Available online at www.elixirpublishers.com (Elixir International Journal)

Applied Mathematics

Elixir Appl. Math. 38 (2011) 4418-4422

ISSN: 2229-712X

Several types of defined attacks on cryptographic hash function

C. Krishna Kumar and G. Jai Arul Jose
Sathyabama University, Chennai.

ARTICLE INFO

Article history:
Received: 22 August 2011;
Received in revised form:
26 August 2011;
Accepted: 31 August 2011;

Keywords
Hash functions,
Plaintext attack,
Preimage.

ABSTRACT
Hash functions, also called message digests and one-way encryption, are algorithms that, in some sense, use no key. Instead, a fixed-length hash value is computed based upon the plaintext that makes it impossible for either the contents or length of the plaintext to be recovered. Hash algorithms are typically used to provide a digital fingerprint of a file's contents, often used to ensure that the file has not been altered by an intruder or virus. Hash functions are also commonly employed by many operating systems to encrypt passwords. Hash functions, then, provide a measure of the integrity of a file. The goal of this paper is to give an overview of the known methods of attack on hash functions.

Introduction

Methods of attack on hash functions

The large number of possible attacks will be classified in five types:

1. attacks independent of the algorithm,
2. attacks dependent on the chaining,
3. attacks dependent on an interaction with the signature scheme,
4. attacks dependent on the underlying block cipher,
5. high level attacks.

One can make a distinction between the following cases, depending whether the value of IV is different from the specified value:

Preimage: here an attacker tries to find a preimage for a given hashcode.

Second preimage: here an attacker tries to find a second preimage for a given hashcode.

Pseudo-preimage: here an attacker tries to find a preimage for a given hashcode, with IV/ □ IV .

Second pseudo-preimage: here an attacker tries to find a second preimage for a given hashcode, with IV/ □ IV (note that this case will not be discussed in the following).

Collision: here an attacker tries to find a collision.

Collision for different IV: here an attacker tries to find a collision for IV/ □ IV.

Pseudo-collision: here an attacker tries to find for some IV/ and IV// a pair X/, X//, such that hIV/(X/) = hIV//(X//).

It is clear from the definitions that finding a pseudo-collision can be not harder than finding a pseudo-preimage, and that finding a collision can be not harder than finding a (second) preimage. A similar taxonomy was suggested in [1], but they make no distinction between second preimage and preimage. Their terminology for second (pseudo-) preimage is "(free-start) target attack", and the two last collision attacks are called "semi-free-start" respectively "free-start collision attack".

For a MAC the situation is more complicated. The proposed taxonomy is equivalent to the taxonomy of [2] for digital signature schemes. Depending on the information available to an attacker, the following types of attacks are distinguished:

Known plaintext attack: here an attacker is able to examine some plaintexts and their corresponding MAC.

Chosen plaintext attack: here an attacker is able to select a set of plaintexts, and subsequently he will obtain a list of MAC's corresponding to these plaintexts.

Adaptive chosen plaintext attack: this is the most general attack where an attacker will choose a plaintext and immediately receive the corresponding MAC: the choice of a plaintext can depend on the outcome of previous questions.

"Breaking" a MAC can have different meanings:

Total break: this means that an attacker can determine the secret key K.

Universal forgery: in this case an attacker can find an algorithm that is functionally equivalent to the MAC evaluation algorithm.

Selective forgery: here an attacker can determine the correct MAC for a particular plaintext chosen a priori by him.

Existential forgery: here an attacker can determine the MAC for at least one plaintext. As he has no control over this plaintext, it may be random or nonsensical.

An evaluation of a scheme for message authentication or a digital signature strongly depends on the information at the disposal of an adversary, the actions he can undertake and finally on the consequences of both a successful and an unsuccessful attack. In general, a conservative approach is recommended. This implies that one assumes that a MAC will be considered to be broken if an attacker can commit an existential forgery based on an adaptive chosen message attack with the only restriction on the number of plaintexts coming from limited storage and computation capacities.

Attacks independent of the algorithm

This class of attacks depends only on the size of the hashcode n and the size of the secret key k (for a MAC), and is independent of the nature of the algorithm. It is assumed that the hashcode is a uniformly distributed and independent random variable: if this is not the case this class of attacks will be even more successful.

Random attack

The opponent selects a random message and hopes that the change will remain undetected. If the hash function has the required random behavior, his probability of success equals $1/2^n$

Tele:
E-mail addresses:

with n the number of bits of the hashcode. A major difference between a MAC and an MDC is that for a MAC the attack has to be carried out on-line, and hence the attack depends on two elements:

• The number of trials T that can be carried out, which depends on the speed of the implementation and on the time an attacker has access to the system with the key in place. The number of trials can be limited by undertaking a special action (e.g., perform manual verifications) if the number of erroneous results exceeds a certain threshold. An example where a large number of trials is possible, is the control of a satellite authenticated by a MAC.

• The expected value V of a successful attack. In wholesale banking this can be on the order of 100 million $ or even more.

The expected value of an attack equals then $T \cdot V/2^n$. If the number of trials can be limited, or if the expected value is limited like in retail banking, a value of n = 32 is sufficient. However, for most applications it is recommended that the size of the hashcode is at least 64 bits. For an MDC the attack can be carried out off-line and in parallel. This means that the length of the hashcode n should be at least 64 bits. If a significant number of messages can be attacked at the same time, it is advisable to select a larger value of n. In section 2.4.1 it has been shown that finding a preimage for long messages is easier, unless the length of the message is included in the padding.

Exhaustive key search

An exhaustive search for a key is only applicable to a MAC. It is a known plaintext attack, where an attacker knows M plaintext-MAC pairs for a given key. He will precompute the MAC for every possible key in order to eliminate wrong keys. The size of the key space is equal to k bits, and the expected number of keys that remain will be denoted with Kexp. If M is sufficiently large, it is possible to determine the key uniquely or Kexp ≈ 1. The relation between Kexp, M and n can be determined [3] under the assumption that the MAC is a random mapping, and that no key clustering occurs, i.e., that there are no equivalent keys. For the correct key, an attacker will perform M MAC calculations, while for a bad key the probability that exactly i trials are performed is equal to

$$\left(1 - \frac{1}{2^n}\right) 2^{-n(i-1)}$$

The expected number of trials is given by the following expression:

$$\left(1 - \frac{1}{2^n}\right) \sum_{i=1}^{M} \frac{i}{2^{n(i-1)}} < \frac{1}{1 - 2^{-n}}$$

The total number of trials to identify the key is upper bounded by

$$M + \frac{2^k - 1}{1 - 2^{-n}}$$

and the number of keys that remains is expected to be

$$K_{exp} = 1 + \frac{2^k - 1}{2^{Mn}}$$

This means that the number of plaintext-MAC pairs to determine the key uniquely is slightly larger than k/n. After the birthday attack it will be discussed how large k has to be in order to offer sufficient security for the next decades.

Birthday attack

The idea behind this attack is that for a group of 23 people the probability that at least two people have a common birthday exceeds 1/2 [4] . Because this number of people is significantly smaller than what one would expect, this has also been called the "birthday paradox". For some applications a related problem is relevant: if two groups of people have 17 persons each, the probability that two people in the two different groups have a common birthday will also exceed 1/2. Note that these results assume that birthdays are randomly distributed over the year; as this is not the case the probability will be even higher. This can be generalized as follows. If two samples of size r1 and r2 are drawn from a set of n elements, and if r1r2 = n with r1, r2 = O(√n), then the probability of a match equals 1 − 1/e or 63%. Note that if the attacker is unlucky, it is sufficient to increase the size of r1 and r2 slightly, which will increase the success probability significantly. If r1 +r2 has to be minimized, one can show that this corresponds to r1 = r2 = √n. This explains why attacks based on this property have also been called "square root" attacks. For a more detailed discussion of the probabilities the reader is referred to appendix B. The first attack based on this property was proposed by G. Yuval [5]. He showed how to attack a digital signature scheme of Rabin [6], more in particular he shows that it is easier to construct collisions for a one-way function than to find a preimage of a given element in the range. A collision can be produced in the following way.

• The adversary generates r1 variations on a bogus message and r2 variations on a genuine message. This is very easy, even if r1 and r2 are large: it is sufficient to have log2(r1) respectively log2(r2) positions where one has two alternatives or synonyms. If r1 = r2 = r = √n the probability of the existence of a match will be 63%. Note that in case of a MAC the opponent is unable to generate the MAC of a message. He could however obtain these MAC's with a chosen plaintext attack. A second possibility is that he collects a large number of messages and corresponding MAC's and divides them in two categories, which corresponds to a known plaintext attack.

The search for a match does not require r2 operations: after sorting the data, which requires O(r log r) operations, comparison is easy.

An algorithmic improvement has been the collision search algorithm proposed by J.-J. Quisquater [7, 8]. It is based on Pollard's _-method for finding cycles [9] in periodic functions on a finite domain. It eliminates almost completely the storage requirements if the attacker is able to call the function (it does not work if a match has to be found in stored data). If a MAC is attacked this corresponds to an adaptive chosen text attack. The basic idea is that if a random mapping is iterated (the output is fed back to the input), it will arrive after a tail with length λ into a cycle with period μ. At the point where the tail enters the cycle (the point of contact), one has found two values x and x` such that f(x) = f(x`). A graphical representation of this process will correspond to the Greek letter ρ. The storage requirements can be reduced to a negligible quantity by only storing points with specific characteristics (distinguished points). The expected number of function evaluations is equal to $\rho = \lambda + \mu = \sqrt{\pi n/8} + \sqrt{\pi n/8} = \sqrt{\pi n/2}$ (for a proof, see [10]). This result is also applicable to several other attacks: it is possible to produce pseudo-collisions for a single iteration step or collisions with initial values chosen from a small set. Other extensions will be discussed in the rest of this section.

Feasibility An important problem is to decide which computations should be considered feasible for the time being and within 10 and 20 years from now. This discussion is partially based on [11]. In terms of computations, one can start from the following facts (mid 1992):

• a single PC or workstation is able to perform a function evaluation in about 25 μsec, which corresponds to 2^{40} function evaluations per year,

• a supercomputer like the Cray-3 or the Cray Y-MP C90 (both with 16 processors) is capable of performing 16 Gigaflops on 64-bit words [12, 13]. If one function evaluation takes 64 operations, this corresponds to 2^{52} function evaluations per year. Based on the observation that the speed of computers is multiplied by four every three years this means that 21 years from now (which seems a reasonable time scale for a number of applications) a single super computer will be able to perform 2^{66} function evaluations per year. It will require 4096 simple processors to perform the same number of operations. However, it can be expected that in the near future even inexpensive computers will have many processors built in, as increasing the number of processors is the most effective way to increase the computational power without excessive increase of the cost (economy of scale) [13]. Hence it is realistic to assume that this computing power will be available in any small size organization. It can be shown that many problems in cryptanalysis can be easily adapted to such a distributed environment [14]: it are probably the applications that will achieve the highest performance on massive parallel machines. These predictions can be extended to dedicated cryptanalytic hardware, if one accepts the assumption that hardware will remain about two orders of magnitude faster. This corresponds to a factor of $2^6 \ldots 2^7$. The disadvantage of dedicated hardware is the higher cost.

For memory requirements, the situation is more complex, as the size of the available memory depends on the access time [15]. Moreover the access time to memory decreases much slower than the cycle time of the processor, and this can be solved only partially by using cache memories. An efficient attack will balance the use of different types of memories such that the access times are comparable to the calculations that have to be done in between. An example of such an attack using one central hardware board that is connected to a large number of PC's with a few Megabytes of memory has been described in [16]. Predictions can be based on the observation that memory devices increase in capacity by a factor of four every three years. Today's supercomputers have a main memory of up to 32 Gigabytes, a disk capacity of 50−100 Gigabytes and a high-performance mass storage system of 200 Gigabytes [13]. For storage with still slower access, like tapes, capacity in the order of several Terabytes is currently available, and in the next decade this will become Pentabytes. The memory available in workstations is much smaller. Fast cache memories have currently a capacity between 256 and 512 Kbytes. For dynamic RAMs, 4 Mbit chips are currently in mass production, which means that main memory of 32 Megabytes is becoming the standard in workstations. For disk storage, 1 Gigabyte can be considered state of the art.

One can conclude that for attacks that require no storage, a size of 128 bits corresponding to 2^{64} operations is sufficient for the next 10 years, but it will be only marginally secure within 20 years. One can predict that a storage of about 64 Gigabytes with an acceptable access time will be available on a single workstation within 10 years. If one has 1024 machines of this type available, this amounts to 64 Terabytes. With this constraint, attacking a 64-bit hashcode requires only 2^{21} operations, but probably the access time to the memory would be dominant. For a 96-bit hashcode this amounts to 2^{54} operations corresponding to a few years on these machines, and to a few months if dedicated hardware is available for the computations. For a 128-bit hashcode this would require 2^{86} operations, which is probably not realistic for the next 20 years (in fact the storage capacity will be a factor 64 larger by then, which yields 2^{80} operations). It is clear that a hashcode of 160 bits offers a sufficient security level for 20 years or more.

Attacks dependent on the chaining

This class of attacks depends on some high level properties of the elementary function f.

Meet in the middle attack

This attack is a variation on the birthday attack, but instead of the hashcode, intermediate chaining variables are compared.

The attack enables an opponent to construct a message with a prespecified hashcode, which is not possible in case of a simple birthday attack. Hence it also applies to a OWHF. The opponent generates r_1 variations on the first part of a bogus message and r_2 variations on the last part. Starting from the initial value and going backwards from the hashcode, the probability for a matching intermediate variable is given by the same formula. The only restriction that applies to the meeting point is that it can not be the first or last value of the chaining variable. The probability to find a match as a function of r_1 and r_2 is described in appendix B. The cycle finding algorithm by J.-J. Quisquater can be extended to perform a meet in the middle attack with negligible storage [7, 11]. The attack can be thwarted by avoiding functions f that are invertible to the chaining variable H_{i-1} and to the message Xi (cf. section 2.4.2 and 2.4.3).

Constrained meet in the middle attack

This type of attack is based on the same principles as the meet in the middle attack, but it takes into account certain constraints that have to be imposed on the solution. Examples of restrictions are that the sum modulo 2 of all blocks should be constant, or that a block of the CBC encryption of the solution with a given initial value and key should take a prespecified value.

Generalized meet in the middle attack

This attack was extended [17, 18] to break the p-fold iterated schemes. In these schemes the message is repeated p times or p hash values are computed corresponding to p initial values. With the extended attack, breaking these schemes does not require $O(2^{\frac{np}{2}})$ but only $O(10^p \cdot 2^{\frac{n}{2}})$ operations. The size of the message in this construction is $2 \cdot 10^{p-1}$ blocks. Modest trade-offs between time, storage, size of the message and processing are possible.

Correcting block attack

This attack consists of substituting all blocks of the message except for some block X_j . This block is then calculated such that the hashcode takes a certain value, which makes it also suitable to attack a OWHF. It often applies to the last block and is then called a correcting last block attack, but it can also apply to the first block or to some blocks in the middle. The hash functions based on modular arithmetic are especially sensitive to this attack.

A correcting block attack can also be used to produce a collision. One starts with two arbitrary messages X and X` and appends one or more correcting blocks denoted with Y and Y`, such that the extended messages X‖Y and X`‖Y` have the same hashcode.

One can try to thwart a correcting block attack by adding redundancy to the message blocks, in such a way that it becomes computationally infeasible to find a correcting block with the necessary redundancy. The price paid for this solution is a degradation of the performance.

Fixed point attack

The idea of this attack is to look for a H_{i-1} and X_i such that $f(X_i,H_{i-1}) = H_{i-1}$. If the chaining variable is equal to H_{i-1}, it is possible to insert an arbitrary number of blocks equal to Xi without modifying the hashcode. Producing collisions or a second preimage with this attack is only possible if the chaining variable can be made equal to H_{i-1}: this is the case if IV can be chosen equal to a specific value, or if a large number of fixed points can be constructed (if e.g., one can find an X_i for every H_{i-1}). Of course this attack can be extended to fixed points that occur after a number of steps. This attack can be prevented easily: one can append a block count to the data or one can (for theoretical constructions) encode the data with a prefix-free code [19].

Key collisions

This type of attack can only be applied to hash functions based on block ciphers. If the chaining mode is poorly designed, attacks can be launched based on key collisions. A key collision is a pair of keys K_1, K_2 such that $E(K_1, P) = E(K_2, P)$ for a plaintext P. The number of collisions for a given plaintext can be obtained from theorem B.2. In the case of DES [8, 108], with a block length of 64 bits and a key size of 56 bits, the number of k-fold collisions for a given P is indicated in table 2.2. Key collisions can be constructed with an ordinary birthday attack, but J.-J. Quisquater has shown how the efficient cycle algorithms combined with the method of the distinguished points can produce a collision in about 233 operations and with negligible storage.

K	2	3	4	5	6	7
	47.0	37.4	27.4	17.1	6.5	-4.3

Table 2.2: Binary logarithm of the expected number of k-fold key collisions for a given plaintext in the case of DES.

An important observation is that doubling the number of operations yields a squaring of the number of different collisions.

The attack can be extended to the case of double encryption. In this case a key collision consists of two pairs of keys (K^1,K^2) and $(K`_1 ,K`_2)$ (with $K_i \neq K`_i$) such that
$E(K_2,E(K_1, P)) = E(K`_2 ,E(K`_1, P))$.

It is also possible to produce a single key pair such that $E(K_2,E(K_1, P)) = C$ for a given plaintext P and ciphertext C.

The collision search is feasible for any block cipher that behaves as a random mapping if the key size is significantly smaller than 128, but a good design of the hash function can make the collisions useless. There is however no easy way to guarantee this, and every scheme has to be verified for this attack.

Differential attacks

Differential cryptanalysis is based on the study of the relation between input and output differences and is applicable to both block ciphers and hash functions. The attack is statistical as one searches for input differences that are likely to cause a certain output difference. If one is looking for collisions this output difference should be equal to zero. In case of hash functions based on block ciphers, the situation is slightly different: depending on the mode one requires that the output difference is zero or that the output difference is equal to the input difference (in case of feed forward of the plaintext). It applies only to iterated ciphers that satisfy particular conditions, the so-called Markov ciphers. It turns out that most known iterated ciphers are of this nature. For well designed block ciphers this attack will find the key based on a large number of plaintexts with a chosen difference, or an even larger number of known plaintexts. One can remark that this class of attacks is in fact more natural in case of an MDC, where there is no secret information. A chosen message attack is the standard way of attacking an MDC, and in this case all calculations can be performed off-line and in parallel.

Analytical weaknesses

Some schemes allow manipulations like insertion, deletion, permutation and substitutions of blocks. A large number of attacks have been based on a blocking of the diffusion of the data input: this means that changes have no effect or can be cancelled out easily in a next stage. This type of attacks has been successful for dedicated hash functions and for hash functions based on modular arithmetic.

Attacks dependent on an interaction with the signature scheme

In some cases it is possible that even if the hash function is a CRHF, it is possible to break the signature scheme. This attack is then the consequence of a dangerous interaction between both schemes. In the known examples of such an interaction both the hash function and the signature scheme have some multiplicative structure.

Attacks dependent on the underlying block cipher

Certain weaknesses of a block cipher are not significant when it is used to protect the privacy, but can have dramatic consequences if the cipher is used in one of the special modes for hashing. These weaknesses can be exploited to insert special messages or to carry out well chosen manipulations without changing the hashcode.

Complementation property

One of the first properties that was known of DES was the symmetry under complementation:

$$\forall P,K : C = \mathrm{DES}(K,P) \iff \overline{C} = \mathrm{DES}(\overline{K},\overline{P})$$

It can reduce an exhaustive key search by a factor 2 but it also allows to construct trivial collisions.

Weak keys

Another well known property of DES is the existence of 4 weak keys. For these keys, encryption equals decryption, or DES is an involution. These keys are also called palindromic keys. This means that $E(K,E(K, P)) = P$, $\forall$ P. There exist also 6 pairs of semi-weak keys, for which $E(K_2,E(K_1, P)) = P$, $\forall$ P. This property can be exploited in certain hash functions to construct fixed points after two iterations steps. Compared to DES, LOKI had more weak keys, but LOKI91 has the same number of weak and semi-weak keys.

It was remarked by B. den Boer that a similar property holds for PES and IDEA: for the all zero key the cipher is an involution.

Fixed points

Fixed points of a block cipher are plaintexts that are mapped to themselves for a certain key. As a secure block cipher is a random permutation, it will probably have fixed points (for every key there is a probability of $1-e^{-1}$ that there is at least a single fixed point). However, it should be hard to find these. Under some conditions it is easy to produce fixed points:

• For DES, this can be done based on a property of the weak keys: for every weak key Kp, there exist 2^{32} values of P that can

be easily found for which DES(K_p, P) = P. A similar property holds for the anti-palindromic keys: these are 4 semi-weak keys for which there exist 232 values of P that can be easily found for which DES(K_{ap}, P) =P.

• The block cipher LOKI has 256 simple fixed points where the key is of the special form $ggggggggghhhhhhhh_x$, and the plaintext is equal to $iiiiiiiiiiiiiii_x$, with i = g ⊕ h [22]. Here g, h and i are 4-bit numbers in hexadecimal notation. For every weak key there exist 2^{32} fixed points.

High level attacks

Even if the above attacks would not be feasible, special care has to be taken to avoid replay of messages and construction of valid messages by combining others. For authentication of transmitted messages, attacks at this level can be thwarted by adding a nonce, this is a quantity that is never transmitted twice in a given context, and through the use of sound cryptographic protocols. It is essential to authenticate the integrity of the nonces together with the message.

Timestamps: the date and time of the moment at which the message is sent. If the resolution of the time is sufficiently high, it will provide a unique identifier of the message. For a resolution of one second, 5 to 6 bytes are sufficient. The two main problems are the cost of maintaining reasonably well synchronized clocks at both ends of the communication line and of delays in communication channels.

Serial numbers: a unique number is assigned to every message. A size of 4 bytes should be sufficient for most applications, depending on the lifetime of the key. If every user keeps a different sequence number for every user he communicates with, the serial numbers should be consecutive, and the deletion of a message can be detected. If every user has only one sequence number for all his communications, one has to check that the serial numbers form an increasing sequence. This is only possible if every user stores the highest sequence number of every communication. This system does not allow for checking for deleted messages. A serial number is less expensive than a time stamp, but the timeliness of the information can not be checked. This should be no problem for applications like electronic mail.

Random numbers: a sufficiently long random number is added to the message. To thwart a birthday attack on the number, it has to be larger than the square of the maximal number of messages that will be sent with a key. For most applications this means a size of about 8 bytes. A random number is not very useful if all previous random numbers have to be stored to detect a replay. However, if the random number is used in the next step of the protocol, it can offer an adequate protection.

In the case of stored information, a ‘replay’ attack becomes a ‘restore’ attack. The serial numbers have to be replaced by version numbers, and a separate file is necessary that contains a single date and time stamp and for every file the current version number. If rearrangements of units that are protected by a different MAC is a problem, the address in the memory space can be protected together with the stored information.

Conclusion

In this paper several types of cryptographic hash functions have been defined, with the emphasis on the system based or practical approach. It has been shown how cryptographic hash functions provide an efficient way to protect integrity and to speed up digital signatures. A general model has been introduced that allows for a compact description of iterated hash functions and attacks.

References

1) X. Lai and J.L. Massey, “Hash functions based on block ciphers,” Advances in Cryptology, Proc. Eurocrypt’92, LNCS 658, R.A. Rueppel, Ed., Springer-Verlag, 1993, pp. 55–70.

2) S. Goldwasser, S. Micali, and R.L. Rivest, “A digital signature scheme secure against adaptive chosen-message attacks,” SIAM Journal on Computing, Vol. 17, No. 2, 1988, pp. 281-308.

3) M. Nuttin, “Cryptanalyse van het DES Algoritme in de CFB Mode (Cryptanalysis of the CFB Mode of the DES – in Dutch),” ESAT Laboratorium, Katholieke Universiteit Leuven, Thesis grad. eng.,1992.

4) W. Feller, “An Introduction to Probability Theory and Its Applications, Vol. 1,”Wiley, 1968.

5) G. Yuval, “How to swindle Rabin,” Cryptologia, Vol. 3, 1979, pp. 187–189.

6) M.O. Rabin, “Digitalized signatures,” in “Foundations of Secure Computation,”R. Lipton and R. DeMillo, Eds., Academic Press, New York, 1978, pp. 155-166.

7) J.-J. Quisquater and J.-P. Delescaille, “How easy is collision search? Application to DES,” Advances in Cryptology, Proc. Eurocrypt’89, LNCS 434, J.-J. Quisquater and J. Vandewalle, Eds., Springer-Verlag, 1990, pp. 429–434.

8) J.-J. Quisquater and J.-P. Delescaille, “How easy is collision search. New results and applications to DES,” Advances in Cryptology, Proc. Crypto’89, LNCS 435, G. Brassard, Ed., Springer-Verlag, 1990, pp. 408–413.

9) R. Sedgewick, T.G. Szymanski, and A.C. Yao, “The complexity of finding cycles in periodic functions,” SIAM Journal Comput., Vol. 11, No. 2, 1982, pp. 376–390.

10) P. Flajolet and A.M. Odlyzko, “Random mapping statistics,” Advances in Cryptology, Proc. Eurocrypt’89, LNCS 434, J.-J. Quisquater and J. Vandewalle, Eds., Springer-Verlag, 1990, pp. 329–354.

11) J.-J. Quisquater, “Collisions,” E.I.S.S. Workshop on Cryptographic Hash Functions, Oberwolfach (D), March 25-27, 1992.

12) G. Zorpette, “Large computers,” IEEE Spectrum, Vol. 29, No. 1, 1992, pp. 33-35.

13) G. Zorpette (Ed.), “Special issue on supercomputing,” IEEE Spectrum, Vol. 29, No. 9, 1992, pp. 26–76.

14) J.-J. Quisquater and Y. Desmedt, “Chinese lotto as an exhaustive code-breaking machine,” Computer, November 1991, pp. 14-22.

15) W.E. Proebster, “The evolution of data memory and storage: an overview,” in “Computer Systems and Software Engineering,” P. Dewilde and J. Vandewalle, Eds., Kluwer Academic Publishers, 1992, pp. 1–23.

16) R.R. Jueneman, “A high speed Manipulation Detection Code,” Advances in Cryptology, Proc. Crypto’86, LNCS 263, A.M. Odlyzko, Ed., Springer-Verlag, 1987, pp. 327–347.

17) D. Coppersmith, “Another birthday attack,” Advances in Cryptology, Proc. Crypto’85, LNCS 218, H.C. Williams, Ed., Springer-Verlag, 1985, pp. 14–17.

18) M. Girault, R. Cohen, and M. Campana, “A generalized birthday attack,”Advances in Cryptology, Proc. Eurocrypt’88, LNCS 330, C.G. G¨unther, Ed., Springer-Verlag, 1988, pp. 129–156.

19) I.B. Damg°ard, “A design principle for hash functions,” Advances in Cryptology, Proc. Crypto’89, LNCS 435, G. Brassard, Ed., Springer-Verlag, 1990, pp. 416–427.