Editors: Tim Grance, tim.grance@nist.gov
Rick Kuhn, kuhn@nist.gov
Susan Landau, susan.landau@sun.com

Cryptographic Hash Standards

Where Do We Go from Here?

William E. Burr
US National Institute of Standards and Technology

As *S&P* readers are surely aware, researchers have successfully attacked MD5 and SHA-1, the two most commonly used cryptographic hash functions. It's no longer advisable to use them in applications such as digital signatures, although some other applications, such as hashed message authentication codes, aren't affected.

The attacks have triggered a kind of feeding frenzy in the cryptographic community; many researchers are now working on hash function attacks, and we can expect new results in this area for the next several years. In this article, I discuss the SHA-1 attack and the US National Institute of Standards and Technology's (NIST's) plans for SHA-1 and hash functions in general.

A short review

Ralph Merkle and Ivan Damgaard laid out the basic design principles for hash functions in 1990.[1,2] As a brief review, recall that a hash function produces a checksum, called a *message digest*, over a message. (For a more complete discussion, see the May/June 2005 Crypto Corner department.[3]) The US National Security Agency (NSA) followed the Merkle-Damgaard principles in designing the SHA-1 hash function, which NIST adopted as a federal standard in 1995, and the SHA-2 functions (SHA-224, SHA-256, SHA-384, and SHA-512), adopted in 2002. Intended to be as strong as the new Advanced Encryption Standard (AES), the SHA-2 functions feature larger message digests and more complex processing than SHA-1.

SHA-1 generates a 160-bit message digest, which means 2^{160} different digests are possible, despite an infinite number of possible messages. Each message pair that has the same digest is called a *collision*. A hash function used for digital signatures must be collision-resistant, so that there's no practical way to calculate the message pairs. That said, it's always theoretically possible to find collisions by brute force—trying a huge number of messages until you find two with the same message digest. For SHA-1, such an attack would take approximately 2^{80} (roughly 10^{24}) attempts. Although this is currently considered to be infeasible, Moore's law could make completing 2^{80} operations practical some time after 2010, which is when NIST plans to withdraw approval for the use of SHA-1 for applications that require collision resistance, along with other algorithms and key sizes that offer 80 bits of security.

Recently, Xiaoyun Wang and her colleagues discovered a way to find SHA-1 collisions in approximately 2^{63} operations[4]—still a very large, expensive effort, but a determined, well-funded attacker could achieve it today. The goal is that it should be completely infeasible to find a collision. Only a few (although important) cryptographic-hash applications require collision resistance. Collision attacks apply primarily to applications in which messages are hashed and then digitally signed. The problem is most acute if one party prepares a message but another party signs it (for example, time stamps and code signing). It is still expensive to find the SHA-1 collisions, and carefully designed applications can incorporate certain countermeasures to prevent attack.

Most applications require a weaker property called second-preimage resistance, which guarantees that, given any specific message, finding another with the same digest is infeasible. (Clearly, any collision-resistant hash function must also be second-preimage resistant.) No brute-force second-preimage attack against realistic SHA-1 messages is known for much less than 2^{160} operations.

What have we learned?

With SHA-1 and SHA-2 in its cryptographic toolkit, NIST had hoped to be done with hash functions for a long time. Aside from a near break of MD5 by Hans Dobbertin in 1996,[5] researchers made little progress in hash function analysis until mid-2004. Since then, Wang, Antonine Joux, and Eli Biham have attacked nearly all the early hash functions, including SHA-1. Given that SHA-2 functions are in the same family as the earlier broken functions, these attacks shook cryptographers' long-term confidence in nearly all hash functions designed to date.

To help plot a course for relevant recommendations and standards over the next several years, NIST recently held a workshop on cryptographic hash function design and analysis. Nearly 180 people, including Wang and other leading cryptographers, discussed current attacks, including analysis of the still unbroken SHA-2 family of functions, how rapidly we could convert to them, and how long they'll likely remain secure, as well as long-term solutions and the need for new hash functions.

Cryptographers have learned much about hash functions and how to attack them in the past couple of years, yet cryptanalysts at the workshop generally agreed that practical attacks on the SHA-2 hash functions are unlikely in the next decade. However, attacks and research results could reduce their strength well below theoretical work levels (2^{112}, 2^{128}, 2^{192}, and 2^{256} operations for SHA-224, SHA-256, SHA-384, and SHA-512, respectively). Attendees thus agreed that starting work now to develop new hash functions would be prudent.

Is it time for new hash functions?

Recall that NIST had expected 80-bit hashes to be secure through 2010. As a practical matter, updating SHA-1 signature applications to use a stronger hash function would be difficult before 2011. For example, it would be useless for a PKI certification authority to sign digital certificates using a new hash function until nearly all the systems that used the certificates could process the new algorithm.

A chain of things must happen before a new hash function can be deployed. To begin, developers must add the algorithm to cryptographic toolkits and operating system cryptographic service providers, from which it will be incorporated into the applications that use the hash function. In many communications applications, the updated software must then be broadly deployed before the new hash function can be widely used.

For comparison, consider the development history of block cipher algorithms. In the early 1970s, IBM designed the Lucifer algorithm, which served as the basis for the 1977 NIST-issued Data Encryption Standard (DES) and essentially began the open academic study of block ciphers. In 1990, Biham and Adi Shamir published the first analytic attack that broke the DES with 2^{47} operations,[6] rather than the expected brute-force requirement of 2^{55} operations. The attack was considered impractical because it required huge amounts of known plaintext, but brute-force attacks on DES became practical by the mid-1990s.[7] To maintain security, applications thus had to employ the slower Triple DES algorithm, which iterates DES three times with three different keys to create, at best, 112-bit security. At a workshop in 1997, NIST began developing a replacement for DES and selected the AES algorithm in October 2000.[8] NIST withdrew support for DES in new applications in 1999. Yet, old algorithms die hard, and "single" DES is still widely used—an important lesson as we consider replacing SHA-1.

The hash function technology cycle isn't as far along today as the block cipher technology cycle was in 1997 when NIST began the AES-development process. Fewer hash

A chain of things must happen before a new hash function can be deployed.

functions have been designed and fewer attacks published. Perhaps because it's easier to attack a hash function than a block cipher algorithm, development is more challenging. A block cipher attacker doesn't know the encryption key and often has to gain access to vast amounts of plaintext to carry out an attack, whereas attackers searching for a hash function collision have no key to find, and can choose any messages they want to use and see the hash output as well as the algorithm's internal state at every stage of the computation. The Merkle-Damgaard construct also has undesirable "generic" properties that can lead to attacks. In particular, given one collision pair for any Merkle-Damgaard hash, you can concatenate the same value on the end of the message and get another collision. Thus, you can produce any number of collisions from the first. This, in turn, can let an attacker construct apparently meaningful, and possibly damaging, message collisions from one original very arbitrary collision.[9] Similarly, naïve keyed hash message authentication code implementations can allow attackers to create forgeries without knowledge of the secret key.

Most workshop participants agreed that our understanding of hash functions is still immature. Given that we arguably don't yet know enough to reliably select a better function, attendees agreed that the best strategy for timely replacement of SHA-1 is to deploy the SHA-2 hash functions as expeditiously as possible. The SHA-2 functions are well into the deployment pipeline: SHA-256, for example, is implemented in the Open SSL Crypto Library (although not in many operating Apache Web servers), the current Sun Solaris operating system, the forthcoming Windows Vista operating system (but not in the current Windows XP), and in the widely used Pretty Good Privacy (PGP) secure email package.[1]

A competition

Unlike general protocol standards, secure cryptographic algorithms must be designed by small teams of experts, rather than developed in committees via compromises between competing interests. They can't be loaded with options to give all participants their due, as is common in consensus protocol standard efforts. Algorithms are often implemented in silicon, where real estate is limited, so we can't make everything that is "good enough" a standard.

The AES selection process was a competition. NIST began with an open organizational workshop to discuss requirements, and then, based on participants' suggestions, produced a call for submissions for candidate algorithms. NIST organized three AES conferences at which the crypto community reviewed the initial 15 submissions, which were winnowed down to five "finalists" for more intensive review. NIST then selected an elegant algorithm designed by two Belgian cryptographers, Joan Daemon and Vincent Rijmen, to become the AES.

Most of the AES finalist algorithms would have been good choices, but the industry (particularly semiconductor vendors) wanted a single algorithm, and that's what they got. The process was perhaps not as democratic as a conventional consensus standard process, but everyone got a chance to be heard, and NIST was able to set a schedule and make clear choices. The competition appealed to cryptographers' competitive instincts, and many of the best in the world participated. The academic crypto research community participated more fully than usual in standards committees because the AES conferences were organized around the regular schedule of academic cryptographic conferences, and academics could also get formal AES conference publications. There was only one winner, but nearly everybody involved agreed that the process worked, the selection was reasonable and made on good technical grounds, and everybody had a chance to participate. The competition was intense, but the participants had fun and most went home happy. Some folks disagreed with NIST's documented rationale for its decisions, but at least they knew the reasons.

Spending some time to better understand the basic science of hash functions seems appropriate before beginning a long-term selection process.

The most valuable aspect of the AES competition was its openness, which engendered the belief in the competition's fairness and, most importantly, the strength of the winner. As a result, AES has been widely adopted around the world, and its broad acceptance is a real benefit for computer security.

Next steps

It's fair to say that the cryptographic community generally loved the AES process and now wants to see it repeated for hash functions. The real argument at the moment seems to be how long the process should take and when it should start. One practical, business-oriented viewpoint says that developers are rewriting operating systems and applications for multicore processors, or soon will be, and this is our big opportunity to introduce a new hash function into use—speed is vital. The opposite argument is that we just aren't ready theoretically; collision-resistant hash functions are harder to design than block ciphers, yet we haven't devoted nearly as much effort to analyzing them. The collision results are very recent, and we need to do more basic work to enhance our understanding before selecting a new hash function that can hold up for many years.

Do we look for something with better performance in hardware? Is it time to move beyond the Merkle-Damgaard paradigm? Should we have standardized hash functions that reduce to hard number-theory problems, such as the RSA assumption that factoring large integers is difficult? No widely used hash function is designed this way today, but attendees considered all these questions at the recent workshop, and opinions varied on the proper approach.

Spending some time to better understand the basic science of hash functions seems appropriate before beginning a long-term selection process. We might discover that we need one type of hash function algorithm for digital signatures and another to replace SHA-1 for hashed message authentication codes (HMACs), which is perhaps the most common single use of hash functions. Alternatively, we might find that SHA-1 will be good enough for HMACs indefinitely, because they don't rely on the hash function's collision resistance.

Another hash function workshop is scheduled to follow immediately after the Crypto 2006 conference in Santa Barbara, California, in August. NIST hopes to use it to revisit the uses of hash functions, discuss desirable properties, identify areas of study to accelerate basic research, and set a long-term schedule for a hash function competition. Major issues will be how much time is needed to study the principles for hash function designs, when to call for submissions, and what criteria to use in determining a winner (or winners). In the meantime, the best answer for hash function security remains to replace SHA-1 with the SHA-2 family sooner, rather than later. ☐

Acknowledgments

Identification of commercial products or trademarks doesn't imply any endorsement by the US National Institute of Standards and Technology.

References

1. R. Merkle, "One Way Hash Functions and DES," *Advances in Cryptology—Crypto 1989*, LNCS 435, Springer-Verlag, 1990, pp. 428–446.
2. I. Damgaard, "A Design Principle for Hash Functions," *Advances in Cryptology—Crypto 1989*, LNCS 435, Springer-Verlag, 1990, pp. 416–427.
3. P. Gutmann, D. Naccache, and C. Palmer, "When Hashes Collide," *IEEE Security & Privacy*, vol. 3, no. 3, 2005, pp. 68–71.
4. X. Wang, A. Yao, and F. Yao, "New Collision Search for SHA-1," presented at Crypto 2005.
5. H. Dobbertin, "Cryptanalysis of MD5 Compress," presented at Eurocrypt 1996.
6. E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," *Advances in Cryptology—Crypto 90*, LNCS 537, Springer-Verlag, 1991 pp. 2–21.
7. *Cracking DES: Secrets of Encryption Research, Wiretap Politics, & Chip Design*, Electronic Frontier Foundation, O'Reilly, 1998.
8. W. Burr, "Selecting the Advanced Encryption Standard," *IEEE Security & Privacy*, vol. 1, no. 2, 2003, pp. 43–52.
9. M. Gebhardt, G. Illies, and W. Schindler, "A Note on the Practical Value of Single Hash Function Collisions for Special File Formats," presented at the NIST Cryptographic Hash Workshop, 2005; http://csrc.nist.gov/pki/HashWorkshop/2005/Oct31_Presentations/Illies_NIST_05.pdf.

***William E. Burr** is the manager of the NIST Security Technology Group, which is responsible for US Federal Information Processing Standards for Cryptography. He is also a member of the Advanced Encryption Standard team and chair of the Federal Public Key Infrastructure Technical working group. Burr has a BEE from the Ohio State University. His interests include research measuring of computer instruction set architecture. He was chair of the X3T9.2 standards committee that developed the Small Computer Systems Interface (SCSI), and has worked on standards for PKI and encryption as a member of the NIST Computer Security Division. Contact him at william.burr@nist.gov.*