Network Working Group
Internet-Draft
Obsoletes: 4270 (if approved)
Intended status: Informational
Expires: October 31, 2013

P. Hoffman
VPN Consortium
B. Schneier
Counterpane Internet Security

April 29, 2013

Attacks on Cryptographic Hashes in Internet Protocols
draft-hoffman-schneier-4270bis-02

Abstract

Announcements in the past decade of better-than-expected collision attacks in popular hash algorithms have caused some people to question whether common Internet protocols need to be changed, and if so, how. This document summarizes the use of hashes in many protocols, discusses how the collision attacks affect and do not affect the protocols, shows how to thwart known attacks on digital certificates, and discusses future directions for protocol designers. It also gives rationales for moving away from some hash algorithms altogether and for choosing when to start using newer, presumably better, hash algorithms in Internet protocols.

This document obsoletes RFC 4270 and introduces significant new material that has been learned since the publication of that document.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 31, 2013.

Table of Contents

1. Introduction

In summer 2004, a team of researchers showed concrete evidence that the MD5 hash algorithm was susceptible to collision attacks [MD5-attack]. In early 2005, the same team demonstrated a similar attack on a variant of the SHA-1 [RFC3174] hash algorithm, with a prediction that the normally used SHA-1 would also be susceptible with a large amount of work (but at a level below what should be required if SHA-1 worked properly) [SHA-1-attack]; that work has been improved on more recently [Stevens]. Also in early 2005, researchers showed a specific construction of PKIX certificates [RFC3280] that use MD5 for signing [PKIX-MD5-construction], and another researcher showed a faster method for finding MD5 collisions (eight hours on 1.6-GHz computer) [MD5-faster].

Because of these announcements, there was a great deal of discussion by cryptography experts, protocol designers, and other concerned people about what, if anything, should be done based on the news.

Unfortunately, some of these discussions have been based on erroneous interpretations of both the news and on how hash algorithms are used in common Internet protocols.

Hash algorithms are used by cryptographers in a variety of security protocols, for a variety of purposes, at all levels of the Internet protocol stack. They are used because they have two security properties: to be "one way" and "collision free". (There is more about these properties in the next section; they're easier to explain in terms of breaking them.) The attacks have demonstrated that one of those security properties is not true. While it is certainly possible, and at a first glance even probable, that the broken security property will not affect the overall security of many specific Internet protocols, the conservative security approach is to change hash algorithms. The Internet protocol community needs to migrate in an orderly manner away from SHA-1 and MD5 -- especially MD5 -- and toward more secure hash algorithms, namely the SHA-2 family[RFC6234].

This document summarizes what is currently known about hash algorithms and the Internet protocols that use them. It also gives advice on how to avoid the currently known problems with MD5 and SHA-1.

A high-level summary of the current situation is:

- Both MD5 and SHA-1 have attacks against them, the attacks against MD5 being much more severe than the attacks against SHA-1.
- The attacks against MD5 are practical on any modern computer.
- The attacks against SHA-1 are practical for some attackers now, and will be more practical for more attackers in the near future.
- Attackers are able to use collisions in PKIX certificates to generate certificates that can act as trusted entities to create other certificates.
- Many common Internet protocols use hashes in ways that are unaffected by these attacks.
- Most of the affected protocols use digital signatures.
- Better hash algorithms will reduce the susceptibility of these attacks to an acceptable level for all users.

2. Hash Algorithms and Attacks on Them

A "perfect" hash algorithm has a few basic properties. The algorithm converts a chunk of data (normally, a message) of any size into a fixed-size result. The length of the result is called the "hash length" and is often denoted as "L"; the result of applying the hash algorithm on a particular chunk of data is called the "hash value" for that data. Any two different messages of any size should have an exceedingly small probability of having the same hash value, regardless of how similar or different the messages are.

This description leads to two mathematical results. Finding a pair of messages M1 and M2 that have the same hash value takes 2^(L/2) attempts. For any reasonable hash length, this is an impossible problem to solve (collision free). Also, given a message M1, finding any other message M2 that has the same hash value as M1 takes 2^L attempts. This is an even harder problem to solve (one way).

Note that this is the description of a perfect hash algorithm; if the algorithm is less than perfect, an attacker can expend less than the full amount of effort to find two messages with the same hash value.

There are two categories of attacks.

Attacks against the "collision free" property:

- A "collision attack" allows an attacker to find two messages M1 and M2 that have the same hash value in fewer than 2^(L/2) attempts.

Attacks against the "one way" property:

- A "first-preimage attack" allows an attacker who knows a desired hash value to find a message that results in that value in fewer than 2^L attempts.

- A "second-preimage attack" allows an attacker who has a desired message M1 to find another message M2 that has the same hash value in fewer than 2^L attempts.

The two preimage attacks are very similar. In a first-preimage attack, you know a hash value but not the message that created it, and you want to discover any message with the known hash value; in the second-preimage attack, you have a message and you want to find a second message that has the same hash. Attacks that can find one type of preimage can often find the other as well.

When analyzing the use of hash algorithms in protocols, it is important to differentiate which of the two properties of hashes are important, particularly now that the collision-free property is

becoming weaker for currently popular hash algorithms. It is certainly important to determine which parties select the material being hashed. Further, as shown by some of the early work, particularly [PKIX-MD5-construction], it is also important to consider which party can predict the material at the beginning of the hashed object.

2.1. Currently Known Attacks

All the currently known practical or almost-practical attacks on MD5 and SHA-1 are collision attacks. This is fortunate: significant first- and second-preimage attacks on a hash algorithm would be much more devastating in the real world than collision attacks, as described later in this document.

It is also important to note that the current collision attacks require at least one of the two messages to have a fair amount of structure in the bits of the message. This means that finding two messages that both have the same hash value *and* are useful in a real-world attack is more difficult than just finding two messages with the same hash value.

Since the time that the collision attacks were first discovered, some attackers have used them to bootstrap other more devastating types of attacks. Probably the most significant is [Chosen-Prefix], which shows how to use collisions to forge new CA certificates that would be accepted by web browsers at the time. The "Flame" worm attack discovered in 2012 relied on spoofed Microsoft code-signing certificates that were acquired using the collision reduction attack on MD5 [Flame-Microsoft].

3. How Internet Protocols Use Hash Algorithms

Hash algorithms are used in many ways on the Internet. Most protocols that use hash algorithms do so in a way that makes them immune to harm from collision attacks. This is not by accident: good protocol designers develop their protocols to withstand as many future changes in the underlying cryptography as possible, including attacks on the cryptographic algorithms themselves.

Uses for hash algorithms include:

- Non-repudiable digital signatures on messages. Non-repudiation is a security service that provides protection against false denial of involvement in a communication. S/MIME and OpenPGP allow mail senders to sign the contents of a message they create, and the recipient of that message can verify whether or not the signature is actually associated with the message. A message is used for

non-repudiation if the message is signed and the recipient of the message can later use the signature to prove that the signer indeed created the message.

- Digital signatures in certificates from trusted third parties. Although this is similar to "digital signatures on messages", certificates themselves are used in many other protocols for authentication and key management.

- Challenge-response protocols. These protocols combine a public large random number with a value to help hide the value when being sent over unencrypted channels.

- Message authentication with shared secrets. These are similar to challenge-response protocols, except that instead of using public values, the message is combined with a shared secret before hashing.

- Key derivation functions. These functions make repeated use of hash algorithms to mix data into a random string for use in one or more keys for a cryptographic protocol.

- Mixing functions. These functions also make repeated use of hash algorithms to mix data into random strings, for uses other than cryptographic keys.

- Integrity protection. It is common to compare a hash value that is received out-of-band for a file with the hash value of the file after it is received over an unsecured protocol such as FTP.

Of the above methods, only the first two are affected by collision attacks, and even then, only in limited circumstances. So far, it is believed that, in general, challenge-response protocols are not susceptible, because the sender is authenticating a secret already stored by the recipient. In message authentication with shared secrets, the fact that the secret is known to both parties is also believed to prevent any sensible attack. All key derivation functions in IETF protocols take random input from both parties, so the attacker has no way of structuring the hashed message.

4. Hash Collision Attacks and Non-repudiation of Digital Signatures

The basic idea behind the collision attack on a hash algorithm used in a digital-signature protocol is that the attacker creates two messages that have the same hash value, causes one of them to be signed, and then uses that signature over the other message for some nefarious purpose. The specifics of the attack depend on the protocol being used and what the victim does when presented with the signed message.

The canonical example is where you create two messages, one of which says "I will pay $10 for doing this job" and the other of which says "I will pay $10,000 for doing this job". You present the first message to the victim, get them to sign it, do the job, substitute the second message in the signed authorization, present the altered signed message (whose signature still verifies), and demand the higher amount of money. If the victim refuses, you take them to court and show the second signed message.

Most non-repudiation attacks rely on a human assessing the validity of the purportedly signed message. In the case of the hash-collision attack, the purportedly signed message's signature is valid, but so is the signature on the original message. The victim can produce the original message, show that they signed it, and show that the two hash values are identical. The chance of this happening by accident is one in 2^L, which is infinitesimally small for either MD5 or SHA-1.

In other words, to thwart a hash collision attack in a non-repudiation protocol where a human is using a signed message as authorization, the signer needs to keep a copy of the original message they signed. Messages that have other messages with the same hash must be created by the same person, and do not happen by accident under any known probable circumstances. The fact that the two messages have the same hash value should cause enough doubt in the mind of the person judging the validity of the signature to cause the legal attack to fail (and possibly bring intentional fraud charges against the attacker).

Thwarting hash collision attacks in automated non-repudiation protocols is potentially more difficult, because there may be no humans paying enough attention to be able to argue about what should have happened. For example, in electronic data interchange (EDI) applications, actions are usually taken automatically after authentication of a signed message. Determining the practical effects of hash collisions would require a detailed evaluation of the protocol.

5. Hash Collision Attacks and Digital Certificates from Trusted Third Parties

Digital certificates are a special case of digital signatures. In general, there is no non-repudiation attack on trusted third parties due to the fact that certificates have specific formatting. Digital certificates are often used in Internet protocols for key management and for authenticating a party with whom you are communicating, possibly before granting access to network services or trusting the party with private data such as credit card information.

It is therefore important that the granting party can trust that the certificate correctly identifies the person or system identified by the certificate. If the attacker can get a certificate for two different identities using just one public key, the victim can be fooled into believing that one person is someone else.

The collision attack on PKIX certificates described in early 2005 relied on the ability of the attacker to create two different public keys that would cause the body of the certificate to have the same hash value. For this attack to work, the attacker needs to be able to predict the contents and structure of the certificate before it is issued, including the identity that will be used, the serial number that will be included in the certificate, and the start and stop dates of the validity period for the certificate.

One effective result of this attack is that one person using a single identity can get a digital certificate over one public key, but be able to pretend that it is over a different public key (but with the same identity, valid dates, and so on). Because the identity in the two certificates is the same, there are probably no real-world examples where such an attack would get the attacker any advantage. At best, someone could claim that the trusted third party made a mistake by issuing a certificate with the same identity and serial number based on two different public keys. This is indeed far-fetched.

After the above attack was described, many CAs continued to use the MD5 hash function. Because of this, a group of researchers went further and showed how to create a rogue CA that would be accepted by all the common web browsers at the time [Chosen-Prefix]. After that, the CAs mostly (but not completely) stopped using the MD5 hash function, and most browsers stopped accepting signatures that used MD5. It should be noted that the same attack would work with SHA-1, but it would take much more effort to mount such an attack. An estimate for the actual amount of effort is described later in this document.

It is very important to note that collision attacks only affect the parts of certificates that have no human-readable information in them, such as the public keys. An attack that involves getting a

certificate with one human-readable identity and making that certificate useful for a second human-readable identity would require more effort than a simple collision attack.

5.1. Reducing the Likelihood of Hash-based Attacks on PKIX Certificates

If a trusted third party who issues PKIX certificates wants to avoid the attack described above, they should sign using a hash algorithm for which collisions are believed to be essentially impossible, such as those from the SHA-2 family. If they want to use weaker hashes such as SHA-1, they can also prevent the attack by making other signed parts of the certificate random enough to eliminate any advantage gained by the attack. Ideas that have been suggested include:

- making part of the certificate serial number unpredictable to the attacker
- adding a randomly chosen component to the identity
- making the validity dates unpredictable to the attacker by skewing each one forwards or backwards

Any of these mechanisms would increase the amount of work the attacker needs to do to trick the issuer of the certificate into generating a certificate that is susceptible to the attack.

6. Cost of Creating Collisions in SHA-1

At the time that this document has been prepared, a scheme for creating collisions in SHA-1 has been widely discussed, but no collision has been published. However, it is widely assumed that such a collision is currently possible in non-public settings, and that a public demonstration of a SHA-1 collision should be expected within a few years. The current estimate is that creating a collision will take approximately 2^60 operations [Stevens].

The following is an estimate, originated by Jesse Walker, of how much money and time it would take to create a SHA-1 hash collision in the future. [Walker] It uses numbers based on hardware from the time this document is published, and has a lot of room for error.

Using the eBACS benchmarks [eBACS], SHA-1 takes about 16 cycles per byte for 64-byte blocks on modern processors. SHA-1 processes input data by blocks of 64 bytes, so the cost of one block of a SHA-1 operation is 16 * 64, or 2^10 cycles. If an attack of 2^60 SHA-1 operations serves as the baseline, then finding a collision costs about 2^10 * 2^60, or 2^70, cycles.

A CPU core today provides about 2^31 cycles per second. The state of the art is 2^3 cores per processor, for a total of 2^3 * 2^31, or 2^34, cycles per second. A server typically has 4 processors, increasing the total to 2^2 * 2^34, or 2^36, cycles per second. Because there are about 2^25 seconds per year, a server can deliver about 2^25 * 2^36, or 2^61, cycles per year, with the beginning of 2013 as a base. This is called a "server year" in this discussion.

For this discussion, we assume that Moore's law as currently applied to server-level CPUs says that the speed will increase approximately 40% per year, or double ever two years, and this will continue for the next decade. (Note that the authors do not have a good reference for a respected estimate of applying Moore's law to current server-level CPUs, and would appreciate one!).

This means that the processor power should double by 2015, double again by 2017, and then double again by 2019. Thus, a commodity "server year" should be about

2^1 * 2^61, or 2^62 cycles per year by 2015
2^2 * 2^61, or 2^63 cycles per year by 2017
2^4 * 2^61, or 2^65 cycles per year by 2019
2^8 * 2^61, or 2^69 cycles per year by 2021

With an attack that takes 2^69 iterations, the attack should cost approximately

2^70 / 2^62, or 2^8 server years by 2015
2^70 / 2^63, or 2^7 server years by 2017
2^70 / 2^65, or 2^5 server years by 2019
2^70 / 2^69, or 2^1 server year by 2021

Today, Amazon rents compute time on commodity servers for about US$0.04 per hour, or about US$350 per year. Assume that compute rental fees remain fixed while server capacity keeps pace with Moore's law. Then, because log2(350) is approximately 8.4, the cost of the attack will be approximately

2^8 * 2^8.4, or about US$86K in 2015
2^7 * 2^8.4, or about US$43K in 2017
2^5 * 2^8.4, or about US$10K in 2019
2^1 * 2^8.4, or about US$700 in 2021

A collision attack is therefore well within the range of what an organized crime syndicate can practically budget by 2015, and essentially any criminal by 2021.

Since this argument only takes into account commodity server hardware and not instruction set improvements (for example, ARM 8 specifies a SHA-1 instruction), other commodity computing devices with even greater processing power (particularly GPUs) and custom hardware, the need to transition away from SHA-1 because of collision resistance functions could be more urgent than this back-of-the-envelope analysis suggests.

7. The SHA-3 Competition and the Future

After the events in 2005, NIST decided that it needed to have a backup hash function in case the SHA-2 family had problems similar to those found in MD5 and SHA-1. NIST held a competition [SHA3-competition] to design a new family of hashes that was as strong as the SHA-2 family but was based on different underlying technologies.

The competition was both a success and a disappointment. It was a success in that there were many different strong candidates for the new family, and the competition caused a leap in understanding of the weaknesses in many proposals for hash functions. It was a disappointment in that none of the finalists were actually faster or provably better than the SHA-2 family.

In late 2012, NIST announced that the winner of the competition was the Keccak hash function [KECCAK-winner]. NIST was pleased about the properties of KECCAK, noting that it was sufficiently different from SHA-2 to have made the competition worthwhile. At the same time, NIST emphasized that "SHA-2 has held up well and NIST considers SHA-2 to be secure and suitable for general use" and that KECCAK is "an essential insurance policy in case SHA-2 is ever broken".

8. Future Attacks and Their Effects

The authors believe that everyone should start migrating to SHA-2 now, due to the weaknesses that have already been demonstrated in both MD5 and SHA-1. There is an old saying inside the US National Security Agency (NSA): "Attacks always get better; they never get worse." The current collision attacks against MD5 are easily done on a single computer; the collision attacks against SHA-1 are at the far edge of feasibility today, but will only improve with time. It is noted that at the time of this writing, there has still not been shown a successful collision in SHA-1 in public; there has also not been shown any problem with the descriptions of how such a collision could probably be found.

It is preferable to migrate to the SHA-2 family of hashes before there is a panic, instead of after. Just as we all migrated from SHA-0 to SHA-1 based on some unknown vulnerability discovered inside the NSA, we need to migrate from SHA-1 to SHA-256 based on these attacks. SHA-256 has a 256-bit hash length. This length will give us a much larger security margin in the event of newly discovered attacks.

The authors acknowledge that deprecating a hash algorithm is difficult from an operational standpoint, but it needs to be done eventually, and this is a good time to again make a significant push to SHA-2. When NIST finalizes the parameters and test vectors for SHA-3, implementors might want to start adding SHA-3 to their products. However, the authors do not believe that any push to start using SHA-3 is warranted at any time soon.

The authors also feel that work should continue to make all Internet protocols able to use different hash algorithms. Fortunately, most protocols today already are capable of this; those that are not really should be fixed soon.

9. Security Considerations

The entire document discusses security on the Internet.

The discussion in this document assumes that the only attacks on hash algorithms used in Internet protocols are collision attacks. Some significant preimaging attacks have already been discovered [Preimaging-attack], but they are not yet practical. If a practical preimaging attack is discovered, it would drastically affect many Internet protocols. In this case, "practical" means that it could be executed by an attacker in a meaningful amount of time for a meaningful amount of money.

A preimaging attack that costs trillions of dollars and takes decades to preimage one desired hash value or one message is not practical;

one that costs a few thousand dollars and takes a few weeks might be very practical. Even a preimaging attack that costs millions of dollars and takes over a year would be practical in high-value environments such as issuance of certificates.

10. Informative References

[Chosen-Prefix]
, "Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate", 2009, <http://eprint.iacr.org/2009/111.pdf>.

[Flame-Microsoft]
, "TechNet Blogs: Flame malware collision attack explained", June 2012, <http://blogs.technet.com/b/srd/archive/2012/06/06/more-information-about-the-digital-certificates-used-to-sign-the-flame-malware.aspx>.

[KECCAK-winner]
, "NIST Selects Winner of Secure Hash Algorithm (SHA-3) Competition", October 2012, <http://www.nist.gov/itl/csd/sha-100212.cfm>.

[MD5-attack]
X. Wang, , D. Feng, , X. Lai, , and H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD", August 2004, <http://eprint.iacr.org/2004/199>.

[MD5-faster]
Vlastimil Klima, , "Finding MD5 Collisions - a Toy For a Notebook", March 2005, <http://cryptography.hyperlink.cz/md5/MD5_collisions.pdf>.

[PKIX-MD5-construction]
Arjen Lenstra, and Benne de Weger, "On the possibility of constructing meaningful hash collisions for public keys", February 2005, <http://www.win.tue.nl/~bdeweger/CollidingCertificates/ddl-final.pdf>.

[Preimaging-attack]
John Kelsey, and Bruce Schneier, "Second Preimages on n-bit Hash Functions for Much Less than 2^n Work", November 2004, <http://eprint.iacr.org/2004/304>.

[RFC3174] Eastlake, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001.

[RFC3280] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[RFC6234] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011.

[SHA-1-attack] Xiaoyun Wang, , Yiqun Lisa Yin, , and Hongbo Yu, "Collision Search Attacks on SHA1", February 2005, <http://theory.csail.mit.edu/~yiqun/shanote.pdf>.

[SHA3-competition] , "Cryptographic Hash Algorithm Competition", October 2012, <http://csrc.nist.gov/groups/ST/hash/sha-3/index.html>.

[Stevens] , "Cryptanalysis of MD5 and SHA-1", 2010, <http://2012.sharcs.org/slides/stevens.pdf>.

[Walker] , "When Will We See Collisions for SHA-1?", 2012, <http://www.schneier.com/blog/archives/2012/10/when_will_we_se.html>.

[eBACS] Dan Bernstein, , "eBACS: ECRYPT Benchmarking of Cryptographic Systems", 2012, <http://bench.cr.yp.to/ebash.html>.

Appendix A. Acknowledgements

Acknowledgements from RFC 4270: The authors would like to thank the IETF community, particularly those active on the SAAG mailing list, for their input. We would also like to thank Eric Rescorla, Arjen Lenstra, and Benne de Weger for material and significant comments on the first draft of this document.

Jesse Walker contributed the analysis of the cost of creating SHA-1 collisions. [[There will probably be more here.]]

Authors' Addresses

Paul Hoffman
VPN Consortium

Email: paul.hoffman@vpnc.org

Bruce Schneier
Counterpane Internet Security

Email: schneier@counterpane.com