CRYPTO TOPICS AND APPLICATIONS I

Jennifer Seberry, Chris Charnes, Josef Pieprzyk, and Rei Safavi-Naini
Centre for Computer Security Research, University of Wollongong

1 INTRODUCTION

In this chapter we discuss four related areas of cryptology, namely: Authentication, Hashing, Message Authentication Codes (MACs), and Digital Signatures. These topics represent currently active and growing research topics in cryptology. Due to space limitations, we concentrate only on the essential aspects of each topic. The bibliography is intended to supplement our survey. We have included sufficiently many items to provide the interested reader with an overall view of the current state of knowledge in the above areas.

Authentication deals with the problem of providing assurance to a receiver that a communicated message originates from a particular transmitter, and that the received message has the same content as the transmitted message. A typical authentication scenario occurs in computer networks, where the identity of two communicating entities is established by means of authentication.

Hashing is concerned with the problem of providing a relatively short *fingerprint* of a much longer message or electronic document. A *hashing function* must satisfy (at least) the critical requirement that the fingerprints of two distinct messages are distinct. Hashing functions have numerous applications in cryptology. They are often taken as primitives in constructing other cryptographic functions.

Message Authentication Codes (MACs) are symmetric key primitives that provide message integrity against active spoofing by appending a cryptographic checksum to a message which is verifiable only by the intended recipient of the message. Message authentication is one of the most important ways of ensuring the integrity of information which is transferred by electronic means.

Digital signatures provide electronic equivalents of handwritten signatures. They preserve the essential features of handwritten signatures, and can be used to sign electronic documents. Digital signatures can potentially be used in legal contexts.

2 AUTHENTICATION

One of the main goals of a cryptographic system is to provide authentication, which simply means providing assurance about the content and origin of communicated messages.

Historically, cryptography began with secret writing and this remained the main area of development until very recently. With the rapid progress in data communication, the need for providing message integrity and authenticity has escalated to the extent that currently authentication is seen as the more urgent goal of cryptographic systems.

Traditionally, it was assumed that a secrecy system provides authentication by the virtue of the secret key being only known by the intended communicants; this would prevent an enemy from constructing a fraudulent message. Simmons [66] argued that the two goals of cryptography are independent. He shows that a system that provides perfect secrecy might not provide any protection against authentication threats. Similarly, a system can provide perfect authentication without concealing messages.

In the rest of this chapter, we use the term *communication system* to encompass message transmission as well as storage. The system consists of a *transmitter* who wants to send a message, a *receiver* who is the intended recipient of the message, and an *enemy* who attempts to construct a fraudulent message with the aim of getting it accepted by the receiver unwittingly. In some cases, there is a fourth party, called the *arbiter*, whose basic role is to provide protection against cheating by the transmitter and/or the receiver. The communication is assumed to take place over a public channel, and hence the communicated messages can be seen by all the principals. An authentication threat is an attempt by an enemy in the system to modify a communicated message or inject a fraudulent message into the channel. In a secrecy system the attacker is passive, while in an authentication system the enemy is active and not only observes the communicated messages and gathers information such as plaintext and ciphertext, but also actively interacts with the system to achieve its goal. This view of the system clearly explains Simmons' motivation for basing authentication systems on game theory.

The most important criteria that can be used to classify authentication systems are:

- the relation between authenticity and secrecy;
- the framework for the security analysis.

The first criterion divides authentication systems into those that provide *authentication with* and *without secrecy.* The second criterion divides systems into systems with *unconditional security,* systems with *computational security* and systems with *provable security. Unconditional security* implies that the enemy has unlimited computational power, while in *computational security* the enemy's resources are limited and the security relies on the required computation exceeding the enemy's computational power. A system with *provable security* is in fact a subclass of computationally secure systems and its compromise is equivalent to a solution of a known difficult problem.

These two classifications are orthogonal and produce four subclasses. Below we review the basic concepts of authentication theory, some known bounds and constructions in unconditional security, and then consider computational security.

2.1 Unconditional security

A basic model introduced by Simmons [66] has remained the mainstay of most of the theoretical research on authentication systems. The model has the same structure as described in the previous section but excludes the arbiter. To provide protection against an enemy, the transmitter and receiver use an *authentication code (A-code).* An A-code is a collection $\mathcal{E}$ of mappings called *encoding rules* (also called *keys*) ¿from a set $\mathcal{S}$ of *source states* (also called *transmitter states*) into the set $\mathcal{M}$ of cryptogram (also called *codewords*). For A-codes without secrecy, also called *cartesian A-codes*, a codeword uniquely determines a source state. That is, the set of codewords

is partitioned into subsets each corresponding to a distinct source state. In a *systematic cartesian A-code*, $\mathcal{M} = \mathcal{S} \times \mathcal{T}$ where $\mathcal{T}$ is a set of *authentication tags* and each codeword is of the form $s.t, s \in \mathcal{S}, t \in \mathcal{T}$ where '.' denotes concatenation. Let the cardinality of the set $\mathcal{S}$ of source states be denoted as k; that is, $|\mathcal{S}| = k$. Let $E = |\mathcal{E}|$ and $M = |\mathcal{M}|$. The encoding process adds key dependent redundancy to the message, so $k < M$. A key (or encoding rule) e determines a subset $\mathcal{M}_e \subset \mathcal{M}$ of codewords that are authentic under e.

The *incidence matrix* A of an A-code is a binary matrix of size $E \times M$ whose rows are labeled by encoding rules and columns by codewords, such that $A(e, m) = 1$ if m is a valid codeword under e, and $A(e, m) = 0$, otherwise.

An *authentication matrix* B of a cartesian A-code is a matrix of size $E \times k$ whose rows are labeled by the encoding rules and columns by the source states, and $B(e, s) = t$ if t is the tag for the source state s under the encoding rule e.

To use the system, the transmitter and receiver must secretly share an encoding rule. The enemy does not know the encoding rule and uses an *impersonation* attack, in which it only uses its knowledge of the system, or a *substitution* attack in which it waits to see a transmitted codeword, and then constructs a fraudulent codeword. The security of the system is measured in terms of the enemy's chance of success with the chosen attack. The enemy's chance of success in an impersonation attack is denoted by P_0, and in a substitution attack by P_1. The best chance an enemy has in succeeding in either of the above attacks is called the *probability of deception*, P_d.

An attack is said to be *spoofing of order* ℓ if the enemy has seen ℓ communicated codewords under a single key. The enemy's chance of success in this case is denoted by P_ℓ.

The chance of success can be defined using two different approaches. The first approach corresponds to an average case analysis of the system and can be described as the enemy's payoff in the *game theory model*. It has been used by a number of authors, including MacWilliams, Gilbert and Sloane [32], Fak [28], and Simmons [66]. The second is to consider the worst case scenario. This approach is based on the relation between A-codes and error correcting codes (also called E-codes).

Using the game theory model, P_j is the value of a zero-sum game between communicants and the enemy. For impersonation

$$P_0 = \max_{m \in \mathcal{M}} (\text{payoff}(m)),$$

and for substitution

$$P_1 = \sum_{j=1}^{E} \sum_{m \in \mathcal{M}} P(m) \max_{m'} \text{payoff}(m, m'),$$

where $P(m)$ is the probability of a codeword m occurring in the channel, and payoff(m, m') is the enemy's payoff (best chance of success) when it substitutes an intercepted codeword m with a fraudulent one, m'.

2.2 Bounds on the performance of the A-code

The first types of bounds relate the main parameters of an A-code, that is, E, M, k, and hence are usually called *combinatorial bounds*. The most important combinatorial bound for A-codes with secrecy is

$$P_i \geq \frac{k - i}{M - i}, \ i = 1, 2, \ldots$$

and for A-codes without secrecy is

$$P_i \geq k/M, \ i = 1, 2, \ldots.$$

An A-code that satisfies these bounds with equality, that is, with $P_i = \dfrac{k-i}{M-i}$ for A-codes with secrecy and $P_i = k/M$ for cartesian A-codes, is said to *provide perfect protection for spoofing of order* i. The enemy's best strategy in spoofing of order i for such an A-code is to randomly select one of the remaining codewords.

A-codes that provide perfect protection for all orders of spoofing up to r are said to be *r-fold secure*. These codes can be characterized using combinatorial structures such as orthogonal arrays and t-designs.

An orthogonal array $OA_\lambda(t, k, v)$ is an array with λv^t rows, each row of size k, from the elements of set X of v symbols, such that in any t columns of the array every t-tuple of elements of X occurs in exactly λ rows. Usually t is referred to as *the strength* of the OA.

Example 1 *The following table gives a* $OA_2(2, 5, 2)$ *on the set* $\{0, 1\}$*:*

0	0	0	0	0
1	1	0	0	0
0	0	0	1	1
1	1	0	1	1
1	0	1	0	1
0	1	1	0	1
1	0	1	1	0
0	1	1	1	0

A t-(v, k, λ) *design* is a collection of b subsets, each of size k, of a set, X, of size v where every distinct subset of size t occurs exactly λ times.

The *incidence matrix* of a t-(v, k, λ) is a binary matrix, $A = (a_{ij})$, of size $b \times v$ such that $a_{ij} = 1$ if element j is in block i and 0 otherwise.

Example 2 *The following table gives a* 3-$(8, 4, 1)$ *design on the set* $\{0, 1, 2, 3, 4, 5, 6, 7\}$*:*

7	0	1	3
7	1	2	4
7	2	3	5
7	3	4	6
7	4	5	0
7	5	6	1
7	6	0	2
2	4	5	6
3	5	6	0
4	6	0	1
5	0	1	2
6	1	2	3
0	2	3	4
1	3	4	5

with incidence matrix:

$$\begin{array}{cccccccc}
1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0
\end{array}$$

The main theorems relating A-codes with r-fold security and combinatorial structures are due to a number of authors, including Stinson [69], and Tombak and Safavi-Naini [77]. The following are the most general forms of these theorems.

Theorem 1 *([77]) Let the source be r-fold uniform. Then an A-code provides r-fold security against spoofing if and only if the incidence matrix of the code is the incidence matrix of a $(r+1)$- (M, k, λ) design.*

In the above theorem, an r-fold uniform source is a source for which every string of r distinct outputs from the source has probability $\frac{1}{k(k-1)\cdots(k-r+1)}$.

Theorem 2 *Let $P_0 = P_1 = P_2 = \cdots = P_r = k/M$. Then the authentication matrix is a $OA(r+1, k, \ell)$ where $\ell = M/k$.*

The so-called *information theoretic bounds* characterize the enemy's chance of success using uncertainty measures. The first such bound for cartesian A-codes, derived by MacWilliams, Gilbert and Sloane [32], is

$$P_1 \geq 2^{-\frac{H(M)}{2}}$$

where $H(M)$ is the entropy of the codeword space. The first general bound on P_0, due to Simmons [66], is

$$P_0 \geq 2^{-(H(E)-H(E|M))}$$

where $H(E)$ is the entropy of the key space and $H(E|M)$ is the conditional entropy of the key when a codeword is intercepted. Write $I(E;M)$ for the mutual information of E and M. Then,

$$P_0 \geq 2^{I(E;M)}.$$

The above bound relates the enemy's best chance of success to the mutual information between the cryptogram space and the key space. A general form of this bound, proved independently by Rosenbaum [63] and Pei [47], is

$$P_\ell \geq 2^{I(E;M^\ell)},$$

where $I(E; M^\ell)$ is the mutual information between a string of ℓ codewords, and where m^ℓ is the key.

Similar bounds for A-codes without secrecy are proved by MacWilliams *et al* [32] and by Walker [79].

A general bound on the probability of deception, P_{d_r}, derived by Rosenbaum [63], is

$$P_{d_r} \geq 2^{-\frac{H(E)}{r+1}} .$$

2.3 Other types of attack

Tombak and Safavi-Naini [76] consider other types of attacks, similar to those for secrecy systems. In a *plaintext attack* against A-codes with secrecy, the enemy not only knows the codeword but also knows the corresponding plaintext. In *chosen content attack* the enemy wants to succeed with a codeword that has a prescribed plaintext. It is shown that by applying some transformation on the A-code it is possible to provides immunity against the above attacks.

A-codes with secrecy are generally more difficult to analyze than cartesian A-codes. Moreover, the verification process for the former is not as efficient. In the case of cartesian A-codes, verification of a received codeword, $s.t$, amounts to recalculating the tag using the secret key and the source state s to obtain t' and comparing it with the received tag t. For an authentic codeword we have $t = t'$. In the case of A-codes with secrecy, when a codeword m is received, the receiver must try all authentic codewords using his secret key, otherwise there must be an inverse algorithm that allows the receiver to find and verify the source state. The former process is costly and the latter does not exist for a general A-code. For practical reasons, the majority of research has concentrated on cartesian A-codes.

2.4 Efficiency

Authentication systems require secure transmission of key information prior to the communication and hence, similar to secrecy systems, it is desirable to have a small number of key bits.

Rees and Stinson [60] prove that: for any (M, k, E) A-code which is 1-fold secure, $E \geq M$. For A-codes with secrecy that provide one-fold security, Stinson [69] shows that:

$$E \geq \frac{M^2 - M}{k^2 - k} .$$

Under similar conditions for cartesian A-codes, Stinson [69] shows that:

$$E \geq k(\ell - 1) + 1$$

where $\ell = k/M$.

For A-codes with r-fold security, Stinson [71] shows that:

$$E \geq \frac{M(M-1)\cdots(M-r)}{k(k-1)\cdots(k-r)} .$$

An A-code *provides perfect authenticity of order* r if $P_{d_r} = k/M$. In such codes the probability of success of the spoofer does not improve with the interception of extra codewords.

The following bound is established by Tombak and Safavi-Naini [75] for codes of the above type:

$$E \geq \frac{M^{r+1}}{k^{r+1}} .$$

Their bound shows that the provision of perfect authenticity requires $\log M - \log k$ extra key bits on average for every added order of spoofing. Hence using the same key for authentication of more than one message is expensive.

A second measure of efficiency, often used for systematic cartesian A-codes, is the size of the tag space for a fixed size of source space and probability of success in substitution. Stinson [72] shows that, for perfect protection against substitution, the size of key space grows linearly with the size of the source. Johansson, Kabatianskii and Smeets [37] show that: if $P_1 > P_0$, A-codes with an exponential (in E) number of source states can be obtained.

2.5 A-codes and E-codes

An error correcting code provides protection against random channel error The study of error correcting codes was motivated by Shannon's channel capacity theorem and has been a very active research area since the early 1950s. Error correcting codes add redundancy to a message in such a way that a codeword corrupted by the channel noise can be detected and/or corrected. The main difference between an A-code and an E-code is that in the former redundancy depends on a secret key while in the latter it only depends on the message being coded. There exists a duality between authentication codes and error correcting codes. In the words of Simmons [66], " .. one (coding theory) is concerned with clustering the most likely alterations as closely about the original code as possible and the other (authentication theory) with spreading the optimal (to the opponent) alterations as uniformly as possible over $\mathcal{M}$."

The relation between E-codes and A-codes is explored in the work of Johansson *et al* [37], who show that it is possible to construct E-codes from A-codes and vice-versa. Their work uses a worst case analysis approach in analyzing the security of A-codes. That is, in the case of substitution attack, they consider the *best* chance of success an enemy has when it intercepts all possible codewords. This contrasts with the information theoretic (or game theory) approach in which the average success probability of the enemy over all possible intercepted codewords is calculated.

The work of Johansson *et al* is especially useful as it allows the well developed body of bounds and asymptotic results from the theory of error correcting codes to be employed in the context of authentication codes, to derive upper and lower bounds on the size of the source for A-codes with given E, T, and P_1.

2.6 Authentication with arbiter

In the basic model of authentication discussed above, the enemy is an outsider and we assume that the transmitter and the receiver are trustworthy. Moreover, because the key is shared by the transmitter and the receiver, the two principals are cryptographically indistinguishable. In an attempt to model authentication systems in which the transmitter and the receiver are distinguished and to remove assumptions about the trustworthiness of the two, Simmons [66] introduced a fourth principal called the *arbiter*. The transmitter and receiver have different keys and the arbiter has access to all or part of the key information. The system has a *key distribution phase* during which keys satisfying certain conditions are chosen. After that there is a *transmission phase* during which the transmitter uses its key to produce a codeword and finally a *dispute phase* during which disputes are resolved with the aid of the arbiter. The arbiter in Simmons' model is active during the transmission phase and is assumed to be trustworthy. Yung and Desmedt [83] remove this assumption and consider a model in which the arbiter is only trusted to resolve disputes. Johansson [35] and Kurosawa [39] derive lower bounds on the probability of deception in such codes. Johansson [36] and Taylor [73] propose constructions.

2.7 Shared generation of authenticators

Many applications require the power to generate an authentic message and/or to verify the authenticity of a message to be be distributed among a number of principals. An example of such a situation is multiple signatures in a bank account or court room. Desmedt and Frankel [25] introduced systems with *shared generation of authenticators (SGA-systems)*, which have been studied in recent papers by: Safavi-Naini [64], Gehrmann, van Dijk and Smeets, [29], and Safavi-Naini and Martin [65]. In such systems there is a group P of transmitters created with an structure Γ that determines authorized subsets of P. Each principal has a secret key which is used to generate a partial tag. The system has two phases. In the key distribution phase, a *trusted authority* generates keys for the transmitters and the receivers and securely delivers the keys to them. In the communication phase, the trusted authority is not active. When an authorized group of transmitters wants to construct an authentic codeword, using its key, each group member generates a partial tag for the source state s which needs to be authenticated, and sends it to a *combiner*. The combiner is a fixed algorithm with no secret input. It combines its inputs to produce a tag, t, to be appended to s. The receiver is able to authenticate this codeword using its secret key. Safavi-Naini and Martin [65] give a general construction for SGA-systems by combining A-codes and secret sharing schemes. Gehrmann *et al* [29] propose an efficient construction for SGA-systems, based on maximum rank distance separable codes.

2.8 Multiple authentication

As noted before, in the theory of A-codes the possible attacks by the enemy are limited to impersonation and substitution. This means that the security of the system is only for one message communication; after that the key must be changed. To extend protection over more than one message transmission, there exist a number of alternatives. The most obvious ones use A-codes that provide perfect protection against spoofing of order ℓ. However, little is known about the construction of such codes and it is preferable to use A-codes that provide protection against substitution for more that one message transmission. Vanroose, Smeets and Wan [82] suggest *key strategies* in which the communicants change their key after each transmitted codeword,

using some pre-specified strategy. In this case the key information shared by the communicants is the *sequence of keys* to be used for consecutive transmission slots. The resulting bounds on the probability of deception generalize the bounds given by the following authors: Pei [47], Rosenbaum [63], and Walker [79].

Another successful approach proposed by Wegman and Carter [80] uses a special class of hash functions together with a one time pad of random numbers. This construction is discussed in more detail in Section 5.

3 Computationally secure systems

The study of computationally secure A-systems is relatively informal, cf. Simmons [67]. The basic framework is similar to unconditionally secure systems. A simple computationally secure A-code can be obtained by considering $\mathcal{S} = GF(2^{40})$ and $\mathcal{M} = GF(2^{64})$. We use $\mathcal{E}$ to be the the collection of DES [51] encryption functions and so $E = 2^{56}$. To construct the codeword corresponding to a source state s, using the key k, we append 24 zeros to s and then use DES with key k to encrypt $s.\mathbf{0}_{24}$, where $\mathbf{0}_{24}$ is the string of 24 zeros.

It is easy to see that the above scheme is an A-code with secrecy. It allows the receiver to easily verify the authenticity of a received codeword by decrypting a received codeword and checking the existence of the string of zeros. If an enemy wants to impersonate the transmitter its chance of success is 2^{-56}, which is the probability of guessing the correct key. For a substitution attack, the enemy waits to see a transmitted codeword. Then it uses all the keys to decrypt the codeword and once a decryption of the right form (ending in 24 zeros) is obtained, a possible key is found. In general there is more than one key with this property. On the average there are $2^{56} \times 2^{40}/2^{64} = 2^{32}$ pairs (s, k) that produce the same cryptogram and hence the chance of guessing correctly is 2^{-32}. A better strategy for the enemy is to randomly choose a cryptogram and send it to the receiver. In this case his chance of success is 2^{-24}, which is better than the previous case.

The security of computationally secure A-systems weakens very quickly as the enemy intercepts more cryptograms. Trying all possible keys on ℓ received cryptograms enables the enemy to uniquely identify the key, in which case the enemy's chance of success is one.

Computationally secure A-systems without secrecy are obtained by appending an *authenticator* to the message which is verifiable by the intended receiver. The authenticator can be produced by a *symmetric key algorithm* or an *asymmetric key algorithm*. The former is the subject of the section on Message Authentication Codes (MAC), while the latter is discussed in the section on digital signatures.

4 HASHING

In many cryptographic applications, it is necessary to produce a relatively short *fingerprint* of a much longer message or electronic document. The fingerprint is also called a digest of the message. Ideally, a hash function should produce a unique digest of a fixed length for a message of an arbitrary length. Obviously, this is impossible as any hash function is, in fact, a

many-to-one mapping. The properties required for secure hashing can be summarized as follows:

- hashing should be a many-to-one function producing a digest which is a complex function of all bits of the message;
- a hash function should behave as a random function which creates a digest for a given message by randomly choosing an element from the whole digest space;
- for any pair of messages, it should be computationally difficult to find a collision; i.e. distinct messages with the same digest; and
- a hash function should be one-way; i.e. it should be easy to compute the digest of a given message but difficult to determine the message corresponding to a given digest.

The main requirement of secure hashing is that it should be *collision-free* in the sense that finding two colliding messages is computationally intractable. This requirement must hold for long as well as short messages.

4.1 Strong and weak hash functions

Hash functions can be broadly classified into two classes: *strong one-way hash functions* (also called *collision-free hash functions*) and *weak one-way hash functions* (also known as *universal one-way hash functions*). A strong one-way hash function is a function h satisfying the following conditions:

1. h can be applied to any message or document M of any size;
2. h produces a fixed size digest;
3. Given h and M, it is easy to compute the digest $h(M)$; and
4. Given h, it is computationally infeasible to find two distinct messages M_1, M_2 which collide, i.e. $h(M_1) = h(M_2)$.

On the other hand, a weak one-way hash function is a function that satisfies conditions (1), (2), (3) and the following:

4'. Given h and a randomly chosen message M, it is computationally intractable to find another message M' which collides with M, i.e. $h(M) = h(M')$.

Strong one-way hash functions are easier to use since there is no precondition on the selection of the messages. On the other hand, for weak one-way hash functions, there is no guarantee that a non-random selection of two messages is collision-free. This means that the space of easily found colliding messages must be small. Otherwise, a random selection of two messages would produce a collision with a non-negligible probability.

4.2 Theoretic Constructions

Naor and Yung [44] introduced the concept of a universal one-way hash function (UOWHF) and suggested a construction based on a one-way permutation. In their construction, they employ the notion of a *universal family of functions with collision accessibility property* [80]. The above functions are defined as follows.

Definition 1 *Suppose $G = \{g \mid A \rightarrow B\}$ is a set of functions. G is* strongly universal$_r$ *if given any r distinct elements $a_1, \ldots, a_r \in A$, and any r elements $b_1, \ldots, b_r \in B$, there are $|G|/|B|^2$ functions which take a_1 to b_1, a_2 to b_2 etc. ($|G|$ and $|B|$ denote the cardinality of sets G and B, respectively.)*

Definition 2 *A strongly universal$_r$ family of functions G has the* collision accessibility property *if it is possible to generate in polynomial time a function $g \in G$ that satisfies the equations:*

$$g(a_i) = b_i, \ 1 \leq i \leq r.$$

Naor and Yung construct a family of UOWHF's by concatenating any one-way permutation with a family of strongly universal$_2$ hash functions having the collision accessibility property. In this construction, the one-way permutation provides the one-wayness of the UOWHF, and the strongly universal$_2$ family of hash functions provides the mapping to the small length output. When a function is chosen randomly and uniformly ¿from the family, the output is distributed randomly and uniformly over the output space.

Zheng, Matsumoto and Imai [84] define a scheme based on the composition of a pairwise independent uniformizer and a strongly universal hash function with a quasi-injection one-way function.

De Santis and Yung [24] construct hash functions from one-way functions with an almost-known preimage size. In other words, if an element of the domain is given, then with a polynomial uncertainty an estimate of the size of the preimage set is easily computable. A *regular* function is an example of such a function. (In a regular function, each image of an n-bit input has the same number of preimages of length n.)

Rompel [62] constructs a UOWHF from any one-way function. His construction is rather elaborate. Briefly, his idea is to transform any one-way function into a UOWHF through a sequence of complicated procedures. First, the one-way function is transformed into another one-way function such that for most elements of the domain except for a fraction, it is easy to find a collision. ¿From this, another one-way function is constructed such that for most of the elements it is hard to find a collision. Subsequently, a length increasing one-way function is constructed for which it is hard to find collisions almost everywhere. Finally, this is turned into a UOWHF which compresses the input in a way that makes it difficult to find a collision (cf. Pieprzyk and Sadeghiyan [49]).

4.3 Hashing based on block ciphers

To minimize the design effort for cryptographically secure hash functions, the designers of hash functions tend to base their schemes on existing encryption algorithms. For example, sequential hashing can be obtained by dividing a given message into blocks and applying an encryption

algorithm on the message blocks. The message block length must be the same as the block length of the encryption algorithm. If the message length is not a multiple of the block length, then the last block is usually padded with some redundant bits. To provide a randomizing element, an initial public vector is normally used. The proof of the security of such schemes relies on the collision freeness of the underlying encryption algorithm.

In the following, let E denote an arbitrary encryption algorithm. Let $E(K, M)$ denote the encryption of message M with key K using E; let IV denote the initial vector.

Rabin [55] shows that any private-key cryptosystem can be used for hashing. Rabin's scheme is the following. First the message is divided into blocks $M_1, M_2, \ldots$ of the same size as the block length of of the encryption algorithm. In the case of DES, the message is divided into 64-bit blocks. To hash a message $M = (M_1, M_2, \ldots, M_t)$, the following computations are performed:

$$\begin{aligned} H_0 &= IV \\ H_i &= E(M_i, H_{i-1}) \qquad\qquad i = 1, 2, \ldots, t \\ H(M) &= H_t \end{aligned}$$

where M_i is a message block, H_i are intermediate results of hashing, and $H(M)$ is the digest. Although Rabin's scheme is simple and elegant, it is susceptible to the so-called *birthday attack* when the size of the hash value is 64 bits.

Winternitz [81] proposes a scheme for the construction of a one-way hash function ¿from any block cipher. In any good block cipher, given an input and an output, it should be difficult to determine the key, but from the the key and the output it should be easy to determine the input. The scheme uses an operation E^* defined by:

$$E^*(K \parallel M) = E(K, M) \oplus M.$$

Based on the above scheme, Davies [23] proposes the following hashing algorithm:

$$\begin{aligned} H_0 &= IV \\ H_i &= E(M_i, H_{i-1}) \oplus H_{i-1} \qquad\qquad i = 1, 2, \ldots, t \\ H(M) &= H_t. \end{aligned}$$

If $E(K, M)$ is DES, then it may be vulnerable to attacks based on weak keys or a key-collision search. The meet-in-the-middle attack is thwarted because $E(K, M)$ is a one-way function.

Merkle [43, 42] proposed hashing schemes based on Winternitz's construction. These schemes use DES to produce digests of size ≈ 128 bits. Their construction follows a general method for constructing hash algorithms, called the *meta method*. This is the same as the serial method of Damgård [22]. The description of the method is as follows. The message is first divided into blocks of 106 bits. Each 106-bit block M_i of data is concatenated with the 128-bit block H_{i-1}. The concatenation $X_i = M_i \parallel H_{i-1}$ contains 234 bits. Each block X_i is further divided into halves, X_{i1} and X_{i2}.

$$\begin{aligned} H_0 &= IV \\ X_i &= H_{i-1} \parallel M_i \\ H_i &= E^*(00 \parallel \text{first 59 bits of}\{E^*(100 \parallel X_{1i})\} \parallel \\ &\quad \text{first 59 bits of}\{E^*(101 \parallel X_{2i})\}) \parallel \\ &\quad E^*(01 \parallel \text{first 59 bits of}\{E^*(110 \parallel X_{1i})\} \parallel \end{aligned}$$

$$\text{first 59 bits of}\{E^*(111 \parallel X_{2i})\})$$
$$H(M) = H_t.$$

In this scheme E^* is defined as in Winternitz's construction. The strings 00, 01, 100, 101, 110, and 111 above are used to prevent the manipulation of weak keys.

4.4 Hashing functions based on intractable problems

Hashing functions can also be based on one-way functions such as: exponentiation, squaring, knapsack (cf. Pieprzyk and Sadeghiyan [49]), and discrete logarithm. More recently, a group-theoretic construction using the SL_2 groups has been proposed by Tillich and Zémor [74].

A scheme based on RSA exponentiation as the underlying one-way function is defined by:

$$\begin{aligned} H_0 &= IV \\ H_i &= (H_{i-1} \oplus M_i)^e \bmod N \qquad i = 1, 2, \ldots, t \\ H(M) &= H_t \end{aligned}$$

where the modulus N and the exponent e are public. A correcting block attack can be used to compromise the scheme by appending or inserting a carefully selected last block message to achieve a desired hash value. To immunize the scheme against this attack, it is necessary to add redundancy to the message so that the last message block cannot be manipulated (cf. Davies and Price [23]). To ensure the security of RSA, N should be at least 512 bits in length, making the implementation of the above scheme very slow.

To improve the performance of the above scheme, the public exponent can be made small. For example, squaring can be used:

$$H_i = (H_{i-1} \oplus M_i)^2 \bmod N.$$

It is suggested that 64 bits of every message block be set to 0, to avoid a correcting block attack.

An algorithm for hashing based on squaring is proposed in Appendix D of the X.509 recommendations of the CCITT standards on secure message handling. The proposal stipulates that 256 bits of redundancy be distributed over every 256-bit message block by interleaving every four bits of the message with 1111, so that the total number of bits in each block becomes 512. The exponentiation algorithm, with exponent two, is then run on the modified message in CBC mode (cf. Pieprzyk and Sadeghiyan [49]). In this scheme, the four most significant bits of every byte in each block are set to 1. Coppersmith [20] shows how to construct colliding messages in this scheme.

Damgård [22] describes a scheme based on squaring, which maps a block of n bits into a block of m bits. The scheme is defined by:

$$\begin{aligned} H_0 &= IV \\ H_i &= \text{extract}(00111111 \parallel H_{i-1} \parallel M_i)^2 \bmod N \\ H(M) &= H_t. \end{aligned}$$

In the above scheme, the role of *extract* is to extract m bits ¿from the result of the squaring function. To obtain a secure scheme, m should be sufficiently large so as to thwart the birthday attack; this attack will be explained later. Moreover, *extract* should select bits for which finding

colliding inputs is difficult. One way to do this is to extract m bits uniformly. However, for practical reasons, it is better to bind them together in bytes. Another possibility is to extract every fourth byte. Daemen, Govaerts and Vanderwalle [21] show that this scheme can be broken.

Impagliazzo and Naor [34] propose a hashing function based on the knapsack problem. The description of the scheme is as follows. Choose at random numbers $a_1, \ldots, a_n$ in the interval $0, \ldots, N$, where n indicates the length of the message in bits, and $N = 2^\ell - 1$ where $\ell < n$. A binary message $M = M_1, M_2, \ldots, M_n$ is hashed as

$$H(M) = \sum_{i=1}^{n} a_i M_i \bmod 2^\ell.$$

Impagliazzo and Naor do not give any concrete parameters for the above scheme, but they have shown that it is theoretically sound.

Gibson [31] constructs hash functions whose security is conditional upon the difficulty of factoring certain numbers. The hash function is defined by:

$$f(x) = a^x \pmod{n},$$

where $n = pq$, p and q are large primes, and a is a primitive element of the ring Z_n. In Gibson's hash function n has to sufficiently large to ensure the difficulty of factoring. This constraint makes the hash function considerably slower than the MD4 algorithm.

Tillich and Zémor [74] propose a hashing scheme where the message digests are given by two-dimensional matrices with entries in the binary Galois fields $GF(2^n)$ for $130 \leq n \leq 170$. The hashing functions are parameterized by the irreducible polynomials of degree n, $P_n(X)$, over $GF(2)$; their choice is left to the user. Their scheme has several provably secure properties: detection of local modification of text; and resistance to the birthday attack as well as a few other attacks. Hashing is fast as digests are produced by matrix multiplication in $GF(2^n)$, which can be parallelized.

Messages (encoded as a binary strings) $x_1 x_2 \ldots$ of arbitrary length are mapped to products of a selected pair of *generators* $\{A, B\}$ of the group $SL(2, 2^n)$, as follows:

$$x_i = \begin{cases} A & \text{if } x_i = 0 \\ B & \text{if } x_i = 1. \end{cases}$$

The resulting product belongs to the (infinite) group $SL(2, GF(2)[X])$, where $GF(2)[X]$ is the ring of all polynomials over $GF(2)$. The product is then reduced modulo an irreducible polynomial of degree n (Euclidean algorithm), mapping it to an element of $SL(2, 2^n)$. The four n-bit entries of the reduced matrix give the $(3n + 1)$-bit message digest of $x_1 x_2 \ldots$.

Charnes and Pieprzyk [15] show that irreducible polynomials which produce collisions for the SL_2 hash functions can be found. Other weaknesses of this scheme are explored in a later paper available from them [17]. The weak parameters are characterized; they can be computed with the algorithms given there.

Geiselmann [30] describes an algorithm to produce potential collisions for the SL_2 hashing scheme, independent of the choice of the irreducible polynomials. The complexity of his algorithm is that of the discrete logarithm problem in $GF(2^n)$ or $GF(2^{2n})$. However, no collisions in the specified range of the hash function have been found using this algorithm. Some pairs of rather long colliding strings are given by Geiselmann for a toy example of $GF(2^{21})$.

4.5 Hashing algorithms

Rivest [58] proposes a hashing algorithm called MD4. It is a software oriented scheme which is especially designed to be fast on 32-bit machines. The algorithm produces a 128-bit output, so it is not computationally feasible to produce two messages having the same hash value. The scheme provides diffusion and confusion using three boolean functions. The MD5 hashing algorithm is a strengthened version of MD4 [57]. MD4 has been broken by Dobbertin [26].

HAVAL stands for a one-way hashing algorithm with a variable length of output. It was designed at the University of Wollongong by Zheng, Pieprzyk and Seberry [85]. It compresses a message of an arbitrary length into a digest of either 128, 160, 192, 224 or 256 bits. The security level can be adjusted by selecting 3, 4 or 5 passes. The structure of HAVAL is based on MD4 and MD5. Unlike MD4 and MD5 whose basic operations are done using functions of three boolean variables, HAVAL employs five boolean functions of seven variables (each function serves a single pass). All functions used in HAVAL are highly nonlinear, 0-1 balanced, linearly inequivalent, mutually output-uncorrelated and satisfy the Strict Avalanche Criterion (SAC). No attack on HAVAL has been reported so far.

Charnes and Pieprzyk [14] proposed a modified version of HAVAL based on five boolean functions of five variables. The resulting hashing algorithm is faster than the five pass, seven variable version of the original HAVAL algorithm. They use the same cryptographic criteria which are used to select the boolean functions in the original scheme. Unlike the seven variable case, the choice of the boolean functions is fairly restricted in the modified setting. Using the shortest algebraic normal form of the boolean functions as one of the criteria (to maximize the speed of processing), it is shown that the boolean functions used are optimal. No attacks have been reported for the five variable version.

4.6 Attacks

The best method to evaluate a hashing scheme is to see what attacks an adversary can perform to find collisions. A good hashing algorithm produces a fixed length number which depends on all the bits of the message. It is generally assumed that the adversary knows the hashing algorithm. In a conservative approach, it is assumed that the adversary can perform an adaptive chosen message attack, where it may choose messages, ask for their digests, and try to compute colliding messages. There are several methods for using such pairs to attack a hashing scheme and to calculate colliding messages. Some methods are quite general and can be applied against any hashing scheme; for example, the so-called *birthday attack*. Other methods are applicable only to specific hashing schemes. Some attacks can be used against a wide range of hash functions. For example, the so-called *meet-in-the-middle attack* is applicable to any scheme that uses some sort of block chaining in its structure. As another example, the so-called *correcting block attack* is applicable mainly to hash functions based on modular arithmetic.

Birthday Attack

The idea behind this attack originates from a famous problem from probability theory, called the *birthday paradox*. The paradox can be stated as follows. What is the minimum number of pupils in a classroom so the probability that at least two pupils have the same birthday is greater than 0.5? The answer to this question is 23, which is much smaller than the value suggested by intuition. The justification for the paradox is as follows. Suppose that the pupils are entering the classroom one at a time. The probability that the birthday of the first pupil falls on a

specific day of the year is equal to $\frac{1}{365}$. The probability that the birthday of the second pupil is not the same as the first one is equal to $1 - \frac{1}{365}$. If the birthdays of the first two pupils are different, the probability that the birthday of the third pupil is different from the first one and the second one is equal to $1 - \frac{2}{365}$. Consequently, the probability that t students have different birthdays is equal to $(1 - \frac{1}{365})(1 - \frac{2}{365}) \ldots (1 - \frac{t-1}{365})$, and the probability that at least two of them have the same birthday is

$$P = 1 - (1 - \frac{1}{365})(1 - \frac{2}{365}) \ldots (1 - \frac{t-1}{365}).$$

It can be easily computed that for $t \geq 23$, this probability is greater than 0.5.

The birthday paradox can be employed for attacking hash functions. Suppose that the number of bits of the hash value is n. An adversary generates r_1 variations of a bogus message and r_2 variations of a genuine message. The probability of finding a bogus message and a genuine message that hash to the same digest is

$$P \approx 1 - e^{-\frac{r_1 r_2}{2^n}}$$

where $r_2 \gg 1$ (see Ohta and Koyama [46]). When $r_1 = r_2 = 2^{\frac{n}{2}}$, the above probability is ≈ 0.63. Therefore any hashing algorithm which produces digests of length around 64 bits is insecure, since the time complexity function for the birthday attack is $\approx 2^{32}$. It is usually recommended that the hash value should be around 128 bits to thwart the birthday attack.

This method of attack does not take advantage of the structural properties of the hash scheme or its algebraic weaknesses. It applies to any hash scheme. In addition, it is assumed that the hash scheme assigns to a message a value which is chosen with a uniform probability among all the possible hash values. Note that if the structure is weak or has certain algebraic properties, the digests do not have a uniform probability distribution. In such cases it may be possible to find colliding messages with a better probability and fewer message-digest pairs.

Meet-in-the-middle attack

This is a variation of the birthday attack, but instead of comparing the digests, the intermediate results in the chain are compared. The attack can be made against schemes which employ some sort of block chaining in their structure. In contrast to birthday attack, a meet-in-the-middle attack enables an attacker to construct a bogus message with a desired digest. In this attack the message is divided into two parts. The attacker generates r_1 variations on the first part of a bogus message. He starts from the initial value and goes forward to the intermediate stage. He also generates r_2 variations on the second part of the bogus message. He starts from the desired false digest and goes backwards to the intermediate stage. The probability of a match in the intermediate stage is the same as the probability of success in the birthday attack.

Correcting-block attack

In a correcting block attack, a bogus message is concatenated with a block to produce a corrected digest of the desired value. This attack is often applied to the last block and is called *correcting last block* attack, although it can be applied to other blocks as well. Hash functions based on modular arithmetic are especially sensitive to the correcting last block attack (cf. Preneel [50]). The introduction of redundancy into the message in these schemes makes finding a correcting block with the necessary redundancy difficult. However, it makes the scheme less efficient. The difficulty of finding a correcting block depends on the nature of the redundancy introduced. For

example, Coppersmith [20] shows that the redundancy built into the CCITT hashing scheme based on modular squaring results in an insecure scheme.

Biham and Shamir [11] have developed a method for attacking block ciphers, known as *differential cryptanalysis*. This is a general method for attacking cryptographic algorithms, including hashing schemes. For example, Berson [8] has applied differential cryptanalysis to MD5.

5 MAC

Message authentication codes provide message integrity and are one of the most important security primitives in current distributed information systems. A *Message Authentication Code* (MAC) is a symmetric key cryptographic primitive that consists of two algorithms. A MAC *generation algorithm*, $G = \{G_k : k = 1, \ldots, N\}$ takes an arbitrary message, s, from a given collection $\mathcal{S}$ of messages and produces a *tag*, $t = G_k(s)$, which is appended to the message to produce an *authentic* message, $m = (s.t)$. A *MAC verification* algorithm, $V = \{V_k(.) : k = 1, \ldots, N\}$, takes authenticated messages of the form $(s.t)$, and produces a true or false value, depending on whether the message is authentic. The security of a MAC depends on the best chance that an active spoofer has to successfully substitute a received message $(s.G_k(s))$ for a fraudulent one, $m' = (s', t)$, so that $V_k(m')$ produces a true result. In MAC systems, the communicants share a secret key, and are therefore not distinguishable cryptographically.

The security of MACs can be studied from the point of view of unconditional or computational security.

Unconditionally secure MACs are equivalent to cartesian authentication codes. However, in MAC systems only multiple communications are of interest. In Section 2, A-codes that provide protection for multiple transmissions were discussed. In the next section, we present a construction for a MAC which has been the basis of all the recent MAC constructions and has a number of important properties. It is provably secure; the number of key bits required is asymptotically minimal; and it has a fast implementation.

Computationally secure MACs have arisen ¿from the needs of the banking community, cf. Preneel, Chaum, Fumy, Jansen, Landrock and Roelofsen [52]. They are also studied under other names, such as *keyed hash functions* and *keying hash functions*. In Section 5.3, we review the main properties and constructions of such MACs.

5.1 Unconditionally secure MACs

When the enemy has unlimited computational resources, attacks against MAC systems and the analysis of security are similar to that of cartesian A-codes. The enemy observes n codewords of the form $s_i.t_i$, $i = 1, \ldots, n$, in the channel and attempts to construct a fraudulent codeword $s.t$ which is accepted by the receiver. (This is the same as spoofing of order n in an A-code.) If the communicants want to limit the enemy's chance of success to p after n message transmissions, the number of authentication functions (number of keys) must be greater than a lower bound which depends on p. If the enemy's chance of success in spoofing of order i, $i = 1, \ldots, n$, is p_i, then at least $1/p_1 p_2 \cdots p_n$ keys are required, see Fak [28], Wegman and Carter [80] for a proof of this. For $p_i = p, i = 1, \ldots, n$, the required number of key bits is $-n \log_2 p$. That is, for every

message, $-\log_2 p$ key bits are required. This is the absolute minimum for the required number of key bits.

Perfect protection is obtained when the enemy's best strategy is a random choice of a tag and appending it to the message; this strategy succeeds with probability $p = 2^{-k}$, if the size of the tag is k bits. In this case the number of required key bits for every extra message is k.

Wegman and Carter [80] give a general construction for unconditionally secure MACs that can be used for providing protection for an arbitrary number of messages.

Their construction uses *universal classes of hash functions*. Traditionally, a hash function is used to achieve fast average performance over all inputs in varied applications, such as databases. By using a universal class of hash functions it is possible to achieve provable average performance without restricting the input distribution.

Let $h : A \rightarrow B$ be a hash function mapping the elements of a set A to a set B. A *strongly universal$_n$ class* of hash function is a class of hash functions with the property that for n distinct elements $a_1, \ldots, a_n$ of A and n distinct elements $b_1, \ldots, b_n$ of B, exactly $|H|/(b^n)$ functions map a_i to b_i, for $i = 1, \ldots, n$. Strongly universal$_n$ hash functions give perfect protection for multiple messages as follows. The transmitter and the receiver use a publicly known class of strongly universal$_n$ hash functions, and a shared secret key determines a particular member of the class that they will use for their communication. Stinson [70] shows that a class of strongly universal$_2$ that maps a set of a elements to a set of b elements is equivalent to an orthogonal array $OA_\lambda(2, a, b)$ with $\lambda = |H|/b^2$. Similar results can be proved for strongly universal$_n$ classes of hash functions. Because of this equivalence, universal$_n$ hash functions are not a practically attractive solution. In particular, this proposal is limited by the constraints of constructing orthogonal arrays with arbitrary parameters. For a survey of known results on orthogonal arrays, see Beth, Jungnickel and Lenz [10].

5.2 Wegman and Carter construction

Wegman and Carter show that, instead of strongly universal$_n$ one can always use a strongly universal$_2$ family of hash functions, together with a one time pad of random numbers. The system works as follows. Let B denote the set of tags consisting of the sequences of k bit strings. Let $\mathcal{H}$ denote a strongly universal$_2$ class of hash functions mapping $\mathcal{S}$ to B. Two communicants share a key that specifies a function $h \in \mathcal{H}$ together with a pad containing k-bit random numbers. The tag for the j^{th} message s_j is $h(s_j) \oplus r_j$, where r_j is the j^{th} number on the pad. It can be proved that this system limits the enemy's chance of success to 2^{-k} as long as the pad is random and not used repeatedly. The system requires $nk + K$ bits of key, where K is the number of bits required to specify an element of $\mathcal{H}$, n is the number of messages to be authenticated, and k is the size of the tags.

This construction has a number of remarkable properties. Firstly, for large n the key requirement for the system approaches the theoretical minimum of k bits per message. This is because for large n the number of key bits is effectively determined by nk. Secondly, the construction of MAC with provable security for multiple communications is effectively reduced to the construction of a better studied primitive, that is, strongly universal$_2$ class of hash functions. Finally, by replacing the one-time pad with a pseudorandom sequence generator, unconditional security is replaced by computational security.

Wegman and Carter's important observation is as follows. By not insisting on the minimum value for the probability of success in spoofing of order one, i.e. allowing $p_1 > 1/k$, it is possible to reduce the number of functions, and thus the required number of keys. This observation leads to the notion of almost strongly universal$_2$ class.

An *ϵ-almost universal$_2$* (or ϵ-AU$_2$) class of hash functions has the following property. For any pair $x, y \in A, x \neq y$, the number of hash functions h with $h(x) = h(y)$ is at most equal to ϵ. The *ϵ-almost strongly-universal$_2$* (or ϵ-ASU$_2$) hash functions have the added property that for any $x \in A, y \in B$ the number of functions with $h(x) = y$ is $|H|/|B|$. Using an ϵ-almost strongly-universal$_2$ class of functions in the Wegman and Carter construction results in MAC systems for which the probability of success for an intruder is ϵ. Such MACs are called *ϵ-otp-secure*, see Krawczyk [38].

Stinson [72] gives several methods for combining hash functions of class AU$_2$ and ASU$_2$. The following theorem shows that an ϵ-ASU$_2$ class can be constructed ¿from an ϵ-AU$_2$ class.

Theorem 3 *[72] Suppose H_1 is an ϵ_1-AU$_2$ class of hash functions ¿from A_1 to B_1, and suppose H_2 is an ϵ_2-ASU$_2$ class of hash functions from B_1 to B_2. Then there exists an ϵ-ASU$_2$ class H of hash functions from A_1 to B_2, where $\epsilon = \epsilon_1 + \epsilon_2$ and $|H| = |H_1| \times |H_2|$.*

This theorem further reduces the construction of MACs with provable security to the construction of ϵ-AU$_2$ class of hash functions.

Several constructions for ϵ-ASU$_2$ hash functions are given by Stinson [72]. Johansson *et al* [37] establish relationships between ASU$_2$ hash functions and error correcting codes. They use geometric error correcting codes to construct new classes of ϵ-ASU$_2$ hash function of smaller size. This reduces the key size.

Krawczyk [38] shows that in the Wegman-Carter construction, ϵ-ASU$_2$ hash functions can be replaced with a less demanding class of hash functions, called *ϵ-otp-secure*. The definition of this class differs from other classes of hash functions, in that it is directly related to MAC constructions and their security; in particular, to the Wegman-Carter construction.

Let $s \in \mathcal{S}$ denote a message that is to be authenticated by a k bit tag $h(s) \oplus r$, constructed by Wegman and Carter's method. An enemy succeeds in a spoofing attack if he can find $s' \neq s$, $t' = h(s') \oplus r$, assuming that he knows H but does not know h and r. A class H of hash functions is *ϵ-otp-secure* if for any message no adversary succeeds in the above attack scenario with probability greater than ϵ.

Theorem 4 *[38] A necessary and sufficient condition for a family H of hash functions to be ϵ-otp-secure is that*

$$\forall a_1 \neq a_2,\ c,\ \ Pr_h(h(a_1) \oplus h(a_2) = c) \leq \epsilon.$$

The need for high speed MACs has increased with the progress in high speed data communication. A successful approach to the construction of such MACs uses hash function families in which the message is hashed by multiplying it by a binary matrix. Because hashing is achieved with exclusive-or operations, it can be efficiently implemented in software. An obvious candidate for such a class of hash functions, originally proposed by Wegman and Carter [13, 80], is the set of linear transformations from A to B. It is shown that this forms an ϵ-AU$_2$ class of hash functions. However the size of the key – the number of entries in the matrix – is too large,

and too many operations are required for hashing. Later proposals by Krawczyk [38] and by Rogaway [61] are aimed at alleviating these problems, and have a fast software implementation. The former uses Topelitz matrices [38], while the latter uses binary matrices with only three ones per column. In both cases, the resulting family is ϵ-AU_2.

The design of a complete MAC usually involves a number of hash functions which are combined by methods, similar to those proposed by Stinson [72]. The role of some of the hash functions is to produce high compression (small b), while others produce the desired spread and uniformity (see Rogaway [61]).

Reducing the key size of the hash function is especially important in practical applications, because the one-time pad is replaced by the output of a pseudorandom generator with a short key (of the order of 128 bits). Hence it is desirable to have the key size of the hash function of similar order.

5.3 Computational security

In the computationally secure approach, protection is achieved because excessive computation is required for a successful forgery. Although a hash value can be used as a checksum to detect random changes in the data, a secret key must be used to provide protection against active tampering. Methods for constructing MACs from hash functions have traditionally followed one of the following approaches: the so-called *hash-then-encrypt* and *keying a hash function.*

Hash-then-encrypt
To construct a MAC for a message x with this method, the hash value of x is calculated and the result is encrypted using an encryption algorithm. This is similar to signature generation, where a public key algorithm is replaced by a private key encryption function.

There are a number of drawbacks to this method. First, the overall scheme is slow. This is because the two primitives used in the construction, i.e. the cryptographic hash functions and encryption functions, are designed for other purposes and have extra security properties which are not strictly required in the construction. Although this construction can produce a secure MAC, the speed of the MAC is bounded by the speed of its constituent algorithms. For example, cryptographic hash functions are designed to be one-way. It is not clear whether this is a required property in the hash-then-encrypt construction, where the output of the hash function is encrypted and one-wayness is effectively obtained through the difficulty of finding the plaintext from the ciphertext.

A serious shortcoming of this method is that existing export restrictions, which usually apply to encryption functions, are inherited by MACs constructed using this method.

Keying a hash function
In the second approach a secret key is incorporated into a hashing algorithm. This operation is sometimes called *keying a hash function* (see Bellare, Canetti and Krawczyk [5]). This method is attractive, because of the availability of hashing algorithms and their relative speed in software implementation; these algorithms are not subject to export restrictions.

Although this scheme can be implemented more efficiently in software than the previous scheme, the objection to the superfluous properties of the hash functions remains.

The keying method depends on the structure of the hash function. Tsudik [78] proposes three

methods of incorporating the key into the data. In the *secret prefix method*, $G_k(s) = H(k\|s)$, while in the *secret suffix*, we have $G_k(s) = H(s\|k)$. Finally, the *envelope method* combines the previous two methods with $G_k(s) = H(k_1\|s\|k_2)$ and $k = k_1\|k_2$.

Instead of including the key into the data, the key information can be included into the hashing algorithm. In iterative hash functions such as MD5 and SHA, the key can be incorporated into the initial vector, compression function or into the output transformation.

There have also been some attempts at defining and constructing *secure keyed hash functions* as independent primitives, namely by Berson, Gong and Lomas [9] and Bakhtiari, Safavi-Naini and Pieprzyk [1]. The former propose a set of criteria for secure keyed hash functions and give constructions using one-way hash functions. The latter argue that the suggested criteria for security is in most cases excessive; relaxing these allows constructions of more efficient secure keyed hash functions. Bakhtiari *et al* also give a design of a keyed hash function from scratch. Their design is based mostly on intuitive principles and lacks a rigorous proof of security. A similar approach is taken in the design of MDx-MAC by Preneel and van Oorschot [53], which is a scheme for constructing a MAC from an MD5-type hash function. It is conjectured that if MDx is a secure hash function, then MDx-MAC is a secure MAC.

5.3.1 Security analysis of computationally secure MACs

The security analysis of computational secure MACs has followed two different approaches. In the first approach, the security assessment is based on an analysis of some possible attacks. In the second approach, a security model is developed and used to examine the proposed MAC.

Security analysis through attacks
Consider a MAC algorithm that produces MACs of length m using a k bit key. In general an attack might result in a *successful forgery*, or in the *recovery of the key*. According to the classification given by Preneel and van Oorschot [54], a forgery in a MAC can be either *existential* – the opponent can construct a valid message and MAC without the knowledge of the key pair, or *selective* – the opponent can determine the MAC for a message of his choice. Protection against the former type of attack imposes more stringent conditions than the latter type of attack. A forgery is *verifiable* if the attacker can determine with a high probability whether the attack is successful. In a *chosen text attack* the attacker is given the MACs for the messages of his own choice. In an *adaptive attack* the attacker chooses text for which he can see the result of his previous request before forming his next request. In a *key recovery* attack the aim of the attacker is to find the key. If the attacker is successful, he can perform selective forgery on any message of his choice and the security of the system is totally compromised.

For an *ideal MAC* any method to find the key is as expensive as an exhaustive search of $O(2^k)$ operations. If $m < k$, the attacker may randomly choose the MAC for a message with the probability of success equal to $1/2^m$. However, in this attack the attacker cannot verify whether his attack has been successful.

The complexity of various attacks is discussed by several authors: Tsudik [78], Bakhtiari *et al* [2], Bellare *et al* [6]. Preneel and van Oorschot [53, 54] propose constructions resistant to such attacks. Some attacks can be applied to all MACs obtained using a specific construction method while other attacks are limited to particular instances of the method.

5.3.2 Formal security analysis

The main attempts at formalizing the security analysis of computationally secure MACs are due to Bellare *et al* [5], and Bellare and Rogaway [7]. In both papers, an attack model is firstly carefully defined and the security of a MAC with respect to that model is considered. Bellare *et al* [5] use their model to prove the security of a generic construction based on pseudorandom functions, while Bellare and Rogaway [7] use their model to prove the security of a generic construction based on hash functions.

MAC from pseudorandom functions

The formal definition of security given by Bellare *et al* [7] assumes that the enemy can ask the transmitter to construct tags for messages of his choice, and also ask the receiver to verify chosen message and tag pairs. The number of these requests is limited, and a limited time t can be spent on the attack. Security of the MAC is expressed as an upper bound on the enemy's chance of succeeding in its best attack.

The construction proposed by Bellare *et al* applies to any pseudorandom function. Their proposal, called XOR-MAC, basically breaks a message into blocks. For each block the output of the pseudorandom function is calculated, and the outputs are finally XORed. Two schemes based on this approach are proposed: the randomized XOR scheme and the counter based scheme.

The pseudorandom function used in the above construction can be an encryption function, like DES, or a hash function, like MD5. It is proved that the counter based scheme is more secure than the randomized scheme, and if DES is used, both schemes are more secure than CBC MAC. Some of the desirable features of this construction are *parallelizability* and *incrementality*. The former means that message blocks can be fed into the pseudorandom function in parallel. The latter refers to the feature of calculating incrementally the value of the MAC for a message s' which differs from s in only a few blocks.

MAC from hash functions

The model used by Bellare and Rogaway [5] is similar to the above one. The enemy can obtain information by asking queries; however, in this case queries are only addressed to the transmitter.

A family of functions $\{F_k\}$ is *(ϵ, t, q, L)-secure MAC* [6] if any adversary that is not given the key k, is limited to spend total time t, and sees the values of the function F_k computed on q messages $s_1, s_2, \cdots, s_q$ of its choice, each of length at most L, cannot find a message and tag pair (s, t), $s \neq s_i$, $i = 1, \ldots q$, such that $t = F_k(s)$ with probability better that ϵ.

Two general constructions for MAC from hash functions, the so-called NMAC (the Nested construction) and HMAC (the Hash based MAC) are given and their security is formally proved.

Theorem 5 *[5] If the keyed compression function f is a (ϵ_f, t, q, L)-secure MAC and the keyed iterated hash function F is (ϵ_F, t, q, L)-weakly collision-resistant then the NMAC is ($\epsilon_f + \epsilon_F$, t, q, L)-secure MAC.*

Weak collision-resistance is a much weaker notion than the collision resistance of (unkeyed) hash functions, because the enemy does not know the secret key and finding collision is much more difficult in this case. More precisely, a family of *keyed hash functions $\{F_k\}$ is (ϵ, t, q, L)-weakly collision-resistant* if any adversary that is not given the key k, is limited to spend total time t,

and sees the values of the function F_k computed on q messages $m_1, m_2, \ldots, m_q$ of its choice, each of length at most L, cannot find messages m and m' for which $F_k(m) = F_k(m')$ with probability better that ϵ.

With some extra assumptions similar results are proved for the HMAC construction.

A related construction is the *collisionful keyed hash function* proposed by Gong [33]. In his construction, the collisions are selectable and the resulting function is claimed to provide security against password guessing attacks. Bakhtiari, Safavi-Naini and Pieprzyk [3], [4] explore the security of Gong's function and a key exchange protocol based on collisionful hash functions.

5.4 Applications

The main application of a MAC is to provide protection against active spoofing (see Wegman and Carter [13]). This is particularly important in open distributed systems such as the Internet. Other applications include secure password checking and software protection. MACs can be used to construct encryption functions and have been used in authentication protocols in place of encryption functions (cf. Bird, Gopal, Herzberg, Janson, Kutten, Molva and Yung [12]). An important advantage of MAC functions is that they are not subject to export restrictions. Other applications of MAC functions are to protect software against viruses (cf. Radai [56]), or to protect computer files against tampering. Integrity checking is an important service in a computer operating system which can be automated with software tools.

6 DIGITAL SIGNATURES

Digital signatures are meant to be electronic equivalents of handwritten signatures. They should preserve the main features of handwritten signatures. Obviously, it is desirable that digital signatures be as legally binding as handwritten ones. There are three elements in every signature: the signer, the document, and the time of signing.

In most cases, the document already includes a timestamp. A digital signature must reflect both the content of the document and the identity of the signer. The signer is uniquely identified by its secret key. In particular, we require the signature to be:

- unique – a given signature reflects the document and can be generated by the signer only;
- unforgeable – it must be computationally intractable for an opponent to forge the signature;
- easy to generate by the signer and easy to verify by recipients; and
- impossible to deny by the signer (non-repudiation).

A digital signature differs from a handwritten signature in that it is not physically attached to the document on a piece of paper. Digital signatures have to be related both to the signer and the document by a cryptographic algorithm. Signatures can be verified by any potential recipient. Therefore, the verification algorithm must be public. Signature verification succeeds only when the signer and document match the signature.

There are two general classes of signature schemes:

- one-time signature schemes, and
- multiple signature schemes.

One-time signature schemes can be used to sign only one message. To sign a second message, the signature scheme has to be re-initialized; however, any signature can be verified repeatedly. Multiple signature schemes can be used to sign several messages without the necessity to re-initialize the signature scheme.

In practice, a signature scheme is required to provide a relatively short signature for a document of an arbitrary length. We sign the document by generating a signature for its digest. The hashing employed to produce the digest must be secure and collision free.

6.1 One-time signature schemes

This class of signature schemes can be implemented using any one-way function. These schemes were first developed using private key cryptosystems. We follow the original notation. An encryption algorithm is used as a one-way function. To set up the signature scheme, the signer chooses a one-way function (encryption algorithm). The signer selects an index k (secret key) randomly and uniformly from the set of keys, K. The index determines an instance of the one-way function, i.e., $E_k : \Sigma^n \to \Sigma^n$ where $\Sigma = \{0, 1\}$; it is known only by the signer. Note that n has to be large enough to avoid birthday attacks.

Lamport scheme
Lamport's scheme [40] generates signatures for n-bit messages. To sign a message, the signer first chooses randomly n key pairs:

$$(K_{10}, K_{11}), (K_{20}, K_{21}), \ldots, (K_{n0}, K_{n1}). \tag{1}$$

The pairs of keys are kept secret and are known to the signer only. Next, the signer creates two sequences, S and R:

$$\begin{aligned} S &= \{(S_{10}, S_{11}), (S_{20}, S_{21}), \ldots, (S_{n0}, S_{n1})\}, \\ R &= \{(R_{10}, R_{11}), (R_{20}, R_{21}), \ldots, (R_{n0}, R_{n1})\}. \end{aligned} \tag{2}$$

The elements of S are selected randomly and the elements of R are cryptograms of S so

$$R_{ij} = E_{K_{ij}}(S_{ij}) \quad \text{for} \quad i = 1, \ldots, n \quad \text{and} \quad j = 0, 1, \tag{3}$$

where E_K is the encryption function of the selected symmetric cryptosystem. S and R are stored in a read-only public register; they are known by the receivers.

The signature of a n-bit message $M = (m_1, \ldots, m_n)$, $m_i \in \{0, 1\}$ for $i = 1, \ldots, n$, is a sequence of cryptographic keys:

$$S(M) = \{K_{1i_1}, K_{2i_2}, \ldots, K_{ni_n}\} \tag{4}$$

where $i_j = 0$ if $m_j = 0$; otherwise $i_j = 1$, $j = 1, \ldots, n$. A receiver validates the signature $S(M)$ by verifying whether suitable pairs of S and R match each other for known keys.

Rabin scheme
In Rabin's scheme [55], a signer begins the construction of the signature by generating $2r$ keys at random:

$$K_1, K_2, \ldots, K_{2r}. \tag{5}$$

The parameter r is determined by the security requirements. The K_i are secret and known only to the signer. Next, the signer creates two sequences which are needed by the recipients to verify the signature. The first sequence,

$$S = \{S_1, S_2, \ldots, S_{2r}\}$$

comprises of binary blocks chosen at random by the signer. The second,

$$R = \{R_1, R_2, \ldots, R_{2r}\}$$

is created using the sequence S, $R_i = E_{K_i}(S_i)$ for $i = 1, \ldots, 2r$. S and R are stored in a read-only public register.

The signature is generated using the following steps. The message to be signed M is enciphered under keys $K_1, \ldots, K_{2r}$. The cryptograms:

$$E_{K_1}(M), \ldots, E_{K_{2r}}(M) \tag{6}$$

form the signature $S(M)$. The pair $(S(M), M)$ is sent to the receivers.

To verify the signature, a receiver selects randomly a $2r$-bit sequence σ of r-ones and r-zeros. A copy of σ is forwarded to the signer. Using σ, the signer forms an r-element subset of the keys with the property that K_i belongs to the subset if and only if the i-th element of the $2r$-bit sequence is '1'; $i = 1, \ldots, 2r$. The subset of keys is then communicated to the receiver. To verify the key subset, the receiver generates and compares r suitable cryptograms of S with the originals kept in the public register.

Matyas-Meyer scheme

Matyas and Meyer [41] propose a signature scheme based on the DES algorithm. However, any one-way function can be used in the scheme.

The signer first selects a random matrix $U = [u_{i,j}]$ $i = 1, \ldots, 30$, $j = 1, \ldots, 31$ and $u_{i,j} \in \Sigma^n$. Using U, a 31×31 matrix $KEY = [k_{i,j}]$ is constructed for $k_{i,j} \in \Sigma^n$. The first row of KEY matrix is chosen at random, the other rows are:

$$k_{i+1,j} = E_{k_{i,j}}(u_{i,j})$$

for $i = 1, \ldots, 30$ and $j = 1, \ldots, 31$. Finally, the signer installs in a public registry the matrix U and the vector $(k_{31,1}, \ldots, k_{31,31})$ (the last row of KEY).

To sign a message $m \in \Sigma^n$ the cryptograms

$$c_i = E_{k_{31,i}}(m) \text{ for } i = 1, \ldots, 31$$

are computed. The cryptograms are considered as integers and ordered according to their values so $c_{i_1} < c_{i_2} < \cdots < c_{i_{31}}$. The signature of m is the sequence of keys

$$SG_k(m) = (k_{i_1,1}, k_{i_2,2}, \ldots, k_{i_{31},31}).$$

The verifier takes the message m, recreates the cryptograms c_i and orders them in increasing order. Next, the signature-keys are put into the 'empty' matrix KEY in the entries indicated by the ordered sequence of c_i's. The verifier then repeats the signer's steps and computes all keys below the keys of the signature. The signature is accepted if the last row of KEY is identical to the row stored in the registry.

6.2 Signature schemes based on public-key cryptosystems

RSA signature scheme

First, a signer sets up the RSA system [59] with the modulus $N = p \times q$, where the two primes p and q are fixed. Next a random decryption key $d \in Z_N$ is chosen; the encryption key e is

$$e \times d \equiv 1 \pmod{(p-1, q-1)}.$$

The signer publishes both the modulus N and the decryption key d.

Given a message $M \in Z_N$, the signature generated by the signer is

$$S \equiv M^e \pmod{N}.$$

Since the decryption key is public, anyone can verify whether

$$M \equiv S^d \pmod{N}.$$

The signature is considered to be valid if this congruence is satisfied. RSA signatures are subject to various attacks which exploit the commutativity of exponentiation.

ElGamal signature scheme

The signature scheme due to ElGamal [27] is based on the discrete logarithm problem. The signer chooses a finite field $GF(p)$ for a sufficiently large prime p. A primitive element $g \in GF(p)$ and a random integer $r \in GF(p)$ are fixed. Next the signer computes

$$K \equiv g^r \pmod{p}$$

and announces K, g and p. To sign a message $M \in GF(p)$, the signer selects a random integer $R \in GF(p)$ such that $\gcd(R, p-1) = 1$ and calculates

$$X \equiv g^R \pmod{p}.$$

Using this data following congruence is solved

$$M \equiv r \times X + R \times Y \pmod{p-1}$$

for Y using Euclid's algorithm. The signature of M is the triple (M, X, Y). Note that r and R are kept secret by the signer. The recipient of the signed message forms

$$A \equiv K^X X^Y \pmod{p}$$

and accepts the message M as authentic if $A \equiv g^M \pmod{p}$. It is worth noting that knowledge of the pair (X, Y), does not reveal the message M. As a matter of fact, there are many pairs matching the message – for every pair (r, R) there is a pair (X, Y).

Since discrete exponent systems can be based on any cyclic group, the ElGamal signature scheme can be extended to this setting. A modification of ElGamal's signature was proposed as a Digital Signature Standard (DSS) in 1991 ([45]).

6.3 Special signatures

Sometimes additional conditions are imposed upon digital signatures. Blind signatures are useful in situations where the message to be signed should not be revealed to the signer. Unlike typical digital signatures, the undeniable versions require the participation of the signer in order to verify the signature. Fail-stop signatures are used whenever there is a need for protection against a very powerful adversary. As these signatures require interactions amongst the parties involved, the signatures are sometimes called *signature protocols*.

Blind Signatures

The concept of blind signatures was introduced by Chaum [18]. They are applicable to situations where the holder of a message M needs to get M signed by a signer (which could be a public registry) without revealing the message. This can be done with the following steps.

- The holder of the message first encrypts it.
- The holder sends a cryptogram of the message to the signer.
- The signer generates the signature for the cryptogram and sends it back to the holder.
- The holder decodes the encryption and obtains the signature of the message.

This scheme works if the encryption and signature operations commute, for example, the RSA scheme can be used to implement blind signatures.

Assume that the signer has set up a RSA signature scheme with modulus N and public decryption key d. The holder of the message M selects at random an integer $k \in Z_N$ and computes the cryptogram

$$C \equiv M \times k^d \pmod{N}.$$

The cryptogram C is now sent to the signer who computes the blind signature

$$S_C \equiv (M \times k^d)^e \pmod{N}.$$

The blind signature S_C is returned to the holder who computes the signature for M as follows

$$S_M \equiv S_C \times k^{-1} \equiv M^e \pmod{N}.$$

It is not necessary to have special signature schemes to generate blind signatures. It is enough for the holder of the message to use a secure hash function $h()$. To get a (blind) signature from the signer, the holder first compresses the message M using $h()$. The digest $D = h(M)$ is sent to the signer. After signing the digest, the signature $SG_k(D)$ is communicated to the holder who attaches the message M to the signature $SG_k(D)$. Note that the signer cannot recover the message M from the digest, since $h()$ is a one-way hash function. Also the holder cannot cheat by attaching a 'false' message unless collisions for the hash function can be found.

Undeniable Signatures

The concept of undeniable signatures is due to Chaum and van Antwerpen [19]. The characteristic feature of undeniable signatures is that signatures cannot be verified without the co-operation of the signer. Assume we have selected a large prime p and a primitive element $g \in GF(p)$.

Both p and g are public. The signer randomly selects its secret $k \in GF(p)$ and announces $g^k \pmod p$. For a message M, the signer creates the signature

$$S \equiv M^k \pmod p.$$

Verification needs the co-operation of the verifier and signer, and proceeds as follows.

- The verifier selects two random numbers $a, b \in GF(p)$ and sends $C \equiv S^a (g^k)^b \pmod p$ to the signer.
- The signer computes k^{-1} such that $k \times k^{-1} \equiv 1 \pmod{p-1}$ and returns $d = C^{k^{-1}} \equiv M^a \times g^b \pmod p$ to the verifier.
- The verifier accepts or rejects the signature as genuine depending on whether $d \equiv M^a \times g^b \pmod p$.

There are two possible ways in which a verification can fail. Either the signer has tried to disavow a genuine signature or the signature is indeed false. The first possibility is prevented by incorporating a disavowal protocol. The protocol requires two runs for verification. In the first run, the verifier randomly selects two integers $a_1, b_1 \in GF(p)$ and sends $C_1 \equiv S^{a_1}(g^k)^{b_1} \pmod p$ to the signer. The signer returns $d_1 = C_1^{k^{-1}}$ to the verifier. The verifier checks whether

$$d_1 \neq M^{a_1} \times g^{b_1} \pmod p.$$

If the congruence is not satisfied, the verifier repeats the process using a different pair $a_2, b_2 \in GF(p)$. The verifier concludes that S is a forgery if and only if

$$(d_1 g^{-b_1})^{a_2} \equiv (d_2 g^{-b_2})^{a_1} \pmod p;$$

otherwise, the signer is cheating.

Fail-Stop Signatures

The concept of fail-stop signatures was introduced by Pfitzmann and Waidner [48]. Fail-stop signatures protect signatures against a powerful adversary. As usual the signature is produced by a signer who holds a particular secret key. There are, however, many other keys which can be used to produce the same signature and which thus work with the original public key. There is a high probability that the key chosen by the adversary differs from the key held by the signer. Fail-stop signatures provide signing and verification algorithms as well as an algorithm to detect forgery.

Let k be a secret key known to the signer only and K be the public key. The signature on a message M is denoted as $s = SG_k(M)$. A fail-stop signature must satisfy the following conditions:

- An opponent with unlimited computational power can forge a signature with a negligible probability. More precisely, an opponent who knows the pair $(s = SG_k(M), M)$ and the signer's public key K, can create a collection of all keys $K_{s,M}$ such that $k^* \in K_{s,M}$ if and only if $s = SG_{k^*}(M) = SG_k(M)$. The size of $\mathcal{K}_{s,M}$ has to increase exponentially as a function of the security parameter n. Not knowing the secret k, the opponent can only randomly chose an element from $K_{s,M}$. Let this element be k^*. If the opponent signs another message $M^* \neq M$, it is a requirement that $s^* = SG_{k^*}(M^*) \neq SG_k(M^*)$ with a probability close to one.

- There is a polynomial-time algorithm which produces a proof of forgery as output, when given the following inputs: a secret key k, a public key K, a message M, a valid signature s, and a forged signature s^*.
- A signer with polynomially bounded computing power cannot construct a valid signature which it can later deny by proving it to be a forgery.

Clearly, after the signer has provided a proof of forgery, the scheme is compromised and is no longer used. This is why it is called 'fail-stop'.

7 Research Issues and Summary

In this chapter we discussed: Authentication, Hashing, Message Authentication Codes (MACs), and Digital Signatures. We have presented the fundamental ideas underlying each topic and indicated the current research developments in these topics. This is reflected in our bibliography.

We shall now summarize the topics covered in this chapter.

Authentication deals with the problem of providing assurance to a receiver that a communicated message originates from a particular transmitter, and that the received message has the same content as the transmitted message. A typical and widely used application of authentication occurs in computer networks. Here the problem is to provide a protocol to establish the identity of two parties wishing to communicate or make transactions via the network. Motivated by such applications, the theory of authentication codes has developed into a mature area of research, drawing from several areas of mathematics.

Hashing algorithms provide a relatively short digest of a much longer input. Hashing must satisfy the critical requirement that the digests of two distinct messages are distinct. A widely used type of hashing functions are constructed from block encryption ciphers. They have numerous applications in cryptology. Algebraic methods have also been proposed as a source of good hashing functions. These offer some provable security properties.

Message Authentication Codes or (MACs), are symmetric key primitives providing message integrity against active spoofing, by appending a cryptographic checksum to a message which is verifiable only by the intended recipient of the message. Message authentication is one of the most important ways of ensuring the integrity of information which is communicated by electronic means. This is especially relevant in the rapidly developing sphere of electronic commerce.

Digital signatures are the electronic equivalents of handwritten signatures. They are designed so as to preserve the essential features of handwritten signatures. They can be used to sign electronic documents and have potential application in legal contexts.

8 Defining Terms

authentication : is one of the main two goals of cryptography (the other is secrecy). An authentication system ensures that massages which are transmitted over a communication channel are authentic.

cryptology: is the art/science of design and analysis of cryptographic systems.

digital signatures : an asymmetric cryptographic primitive which is the digital counterpart of a signature and links a document to a unique person.

encryption algorithm: transforms an input text by "mixing" it with a randomly chosen bit string – the key, to produce the cipher text. In a symmetric encryption algorithm, the plain text can be recovered by applying the key to the cipher text.

hashing: hashing is accomplished by applying a function to an arbitrary length message to create a digest/hash value, which is usually of fixed length.

key: an input provided by the user of a cryptographic system. This piece of information is kept secret and is the source of security in a cryptographic system. Although some times a part of key information is made public, in which case the secret part is the source of security.

message authentication codes : is a symmetric cryptographic primitive that is used for providing authenticity.

plain text, cipher text: the cipher text is the "scrambled" version of an original source – the plain text. It is assumed that the scrambled text, produced by an encryption algorithm, can be inspected by persons not having the key and not reveal the content of the source.

9 Further Information

Current research in cryptology is represented in the Proceedings the conferences: CRYPTO, EUROCRYPT, ASIACRYPT, AUSCRYPT. There are also more specialized conferences dealing with topics such as: hashing, fast software encryption, and security. The proceedings are published by Springer in their LNCS series. *The Journal of Cryptology, IEEE Proceedings on Information Theory, Designs Codes and Cryptography* and several other journals publish extended versions of the articles which were presented in the above mentioned conferences.

Acknowledgments

We thank Anish Mathuria for all his comments and suggestions which have greatly helped us improve our exposition.

References

[1] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. Keyed hash functions. *Cryptography: Policy and Algorithms Conference*, LNCS Vol. 1029, Springer-Verlag, Berlin, 1995, pp. 201-214.

[2] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. Practical and secure message authentication. *Proc. Second Annual Workshop on Selected Areas in Cryptography (SAC'95)*, Ottawa, Canada, May 1995, pp. 55-68.

[3] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. On selectable collisionful hash functions. *Proc. Australasian Conference on Information Security and Privacy*, LNCS Vol. 1172, Springer-Verlag, Berlin, 1996, pp. 287-294.

[4] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. Password-based authenticated key exchange using collisionful hash functions. *Proc. Australasian Conference on Information Security and Privacy*, LNCS Vol. 1172, Springer-Verlag, Berlin, 1996, pp. 299-310.

[5] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. *Proc. Crypto'96*, LNCS Vol. 110, Springer-Verlag, Berlin, 1996, pp. 1-15.

[6] M. Bellare, J. Kilian, and P. Rogaway. The security of cipher block chaining. *Proc. Crypto'94*, LNCS Vol. 839, Springer-Verlag, Berlin, 1994, pp. 348-358.

[7] M. Bellare and P. Rogaway. The exact security of digital signatures - how to sign with RSA and Rabin. *Proc. Eurocrypt'96*, LNCS Vol. 1070, Springer-Verlag, Berlin, May 1996, pp. 399-416.

[8] T. A. Berson. Differential cryptanalysis mod 2^{32} with applications to MD5. *Proc. Eurocrypt'92*, LNCS, Vol. 658, Springer-Verlag, Berlin, 1993, pp. 71-80.

[9] T. A. Berson, L. Gong, and T. M. A. Lomas. Secure, keyed, and collisionful hash functions. TR SRI-CSL-94-08, SRI International, Dec. 1993. Revised version (Sept. 2, 1994).

[10] T. Beth, D. Jungnickel, and H. Lenz. *Design Theory*. Cambridge University Press, Cambridge, 1986.

[11] E. Biham and A. Shamir. Differential cryptanalysis of DES-like Cryptosystems. *Journal of Cryptology*, 4:3-72, 1991.

[12] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung. The KryptoKnight family of light-weight protocols for authentication and key distribution. *IEEE/ACM Transactions on Networking*, 3, 1995, pp. 31-41.

[13] J. L. Carter and M. N. Wegman. Universal class of hash functions. *Journal of Computer and System Sciences*, 18(2):143-154, 1979.

[14] C. Charnes and J. Pieprzyk. Linear nonequivalence versus nonlinearity. *Proc. Auscrypt'92*, LNCS Vol. 718, Springer-Verlag, Berlin, 1993, pp. 156-164.

[15] C. Charnes and J. Pieprzyk. Attacking the SL_2 Hashing scheme. *Proc. Asiacrypt'94*, LNCS Vol. 917, Springer-Verlag, Berlin, 1995, pp. 322-330.

[16] C. Charnes, L. O'Connor, J. Pieprzyk, R. Safavi-Naini, and Y. Zheng. Comments on GOST encryption algorithm. *Proc. Eurocrypt'94*, LNCS Vol. 950, Springer-Verlag, Berlin, 1995, pp. 433-438.

[17] C. Charnes and J. Pieprzyk. Weak parameters for the SL_2 hashing function. Manuscript, 1996.

[18] D. Chaum. Blind signatures for untraceable payments. *Proc. Crypto 82*, Plenum Press, New York, 1983, pp. 199-203.

[19] D. Chaum and H. Van Antwerpen. Undeniable signatures. *Proc. Crypto 89*, LNCS Vol. 435, Springer-Verlag, Berlin, 1990, pp. 212-217.

[20] D. Coppersmith. Analysis of ISO/CCITT Document X.509 Annex D. Internal Memo, IBM T. J. Watson Center, June 11, 1989.

[21] J. Daemen, R. Govaerts, and J. Vandewalle. A framework for the design of one-way hash functions including cryptanalysis of Damgard one-way function based on a cellular automaton. *Proc. Asiacrypt'91*, LNCS Vol. 739, Springer-Verlag, Berlin, 1993, pp. 82-97.

[22] I. Damgård. A design principle for hash functions. *Proc. Crypto'89*, LNCS Vol. 435, Springer-Verlag, Berlin, 1990, pp. 416-427.

[23] D. W. Davies and W. L. Price. The application of digital signatures based on public-key cryptosystems. *Proc. Fifth Int. Computer Communications Conference*, Oct. 1980, pp. 525-530.

[24] A. De Santis and M. Yung. On the design of provably-secure cryptographic hash functions. *Proc. Eurocrypt'90*, LNCS Vol. 473, Springer-Verlag, Berlin, 1990, pp. 377-397.

[25] Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures. *Proc. Crypto'91*, LNCS, Vol. 576, Springer-Verlag, Berlin, 1992, pp. 457-469.

[26] H. Dobbertin. Cryptanalysis of MD4. *Proc. Fast Software Encryption Workshop*, LNCS Vol. 1039, Springer-Verlag, Berlin, 1996, pp. 71-82.

[27] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469-472, 1985.

[28] V. Fak. Repeated use of codes which detect deception. *IEEE Transactions on Information Theory*, 25(2):233-234, Mar. 1979.

[29] C. Gehrmann, M. van Dijk, and B. Smeets. Unconditionally secure group authentication. Submitted for publication.

[30] W. Geiselmann. A note on the hash function of Tillich and Zémor. *Cryptography and Coding*, LNCS Vol. 1025, Springer-Verlag, Berlin, 1995, pp. 257-263.

[31] J. K. Gibson. Discrete logarithm hash function that is collision free and one way. *IEE Proc.-E*, 138(6):407-427, 1991.

[32] F. J. MacWilliams, E. N. Gilbert, and N. J. A. Sloane. Codes which detect deception. *Bell System Technical Journal*, 53(3):405-424, 1974.

[33] L. Gong. Collisionful keyed hash functions with selectable collisions. *Information Processing Letters*, 55:167-170, 1995.

[34] R. Impagliazzo and M. Naor. Efficient cryptographic schemes as provably secure as subset sum. *Proc. 30th IEEE Symposium on Foundations of Computer Science*, 1989, pp. 236-241.

[35] T. Johansson. Lower bound on the probability of deception in authentication with arbitration. *IEEE Transaction on Information Theory*, 40:1573-1585, 1994.

[36] T. Johansson. Authentication codes for nontrusting parties obtained from rank metric codes. *Designs, Codes and Cryptography*, 6:205-218, 1995.

[37] T. Johansson, G. Kabatianskii, and B. Smeets. On the relation between A-codes and codes correcting independent errors. *Proc. Eurocrypt'93*, LNCS Vol. 765, Springer-Verlag, Berlin, 1994, pp. 1-11.

[38] H. Krawczyk. LFSR-based hashing and authentication. *Proc. Crypto'94*, LNCS Vol. 839, Springer-Verlag, Berlin, 1994, pp. 129-139.

[39] K. Kurosawa. New bounds on authentication code with arbitration. *Proc. Crypto'94*, LNCS Vol. 839, Springer-Verlag, Berlin, 1994, pp. 140-149.

[40] L. Lamport. Constructing digital signatures from a one-way function. TR CSL-98, SRI International, Oct. 1979.

[41] S. M. Matyas and C. H. Meyer. Electronic signature for data encryption standard. *IBM Tech. Disc. Bull.*, 24(5), 1981.

[42] R. C. Merkle. A fast software one-way hash function. *Journal of Cryptology*, 3(1):43–58, 1989.

[43] R. C. Merkle. One way hash functions and DES. *Proc. Crypto'89*, LNCS Vol. 435, Springer-Verlag, Berlin, 1990, pp. 428-446.

[44] M. Naor and M. Yung. Universal one-way hash functions and their Cryptographic applications. *Proc. 21st ACM Symposium on Theory of Computing*, Seattle, 1989, pp. 33-43.

[45] National Institute for Standards and Technology. Digital Signature Standard (DSS). *Federal Register*. 56(169), Aug. 30 1991.

[46] K. Ohta and K. Koyama. Meet-in-the-middle attack on digital signature schemes. *Proc. Auscrypt'90*, LNCS Vol. 453, Springer-Verlag, Berlin, 1990, pp. 110-121.

[47] D. Pei. Information theoretic bounds for authentication codes and PBIB. *Presented at the Rump session of Asiacrypt'91.*

[48] B. Pfitzmann and M. Waidner. Fail-stop signatures and their applications. *Proc. Securicom'91*, 1991, pp. 338-350.

[49] J. Pieprzyk and B. Sadeghiyan. *Design of Hashing Algorithms.* LNCS Vol. 756, Springer-Verlag, New York, 1993.

[50] B. Preneel. *Analysis and Design of Cryptographic Hash Functions.* PhD thesis, Katholieke Universiteit, Leuven, 1993.

[51] F. Piper and H. Beker. *Cipher systems.* Northwood Books, London, 1982.

[52] B. Preneel, D. Chaum, W. Fumy, C. J. A. Jansen, P. Landrock, and G. Roelofsen. Race integrity primitives evaluation (RIPE): A Status Report. *Proc. Eurocrypt'91*, LNCS Vol. 547, Springer-Verlag, Berlin, 1991, pp. 547-551.

[53] B. Preneel and P. C. van Oorschot. MDx-MAC and building fast MACs from hash functions. *Proc. Eurocrypt'96*, LNCS Vol. 1070, Springer-Verlag, Berlin, 1996, pp. 1-14.

[54] B. Preneel and P. C. van Oorschot. On the security of two MAC Algorithms. *Proc. Eurocrypt'96*, LNCS Vol. 1070, Springer-Verlag, Berlin, 1996, pp. 19-32.

[55] M. O. Rabin. Digitalized signatures. *Foundations of Secure Computation.* Academic Press, 1978, pp. 155-168.

[56] Y. Radai. Checksumming techniques for anti-viral purposes. *International Virus Bulletin Conference*, Sept. 1991.

[57] R. L. Rivest. *RFC* 1321*: The MD5 message-digest algorithm.* Internet Activities Board, Apr. 1992.

[58] R. L Rivest. The MD4 message digest algorithm. *Proc. Crypto'90*, LNCS Vol. 537, Springer-Verlag, Berlin, 1991, pp. 303-311.

[59] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[60] R. S. Rees and D. R. Stinson. Combinatorial characterization of authentication codes II. Preprint.

[61] P. Rogaway. Bucket hashing and its application to fast message authentication. *Proc. Crypto'95*, LNCS Vol. 963, Springer-Verlag, Berlin, 1995, pp. 30-42.

[62] J. Rompel. One-way functions are necessary and sufficient for secure signatures. *Proc.* 22*nd ACM Symposium on Theory of Computing*, Baltimore, Maryland, 1990, pp. 387-394.

[63] U. Rosenbaum. A lower bound on authentication after having observed a sequence of messages. *Journal of Cryptology*, 6(3):135-156, 1993.

[64] R. Safavi-Naini. Three systems for shared generation of authenticators. *Proc. Cocoon'96*, LNCS Vol. 1090, Springer-Verlag, Berlin, 1996, pp. 401-411.

[65] R. Safavi-Naini and K. Martin, Unconditionally secure authentication systems with shared generation of authenticators. Submitted for publication.

[66] G. J. Simmons, A game theory model of digital message authentication. *Congressus Numerantium*, 34: 413-424, 1982.

[67] G. J. Simmons. A survey of information authentication. *Contemporary Cryptology: The Science of Information Integrity*, IEEE Press, 1992, pp. 379-419.

[68] D. R. Stinson. A provably secure hash function equivalent to the discrete logarithm problem. TR CCIS Lincoln, Mar. 1992.

[69] D. R. Stinson. Combinatorial characterisation of authentication codes. *Proc. Crypto'91*, LNCS, Vol. 576, Springer-Verlag, Berlin, 1991, pp. 62-73.

[70] D. R. Stinson. Combinatorial techniques for universal hashing. *Journal of Computer and System Sciences*, 48:337-346, 1994.

[71] D. R. Stinson. The combinatorics of authentication and secrecy codes. *Journal of Cryptology*, 2(1):23-49, 1990.

[72] D. R. Stinson. Universal hashing and authentication codes. *Designs, Codes and Cryptography*, 4:369-380, 1994.

[73] R. Taylor. Near optimal unconditionally secure authentication. *Proc. Eurocrypt'94*, LNCS, Vol. 950, Springer-Verlag, Berlin, 1995, pp. 245-255.

[74] J-P. Tillich and G. Zémor. Hashing with SL_2. *Proc. Crypto'94*, LNCS Vol. 839, Springer-Verlag, Berlin 1994, pp. 40-49.

[75] L. Tombak and R. Safavi-Naini. Authentication codes that are r-fold secure against spoofing. *Proc. 2nd ACM Conference on Computer and Communication Security*, 1994, pp. 166-169.

[76] L. Tombak and R. Safavi-Naini. Authentication codes in plaintext and content-chosen attacks. *Designs, Codes and Cryptography*, 6:83-99, 1995.

[77] L. Tombak and R. Safavi-Naini. Combinatorial characterization of A-codes with r-fold security. *Proc. Asiacrypt'94*, LNCS Vol. 917, Springer-Verlag, Berlin, 1995, pp. 211-223.

[78] G. Tsudik. Message authentication with one-way hash functions. *IEEE Infocom'92*, May 1992, pp. 2055-2059.

[79] M. Walker. Information theoretic bounds for authentication schemes. *Journal of Cryptology*, 2(3): 133-138, 1990.

[80] M. N. Wegman and J. L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265-279, 1981.

[81] R. S. Winternitz. Producing a one-way hash function from DES. *Proc. Crypto'83*, Plenum Press, New York, 1984, pp. 203-207.

[82] P. Vanroose, B. Smeets, and Zhe-Xian Wan. On the construction of authentication codes with secrecy and codes withstanding spoofing attacks of order $L \geq 2$. *Proc. Eurocrypt'90*, LNCS Vol. 473, Springer-Verlag, Berlin, 1990, pp. 306-312.

[83] M. Yung and Y. Desmedt. Arbitrated unconditionally secure authentication can be unconditionally protected against arbiter's attack. *Proc. Crypto'90*, LNCS Vol. 537, Springer-Verlag, Berlin, 1990, pp. 177-188.

[84] Y. Zheng, T. Matsumoto, and H. Imai. Structural properties of one-way hash functions. *Proc. Crypto'90*, LNCS Vol. 537, Springer-Verlag, Berlin, 1991, pp. 285-302.

[85] Y. Zheng, J. Pieprzyk, and J. Seberry. HAVAL - a one-way hashing algorithm with variable length of output. *Proc. Auscrypt'92*, LNCS Vol. 718, Springer-Verlag, Berlin, 1993, pp. 83-104.