Further progress in hashing cryptanalysis*

Arjen K. Lenstra
Lucent Technologies, Bell Laboratories

February 26, 2005

Abstract

Until further notice all new designs should use SHA-256. Existing systems using SHA1 or MD5 should confirm that they only need second pre-image resistance, not random collision resistance. Usage of MD5 in certificates should be discontinued unless the presence of adequate mitigating controls has been verified.

1 New attacks against hash functions

There is a lot of turmoil in the hashing world. First, in August 2004, new attack methods were presented[1,2,3,4] that mostly affected hash functions that were already known to be weak, among others MD4, MD5, and SHA-0. In February 2005, a similar result was announced[5] that affects SHA-1, a hash function that was believed to resist the new types of attack. Furthermore, independent developments cast serious doubt on the iterative design principle underlying most current hash functions[6,7].

If the security of your design relies on MD4 or MD5 (already known to be weak since at least 1995) or on SHA-1 (believed to be sufficiently strong

*`http://cm.bell-labs.com/who/akl/hash.pdf`

[1]X. Wang, F. Guo, X. Lai, H. Yu, *Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD*, eprint archive 2004/199, `http://eprint.iacr.org/2004/199`, presented at the Crypto 2004 rump session, August 17, 2004.

[2]X. Wang, H. Yu, *How to Break MD5 and Other Hash Functions*, Eurocrypt 2005, to appear.

[3]X. Wang, X. Lai, F. Guo, H. Chen, X. Yu, *Cryptanalysis for Hash Functions MD4 and RIPEMD*, Eurocrypt 2005, to appear.

[4]A. Joux, *Collisions for SHA0*, presented at the Crypto 2004 rump session, August 17, 2004.

[5]X. Wang, Y.L. Yin, H. Yu, *Finding collisions in the full SHA1*, submitted for publication, February 2005.

[6]A. Joux, *Multicollisions in iterated hash functions. Application to cascaded constructions.* Proceedings Crypto 2004, Springer-Verlag LNCS 3152 (2004) 306–316.

[7]J. Kelsey, B. Schneier, *Second preimages on n-bit hash functions for much less than 2^n work*, eprint archive 2004/304, `http://eprint.iacr.org/2004/304`; Eurocrypt 2005, to appear.

until February 2005), should you rush out and replace it? Chances are you don't, but it depends on what is required for a successful spoofing attempt given the particular properties of your design. The purpose of the present note is to describe in which circumstances the new attacks can be exploited. It is recommended to review each hash application on a case by case basis to decide if the new attack poses a realistic threat. If so it must be analysed if there are factors mitigating the vulnerability and to evaluate its associated risk. Finally, it must be assessed if the potential damages outweigh the replacement cost.

More absolute general recommendations were issued by the U.S. National Institute of Standards and Technology (NIST)[8]: usage of MD5 – and thus MD4 – in certificates or for digital signatures should be discontinued, and based on advances in computer processing capabilities it would be prudent to phase out SHA-1 by 2010. It can be argued that there are applications where the first recommendation may be overly conservative because the replacement cost would not be commensurate with the risk associated with continued usage. Furthermore, the second recommendation may be regarded as overly liberal in view of the latest SHA-1 developments.

2 Exploiting hash collisions

A widespread application of hash functions is in *digital certificates*. An existing digital certificate may contain the hash $H(b)$ of a bitstring b. To spoof that certificate, the attacker has to find a bitstring b', different from b, but with the same hash as b. Given $H(b)$, and possibly b itself, finding a different b' whose hash value *collides* with it (i.e., $H(b) = H(b')$) is still considered to be sufficiently hard for all common hash functions.

This property of hash functions, that they are hard to invert, is often referred to as *target collision resistance*. If b is given it is called *second pre-image resistance*. Hash functions with n-bit values are supposed to be designed in such a way that finding a second pre-image should take on the order of 2^n operations. Unfortunately, that is not the case for iterative hash functions – and all common hash functions such as MD4, MD5, and all SHA-variants are iterative. For all these functions, second pre-images can be computed 2^k times faster, for values of $k \geq 0$ that cannot be too large compared to the hashlength n. For instance $k \leq 55$ for SHA-1. However, the computation requires an amount of memory proportional to 2^k. It is therefore considered to be practically infeasible. It was shown recently that this very general result is a straightforward consequence of Joux's work (about which more below). For MD4, MD5, and the SHA functions it was already known for several years. This was never found to be a point of concern. So, all common hash functions such as MD4 and MD5 (both with $n = 128$) or SHA-0 and SHA-1 (with $n = 160$) are still believed to offer adequate protection against spoofing attempts that require finding a second pre-image for an existing hash value.

[8]http://www.fcw.com/fcw/articles/2005/0207/web-hash-02-07-05.asp

Thus, the security of existing certificates relies on the fact that for a given particular bitstring and its hash value it is hard to come up with another bitstring with the same hash value. But there are also applications of hash functions where the hash value is not given in advance. That gives the attacker a much larger degree of freedom. For instance, in digital signatures one customarily signs the hash $H(d)$ of a document d (a bitstring). In this scenario an attacker may construct a pair of different bitstrings d, d' such that $H(d) = H(d')$, but without any further restrictions on the particular value $H(d)$. The attacker then digitally signs d (using $H(d)$), but later claims that not d but d' was signed. Since $H(d) = H(d')$ there is no way to deny the attacker's claim. Therefore hash functions must be *collision resistant*: it must be computationally infeasible to construct different bitstrings that hash to the same value.

Collision resistance puts a much heavier demand on the design and length of hash functions than target collision resistance. This is due to the so-called *birthday paradox*: the probability that among a group of 23 randomly selected people at least two people have the same birthday is more than 50 percent. Because this probability is much higher than most people would expect it is referred to as a paradox – but there is nothing paradoxical about it: it is a plain fact that can be proved mathematically and verified empirically that if elements are drawn at random from a collection of N objects (with replacement) then the expected number of draws before an element is drawn twice is approximately $1.26\sqrt{N}$. In the birthday example N would be 365. In the context of n-bit hash functions, there are $N = 2^n$ different objects (i.e., hash values). After hashing about $1.26 \times 2^{n/2}$ randomly selected different bitstrings one may expect (under reasonable assumptions about the random behavior of the hash function) that two bitstrings are found with the same hash value. Thus, for the digital signature application sketched above, the effort required by an attacker to successfully create a collision is at most on the order of 2^{64} if any 128-bit hash function is used, and at most about 2^{80} for any 160-bit hash.

From a security point of view collision resistance effectively doubles the hash length requirement compared to second pre-image resistance. This is a general fact that applies to all hash functions, past and future. The new hash cryptanalysis results, however, made clear that for many common hash functions the situation is much worse with respect to collision resistance. It was shown that random collisions (not second pre-images) can be found by hand for MD4 and computed in a matter of minutes for MD5. With $n = 128$, both were designed to resist an effort of 2^{64} against finding such collisions. For SHA-0 with $n = 160$, collisions can be found[9] with effort less than 2^{39}, much lower than the intended design effort 2^{80}. And, last but not least, for SHA-1, also with $n = 160$, collisions can be found with effort 2^{69}, somewhat lower than the intended 2^{80}.

For practical applications this means that MD4, MD5, and SHA-0 should no longer be used in circumstances where an attacker is free to choose the value

[9] X. Wang, H. Yu, Y.L. Yin, *Efficient collision search attacks on SHA0*, submitted for publication, February 2005.

that will be hashed. This is the case even if the values to be hashed have to meet stringent structural requirements, such as values that are contained in certificates, because actual hash collisions have been constructed[10] that meet those requirements. On the other hand, because MD4 and MD5 were already for a long time known to be weak with respect to collision resistance, and SHA-0 should never have been used anyway, the impact of the new findings should be limited. Nevertheless, the cryptographic community was astonished to see how much weaker MD4, MD5, and SHA-0 turned out to be than anyone had expected. After these attacks had been announced, NIST wrote[11] 'SHA-1 not broken' and 'Not much reason to expect it will be any time soon'. It remains to be seen if the 2^{69} random collision attack against SHA-1 will be a real concern. But given the developments that were taking place in the fall of 2004 it is hard to imagine that it came as a real surprise. The recent SHA-1 extensions SHA-i for $i = 224, 256, 384$, and 512, are as far as we know now not affected by the new attack methods.

3 Conclusion

What should be done in practice? For existing applications of affected hash functions one should consider if the random collision attack scenario applies. If it does, a proper risk analysis should be carried out. If there are mitigating factors that may render the likelihood of successful attacks sufficiently low, one may decide not to take action; otherwise replacement of the hash function may be in order. For new applications the affected hash functions must not be used. For the moment this limits the choice of hash functions mostly to the extensions of SHA-1, **not** including SHA-1 itself. This is different from the situation before the new findings were announced because SHA-1 can no longer be recommended. NIST recommends using SHA-256, and does not seem to be overly concerned about usage of SHA-1 until 2010.

Is this the end of the story? Should we now all happily use SHA-256 and hope for the best? Or should we go for an overhaul and total redesign of our hashing methods and come up with something that is better? Consider the situation in more detail. All common hash functions, including MD4, MD5, and SHA-0/1/224/256/384/512, follow the same basic approach, with just a single relatively minor change setting apart SHA-1 from MD4, MD5, and SHA-0, and with some additional twists for SHA-224/256/384/512. Until a few weeks ago, it was hoped that the minor change offers adequate protection against the collision attacks that were announced in August of 2004. The February 2005 cryptanalytic announcement about SHA-1 dashed those hopes. With their predecessors already completely picked apart, and the apparently unhealthy iterative design principle to begin with, how long before also SHA-224/256/384/512 are

[10] A.K. Lenstra, B. de Weger, *On the possibility of constructing meaningful hash collisions for public keys*, submitted for publication, February 2005; see also http://www.win.tue.nl/~bdeweger/CollidingCertificates/

[11] http://csrc.nist.gov/pki/twg/y2004/Presentations/twg-04-14.pdf

affected? Adi Shamir, one of the world's most dreaded cryptanalysts and most respected cryptographers, recommends starting afresh, from scratch. Given how long the Advanced Encryption Standard process took, it may be a while before a better new hashing standard emerges. For the foreseeable future, however, SHA-256/384/512 will be the hashes of choice. Given the recent developments, protocol designers need to provide for changes in hash functions, as many already do, and when doing so need to ensure that version rollback attacks are prevented.

Sketch of Joux's argument. As a final point of interest, independent of the commotion caused by the various collision attacks, there is a very elegant and surprisingly simple result by Antoine Joux about the concatenation of hash functions. If the results of two independent n-bit hash functions are concatenated, then according to crypto-folklore the result is as good as a $2n$-bit hash function: finding a second pre-image should take effort $2^n \times 2^n = 2^{2n}$, and finding a random collision should take effort $2^{n/2} \times 2^{n/2} = 2^n$. It was shown that if one of the hash functions is a so-called iterative hash function – and all common hash functions are iterative – then concatenation leads to hardly any additional security, thereby not only for all practical purposes refuting the folklore assumption but also showing a severe weakness in the basic design principle underlying most current hash functions. But the most remarkable aspect of Joux's result may be that it took so long for such a straightforward argument to be published, strongly suggesting that hash-research is still in its infancy. A sketch of Joux's argument follows.

An n-bit iterative hash function splits the input in a number of fixed size blocks, say B_1, B_2, ..., B_r. The hash is calculated in r rounds as a function of the r blocks and a fixed n-bit *initialization vector* R_0 that depends only on the hash function: for $i = 1, 2, \ldots, r$ a round function is applied to R_{i-1} and B_i and produces an n-bit value R_i. The resulting hash is the n-bit value R_r. Thus, in the ith round the round function is applied to the result of the previous round (or the fixed initial value R_0 if $i = 1$) and the ith input block.

Now construct a collision for an n-bit iterative hash function H: values x_{11} and x_{12} with $x_{11} \neq x_{12}$ such that $H(x_{11}) = H(x_{12})$. This takes at most about $2^{n/2}$ operations. Denote $H(x_{11}) = H(x_{12})$ by C_1. Similarly, it takes at most about $2^{n/2}$ operations to construct a collision for H where its initialization vector is replaced by C_1: x_{21} and x_{22} with $x_{21} \neq x_{22}$ such that $H_{C_1}(x_{21}) = H_{C_1}(x_{22})$, where the subscript C_1 indicates usage of C_1 as initialization vector as opposed to the default initialization vector. It then follows from the way iterative hash functions work that H applied to the concatenation of x_{1i} and x_{2j}, with $i, j \in \{1, 2\}$, always results in the same value, say C_2, independent of the choices of i and j.

So, the two pairs (x_{11}, x_{12}) and (x_{21}, x_{22}) result in a four-way collision: four distinct values $x_{11}||x_{21}$, $x_{11}||x_{22}$, $x_{12}||x_{21}$, and $x_{12}||x_{22}$ (with '$||$' denoting concatenation) that all have the same hash value C_2. This four-way collision can then be concatenated with a newly constructed collision for H_{C_2} resulting in an eight-way collision to C_3, the eight-way collision is concatenated with a collision for H_{C_3} for a sixteen-way collision, etc. Repeating this construction $m/2$ times we find that a $2^{m/2}$-way collision for H can be found after at most about $(m/2) \cdot 2^{n/2}$ operations: $2^{m/2}$ different inputs that all hash to the same value under H.

For *any* m-bit hash function G one may expect, based on the birthday paradox, that among those $2^{m/2}$ different inputs that collide for H there is a pair that collides for G as well. This implies that a collision can be found for the hash function consisting of the $n+m$-bit concatenation of the hash functions H and G. This takes, essentially, only $m/2$ times the effort to find a collision for H, plus about $2^{m/2}$ applications of G to identify the G-collision. That is much less than the effort $2^{(n+m)/2}$ that one would expect for a decent $n+m$-bit hash function. Note that the argument works for any m-bit hash function G, and that only H has to be iterative.