Carribault, Lemuet, and Jalby. This was done by using a generalization of the Chabaud and Joux attack. Finding the collision had complexity 2^{51} and took about 80,000 processor-hours on a supercomputer with 256 Itanium 2 processors (equivalent to 13 days of full-time use of the computer).

On 17 August 2004, at the Rump Session of CRYPTO 2004, preliminary results were announced by Wang, Feng, Lai, and Yu, about an attack on MD5, SHA-0 and other hash functions. The complexity of their attack on SHA-0 is 2^{40}, significantly better than the attack by Joux *et al.*[49][50]

In February 2005, an attack by Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu was announced which could find collisions in SHA-0 in 2^{39} operations.[26][51]

Another attack in 2008 applying the boomerang attack brought the complexity of finding collisions down to $2^{33.6}$, which is estimated to take 1 hour on an average PC.[52]

In light of the results for SHA-0, some experts suggested that plans for the use of SHA-1 in new cryptosystems should be reconsidered. After the CRYPTO 2004 results were published, NIST announced that they planned to phase out the use of SHA-1 by 2010 in favor of the SHA-2 variants.[53]

Official validation

Implementations of all FIPS-approved security functions can be officially validated through the CMVP program, jointly run by the National Institute of Standards and Technology (NIST) and the Communications Security Establishment (CSE). For informal verification, a package to generate a high number of test vectors is made available for download on the NIST site; the resulting verification, however, does not replace the formal CMVP validation, which is required by law for certain applications.

As of December 2013, there are over 2000 validated implementations of SHA-1, with 14 of them capable of handling messages with a length in bits not a multiple of eight (see SHS Validation List (http://csrc.nist.gov/groups/STM/cavp/documents/shs/shaval.htm)).

Examples and pseudocode

Example hashes

These are examples of SHA-1 message digests in hexadecimal and in Base64 binary to ASCII text encoding.

```
SHA1("The quick brown fox jumps over the lazy dog")
gives hexadecimal: 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
gives Base64 binary to ASCII text encoding: L9ThxnotKPzthJ7hu3bnORuT6xI=
```

Even a small change in the message will, with overwhelming probability, result in many bits changing due to the avalanche effect. For example, changing `dog` to `cog` produces a hash with different values for 81 of the 160 bits:

```
SHA1("The quick brown fox jumps over the lazy cog")
gives hexadecimal: de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b3
gives Base64 binary to ASCII text encoding: 3p8sf9JeGzr60+haC9F9mxANtLM=
```

The hash of the zero-length string is:

```
SHA1("")
gives hexadecimal: da39a3ee5e6b4b0d3255bfef95601890afd80709
gives Base64 binary to ASCII text encoding: 2jmj7l5rSw0yVb/vlWAYkK/YBwk=
```

SHA-1 pseudocode

Pseudocode for the SHA-1 algorithm follows:

```
Note 1: All variables are unsigned 32-bit quantities and wrap modulo $2^{32}$ when calculating, except for
        ml, the message length, which is a 64-bit quantity, and
        hh, the message digest, which is a 160-bit quantity.
Note 2: All constants in this pseudo code are in big endian.
        Within each word, the most significant byte is stored in the leftmost byte position

Initialize variables:

h0 = 0x67452301
h1 = 0xEFCDAB89
h2 = 0x98BADCFE
h3 = 0x10325476
h4 = 0xC3D2E1F0

ml = message length in bits (always a multiple of the number of bits in a character).

Pre-processing:
append the bit '1' to the message e.g. by adding 0x80 if message length is a multiple of 8 bits.
append 0 ≤ k < 512 bits '0', such that the resulting message length in bits
   is congruent to −64 ≡ 448 (mod 512)
append ml, the original message length, as a 64-bit big-endian integer.
   Thus, the total length is a multiple of 512 bits.

Process the message in successive 512-bit chunks:
break message into 512-bit chunks
for each chunk
    break chunk into sixteen 32-bit big-endian words w[i], 0 ≤ i ≤ 15

    Extend the sixteen 32-bit words into eighty 32-bit words:
    for i from 16 to 79
        w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1

    Initialize hash value for this chunk:
    a = h0
    b = h1
    c = h2
    d = h3
    e = h4

    Main loop:[3][54]
    for i from 0 to 79
        if 0 ≤ i ≤ 19 then
            f = (b and c) or ((not b) and d)
            k = 0x5A827999
        else if 20 ≤ i ≤ 39
            f = b xor c xor d
            k = 0x6ED9EBA1
        else if 40 ≤ i ≤ 59
            f = (b and c) or (b and d) or (c and d)
            k = 0x8F1BBCDC
        else if 60 ≤ i ≤ 79
            f = b xor c xor d
            k = 0xCA62C1D6

        temp = (a leftrotate 5) + f + e + k + w[i]
        e = d
        d = c
        c = b leftrotate 30
        b = a
        a = temp

    Add this chunk's hash to result so far:
    h0 = h0 + a
    h1 = h1 + b
    h2 = h2 + c
    h3 = h3 + d
    h4 = h4 + e

Produce the final hash value (big-endian) as a 160-bit number:
hh = (h0 leftshift 128) or (h1 leftshift 96) or (h2 leftshift 64) or (h3 leftshift 32) or h4
```

The number `hh` is the message digest, which can be written in hexadecimal (base 16), but is often written using Base64 binary to ASCII text encoding.

The chosen constant values used in the algorithm were assumed to be nothing up my sleeve numbers:

- The four round constants `k` are 2^{30} times the square roots of 2, 3, 5 and 10. However they were incorrectly rounded to the nearest integer instead of being rounded to the nearest odd integer, with equilibrated proportions of zero and one bits. As well, choosing the square root of 10 (which is not a prime) made it a common factor for the two other chosen square roots of primes 2 and 5, with possibly usable arithmetic properties across successive rounds, reducing the strength of the algorithm against finding collisions on some bits.
- The first four starting values for `h0` through `h3` are the same with the MD5 algorithm, and the fifth and sixth (for `h4` and `h5`) are similar. However they were not

properly verified for being resistant against inversion of the few first rounds to infer possible collisions on some bits, usable by multiblock differential attacks.

Instead of the formulation from the original FIPS PUB 180-1 shown, the following equivalent expressions may be used to compute `f` in the main loop above:

```
Bitwise choice between c and d, controlled by b.
(0  ≤ i ≤ 19): f = d xor (b and (c xor d))                (alternative 1)
(0  ≤ i ≤ 19): f = (b and c) xor ((not b) and d)          (alternative 2)
(0  ≤ i ≤ 19): f = (b and c) + ((not b) and d)            (alternative 3)
(0  ≤ i ≤ 19): f = vec_sel(d, c, b)                       (alternative 4)

Bitwise majority function.
(40 ≤ i ≤ 59): f = (b and c) or (d and (b or c))          (alternative 1)
(40 ≤ i ≤ 59): f = (b and c) or (d and (b xor c))         (alternative 2)
(40 ≤ i ≤ 59): f = (b and c) xor (d and (b xor c))        (alternative 3)
(40 ≤ i ≤ 59): f = (b and c) + (d and (b xor c))          (alternative 4)
(40 ≤ i ≤ 59): f = (b and c) xor (b and d) xor (c and d)  (alternative 5)
(40 ≤ i ≤ 59): f = vec_sel(c, b, c xor d)                 (alternative 6)
```

It was also shown[55] that for the rounds 32–79 the computation of:

```
w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1
```

can be replaced with:

```
w[i] = (w[i-6] xor w[i-16] xor w[i-28] xor w[i-32]) leftrotate 2
```

This transformation keeps all operands 64-bit aligned and, by removing the dependency of `w[i]` on `w[i-3]`, allows efficient SIMD implementation with a vector length of 4 like x86 SSE instructions.

Comparison of SHA functions

In the table below, *internal state* means the "internal hash sum" after each compression of a data block.

Comparison of SHA functions

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Rounds	Operations	Security (in bits) against collision attacks	Capacity against length extension attacks	Performance on Skylake (median cpb)[56]		First published
									long messages	8 bytes	
MD5 (as reference)		128	128 (4 × 32)	512	64	And, Xor, Rot, Add (mod 2^{32}), Or	≤18 (collisions found)[57]	0	4.99	55.00	1992
SHA-0		160	160 (5 × 32)	512	80	And, Xor, Rot, Add (mod 2^{32}), Or	<34 (collisions found)	0	≈ SHA-1	≈ SHA-1	1993
SHA-1							<63 (collisions found[58])		3.47	52.00	1995
SHA-2	*SHA-224* *SHA-256*	224 256	256 (8 × 32)	512	64	And, Xor, Rot, Add (mod 2^{32}), Or, Shr	112 128	32 0	7.62 7.63	84.50 85.25	2004 2001
	SHA-384 *SHA-512*	384 512	512 (8 × 64)	1024	80	And, Xor, Rot, Add (mod 2^{64}), Or, Shr	192 256	128 (≤ 384) 0	5.12 5.06	135.75 135.50	2001
	SHA-512/224 *SHA-512/256*	224 256					112 128	288 256	≈ SHA-384	≈ SHA-384	2012
SHA-3	*SHA3-224* *SHA3-256* *SHA3-384* *SHA3-512*	224 256 384 512	1600 (5 × 5 × 64)	1152 1088 832 576	24[59]	And, Xor, Rot, Not	112 128 192 256	448 512 768 1024	8.12 8.59 11.06 15.88	154.25 155.50 164.00 164.00	2015

	SHAKE128
	SHAKE256
		d (arbitrary)
		d (arbitrary)
				1344
				1088
							min(*d*/2, 128)
							min(*d*/2, 256)
								256
								512
									7.08
									8.59
										155.25
										155.50	

Implementations

Below is a list of cryptography libraries that support SHA-1:

- Botan
- Bouncy Castle
- cryptlib
- Crypto++
- Libgcrypt
- Nettle
- OpenSSL
- wolfSSL

See also

- Comparison of cryptographic hash functions
- Hash function security summary
- International Association for Cryptologic Research
- Secure Hash Standard

Notes

1. Stevens, Marc (19 June 2012). "Attacks on Hash Functions and Applications" (https://marc-stevens.nl/research/papers/PhD%20Thesis%20Marc%20Stevens%20-%20Attacks%20on%20Hash%20Functions%20and%20Applications.pdf) (PDF). *PhD Thesis*.
2. Stevens, Marc; Bursztein, Elie; Karpman, Pierre; Albertini, Ange; Markov, Yarik. "The first collision for full SHA-1" (https://shattered.io/static/shattered.pdf) (PDF). *Shattered IO*. Retrieved 23 February 2017.
3. http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf
4. Schneier, Bruce (February 18, 2005). "Schneier on Security: Cryptanalysis of SHA-1" (https://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html).
5. "NIST.gov – Computer Security Division – Computer Security Resource Center" (https://web.archive.org/web/20110625054822/http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html). Archived from the original (http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html) on 2011-06-25. Retrieved 2019-01-05.
6. Stevens1, Marc; Karpman, Pierre; Peyrin, Thomas. "The SHAppening: freestart collisions for SHA-1" (https://sites.google.com/site/itstheshappening/). Retrieved 2015-10-09.
7. Schneier, Bruce (8 October 2015). "SHA-1 Freestart Collision" (https://www.schneier.com/blog/archives/2015/10/sha-1_freestart.html). *Schneier on Security*.
8. Goodin, Dan (2016-05-04). "Microsoft to retire support for SHA1 certificates in the next 4 months" (https://arstechnica.com/information-technology/2016/05/microsoft-to-retire-support-for-sha1-certificates-in-the-next-4-months/). *Ars Technica*. Retrieved 2019-05-29.
9. "Google will drop SHA-1 encryption from Chrome by January 1, 2017" (https://venturebeat.com/2015/12/18/google-will-drop-sha-1-encryption-from-chrome-by-january-1-2017/). *VentureBeat*. 2015-12-18. Retrieved 2019-05-29.
10. "The end of SHA-1 on the Public Web" (https://blog.mozilla.org/security/2017/02/23/the-end-of-sha-1-on-the-public-web/). *Mozilla Security Blog*. Retrieved 2019-05-29.
11. "CWI, Google announce first collision for Industry Security Standard SHA-1" (https://phys.org/news/2017-02-cwi-google-collision-industry-standard.html). Retrieved 2017-02-23.
12. "Announcing the first SHA1 collision" (https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html). *Google Online Security Blog*. 2017-02-23.
13. "SHAttered" (https://shattered.io/). Retrieved 2017-02-23.
14. RSA FAQ on Capstone (http://x5.net/faqs/crypto/q150.html)
15. Selvarani, R.; Aswatha, Kumar; T V Suresh, Kumar (2012). *Proceedings of International Conference on Advances in Computing* (https://books.google.com/books?id=L2OFg7OiV9YC&pg=PA551). Springer Science & Business Media. p. 551. ISBN 978-81-322-0740-5.
16. *Secure Hash Standard, Federal Information Processing Standards Publication FIPS PUB 180*, National Institute of Standards and Technology, 11 May 1993
17. National Institute on Standards and Technology Computer Security Resource Center, NIST's March 2006 Policy on Hash Functions (http://csrc.nist.gov/groups/ST/hash/policy_2006.html) Archived (https://web.archive.org/web/20140102194145/http://csrc.nist.gov/groups/ST/hash/policy_2006.html) 2014-01-02 at the Wayback Machine, accessed September 28, 2012.
18. National Institute on Standards and Technology Computer Security Resource Center, NIST's Policy on Hash Functions (http://csrc.nist.gov/groups/ST/hash/policy.html) Archived (https://web.archive.org/web/20110609064344/http://csrc.nist.gov/groups/ST/hash/policy.html) 2011-06-09 at the Wayback Machine, accessed September 28, 2012.
19. "Tech Talk: Linus Torvalds on git" (https://www.youtube.com/watch?v=4XpnKHJAok8&t=56m20s). Retrieved November 13, 2013.
20. Torvalds, Linus. "Re: Starting to think about sha-256?" (https://marc.info/?l=git&m=115678778717621&w=2). *marc.info*. Retrieved 30 May 2016.
21. Wang, Xiaoyun; Yin, Yiqun Lisa; Yu, Hongbo (2005-08-14). *Finding Collisions in the Full SHA-1*. *Advances in Cryptology – CRYPTO 2005*. Lecture Notes in Computer Science. **3621**. Springer, Berlin, Heidelberg. pp. 17–36. doi:10.1007/11535218_2 (https://doi.org/10.1007%2F11535218_2). ISBN 978-3-540-28114-6.
22. Sotirov, Alexander; Stevens, Marc; Appelbaum, Jacob; Lenstra, Arjen; Molnar, David; Osvik, Dag Arne; de Weger, Benne (December 30, 2008). "MD5 considered harmful today: Creating a rogue CA certificate" (http://www.win.tue.nl/hashclash/rogue-ca/). Retrieved March 29, 2009.
23. "Strengths of Keccak – Design and security" (http://keccak.noekeon.org/). *The Keccak sponge function family*. Keccak team. Retrieved 20 September 2015. "Unlike SHA-1 and SHA-2, Keccak does not have the length-extension weakness, hence does not need the HMAC nested construction. Instead, MAC computation can be performed by simply prepending the message with the key."
24. Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno, Cryptography Engineering (http://www.schneier.com/book-ce.html), John Wiley & Sons, 2010. ISBN 978-0-470-47424-2
25. "Cryptology ePrint Archive: Report 2005/010" (http://eprint.iacr.org/2005/010).
26. "SHA-1 Broken – Schneier on Security" (http://www.schneier.com/blog/archives/2005/02/sha1_broken.html).
27. MIT.edu (http://theory.csail.mit.edu/~yiqun/shanote.pdf) Archived (https://web.archive.org/web/20050219180957/http://theory.csail.mit.edu/~yiqun/shanote.pdf) 2005-02-19 at the Wayback Machine, Massachusetts Institute of Technology
28. Lemos, Robert. "Fixing a hole in security" (http://www.zdnet.com/news/fixing-a-hole-in-security/141588). *ZDNet*.
29. "New Cryptanalytic Results Against SHA-1 – Schneier on Security" (http://www.schneier.com/blog/archives/2005/08/new_cryptanalyt.html).
30. Notes on the Wang et al. 2^{63} SHA-1 Differential Path (http://eprint.iacr.org/2007/474)
31. De Cannière, Christophe; Rechberger, Christian (2006-11-15). "Finding SHA-1 Characteristics: General Results and Applications" (http://www.springerlink.com/content/q42205u702p5604u/).
32. "IAIK Krypto Group — Description of SHA-1 Collision Search Project" (http://www.iaik.tugraz.at/content/research/krypto/sha1/SHA1Collision_Description.php). Retrieved 2009-06-30.
33. "Collisions for 72-step and 73-step SHA-1: Improvements in the Method of Characteristics" (http://eprint.iacr.org/2010/413). Retrieved 2010-07-24.
34. "SHA-1 Collision Search Graz" (https://web.archive.org/web/20090225115007/http://boinc.iaik.tugraz.at/sha1_coll_search/). Archived from the original (http://boinc.iaik.tugraz.at/sha1_coll_search/) on 2009-02-25. Retrieved 2009-06-30.
35. "heise online – IT-News, Nachrichten und Hintergründe" (http://www.heise-online.co.uk/security/SHA-1-hash-function-under-pressure--/news/77244). *heise online*.
36. "Crypto 2006 Rump Schedule" (https://www.iacr.org/conferences/crypto2006/rumpsched.html).
37. Manuel, Stéphane. "Classification and Generation of Disturbance Vectors for Collision Attacks against SHA-1" (http://eprint.iacr.org/2008/469.pdf) (PDF). Retrieved 2011-05-19.
38. Manuel, Stéphane. "Classification and Generation of Disturbance Vectors for Collision Attacks against SHA-1" (http://www.springerlink.com/content/u320751102001580/). Retrieved 2012-10-04. *the most efficient disturbance vector is Codeword2 first reported by Jutla and Patthak*
39. SHA-1 collisions now 2^52 (http://eurocrypt2009rump.cr.yp.to/837a0a8086fa6ca714249409ddfae43d.pdf)
40. "Cryptology ePrint Archive: Report 2009/259" (http://eprint.iacr.org/2009/259).
41. Cryptanalysis of MD5 & SHA-1 (http://2012.sharcs.org/slides/stevens.pdf)
42. "When Will We See Collisions for SHA-1? – Schneier on Security" (http://www.schneier.com/blog/archives/2012/10/when_will_we_se.html).

43. "Google Project Hosting" (https://code.google.com/p/hashclash/).
44. Marc Stevens (2012-06-19). "Attacks on Hash Functions and Applications" (https://marc-stevens.nl/research/papers/PhD%20Thesis%20Marc%20Stevens%20-%20Attacks%20on%20Hash%20Functions%20and%20Applications.pdf) (PDF). *PhD Thesis*.
45. "From Collisions to Chosen-Prefix Collisions Application to Full SHA-1" (https://link.springer.com/chapter/10.1007%2F978-3-030-17659-4_18). *SpringerLink*. Retrieved 2019-05-15.
46. Gaëtan Leurent; Thomas Peyrin (2019-04-24). "From Collisions to Chosen-Prefix Collisions - Application to Full SHA-1" (https://eprint.iacr.org/2019/459.pdf) (PDF). *Eurocrypt 2019*.
47. Chabaud, Florent; Joux, Antoine (1998). *Differential Collisions in SHA-0* (http://fchabaud.free.fr/English/Publications/sha.pdf) (PDF). CRYPTO '98.
48. Biham, Eli; Chen, Rafi. "Near-Collisions of SHA-0" (https://www.iacr.org/archive/crypto2004/31520290/biham-chen-sha0-proc-real-one.pdf) (PDF).
49. "Report from Crypto 2004" (https://web.archive.org/web/20040821031401/http://www.freedom-to-tinker.com/archives/000664.html). Archived from the original (http://www.freedom-to-tinker.com/archives/000664.html) on 2004-08-21.
50. Grieu, Francois (18 August 2004). "Re: Any advance news from the crypto rump session?". Newsgroup: sci.crypt (news:sci.crypt). Event occurs at 05:06:02 +0200. Usenet: fgrieu-05A994.05060218082004@individual.net (news:fgrieu-05A994.05060218082004@individual.net).
51. ‹See Tfd›(in Chinese) Sdu.edu.cn (http://www.infosec.sdu.edu.cn/paper/sha0-crypto-author-new.pdf) Archived (https://web.archive.org/web/20050910132832/http://www.infosec.sdu.edu.cn/paper/sha0-crypto-author-new.pdf) 2005-09-10 at the Wayback Machine, Shandong University
52. Manuel, Stéphane; Peyrin, Thomas (2008-02-11). "Collisions on SHA-0 in One Hour". *Fast Software Encryption*. Lecture Notes in Computer Science. **5086**. pp. 16–35. doi:10.1007/978-3-540-71039-4_2 (https://doi.org/10.1007%2F978-3-540-71039-4_2). ISBN 978-3-540-71038-7.
53. "National Institute of Standards and Technology" (https://web.archive.org/web/20110604144720/http://csrc.nist.gov/groups/ST/toolkit/documents/shs/hash_standards_comments.pdf) (PDF). Archived from the original (http://csrc.nist.gov/groups/ST/toolkit/documents/shs/hash_standards_comments.pdf) (PDF) on 2011-06-04. Retrieved 2010-05-05.
54. "RFC 3174 – US Secure Hash Algorithm 1 (SHA1)" (http://www.faqs.org/rfcs/rfc3174.html).
55. Locktyukhin, Max; Farrel, Kathy (2010-03-31), "Improving the Performance of the Secure Hash Algorithm (SHA-1)" (http://software.intel.com/en-us/articles/improving-the-performance-of-the-secure-hash-algorithm-1/), *Intel Software Knowledge Base*, retrieved 2010-04-02
56. "Measurements table" (http://bench.cr.yp.to/results-hash.html#amd64-skylake). *bench.cr.yp.to*.
57. Xie Tao; Fanbao Liu & Dengguo Feng (2013). "Fast Collision Attack on MD5 <img src=x>" (https://eprint.iacr.org/2013/170.pdf) (PDF).
58. "Announcing the first SHA1 collision" (https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html). Retrieved 2017-02-23.
59. "The Keccak sponge function family" (http://keccak.noekeon.org/specs_summary.html). Retrieved 2016-01-27.

References

- Florent Chabaud, Antoine Joux: Differential Collisions in SHA-0. CRYPTO 1998. pp56–71
- Eli Biham, Rafi Chen, Near-Collisions of SHA-0, Cryptology ePrint Archive, Report 2004/146, 2004 (appeared on CRYPTO 2004), IACR.org (http://eprint.iacr.org/2004/146/)
- Xiaoyun Wang, Hongbo Yu and Yiqun Lisa Yin, Efficient Collision Search Attacks on SHA-0, CRYPTO 2005, CMU.edu (https://web.archive.org/web/20080725020713/http://www.cs.cmu.edu/~dbrumley/srg/spring06/sha-0.pdf)
- Xiaoyun Wang, Yiqun Lisa Yin and Hongbo Yu, Finding Collisions in the Full SHA-1, Crypto 2005 MIT.edu (http://people.csail.mit.edu/yiqun/SHA1AttackProceedingVersion.pdf)
- Henri Gilbert, Helena Handschuh: Security Analysis of SHA-256 and Sisters. Selected Areas in Cryptography 2003: pp175–193
- unixwiz.net (http://www.unixwiz.net/techtips/iguide-crypto-hashes.html)
- "Proposed Revision of Federal Information Processing Standard (FIPS) 180, Secure Hash Standard" (http://frwebgate1.access.gpo.gov/cgi-bin/waisgate.cgi?WAISdocID=5963452267+0+0+0&WAISaction=retrieve). *Federal Register*. **59** (131): 35317–35318. 1994-07-11. Retrieved 2007-04-26.
- A. Cilardo, L. Esposito, A. Veniero, A. Mazzeo, V. Beltran, E. Ayugadé, A CellBE-based HPC application for the analysis of vulnerabilities in cryptographic hash functions, High Performance Computing and Communication international conference, August 2010

External links

- CSRC Cryptographic Toolkit (https://web.archive.org/web/20110625054822/http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html) – Official NIST site for the Secure Hash Standard
- FIPS 180-4: Secure Hash Standard (SHS) (https://web.archive.org/web/20161126003357/http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf)
- RFC 3174 (with sample C implementation)
- Interview with Yiqun Lisa Yin concerning the attack on SHA-1 (http://www.zdnet.com/news/fixing-a-hole-in-security/141588)
- Explanation of the successful attacks on SHA-1 (http://www.heise-online.co.uk/security/Hash-cracked--/features/75686) (3 pages, 2006)
- Cryptography Research – Hash Collision Q&A (http://www.cryptography.com/cnews/hash.html)
- Hash Project Web Site: software- and hardware-based cryptanalysis of SHA-1 (https://web.archive.org/web/20181103112717/http://hashproject.eu/)
- SHA-1 (https://curlie.org/Science/Math/Applications/Communication_Theory/Cryptography/Algorithms/Message_Digests) at Curlie
- Lecture on SHA-1 (https://www.youtube.com/watch?v=5q8q4PhN0cw) on YouTube by Christof Paar (http://www.emsec.rub.de/chair/_staff/christof-paar/)

Retrieved from "https://en.wikipedia.org/w/index.php?title=SHA-1&oldid=902464602"

This page was last edited on 19 June 2019, at 00:52 (UTC).