Cryptographic hash functions map input strings of arbitrary length to short output strings (see Fig. 1). Unlike all the other cryptographic algorithms, no key or secret value is involved in their definition. Hash functions are used in a broad range of applications: to compute a short unique identifier of a string (e.g. for digitally signing a document or code in combination with a digital signature scheme), as one-way function to hide a string (e.g. for the protection of passwords or passphrases), to commit to a string in a cryptographic protocol, for key derivation (e.g., to compute an AES key from a key agreed with the Diffie-Hellman protocol) and for entropy extraction in pseudo-random bit generators. As very fast hash functions became available in the early 1990s, cryptographers started to design other primitives such as stream ciphers, block ciphers and MAC algorithms based on hash functions. The HMAC construction is perhaps the most successful example, as it is widely used in protocols such as IPsec, SSH, and SSL/TLS.

The first proposal to use hash functions in cryptography can be traced back to the 1976 seminal paper of Diffie and Hellman on public-key cryptography. Between 1976 and 1996, about 100 designs of hash functions have been proposed. Most of them have been broken, frequently even within a few months or even weeks after the publication of the design. Three of these hash functions, namely MD4, MD5 and the US government standard SHA-1 became very popular; as an example, in 2004 Microsoft Windows had 800 uses of the hash function MD5. Unfortunately, security analysis has demonstrated that the above three hash functions are insecure as well; MD4 and MD5 are particularly weak, since they can be broken in microseconds.

This article reviews the security requirements for hash functions. Next it explains why MD4, MD5 and SHA-1 are so widespread, discusses the weaknesses found in these functions and how these affect applications. We conclude by explaining what the solutions are to the hash function crisis: one can make some modifications to the applications or upgrade to SHA-2, or wait for the outcome of the SHA-3 competition.

Security Properties of Hash Functions
Cryptographic hash functions require three main security properties (see Fig. 2).

- **One-wayness or preimage resistance**: given a hash result \( y = h( x ) \) it should be hard to find any input \( x \) that maps to \( y \). This property is required when one stores in a computer system the hash value of a secret password or passphrase rather than the value itself. The assumption is that an attacker may obtain the list of hash values (in UNIX system this list is stored in etc/passwd) but that this should not reveal the passwords.
- **Second preimage resistance**: given an input \( x \) and its hash result \( y = h( x ) \) it should be hard to find a second distinct input \( x' \)
Collision resistance: it should be hard to find two distinct inputs x and x' such that h(x)=h(x'). Assume that an attacker who writes device drivers wants to use them to spread malware. In order to prevent this, the operating system vendor checks the device drivers; if they are clean they will be digitally signed; every copy of the operating system checks the digital signature before installing a new device driver. The attacker can defeat this measure by creating two versions of the device driver, namely a clean version x and a version with malware x' with the property that h(x)=h(x'). He can now submit x for inspection, and obtain the signature Sig(h(x)). Later on he can use the same signature to distribute the version x' with the malware. Collision resistance is also needed if one party commits in a protocol to a secret value x by sending h(x||r) to the other party, where r is a random string.

At first sight, second preimage resistance and collision resistance seem very similar: the result is that an attacker has two distinct messages with the same hash value. However, finding collisions is much easier than finding second preimages, because an attacker has much more freedom in a collision attack: he can freely choose both messages, while for second preimages the first message is fixed. For a flawless hash function with an n-bit result, finding a preimage or a second preimage takes about 2n hash function evaluations, while finding a collision requires only 2n/2 hash function evaluations. The reason for this is known as the birthday paradox: for a group of 23 people, the probability that two people have the same birthday is about 50%. The explanation is that such a group has 23^222 = 253 distinct pairs of people. On the other hand, a group of 182 people is needed to have a probability of 50% to have someone with a birthday on any given date.

While one typically considers in cryptography individual problems, solving one out of multiple instances can be a lot easier. If one has 2t inputs, finding a second preimage for any of the values requires only 2t-1 hash function evaluations; a similar observation holds for preimages. This problem can be solved by randomizing a hash function: every instance is made unique with a second randomly chosen input. In the context of UNIX passwords this randomizing parameter is called a ‘salt’.

In practice, one uses for (second) preimage resistance a hash function with at least n=128 bits. Even if one can attack 1 billion hash values in parallel (t=30), finding a (second) preimage within 1 year requires more than 10 trillion US$. On the other hand, one could find for such a hash function a collision in a few hours for 1 million US$. For long term collision resistance, a hash result of at least 256 bits is required.

In the past years other security properties have been identified, such as indistinguishability from a random oracle; however, the detailed discussion of these technical properties is beyond the scope of this article.

The rise and fall of MD4, MD5 and SHA-1

The first generation of hash functions was designed during the 1980s; many schemes were broken, and it was only near the end of the decade that the first theoretical results appeared. Around 1990, a very important developed occurred in cryptography: until then, most cryptographic algorithms were implemented in hardware, either in dedicated boxes to encrypt network communications or in hardware security modules to protect sensitive information on computers. As PCs became more powerful, and got connected to LANs and later on to the Internet, there was a growing need to implement cryptographic algorithms in software. However, the symmetric algorithms available at that time such as DES and LFSR-based stream ciphers were designed to be efficient and compact in hardware. In order to solve this problem, researchers started proposing new cryptographic algorithms that were more suitable to software implementations, such as the Snefru (from Merkle, who invented public key agreement in the mid 1970s) and MD4 and MD5 (from Rivest, the R in the RSA algorithm). Around the same time, Biham and Shamir invented differential cryptanalysis and managed to break DES (with a theoretical shortcut attack) and FEAL-8 (with a very efficient attack); Snefru, based on large tables, turned out to be vulnerable to this powerful technique, but MD4 and MD5 held up remarkably well. Both algorithms used addition mod 232, XOR, and bitwise operations, which were extremely efficient on the upcoming 32-bit RISC architectures. Overall, these algorithms were about 10 times faster than DES, which was a crucial advantage in the early 1990s. In addition, free source code for both algorithms was made available in 1991 and the algorithms and the code could be freely used (unlike Snefru that was patented). The RFCs 1320 and 1321 containing the code were both published in 1992. At the time all algorithms and code for encryption and decryption was tightly controlled by export laws; the restrictions on export of hash functions were less strict. All these elements contributed to the enormous popularity of MD4 and MD5 and can help to explain why Microsoft Windows had 800 uses of MD5. Internet protocols such as AP-OP, IPsec, SSH, SSL/TLS all use MD5 (and sometimes MD4). For authenticating network packets, a hash function had to be turned into a MAC algorithm, that takes as second input a shared secret key K. After failures of attempts such as the secret prefix method, h(K||x), the secret suffix method h(x||K), and the secret envelope method h(K||x||K), the standardized solution was HMAC, defined as MACG(x) = h(h(K ⊕ ipad || x) ⊕ opad), where ipad and opad are fixed strings. Note that APOP uses the secret suffix method based on MD5.

Very quickly after the publication of MD4 in 1990, it became apparent that with 48 simple steps its security margin was very small; this prompted Rivest to design MD5, that had 25% more steps (namely 64), where each step had some extra operations. In 1996, Dobbertin found collisions for MD4 in 220 operations which is much faster than the design goal of 264; his attack used sophisticated improvements of differential cryptanalysis. Eight years later, Wang et al. showed how to further extend differential cryptanalysis in order...
to find collisions for MD4 in a few operations (by hand). While MD5 was intended to be more secure, early results in 1993 and 1996 indicated that its security margin was very small; as a consequence a recommendation was issued in 1996 to stop using MD5 for applications that require collision resistance. In 2004, Wang et al. found collisions for MD5 in 15 minutes, again by using enhancements to differential attacks. Later on, these techniques were fine-tuned, resulting in collisions in microseconds. Stevens et al. managed to strengthen the techniques further; their work culminated in an attack in 2008 on a Certification Authority (CA) that still used MD5 to sign certificate; with the chosen prefix attack (also known as a correcting block attack), they managed to obtain a signature on a user public key that could also be used as a key for a rogue CA and thus impersonate any website on the internet. The attack was launched four years after the publication of the results by Wang et al., yet 6 CAs had still not upgraded their hash function. It is perhaps important to point out that the security of both MD4 and MD5 against brute force collision attacks (that do not require any knowledge of cryptanalysis) is 264 operations; this was already insufficient to protect against a motivated opponent in 2000. The best preimage attack for MD4 requires 2102 operations; this is less than the design goal of 2128, but still far beyond reach today; it has also been shown that for a small fraction of messages, finding second preimages is easy. The best known preimage attack on MD5 requires 2\textsuperscript{123} operations.

In 1993 NIST (the National Institute for Standards and Technology in the US) decided to standardize a hash function; they did not trust the security of MD4 and MD5 (perhaps based on their own cryptanalytic work) hence they proposed the Secure Hash Algorithm (SHA) designed by NSA. The SHA algorithm (today called SHA-0) had a 160-bit result, hence offering a security level of 2\textsuperscript{80} operations against brute force collision attacks. It is more than twice slower than MD5. Two years later, SHA was withdrawn and a new version called SHA-1 was published; the reason was an attack identified by NIST that was never published. Later on, the academic community has discovered serious weaknesses in SHA-0; the best known attack today finds collisions in about 1 hour. In 2004, Wang et al. surprised the cryptographic community by showing a collision attack on SHA-1 that requires 2\textsuperscript{80} operations rather than 2\textsuperscript{80}. Several teams have since then announced improvements, but so far no one has managed to produce a collision for SHA-1 or a convincing description of an attack with complexity less than 2\textsuperscript{80} operations. The best result is a collision for SHA-1 reduced to 75 out of 80 steps that was found in November 2011. For second preimages, a theoretical attack shows that for up to 61 steps SHA-1 does not have perfect behavior.

Solutions to the hash function crisis

A first solution is to replace MD4, MD5 and SHA-1 by hash functions with a larger security margin that are currently standardized. If that is not possible, one has to carefully examine the application in which the hash function is used to evaluate whether the security is still adequate. A third solution is to wait for the new standard SHA-3 that will be selected in late 2012.

In 2002, NIST published the SHA-2 family of hash functions that intend to offer much higher security levels than SHA-1; the SHA-2 family has output results varying from 192 bits to 512 bits. For outputs of 192, 224, and 256 bits, the operations are on 32-bit words (as for SHA-1) and the number of steps is 64. For the larger output lengths (384 and 512), 64-bit words are used and the number of steps is increased to 80. The steps themselves have become more complex, which clearly enhances the security. On the other hand, SHA-2 is still based on a combination of additions, XORs and Boolean operations and the main non-linear component consists of carries, just as for the other members of the MD4 family. No document has been published that justifies the design decisions; as NSA has made some mistakes earlier with SHA-0 and SHA-1, this has cast some doubts on the design. After one decade, the conclusion is that SHA-2 has withstood the current attack techniques: the most powerful attack is an attack that demonstrates deviations from randomness for 47 out of 64 steps of SHA-256; collisions faster than the birthday paradox seem to be possible for 53 steps with current techniques. On 32-bit architectures SHA-2 is more than four times slower than MD5, but for 64-bit architectures this factor is reduced to two.

There are other alternatives to SHA-1 that have been standardized in ISO 10118-3 (but not by NIST): RIPEMD-160 is a hash function from 1996 with a 160-bit result; it is 20% slower than SHA-1 but seems to have a substantial security margin. Whirlpool offers a 512-bit result; it security margin is not as large as hoped for, but it is still an interesting alternative based on very different design principles.

If it is not possible to replace the hash function, one can examine whether or not collision resistance is needed. While hash functions are widely used, there are only two important applications where collision resistance is needed: digital signatures in which an attacker can freely choose both documents that are signed and protocols using commitments. The main commercial applications are code signing and digital certificates. NIST has published the RMX mode, in which the data to be signed is randomized by the signer, hence collision attacks are rendered useless. This mode may not be sufficient for MD4 and MD5 but SHA-1 is likely to possess the security properties to make this solution work. One caveat is that of course the signer himself can still defeat this mode by choosing the randomness prior to the message. Stevens has also published an ad hoc solution: the collisions found with the current attacks have a particular structure, and one could scan for messages with this structure and reject them. This method can likely be defeated by a clever opponent who creates a variant of the current collision attacks.

If the opponent does not have any control over the message to be signed (or the message has been signed before 2004), an opponent needs to launch a second preimage attack. While one can imagine that such an attack becomes feasible for MD4 in the next few years, for MD5 this is still beyond reach, and for SHA-1 there is still a substantial security margin.

On the Internet, the most popular application of MD4, MD5 and SHA-1 is the HMAC construction. For HMAC-MD4, the best known attack has complexity 2\textsuperscript{64} (in both texts and computation). HMAC-MD5 can only be broken in a related key setting, in which an opponent can compute MAC values for different keys that are unknown but related in a specific way; the complexity of this attack is 2\textsuperscript{256} texts and 2\textsuperscript{100} operations; if proper key management is used, related key attacks should not be a concern. In a regular attack setting only 33 out of 64 steps can be broken. For HMAC-SHA-1 only 53 out of 80 steps have been broken so far. The conclusion is that HMAC-MD4 should not longer be used; HMAC-MD5 should be phased out as soon as convenient, while HMAC-SHA-1 seems still acceptable for the next 5-10 years. For the secret suffix method in APOP, the situation is much worse: for MD4 and MD5 secret keys can be recovered with a few thousand chosen texts and with a few seconds of computation. The security of SHA-1 with APOP is likely to be insufficient as well.

In the last decade some new structural or generic attacks have been identified, that all apply to most hash functions designed before 2000, that are iterated hash functions with an internal state size equal to the output size. One of these attacks (by Joux) shows that if the result of two iterated hash functions are concatenated (that is h(x) = h1(x) \| h2(x)) in order to get a much strong hash function,
The hash function crisis and its solution

The NIST SHA-3 Competition
An open call was published on November 2, 2007 for a hash function SHA-3 that would be compatible in terms of parameters with SHA-2 (results from 192 to 512 bits). The winner of the competition needs to be available worldwide without royalties or other intellectual property restrictions. Preparing a submission required a substantial effort, yet NIST received 64 submissions. Early December 2008, NIST has announced that 51 designs have been selected for the first round. On July 24, 2009, NIST announced that 14 algorithms have been selected for the second round. On December 10, 2010, the five finalists were announced: Blake, Grostl, JH, Keccak and Skein. Blake and Skein have a smaller internal state (although Skein has also a variant with a larger internal state) and both use the same operations as in MD4/MD5/SHA-1/SHA-2; moreover, the main building block is a kind of block cipher, while the other designs are built based on one (or two) permutations. Grostl and JH have a medium size internal state and Keccak has a large one (200 bytes). Grostl uses 8-bit S-boxes like AES, while JH and Keccak rely on smaller S-boxes (with 4 respectively 5 bits). In terms of performance, Blake and Skein seem to be more performant on high end processors, while Keccak is performing best in hardware; for embedded machines, all designs are slower than SHA-2. Keccak is the most original design, as it uses a new kind of construction called a sponge. For security, there is no clear picture yet. What is important to note is that all designs have been tweaked since their submission (in many cases rounds have been added to increase the security margin in response to attacks); some designs have been even changed twice.

A first observation is that the half-life of a hash function is about 9 months: by June 2008 half of the submissions were already broken. After this date, only strong functions remained (that were further improved), and the number of attacks has decreased. Most of the cryptanalysis work has been performed by European researchers; 3 of the 5 finalists have been designed in Europe, while the original 64 submissions had a much broader geographic spread. It is also interesting to point out that only 2 of the 64 submissions were based on a primitive the security of which could be reduced to a mathematical problem; as they were too slow, they were not selected for the second round. On the other hand, a large number of security reductions have been proven under the assumption that the underlying building block (such as a block cipher or a permutation) is ideal.

Security and performance updates on the SHA-3 competition can be found in the SHA-3 Zoo and eBASH websites that are maintained by the ECRYPT II project (http://www.ecrypt.eu.org).

Conclusions
We have witnessed a cryptographic meltdown in terms of collision resistance of widely used hash functions: schemes that were believed to be secure could be broken in milliseconds. Fortunately the implications of this meltdown have been very limited, because very few applications rely on collision resistance. For second preimage resistance and for constructions such as HMAC, the attacks have been less dramatic, but replacing MD4 and MD5 is essential.

One can be confident that the new SHA-3 algorithm will have a solid security margin and a good performance, even if it may be slower in some environments than SHA-2. Even if the SHA-3 design reflects the state of the art in 2008, there have been substantial advances in the theory of hash functions and our understanding today is much better than 10 years ago. Developers should start to plan an upgrade to SHA-3 by the end of 2012 or in early 2013.

Finally, application developers need to rethink how they use cryptography. In the early 1990s, the hash functions MD4 and MD5 were more than 10 times faster than DES and they were (wrongly) believed to be also much more secure. This explains why most cryptographic applications (both for network and computer security) prefer hash functions over block ciphers. An example of this is the use of HMAC rather than CBC-MAC. Today the roles are reversed: block ciphers are faster than hash functions, hence if performance is a concern block ciphers should be preferred. On modern processors, AES in software is six times faster than DES, while SHA-3 is likely to be two to three times slower than MD5, hence block ciphers are about twice faster than hash functions (on 64-bit machines the factor may be a bit smaller). This is illustrated in Fig. 3, that presents the performance of hash functions and block ciphers on AMD Intel Pentium D. Moreover, since 2010 high end Intel processors have dedicated AES instructions that give a speedup of a factor up to 10. This will further increase the advantage of AES, at least until special instructions are added for SHA-3.

While one can expect SHA-3 to be used for the next two decades, cryptographers will still keep looking for new hash function designs: one challenge is to design lightweight hash functions for environments with limited resources (power, energy, area); another problem is the design of hash functions with solid security proofs.

![Figure 2. Performance in cycles/byte of the hash functions MD4, MD5, SHA-1, RIPEMD-160, SHA-256, SHA-512, Whirlpool and the block ciphers DES and AES on an AMD Intel Pentium D 2992 MHz (f64) (source: http://bench.cr.yp.ta/index.html)](http://bench.cr.yp.ta/index.html)

### AUTHOR’S BIO

Prof. Bart Preneel received the Elect. Eng. and Ph.D. degrees from the University of Leuven (Belgium) in 1987 and 1993. He is a full professor in the COSIC research group at the University of Leuven. He has authored more than 400 scientific publications and is inventor of 3 patents. His main research interests are cryptography and information security and he frequently consults on these topics. He is president of the IACR (International Association for Cryptologic Research). He has served as program chair of 14 international conferences and he has been invited speaker at more than 70 conferences in 30 countries. In 2003, he has received the European Information Security Award in the area of academic research, and he received an honorary Certified Information Security Manager (CISM) designation by the Information Systems Audit and Control Association (ISACA).