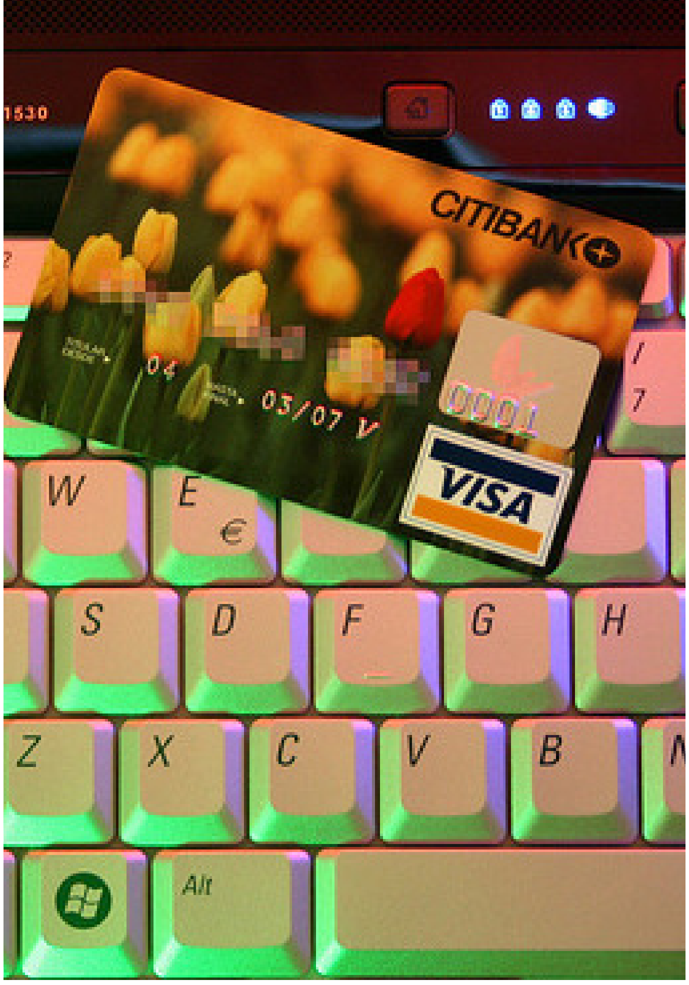
The Hex Factor: The NIST Hash Function Competition

Jacob Hilton

The fields of cryptography (code-making) and cryptanalysis (code-breaking) were transformed into mathematical disciplines by the fundamental breakthroughs of Claude Shannon in the 1940s. These subjects have since become an integral part of secure electronic communication, whose prevalence in everyday life increases as the digital revolution continues. One of the most significant recent advances in cryptography has been the development of new *cryptographic hash functions*. This process often requires considerable creativity because, despite their inherently non-random character, it is desirable for cryptographic hash functions to exhibit behaviour characteristic of random functions [1]. It is perhaps in part for this reason that the development of a new cryptographic hash function, to be called SHA–3, takes the form of a public competition. The USA's National Institute of Standards and Technology (NIST) announced the competition in November 2007 and received 64 entries, including contributions from France Télécom, IBM and Sony [2–4]. Five finalists were selected in December 2010, and the winner is due to be announced in 2012 [5].

Cryptography is essential to the security of online shopping. Reproduced from [16]



“A cryptographic hash function is akin to a method of encryption for which no method of decryption exists”

What is a cryptographic hash function?

A *hash function* (or *hash algorithm*) is an easy-to-perform, non-random procedure that takes a variable-length piece of data, known as a *message*, and produces a condensed representation, known as a *message digest* or *hash*. A basic example of a hash function for numerical data is adding together the digits of a number. Hash functions have many non-cryptographic applications, most importantly in optimising data retrieval and comparison. A *cryptographic hash function*, however, is distinguished by its resistance to cryptanalytic attack, known as its *security*. The most important types of resistance are the following [6]:

- *collision resistance*: hard to find two messages with the same hash
- *preimage resistance:* hard to find a message with a particular hash
- *second-preimage resistance:* hard to find a message with the same hash as another particular message

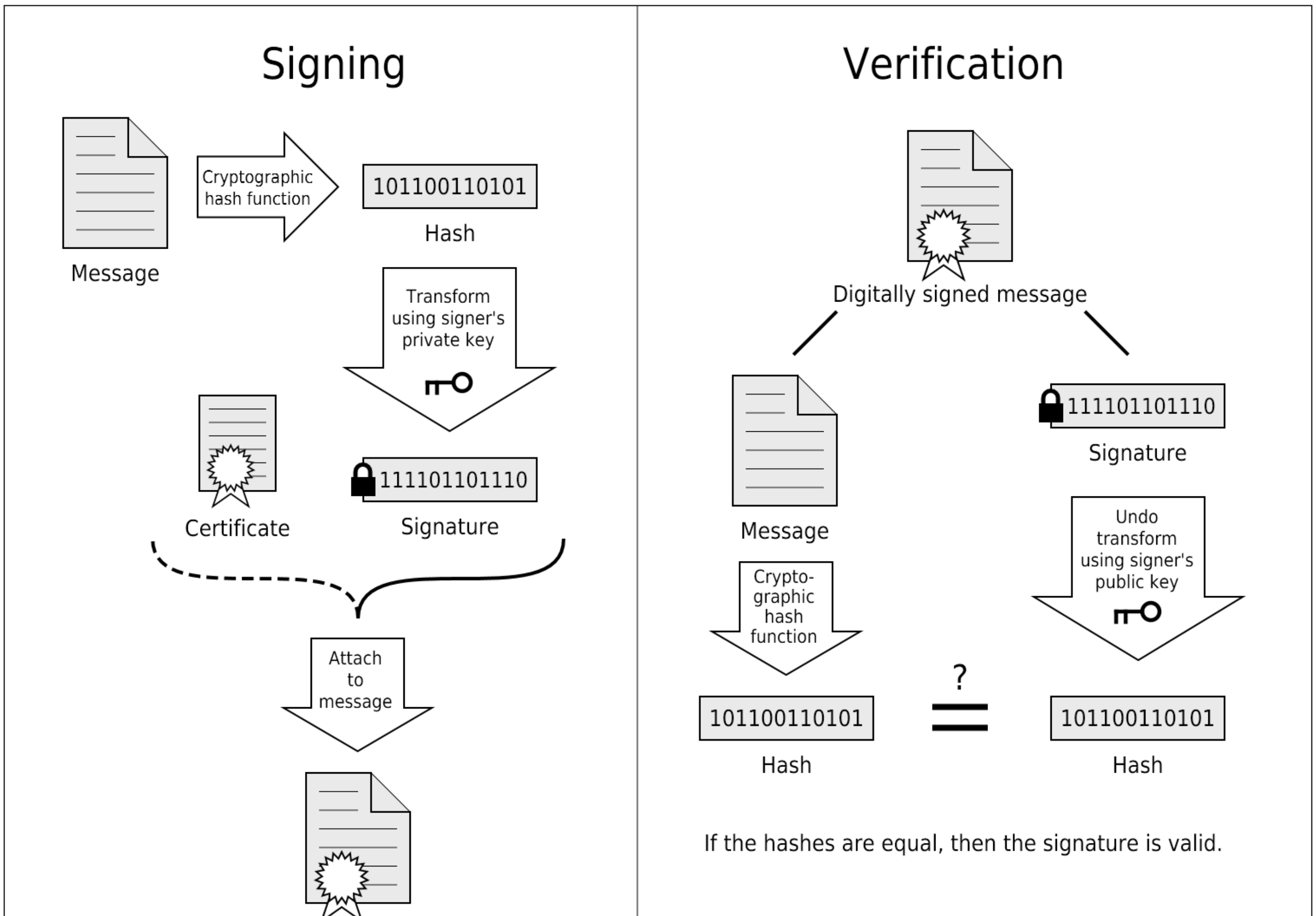
The example of adding together the digits of a number, for instance, would be a poor choice of cryptographic hash function on all three counts.

Cryptographic hash functions underpin many of the techniques of modern cryptography. Their applications include digital signature algorithms, password verification, message authentication algorithms, pseudorandom number generators and cryptographic key derivation functions. The first two of these are explained below.

Application to digital signature algorithms

Digital signature algorithms are the application for which most modern cryptographic hash functions were originally designed [6]. As the name suggests, the purpose of a digital signature is to provide the recipient of a message with confirmation that the message originated from a specific source. For example, the message may be a piece of software, in which case digital signature verification may be performed by anti-virus software.

A typical digital signature algorithm will involve *public-key cryptography* (such as the RSA algorithm), in which two *keys* are generated by the signer in advance: a *public key*, which is made available to the recipient, and a *private key*, which is kept secret, and is designed to be hard to deduce from the public key. A simple digital signature can then be realised as a transformation of the message using the private key that can be undone using the public key: forging a signature for a particular message is hard since the private



Schematic representation of how a typical digital signature is applied and verified. The *certificate* identifies the public key with the sender and is itself digitally signed by a *certificate authority*. Adapted from [17]

key is secret, yet the recipient is able to verify the signature by undoing the transformation and comparing the result with the message.

In a more typical version of this algorithm, a hash, acting as a proxy for the message, is transformed instead (see figure). This can dramatically improve the speed of the algorithm by eliminating the need for the entire message, which may be very long, from having to be transformed. It has the added benefit of preventing an attacker from being able to create a random forgery by simply choosing a random signature and computing the message to which it applies: if the attacker only has access to the hash, then the cryptographic hash function's preimage resistance makes it hard for the attacker to compute the message itself. Second-preimage resistance is also important in this application: otherwise it may be possible to pass off one message as another, such as a virus as another piece of software.

“A successful collision attack on an algorithm in the SHA-2 family could have catastrophic effects for digital signatures”

Application to password verification

Cryptographic hash functions can also be used to make passwords harder to steal. A simple method of password verification involves comparing a password entered by a user with the correct password. However, if hashes of these passwords are compared instead, then it becomes no longer necessary to store the password itself, only its hash. Hence, because of the cryptographic hash function's preimage resistance, someone with unauthorised access to this data will find it harder to recover the password. Moreover, its collision resistance ensures that comparing the hashes is a reliable way of testing whether the correct password has been entered.

This example illustrates the sense in which a cryptographic hash function can be said to protect data: it is akin to a method of encryption for which no method of decryption exists.

History of the SHA functions

NIST began the standardisation of cryptographic hash functions in May 1993 with the specification of the Secure Hash Algorithm (SHA), now called SHA–0 [6]. In April 1995, SHA–1, a revision of SHA–0 with improved security, was announced [7], though it took until August 1998 for a specific weakness in the collision resistance of SHA–0 to be demonstrated [8]. In August 2002, a new family of three algorithms, known collectively as SHA–2, was specified [9], each producing hashes of a different length (256, 384 and 512 bits compared with 160 bits for an SHA–1 hash). An additional variant (224 bits) was added to the family in February 2004.

Following the discovery in August 2004 of a weakness in a modified version of SHA–1, NIST announced plans to phase out SHA–1 in favour of SHA–2 by 2010 [10]. Then, in February 2005, a similar weakness was discovered in the original version of SHA–1 [11]. This, along with the similarities between SHA–1 and SHA–2, prompted NIST to announce the competition to choose SHA–3 in November 2007. They reasoned, "Although there is no specific reason to believe that a practical attack on any of the SHA–2 family of hash functions is imminent, a successful collision attack [i.e. weakness in collision resistance] on an algorithm in the SHA–2 family could have catastrophic effects for digital signatures." [6]

"The strength of the remaining candidates suggests a bright future for SHA-3"

Selection of the finalists

The second round of the competition ended in December 2010 with the selection of five algorithms as finalists: BLAKE, Grøstl, JH, Keccak and Skein [12]. NIST based the decision on the security, performance (i.e. efficiency of computer implementations), flexibility and simplicity of the candidates [12], stating, "Security was our greatest concern … . However, it is meaningless to discuss the security of a hash function without relating security to performance" [13]. The security of an algorithm was evaluated based on arguments presented by its designers, feedback from the community and NIST's own cryptanalysis. Evaluation involved estimating the susceptibility of the algorithm to future attacks, known as its *security margin*, whilst taking into consideration the quantity of cryptanalysis received [12].

As an example of one of the finalists, Grøstl is considered here in more detail. In common with SHA–1 and SHA–2, Grøstl uses a version of the *Merkle–Damgård hash function construction,* meaning that the message is split into fixed-length blocks, which are combined one by one using a *compression function,* which converts two fixed-length messages (denoted here by m_1 and m_2) into a single message of the same fixed length. The compression function used in Grøstl (denoted here by f) incorporates two functions (denoted here by P and Q) that essentially reorder the characters of a message, and a method of combining messages called *bitwise exclusive OR*, denoted ⊕, which adds the binary representations of the messages without carries (for example, $10 \oplus 11 = 01$). The compression function is given by [14]:

$$f(m_1,m_2) = P(m_1 \oplus m_2) \oplus Q(m_2) \oplus m_1$$

NIST stated, "Grøstl was selected as a finalist because of its well-understood design and solid performance, especially in hardware. While Grøstl's security margin is not ideal, NIST views it in light of the extensive amount of cryptanalysis that has been published, both on Grøstl itself and the … structure on which Grøstl is based" [12].

Conclusion

When the competition was announced, NIST's stated aim was to "specify an unclassified, publicly disclosed algorithm, which is available worldwide without royalties or other intellectual property restrictions, and is capable of protecting sensitive information for decades." [6] While hash functions remain one of the most poorly understood areas of cryptography [15], leaving considerable uncertainty, the enormous progress made in recent years and the strength of the remaining candidates suggest a bright future for SHA–3, in which NIST's ambitious target may well be realised. ■

Jacob Hilton is a third year student studying Mathematics at Trinity College

References:

1. Bellare M, Rogaway P. Random Oracles are Practical: A Paradigm for Designing Effcient Protocols. In: Denning D, Pyle R, Ganesan R, Sandhu R, Ashby V, editors. Proceedings of the 1st ACM Conference on Computer and Communications Security; 1993 Nov 3–5; New York City: ACM; 1993. p. 63.
2. ECHO hash function [homepage on the Internet]. Paris: France Télécom; [cited 2011 Aug 23]. Available from: http://crypto.rd.francetelecom.com/ECHO/
3. The Hash Function Fugue [homepage on the Internet]. Hawthorne, New York: IBM; [updated 2010 Dec 7; cited 2011 Aug 23]. Available from: http://domino.research.ibm.com/comm/research_projects.nsf/pages/fugue.index.html
4. Iwata T, Shibutani K, Shirai T, Moriai S, Akishita T. AURORA: A Cryptographic Hash Algorithm Family [monograph on the Internet]. Graz, Austria: Graz University of Technology; 2008 Oct 31 [cited 2011 Aug 23]. Available from: http://ehash.iaik.tugraz.at/uploads/b/ba/AURORA.pdf
5. Cryptographic Hash Algorithm Competition [homepage on the Internet]. Gaithersburg, Maryland: NIST; [updated 2010 Dec 13; cited 2011 Aug 23]. Available from: http://csrc.nist.gov/groups/ST/hash/sha-3/index.html
6. Kayser RF. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA–3) Family. Federal Register. 2007 Oct 29;72(212):62212–62220. Available from: http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf
7. US Department of Commerce, NIST. Secure Hash Standard. FIPS 180–1. 1993 May 11. Available from: http://www.itl.nist.gov/fipspubs/fip180-1.htm
8. Chabaud F, Joux A. Differential Collisions in SHA–0. In: Krawczyk H, editor. Advances in Cryptology – CRYPTO '98, 18th Annual International Cryptology Conference; 1998 Aug 23–27; Berlin: Springer; 1998. p. 56–71. Available from: http://fchabaud.free.fr/English/Publications/sha.pdf
9. US Department of Commerce, NIST. Secure Hash Standard. FIPS 180–2. 2002 Aug 1. Available from: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf
10. NIST. NIST Brief Comments on Recent Cryptanalytic Attacks on Secure Hashing Functions and the Continued Security Provided by SHA–1[document on the Internet]. Gaithersburg, Maryland: NIST; 2004 Aug 25 [cited 2011 Aug 23]. Available from: http://csrc.nist.gov/groups/ST/toolkit/documents/shs/hash_standards_comments.pdf
11. Wang X, Yin YL, Yu H. Collision Search Attacks on SHA1. In: Shoup V, editor. Advances in Cryptology – CRYPTO 2005: 25th Annual International Cryptology Conference; 2005 Aug 14–18; Berlin: Springer; 2005. p. 17–36. Available from: http://www.c4i.org/erehwon/shanote.pdf
12. Turan MS, Perlner R, Bassham LE, Burr W, Chang D, Chang S, Dworkin MJ, Kelsey JM, Paul S, Peralta R. Status Report on the Second Round of the SHA–3 Cryptographic Hash Algorithm Competition. NIST Interagency Report 7764. 2011 Feb. Available from: http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/documents/Round2_Report_NISTIR_7764.pdf
13. Burr WE. The SHA–3 Finalists [document on the Internet]. Gaithersburg, Maryland: NIST; 2010 Dec 9 [cited 2011 Aug 23]. Available from: http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/documents/Email_Announcing_Finalists.pdf
14. Gauravaram P, Knudsen LR, Matusiewicz K, Mendel F, Rechberger C, Schläffer M, Thomsen SS. Grøstl – a SHA–3 candidate [monograph on the Internet]. 2011 Mar 2 [cited 2011 Aug 23]. Available from: http://www.groestl.info/Groestl.pdf
15. Schneier B. America's Next Top Hash Function Begins [article on the Internet]. New York City: Condé Nast; 2008 Nov 19 [cited 2011 Aug 23]. Available from: http://www.wired.com/politics/security/commentary/securitymatters/2008/11/securitymatters_1120/
16. CC-BY. By Franganillo, J. Seguridad. Available from: http://www.flickr.com/photos/franganillo/4458502219/
17. CC-BY-SA. Digital Signature diagram originally by Acdx. Available from: http://en.wikipedia.org/wiki/File:Digital_Signature_diagram.svg