CryptoCorner

Editors: Peter Gutmann, pgut001@cs.auckland.ac.nz
David Naccache, david.naccache@gemplus.com
Charles C. Palmer, ccpalmer@us.ibm.com

When Hashes Collide

PETER GUTMANN
University of Auckland

DAVID NACCACHE
Gemplus

CHARLES C. PALMER
IBM

Welcome to *IEEE Security & Privacy*'s new department on applied cryptography. Most people think that crypto is beyond the understanding of mere mortals because of the esoteric mathematics that always seem to surround it, but this attitude is dangerous: crypto provides the foundation for any real attempt at security or privacy. Without it, we couldn't build truly secure solutions for the challenges of information confidentiality and integrity or individual user authentication and authorization; we also wouldn't be able to deal with more complicated concepts such as signing digital documents or enabling nonrepudiation for electronic messages or contracts.

Cryptography isn't in the public eye very often, but when it is, the news can be disconcerting. For the inaugural installment of this new department, we'll discuss the issues that have arisen around the recently announced problems with the SHA-1 hash function and its ancestor, MD5.

What's a hash function?

In his book *Applied Cryptography*,[1] Bruce Schneier defines a hash function as

> … a function, mathematical or otherwise, that takes a variable-length input string and converts it to a fixed-length (generally smaller) output string … .

This output string, called the hash value or message digest, should be the same each time the same input is hashed. Although hashes are common in computing, cryptographers extend a hash function's requirements to include being *one way* and *collision resistant*. That is,

- if we have a hash value h, determining which original string produced it should be extremely difficult (one way), and
- producing two strings that hash to the same value should be extremely difficult (collision resistant).

Because hash functions hash variable-length inputs into fixed-length ones, collisions necessarily exist. Think of it like this: at least two non-bald humans on this planet must have exactly the same number of hairs on their heads, simply because the number of hairs on a human head is measured in millions whereas there are billions of humans on Earth. That said, actually finding these two people would be close to impossible.

At a recent crypto conference, cryptographer Ralph Merkle observed that hash functions are the duct tape of crypto. They're used in every Internet security protocol, although usually not in the manner for which they were originally designed. The two main uses of hash functions, besides simple hashing, are as pseudorandom functions (PRFs) and MACs (message authentication codes—specifically, a hash-based MAC, or HMAC).

PRFs convert input strings into output strings that are indistinguishable from random data. This is useful for, say, converting structured input data such as a password into a random (but input-dependent) encryption key, or "stretching" key material such as the output from the IPsec, SSH, or SSL/TLS key exchange into the various encryption, decryption, and MAC keys those protocols require.

HMACs provide a message's integrity protection and authentication by adding a key or password to the hashing process. The recipient can verify that the message is authentic by using the same key to compute a hash of the received message—if these hashes match, then the message is believed to have arrived unchanged from the sender. It is this key hashing that prevents attackers from making undetectable changes to the message because they can't calculate the modified data's hash/HMAC value. If the hash is unkeyed, an attacker could change the message's contents, recalculate the hash value, and replace the message and hash value during transmission, thereby ensuring that the fraudulent message will go undetected.

If a hash function is successfully attacked and made to produce collisions, it's useless for simple hashing. Luckily, the possibility of collisions doesn't affect HMACs or PRFs: HMACs are protected by adding the key, and PRFs don't require collision resistance as a security property. This is

why many major Internet security protocols that use MD5 and SHA-1 aren't affected by the recently published attacks—they only use the PRF and HMAC form of the hash functions, not the pure hash form. For this reason, when Wang and colleagues published details about the recently announced attacks against SHA-1, SSH, SSL/TLS, and IPsec administrators only looked at them briefly before continuing to argue about whose data packets were more standards-conformant. The SSL/TLS protocol stood up particularly well because it uses two hash functions (MD5 and SHA-1) in all its critical steps, thus requiring attackers to be able to simultaneously break both algorithms. An additional safety factor for the aforementioned protocols is that all exchanged messages are transient: a security breach would have to be performed in real time (while the data packets were in flight) and would only affect that one session. In contrast, a successful attack on the hash function used in a digital signature would compromise it for the signature's lifetime—for years, even decades.

Attacks on hash functions

The most recent attack on MD5 (by Arjen Lenstra, Xiaoyun Wang, and Benne de Weger) allowed the creation of different digital certificates with the same signature, making it possible to take the signature from a genuine certificate C and move it across to a bogus certificate C' that contained an attacker-specified key. To create the bogus C' contents, the attackers had to be able to predict C's contents before it was issued. This isn't as hard as it sounds: an attacker intent on getting a bogus certificate would first get his or her own C, convert it into C', and then use the existence of C' to deny signing a document that was actually signed with C, claiming that the signature key C' couldn't have created the signature on the document.

One mitigating factor in this situation is that some certificate authorities (CAs) use a random serial number in the certificate, making it impossible to predict C's contents in advance. However, other CAs use serial numbers for which it's not too hard to predict future values based on current certificates published in directories. The fact that this happens to "save" MD5 is pure blind luck; MD5 is known to have security problems and should have been replaced as a hash function in security protocols long ago. The next time, we might not be so lucky.

The attack Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu recently announced on SHA-1 indicates that the algorithm isn't quite as strong as it was thought to be: it takes 2^{69} steps to find a collision instead of the expected 2^{80} steps. (Its predecessor, SHA-0, is more or less demolished with only 2^{39} operations, the same as a 40-bit encryption key.) Requiring 2^{69} steps is about 2,000 times faster than the pure brute-force 2^{80} steps, but it's still such a vast number that there's no need to panic … yet.

What are the alternatives?

Not many alternatives to SHA-1 are ready to slide into place. Europe has RIPEMD-160, a drop-in replacement for SHA-1 that improves the design by using two parallel data paths that must both be broken to compromise the overall hash function, but it barely has any presence in the US. SHA-1's successors—essentially, the SHA-2 family—consist of SHA-256, SHA-384, and SHA-512 (derived from the general SHA design and named after their output size in bits). However, they haven't been widely analyzed yet and could be vulnerable to extensions of the SHA-1 attacks as well as new ones.

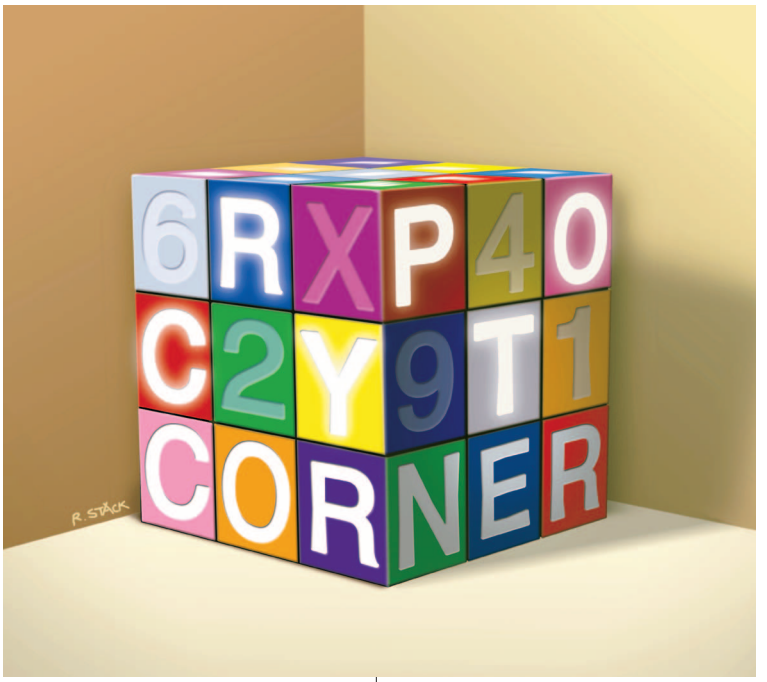
The only other alternative that's gained much acceptance so far is a hash function called Whirlpool (http://paginas.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html), which is based on a completely different design than the MD-x or SHA-x families. This difference in design renders it immune to attacks on those two families, but it could be vulnerable to other attacks. Whirlpool is just too new anyone to make any strong statements about its long-term security.

We simply don't know enough about hash function design to prove if a particular design is good or not. We're at the point where block cipher encryption algorithms were roughly 10 to 15 years ago, an era one observer called "design by twiddle:" create a fast transform, twiddle

About this department

Our goal for this department is simple: enlightenment. We promise timely, clearly written pieces from some of the leading practitioners and researchers in the field. They'll present objective discussions, both pro and con, on various crypto topics, ranging from tutorials on proper use to assessments of new developments to true stories about misadventures to governmental regulation. We'll use a minimum of mathematics and funny symbols, depending instead on lots of practical examples and experiences. Although we don't expect this department to provide a short course in crypto, we'll occasionally provide practice-oriented tutorials on hot topics in the field.

Of the three department editors, David Naccache is a real cryptographer with lots of technical publications and worldwide recognition, Peter Gutmann is a crypto engineer with extensive experience in implementing and applying crypto in practice, and Charles C. Palmer manages teams of cryptographers and has spent many years explaining security and privacy, including crypto, to customers, students, and the news media.



it until you can't think of any further attacks, and then publish the resulting algorithm—as time passes, public scrutiny increases confidence in its resistance. Whirlpool is based on recent research into what are thought to be rigorous security principles for hash functions, but it's too early to know whether they're sufficient.

If a completely new hash function is needed, why couldn't an organization like the US National Institute of Standards and Technology sponsor an Advanced Hash Standard (AHS) initiative alongside its Advanced Encryption Standard (AES)? AES took precedence over AHS for several reasons:

- A significant demand existed for replacing the slow triple-DES encryption algorithm.
- Running something like the AES evaluation process requires a considerable amount of time, money, and people.
- The lack of rigorous and widely accepted design principles for hash functions presented security analysis difficulties.

The New European Schemes for Signatures, Integrity, and Encryption (NESSIE) project specifies the use of Whirlpool, but it seems unlikely to win US government—and thus US software vendor—support.

Fixing the problem

Nathan Bedford Forrest's motto for success was, "Get there first with the most men." In crypto, it's, "Get there first with something adequate" because once you've got a crypto algorithm in place, nothing will be able to displace it. Displacing a security mechanism once it's been widely adopted is extremely difficult. In particular, it's unlikely that a mechanism requiring heavy amounts of reengineering of deployed hardware and software will be adopted in a hurry unless it fixes a major security problem. Even if the change is reasonably critical, it can take a significant amount of time to get it deployed.

The reason for these slow adoption rates rises from a combination of existing protocols and implementations. All browsers depend upon SSL/TLS almost universally, for example, and several implementations use the fairly agile open-source OpenSSL code base; deployment of a new algorithm (once it's deemed necessary) to at least some portion of the user base is fairly quick. Other crypto libraries, especially those used in proprietary systems, tend to move much more slowly, thus the distribution and spread of fixes can take much longer.

Crypto hardware presents special problems of its own. In all cases except the most primitive devices (consisting of nothing more than a public-key encryption engine), moving to a new algorithm requires replacing actual hardware. Crypto hardware boxes typically cost up to US$20,000 each, and a server farm needs a great many of the most expensive boxes. Smaller devices such as smart cards might only cost $10 to $20 each, but if you have to replace 100,000 of them, the numbers add up. Some of the more sophisticated devices support the ability to upload new firmware, but will destroy their keys when this occurs for security reasons (to prevent an attacker from uploading the firmware for a compromised algorithm that leaks the stored keys), requiring a complete redeployment of all keys and certificates from scratch.

All these issues can lead to rather slow deployment of new algorithms, especially when users hold on to older software and equipment until it becomes obsolete so that they can justify the expense of purchasing new crypto gear.

Replacing a compromised algorithm leads to a variant of Epimenides of Crete's paradoxical claim that all Cretans are liars. If the authentication/authorization system is compromised, who do you trust to tell you that? Assume we have two hash algorithms, *A* and *B*, and that two messages arrive, one authenticated with *A* stating that *B* is broken and shouldn't be trusted, and one authenticated with *B* stating that *A* is broken and shouldn't be trusted. Although it's possible to try and fix this with various

kludges like majority-voting mechanisms, on what basis can something be outvoted? If MD2 and MD5 claim that SHA-1 is bad, but SHA-1 claims that MD2 and MD5 are bad, which one should you believe?

The truth is that no real process for quickly replacing compromised crypto algorithms or mechanisms exists yet. So far, we've been fairly lucky in that we've had plenty of warnings when things look dicey for a particular algorithm. Unfortunately, many developers find it easier to just ignore the problems found in MD5 and now SHA-1. The really scary thing is that if SHA-1 ever seriously broke, almost nothing could actually be deployed to replace it.

The new vulnerabilities described by Wang, Yin, and Yu shouldn't precipitate widespread panic that the sky is falling; rather, we should use them as evidence that the sky has its limits, and parts of it could indeed fall someday. These results should provide ample motivation for cryptographers to take a hard look at how hash functions are designed and tested, and for the IT industry to ensure that keeping up with cryptographic advances is both possible and cost-effective. ▢

Reference

1. B. Schneier, *Applied Cryptography*, John Wiley & Sons, 1995.

David Naccache *is the managing director of Gemplus's Applied Research Centre and a professor at Royal Holloway, University of London and at University College, London. His research interests include public-key cryptography, mobile code security, and side channel attacks. Naccache has a PhD in cryptology from the Ecole Nationale Supérieure des Télécommunications Paris. He is a member of IACR, the ACM, and the IEEE. Contact him at david.naccache@gemplus.com.*

Peter Gutmann *is a researcher at the Department of Computer Science, University of Auckland, New Zealand. His research interests include security engineering and the practical application of cryptography. Gutmann has a PhD in computer science from the University of Auckland. He is a member of the IEEE, ACM, and IACR. Contact him at pgut001@cs.auckland.ac.nz.*

Charles C. Palmer *is a Department Group Manager at IBM's T.J. Watson Research Center in Hawthorne, New York. His research interests include security engineering, trusted computing, and high-assurance systems. Palmer has a PhD in computer science from Polytechnic University. He is a member of the IEEE Computer Society and the ACM. Contact him at ccpalmer@us.ibm.com.*