PROPERTIES AND APPROACH OF CRYPTOGRAPHIC HASH ALGORITHMS

T.LALITHA
Senior Lecturer in MCA,
Sona College of Technology, Salem
Tamilnadu, India
E-Mail: lalithasrilekha@rediffmail.com

DR.R.UMARANI
Reader in Computer Science,
Sri Saradha College for Women, Salem,
Tamilnadu,India
E-Mail: umainweb@gmail.com

Abstract

The importance of hash functions for protecting the authenticity of information is demonstrated. Applications include integrity protection, conventional message authentication and digital signatures. An overview is given of the study of basic building blocks of cryptographic hash functions leads to the study of the cryptographic properties of Boolean functions and the information theoretic approach to authentication is described. An overview is given of the complexity theoretic definitions and constructions .New criteria are defined and functions satisfying new and existing criteria are studied.

1. Introduction

Authentication is the protection of the communicating parties against attacks of a third party. However, a different threat emerges when the two communicating parties are mutually distrustful and try to perform repudiation. This means that sender or receiver will try to modify a message and/or deny have sending or receiving a particular message. In paper documents, protection against this type of attack is offered by a handwritten signature. It is clear that in case of electronic messages, a simple name at the end of the message offers no protection at all. This is analogous to the fact that a photocopy of a document with a manual signature has no value, as one can always produce a bogus document with cut and paste operations. A typical example of this fraud is the electronic communication of orders to a stockbroker.

2. Three approaches to the authentication problem

In present day cryptography, three approaches can be identified to solve most problems comprising information secrecy and information authenticity. These approaches differ in the assumptions about the capabilities of an opponent, in the definition of a cryptanalytic success, and in the notion of security. This taxonomy is based on the taxonomy that was developed for stream ciphers by R. Rueppel and deviates from the taxonomy for authentication developed by G. Simmons.

A first method is based on information theory, and it offers unconditional security, i.e., security independent of the computing power of an adversary. The complexity theoretic approach starts from an abstract model for computation, and assumes that the opponent has limited computing power. The system based approach tries to pro- duce practical solutions, and the security estimates are based on the best algorithm known to break the system and on realistic estimates of the necessary computing power or dedicated hardware to carry out the algorithm. In [310] the second and third approach are lumped together as computationally secure, and in [287] a fourth approach is considered, in which the opponent has to solve a problem with a large size (namely examining a huge publicly accessible random string); it can be considered as both computationally secure and information theoretically secure.

It should be noted that the information theoretic approach is impractical for most applications because of the size of the secret key. Sometimes the complexity theoretic approach allows for efficient constructions, but in a practical scheme the dimensions of the scheme are fixed and the proof of security has only a limited value.

2.1 Information theoretic approach

This approach results in a characterization of unconditionally secure solutions, which implies that the security of the system is independent of the computing power of the opponent. E.g., in case of privacy protection, it has been shown by C. Shannon that unconditional privacy protection requires that the entropy of the key is lower bounded by the entropy of the plaintext. It should be remarked that both unconditional privacy and unconditional authenticity are only probabilistic: even if the system is optimal with respect to some definition, the opponent has always a non-zero probability to cheat. However, this probability can be made exponentially small. The advantage of this approach lies in the unconditional security. Like in the case of the Vernam scheme, the price paid for this is that these schemes are rather impractical.

The cryptographer considers a game-theoretic model, in which the opponent observes l messages and subsequently tries to impersonate or substitute messages. The cryptographer will encipher his messages under control of a secret key. Because the goal of the cryptographer is now the protection of the authenticity (or the combination of secrecy and authenticity), the transformation will be called an authentication code. The information theoretic study of authentication has now been reduced to the design of authentication codes, that are in some sense dual to error correcting codes . In both cases redundant information is introduced: in case of error correcting codes the purpose of this redundancy is to allow the receiver to reconstruct the actual message from the received codeword, and to facilitate this the most likely alterations are in some metric close to the original codeword; in case of authentication codes, the goal of the redundancy is to allow the receiver to detect substitutions or impersonations by an active eavesdropper, and this is obtained by spreading altered or substituted messages as uniformly as possible.

2.2 Complexity theoretic approach

The approach taken here is to define at first a model of computation, like a Turing machine or a Boolean circuit. All computations in this model are parameterized by a security parameter, and only algorithms or circuits that require asymptotically polynomial time and space in terms of the size of the input are considered feasible. Examples of cryptographic primitives are: secure message sending, cryptographically secure pseudo-random generation, general zero-knowledge interactive proofs, Universal One- Way Hash Functions (UOWHF), Collision Resistant Hash Functions (CRHF), and digital signatures. It will be shown that the latter three are relevant for information authentication. Examples of general assumptions to which these primitives can be reduced are the existence of one-way functions, injections, or permutations, and the existence of trapdoor one-way permutations. A third aspect is the efficiency of the reduction, i.e., the number of executions of the basic function to achieve a cryptographic primitive, and the number of interactions between the players in the protocol.

Several lines of research have been followed. A first goal is to reduce cryptographic primitives to weaker assumptions, with as final goal to prove that the reduction is best possible. A different approach is to produce statements about the impossibility of basing a cryptographic primitive on a certain assumption [150]. One can also try to improve the efficiency of a reduction, possibly at the cost of a stronger assumption. If someone wants to build a concrete implementation, he will have to choose a particular one-way function, permutation, etc. The properties of a particular problem that is believed to be hard can be used to increase the efficiency of the solutions. Examples of problems that have been intensively used are the factoring of a product of two large primes, the discrete logarithm problem modulo a prime and modulo a composite that is the product of two large primes, and the quadratic residuosity problem.
The complexity theoretic approach has several advantages:

1. It results in provable secure systems, based on a number of assumptions.

2. The constructions of such proofs requires formal definitions of the cryptographic Primitives and of the security of a cryptographic primitive.
3. The assumptions on which the security of the systems is based are also defined formally.

The disadvantage is that the complexity theoretic approach has only a limited impact on practical implementations, due to limitations that are inherently present in the models.

1. In complexity theory, a number of operations that is *polynomial* in the size of the input is considered to be feasible, while a super polynomial or exponential number of operations in the size of the input is Infeasible. In an asymptotic setting, abstraction is made from both constant factors and the degrees of the polynomials. This implies that this approach gives no information on the security of concrete instances (a Practical problem has a finite size). Secondly, the scheme might be impractical because the number of An operation to be carried out is polynomial in the size of the input but impractically large.
2. The complexity theoretic approach yields only results on the worst case or average case problems in a general class of problems. However, cryptographers studying the security of a scheme are more interested in the subset of problems that is easy.
3. Complexity usually deals with single isolated instances of a problem. A cryptanalyst often has a large collection of statistically related problems to solve.

2.3 System based or practical approach

In this approach schemes with fixed dimensions are designed and studied, paying special attention to the efficiency of software and hardware implementations. The objective of this approach is to make sure that breaking a cryptosystem is a difficult problem for the cryptanalyst.

By trial and error procedures, several cryptanalytic principles have emerged, and it is the goal of the designer to avoid attacks based on these principles. Typical examples are statistical attacks and meet in the middle attacks.

The second aspect is to design building blocks with provable properties. These building blocks are not only useful for cryptographic hash functions, but also for the design of block ciphers and stream ciphers. Typical examples are statistical criteria, diffusion and confusion, correlation, and non-linearity criteria. Thirdly, the assembly of basic building blocks to design a cryptographic hash functions can be based on theorems. Results of this type are often formulated and proven in a complexity theoretic setting, but can easily be adopted for a more practical definition of "hardness" that is useful in a system based approach. A typical example is the theorem stating that a collision-resistant hash function can always be constructed if a collision-resistant function exists, where the first reference uses a complexity theoretic approach and the second a more practical definition.

3. Hash Algorithms

The four secure hash algorithms such as SHA-1, SHA-256, SHA-384, and SHA- 512. All four of the algorithms are iterative, one-way hash functions that can process a message to produce a condensed representation called a message digest. These algorithms enable the determination of a message's integrity: any change to the message wills, with a very high probability, result in a different message digests. This property is useful in the generation and verification of digital signatures and message authentication codes, and in the generation of random numbers (bits).

Each algorithm can be described in two stages: preprocessing and hash computation. Preprocessing involves padding a message, parsing the padded message into m-bit blocks, and setting initialization values to be used in the hash computation. The hash computation generates a message schedule from the padded message and uses that schedule, along with functions, constants, and word operations to iteratively generate a series of hash values. The final hash value generated by the hash computation is used to determine the message digest.

The four algorithms differ most significantly in the number of bits of security that are provided for the data being hashed - this is directly related to the message digest length. When a secure hash algorithm is used in conjunction with another algorithm, there may be requirements specified elsewhere that require the use of a secure hash algorithm with a certain number of bits of security. For example, if a message is being signed with a digital signature algorithm that provides 128 bits of security, then that signature algorithm may require the use of a secure hash algorithm that also provides 128 bits of security (e.g., SHA-256). Additionally, the four algorithms differ in terms of the size of the blocks and words of data that are used during hashing.

4. Notation and Conventions

4.1 *Bit Strings and Integers*

The following terminology related to bit strings and integers will be used.
1. A hex digit is an element of the set {0, 1,…, 9, a,…, f}. A hex digit is the representation of a 4-bit string. For example, the hex digit "7" represents the 4-bit string "0111", and the hex digit "a" represents the 4-bit string "1010".
2. A word is a w-bit string that may be represented as a sequence of hex digits. To convert a word to hex digits, each 4-bit string is converted to its hex digit equivalent, as described in (1) above.

For example, the 32-bit string

1010 0001 0000 0011 1111 1110 00 can be expressed as "a103fe23",

and the 64-bit string 1010 0001 0000 0011 1111 1110 0010 0011 0011 0010 1110 1111 0011 0000
0001 1010 can be expressed as "a103fe2332ef301a".

Throughout this specification, the "big-endian" convention is used when expressing both 32- and 64-bit words, so that within each word, the most significant bit is stored in the left-most bit position.
3. An integer may be represented as a word or pair of words. A word representation of the message length, l , in bits, is required for the padding techniques .
An integer between 0 and 2^{32}-1 inclusive may be represented as a 32-bit word. The least significant four bits of the integer are represented by the right8-most hex digit of the word representation.

For example, the integer 291=2 + 2^5 + 2^1 +2^0=256+32+2+1 is represented by the hex word 00000123.
The same holds true for an integer between 0 and 2^{64}-1 inclusive, which may be represented as a 64-bit word.

If Z is an integer, $0 \le Z < 2^{64}$, then $Z=2^{32}X+Y$, where $0 \le X < 2^{32}$ and $0 \le Y < 2^{32}$. Since X and Y can be represented as 32-bit words x and y, respectively, the integer Z can be represented as the pair of words (x, y). This property is used for SHA-1 and SHA-256. If Z is an integer, $0 \le Z < 2^{128}$, then $Z=2^{64}X+Y$, where $0 \le X < 2^{64}$ and $0 \le Y < 2^{64}$.

Since X and Y can be represented as 64-bit words x and y, respectively, the integer Z can be represented as the pair of words (x, y). This property is used for SHA-384 and SHA-512.

4. For the secure hash algorithms, the size of the message block - m bits - depends on the algorithm.
a) For SHA-1 and SHA-256, each message block has 512 bits, which are represented as a sequence of sixteen 32-bit words.

b) For SHA-384 and SHA-512, each message block has 1024 bits, which are represented as a sequence of sixteen 64-bit words.

4.2 *Operations on Words*

The following operations are applied to w-bit words in all four secure hash algorithms. SHA-1 and SHA-256 operate on 32-bit words (w=32), and SHA-384 and SHA-512 operate on 64-bit Words (w=64).

1. Bitwise logical word operations: : $\wedge$, $\vee$, $\oplus$, and $\leftarrow$

2. Addition modulo 2^w.

The operation x + y is defined as follows. The words x and y represent integers X and Y, where $0 \leq X < 2^w$ and $0 \leq Y < 2^w$. For positive integers U and V, let U mod V be the remainder upon dividing U by V. Compute
$Z = (X + Y) \bmod 2^w$.
Then $0 \leq Z < 2^w$. Convert the integer Z to a word, z, and define z=x + y.

3. The right shift operation SHR $^n(x)$, where x is a w-bit word and n is an integer with $0 \leq n < w$, is defined by
SHR $^n(x)$=x >> n.
This operation is used in the SHA-256, SHA-384, and SHA-512 algorithms.

4. The rotate right (circular right shift) operation ROTR $^n(x)$, where x is a w-bit word and n is an integer with $0 \leq n < w$, is defined by ROTR$^n(x) = (x >> n) \vee (x << w - n)$.
Thus, ROTR $^n(x)$ is equivalent to a circular shift (rotation) of x by n positions to the right. This operation is used by the SHA-256, SHA-384, and SHA-512 algorithms.

5. The rotate left (circular left shift) operation, ROTL $^n(x)$, where x is a w-bit word and n is an integer with $0 \leq n < w$, is defined by ROTL $^n(x) = (x << n) \vee (x >> w - n)$.
Thus, ROTL $^n(x)$ is equivalent to a circular shift (rotation) of x by n positions to the left.
This operation is used only in the SHA-1 algorithm. Note that in Ref. [180-1] this operation was referred to as "S^n(X)"; however, the notation has been modified for clarity and consistency with the notation used for operations in the other secure hash algorithms.

6. Note the following equivalence relationships, where w is fixed in each relationship:

$$\text{ROTL}^n(x) \approx \text{ROTR}^{w-n}(x)$$
$$\text{ROTR}^n(x) \approx \text{ROTL}^{w-n}(x).$$

5. Preprocessing

Preprocessing shall take place before hash computation begins. This preprocessing consists of three steps: padding the message, *M*, parsing the padded message into message blocks and setting the initial hash value, $H^{(0)}$.

5.1 *Padding the Message*

The message, *M*, shall be padded before hash computation begins. The purpose of this padding is to ensure that the padded message is a multiple of 512 or 1024 bits, depending on the algorithm.

5.1.1 SHA-1 and SHA-256

Suppose that the length of the message, *M*, is l bits. Append the bit "1" to the end of the message, followed by *k* zero bits, where *k* is the smallest, non-negative solution to the equation $l + 1 + k \equiv 448 \bmod 512$. Then append the 64-bit block that is equal to the number l expressed using a binary representation. For example, the (8-bit ASCII) message "**abc**" has length $8 \times 3 = 24$, so the message is padded with a one bit, then $448 - (24 + 1) = 423$ zero bits, and then the message length, to become the 512-bit padded message .

$$\underbrace{01100001}_{\text{“a”}}\quad \underbrace{01100010}_{\text{“b”}}\quad \underbrace{01100011}_{\text{“c”}}\quad 1\quad \overbrace{00\ldots00}^{423}\quad \underbrace{\overbrace{00\ldots011000}^{64}}_{\mathbf{L=24}}\ .$$

The length of the padded message should now be a multiple of 512 bits.

5.1.2 ***SHA-384 and SHA-512***

Suppose the length of the message M, in bits, is l bits. Append the bit "1" to the end of the message, followed by k zero bits, where k is the smallest non-negative solution to the equation $l+1+k \equiv 896 \bmod 1024$. Then append the 128-bit block that is equal to the number l expressed using a binary representation. For example, the (8-bit ASCII) message "**abc**" has length $8\times 3 = 24$, so the message is padded with a one bit, then $896-(24+1)=871$ zero bits, and then the message length, to become the 1024-bit padded message

$$\underbrace{\mathbf{01100001}}_{\text{"a"}}\quad \underbrace{\mathbf{01100010}}_{\text{"b"}}\quad \underbrace{\mathbf{01100011}}_{\text{"c"}}\quad \mathbf{1}\quad \overbrace{\mathbf{00\ldots00}}^{871}\quad \overbrace{\mathbf{00\ldots011000}}^{128}\ .$$

The length of the padded message should now be a multiple of 1024 bits.

5.3 ***Setting the Initial Hash Value ($H^{(0)}$)***

Before hash computation begins for each of the secure hash algorithms, the initial hash value, $H^{(0)}$, must be set. The size and number of words in $H^{(0)}$ depends on the message digest size.

5.3.1 *SHA-1*

For SHA-1, the initial hash value, $H^{(0)}$, shall consist of the following five 32-bit words, in hex:

$H^{(0)}$= 67452301

H_1(0)= efcdab89

H_2(0)= 98badcfe

H_3(0)= 10325476

H_4(0)=c3d2e1f0.

5.3.2 SHA-256

For SHA-256, the initial hash value, $H^{(0)}$, shall consist of the following eight 32-bit words, in hex:

H_0(0) = 6a09e667

H_1(0) = bb67ae85

H_2(0) = 3c6ef372

H_3(0) = a54ff53a

H_4(0)= 510e527f

H_5(0) = 9b05688c

H_6(0) = 1f83d9ab

H_7(0) = 5be0cd19.

These words were obtained by taking the first thirty-two bits of the fractional parts of the square roots of the first eight prime numbers.

5.3.3 *SHA-384*

For SHA-384, the initial hash value, $H^{(0)}$, shall consist of the following eight 64-bit words, in hex:

H_0(0) = cbbb9d5dc1059ed8

H_1(0) = 629a292a367cd507

H_2(0) = 9159015a3070dd17

H_3(0= 152fecd8f70e5939

H_4(0) = 67332667ffc00b31

H_5(0) = 8eb44a8768581511

H_6(0) = db0c2e0d64f98fa7

H_7(0) = 47b5481dbefa4fa4.

These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the ninth through sixteenth prime numbers.

5.3.4 *SHA-512*

For SHA-512, the initial hash value, $H^{(0)}$, shall consist of the following eight 64-bit words, in hex:

$H_0^{(0)}$= 6a09e667f3bcc908

$H_1(0)$ = bb67ae8584caa73b

$H_2(0)$ = 3c6ef372fe94f82b

$H_3(0)$ = a54ff53a5f1d36f1

$H_4(0)$ = 510e527fade682d1

$H_5(0)$ = 9b05688c2b3e6c1f

$H_6(0)$ = 1f83d9abfb41bd6b

$H_7(0)$ = 5be0cd19137e2179.

These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eight prime numbers.

Conclusion

The use of cryptographic hash functions like MD5 or SHA for message authentication has become a standard approach in many Internet applications and protocols. Though very easy to implement, these mechanisms are usually based on ad hoc techniques that lack a sound security analysis. We present new constructions of message authentication schemes based on a cryptographic hash function. Our schemes are proven to be secure as long as the underlying hash function has some reasonable cryptographic strengths. Moreover we show, in a quantitative way, that the schemes retain almost all the security of the underlying hash function. In addition our schemes are efficient and practical. Their performance is essentially that of the underlying hash function. Moreover they use the hash function (or its compression function) as a black box, so that widely available library code or hardware can be used to implement them in a simple way, and replacability of the underlying hash function is easily supported.

References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
[2] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.
[3] Bagnulo, M. and J. Arkko, "Cryptographically Generated Addresses (CGA) Extension Field Format", October 2006.
[4] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "Secure Neighbor Discovery (SEND)", RFC 3971, March 2005