COMPUTER SCIENCE

Cryptologists Cook Up Some Hash for New 'Bake-Off'

A worldwide competition aims to keep the algorithms for authenticating electronic documents a jump ahead of forgers' ability to defeat them

In November, three Dutch cryptologists published on their Web site a "digital fingerprint" of their prediction of the winner of the United States's 2008 presidential election. According to them, the next commander in chief will be 3D515DEAD7AA16560ABA3E9DF05CBC80.

After the election, these modern-day Nostradamuses will use a simple mathematical procedure called a hash function, which is available on nearly all computers, to show that a PDF document created before the election has that hexadecimal number as its digital fingerprint. This will prove that they knew the winner all along, because the PDF document could not have been altered after the election. Any such change would make its fingerprint, known as a hash value, no longer match the one that has been published.

But it's all a trick. The "Nostradamus attack" by Marc Stevens, Arjen Lenstra, and Benne de Weger is designed to highlight a serious problem in cryptology: The so-called hash functions that many of the world's computers use for authenticating documents are dangerously out of date. The cryptologists prepared 12 separate documents, one saying that John McCain will win, one naming Hillary Clinton, and one even predicting Paris Hilton. By carefully tweaking the contents of each PDF document, in a way not obviously noticeable, they made it so each one generates the same digital fingerprint, computed by a widely used hash function called MD5. As the Dutch effort shows, the ability to produce multiple documents with the same fingerprint renders MD5 useless for authentication.

Without the ability to authenticate files, such as passwords and online transactions, Internet commerce would be seriously threatened. Therefore, the U.S. National Institute of Standards and Technology (NIST) in November announced a worldwide competition to select a new standard for hash functions, which is expected to conclude by 2012. The winner will be certified for U.S. government use, and if past history is any guide, that will make it a de facto standard for the rest of the world.

NIST held a similar "bake-off" (as some cryptologists called it) from 1997 to 2000 to select a new standard cipher for government use, called the Advanced Encryption Standard (AES). "The AES competition was the most fun I've ever had in cryptography," says Bruce Schneier of BT Counterpane in Santa Clara, California, who designed one of the five AES finalists and who plans to enter the hash-function competition as well. "Think of it as a giant cryptographic demolition derby: A bunch of us put our best work into the ring, and then we beat on each other until there was only one standing. ... I personally learned an enormous amount about [cipher design] from the AES competition, and we as a community benefited immeasurably."

Like the previous competition, the new bake-off offers no financial reward, and the submissions must be unpatented so that they can be incorporated into any software. The real payoff to the winner will be prestige. The AES competition drew 15 entries, and the hash-function competition is expected to attract as many as 50.

Although the contests will be similar, the products could hardly be more different. A cipher, like AES, encrypts data in such a way that it can be recovered but only by someone with a key. Hash functions, on the other hand, are not meant to be reversible, and they have no secret key. They merely show that a document is what it claims to be. They tend to be much simpler than ciphers and have a much broader range of applications—so many that they are often called the "duct tape" or "Swiss army knife" of cryptology.

> *"It may turn out that they aren't broken or can't be broken, but we didn't want to get caught out on the wrong side."*
>
> —WILLIAM BURR, NIST

Hash functions work by converting any string or message of 1s and 0s to a new string, usually much shorter. For example, the function might take a gigabyte MPEG movie file and boil it down to anywhere from 128 to 512 bits. In the current government standard, called SHA-1, the output, known as the hash, hash value, or hash sum, is 160 bits, whereas for MD5 it is 128 bits. The hash should look more or less random so that no one can guess what the original message said. On the other hand, it should be generated in a completely deterministic manner so that anybody who knows the hash function used can have a computer verify that the hash value matches the hashed file's contents. In addition, it should be staggeringly unlikely that any other document hashes to the same value. When that happens, cryptographers call it a "collision."

Compressing gigabyte-sized files to 160 bits inevitably results in many collisions. But the key point is that it is virtually impossible to find one by random search. That is because there are so many possible hash values—2^{160} of them for a 160-bit hash sum. For such a hash value, the number of movies you would have to hash before generating the same hash twice would be roughly 2^{80}, or a trillion a year for a billion years. Not even Bollywood is that prolific.

However, a hash function can be defeated if it is possible to deliberately create two colliding documents. If this takes more than 2^{80} attempts, then the hash function is considered secure, because such an attack is no better than random guessing. However, if a collision can be found in less than 2^{64} tries, then a supercomputer, or a very large network of personal computers, might be able to do it in a year. If it takes less than 2^{32} tries, one can do it on a PlayStation in a few minutes—as the Dutch "Nostradamuses" did.

The current standard, SHA-1, borrows its basic architecture from MD5, which was invented in 1992. You feed your message into a device that compresses and randomizes 512 bits at a time. You add another piece of your file to the first piece and feed it through again, and you repeat this procedure until you run out of message.

Iterative algorithms, like MD5 and SHA-1, are very easy to program and quick to run. But recently, they have come under heavy attack from cryptologists. In 2004 and 2005, cryptologist Xiaoyun Wang of Shandong University in China showed that MD5 could be cracked in fewer than 2^{40} steps, and SHA-1 in fewer than 2^{64}. No one has produced an actual collision in SHA-1, but a search, using the spare time of many personal computers, is under way at Graz University of Technology in Austria.

"If you find a [SHA-1] collision, people in industry will be forced to upgrade their products," says Bart Prencel, a cryptologist at

Published by AAAS

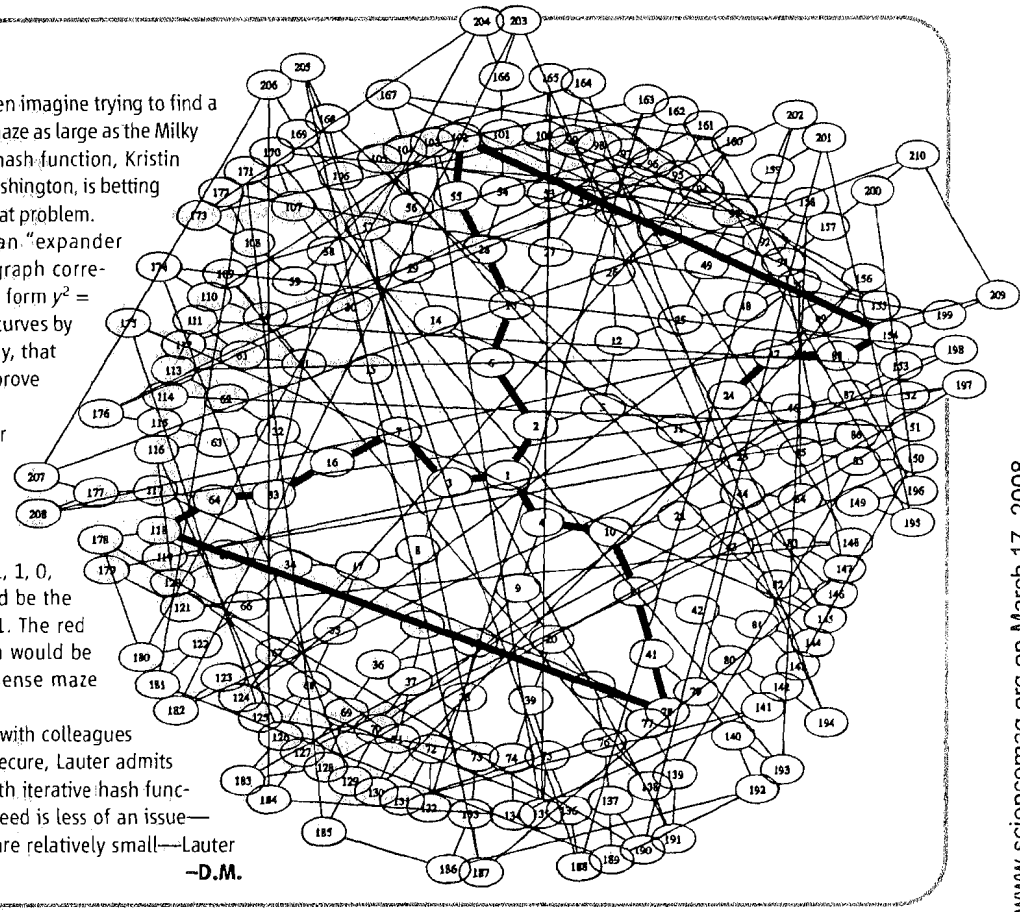
Hash of the Future?

Have you ever struggled to solve a maze? Then imagine trying to find a path through a tangled, three-dimensional maze as large as the Milky Way. By incorporating such a maze into a hash function, Kristin Lauter of Microsoft Research in Redmond, Washington, is betting that neither you nor anyone else will solve that problem.

Technically, Lauter's maze is called an "expander graph" (see figure, right). Nodes in the graph correspond to elliptic curves, or equations of the form $y^2 = x^3 + ax + b$. Each curve leads to three other curves by a mathematical relation, now called isogeny, that Pierre de Fermat discovered while trying to prove his famous Last Theorem.

To hash a digital file using an expander graph, you would convert the bits of data into directions: 0 would mean "turn right," 1 would mean "turn left." In the maze illustrated here, after the initial step 1-2, the blue path encodes the directions 1, 0, 1, 1, 0, 0, 0, 0, 1, ending at point 24, which would be the digital signature of the string 101100001. The red loop shows a collision of two paths, which would be practically impossible to find in the immense maze envisioned by Lauter.

Although her hash function (developed with colleagues Denis Charles and Eyal Goren) is provably secure, Lauter admits that it is not yet fast enough to compete with iterative hash functions. However, for applications in which speed is less of an issue—for example, where the files to be hashed are relatively small—Lauter believes it might be a winner. **–D.M.**



CREDIT: MICROSOFT RESEARCH

Katholieke Universiteit Leuven in Belgium. Anticipating such a breakdown, Microsoft in 2005 banned both SHA-1 and MD5 from new products and has removed MD5 from all its current products, says Kristin Lauter, head of the Cryptography Group at Microsoft Research in Redmond, Washington. Fortunately, a good backup is already available. In 2004, NIST issued several new standards, collectively called SHA-2, which are more secure than SHA-1 because they produce longer hashes (up to 512 bits instead of 160).

But NIST worries that SHA-2 could eventually fall, too. "Everything that has been attacked is in the same family," says William Burr of NIST's Security Technology Group. "It may turn out that they aren't broken or can't be broken, but we didn't want to get caught out on the wrong side."

After extensive debate, including two international workshops in 2005 and 2006, NIST decided that a new competition could turn up completely new approaches to hash functions. "We'll be reluctant to pick something that looks just like SHA-2," says Burr. "We want some biodiversity."

Although no designs have been formally submitted yet—the deadline is in October—experts predict that most entrants will continue to be iterative algorithms subtly retooled to defeat the new kinds of attacks. For instance, Preneel's RIPEMD—one of the few first-generation hash functions still standing—performs two parallel iterations, making it difficult for an attacker to figure out which one to attack.

A second approach, called "provably secure" hash functions, derives its presumptive security from math problems that are considered to be hard to crack (see sidebar, above). This type of algorithm typically does not require multiple iterations, but it does require cryptologists to put their faith in a mathematical "black box." Also, such algorithms tend to be slower than iterative algorithms because they require a more elaborate calculation—even though it is performed only once. Speed is at a premium for hash functions, as they are typically used to tag a document in the split-second it's electronically transmitted.

Not surprisingly, mathematicians love provably secure systems, whereas cryptologists have little use for them. "They are typically only provable with respect to one property but are weak with respect to other properties," says Joan Daemen of STMicroelectronics, co-winner of the AES competition. For instance, a "provably secure" hash developed by Lenstra and his colleagues, called Very Smooth Hash (VSH), was compromised last year when Markku-Juhani Saarinen at a Spanish company called Kinamik showed that it was easy to find "near-collisions" in VSH. In practice, engineers often truncate a long hash value to a shorter one, assuming that the truncated hash will inherit the long one's security. Saarinen's result means that they can't count on that with VSH.

In the final analysis, what makes it so hard to come up with good hash functions—and prove they work—is that they are expected to do so many things. "You expect them to do everything and blame them when they don't work," says Preneel. Perhaps a 4-year bake-off will be just what the chef ordered to make some new hash that will satisfy everybody's tastes.

–DANA MACKENZIE

Dana Mackenzie is a freelance writer in Santa Cruz, California.

Published by AAAS