M

m-Sequence

►Maximal-Length Sequences

M-Invariance

Xiaokui Xiao
School of Computer Engineering, Nanyang Technological University, Singapore

Related Concepts

►ℓ-Diversity; ►Generalization; ►k-Anonymity; ►Microdata Protection

Definition

m-invariance is a technique for protecting privacy when publishing the snapshots of dynamic datasets that contain sensitive personal information.

Theory

Organizations like census bureaus and hospitals maintain large datasets (referred to as *microdata*) that contain personal information (e.g., census data and medical records). Such data collections are of significant research value, and there is much benefit in making them publicly available. Nevertheless, as the data is sensitive in nature, proper measures must be taken to ensure that its publication does not endanger the privacy of the individuals that contributed the data. A canonical solution to this problem is to modify the data before releasing it to the public, such that the modification prevents inference of private information while retaining statistical characteristics of the data.

For example, suppose that a hospital wants to release the microdata in Table 1 while preventing an adversary to infer the disease of any individual. A naive solution is to remove the attribute *Name* and publish the rest of the data, which, however, may lead to privacy breach if the adversary knows the *Age* and *Zip Code* values of each individual in advance [1]. For instance, consider that the adversary knows Bob's age 21, Zip code 12000, and the fact that Bob has been hospitalized before (and thus has a tuple in the microdata). Then, the adversary can find out that the first tuple in Table 1 is associated with Bob, namely, Bob must have contracted *dyspepsia*. Here, the attributes *Age* and *Zip Code* are referred to as the *quasi-identifier* attributes, as they can be combined to pinpoint individuals. On the other hand, the attribute *Disease* is referred as the *sensitive* attribute, as it captures the private information the hospital aims to protect.

Generalization [1] is a popular method for preserving privacy in the scenario like the above. Given a microdata table, generalization first divides the tuples into several *QI-groups*, such that each group contains a sufficiently diverse set of sensitive values [2]; after that, the QI values in each QI-group is transformed into a uniform format. For example, Table 2 illustrates a generalized version of Table 1. The transformation is based on five QI-groups, each of which is assigned a group ID as indicated in the first column of Table 2. Suppose that the hospital publishes Table 2. The previous adversary can no longer uniquely decide Bob's disease, since both of the first two tuples in Table 2 can be matched to Bob, i.e., Bob's disease may be *dyspepsia* or *bronchitis*.

One drawback of generalization is that it cannot support republication of the microdata after the data is updated with tuple insertions or deletions. For instance, suppose that a hospital releases patients' records quarterly, but each publication includes only the results of diagnoses in the 6 months preceding the publication time. Table 1 shows the microdata for the first release, at which time the hospital publishes the generalization in Table 2. The microdata at the second release is presented in Table 3. The tuples of Alice, Andy, Helen, Ken, and Paul have been deleted (as they correspond to diagnoses made over 6 months ago), while 5 new tuples (with names italicized) have been inserted. Accordingly, the hospital publishes the generalization in Table 4.

Even though both Tables 2 and 4 have been generalized, an adversary can still precisely determine the disease of a patient by exploiting the correlation between the two snapshots. Assume, again, an adversary has Bob's age and Zip code, and knows that Bob has a record in both Tables 1 and 3 (i.e., Bob was admitted for treatment within six

Henk C.A. van Tilborg & Sushil Jajodia (eds.), *Encyclopedia of Cryptography and Security*, DOI 10.1007/978-1-4419-5906-5,

M-Invariance. Table 1 Microdata T_1

Name	Age	Zip Code	Disease
Bob	21	12000	Dyspepsia
Alice	22	14000	Bronchitis
Andy	24	18000	Flu
David	23	25000	Gastritis
Gary	41	20000	Flu
Helen	36	27000	Gastritis
Jane	37	33000	Dyspepsia
Ken	40	35000	Flu
Linda	43	26000	Gastritis
Paul	52	33000	Dyspepsia
Steve	56	34000	Gastritis

M-Invariance. Table 2 Generalization T_1^*

Group ID	Age	Zip Code	Disease
1	[21, 22]	[12k, 14k]	Dyspepsia
1	[21, 22]	[12k, 14k]	Bronchitis
2	[23, 24]	[18k, 25k]	Flu
2	[23, 24]	[18k, 25k]	Gastritis
3	[36, 41]	[20k, 27k]	Flu
3	[36, 41]	[20k, 27k]	Gastritis
4	[37, 43]	[26k, 35k]	Dyspepsia
4	[37, 43]	[26k, 35k]	Flu
4	[37, 43]	[26k, 35k]	Gastritis
5	[52, 56]	[33k, 34k]	Dyspepsia
5	[52, 56]	[33k, 34k]	Gastritis

M-Invariance. Table 3 Microdata T_2

Name	Age	Zip Code	Disease
Bob	21	12000	Dyspepsia
David	23	25000	Gastritis
Emily	25	21000	Flu
Jane	37	33000	Dyspepsia
Linda	43	26000	Gastritis
Gary	41	20000	Flu
Mary	46	30000	Gastritis
Ray	54	31000	Dyspepsia
Steve	56	34000	Gastritis
Tom	60	44000	Gastritis
Vince	65	36000	Flu

months before both publication times). Based on Table 2, the adversary is certain that Bob must have contracted either *dyspepsia* or *bronchitis*. From Table 4, the adversary finds out that Bob's disease must be either *dyspepsia* or *gastritis*. By combining the above knowledge, the adversary can easily infer Bob's real disease *dyspepsia*.

M-Invariance. Table 4 Generalization T_2^*

Group ID	Age	Zip Code	Disease
1	[21, 23]	[12k, 25k]	Dyspepsia
1	[21, 23]	[12k, 25k]	Gastritis
2	[25, 43]	[21k, 33k]	Flu
2	[25, 43]	[21k, 33k]	Dyspepsia
2	[25, 43]	[21k, 33k]	Gastritis
3	[41, 46]	[20k, 30k]	Flu
3	[41, 46]	[20k, 30k]	Gastritis
4	[54, 56]	[31k, 44k]	Dyspepsia
4	[54, 56]	[31k, 44k]	Gastritis
5	[60, 65]	[36k, 44k]	Gastritis
5	[60, 65]	[36k, 44k]	Flu

In fact, it is simply impossible to publish a generalization of Table 3 without incurring privacy breach due to a phenomenon referred to as *critical absence*. To illustrate this, consider the hospital publication scenario depicted in Tables 1, 2, 3, and 4. Given Table 1, an adversary (having Bob's QI-particulars) is sure that Bob contracted *dyspepsia* or *bronchitis*. The value *bronchitis*, however, is absent in the microdata (Table 4) at the second release. As a result, no matter how Table 3 is generalized, publishing the generalized version always enables the adversary to eliminate the possibility that Bob contracted *bronchitis*. Therefore, Bob's privacy will necessarily be breached after the second release.

m-invariance is a technique that incorporates *counterfeits* with generalization to handle republication of dynamic microdata. To illustrate the idea, let us consider the moment when the hospital has published Table 2 (with respect to the microdata Table 1) and tries to release an anonymized version of Table 3. Now, imagine that the hospital publishes Table 5, and an auxiliary Table 6. Table 5 involves a generalized tuple for every row in Table 3, together with two counterfeit tuples c_1 and c_2 The thirteen tuples are partitioned into six QI-groups. Table 6 indicates that a counterfeit is placed in QI-groups 1 and 3, respectively. The purpose of releasing such statistics is to enhance the effectiveness of data analysis. Even with these statistics, an adversary's chance of figuring out individual privacy is still limited, as explained shortly.

From an adversary's perspective, a counterfeit tuple is indistinguishable from the other rows in the QI-group (that contains the counterfeit). Let us consider once more the adversary who has the precise QI values of Bob and attempts to infer the disease of Bob from Tables 2, 5, and 6. The adversary knows that the tuple of Bob must have been generalized to the first QI-groups of Tables 2

M-Invariance. Table 5 Counterfeited generalization T_3^*

Name	Group ID	Age	Zip Code	Disease
Bob	1	[21, 22]	[12k, 14k]	Dyspepsia
c_1	1	[21, 22]	[12k, 14k]	Bronchitis
David	2	[23, 25]	[21k, 25k]	Gastritis
Emily	2	[23, 25]	[21k, 25k]	Flu
Jane	3	[37, 43]	[26k, 33k]	Dyspepsia
c_2	3	[37, 43]	[26k, 33k]	Flu
Linda	3	[37, 43]	[26k, 33k]	Gastritis
Gary	4	[41, 46]	[20k, 30k]	Flu
Mary	4	[41, 46]	[20k, 30k]	Gastritis
Ray	5	[54, 56]	[31k, 34k]	Dyspepsia
Steve	5	[54, 56]	[31k, 34k]	Gastritis
Tom	6	[60, 65]	[36k, 44k]	Gastritis
Vince	6	[60, 65]	[36k, 44k]	Flu

M-Invariance. Table 6 The auxiliary relation

Group ID	Count
1	1
3	1

and 5, respectively. These groups encompass the same set of sensitive values {*dyspepsia, bronchitis*}. Therefore, the adversary cannot eliminate any disease (from the previous set) that Bob cannot have contracted. Even if the adversary learns (from Table 6) that a counterfeit exists in QI-group 1 of Table 5, he still cannot narrow down the possible diseases of Bob. In fact, to the adversary, there is a 50% chance that the first tuple of Table 3.1a would be the counterfeit. The generalization in Table 5 is referred as a *counterfeited generalization*.

The two releases (Tables 2 and 5) have an important property If a tuple appears in the microdata at both publication timestamps, it is generalized to two QI-groups (one per timestamp) containing the same sensitive values. For instance, the tuple <Jane, 37, 33k, dyspepsia> belongs to both Tables 2 and 5. It is generalized to QI-groups 4 and 3 in Tables 2.1b and 3.1a, respectively. The two groups include an equivalent set of diseases: {*dyspepsia, flu, gastritis*} (as is achieved via a counterfeit c_2). As a result, even if an adversary finds out both QI-groups, he can only conjecture that Jane's disease may be an element in that equivalent set.

In general, m-invariance requires that the counterfeited generalizations of a dynamic dataset should satisfy the following three conditions. First, each QI-group in any anonymized snapshot $T^*(i)$ (notation for the i-th release) must have at least m tuples, where m is a userspecified parameter. Second, no QI-group should contain two tuples with the same sensitive value. Finally, if a tuple t (from the microdata) is involved in several anonymized snapshots, the QI-groups in all those snapshots containing t must have exactly the same set of sensitive values signature.

The privacy guarantee of m-invariance can be formalized as follows. Assume that the adversary (1) knows the QI values of every individual, as well as the timestamp at which the individual's record is inserted into or deleted from the microdata, but (2) has no knowledge of the sensitive value of any individual. Let o be an arbitrary individual in the microdata and v be an arbitrary sensitive value. Then, when the adversary observes the generalized snapshots of the microdata produced with m-invariance, his/her posterior belief in the event that "o has a sensitive value v" is at most $1/m$ [3].

Recommended Reading

1. Samarati P (2001) Protecting respondents' identities in microdata release. IEEE Trans Knowl Data Eng 13(6):1010–1027
2. Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M (2007) L-diversity: privacy beyond k-anonymity. TKDD 1(1)
3. Xiao X, Tao Y (2007) M-invariance: towards privacy preserving re-publication of dynamic datasets. In: SIGMOD conference, Beijing, China, pp 689–700

MAA

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Synonyms

Message authentication algorithm

Related Concepts

►MAC Algorithms

Definition

MAA is a software oriented dedicated MAC algorithm with a 64-bit key and a 32-bit result.

Background

The Message Authentication Algorithm (MAA) was published in 1983 by Davies and Clayden in response to a

request of the UK Bankers Automated Clearing Services (BACS) [1, 2]. In 1987 it became a part of the ISO 8731 banking standard [3]; this standard was revised in 1992 and withdrawn in 2002.

Theory

MAA is software oriented ►MAC Algorithm with a 64-bit key and a 32-bit result; as MAA was designed for mainframes in the 1980s, its performance on 32-bit processors is excellent (about five times faster than *DES*).

A serious concern is that a 64-bit key no longer offers an adequate security level against *exhaustive key search*. Moreover, several undesirable properties of MAA have been identified by Preneel et al. in 1997 [4]; all these attacks exploit internal collisions (cf. ►MAC Algorithms). A forgery attack requires 2^{17} messages of 256 Kbytes or 2^{24} messages of 1 Kbyte; the latter circumvents the special MAA mode for long messages defined in the ISO standard. A key recovery attack on MAA requires 2^{32} chosen texts consisting of a single message block. The number of off-line 32-bit multiplications for this attack varies between 2^{44} for one key in 1,000 to about 2^{51} for one key in 50. This represents a significant reduction w.r.t. *exhaustive key search*, which requires $3 \cdot 2^{65}$ multiplications. Finally it is shown that MAA has 2^{33} *weak keys* for which it is rather easy to create a large cluster of collisions. These keys can be detected and recovered with 2^{27} chosen texts. None of the shortcut attacks described above offer an immediate threat to banking applications, in which a single chosen text is often sufficient to perform a serious attack.

Recommended Reading

1. Davies D (1985) A message authenticator algorithm suitable for a mainframe computer. In: Blakley GR, Chaum D (eds) Advances in cryptology – CRYPTO '84: proceedings, Santa Barbara, 19–22 August 1984. Lecture notes in computer science, vol 196. Springer, Berlin, pp 393–400
2. Davies D, Price WL (1989) Security for computer networks: an introduction to data security in teleprocessing and electronic funds transfer, 2nd edn. Wiley, Chichester
3. ISO 8731 (1992) Banking – approved algorithms for message authentication, Part 2: Message Authentication Algorithm (MAA) (withdrawn in 2002)
4. Preneel B, Rijmen V, van Oorschot PC (1997) A security analysis of the Message Authenticator Algorithm (MAA). Eur Trans Telecommun 8(5):455–470
5. Preneel B, van Oorschot PC (1996) On the security of two MAC algorithms. In: Maurer U (ed) Advances in cryptology – EUROCRYPT '96: proceedings, Saragossa, 12–16 May 1996. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 19–32

MAC Algorithms

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC,
Katholieke Universiteit Leuven and IBBT,
Leuven-Heverlee, Belgium

Related Concepts

►CMAC; ►HMAC; ►MAA; ►PMAC

Introduction

Electronic information stored and processed in computers and transferred over communication networks is vulnerable to both passive and active attacks. In a passive attack, the opponent tries to obtain information on the data sent; in an active attack, the opponent will attempt to modify the information itself, the claimed sender, and/or intended recipient.

Cryptographic techniques for information authentication focus on the origin of the data (*data origin authentication*) and on the *integrity* of the data, that is, the fact that the data has not been modified. Other aspects that can be important are the timeliness, the sequence with respect to other messages, and the intended recipient(s). Information authentication is particularly relevant in the context of financial transactions and electronic commerce. Other applications where information authentication plays an important role are alarm systems, satellite control systems, distributed control systems, and systems for ►access control. One can anticipate that authentication of voice and video will become increasingly important.

One can distinguish between three mechanisms for information authentication: MAC algorithms (here MAC is the abbreviation of Message Authentication Code), ►authentication *codes*, and ►digital signatures. The first two mechanisms are based on a secret key shared between sender and recipient. This means that if the sender authenticates a message with a MAC algorithm or an authentication code and later denies this, there is no way one can prove that she has authenticated the message (as the recipient could have done this as well). In technical terms, MAC algorithms and authentication codes cannot provide ►non-repudiation of origin. MAC algorithms are computationally secure: A necessary (but not sufficient) condition for their security is that the computing power of the opponent is limited. Authentication codes are combinatorial objects; one can compute the probability of the success of an attack exactly; this probability is independent on the computing power of the attacker. ►Digital signatures, introduced in 1976 by W. Diffie and M. Hellman, allow us to establish in an irrefutable way

the origin and content of digital information. As they are an asymmetric cryptographic technique, they can resolve disputes between the communicating parties.

MAC algorithms have been used for a long time in the banking community and are thus older than the open research in cryptology that started in the mid-1970s. However, MAC algorithms with good cryptographic properties were only introduced after the start of open research in the field. The first reference to a MAC is a 1972 patent application by Simmons et al. (reference 10 in [31]). Financial applications in which MACs have been introduced include electronic purses (such as Proton, CEPS (Common European Purse Specification), and Mondex) and credit/debit applications (e.g., the ►EMV-standard). MACs are also being deployed for securing the Internet (e.g., IP security, ►Ipsec and transport layer security, ►Transport Layer Security (TLS)). For all these applications MACs are preferred over ►digital signatures because they are two to three orders of magnitude faster, and MAC results are 4…16 bytes long compared to 40…128 bytes for signatures. On present-day computers, software implementations of MACs can achieve speeds from 8 to 50 cycles/byte, and MAC algorithms require very little resources on inexpensive 8-bit smart cards and on the currently deployed Point of Sale (POS) terminals. The disadvantage is that they rely on shared symmetric keys, the management of which is more costly and harder to scale than that of asymmetric key pairs.

Definition

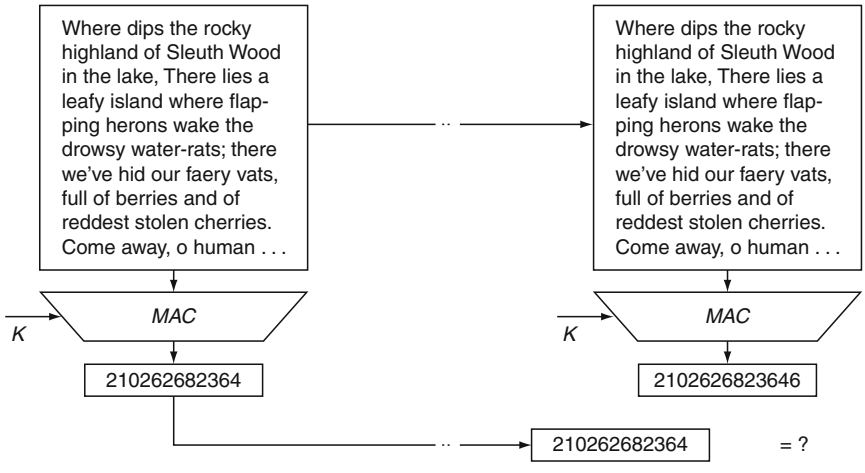
A MAC algorithm MAC() consists of three components:

- A key generation algorithm; for most MAC algorithms the secret key K is a random bit-string of length k – typical values for k are 56…256.
- A MAC generation algorithm; this algorithm computes from the text input x and the secret key K a MAC value $\mathrm{MAC}_K(x)$, which is bit-string of fixed length m – typical values for m are 24…96. A MAC generation algorithm can be randomized; in this case, a random string is added to the set of inputs and outputs of this algorithm. A MAC generation algorithm can be stateful; in this case the MAC generation algorithm keeps an internal state, which influences the result (e.g., a counter which is incremented after every use).
- A MAC verification algorithm; on input the text x, the MAC value $\mathrm{MAC}_K(x)$, the key K (and possibly a random string), the algorithm verifies whether the MAC value is correct or not; in practice this verification consists of a computation of the MAC value on the text and a check whether the result is identical to the MAC value provided.

Note that it is common to abuse terminology by abbreviating both the "MAC value" and the "MAC algorithm" as the "MAC."

In a communication context, sender and receiver will agree on a secret key (using a ►key agreement protocol. The sender will compute a MAC value for every message and append this to the message; on receipt of the message, the receiver will apply the MAC verification algorithm, which typically corresponds to recomputing the MAC value (see Fig. 1).

The main security requirement for a MAC algorithm is that it should be hard to forge a MAC value on a new text, that is, to compute a MAC for a new text. The resistance of a MAC algorithm against forgeries is also known as computation resistance. The next section investigates in more detail the security of MAC algorithms.



MAC Algorithms. Fig. 1 Using a MAC algorithm for data authentication

Security of MAC Algorithms

Attacks on MAC algorithms can be classified according to the type of control an adversary has over the device computing or verifying the MAC value. In a *chosen text attack*, an adversary may request and receive MACs corresponding to a number of texts of his choice, before completing his attack. In an *adaptive* chosen-text attack, requests may depend on the outcome of previous requests. In a *MAC-verification attack*, the opponent can submit text-MAC pairs of his choice to the verification device.

An opponent who tries to deceive the receiver, knows the description of the MAC algorithm, but he does not know the secret key. Attacks can be further distinguished based on their goals:

Forgery Attack

This attack consists of predicting the value of $\mathrm{MAC}_K(x)$ for a text x without initial knowledge of K. If the adversary can do this for a single text, he is said to be capable of ►existential forgery. If the adversary is able to determine the MAC for a text of his choice, he is said to be capable of ►selective forgery. Ideally, existential forgery is computationally infeasible; a less demanding requirement is that only selective forgery is so. Practical attacks often require that a ►forgery is *verifiable*, that is, that the forged MAC is known to be correct on beforehand with probability near 1. The text on which a MAC is forged shall be new, which means that it should not be one of the texts used in the MAC generation or verification queries (as this would allow for a trivial attack).

Key Recovery Attack

This attack consists of finding the key K itself from a number of text/MAC pairs. Such an attack is more powerful than forgery, since it allows for arbitrary selective forgeries. One distinguishes between exhaustive search and shortcut key recovery attacks; ideally, no shortcut key recovery attacks should exist.

We can now informally state the security requirement for a MAC algorithm: It should be computationally infeasible to generate an existential forgery under an adaptive chosen text attack (which also includes MAC verification queries). The success probability of an attacker is often computed as a function of m (the bit-length of the MAC) and the number q of queries.

Note that in certain environments, such as in wholesale banking applications, a chosen text attack is not a very realistic assumption: If an opponent can choose a single text and obtain the corresponding MAC, he can already make a substantial profit. Moreover, texts that are relevant may have a specific structure, which implies that an existential forgery may not pose a threat at all. However, it is better to be on the safe side, and to require resistance against chosen text attacks.

Below four attacks on MAC algorithms are considered: brute force key search; guessing of the MAC; a generic forgery attack based on internal collisions; and attacks based on cryptanalytical weaknesses.

Brute Force Key Search

If k denotes the bit-length of the key K, one can always try all 2^k key values and check which one is correct. If m is the size of the MAC and if one assumes that $\mathrm{MAC}_K(x)$ is a random function from the key to the MAC, then verification of such an attack requires about $\lceil k/m \rceil$ text-MAC pairs. To see this, note that the expected number of keys which will take the text x to a certain given MAC value is 2^{k-m}. Extending this argument, the expected number of keys which will take $\lceil k/m \rceil$ texts to certain given MAC values is $2^{k-(m\lceil k/m \rceil)} \leq 1$. For most MAC algorithms, the value of k/m lies between 1 and 4; it is reasonable to assume that such a small number of text-MAC pairs are available. The only exceptions are certain banking systems which use one key per transaction; in this case one exploits the combinatorial rather than the cryptographic properties of the MAC algorithm; this corresponds to the use of an ►authentication *code*.

Note that unlike for confidentiality protection, the opponent can only make use of the key if it is recovered within its active lifetime (which can be reasonably short). On the other hand, a single success during the lifetime of the system might be sufficient. This depends on a cost/benefit analysis, that is, how much one loses as a consequence of a forgery.

The only way to preclude a key search is to choose a sufficiently large key. In 2004, the protection offered by a 56-bit key is clearly insufficient. One also has to take into account what is known as a variant of "Moore's Law": The computing power for a given cost is multiplied by four every 3 years. This implies that if a system is deployed with an intended lifetime of 15 years, an extra security margin of about 10 bits is recommended. Keys of 80–90 bits are adequate for medium-term security (10–15 years), and long-term protection (50 years or more) is offered by keys of 128 bits. The entry on ►exhaustive key search provides more details.

MAC Guessing Attack

A second very simple attack is to choose an arbitrary (fraudulent) text, and to append a randomly chosen MAC

value. An alternative strategy is to guess the key value and compute the corresponding MAC value. Ideally, the probability that this MAC value is correct is equal to $\max(1/2^m, 1/2^k)$, where m is the number of bits of the MAC value and k is the number of bits in the key. This value should be multiplied with the expected profit corresponding to a fraudulent text, which results in the expected value of one trial. Repeated trials can increase this expected value, but note that in a good implementation, repeated MAC verification errors will result in a security alarm (the forgery is not verifiable). For most applications, $k > m$ and $m = 32...64$ is sufficient to make this attack uneconomical.

Internal Collision Attack on Iterated MAC Algorithms

The most common message authentication algorithms today are iterated MAC algorithms. The MAC input x is padded to a multiple of the block size, and is then divided into t blocks denoted x_1 through x_t. The MAC involves an initial value $H_0 = IV$, a *compression function* f, an *output transformation* g, and an n-bit ($n \geq m$) *chaining variable* H_i between stage $i-1$ and stage i:

$$H_i = f(H_{i-1}, x_i), \quad 1 \leq i \leq t$$
$$\mathrm{MAC}_K(x) = g(H_t).$$

The secret key may be employed in f and/or in g.

Next we describe a general forgery attack by Preneel and van Oorschot [27, 29] that applies to all iterated MACs. Its feasibility depends on the bit sizes n of the chaining variable and m of the MAC result, the nature of the output transformation g, and the number s of common trailing blocks of the known texts ($s \geq 0$). The basic attack requires several known texts, but only a single chosen text. However, under certain conditions restrictions are imposed on the known texts; for example, if the input length itself is an input to the output transformation, all inputs must have an equal length.

First, some terminology is introduced. For an input pair (x, x') with $MAC_K(x) = g(H_t)$ and, $MAC_K(x') = g(H'_t)$ a collision is said to occur if $MAC_K(x) = MAC_K(x')$. This collision is called an *internal* collision if $H_t = H'_t$, and an *external* collision if $H_t \neq H'_t$ but $gH_t = gH'_t s$.

The attack starts with the following simple observation:

Lemma 1 *An internal Collision for an iterated MAC algorithm allows a verifiable MAC forgery, through a chosen-text attack requiring a single chosen text.*

This follows since for an internal collision (x, x'), $MAC_K(x||y) = MAC_K(x'||y)$ for any single block y; thus a requested MAC on the chosen text $x||y$ provides a forged MAC (the same) for $x'||y$ (here $||$ denotes concatenation). Note this assumes that the MAC algorithm is deterministic. Also, the forged text is of a special form, which may limit the practical impact of the attack.

The following propositions indicate the complexity to find an internal collision. They are based on the ▶birthday paradox and extensions thereof.

Proposition 1 *Let MAC() be an iterated MAC algorithm with n-bit chaining variable and m-bit result, and an output transformation g that is a permutation. An internal collision for MAC can be found using an expected number of* $u = \sqrt{2} \cdot 2^{n/2}$ *known text-MAC pairs of at least* $t = 2$ *blocks each.*

Proposition 2 *Let MAC() be an iterated MAC algorithm with n-bit chaining variable and m-bit result, and output transformation g, which is a random function. An internal collision for h can be found using u known text-MAC pairs of at least* $t = 2$ *blocks each and v chosen texts of at least three blocks. The expected values for u and v are as follows:* $u = \sqrt{2} \cdot 2^{n/2}$ *and* $v \approx \min(2^{n/2}, 2^{n-m})$.

Proposition 3 *Let MAC() be an iterated MAC with n-bit chaining variable, m-bit result, a compression function f which behaves like a random function (for fixed* x_i*), and output transformation g. An internal collision for MAC can be found using u known text-MAC pairs, where each text has the same substring of* $s \geq 0$ *trailing blocks, and v chosen texts. The expected values for u and v are:* $u = \sqrt{2/(s+1)} \times 2^{n/2}$; $v = 0$ *if g is a permutation or* $s + 1 \geq 2^{n-m+6}$*, and otherwise* $v \approx 2^{n-m}/(s+1)$.

Weaknesses of the Algorithm

The above attacks assume that no shortcuts exist to break the MAC algorithm (either for forgery or for key recovery). The security of existing MAC algorithms relies on unproven assumptions: Even if the security of the MAC algorithm is reduced in a provable way to the pseudo-randomness properties of a block cipher or of the compression function of a hash function, these properties themselves cannot be proved. Therefore, it is recommended to use only well-established algorithms which have been subjected to an independent evaluation and a regular review of the algorithm based on progress in cryptanalysis is recommended. A typical example of a weak construction for a MAC algorithm is one which consists of inserting a secret key into the input of a hash function.

Practical MAC Algorithms

Compared to the number of ►block ciphers and ►hash functions, relatively few dedicated MAC algorithms have been proposed [12]. The main reason is that MACs have been derived from other primitives (initially from block ciphers, but also from hash functions), which reduces the need for dedicated proposals. The following section lists the most important constructions.

Based on Block Ciphers

The oldest and most popular MAC algorithm is certainly CBC-MAC, which is based on a block cipher and which has been widely standardized. For a more detailed discussion see ►CBC-MAC and variants. CBC-MAC is an iterated MAC algorithm. The most common padding method consists of appending a one bit followed by between 0 and $n-1$ zero bits such that the resulting string has an input length that is a multiple of n [17]. Denote the resulting string as $x_1, x_2, \ldots, x_t$. The compression function for CBC-MAC has the following form:

$$H_i = E_K(H_{i-1} \oplus x_i), \quad 1 \le i \le t.$$

Here $E_K(x)$ denotes the encryption of x using the k-bit key K with an n-bit block cipher E and $H_0 = IV$ is a fixed initial value, which is set equal to the all zero string. The MAC is then computed as $\mathrm{MAC}_K(x) = g(H_t)$, where g is the output transformation.

Bellare et al. [5] have provided a security proof for this scheme with g the identity mapping; their proof is based on the pseudo-randomness of the block cipher and requires that the inputs are of fixed length. They show that CBC-MAC is a pseudo-random function, which is in fact a stronger requirement than being a secure MAC. Most of these variants are vulnerable to an internal collision attack, which requires a single chosen text and about $2^{n/2}$ known texts with n the block length of the block cipher; for a 64-bit block cipher such as DES (►Data Encryption Standard) this corresponds to 2^{32} known texts. For $m < n$, an additional 2^{m-n} chosen texts are required, which makes the attack less realistic. It is important to note that for most schemes the gap between the lower bound (security proof) and upper bound (best known attack) is quite small. For several of these schemes shortcut key recovery attacks exist as well; lower bounds for the security against these attacks are not known for these schemes.

In practice one needs security for inputs of variable length, which can be achieved by using a different mapping g. These variants and attacks on them are discussed in more detail under ►CBCMAC and variants.

The most popular choice for g was the selection of the leftmost $m < n$ bits, $m = 32$ being a very popular variant for CBC-MAC based on DES. However, Knudsen has shown that a forgery attack on this scheme requires $2 \cdot 2^{(n-m)/2}$ chosen texts and two known texts [20].

A better solution for g is the encryption of the last block with a different key, which is known as EMAC. This solution was proposed by the RIPE Consortium in [30]; Petrank and Rackoff have provided a security proof in [24].

$$g(H_t) = E_{K'}(H_t) = E_{K'}(E_K(x_t \oplus H_{t-1})),$$

where K' is a key derived from K. Further optimizations which reduce the overhead due to padding are known as XCBC (three-key MAC) [7] and OMAC [18]. EMAC and OMAC are recommended for use with the ►Rijndael/AES.

An alternative for g which is popular for use with *DES* consists of replacing the processing of the last block by a two-key triple encryption (with keys $K_1 = K$ and K_2); this is commonly known as the ANSI retail MAC, since it first appeared in [1]:

$$g(H_t) = E_{K_1}(D_{K_2}(H_t)).$$

Here $D_K()$ denotes decryption with key K. This construction increases the strength against exhaustive key search, but it is not without its weaknesses [29]. A better alternative is MacDES [21].

XOR-MAC by Bellare et al. [4] is a randomized construction that allows for a full parallel evaluation; a fixed number of bits in every block (e.g., 32 bits) is used for a counter, which reduces the performance. It has the advantage that it is incremental: Small modifications to the input (and to the MAC) can be made at very low cost. Improved variants of XOR-MAC are XECB [14] and ►PMAC [8].

RMAC increases the security level of EMAC against an internal collision attack by modifying the key in the last encryption with a randomizer [19]. It has the disadvantage that its security proof requires resistance of the underlying block cipher against related key attacks. It was included in NIST's draft special publication [23] which has been withdrawn.

3GPP-MAC and RIPE-MAC are discussed in the item ►CBC-MAC and variants.

Based on Cryptographic Hash Functions

The availability of very fast dedicated ►hash functions (such as ►MD4 and ►MD5) has resulted in several proposals for MAC algorithms based on these functions. As it became clear that these hash functions are weaker than intended, they were replaced by ►RIPEMD-160 and by ►SHA-1.

The first proposed constructions were the secret prefix and secret suffix methods which can be described as follows: $\mathrm{MAC}_K(x) = h(K||x)$, $\mathrm{MAC}_K(x) = h(x||K)$.

However, these schemes have some security problems [29]. Consider the secret prefix method and assume that the hash function is an iterated hash function, where in each iteration n bits of the text (or the key) is processed. Then if one has the MAC of a text x such that the length of $K||x$ (including padding) is a multiple of n, then it is trivial to compute the value of $\mathrm{MAC}_K(K||y)$ from $\mathrm{MAC}_K(x)$ (this assumes that the output transformation is the identity function). Moreover, if the inputs x and x' have the same MAC value and if this is the result of an internal collision, the inputs $x||y$ and $x'||y$ will have the same MAC values.

Consider the secret suffix method and assume that the hash function is an iterated hash function. Here an attacker can try to find an internal collision for two texts x and x' ignoring the secret key K. Then if an attacker succeeds, the MACs of x and x' will be identical regardless of the value of K. It is important to notice here that the attacker can perform the computations off-line, that is, one needs no access to the MAC generation device during the first step.

A better proposal is the secret envelope method, or *envelope MAC* which can be described as $\mathrm{MAC}_K(x) = h(K_1||x||K_2)$. For this method, Bellare et al. provide a security proof in [2]. This method has been shown to be secure based on the assumption that the compression function of the hash function is a pseudorandom function (when keyed through the chaining variable). While this is an interesting result, it should be pointed out that the compression function of most hash functions has not been evaluated with respect to this property. For the particular envelope method of Internet RFC 1828 [22], it was shown by Preneel and van Oorschot [28] that an internal collision attack (which requires about $2^{n/2}$ known texts) does not only allow for a forgery but also a key recovery attack. This attack exploits the standard padding algorithms in modern hash functions and illustrates that one has to be very careful when transforming a hash function into a MAC algorithm.

To account for such pitfalls, the MDx-MAC has been proposed which extends the envelope method by also introducing secret key material into every iteration [27]. This makes the pseudo-randomness assumption more plausible. Moreover, it precludes the key recovery attack by extending the keys to complete blocks. MDx-MAC has been included in the ISO standard [17].

►HMAC is the most popular hash function variant, which uses a nested construction (with padded keys):

$$\mathrm{MAC}_K(x) = h\left(K_2 \,\|h\left(K_1\|\, x\right)\right).$$

The security of HMAC is guaranteed if the hash function is collision resistant for a secret value H_0, and if the compression function itself is a secure MAC for one block (with the secret key in the H_i input and the text in the x_i input) [3].

While these assumptions are weaker than for the secret envelope method, it still requires further validation for existing hash functions. HMAC is used for providing message authentication in the Internet Protocol ►Ipsec, TLS (►Transport Layer Security) and has been included in an ISO [17] and FIPS standard [13].

Two-Track-MAC is another construction which exploits the presence of two parallel internal trails in RIPEMD-160 [11].

Dedicated MAC Algorithms

The Message Authentication Algorithm (►MAA) was designed in 1983 by Davies and Clayden [9, 10]. In 1987, it became a part of the ISO 8731 banking standard [16]. Several weaknesses of MAA have been identified by Preneel et al. [26], but none of these form an immediate threat to existing applications. However, it would be advisable to check for the known classes of weak keys [26] and to change the key frequently.

A cryptanalysis of an early MAC algorithm can be found in [25]. A few MAC algorithms in use have not been published, such as the S.W.I.F.T. authenticator and the Swedish algorithm Data Seal. Several proprietary MAC algorithms that can process only short messages algorithm have been leaked: This includes the Sky Videocrypt system of British Sky Broadcasting (which was leaked out in 1993 and replaced), the COMP128 algorithm which was used by certain GSM operators as A3/A8 algorithm (a fix for its weaknesses is proposed in [15]; it has been upgraded to COMP128–2 and COMP128–3) and the function used in the SecureID token (for which an analysis can be found in [6]). Proprietary algorithms which have not been leaked include Telepass 1 (from Bull) and the DECT Standard Authentication Algorithm (DSAA) for cordless telephony.

Recommended Reading

1. ANSI X9.19 (1986) Financial institution retail message authentication. American Bankers Association, Washington, DC
2. Bellare M, Canetti R, Krawczyk H (1996) Pseudorandom functions revisited: the cascade construction and its concrete security. In: Proceedings of the 37th annual symposium on the foundations of computer science. IEEE, Washington, DC, pp 514–523. Full version http://www.cs.ucsd.edu/users/mihir/papers/cascade.html
3. Bellare M, Canetti R, Krawczyk H (1996) Keying hash functions for message authentication. In: Koblitz N (ed) Advances in cryptology – CRYPTO'96. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 1–15. Full version http://www.cs.ucsd.edu/users/mihir/papers/hmac.html

4. Bellare M, Guérin R, Rogaway P (1995) XOR MACs: new methods for message authentication using block ciphers. In: Coppersmith D (ed) Advances in cryptology – CRYPTO'95. Lecture notes in computer science, vol 963. Springer, Berlin, pp 15–28
5. Bellare M, Kilian J, Rogaway P (2000) The security of cipher block chaining. J Comput Syst Sci 61(3):362–399. Earlier version in Desmedt Y (ed) (1994) Advances in cryptology – CRYPTO'94. Lecture notes in computer science, vol 839. Springer, Berlin, pp 341–358
6. Biryukov A, Lano J, Preneel B (2004) Cryptanalysis of the alleged SecurID hash function. In: Matsui M, Zuccherato RJ (eds) Selected areas in cryptography 2003. Lecture notes in computer science, vol 3006. Springer, Berlin, pp 130–144
7. Black J, Rogaway P (2000) CBC-MACs for arbitrary length messages. In: Bellare M (ed) Advances in cryptology – CRYPTO 2000. Lecture notes in computer science, vol 1880. Springer, Berlin, pp 197–215
8. Black J, Rogaway P (2002) A block-cipher mode of operation for parallelizable message authentication. In: Knudsen L (ed) Advances in cryptology – EUROCRYPT 2002. Lecture notes in computer science, vol 2332. Springer, Berlin, pp 384–397
9. Davies D (1985) A message authenticator algorithm suitable for a mainframe computer. In: Blakley GR, Chaum D (eds) Advances in cryptology – CRYPTO'84. Lecture notes in computer science, vol 196. Springer, Berlin, pp 393–400
10. Davies D, Price WL (1989) Security for computer networks: an introduction to data security in teleprocessing and electronic funds transfer, 2nd edn. Wiley, New York
11. den Boer B, Van Rompay B, Preneel B, Vandewalle J (2001) New (two-track-)MAC based on the two trails of RIPEMD. In: Vaudenay S, Youssef AM (eds) Selected areas in cryptography 2001. Lecture notes in computer science, vol 2259. Springer, Berlin, pp 314–324
12. FIPS 180-1 (1995) Secure hash standard. NIST, US Department of Commerce, Washington, DC
13. FIPS 198 (2002) The keyed-hash message authentication code (HMAC). NIST, US Department of Commerce, Washington, DC
14. Gligor V, Donescu P (2002) Fast encryption and authentication: XCBC encryption and ECB authentication modes. In: Matsui M (ed) Fast software encryption. Lecture notes in computer science, vol 2355. Springer, Berlin, pp 92–108
15. Handschuh H, Paillier P (2000) Reducing the collision probability of alleged Comp128. In: Quisquater J-J, Schneier B (eds) Smart card research and applications. Lecture notes in computer science, vol 1820. Springer, Berlin, pp 380–385
16. ISO 8731 (1992) Banking – approved algorithms for message authentication, Part 1: DEA, 1987, Part 2: message authentication algorithm (MAA). ISO, Geneva
17. ISO/IEC 9797 (2002) Information technology – security techniques – message authentication codes (MACs). Part 1: mechanisms using a block cipher, 1999. Part 2: mechanisms using a dedicated hash-function, 2002. Standards South Africa, Pretoria
18. Iwata T, Kurosawa K (2003) OMAC: one key CBC MAC. In: Johansson T (ed) Fast software encryption. Lecture notes in computer science, vol 2887. Springer, Berlin, pp 129–153
19. Jaulmes E, Joux A, Valette F (2002) On the security of randomized CBC-MAC beyond the birthday paradox limit: a new construction. In: Daemen J, Rijmen V (eds) Fast software encryption. Lecture notes in computer science, vol 2365. Springer, Berlin, pp 237–251
20. Knudsen L (1997) Chosen-text attack on CBC-MAC. Electron Lett 33(1):48–49
21. Knudsen L, Preneel B (1998) MacDES: MAC algorithm based on DES. Electron Lett 34(9):871–873
22. Metzger P, Simpson W (1995) IP authentication using keyed MD5. Internet Request for Comments 1828
23. NIST Special Publication 800-38B (2002) Draft recommendation for block cipher modes of operation: the RMAC authentication mode
24. Petrank E, Rackoff C (2000) CBC MAC for real-time data sources. J Cryptol 13(3):315–338
25. Preneel B, Bosselaers A, Govaerts R, Vandewalle J (1990) Cryptanalysis of a fast cryptographic checksum algorithm. Comput Secur 9(3):257–262
26. Preneel B, Rijmen V, van Oorschot PC (1997) A security analysis of the message authenticator algorithm (MAA). Eur Trans Telecommun 8(5):455–470
27. Preneel B, van Oorschot PC (1995) MDx-MAC and building fast MACs from hash functions. In: Coppersmith D (ed) Advances in cryptology – CRYPTO'95. Lecture notes in computer science, vol 963. Springer, Berlin, pp 1–14
28. Preneel B, van Oorschot PC (1996) On the security of two MAC algorithms. In: Maurer U (ed) Advances in cryptology – EUROCRYPT'96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 19–32
29. Preneel B, van Oorschot PC (1999) On the security of iterated message authentication codes. IEEE Trans Info Theory IT-45(1):188–199
30. RIPE (1995) Integrity primitives for secure information systems. In: Bosselaers A, Preneel B (eds) Final report of RACE integrity primitives evaluation (RIPE-RACE1040). Lecture notes in computer science, vol 1007. Springer, Berlin
31. Simmons GJ (1991) How to insure that data acquired to verify treaty compliance are trustworthy. In: Simmons GJ (ed) Contemporary cryptology: the science of information integrity. IEEE Press, Piscataway, pp 615–630

Machine Readable Travel Document Security

►Passport Security

Macrodata Disclosure Limitation

►Macrodata Protection

Macrodata Disclosure Protection

►Macrodata Protection

Macrodata Protection

Sara Foresti
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Synonyms

Macrodata disclosure limitation; Macrodata disclosure protection

Related Concepts

►Microdata Protection

Definition

Macrodata protection techniques protect the confidentiality of sensitive information when releasing aggregated values.

Background

Governmental, public, and private organizations are more and more frequently required to make data available for external release in a secure way. Indeed, if on the one hand there is a need to disseminate some data, there is on the other hand an equally strong need to protect those data that, for various reasons, should not be disclosed. Traditionally, data collected from surveys are released in tabular form (*macrodata*). Macrodata are aggregate information (statistics) on users or organizations usually presented as two-dimensional tables. Many techniques have been developed for protecting data released publicly or semi-publicly from improper disclosure. These techniques depend on the method in which such data are released. Macrodata protection techniques are specifically targeted to meeting the protection needs of macrodata and are typically based on the selective obfuscation of sensitive cells.

Theory

Macrodata represent estimated values of *statistical characteristics* concerning a given population. A statistical characteristic is a measure that summarizes the values of one or more *properties/attributes* (*variables*, in statistical terminology) of *respondents* (i.e., individuals to whom data refer). An example of a statistical characteristic can be the average age of people living in each continent. Macrodata can be represented as tables, where each cell of the table is the aggregate value of a quantity over the considered properties.

Macrodata tables can be classified into the following three groups (types of tables).

a

Number of respondents with a disease					
	Hypertension	**Obesity**	**Chest Pain**	**Short Breath**	**Total**
M	1	2	2	1	6
F	1	2	0	2	5
Total	2	4	2	3	11

b

Percentage of respondents with a disease					
	Hypertension	**Obesity**	**Chest Pain**	**Short Breath**	**Total**
M	9.1	18.2	18.2	9.1	54.6
F	9.1	18.2	0	18.2	45.4
Total	18.2	36.4	18.2	27.2	100

c

Average number of days spent in the hospital by respondents with a disease					
	Hypertension	**Obesity**	**Chest Pain**	**Short Breath**	**Total**
M	2	8.5	23.5	3	37
F	3	30.5	0	5	38.5
Total	5	39	23.5	8	75.5

Macrodata Protection. Fig. 1 An example of count (**a**), frequency (**b**), and magnitude (**c**) macrodata tables

- *Count tables.* Each cell of the table contains the *number of respondents* that have the same value over all attributes of analysis associated with the table. For instance, the table in Fig. 1a contains the number of males and females for each given disease.
- *Frequency tables.* Each cell of the table contains the *percentage of respondents*, evaluated over the total population, that have the same value over all the attributes of analysis associated with the table. For instance, the macrodata table in Fig. 1b contains the percentage of males and females for each given disease.
- *Magnitude tables.* Each cell of the table contains *an aggregate value of a quantity of interest* over all attributes of analysis associated with the table. For instance, the macrodata table in Fig. 1c contains the average number of days that males and females have spent in the hospital for each given disease.

Several macrodata protection techniques have been developed to guarantee the *confidentiality* of the data, that is, the assurance that information about individual respondents cannot be derived (or approximated) from the released macrodata. Typically, macrodata protection techniques work in two steps: (1) they first *discover sensitive cells*, that is, cells that can be easily associated with a specific respondent, and then (2) *protect* these cells.

Discovering Sensitive Cells

The strategies for discovering sensitive cells vary depending on the type of macrodata (count and frequency tables

versus magnitude tables). For count and frequency tables, the most important strategy used to detect sensitive cells is the *threshold rule*, according to which a cell is sensitive if the number of respondents is less than a given threshold [1]. As an example, consider the macrodata table Fig. 1a and suppose that the threshold is 2. The first cell and the last cell in the first tuple, and the first cell and the third cell in the second tuple are sensitive because their value is below the given threshold.

For magnitude macrodata, there are many rules that can be used to detect sensitive cells [1]. For instance, the *(n,k)-rule* states that a cell is sensitive if less than n respondents contribute to more than k% of the total cell value. As an example, consider the macrodata table in Fig. 1c and suppose to apply the (1, 50)-rule. A cell is sensitive if one respondent contributes to more than 50% of its value. The first cell and the last cell in the first tuple as well as the first cell in the second tuple are sensitive since, as visible in the associated table in Fig. 1a, there is only one male and one female with hypertension and one male with short breath and therefore their contribution to these cells is 100%. Other similar rules are the *p-percent* rule and the *pq-rule*. The *p-percent* rule states that a cell is sensitive if the total value t of the cell, minus the sum of the largest reported value v_1 and the second largest reported value v_2, is less than $(p/100)\cdot v_1$. Intuitively, this rule means that a user can closely estimate the reported value of some respondents. In the *pq-rule*, q represents how closely respondents can estimate another respondent's value ($p < q < 100$).

Protecting Sensitive Cells

Sensitive cells can be protected by applying several strategies, such as *cell suppression*, *rounding*, *roll up categories*, *sampling*, *controlled tabular adjustment function* (CTA), and *confidential edit* [1]. Cell suppression consists in protecting sensitive cells by removing their values. These suppressions are called *primary suppressions*. However, if the marginal totals of the table are published, it may be possible to determine the value of the suppressed cell or restrict the interval of possible values. If the size of such an interval is small, the suppressed cell can be estimated rather precisely. To avoid such inferences, additional cells may need to be suppressed (*secondary suppression*) to guarantee that the intervals are sufficiently large. Linear programming techniques have been proposed to minimize the total number of cells to be suppressed. Such techniques are suitable for small tables, although they are usually not applicable to more complex structures. *Rounding* consists in choosing a base number and in modifying the original value of sensitive cells by rounding it up or down to a near multiple of the base number. *Roll up* categories reduces the size of the table: instead of releasing a table with N tuples and M columns, a less detailed table (e.g., a table with $N-1$ tuples and $M-1$ columns) is released. *Sampling* means that the table is obtained with a sample survey rather than a census. The *CTA* technique is based on the selective adjustment of cell values. In other words, the value of sensitive cells is replaced by a safe value, that is, a value that is not sensitive w.r.t. the rule chosen to detect sensitive cells, and then a linear programming technique is used to adjust the values of the nonsensitive cells, such that the sum of the values in each column/row in the table coincides with the marginal total of the column/row. *Confidential edit* modifies the original data used to compute aggregate values and recomputes the macrodata table on the obtained collection of data. This techniques selects a sample of the records in the dataset, finds a match for the selected records (i.e., a set of records with the same values on a specific set of attributes) in other geographical regions, and swaps the attributes of the matching records.

Applications

Macrodata protection techniques can be applied whenever aggregate information, computed starting from collected data, containing some sensitive information, is released. As an example, consider the case of a hospital releasing the number of patients aggregated by their disease and city where they live. When releasing statistics over the hospital activity, the hospital must guarantee that possible sensitive information, such as the association between individual patients and their illnesses, is not released. As another example, statistical agencies, when releasing aggregated financial data, may require a sanitization process to protect information considered sensitive, such as the correlation between individuals and their financial status.

Recommended Reading

1. Federal Committee on Statistical Methodology (1994) Report on statistical disclosure limitation methodology. Statistical policy Working Paper 22, Washington, DC, May 1994

Maliciously Modified Set of Administrative Tools

►Rootkits

Malware

►Spyware

Malware Behavior Clustering

Engin Kirda
Institut Eurecom, Sophia Antipolis, France

Related Concepts

►Behavioral-based Detection; ►Dynamic Analysis; ►Intrusion Detection; ►Malware Analysis; ►Malware Classification; ►Malware Detection; ►Malware Removal

Definition

One of the major threats on the Internet today is malicious software, often referred to as malware. Malware comes in a wide range of forms and variations, such as viruses, worms, botnets, rootkits, Trojan horses, and denial-of-service tools. To spread, malware exploits software vulnerabilities in browsers and operating systems, or uses social engineering techniques to trick users into running the malicious code. An anti-malware company typically receives thousands of new malware samples every day. These samples are submitted by users who have found suspicious code on their systems, by other anti-malware companies that share their samples, and by organizations that collect malware. The ability to automatically and effectively cluster analyzed malware samples into families with similar behavioral characteristics is referred to as *Malware Behavioral Clustering*.

Theory

Because of the growing need for automated techniques to examine malware, dynamic malware analysis tools such as CWSandbox, Norman Sandbox, and ANUBIS have increased in popularity. These systems execute the malware sample in a controlled environment and monitor its actions. Based on the execution traces, reports are generated that aim to support an analyst in reaching a conclusion about the type and severity of the threat imposed by a malware sample.

For example, when observing a sample that modifies the autorun registry entry and opens a connection to a notorious IRC server (which is often used for botnet command and control), the analyst can quickly conclude that the sample is an IRC bot. Even when the analyst is not able to reach a detailed conclusion about a sample, the automated analysis is beneficial to help separate interesting samples from those that are less relevant.

While automating the analysis of the behavior of a single malware sample is a first step, it is not sufficient. This is because the analyst is now facing thousands of reports every day that need to be examined. Thus, there is a need to prioritize these reports and guide an analyst in the selection of those samples that require most attention. One approach to process reports is to cluster them into sets of malware that exhibit similar behavior. The ability to automatically and effectively cluster analyzed malware samples into families with similar characteristics is beneficial for the following reasons: First, every time a new malware sample is found in the wild, an analyst can quickly determine whether it is a new malware instance or a variant of a well-known family. Moreover, given sets of malware samples that belong to different malware families, it becomes significantly easier to derive generalized signatures, implement removal procedures, and create new mitigation strategies that work for a whole class of programs.

The recent advances in the field of automated malware analysis (e.g., [1–3]) have created a rising interest in the automatic grouping of the analysis results (and reports) that are created. For this purpose, researchers have proposed supervised as well as unsupervised machine learning techniques. Because it is crucial that these techniques can process a large number of samples, their scalability is one of the decisive properties.

At the core of every system that aims to find malware families is the notion of similarity. Therefore, these systems need to solve two problems. First, they need to find a suitable representation of a malware sample. Second, based on these representations, they need to compute a distance between two samples. In the literature, content-based and behavior-based comparison approaches have been proposed.

Content-Based Analysis. The first attempts to cluster malware samples were based on static analysis of the malware samples. For example, in [4], the author proposes an automated virus classification system that works by first disassembling the binaries, and subsequently, comparing their basic code blocks. Other researchers have proposed to represent a malware program as a hex-dump of its code segment, building a classification system on top of this (e.g.,[5]). In [6], Dullien and Rolles propose a system for comparing executables based on their control flow graph.

All content-based analysis approaches share the problem that they need to disassemble the binary. This is often difficult or even impossible, given that malware is frequently obfuscated and packed. Also, it is possible to write semantically equivalent programs that have large difference in their code. Thus, it is possible for malware authors to thwart content-based similarity calculations.

Behavior-Based Analysis. In 2008, Holz et al. [7] presented a system that classifies unknown malware samples

based on their behavior. A significant limitation is that the system requires supervised learning, using a virus scanner for labeling the training set. Lee et al. developed a system for classifying malware samples that relies on system calls for comparing executables [8]. The scalability of the technique is limited; the system required several hours to cluster a set of several hundred samples. Also, the tight focus on system calls implies that the collected profiles do not abstract the observed behavior.

Bailey et al. [9] have proposed a system that abstracts from system call traces and clusters samples that exhibit similar behavior.

Leita et al. [10] suggest classifying malware based on the epsilon-gamma-pi-mu model. In this model, additional information on how the malware is originally installed on the target system is considered for classification. This can include information on the exploit and exploit payload used to install the malware dropper, and on the way the dropper in turn downloads and installs the malware.

In [11], Bayer et al. proposed an approach for clustering large collections of malware samples. The goal is to find a partitioning of a given set of malicious programs so that subsets exhibit similar behavior. The system begins by analyzing each sample in a dynamic analysis environment that has been enhanced with taint tracking and additional network analysis. Then, behavioral profiles are extracted by abstracting system calls, their dependences, and the network activities to a generalized representation consisting of OS objects and OS operations. These profiles serve as the input to the clustering algorithm, which requires less than a quadratic amount of distance computations. This is important to handle large data sets that are commonly encountered in the real world.

Recommended Reading

1. Brumley D, Hartwig C, Liang Z, Newsome J, Poosankam P, Song D, Yin H (2007) Automatically identifying trigger-based behavior in Malware. In: Lee W et al. (eds) Botnet analysis and defense
2. Egele M, Kruegel C, Kirda E, Yin H, Song D (2007) Dynamic spyware analysis. In: USENIX annual technical conference, Santa Clara, 2007
3. Moser A, Kruegel C, Kirda E (2007) Exploring multiple execution paths for malware analysis. In: IEEE symposium on security and privacy, Berkeley, 2007
4. Gheorghescu M (2005) An automated virus classification system. In: Virus bulletin conference, Dublin, 2005
5. Kolter JZ, Maloof MA (2006) Learning to detect and classify malicious executables in the wild. J Mach Learn Res 7:2721–2744
6. Dullien T, Rolles R (2005) Graph-based comparison of executable objects. In: Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC), Rennes, 2005
7. Holz T, Willems C, Rieck K, Duessel P, Laskov P (2008) Learning and classification of malware behavior. In: Fifth conference on detection of intrusions and malware and vulnerability assessment (DIMVA 08), Paris, 2008
8. Lee T, Mody JJ (2006) Behavioral classification. In: EICAR conference, Hamburg, 2006
9. Bailey M, Oberheide J, Andersen J, Mao ZM, Jahanian F, Nazario J (2007) Automated classification and analysis of internet malware. In: 10th international symposium on recent advances in intrusion detection (RAID'07), Gold Coast, 2007
10. Leita C, Dacier M (2008) SGNET: a worldwide deployable framework to support the analysis of malware threat models. In: European dependable computing conference, Kaunas, 2008
11. Bayer U, Milani P, Hlauschek C, Kruegel C, Kirda E (2009) Scalable, behavior-based malware clustering. In: 16th annual network and distributed system security symposium (NDSS 2009), San Diego, 2009

Malware Detection

Stefan Katzenbeisser[1], Johannes Kinder[2], Helmut Veith[3]

[1]Security Engineering Group, Technische Universität Darmstadt, Darmstadt, Germany

[2]Formal Methods in Systems Engineering, Technische Universität Darmstadt, Darmstadt, Germany

[3]Institute of Information Systems, Technische Universität Wien, Wien, Austria

Synonyms

Antivirus; Virus scanner

Related Concepts

►Dynamic Analysis; ►Intrusion Detection; ►Static Analysis

Definition

Malware is malicious software that was intentionally developed to infiltrate or damage a computer system without consent of the owner. This includes, among others, viruses, worms, and Trojan horses. Malware detection refers to the process of detecting the presence of malware on a host system or of distinguishing whether a specific program is malicious or benign.

Background

Malware is one of the most serious security threats and spreads autonomously through vulnerabilities or carelessness of users. In order to protect a computer from infection or remove malware from a compromised computer system, it is essential to accurately detect malware. As a

consequence of Rice's Theorem (and shown by Cohen for the case of computer viruses [1]), determining whether a given program contains malicious functionality is generally undecidable. Thus, malware detection focuses on practical methods that reliably and efficiently detect certain classes of malware. Besides theoretical limitations, malware detection faces two main practical challenges: An ever increasing development and distribution speed of malware, and sophisticated methods for evading detection such as poly- and metamorphism.

Rapid Releases of Malware Variants

The Internet has decreased the development time and increased the distribution speed of new malware by several orders of magnitude. While early computer viruses and worms were often developed for the thrill of vandalism by adolescent computer enthusiasts, the new millennium has seen an increase in professional malware authors who command vast *botnets* of infected computers in order to harvest sensitive data or perform distributed denial of service attacks. This development has spurred the rapid release of modified and improved malware variants in short succession. The time it takes from the release of a new malware until it can be reliably detected by common antivirus tools, the *window of vulnerability*, is often enough to infect a large number of computer systems despite antivirus protection. One of the declared goals of research in malware detection is thus to shorten or eliminate this window of vulnerability.

Poly- and Metamorphic Malware

Polymorphic and metamorphic malware is particularly difficult to detect [2]. *Polymorphic malware* encrypts its malicious executable code (the *virus body*) and attaches a small decryption engine, which decrypts the code only at runtime. Whenever the malware creates copies of itself, it generates a fresh key and re-encrypts its executable body with the new key. Furthermore, it obfuscates the decryption engine, which cannot be encrypted itself, by inserting no-operation instructions or reordering the control flow without altering its semantics. Each malware instance is thus syntactically different on disk, but at runtime decrypted into the same original virus body. A trivial but widely used special case of polymorphism is the use of *executable packers*, tools that compress and/or encrypt an executable file with a static key. By choosing different keys or adding another layer of packing around an already packed file, malware authors can manually create syntactically different instances of the same malware.

Metamorphic malware achieves syntactically different instances both on disk and at runtime by completely rewriting itself upon distribution into a functionally equivalent program. The malware parses its own code, randomly applies code transformations that do not change the program semantics (obfuscations), and writes the transformed code into a new executable.

Theory and Applications

Techniques for malware detection are widely deployed in commercial anti-virus products installed on end-user clients. Anti-virus software typically includes a background service that scans files on access, and an on-demand scanner, which is invoked in regular intervals. Speed is critical for these end-user products, since a lengthy analysis of an executable on access delays the startup of programs and impairs the overall responsiveness of the computer system. Other malware detectors are implemented as part of intrusion detection systems that monitor activity on the network or on a specific host. Computationally expensive malware detection methods can be used on dedicated systems for malware analysis and forensics, e.g., within anti-virus research labs or as part of a security audit [3, 4].

Signature Matching

The fastest and most common detection method is the use of static malware signatures. In the most simple form, they are binary sequences of code or data that appear in a particular malware instance. Other forms of static malware signatures are hash values of code blocks starting at a given offset in an executable, or binary sequences including wildcards to allow some syntactic variation. Signatures have to be generated from sufficiently long code sequences to minimize false positives (i.e., false detections of malware in benign programs). To detect a certain malware instance, anti-virus software simply checks for the presence of its signature in a given program (*scanning*). Commercial anti-virus products maintain large databases of these signatures, and scan every file for all signatures of viruses and worms they know of. These databases constantly have to be kept up to date and steadily increase in size due to the emergence of new malware. Furthermore, a growing size of the signature database implies an increased chance of false positives. Signature matching therefore reaches its limits in the light of rapidly rising numbers of malware and the inherent window of vulnerability between emergence of a new malware and the next signature release.

Polymorphic malware is difficult to detect through signature matching, since the syntactic image on disk changes with each replicated instance. For simple polymorphic malware whose decryption routine does not vary significantly between malware instances, signatures for the

decryption routine can be generated. Once the polymorphic malware is run (e.g., as part of *dynamic analysis*, see below) and has decrypted the virus body, signature matching can be applied to detect the malware in memory.

Metamorphic malware, on the other hand, completely eludes classic signature matching. A carefully written metamorphic engine generates enough entropy such that no sufficiently long common sequence of instructions is shared between different instances of the same malware. However, a sloppy implementation of the metamorphic engine by the malware author can introduce commonalities into all instances, such as fixed opcodes with only varying registers. Such weaknesses in the obfuscation scheme can be exploited by scanning for wildcard signatures. In general, however, more advanced detection methods are required.

Research on malware detection thus focuses on approaches for *proactive detection*: Malware that is sufficiently similar in structure or behavior to malware found and analyzed in the past should be detected without requiring new syntactic signatures. Most vendors of antivirus software implement a form of proactive detection by augmenting their signature-based detection scheme with *heuristics*. They search for several indicators of malicious code, such as the presence of certain system calls or anomalies in the file header, assign scores to each indicator, and classify an executable as malicious if its total score exceeds a certain threshold. Another strategy to detect multiple variants of the same malware as well as polymorphic and metamorphic malware is *malware normalization*, which attempts to transform obfuscated binary code into a canonical form that can be matched against traditional static signature databases [5, 6].

Besides augmenting existing signature-based detection methods, there are two general approaches to detect rapidly changing malware: *Static analysis* classifies the potential behavior of a program as malicious or benign without executing it; *dynamic analysis* runs an executable for a certain time in an appropriately protected virtual environment and monitors its behavior at runtime.

Static Analysis

Malware detection by static analysis relies on approximating (abstracting) the semantics of a piece of code, and determining whether it may perform malicious actions. By doing this, static analysis explores *all* possible execution paths of the program. The fundamental difference to static signature matching is that it operates on the semantics, i.e., the potential behavior of malicious code, instead of its syntactic representation. To this end, semantic malware signatures specify "malicious behavior" in an appropriate language. This allows to detect variants of the same malware with a single semantic signature (and thus eliminate the window of vulnerability), as long as they contain the same core malicious behavior. Furthermore, since common transformations applied by metamorphic malware do not change the program semantics, semantic malware signatures should be robust against metamorphism.

However, implementation of a practical malware detector based on static analysis is challenging. An analysis powerful enough to directly handle self-modifying code as in packed or polymorphic malware is considered prohibitively expensive. Existing approaches thus usually focus on analyzing the unencrypted virus body and make use of separate tools to obtain it [7–9]. Several steps have to be performed: First, for polymorphic code, the original virus body has to be exposed either by static decryption or dynamic analysis. The resulting binary is passed to a disassembler, which converts binary machine code into assembly instructions and provides a control flow graph to the static analysis component.

Note that in order to reliably detect malware, all these steps must be correctly performed. This yields several practical problems, which can be exploited by malware authors to evade static detection. The preparation step to expose the virus body must succeed. For well-known packers or simple encryption schemes, unpacking is an easy and automatic process. More intricate polymorphic schemes, however, require the use of emulation techniques from dynamic analysis. Once the code is available, further processing of the code faces the classic disassembly problem, complicated by the fact that malware is often deliberately obfuscated.

Several approaches to static analysis–based malware detection with different abstractions of behavior and specification mechanisms have been proposed. Very coarse abstractions of executable code only consider system calls; specifications for malicious code then describe system call sequences known to be characteristic to malware. Finer-grained abstractions analyze the control flow graph on the level of individual machine instructions. On this level, it is possible to directly specify patterns of instructions and relations between their operands. Among other patterns, this allows to detect possible data flow between return values and parameters of system calls [7, 8] and yields more fine-grained malware specifications.

Dynamic Analysis

Malware detection based on dynamic analysis monitors the behavior of potentially malicious processes at runtime

and inspects their interactions with the system for suspicious activity. The processes can either be observed live while running on a real system [10] (as in *Host-Based Intrusion Detection Systems*) or in an isolated virtual machine or sandbox. Detection engines implemented in on-demand scanners of commercial anti-virus products make use of relatively lightweight *sandboxes*, which provide a partial environment (e.g., function stubs for system calls) and emulate the executable for a short amount of time [11]. Although fast, the sandbox can only give a rough estimate of the actions the program would perform when run freely on a system. For detection of polymorphic malware, lightweight sandboxes have proven extremely effective, since only the initial decryption routine needs to be emulated for the original virus body to appear in the sandbox memory. Once the virus body is plainly visible in the sandbox, classic signature matching can be used for detection. For detecting malicious activity of unknown malware, a sandbox can also be used to record traces of system calls, which are then matched against known malware behavior.

Virtual machines emulate a full system including processor and devices, and host a full operating system installation [12]. Their advantage is that executables will usually exhibit the same behavior as on a real system, since all system state is correctly emulated and no system calls are abstracted by stubs.

Dynamic analysis is robust against polymorphic and metamorphic malware, including low-level obfuscations that can thwart disassembly and thus static detection, but it faces specific challenges of its own. Malware can attempt to detect whether it is run within a sandbox or virtual machine or actively attack the emulator. Several anti-emulation techniques exist, which check for certain low-level processor features (e.g., undocumented instructions) or timings. Once an executable has determined that it is being emulated, it can terminate execution without performing any malicious actions.

Another limitation of purely dynamic approaches to malware detection is their only partial observation of the program behavior. Only a single execution is observed for a limited time; if malicious activity is triggered randomly or only under certain conditions, the emulator might not witness the malicious behavior and falsely consider the executable benign. Some file-infecting viruses specifically target this weakness by hooking their malicious code into the exit routines of a program, or placing it randomly inside benign procedures (Entry Point Obfuscation).

The chance of missing malicious behavior that depends on certain conditions can be reduced by using methods from dynamic test case generation that combine features from static and dynamic analysis [13]. While monitoring the program execution, an emulator also executes the program symbolically to relate program inputs (e.g., return values of some system calls) to branch conditions. In successive runs of the executable, the inputs are then modified to drive execution into different branches. This way, the coverage of dynamic analysis can be increased.

Recommended Reading

1. Cohen F (1987) Computer viruses: theory and experiments. Comput Secur 6(1):22–35
2. Ször P, Ferrie P (2001) Hunting for metamorphic. In: Proceedings of 2001 Virus Bulletin Conference, Virus Bulletin, pp 123–144
3. Christodorescu M, Jha S, Maughan D, Song, Wang C (eds) Advances in information security. Malware detection, vol 27. Springer, New York, p 311
4. Ször P (2005) The art of computer virus research and defense. Addison Wesley, Upper Saddle River, p 713
5. Christodorescu M, Jha S, Kinder J, Katzenbeisser S, Veith H (2007) Software transformations to improve malware detection. J Comput Virol 3(4):253–265
6. Lakhotia A, Mohammed M (2004) Imposing order on program statements to assist anti-virus scanners. In: 11th Working Conference on Reverse Engineering Proceedings (WCRE 2004), Delft, 8–12 November 2004. IEEE Computer Society Press, Los Alamitos, pp 161–170
7. Preda MD, Christodorescu M, Jha S, Debray SK (2007) A semantics based approach to malware detection. In: Conference record of POPL 2007: the 34th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, French Riviera, 17–19 January 2007. ACM Press, New York, pp 377–388
8. Kinder J, Katzenbeisser S, Schallhart C, Veith H (2010) Proactive detection of computer worms using model checking. IEEE Trans Depend Secure Comput 7(4):424–438
9. Kolbitsch C, Comparetti PM, Kruegel C, Kirda E, Zhou X, Wang X (2009) Effective and efficient malware detection at the end host. In: Proceedings of the 18th Usenix Security Symposium (USENIX'09), Montreal, 10–14 August 2009. USENIX Association, Berkeley
10. Holz T, Freiling F, Willems C (2007) Toward automated dynamic malware analysis using CWSandbox. In: SP'07: Proceedings of the 2007 IEEE Symposium on Security and Privacy, Berkeley, 20–23 May 2007. IEEE Computer Society Press, Los Alamitos, pp 32–39
11. Natvig K (2002) Sandbox II: Internet. In: Proceedings of 2002 Virus Bulletin Conference, Virus Bulletin
12. Bayer U, Moser A, Krügel C, Kirda E (2006) Dynamic analysis of malicious code. J Comput Virol 2(1):67–77
13. Moser A, Kruegel C, Kirda E (2007) Exploring multiple execution paths for malware analysis. In: SP'07: Proceedings of the 2007 IEEE Symposium on Security and Privacy, Berkeley, 20–23 May 2007. IEEE Computer Society Press, Los Alamitos, pp 231–245

Mandatory Access Control

Shambhu Upadhyaya
Department of Computer Science and Engineering, University at Buffalo The State University of New York, Buffalo, NY, USA

Synonyms

Rule-based access control

Related Concepts

►Access Control from an OS Security Perspective; ►Access Control Policies, Models, and Mechanisms; ►Access Control Lists; ►Bell-LaPadula Confidentiality Model; ►Mandatory Access Control; ►Mandatory Access Control; ►Reference Monitor; ►Role-based; ►Trusted Operating System

Definition

Mandatory access control (MAC) is a security policy that encapsulates confidentiality of an object in the realm of computer security. This policy goes beyond the control of the owner of an object and is defined as a control policy set up by a central authority who can determine what information can be accessed by whom [1]. This is in contrast with discretionary access control (DAC) where the owner is empowered with the setting of access control on an object. More formally, MAC is a "means of restricting access to objects based on the sensitivity (as represented by a security label) of the information contained in the objects and the formal authorization (i.e., clearance, formal access approvals, and need-to-know) of subjects to access information of such sensitivity" [2].

Background

In an operating system, there are primarily three entities, namely, users, subjects, and objects. A user corresponds to a user account, a subject is a process or program in execution and an object represents a key resource such as files or sockets. When a process issues a request to access a file, certain restrictions are applied according to some well-defined control policies. Discretionary access control (DAC) and mandatory access control (MAC) policies provide the basic protection to system resources and user data. The Trusted Computer System Evaluation Criteria (TCSEC) [3], commonly referred to as the US *Orange Book* or the Department of Defense (DoD) directive 5200.28, developed the MAC policy out of the need to protect classified information in the military. MAC is closely related to the multilevel security (MLS) notion [2] built around the Bell–La Padula confidentiality model [4], which gives a formal description of secure information flow across multiple levels of classification in a military system.

Some of the early implementations of MAC are Honeywell's SCOMP [5], Boeing's MLS LAN [6], and NSA's Blacker [7] which are based on the MLS notion. The US *Orange Book*, published in 1972 and subsequently revised, has been promoting strong security measures for all defense computer systems resulting in several MAC implementations such as the USAF SACDIN [8], Novell's AppArmor [9], and NSA's SELinux [10]. With the growing use of computers, wireless networks, and mobile use, there are efforts that have departed from the original MLS notion to develop models that meet the advanced requirements. The location-based MAC [11] and dynamic MAC [12] are some examples. Though powerful, the traditional MAC implementations have been viewed as complex and difficult to configure for end users. A noteworthy development in this regard is the usable MAC model, namely, the usable mandatory integrity protection (UMIP) model [13].

Theory

The theory of MAC is drawn from the Bell–La Padula model [4]. While MAC is described by many authors, a simplistic description provided by Bishop [14] and Ray and Kumar [11] is adopted here. The Bell–La Padula model is defined in terms of a security structure $(\mathbf{L}, \leq)$ where $\mathbf{L}$ represents the set of security levels, and $\leq$ is a partial ordering relation defined on these levels. The partial ordering relation is one that is transitive and antisymmetric. Under this relation, $L_i \leq L_j$ signifies that security level L_i is dominated by security level L_j. Here, element L_i is said to be dominated and element L_j is dominating. Two elements L_i and L_j are not comparable if neither $L_i \leq L_j$ nor $L_j \leq L_i$.

MAC being a security model of an operating system, the main entities this model works on are users, subjects, and objects as described earlier. A user is someone who has a legitimate user account in the system. Each object is associated with a security classification which is also called the clearance level. A user can log in at any security level that is determined by the security clearance of the user. Each user is associated with one or more subjects. Subjects are generally processes that are generated by a user who is logged in at a specific security level.

The mandatory access control policies in the Bell–La Padula model are specified in terms of subjects and objects and the security levels of these subjects and objects [11, 14]. Assuming that function L maps an entity to its security level, the policies for reading and writing objects are given by the Simple Security and Restricted-* Properties. Ray

and Kumar [11] use the Restricted-* properties instead of the conventional *-property [14] for reasons of integrity. The Simple Security Property and the Restricted-* Property are defined below [11].

- *Simple Security Property*: A subject S is allowed to read object O only if the security level of the subject L(S) dominates the security level of the object L(O), that is, $L(O) \leq L(S)$.
- *Restricted-* Property*: A subject S is allowed to write to an object O only if the security level of the object *L*(O) equals the security level of the subject L(S), that is, $L(O) = L(S)$.

Systems that use MAC have to assign proper security labels to subjects and objects when they are created. Bishop [14] discusses an example of assigning and using MAC labels for the Data General B2 Unix system that provides mandatory access controls. When a process is generated in a system, it is assigned the MAC label of its parent. Objects are assigned labels upon their creation and are stored as parts of the object's attributes. An object can be associated with more than one region called a compartment. The highest region is dedicated for sensitive information such as logs and MAC label definitions and is not accessible for users. Reading up and writing up are not allowed and hence the users cannot modify data in the highest region. If data needs to be sent to user processes with MAC labels in lower user regions, it has to be sanitized.

Applications

MAC is the foundational concept of many security applications. Multilevel security (MLS) systems such as Trusted Solaris [15], TrustedBSD [16], and Trusted HP-UX [17] have MAC at their core. They implement a strong mandatory access control policy based on the Bell–La Padula model and a least privilege model to replace the root user in traditional Unix [18]. In these legacy MLS systems, each subject and object is assigned a security level according to the Bell–La Padula model and access is strictly controlled according to the MAC policy. However, flexibility of this policy is required for commercial operating systems so that a wide variety of applications can run in a more secure environment [18]. SELinux [10] is the result of this flexible approach, and implementing this MLS-variant in Linux distributions caters to a wide audience. SELinux provides a strong type enforcement MAC in addition to traditional MLS security labels in a flexible security architecture [18]. The notion of type enforcement here addresses both confidentiality and integrity and provides much more flexible control than the strictly hierarchical security labels as in the military MLS systems.

A recent extension of MAC is the location-based mandatory access control [11] that is necessitated by the increase in the growth of wireless networks and sensor and mobile devices. In this new operational environment, location of the user plays a key role in the performance and security of applications. Ray and Kumar [11] show how location information can be used to augment traditional access control mechanism to secure such ubiquitous computing applications. The location-based access control requires one to perform operations on location information and also protect the location information. In essence, it requires the formalization of the concept of location, the association of location with security levels, and the protection of location information by securely storing it and providing access through fine-grained role-based access control [19]. Though the initial location-based MAC work focused on military applications, subsequent refinements address generic applications [20] and context-aware dynamic situations [12].

Recently, Li et al. [13] made the observations that the traditional operating systems that use DAC and MAC continue to be vulnerable to Trojan horse attacks, and that later implementations of MAC extensions are too complex and intimidating to configure. They identified several design principles toward developing usable access control mechanism, the outcome of which is the usable mandatory integrity protection (UMIP) model. The new design principles for usable access control systems are [13]: (1) provide "good enough" security with a high level of usability, rather than "better" security with a low level of usability; (2) provide policy, not just mechanism; (3) have a well-defined security objective; (4) carefully design ways to support exceptions in the policy model; (5) rather than trying to achieve "strict least privilege," aim for "good-enough least privilege"; and (6) use familiar abstractions in policy specification interface. They have implemented the UMIP model in a prototype protection system for Linux using the Linux security module (LSM) framework [21]. Detailed evaluation and performance results are available in [13].

Recommended Reading

1. Pfleeger CP, Pfleeger SL (2007) Security in computing, 4th edn. Prentice Hall, Upper Saddle River
2. Committee of National Security Systems (2001) National information assurance (IA) Glossary, CNSS Instruction No. 4009, 26 April 2001
3. Department of Defense (1985) Trusted computer system evaluation criteria, DOD 5200.28-STD, December 1985

4. Bell DE, La Padula LJ (1976) Secure computer system: unified exposition and MULTICS, Technical Report ESD-TR-75–306, The MITRE Corporation, Bedford
5. Fraim LJ (1983) SCOMP: a solution to the multilevel security problem. IEEE Comput 16(7):26–34
6. National Computer Security Center (1991) Final evaluation report: Boeing space and defense group, MLS LAN Secure Network Server System, 28 August 1991
7. Weissman C (1992) BLACKER: security for the DDN, examples of A1 security engineering trades. In: Proceedings of the IEEE symposium on security and privacy, Oakland, pp 286–292
8. Committee on Computer-Computer Communication Protocols (1985) Transport protocols for department of defense data networks. National Academies Press, Washington, DC
9. Bauer M (2006) An introduction to Novell AppArmor. Linux J, (148):36, 38, 40–41, August 2006
10. McCarty B (2004) SELINUX: NSA's open source security enhanced Linux. O'Reilly Media, Sebastopol
11. Ray I, Kumar M (2006) Towards a location-based mandatory access control model. Comput Secur 25(1):36–44
12. Jafarian JH, Amini M, Jalili R (2009) A dynamic mandatory access control model. In: Sarbazi-Azad H, Parhami B, Miremadi S-G, Hessabi S (eds) Advances in computer science and engineering. Springer, Berlin Heidelberg, pp 862–866
13. Li N, Mao Z, Chen H (2009) Usable mandatory access control for operating systems. In: Raghav Rao H, Upadhyaya S (eds) Information assurance, security and privacy services. Emerald, Bingley
14. Bishop M (2005) Introduction to computer security. Addison Wesley Professional, Reading
15. Trusted Solaris 8 Operating Environment, White Paper (2000) Sun Microsystems, Palo Alto
16. FreeBSD handbook, FreeBSD Documentation Project (2000)
17. HP-UX Trusted Computing Services Administrator's Guide (2007) HP Part Number: 5991–7466
18. Legacy MLS/Trusted Systems and SELinux – Concepts and Comparisons to Simplify Migration and Adoption (2006) Hewlett-Packard White Paper 4AA1–0827ENW
19. Ferraiolo DF, Kuhn DR, Chandramouli R (2003) Role-based access control. Artech House, Boston and London
20. Decker M (2008) Requirements for a location-based access control model. In: Proceedings of the 6th international conference on advances in mobile computing & multimedia (MoMM2008), Linz, Austria, November 2008
21. Wright C, Cowan C, Smalley S, Morris J, Kroah-Hartman G (2002) Linux security modules. In: 11th Ottawa Linux symposium, Ottawa
22. Smalley S, Vance C, Salamon W (2001) Implementing SELinux as a Linux security module. Technical Report 01-43, NAI Labs
23. Ferraiolo DF, Sandhu R, Gavrila S, Kuhn DR, Chandramouli R (2001) Proposed NIST standard for role-based access control. ACM Trans Inf Syst Secur 4:224–274
24. Shankar U, Jaeger T, Sailer R (2006) Toward automated information-flow integrity verification for security-critical applications. In: Proceedings of the network and distributed systems security symposium, San Diego, February 2006, pp 267–280
25. Hicks B, Rueda S, Jaeger T, McDaniel P (2007) From trusted to secure: building and executing applications that enforce systems security. In: Proceedings of the 2007 USENIX annual technical conference, Santa Clara, May 2007, pp 205–218
26. St. Clair L, Schiffman J, Jaeger T, McDaniel P (2007) Establishing and sustaining system integrity via root of trust installation. In: Proceedings of the 2007 annual computer security applications conference, Miami Beach, December 2007, pp 19–29

Mandatory Access Control Policy (MAC)

Sabrina De Capitani di Vimercati, Pierangela Samarati
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Access Control Policies, Models, and Mechanisms; ►Mandatory Access Control

Definition

Mandatory access control policies (MACs) control access based on mandated regulations determined by a central authority.

Theory

With a mandatory access control policy, access decisions are made by a central authority [1]. The most common form of mandatory policy is the ►*multilevel security policy*, based on the classifications of *subjects* and *objects* in the system. Objects are passive entities storing information. Subjects are active entities that request access to the objects. Note that there is a distinction between *subjects* of the mandatory policy and the *authorization subjects* considered in the discretionary policies. While authorization subjects typically correspond to users (or groups thereof), mandatory policies make a distinction between *users* and *subjects*. Users are human beings who can access the system, while subjects are processes (i.e., programs in execution) operating on behalf of users. This distinction allows the mandatory policy to control the indirect accesses (leakages or modifications) caused by the execution of processes.

Recommended Reading

1. Samarati P, De Capitani di Vimercati S (2001) Access control: policies, models, and mechanisms. In: Focardi R, Gorrieri R (eds) Foundations of Security Analysis and Design, LNCS, vol 2171. Springer, Heidelberg

Man-in-the-Middle Attack

Yvo Desmedt
Department of Computer Science, University College London, London, UK

Related Concepts
►Diffie–Hellman Key Agreement

Definition
An adversarial computer between two computers pretending to one to be the other.

Theory
The man-in-the-middle attack is a very old attack that has been used against a wide range of protocols, going from login protocols, ►entity authentication protocols, etc.

To illustrate, consider ►Secure Socket Layer (SSL), used to protect the privacy and authenticity of WWW traffic. Current Public Key Infrastructures are either nonexistent or have very poor security, if any (for an incident example, see [5]). This implies that a man-in-the-middle can be launched as following. Suppose Alice wants to have a secure WWW connection to Bob's WWW page. When Eve is between Alice and Bob, Eve will pretend that her made-up public key is the one of Bob. So, when Alice accepts the fake certificate, she is in fact sending information to Eve. Eve can then start an SSL connection with the real WWW page of Bob. Even though encryption and authentication is used, once Eve has convinced Alice that her made-up key is the public key of Bob, Eve can be an active eavesdropper.

Man-in-the-middle attacks can also be launced against entity authentication schemes [2], allowing a third party, let say Eve, to pretend to be Alice. For possible solutions consult e.g., [2–4].

Experimental Results
Consult, e.g., [1].

Recommended Reading
1. Bart J (2011) Cars with keyless entry fall prey to antenna hack. http://hothardware.com/News/Cars%2Dwith%2Dkeyless%2Dentry%2Dfall%2Dpre%y%2Dto%2Dantenna%2Dhack/, 11 January, 2011
2. Bengio S, Brassard G, Desmedt YG, Goutier C, Quisquater J-J (1991) Secure implementations of identification systems. J Cryptol 4(3):175–183
3. Beth T, Desmedt Y (1991) Identification tokens or: solving the chess grandmaster problem. In: Menezes AJ, Vanstone SA (eds) Advances in cryptology | crypto '90, proceedings. Lecture notes in computer science, vol 537. Springer, Santa Barbara, 11–15 August 1991, pp 169–176
4. Brands S, Chaum D (1994) Distance-bounding protocols. In: Helleseth T (ed) Advances in cryptology | eurocrypt '93, proceedings. Lecture notes in computer science, vol 765. Springer, Lofthus, May 1993, pp 344–359
5. Erroneous verisign-issued digital certificates pose spoofing hazard. Updated: June 23, 2003, Microsoft security bulletin MS01-017, http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-017.asp, 22 March, 2001

MARS

Christophe De Cannière
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

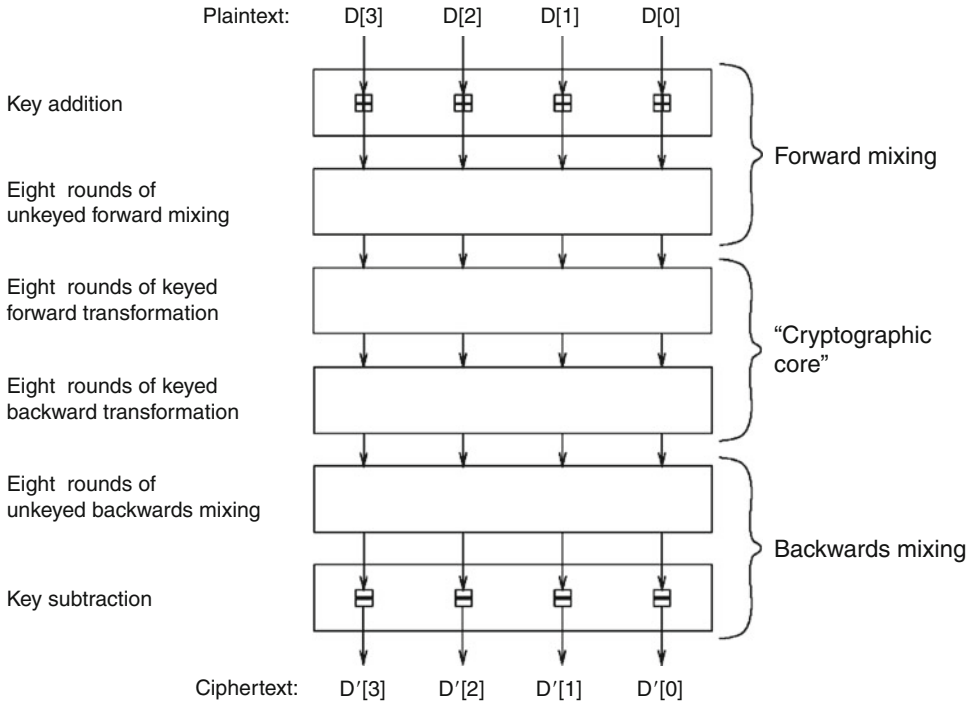
Related Concepts
►AES candidate; ►Block Ciphers

MARS [2] is a block cipher designed by IBM. It was proposed as an *Advanced Encryption Standard* (►Rijndael/AES) candidate in 1998 and was one of the five finalists in the AES selection process.

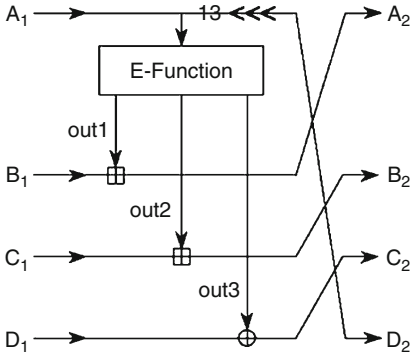
MARS has a block size of 128 bits and accepts secret ►keys of variable lengths, ranging from 128 to 1248 bits. It is a word-oriented cipher, i.e., all operations are performed on 32-bit words. The main distinguishing feature of MARS is its heterogeneous structure, inspired by the theoretical work of Naor and Reingold. The cipher consists of three stages:

Forward Mixing: First, four 32-bit subkeys are added to the data entering the cipher. The resulting block of four 32-bit words is then passed through eight rounds of a "type-3" ►Feistel network. In each round, one data word is used to modify the three other words. The Feistel network uses two fixed 8 × 32-bit S-boxes S_0 and S_1, and does not depend on the secret key in any way.

Cryptographic Core: The core of the encryption algorithm consists of a type-3 Feistel network of 2 × 8 rounds. In each round, the data is modified using an *E*-function which takes as input one data word and two key words, and produces three output words. The *E*-function itself uses many different components: a 9 × 32-bit S-box *S*, a 32-bit multiplication, fixed and data-dependent rotations, an addition, and XORs. After eight "forward"



MARS. Fig. 1 High-level structure of MARS



MARS. Fig. 2 One round of the type-3 Feistel network of the core (forward mode)

rounds, eight slightly different "backward" rounds are applied.

Backwards Mixing: This layer is essentially the inverse of the forward mixing layer. Notice, however, that different subkeys are used.

At present, only a few attacks on reduced-round versions of MARS have been presented. Note that, due to its heterogeneous structure, MARS can be downscaled in many different ways. A first approach is to concentrate on the core rounds. In [1], Biham and Furman have shown ►impossible differentials over 8 out of 16 core rounds. An attack breaking 11 rounds using amplified ►boomerang techniques is presented by Kelsey, Kohno, and Schneier [3, 4]. The same authors also proposed a straightforward ►meet-in-the-middle attack on a MARS version with only five core rounds, but with full forward and backward mixing.

Recommended Reading

1. Biham E, Furman V (2000) Impossible differential on 8-round MARS core. In: Proceedings of the Third AES candidate conference, New York, 13–14 April 2000, pp 186–194
2. Burwick C, Coppersmith D, Avignon ED, Gennaro R, Halevi S, Jutla C, Matyas SM Jr, O'Connor L, Peyravian M, Safford D, Zunic N (1998) MARS – a candidate cipher for AES. In: Proceedings of the First AES candidate conference, 20–22 August 1998. National Institute of Standard and Technology, Gaithersburg
3. Kelsey J, Schneier B (2000) MARS attacks! Preliminary cryptanalysis of reduced-round MARS variants. In: Proceedings of the Third AES candidate conference, New York, 13–14 April 2000, pp 169–185
4. Kelsey J, Kohno T, Schneier B (2001) Amplified boomerang attacks against reduced-round MARS and Seprepent. In: Schneier B (ed) Fast software encryption (FSE 2000), New York, 10–12 April 2000. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 75–93

MASH Hash Functions (Modular Arithmetic Secure Hash)

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC,
Katholieke Universiteit Leuven and IBBT,
Leuven-Heverlee, Belgium

Related Concepts

►Correcting–Block Attack; ►Hash Functions

Definition

MASH-1 and MASH-2 are constructions for *Hash Functions* based on *modular arithmetic*. Both are unkeyed cryptographic hash functions that have been designed to have the following properties: *preimage resistance*, *second preimage resistance*, and *collision resistance*.

Background

The history of hash functions based on modular arithmetic dates back to the mid 1980s. Many designs have been broken, including the one in CITT Recommendation X.509 Annex D (ISO 9594-8) (cf. ►Correcting Block Attack). The MASH design is the result of a long design effort within ISO/IEC JTC1/SC27/WG2; the final result was published as Part 4 of ISO/IEC 10118 [4] in 1998. The MASH hash functions are among the first hash function designs that use a strong output transformation.

Theory

In the following, N denotes an *RSA* modulus, that is, the product of two large primes and m denotes its length in bits or $m = \lceil \log_2 N \rceil$. The length n of the chaining variables in bits is then equal to the largest multiple of 16 strictly smaller than m. The length of the message blocks is equal to $n/2$ bits. The specification also needs a prime number p with $\lceil \log_2 p \rceil \leq m/2$; the prime p shall not be a divisor of N and the three most significant bits of p shall be equal to 1. The operation $\|$ denotes the concatenation of strings.

MASH-1 is defined for input strings of length $< 2^{n/2}$ bits. If necessary, the string X is right-padded with "0" bits to obtain a string with bit-length a multiple of $n/2$ and the resulting string is divided into t $n/2$-bit blocks denoted with $X_1, X_2, \ldots, X_t$. Next, a block X_{t+1} is added which contains the binary representation of the input string X in bits, left-padded with "0" bits. Subsequently each block X_i is expanded to an n-bit block $\tilde{X}_i$ as follows: insert four 1 bits before every 4-bit substring of X_i.

The MASH-1 compression function, which maps $3n/2$ bits to n bits, is defined as follows:

$$H_i = \left(\left(\left(\tilde{X}_i \oplus H_{i-1} \right) \vee A \right)^2 \pmod N \right) \sim n \ \oplus H_{i-1}. \qquad (1)$$

Here A = 0xF00...00 and $\sim n$ denotes that the rightmost n bits of the m-bit result are kept. The iteration starts with the all '0' string or $H_0 = 0^n$ and runs for $1 \leq i \leq t+1$.

At the end, a rather complex output transformation is applied to H_{t+1}. First H_{t+1} is divided into four $n/4$-bit blocks defined as follows: $H_{t+1} = Y_0 \| Y_1 \| Y_1 \| Y_3$. Define 12 $n/4$-bit blocks $Y_i = Y_{i-1} \oplus Y_{i-4}$, $4 \leq i \leq 15$. Combine the Y_i to eight additional $n/2$-bit blocks $X_{t+1+i} = Y_{2i-2} \| Y_{2i-1}, 1 \leq i \leq 8$, transform the X_{t+1+i} blocks to $\tilde{X}_{t+1+i}$, and perform eight additional iterations of the compression function with these blocks. Finally the hash result is computed as $H_{t+1+8} \bmod p$.

MASH-2 is obtained by replacing in MASH-1 the exponent 2 by the exponent $257 = 2^8 + 1$.

The redundancy in the block $\tilde{X}_i$ (four "1" bits in every byte) and the additional operations ($\vee A$ and $\sim n$) intend to preclude a ►Correcting Block Attack. The complex output transformation destroys some of the mathematical structure of the modular exponentiation.

When the factorization of the modulus is not known, the best known (2nd) preimage and collision attacks on MASH-1 require $2^{n/2}$ and $2^{n/4}$ operations [2]; they are thus not better than brute force attacks. While to date no efficient attacks are known that exploit the factorization of the modulus, knowledge of the factorization may reduce the security level. Therefore it is strongly recommended that the modulus N is generated by a trusted party (who deletes the factors after generation) or by using multi-party computation (e.g., Boneh and Franklin [1] and Frankel et al. [3]).

Recommended Reading

1. Boneh D, Franklin M (1997) Efficient generation of shared RSA keys. In: Kaliski B (ed) Advances in cryptology – CRYPTO '97: proceedings, Santa Barbara, 17–21 August 1997. Lecture notes in computer science, vol 1294. Springer, Berlin, pp 425–439
2. Coppersmith D, Preneel B (1995) Comments on MASH-1 and MASH-2. 21 February 1995, ISO/IEC JTC1/SC27/N1055
3. Frankel Y, MacKenzie PD, Yung M (1998) Robust efficient distributed RSA-key generation. In: Proceedings 30th ACM Symposium on the Theory of Computing, Dallas, 23–26 May 1998. ACM Press, New York, pp 663–672
4. ISO/IEC 10118 Information technology – Security techniques v Hash-functions. Part 1: General, 2000. Part 2: Hash-functions using an n-bit block cipher algorithm, 2010. Part 3: Dedicated hash-functions, 2004. Part 4: Hash-functions using modular arithmetic, 1998

Master Key

Carlisle Adams
School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada

Related Concepts

►Hash Functions; ►Key; ►Symmetric Cryptosystem

Definition

A *master key* is a cryptographic key used only for the protection of other keys.

Applications

A *master key* is a cryptographic ►key (typically a symmetric key (►Symmetric Cryptosystem)) whose sole purpose is to protect other keys, such as session keys, while those keys are in storage, in use, or in transit. This protection may take one of two forms: the master key may be used to encrypt the other keys, or the master key may be used to generate the other keys (e.g., if the master key is k_0, session key k_1 may be formed by hashing (►Hash Function) the concatenation of k_0 and the digit "1," session key k_2 may be formed by hashing the concatenation of k_0 and the digit "2," and so on).

Master keys are usually not themselves cryptographically protected; rather, they are distributed manually or initially installed in a system and protected by procedural controls and/or by physical or electronic isolation [1, 2].

Recommended Reading

1. Menezes A, van Oorschot P, Vanstone S (1997) Handbook of applied cryptography. CRC, Boca Raton, FL
2. Schneier B (1996) Applied cryptography: protocols, algorithms, and source code in C, 2nd edn. Wiley, New York

Maurer's Algorithm

►Maurer's Method

Maurer's Method

Moses Liskov
Department of Computer Science, The College of William and Mary, Williamsburg, VA, USA

Synonyms

Maurer's algorithm

Related Concepts

►Primality Test; ►Prime Generation; ►Prime Number

Definition

Maurer's method (or Maurer's algorithm) generates provably ►prime numbers that are nearly random.

Background

Maurer's algorithm was first described by Ueli Maurer in [2].

Theory and Applications

In Maurer's method, a number n is generated along with a ►certificate of primality to prove that n is prime. The certificate consists of a triple of number R, F, a, along with the prime factorization $q_1^{\beta_1} \ldots, q_r^{\beta_r}$ of F, where $2RF+1 = n$, and such that

1. $a^{n-1} \equiv 1 \bmod n$
2. $\gcd(a^{\frac{n-1}{q_j}} - 1, n) = 1$ for all $1 \le j \le r$.

This triple of numbers guarantees that all prime factors of n are of the form $mF+1$ for some positive integer m (the proof of this lemma can be found in [2] but is too complicated to be included here). In particular, if $F \ge \sqrt{n}$ then n must be prime as the product of any two primes of the form $mF+1$ is at least $F^2 + 2F + 1 > F^2 \ge n$.

Maurer's algorithm generates a prime at random by generating R and F at random with the prime factorization of F known, and testing to see if a random a makes a certificate of primality with R and F for $n = 2RF + 1$. In order to generate F at random with known factorization, we pick sizes for the primes of F at random according to a properly constructed distribution and generate primes of those sizes recursively. (As a base case, random selection and trial division or some other simple test is used to generate sufficiently small primes.) The manner in which these sizes are generated is rather complicated, but it is essentially along the lines of Bach's algorithm [1]. As any certificate actually proves that n is prime, none will ever be found for a composite number. In fact, most will fail the first step which is a single ►Fermat primality test. Furthermore, the probability that a random base a does form a certificate for $n = 2RF+1$ when n is prime is approximately $\phi(F)/F$ where $\phi(n)$ is ►Euler's totient function, the number of positive values less than or equal to n which are relatively prime to n. This ratio is high (approximately $1 - \sum_{j=1}^{r} \frac{1}{q_j}$, so the probability that a prime number will be recognized is nearly 1.

As is, Maurer's method generates prime numbers close to uniformly, so they are "nearly" random. More efficient

variants are also possible at the cost of a less uniform distribution.

Recommended Reading

1. Bach E (1988) How to generate random factored numbers. SIAM J Comput 17(4):173–193
2. Maurer UM (1995) Fast generation of prime numbers and secure public-key cryptographic parameters. J Cryptol 8(3):123–155

Maximal-Length Sequences

Tor Helleseth
The Selmer Center, Department of Informatics, University of Bergen, Bergen, Norway

Synonyms

m-Sequence; ML-Sequence

Related Concepts

►Autocorrelation; ►Cross-Correlation; ►Cyclic Codes; ►Euler's Totient Function; ►Finite Field; ►Gap; ►Golomb's Randomness Postulates; ►Irreducible Polynomial; ►Linear Feedback Shift Register; ►Modular Arithmetic; ►Primitive Element; ►Pseudo-Noise Sequences (PN-Sequences); ►Rijndael/AES; ►Run; ►Sequences; ►Stream Cipher

Definition

A binary maximal-length linear sequence is a sequence of period $2^n - 1$ generated by a linear recursion of degree n.

Theory

Among the most popular ►sequences for applications are the maximal-length ►linear feedback shift register sequences (m-sequences) [8, 9]. The balance, run distribution and ►autocorrelation properties of these sequences resemble properties one expect to find in random sequences. The m-sequences are the main ingredients in many important sequence families used in communication [5, 6] and in many ►stream cipher systems. Following is a description of m-sequences over the binary alphabet $GF(2) = \{0,1\}$, even though the sequences can be defined with symbols from any finite field with q elements, where q is a prime power.

A simple and efficient method to generate a sequence is using a linear recursion. For example, the recurrence relation (►Modular Arithmetic)

$$s_{t+3} = s_{t+1} + s_t \pmod 2$$

with initial state $(s_0, s_1, s_2) = (001)$ generates a periodic sequence

$$0010111 \quad 0010111 \quad 0010111 \quad 0010111 \ldots$$

of period $e = 7$. Different initial states lead to different sequences which can be cyclic shifts of each other.

Binary sequences can easily be generated in hardware using a linear shift register. One example of a shift register is shown in Fig. 1. The register consists of "flip-flops" each containing a 0 or a 1. The shift register is controlled by an external clock (not shown in the figure). Each time unit shifts each bit one step to the left and replaces the rightmost bit by the sum (mod 2) of the two leftmost bits. The register implements the recursion $s_{t+3} = s_{t+1} + s_t \pmod 2$, which with initial state $(s_0, s_1, s_2) = (001)$ gives the periodic sequence 0010111 . . . above. Table 1 shows the content of the shift register at each time unit.

A linear recursion of degree n is given by

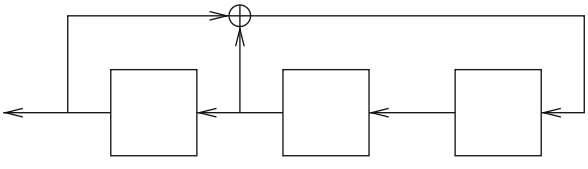
$$\sum_{i=0}^{n} f_i s_{t+i} = 0$$

where coefficients $f_i \in GF(2)$ for $0 < i < n$ and $f_0 = f_n = 1$. The *characteristic polynomial* of the recursion is defined by

$$f(x) = \sum_{i=0}^{n} f_i x^i.$$

The initial state and the given recurrence relation uniquely determine the generated sequence. A linear shift register with a characteristic polynomial of degree n generates 2^n different sequences corresponding to the 2^n different initial states and these form a vector space over $GF(2)$.

An n-bit linear shift register has at most 2^n different states. Since the zero state always is followed by the zero state, all sequences generated by a linear shift register have period at most $2^n - 1$. A *maximal length shift register sequence* (m-sequence) is a periodic sequence of maximal period $2^n - 1$ generated by a linear shift register of degree n. *The period of a polynomial* $f(x)$ is defined as the smallest positive integer e such that $f(x) \mid x^e - 1$.



Maximal-Length Sequences. Fig. 1 Shift register for $s_{t+3} = s_{t+1} + s_t \pmod 2$

Maximal-Length Sequences. Table 1 Shift register content in Fig. 1 generating an *m*-sequence

t	*S0*	*S1*	*S2*
0	0	0	1
1	0	1	0
2	1	0	1
3	0	1	1
4	1	1	1
5	1	1	0
6	1	0	0
7	0	0	1

Maximal-Length Sequences. Table 2 Characteristic polynomials and *m*-sequences

Degree	$f(x)$	*m*-sequence	Period
2	x^2+x+1	011	3
3	x^3+x+1	0010111	7
3	x^3+x^2+1	0011101	7
4	x^4+x+1	000100110101111	15
4	x^4+x^3+1	000111101011001	15

Let $f(x)$ be an ►irreducible polynomial of degree n and period $e = 2^n - 1$. Such a polynomial is called a *primitive polynomial* (►Primitive Element). The corresponding shift register generates an *m*-sequence when the initial state is nonzero. Any *m*-sequences have a primitive characteristic polynomial.

Binary *m*-sequences are perhaps the best known family of sequences. Table 2 shows some *m*-sequences and the corresponding characteristic polynomials. Figure 1 shows a shift register having $f(x) = x^3 + x + 1$ as characteristic polynomial and that generates an *m*-sequence of period $e = 7$ when the initial state is nonzero. Important properties for a binary *m*-sequence $\{s_t\}$ of period $2^n - 1$ are:

- (Balance property) In a period of the *m*-sequence there are 2^{n-1} ones and $2^{n-1} - 1$ zeros.
- (Multi-gram property) When t runs through $0, 1, \ldots, 2^n - 2$, the n-tuple

$$(s_t, s_{t+1}, \cdots, s_{t+n-1})$$

  runs through all binary n-tuples except for the n-tuple (0,0,…,0), which does not occur.
- (Shift-and-add property) For any τ, $0 < \tau \le 2^n - 2$, there exists a δ, depending on τ, such at

$$s_{t+\tau} + s_t = s_{t+\delta}$$

  for all $t = 0, 1, 2, \cdots$.
- (Invariance under decimating by 2) There exists a shift τ of the *m*-sequence such that the shifted sequence $\{u_t\} = \{s_{t+\tau}\}$ is invariant under decimation with two (when every second term of the sequence is selected), i.e., $\{u_t\} = \{u_{2t}\}$.
- (Run property) Let a ►run denote a consecutive set of zeros or ones in the sequence. In a period of the *m*-sequence, half of the runs have length 1, one-fourth have length 2, one-eight have length 3, etc., as long as the number of runs exceeds 1. Moreover, for each of these lengths, there are equally many 0-runs (►gaps) and 1-runs (*blocks*).

Example 1 *Consider the sequence with characteristic polynomial* $x^4 + x + 1$. *This is a primitive polynomial and generates the m-sequence* $\{s_t\} = 000100110101111$. *The sequence has the properties above, and it is balanced and contains 7 zeros and 8 ones. Each 4-tuple except the all zero 4-tuple occurs exactly once during a period of the sequence. The shift-and-add property is illustrated by the example*

$$\begin{aligned} s_{t+3} + s_t &= 100110101111000 + 000100110101111 \\ &= 100010011010111 \\ &= s_{t+11}. \end{aligned}$$

The sequence $\{s_t\}$ *is invariant by decimating by 2, i.e., in this case* $\tau = 0$, *since* $\{s_{2t}\} = \{s_t\}$. *Further, there are 4 runs of length 1, 2 runs of length 2, 1 run of length 3, and 1 run of length 4. The number of 0 runs and 1 runs of length* < 3 *are the same.*

Given a sequence $\{s_t\}$ of period e. In many application it is important to compare the sequence with its cyclic shifts. The ►autocorrelation of the binary sequence $\{s_t\}$, at shift τ, is defined as

$$A(\tau) = \sum_{t=0}^{e-1} (-1)^{s_{t+\tau} - s_t}.$$

In particular $A(\tau)$ gives the number of agreements minus the number of disagreements between $\{s_{t+\tau}\}$ and $\{s_t\}$. In most applications it is desirable that a shift of the sequence looks like a "random" sequence compared to itself, i.e., that $|A(\tau)|$ is small for all $\tau \not\equiv 0 \pmod{e}$. A very important property for an *m*-sequence is its two-level out-of-phase autocorrelation when $\tau \not\equiv 0 \pmod{e}$.

The autocorrelation of an *m*-sequence $\{s_t\}$ of period $e = 2^n - 1$ is given by:

$$A(\tau) = \begin{cases} -1 & \text{for} \quad \tau \not\equiv 0 \pmod{2^n - 1}, \\ 2^n - 1 & \text{for} \quad \tau \equiv 0 \pmod{2^n - 1}. \end{cases}$$

To prove this, define $u_t = s_{t+\tau} - s_t$ and observe that $\{u_t\}$ obeys the same linear recursion as $\{s_t\}$. This implies that $\{u_t\}$ is an *m*-sequence when $\tau \not\equiv 0 \pmod{2^n - 1}$ and the balance property of *m*-sequences gives

$$A(\tau) = \sum_{t=0}^{e-1} (-1)^{s_{t+\tau} - s_t} = \sum_{t=0}^{e-1} (-1)^{u_t} = -1.$$

Balance, multi-gram, and autocorrelation properties of m-sequences are properties one can expect in random binary sequences. The m-sequences obey the three ►Golomb's randomness postulates R1, R2, and R3 [2, 3]. R1 is the balance property, R2 the run property, and R3 the two-level autocorrelation property. Actually, these properties of m-sequences were the models for his postulates.

In order to describe the m-sequence it is useful to introduce the *trace function* Tr, which is a mapping from the ►finite field $GF(2^n)$ to the subfield $GF(2)$ given by:

$$Tr(x) = \sum_{i=0}^{n-1} x^{2^i}.$$

The trace function satisfies the following:

1. $Tr(x+y) = Tr(x) + Tr(y)$, for all $x, y \in GF(2^n)$.
2. $Tr(x^2) = Tr(x)$, for all $x \in GF(2^n)$.
3. $|\{x \in GF(2^n) \mid Tr(x) = b\}| = 2^{n-1}$ for all $b \in GF(2)$.
4. Let $a \in GF(2^n)$. If $Tr(ax) = 0$ for all $x \in GF2^n$, then $a = 0$.

Let $f(x)$ be the characteristic polynomial of the binary m-sequence $\{s_t\}$. It is well known that the zeros of $f(x)$ belong to the finite field $GF(2^n)$. The zeros are α^{2^i} for $i = 0, 1, \cdots, n-1$, where α is a ►primitive element of $GF(2^n)$, i.e., an element of order $2^n - 1$. The m-sequence can be written simply in terms of the trace representation as

$$s_t = Tr(a\alpha^t),\ a \in GF(2^n)^*,$$

where $GF(2^n)^* = GF(2^n)\backslash\{0\}$. This follows from the properties of the trace function. First observe that $\{s_t\}$ has $f(x)$ as its characteristic polynomial since,

$$\begin{aligned}\sum_{i=0}^{n} f_i s_{t+i} &= \sum_{i=0}^{n} f_i Tr(a\alpha^{t+i}) \\ &= Tr\left(a\alpha^t \sum_{i=0}^{n} f_i \alpha^i\right) \\ &= Tr(a\alpha^t f(\alpha^i)) \\ &= 0.\end{aligned}$$

The $2^n - 1$ different nonzero values of $a = \alpha^\tau, 0 \le \tau \le 2^n - 2$, correspond to all the cyclic shifts of the m-sequence. The case $a = 1$ gives the sequence with the property that $\{s_{2t}\} = \{s_t\}$.

Given an m-sequence $\{s_t\}$ of period $2^n - 1$ and let d be relatively prime to $2^n - 1$. The sequence $\{s_{dt}\}$ defined by selecting every dth term in $\{s(t)\}$ is also an m-sequence and all m-sequences of the same period can be obtained in this way. It follows from the trace representation that the characteristic polynomial of $\{s_{dt}\}$ is the primitive polynomial whose zeros are dth powers of the zeros of $f(x)$. The properties of the trace function implies that different m-sequences of the same period generated by distinct primitive characteristic polynomials are cyclically distinct. The number of binary primitive polynomials of degree n is $\phi(2^n - 1)/n$, where $\phi(x)$ is the ►Euler's totient function, the number of positive integers less than x that are relatively prime to x (►Modular Arithmetic). Thus, there are $\phi(2^n - 1)/n$ cyclically distinct m-sequences of period $2^n - 1$. The example below shows the two $\phi(7)/3 = 2$ cyclically distinct m-sequences of period $e = 7$.

Example 2 *The primitive polynomial* $f_1(x) = x^3 + x + 1$ *generates the m-sequence* $\{a_t\} = 0010111\ldots$ *of period* $e = 2^3 - 1 = 7$. *The primitive polynomial* $f_2(x) = x^3 + x^2 + 1$ *generates the m-sequence* $\{b_t\} = 0011101\ldots$ *of period* $e = 2^3 - 1 = 7$. *Note that* $\{b_t\} = \{a_{3t}\}$ *is the sequence obtained by selecting every third element of* $\{a_t\}$, *where indices are calculated modulo e.*

A well-studied problem is to compare two different m-sequences of the same period. Let $C_d(\tau)$ denote the ►cross-correlation function between the m-sequence $\{s_t\}$ and its decimation $\{s_{dt}\}$. By definition,

$$C_d(\tau) = \sum_{i=0}^{2^n-2} (-1)^{s_{t+\tau} - s_{dt}}.$$

If $d \notin \{1, 2, \cdots, 2^{n-1}\}$, (i.e., when the two m-sequences are cyclically distinct), then $C_d(\tau)$ takes on at least three distinct values as τ varies over the set $\{0, 1, \cdots, 2^n - 2\}$. It is therefore of special interest to study the cases when exactly three values occur. The following six decimations give three-valued cross correlation.

1. $d = 2^k + 1$, $n/gcd(n,k)$ odd.
2. $d = 2^{2k} - 2^k + 1$, $n/gcd(n,k)$ odd.
3. $d = 2^{\frac{n}{2}} + 2^{\frac{n+2}{4}} + 1$, $n \equiv 2 \pmod 4$.
4. $d = 2^{\frac{n+2}{2}} + 3$, $n \equiv 2 \pmod 4$.
5. $d = 2^{\frac{n-1}{2}} + 3$, n odd.
6. $d = \begin{cases} 2^{\frac{n-1}{2}} + 2^{\frac{n-1}{4}} - 1 & \text{if } n \equiv 1 \pmod 4 \\ 2^{\frac{n-1}{2}} + 2^{\frac{3n-1}{4}} - 1 & \text{if } n \equiv 3 \pmod 4. \end{cases}$

Applications

The cross-correlation function of m-sequences have many applications. Gold sequences, which are based on adding m-sequences that differ by the decimation (1) above, have found extensive practical applications during several decades. The cross correlation of m-sequences has also several close connections to almost bent functions as well as to almost perfect nonlinear (APN) functions, which are very important in studying S-boxes in cryptography

(see [1, 7] and ►Cyclic Codes). In the *Advanced Encryption Standard* (►Rijndael/AES), properties of the S-boxes come from properties of the cross correlation of an m-sequence and its reversed m-sequence. Recently the cross correlation of binary and nonbinary m-sequences has been important in constructing new families of sequences with two-level autocorrelation.

The pseudo-random properties of m-sequences have made them a popular building block in many communication systems and has lead to numerous practical applications, including synchronization, positioning systems, random number generation, stream cipher systems [4], and multiple-access communication.

Recommended Reading

1. Chabaud F, Vaudenay S (1995) Links between differential and linear cryptanalysis. In De Santis A (ed) Advances in cryptology – EUROCRYPT'94, vol 950, Lecture notes in computer science, pp 356–365. Springer, Berlin
2. Golomb SW (1967) Shift register sequences. Holden-Day series in information systems. Holden-Day, San Francisco, 1967. Revised ed., Aegean Park Press, Laguna Hills
3. Golomb SW, Gong G (2005) Signal design for good correlation – for wireless communication, cryptography, and radar. Cambridge University Press, Cambridge
4. Helleseth T (2009) Linear and nonlinear sequences and applications to stream ciphers. In Luengo I (ed) Recent trends in cryptography, vol 477, Contemporary mathematics, pp 21–46. American Mathematical Society, Providence, Rhode Island
5. Helleseth T, Vijay Kumar P (1998) Sequences with low correlation. In Pless VS, Huffman WC (eds), Handbook in coding theory, vol II, pp 1765–1853. Elsevier, Amsterdam
6. Helleseth T, Vijay Kumar P (2002) Pseudonoise sequences. In Gibson JD (ed) The communications handbook 2nd edn, pp 8–1–8–12. CRC Press, London
7. Nyberg K (1994) Differentially uniform mappings for cryptography. In Helleseth T (ed) Advances in cryptology - EUROCRYPT'93, vol 765 of Lecture notes in computer science, pp 55–64. Springer, Berlin
8. Selmer ES (1996) Linear recurrence relations over finite fields. Department of Mathematics, University of Bergen, Norway
9. Zierler N (1959) Linear recurring sequences. J Soc Ind Appl Math 7(1):31–48

Maxims

Friedrich L. Bauer
Kottgeisering, Germany

Related Concepts

►Cryptology (Classical); ►Shannon's Model

Here the most often quoted cryptological security maxims are listed [1].

Maxim Number One: "One should not underrate the adversary."

Della Porta's maxim: Only a cryptanalyst, if anybody, can judge the security of a cryptosystem (Auguste Kerckhoffs, [5] formulating the knowledge of the sixteenth century cryptologist Giambattista Della Porta[6]). To this, David Kahn remarked: "Nearly every inventor of a cipher system has been convinced of the unsolvability of his brainchild."

Kerckhoffs' maxim: "No inconvenience should occur if the cryptosystem falls into the hands of the enemy" [5].

Givierge's maxim: "Superficial complications can be illusory, for they can provide the cryptographer with a false sense of security" (Marcel Givierge, French cryptanalyst in WWI [2, 3]).

Rohrbach's maxim: "In judging the encryption security of a class of methods, cryptographic faults and other infringements of security discipline are to be taken into account." To this, Otto Horak remarked: "Security of a weak cipher method is not increased by trying to keep it [the method] secret." Thus, among other recommendations, the key has to be changed frequently, a periodic key is dangerous and, in the ideal case, a random one-time key is to be used [7].

Shannon's maxim: "The enemy knows the general system being used" [8].

Kahn's maxim: "Cryptographic errors, blunders, and faults can significantly simplify unauthorized decryption. To this, David Kahn [4] remarked "A cryptographer's error is the cryptanalysts only hope."

Recommended Reading

1. Bauer FL (2002) Decrypted secrets. In: Methods and maxims of cryptology. Springer, Berlin
2. Givierge M (1925) Questions de Chiffre, Revue Militaire Française, vol 94 (June 1924), 398–417 (July 1924) 59–78, Paris
3. Givierge M (1925) Cours de Cryptographie. Berger-Levrault, Paris
4. Kahn, D (1967) The codebreakers. Macmillan, New York
5. Kerckhoffs (1883) Auguste, La Cryptographie militaire. Journal des Sciences Militaires, 9 (January) 5–38, (February) 161–191. Available on http://www.cl.ac.uk./usweatfapp2/kerckhoffs/
6. Porta, GD (1563) De Furtivis Literarum Notis. Naples
7. Rohrbach H (1939–1946) Mathematische und Maschinelle Methoden beim Chiffrieren und Dechiffrieren. *FIAT Review of German Science*, 1939–1946: *Applied Mathematics*, 3 (I):233–257, Wiesbaden: Office of Military Government for Germany, Field Information Agencies, 1948
8. Shannon CE (1949) Communication theory of secrecy systems. Bell Syst Tech J 28:656–715

McEliece Public Key Cryptosystem

Nicolas Sendrier
Project-Team SECRET, INRIA Paris-Rocquencourt,
Le Chesnay, France

Related Concepts

►Error Correcting Codes; ►Public-Key Encryption; ►Syndrome Decoding Problem

Definition

The McEliece PKC is a public-key encryption scheme based on error correcting codes. The cryptogram is a code word of a binary Goppa code to which errors are added. Only the legal user, who knows the hidden algebraic structure of the code, can remove those errors and recover the cleartext.

Theory

It was introduced by Robert J. McEliece in 1978 [1] and is among the oldest public-key encryption schemes. Its security is related to hard algorithmic problems of algebraic coding theory. Its main advantages are very efficient encryption and decryption procedures and a good practical and theoretical security. On the other hand, its main drawbacks are a public key of large size and a ciphertext which is larger than the cleartext.

General Idea

The cleartext of k binary digits is encoded into a code word of $n > k$ binary digits by means of some encoder of a t-error correcting binary irreducible Goppa code of length n and dimension k. The ciphertext is obtained by flipping t randomly chosen bits in this code word. The cleartext is recovered from the ciphertext by applying the decoding procedure. The encoder (a generator matrix of the code) is public but the decoder is known only by the legal user.

Description

Let $\mathcal{F}$ denote a family of binary irreducible t-error correcting Goppa codes of length n and dimension k (where $k \le n - t\log_2 n$).

- *Key generation*. The legal user picks randomly and uniformly a code C in the family $\mathcal{F}$. Let G_0 be a generator matrix of C. The public key is equal to $G = SG_0P$, where S a random $k \times k$ non-singular binary matrix and P a random $n \times n$ permutation matrix.
- *Encryption*. The cleartext is a word x of $\mathbf{F}_2^k$. The ciphertext is a word of $\mathbf{F}_2^n$ equal to $xG+e$, where e is randomly chosen with a Hamming weight t.
- *Decryption*. The cleartext is recovered by applying the t-error correcting procedure of C to yP^{-1}.
- *In practice*. The initial proposal of McEliece was to use irreducible binary Goppa codes of length $n = 1024$ and dimension $k = 524$ correcting $t = 50$ errors. To keep up with 30 years of progress in algorithmics and computers [2–4], larger codes are required to obtain secure cryptosystems. We now need a length $n = 2048$ and a dimension $k = 2048 - 11t$, with $30 \le t \le 120$.

Security and Practice

We are given a family of binary Goppa codes. For an error-correcting capability of t errors, those codes have length $n \le 2^m$ and dimension $k = n - mt$ (codimension $r = mt$) for some integer m.

Security Reduction

The security of the McEliece encryption scheme is provably reduced to the easiest of the two following decision problems.

Problem 1 (Goppa Bounded Decoding - GBD)
Instance: *A matrix H in $\{0,1\}^{r\times n}$ and a word s of $\{0,1\}^r$.*
Question: *Is there a word e in $\{0,1\}^n$ of weight $\le \frac{r}{\log_2 n}$ such that $He^T = s$?*

Problem 2 (Goppa Code Distinguishing - GD)
Instance: *A matrix G in $\{0,1\}^{k\times n}$.*
Question: *Is G a generator matrix of a binary Goppa code?*

Problem 1 is a variant of the NP-complete Syndrome Decoding (SD) problem [5] and is also NP-complete [6]. The status of Problem 2 is unknown [7]. Both of them are conjectured hard in the average case.

Best Known Attacks

The two approaches for the cryptanalysis are:

- *Message* or *decoding attack* : decode t error in a known binary linear code of length n and dimension k
- *Key* or *structural attack* : deduce from the public key G an efficient t-error correcting procedure

The best known decoding attack is Stern's variant of information set decoding [2]. Its first full scale implementation is given in [3]. An effective attack breaking the original McEliece parameters ($n = 1024$, $k = 524$ and $t = 50$) in $\approx 2^{60}$ CPU operations is reported in [4].

The best known structural attack is reported in [8]. It is in fact an exhaustive search over the code family using the support splitting algorithm [9]. It is always less efficient

McEliece Public Key Cryptosystem. Table 1 Some parameters for the McEliece cryptosystem

(m, t)	(10, 50)	(11, 32)	(11, 70)	(12, 21)	(12, 40)
Ciphertext bits	1024	2048	2048	4096	4096
Cleartext bits	524	1696	1278	3844	3616
Information rate	0.51	0.83	0.62	0.94	0.88
Key size (in KB)	32	73	120	118	212
Security bits	**59.2**	**86.1**	**105.7**	**86.0**	**126.2**

Systems using binary Goppa code of length $n = 2^m$ and dimension $k = n - mt$
The security is the logarithm in base 2 of a lower bound [11] for the cost of the best decoding attack

McEliece Public Key Cryptosystem. Table 2 Performances and features of the hybrid McEliece encryption scheme

(m, t)	(10, 50)	(11, 32)	(11, 70)	(12, 21)	(12, 40)
Ciphertext bits	1024	2048	2048	4096	4096
Cleartext bits	807	1928	1713	4029	3936
Information rate	0.79	0.94	0.84	0.98	0.96
Encryption cost[a]	245	176	285	126	195
Decryption cost[a]	8080	1790	7080	571	1380

Systems using binary Goppa code of length $n = 2^m$ and dimension $k = n - mt$
The cleartext size is slightly smaller than $k + \log_2 \binom{n}{t}$
[a]Performances given in cycles per cleartext byte, measured on a single core of a Core 2 Intel processor. The C source code of [12] was compiled with icc

than the decoding attack. Table 1 presents the main features of the scheme for some typical sets of parameters. The key size is for a systematic generator matrix (i.e., the first $k \times k$ block of the public key G is the identity matrix). Using systematic generator matrices is provably as secure as the original scheme [10].

System Implementation

There exists one free open source implementation [12] of a variant of the McEliece encryption scheme. The error pattern is used to encode additional information, which increases the information rate. Table 2 presents the performances of this particular implementation.

Recommended Reading

1. McEliece RJ (1978) A public-key cryptosystem based on algebraic coding theory. DSN Progress Report, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, pp 114–116
2. Stern J (1989) A method for finding codewords of small weight. In: Cohen G, Wolfmann J (eds) Coding theory and applications. Lecture notes in computer science, vol 388. Springer, Berlin, pp 106–113
3. Canteaut A, Chabaud F (1998) A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. IEEE Trans Inf Theory 44(1):367–378
4. Bernstein D, Lange T, Peters C (2008) Attacking and defending the McEliece cryptosystem. In: Buchmann J, Ding J (eds) Post-quantum cryptography. Lecture notes in computer science, vol 5299. Springer, Berlin, pp 31–46
5. Berlekamp ER, McEliece RJ, van Tilborg HC (1978) On the inherent intractability of certain coding problems. IEEE Trans Inf Theory 24(3):384–386
6. Finiasz M (2004) Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie clef publique. Thèse de doctorat, École Polytechnique
7. Sendrier N (2002) On the security of the McEliece public-key cryptosystem. In: Blaum M, Farrell P, van Tilborg H (eds) Information, coding and mathematics. Kluwer international series in engineering and computer science, vol 687. Kluwer, Dordrecht, pp 141–163. Proceedings of Workshop honoring Prof. Bob McEliece on his 60th birthday
8. Loidreau P, Sendrier N (2001) Weak keys in McEliece public-key cryptosystem. IEEE Trans Inf Theory 47(3):1207–1212
9. Sendrier N (2000) Finding the permutation between equivalent codes: the support splitting algorithm. IEEE Trans Inf Theory 46(4):1193–1203
10. Biswas B, Sendrier N (2008) McEliece cryptosystem in real life: theory and practice. In: Buchmann J, Ding J (eds) PQCrypto 2008. Lecture notes in computer science, vol 5299. Springer, Berlin, pp 47–62
11. Finiasz M, Sendrier N (2009) Security bounds for the design of code-based cryptosystems. In: Matsui M (ed) Advances in cryptology – ASIACRYPT 2009. Lecture notes in computer science, vol 5912. Springer, Berlin, pp 88–105
12. HyMES: Hybrid McEliece Encryption Scheme, http://www-roc.inria.fr/secret/CBCrypto/index.php?pg=hymes Open source software

MD4 Hash Function

►MD4-MD5

MD4-MD5

Nicky Mouha
Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

Synonyms

MD4 hash function; MD5 hash function

Related Concepts

►Collision Resistance; ►Davies-Meyer Hash Function; ►Hash Functions; ►Iterated Hash Function; ►Preimage Resistance; ►Second Preimage Resistance

Definition

MD4 and MD5 are cryptographic hash functions designed by Rivest. Several hash functions have been influenced by their design. Practical attacks exist for MD4 and MD5, with high impact on commonly used applications.

Theory

Description

The MD4 [1] and MD5 [2] algorithms are cryptographic ►Hash Functions designed by Rivest. A cryptographic hash function converts a variable-length input into a fixed-length output. It is important that certain security requirements are met, such as ►Preimage Resistance, ►Second Preimage Resistance, and ►Collision Resistance. For both algorithms, the output length is 128 bits.

MD4 and MD5 are iterated hash functions, using the Merkle-Damgård mode of iteration. Messages are padded using the Merkle-Damgård strenghtening technique and split into 512-bit blocks. Each of these are processed by the compression function.

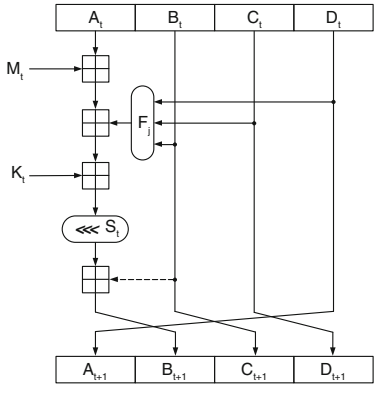
The MD4 compression function consists of three rounds, each further divided into 16 steps. A message expansion determines which 32-bit message words are used in every step. The step function is shown in Fig. 1.

Afterwards, a feedforward is applied to obtain a ►Davies-Meyer Hash Function.

MD5 was designed as a successor to MD4, with a more conservative design from a security point of view. The most significant changes include a fourth round, an extra addition in the step function, and different constants for every step. The step function of MD5 is shown in Fig. 1.

The MD4 and MD5 algorithms use addition modulo 2^{32} and 3-input Boolean functions as a source of nonlinearity in $\mathbb{F}_2$. This design allows for a fast implementation in software.

Several hash functions were influenced by the design of MD4 and MD5, such as the *SHA Family* and the *RIPEMD Family*.



MD4-MD5. Fig. 1 The step function of MD4 and MD5. The addition of B_t (*dashed line*) does not occur in MD4

Applications

MD5 is currently one of the most widely used hash functions. The use of MD4 is less common today. In the Linux kernel, an implementation of both MD4 and MD5 can be found.

To avoid storing passwords in the clear, the hash value of the password can be stored instead. On Unix-like systems (including Linux), the MD5 hash value of user account passwords can be stored in "`/etc/shadow`". Similarly, the Apache web server can store the MD5 hash of passwords for web site authentication [3]. On Windows, this is the default setting. NTLM (NT LAN Manager) [4], used for file and printer sharing under Windows, uses MD4 in the first version of its protocol, and HMAC-MD5 in the second version.

Unix-like operating systems include the md5sum tool to check files for integrity. In the rsync utility [5], used to synchronize files and directories, MD4 was used to check files for modification. Since version 3.0.0, MD5 is used instead.

For digital signatures, not the message itself, but the hash value of the message is signed. Digital signatures can be used to generate certificates for HTTPS [6]. HTTPS is used to secure web sites, for example, for banking or e-commerce applications. MD5 is currently still used in existing certificates, but is not used anymore for newly issued certificates [7].

Attacks

Attacks on the Short Output Length

MD4 and MD5 have an output length of 128 bits. According to the *birthday paradox*, a collision can be found after about 2^{64} compression function evaluations. As Yuval [8] showed, an attacker can choose both messages of this collision to be meaningful (instead of random-looking).

Van Oorschot and Wiener [9] estimated that such a machine could be built with $10 million in 1994, and obtain two meaningful messages with the same MD5 hash value

in 24 days. In 2009, a similar machine would cost only a few 10,000 dollars in off-the-shelf FPGAs [10].

Collision Attacks

Dobbertin showed that collisions for MD4 can be obtained with a complexity of 2^{20} equivalent compression function evaluations [11]. This attack can be carried out in seconds on a PC. Wang et al. later presented an improved attack with a complexity of 2^8 [12]. Naito et al. further improved the attack complexity to less than 3 compression function evaluations [13].

For MD5, Den Boer and Bosselaers found how a pseudo-collision can be obtained in 2^{16} [14]. Although they could not find a collision for the MD5 hash function, their result shows that the security proof of the Merkle-Damgård mode of iteration cannot be applied to MD5. Wang et al. were the first to find an actual collision for MD5 [15]. Improvements by Klima and Stevens bring the complexity of a collision search down to a few seconds on a standard PC.

For POP3 [16], the collision attacks on MD5 can be used to mount a password recovery attack. Together with IMAP [17], POP3 [18] is one of the most used protocols for retrieving e-mail.

Stevens et al. extended these results to chosen-prefix attacks [19]. In this scenario, the attacker can choose two arbitrary messages, and append a few extra blocks to them to make their hash values collide. They showed how this can be used to construct colliding X.509 certificates. A further improvement of their attack allowed them to obtain a rogue CA certificate [20]. This certificate allows an attacker to impersonate any website secured by HTTPS [6].

Preimage Attacks

Leurent showed how a preimage of MD4 can be constructed with a complexity of 2^{102} [21]. For MD5, Sasaki found a preimage attack with a complexity of 2^{123} [22].

Attacks on Concatenated Combiners

Very recently, attacks were proposed by Mendel et al. [23] when the message is hashed using both MD5 and and another hash function, and the outputs are concatenated. Such combiners are designed to hedge against nongeneric attacks on particular hash functions. In SSL 3.0/TLS 1.0 and TLS 1.1 [24, 25], the combiner MD5||SHA-1 is used. They estimate that an attack on a concatenated combiner using MD5 would have a complexity of about 2^{60}, if the collision attack on the other hash function is fast enough.

Open Problems

Fast progress is being made in the cryptanalysis of MD4 and MD5, as new techniques are being developed and existing methods are optimized. The improvement of the attacks on MD4 and MD5 can therefore be seen as an open problem.

Recommended Reading

1. Rivest RL (1990) The MD4 Message Digest Algorithm. In: Menezes A, Vanstone SA (eds) Advances in cryptology – CRYPTO '90: proceedings, Santa Barbara, 11–15 August 1990. Lecture notes in computer science, vol 537. Springer, New York, pp 303–311
2. Rivest RL (1992) The MD5 Message-Digest Algorithm. RFC 1321 (April 1992)
3. Franks J, Hallam-Baker P, Hostetler J, Lawrence S, Leach P, Luotonen A, Stewart L (1999) HTTP Authentication: Basic and Digest Access Authentication. RFC 2617 (Draft Standard) (June 1999)
4. Microsoft Corporation: NTLM v1 and NTLM v2 Messages. http://msdn.microsoft.com/en-us/library/cc236698(PROT.10).aspx (2010)
5. Davison W (2009) rsync. http://samba.anu.edu.au/rsync/
6. Rescorla E (2000) HTTP Over TLS. RFC 2818 (Informational) (May 2000)
7. Hoffman S (2008) Verisign Discontinues Flawed MD5 Certificates. http://www.crn.com/security/212700354 (December 2008)
8. Yuval G (1979) How to Swindle Rabin. Cryptologia 3:187–189
9. van Oorschot PC, Wiener MJ (1994) Parallel collision search with application to hash functions and discrete logarithms. In: 2nd ACM Conference on Computer and Communications Security, Fairfax, November 1994. ACM, New York, pp 210–218
10. Smart N et al (2009) ECRYPT II yearly report on Algorithms and Keysizes (2008–2009). Technical report, ECRYPT II Network of Excellence in Cryptography
11. Dobbertin H (1996) Cryptanalysis of MD4. In: Gollmann D (ed) FSE'96: proceedings, Cambridge, 21–23 February 1996. Lecture notes in computer science, vol 1039. Springer, Berlin, pp 53–69
12. Wang X, Lai X, Feng D, Chen H, Yu X (2005) Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer R (ed) Advances in cryptology – EUROCRYPT '05: proceedings, Aarhus, 22–26 May 2005. Lecture notes in computer science, vol 3494. Springer, Berlin, pp 1–18
13. Naito Y, Sasaki Y, Kunihiro N, Ohta K (2005) Improved collision attack on MD4 with probability almost 1. In: Won D, Kim S (eds) ICISC 2005: proceedings, Seoul, 1–2 December 2005. Lecture notes in computer science, vol 3935. Springer, Berlin, pp 129–145
14. den Boer B, Bosselaers A (1994) Collisions for the compression function of MD5. In: Advances in cryptology – EUROCRYPT '93: proceedings, Lofthus, 23–27 May 1993. Lecture notes in computer science, vol 756. Springer, Berlin, pp 293–304
15. Wang X, Yu H (2005) How to break MD5 and other hash functions. In: Cramer R (ed) Advances in cryptology – EUROCRYPT '05: proceedings, Aarhus, 22–26 May 2005. Lecture notes in computer science, vol 3494. Springer, Berlin, pp 19–35
16. Leurent G (2007) Message freedom in MD4 and MD5 collisions: application to APOP. In: Biryukov A (ed) FSE'07: proceedings,

Luxembourg, 26–28 March 2007. Lecture notes in computer science, vol 4593. Springer, Berlin, pp 309–328
17. Crispin M (2003) Internet Message Access Protocol – Version 4rev1. RFC 3501 (Proposed Standard) (March 2003) Updated by RFCs 4466, 4469, 4551, 5032, 5182
18. Myers J, Rose M (1996): Post Office Protocol – Version 3. RFC 1939 (Standard) (May 1996) Updated by RFCs 1957, 2449
19. Stevens M, Lenstra AK, de Weger B (2007) Chosen-prefix collisions for MD5 and colliding X.509 Certificates for different identities. In: Naor M (ed) Advances in cryptology – EUROCRYPT '07: proceedings, Barcelona, 20–24 May 2007. Lecture notes in computer science, vol 4515. Springer, Berlin, pp 1–22
20. Sotirov A, Stevens M, Appelbaum J, Lenstra A, Molnar DA, Osvik DA, de Weger B (2008) MD5 considered harmful today: creating a rogue CA certificate (December 2008) 25th Chaos Communications Congress, Berlin, Germany
21. Leurent G (2008) MD4 is not one-way. In: Nyberg K (ed) FSE'08: proceedings, Lausanne, 10–13 February 2008. Lecture notes in computer science, vol 5086. Springer, Berlin, pp 412–428
22. Sasaki Y, Aoki K (2009) Finding preimages in full MD5 faster than exhaustive search. In: Joux A (ed) Advances in cryptology – EUROCRYPT '09: proceedings, Cologne, 26–30 April 2009. Lecture notes in computer science, vol 5479. Springer, Berlin, pp 134–152
23. Mendel F, Rechberger C, Schläffer M (2009) MD5 is weaker than weak: attacks on concatenated combiners. In: Matsui M (ed) Advances in cryptology – ASIACRYPT '09: proceedings, Tokyo, 6–10 December 2009. Lecture notes in computer science, vol 5912. Springer, Berlin, pp 144–161
24. Dierks T, Allen C (1999) The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard) (January 1999) Obsoleted by RFC 4346, updated by RFC 3546
25. Dierks T, Rescorla E (2006) The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard) (April 2006) Obsoleted by RFC 5246, updated by RFCs 4366, 4680, 4681
26. Cramer R (ed) (2005) Proc. Advances in cryptology – EUROCRYPT '05: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, 22–26 May 2005. Lecture notes in computer science, vol 3494. Springer, Berlin

MD5 Hash Function

►MD4-MD5

MDC-2 and MDC-4

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers; ►Hash Functions

Definition

MDC-2 and MDC-4 are constructions for ►Hash Functions based on a ►Block Cipher, where the length in bits of the hash result is twice the block length of the block cipher.

Background

MDC-2 and MDC-4 have been described in a patent by Brachtl et al. [1] that was filed in 1988 and issued in 1990; they are also known as the Meyer-Schilling hash functions after the authors of the first paper describing these schemes [5]. MDC-2 is included in ISO/IEC 10118-2 [2] and used in the financial and health sectors.

Theory

MDC-2 and MDC-4 are unkeyed cryptographic hash functions which may have the following properties: *preimage resistance*, *second preimage resistance*, and *collision resistance*; these properties may or may not be achieved depending on the properties of the underlying block cipher.

In the following, the block length and key length of the block cipher will be denoted with n and k, respectively. The encryption with the block cipher E using the key K will be denoted with $E_K(.)$ and $\|$ denotes the concatenation of strings.

MDC-2 is an iterated hash function with a compression function that maps $2k + n$ bits to $2n$ bits. It requires two encryptions to hash an n-bit block, hence its rate is 1/2.

$$T_i^1 = E_{u(H_{i-1}^1)}(X_i) \oplus X_i = LT_i^1 \| RT_i^1$$
$$T_i^2 = E_{v(H_{i-1}^2)}(X_i) \oplus X_i = LT_i^2 \| RT_i^2$$
$$H_i^1 = LT_i^1 \| RT_i^2$$
$$H_i^2 = LT_i^2 \| RT_i^1 .$$

The variables H_0^1 and H_0^2 are initialized with the values IV_1 and IV_2, respectively, and the hash result is equal to the concatenation of H_t^1 and H_t^2. The functions u, v map the ciphertext space to the key space and need to satisfy the condition $u(IV^1) \neq v(IV^2)$. For *DES*, these mappings from 64 to 56 bits drop the parity bits in every byte and fix the second and third key bits to 01 and 10, respectively (to preclude attacks based on the complementation property and based on *weak keys* and semi-weak keys). By iterating this compression function in combination with MD-strengthening (►Hash Functions) one can construct a hash function based on this compression function.

For $k = n$, the best known (second) preimage attacks have a time/space product approximately equal to 2^{2n}, with as minimal time complexity $(n + 1)2^n$ and space complexity 2^{n+1} [4]; an earlier result of [6] lies on this curve

with a time complexity $2^{3n/2}$ and space complexity $2^{n/2}$. The complexity of the generic collision attack (time 2^n, negligible space) can be reduced with a small factor (about $log_2 n/n$) at the cost of a space requirement equal to the time complexity [4]. Note that the compression function of MDC-2 is rather weak: Preimage and collision attacks on the compression function require at most 2^n and $2^{n/2}$ encryptions. Steinberger [9] has proved a lower bound of $2^{3n/5}$ for collisions for the hash function in the ideal cipher model.

For *DES*, collisions can be found in time 2^{54} with negligible memory, or time and memory $2^{51.5}$; this is not an acceptable security level in 2011. For *AES*, collisions can be found in time 2^{128} with negligible memory, or time and memory $2^{124.5}$; this should be compared to the lower bound of 2^{75}.

The compression function of MDC-4 consists of the concatenation of two MDC-2 steps, where the plaintexts in the second step are equal to H^2_{i-1} and H^1_{i-1}. MDC-4 requires four encryptions to hash an n-bit block, hence its rate is 1/4. For $k = n$, the best known preimage attack for MDC-4 requires $2^{7n/4}$ operations. This shows that MDC-4 is probably more secure than MDC-2 against preimage attacks. However, finding a collision for MDC-4 itself requires only 2^{n+2} encryptions. The best known attacks on the compression function of MDC-4 require $2^{3n/2}$ encryptions for a (2nd) preimage and $2^{3n/4}$ encryptions for a collision [3, 7]. So far no lower bounds have been obtained on the collision or preimage resistance of MDC-4.

It is conjectured that both MDC-2 and MDC-4 achieve an acceptable security level (in 2011) against (2nd) preimage attacks for block ciphers with a block length and key length of 80 bits or more. It is also conjectured that both functions achieve an acceptable security level (in 2011) against collision attacks for block ciphers with a block length and key length of 128 bits or more (e.g., *AES*, *Camellia*, *CAST-256*, *MARS*, *RC6*, *TWOFISH*, and *SERPENT*).

It is also important to note that a block cipher may have properties which pose no problem at all when they are used only for encryption, but which may result in MDC-2 and/or MDC-4 to be insecure [7, 8]. Any deviation from "random behavior" of the encryption steps or of the key schedule could result in security weaknesses (for example, it would not be advisable to use *DES-X* due to the absence of a key schedule for part of the key).

Recommended Reading

1. Brachtl BO, Coppersmith D, Hyden MM, Matyas SM, Meyer CH, Oseas J, Pilpel S, Schilling M (1990) Data authentication using modification detection codes based on a public one way encryption function. US Patent 4,908,861, 13 Mar 1990
2. ISO/IEC 10118 Information technology – Security techniques – Hash-functions. Part 1: General, 2000, Part 2: Hash-functions using an n-bit block cipher algorithm, 2000, Part 3: Dedicated hash-functions, 2003. Part 4: Hash-functions using modular arithmetic, 1998
3. Knudsen L, Preneel B (2002) Enhancing the security of hash functions using non-binary error correcting codes. IEEE Trans Inf Theory 48(9):2524–2539
4. Knudsen LR, Mendel F, Rechberger C, Thomsen SS (2009) Cryptanalysis of MDC-2. In: Joux A (ed) Advances in cryptology – EUROCRYPT'09: proceedings, Cologne, 26–30 April 2009. Lecture notes in computer science, vol 5479. Springer, Berlin, pp 106–120
5. Meyer CH, Schilling M (1988) Secure program load with manipulation detection code. In: Proceedings of SECURICOM '88, Paris, pp 111–130
6. Lai X, Massey JL (1993) Hash functions based on block ciphers. In: Rueppel RA (ed) Advances in cryptology – EUROCRYPT '92: proceedings, Balatonfüred, 24–28 May 1992. Lecture notes in computer science, vol 658. Springer, Berlin, pp 55–70
7. Preneel B (1993) Analysis and design of cryptographic hash functions. Doctoral Dissertation, Katholieke Universiteit Leuven
8. Preneel B, Govaerts R, Vandewalle J (1994) Hash functions based on block ciphers: a synthetic approach. In: Stinson D (ed) Advances in cryptology – CRYPTO '93: proceedings, Santa Barbara, 22–26 August 1993. Lecture notes in computer science, vol 773. Springer, Berlin, pp 368–378
9. Steinberger JP (2007) The Collision Intractability of MDC-2 in the Ideal-Cipher Model. In: Naor M (ed) Advances in cryptology – EUROCRYPT '07: proceedings, Barcelona, 20–24 May 2007. Lecture notes in computer science, vol 4515. Springer, Berlin, pp 34–51

Measurement Models of Software Security

►Metrics of Software Security

Meet-in-the-Middle Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Hash Functions; ►Multiple Encryption

Definition

Meet-in-the-middle is a classical technique of ►*cryptanalysis* which applies to many constructions. The idea is that the attacker constructs patterns that propagate

from both ends to the middle of the cipher, in some cases by partial key-guessing. If the events do not match in the middle, the key-guess was wrong and may be discarded. Such attack has been applied to seven-round DES (►Data Encryption Standard) [1], and to ►*structural cryptanalysis* of *multiple-encryption* (e.g., two-key triple encryption) [2, 3]. Note that the technique can be mounted in *memoryless mode* [4, 5] using collision finding algorithms of Floyd or Nivasch.

A ►*miss-in-the-middle attack* may also be seen as a variant of the meet-in-the-middle technique in which the events in the middle should *not* match, and the keys that suggest a match in the middle are filtered as wrong keys.

Recently, this technique has been used to find preimages for hash functions, including *sponge functions* based on permutations (*P-sponges*) and hashes based on ►*Davies-Meyer* mode. Meet-in-the-middle attack is an important component of the *rebound attack* [6].

Recommended Reading

1. Chaum D, Evertse J-H (1986) Cryptanalysis of DES with a reduced number of rounds; sequence of linear factors in block ciphers. In: Williams HC (ed) Advances in cryptology – CRYPTO'85. Lecture notes in computer science, vol 218. Springer, Berlin, pp 192–211
2. Markle RC, Hellman ME (1981) On the security of multiple encryption. Commun ACM 24, 465–467
3. van Oorschot PC, Wiener MJ (1990) A known plaintext attack, on two-key triple encryption. In: Dawgard I (ed) Advances in cryptology – EUROCRYPT'90. Lecture notes in computer science, vol 473. Springer, Berlin, pp 318–325
4. Morita H, Ohta K, Miyaguchi S (1992) A switching closure test to analyze cryptosystems. In: Feigenbaum J (ed) CRYPTO 1991. LNCS, vol 576. Springer, Heidelberg, pp 183–193
5. Khovratovich D, Nikolic I, Weinmann R-P (2003) Meet-in-the-middle attacks on SHA-3 candidates. FSE 2009, pp 228–245
6. Mendel F, Rechberger C, Schläffer M, Thomsen SS (2009) The rebound attack: cryptanalysis of reduced whirlpool and grøstl. FSE 2009, pp 260–276

Memory and State Exhaustion Denial of Service

XiaoFeng Wang
School of Informatics and Computing, Indiana University at Bloomington, Bloomington, IN, USA

Synonyms

Memory and state exhaustion DoS

Related Concepts

►Computational Puzzles; ►Protocol Cookies; ►SYN Cookie Defense; ►SYN Flood Attack

Definition

Memory and state exhaustion denial of service (DoS) is one type of DoS in which an attacker or a group of attackers deplete the memory resources of a computing system to prevent it from providing services to legitimate users.

Background

Virgil Gligor discussed the DoS problem in 1983 [1]. Bill Cheswick and Steve Bellovin first discovered the weakness of the Transmission Control Protocol (TCP) three-way handshaking process [2] that enables the SYN flooding attack, the most famous memory and state exhaustion DoS.

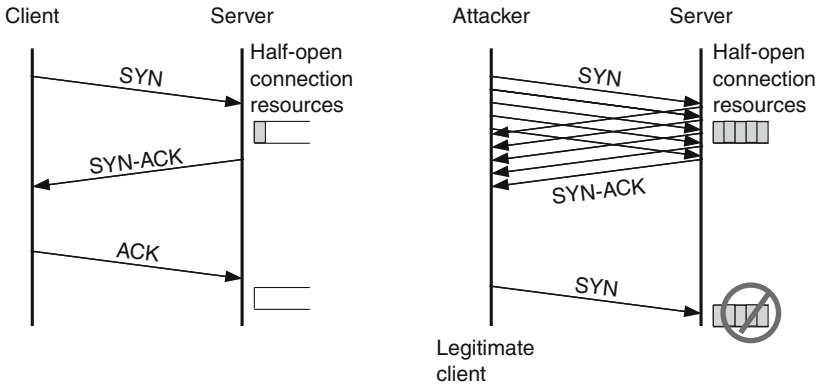
Theory

In a memory and state exhaustion DoS attack, the attacker exploits the weakness within a system's memory allocation mechanism to occupy a significant amount of memory or state resources at a relatively small cost. The most famous example of such an attack is *TCP SYN Flooding* in which the attacker continuously sends TCP connection requests to a server without completing a single connection to deplete the server's half-open connection resources, as illustrated in Fig. 1.

The attack can also be launched locally, on a multiuser system. For example, Moscibroda and Mutlu [3] show that a memory performance attack can happen to a multi-core system by exploiting its design weakness. A multi-core system has at least two cores (processors) integrated onto the same chip. These cores share the Dynamic Random Access Memory (DRAM) memory system that uses first-ready first-come-first-serve (FR-FCFS) strategy to serve the requests from different threads [3]. This strategy is designed to maximize memory bandwidth, but also introduces unfairness in memory use that can be exploited by a thread aggressively requesting DRAM resources. Such a thread can deplete the request buffers of the memory system with its requests, and as a result, prevents the system from serving other threads. The researchers found that the attack thread can slow down a victim thread by 2.9 times without significantly reducing its own performance.

Applications

Memory and state exhaustion DoS is traditionally considered to be a threat to network connections. More recent studies, however, show that the attack can actually be applied widely: besides the memory performance attack discussed above [3], such a DoS threat also endangers Voice over Internet Protocol (VoIP) infrastructures [4], wireless sensor networks [5] and other computing systems.



Memory and State Exhaustion Denial of Service. Fig. 1 TCP SYN flooding attack

Defense against memory and state exhaustion DoS relies on reducing the state information stored on the server [6], fairly allocating memory resources [3] and pricing service requestors through proof of work [7, 8]. For example, *SYN Cookies* [6] remove the queue for half-open connections and instead save the state information to the SYN-ACK packet through cryptographic means. As another example, client puzzle approaches [7, 8] force clients to commit a certain amount of computation resources to solve puzzles before the server commits its memory resources to provide them services.

Recommended Reading

1. Gligor VD (1983) A note on the denial-of-service problem. In Proceedings of the IEEE Symposium on Security and Privacy. Oakland, California, p 139
2. Bennahum, D. Panix attack. In MEME 2.12, October 1996, http://memex.org/meme2-12.html
3. Moscibroda T, Mutlu O (2007) Memory performance attacks: denial of memory service in multi-core systems. In Proceedings of the 16th USENIX Security Symposium. Boston, Massachusetts, pp 1–18
4. Sislaem D, Kuthan J, Ehlert S (2006) Denial of service attacks targeting a SIP VoIP infrastructure – attack scenarios and prevention mechanisms. In IEEE Networks Magazine, 20(5), IEEE Press
5. Deng J, Han R, Mishra S (2005) Defending against path-based DoS attacks in wireless sensor networks. In: The Third ACM Workshop on Security of Ad Hoc and Sensor Networks. Alexandria, Virginia, pp 89–96
6. Bernstein DJ. SYN cookies. In: http://cr.yp.to/syncookies.html
7. Juels A, Brainard J (1999) Client puzzle: a cryptographic defense against connection depletion attacks. In Kent S (ed) Proceedings of the 16th Annual Network and Distributed System Security Symposium. Boston, Massachusetts, pp 151–165
8. Wang X, Reiter M (2003) Defending against denial-of-service attacks with puzzle auctions. In Proceedings of the IEEE Symposium on Security and Privacy. Oakland, California, pp 78–92

Memory and State Exhaustion DoS

►Memory and State Exhaustion Denial of Service

Memory Overflow

►Buffer Overflow Attacks

Merkle-Hash-Trees Signatures

►Hash-Based Signatures

Mersenne Prime

Jerome A. Solinas
National Security Agency, Ft Meade, MD, USA

Definition

A *Mersenne number* is a number of the form $M_n = 2^n - 1$, where n is a positive integer. If M_n is also *prime*, then it is said to be a *Mersenne prime*. The number M_n can only be prime if n is prime. Some authors reserve the term "Mersenne number" for the numbers M_p for p prime. The Mersenne primes M_p for $80 \leq p \leq 700$ occur for $p = 89, 107, 127, 521$, and 607.

Background

Mersenne primes have been a topic of interest in *number theory* since ancient times. They appear in the theory of *linear feedback shift registers* (LFSR). If M_p is prime, then any binary LFSR of length p with irreducible feedback polynomial and nonzero initial state generates a maximal-length shift register sequence. Equivalently, every nonzero element of the *finite field* of 2^p elements is a *generator* for the entire *group* of nonzero elements.

Applications

Mersenne primes are also of interest in *public-key cryptography*. In public-key settings such as *elliptic curve cryptography*, the prime modulus p (►modular arithmetic) can be chosen to optimize the implementation of the arithmetic operations in the implementation of the cryptography. Mersenne primes provide a particularly good choice because modular reduction can be performed very quickly. One typically wishes to reduce modulo M_p a 2p-bit integer n (e.g., as a step in modular multiplication). In general, modular reduction requires an integer division. However, in the special case of reduction modulo M_p, one has

$$2^p \equiv 1 (\mathrm{mod} M_p).$$

Thus one can reduce n by writing

$$n = a \cdot 2^p + b,$$

where a and b are each positive and less than M_p. Then

$$n \equiv a + b (\mathrm{mod} M_p),$$

so that

$$n \bmod M_p = \begin{cases} a + b & \text{if } a + b < M_p, \\ a + b + 1 - 2^p & \text{otherwise.} \end{cases}$$

Thus reduction modulo a Mersenne prime requires an integer addition, as opposed to an integer division for modular reduction in the general case. There are two drawbacks to this method of modular reduction:

- Finding the integers a and b is easiest when p is a multiple of word size of the machine, since then there is no actual shifting of bits needed to align a and b for the modular addition. But word sizes are in practice powers of two, whereas p must be an odd prime.
- The Mersenne primes are so rare, with none between M_{127} and M_{521}, that usually there will be none of the desired magnitude.

For these reasons, cryptographers tend not to use Mersenne primes, preferring similar moduli such as *pseudo-Mersenne primes* and *generalized Mersenne primes*.

Message Authentication Algorithm

►MAA

Metrics of Software Security

Guido Salvaneschi[1], Paolo Salvaneschi[2]
[1]Department of Electronic and Information, Polytechnic of Milan, Milano, Italy
[2]Department of Information Technology and Mathematical Methods, University of Bergamo, Dalmine BG, Italy

Synonyms

Measurement models of software security

Related Concepts

►Fault Trees; ►Patterns for Software Security

Definition

Measuring software security requires to identify measurable properties of a software artifact and to build models that can relate the measures to a qualitative or quantitative value of the property "security."

Background

Security measurement of software products is an instance of the more general issue of measuring nonfunctional properties of software (the so-called software qualities).

This includes both the need to identify measurable properties of a software artifact and to build models that can relate the measures to a qualitative or quantitative evaluation of the more abstract property "security."

This requires, in general, a variety of measures to be integrated (through models) into the more abstract property "security." The reason is that different aspects concerning different development phases (for instance design and programming) may affect the software quality property.

These models can be useful for different purposes, ranging from evaluation (assessing the security for accepting or comparing software products) to design support (explaining the reasons for low security of a software product and provide advice for improvement).

The underlying assumption is that security is not a Boolean property. Like other properties of engineering artifacts, software security may have increasing levels

requiring both an increasing cost to be delivered and an increasing effort to be broken.

Finally, measuring "software security" is a very general task including the measurement of the protection against many types of attacks delivered for reaching different goals on different software products. This suggests the need of specialized metrics and models for security.

Theory

Measurements and Qualitative Models

A basic approach for software security measurement is the collection of measures from the software product and the integration of them into a global merit value in a qualitative space [1]. An example of a measure for a web application may be "Percent of Validated Input." The measure is computed through the ratio V/T where T is the count of the amount of input forms or interfaces the application exposes and V is the number of these interfaces that use input validation mechanisms. The ratio V/T makes a statement about the Web application's vulnerability to exploits from invalid input. Other measures may be related to other types of vulnerabilities.

The application (at design and program level) is inspected (using tools or manually) and the derived measures are aggregated through rules or functions (for instance, thresholds and mean functions) to provide an overall qualitative status report on the application security.

These models are specific applications for the security evaluation of the so-called "quality models" such as the ISO 9126 model. They essentially collect measures of security-related attributes and generate a diagnosis (the security level of the application) through heuristic relationships. It is common that, even if the elementary measures are quantitative, the values are finally mapped into a qualitative space. This procedure is typically composed of the following items:

- A set of elementary measures to be taken on the software product
- A tree of attributes and sub-attributes linking low-level measures to high-level abstractions
- An algorithm for generating values of high-level attributes from measures

This approach does not use explicit models of attacks (patterns of how attacks are conducted) and defenses (patterns of the software product structures resisting to the attacks), as well as relations between attack and defense patterns.

Defense Patterns, Attack Patterns, and Related Measurements

An evolution of qualitative models toward a richer model of the software artifact exploits the available knowledge on defense and attack patterns [2, 3].

Known attacks for various types of software applications have been classified and described through patterns. It is also known that the success of attacks to real software systems depends on poorly designed and implemented code. Security patterns implementing techniques for resisting to attacks have been described, for example, to protect web applications and a large body of knowledge of this type exists for software applications of various types [4–6]. For instance, SQL Injection is a well-known attack pattern for web applications with a database backend. A defense pattern for this attack describes the techniques required for validating input parameters that interact with a database through SQL queries. Measurable features like "number of parameters not validated against SQL injection" may be associated to the defense pattern.

A first method for exploiting security patterns is linking measures to defense patterns. The remaining part of the model is similar to qualitative models. The measures are processed through an aggregation algorithm to compute an overall security indicator. The advantage is that the measures are related to structural elements of the software product and this provides stronger suggestions for the designer. A poor security indicator may be explained as a poor use of defense patterns and the model may help in comparing design alternatives and choosing the best candidate.

Attack Trees

An additional improvement can be obtained taking into account possible attack patterns (in case decomposed into sub-attacks steps). This requires modelling a specific characteristic of security. A software application (for instance, a web application) may be attacked for different purposes, for instance "Denial of service" or "Obtain valid credential." For each goal, many different attacks are possible, each defendable by recurring to defense patterns. What is important is the weakest path for reaching the attack purpose through the attack patterns. This behavior is modelled through attack trees [7]. An attack tree is an AND/OR graph modelling various attacks, the activities composing attacks and their logical relations. If the leaves of the attack tree are associated to defense patterns and related measures, it is possible to use the graph for computing the value of the root node, which represents the overall security value. The leaf values are propagated upward using the rule "OR nodes take the value of their cheapest child; AND

nodes take the value of the sum of their children." This makes possible to apply to the tree the minimum path set analysis taken from fault tree analysis in order to determine the areas of greatest risk. The security value is the value associated to the minimum path.

Toward Quantitative Predictive Models

Specific models (for specific classes of software products) have been developed through the integration of the various components of the existing deterministic approaches: attack patterns, attack trees, defense patterns and associated measures [8].

These models also introduce a proposed quantification of security. The metric is the estimated time-to-break: the estimated time in minutes required to break the existing defenses.

A model is composed of an attack tree linking possible attacks patterns and sub-attacks. The tree is augmented, at the leaf nodes, with "cost-to-break functions." These functions implement flow diagrams representing the experimental knowledge (tuned through experiments) of how to conduct an attack or a sub-attack. They compute the time required to successfully execute the attacks and are influenced by the values of the defense pattern attributes. They are also influenced by the values of context parameters (the type of attacker and the resources available to the attacker).

Using suitable tools or manual inspections, it is possible to measure the attributes of the defense patterns and feed the model. The security strength metric is computed propagating, through the attack tree, the output values of the experimental cost-to-break functions. The resulting value (the value computed through the weakest path) is the estimated time required to break the existing defenses, given a tree of attacks and an implemented set of defenses and their attributes.

Probability-Based and Reliability-Like Models

What has been presented above is based on a deterministic approach. Another approach tries to exploit the same type of measures and concepts using probabilities [9–12].

A proposed model describes a decision tree for quantifying risks. This model depends on probabilistic descriptions of both vulnerabilities and countermeasures to known threats. Other proposals explicitly model the attacks and use the attack trees but associate probabilities of occurrence to the leaves, computing a probability through the graph.

Another proposal links a set of attacks and the related defense patterns. For each attack, a separate fault tree is defined. The factors of the fault trees are added gradually by examining the implemented/missing security patterns of the design. For each factor, the values (in a qualitative scale) for likelihood, exposure, and consequences are added and the root qualitative value is calculated through fuzzy computation. The qualitative values assigned to the model are based on the experimental analysis of existing systems as well as on subjective judgment.

A different approach tries to apply the dependability concepts to the software security evaluation [13]. This type of models is based on calculating security measures employing Markov models. For instance, discrete time Markov chains are used to model security risks based on the vulnerability knowledge of its components.

Applications

Security measures and models may be used for different purposes ranging from the evaluation of existing product to design support.

Different measuring and modelling approaches can provide different levels of support to these tasks.

Qualitative models based on a tree of attributes and measures may be used as a first level tool to rank the security of the application and reject them or suggest a significant improvement.

Even this simple type of models may be useful not only for evaluation purposes, but also for suggesting improvements. This can be done generating an explanation derived from the analysis of the contribution of each measure to the global qualitative value. Following the computation through the tree, from the leaves to the root, it is possible to highlight the most important contributions to a low value of the root. This path can be used as a constructive explanation for a poor value of security.

More sophisticated models can provide better support for the design. This is of course possible if the models have been reasonably tuned through a significant experimental activity.

The quantitative predictive type of models presented above may be also used for managing what-if scenarios. Given a set of delivered defenses, it is possible to predict the security value and simulate the effects of a defense improvement. For instance, it could be possible to discover that improving a defense could not be relevant due to the existence of another weaker path through the attack tree. The what-if experiments can be a tool for a cost-benefit analysis.

Open Problems and Future Directions

The problem of assessing the level of security of a software system is still largely open. Two main areas of improvement may be identified.

The first area is the development of measures and models that are more robust and are able to produce quantitative predictions of software security. This will allow moving from the qualitative evaluations and their limited engineering support. Improving this area should require both better measures and models and significant experimentation (as well as a shared definition of measuring units for security).

A second area is the specialization of measures and models for specific types of software artifacts. The phrase "measuring software security" is too vague for hosting an engineering content. Measuring the security of a software copy protection mechanism or measuring the security of a web application against attacks aimed at a specific goal are different problems. It is foreseeable that the software security measurement will move, like more mature engineering fields, toward more specialized approaches.

Recommended Reading

1. Nichols EA, Peterson G (2007) A metrics framework to drive application security improvement. IEEE Secur Priv 5(2): 88–91
2. Heyman T, Scandariato R, Huygens C, Joosen W (2008) Using security patterns to combine security metrics. In: Proceedings of the third international conference on availability, security and reliability, ARES, Barcelona, Spain. IEEE Computer Society, pp 1156–1163
3. Barnum S, McGraw G (2005) Knowledge for software security. IEEE Secur Priv 3(2):74–78
4. Owasp Top 10. http://www.owasp.org/index.php/Top_10_2007
5. Hoglund G, McGraw G (2004) Exploiting software: how to break code. Addison Wesley, Boston
6. Andrews M, Whittacker JA (2006) How to break web software. Addison-Wesley, Upper Saddle River
7. Schneier B (1999) Attack trees: modeling security threats. Dr. Dobb's J Softw Tools 24(12):21–29
8. Piazzalunga U, Salvaneschi P, Balducci F, Jacomuzzi P, Moroncelli C (2007) Security strength measurement for dongle protected software. IEEE Secur Priv 5(6):32–40
9. Sahinoglu M (2005) Security meter: a practical decision-tree model to quantify risk. IEEE Secur Priv 3(3):18–24
10. Grunske L, Joyce D (2008) Quantitative risk-based security prediction for component-based systems with explicitly modeled attack profiles. J Syst Softw 81(8):1327–1345
11. Nicol DM, Sanders WH, Trivedi KS (2004) Model-based evaluation: from dependability to security. IEEE Trans Dependable Secure Comput 1(1):48–65
12. Halkidis ST, Tsantalis N, Chatzigeorgiou A, Stephanides G (2008) Architectural risk analysis of software systems based on security patterns. IEEE Trans Dependable Secure Comput 5(3):129–142
13. Sharma VS, Trivedi KS (2005) Architecture based analysis of performance, reliability and security of software systems. In: Proceedings of the 5th international workshop on software and performance, Palma, Iles Balears, Spain. ACM Press, pp 17–227

Microdata Anonymization Techniques

►Microdata Masking Techniques

Microdata Disclosure Limitation

►Microdata Protection

Microdata Disclosure Protection

►Microdata Protection

Microdata Masking Techniques

Josep Domingo-Ferrer
Department of Computer Engineering and Mathematics, Universitat Rovira i Virgili, Tarragona, Catalonia

Synonyms

Microdata anonymization techniques; Microdata statistical disclosure control

Related Concepts

►k-Anonymity; ►Microdata Protection; ►Privacy-Preserving Data Mining; ►Statistical Disclosure Control; ►Synthetic Data Generation

Definition

Inference control in databases, also known as Statistical Disclosure Control (SDC), is a discipline that seeks to protect data so they can be published without revealing confidential information that can be linked to specific individuals among those to which the data correspond. SDC is applied to protect respondent privacy in areas such as official statistics, health statistics, e-commerce (sharing of consumer data), etc. Since data protection ultimately means data modification, the challenge for SDC is to achieve protection with minimum loss of the accuracy sought by database users.

Given a set **V** of original microdata (records corresponding to individual respondents), a *microdata masking* technique is a method which generates a modified version

$\mathbf{V}'$ in such a way that some statistical analyses yield similar results in $\mathbf{V}$ and $\mathbf{V}'$, but records in $\mathbf{V}'$ cannot easily be mapped to records in $\mathbf{V}$.

An alternative to microdata masking is synthetic data generation, whereby a simulated data set $\mathbf{V}'$ is generated that preserves some statistical properties of the original data $\mathbf{V}$.

Theory

The literature on inference control started in the 1970s, with the seminal contribution by Dalenius in the statistical community and the works by Schlörer and others in the database community. The 1980s saw moderate activity in this field. In the 1990s, there was renewed interest in the statistical community and the discipline was further developed under the names of statistical disclosure control in Europe and statistical disclosure limitation in America. Subsequent evolution has resulted in at least three clearly differentiated subdisciplines: tabular data protection, queryable database protection, and microdata protection. Good general works on SDC are [3, 5].

Microdata protection is about protecting static individual data, also called microdata [1]. It is only recently that data collectors (statistical agencies and the like) have been persuaded to publish microdata. Therefore, microdata protection is the youngest subdiscipline and is experiencing continuous evolution in the last years.

Given an original microdata set $\mathbf{V}$, the goal of *microdata masking* techniques is to produce a protected microdata set $\mathbf{V}'$ that can be released in such a way that:

1. Disclosure risk (i.e., the risk that a user or an intruder can use $\mathbf{V}'$ to determine confidential attributes on a specific individual among those in $\mathbf{V}$) is low.
2. User analyses (regressions, means, etc.) on $\mathbf{V}'$ and $\mathbf{V}$ yield the same or at least similar results.

Masking methods can in turn be divided in two categories depending on their effect on the original data:

- *Perturbative masking*. The microdata set is distorted before publication. In this way, unique combinations of scores in the original dataset may disappear and new unique combinations may appear in the perturbed dataset; such confusion is beneficial for preserving statistical confidentiality. The perturbation method used should be such that statistics computed on the perturbed dataset do not differ significantly from the statistics that would be obtained on the original dataset. Examples of perturbative masking methods are *noise addition*, *microaggregation*, *post-randomization* (PRAM), *data/rank swapping*, *microdata rounding*, etc.

  See [3] for more information. Noise addition and microaggregation are briefly discussed next.

  – *Noise addition* is a masking method for statistical disclosure control of numerical microdata that consists in adding random noise to original microdata. The vector of observations x_j for the j-th attribute X_j of the original dataset is replaced by a vector $z_j = x_j + \epsilon_j$, where ϵ_j is a vector of random noise. There are several types of noise addition. In uncorrelated noise addition, ϵ_j is drawn from a random variable ε_j following a normal distribution with mean 0 and variance $\sigma^2_{\varepsilon_j}$, and the covariance between any two different noise variables, say ε_t and ε_l for any $t \neq l$, is 0; this does not preserve variances or correlations. In correlated noise addition, the covariance matrix of the errors is proportional to the covariance matrix Σ of the original data, i.e., ε follows a multivariate normal distribution with mean vector 0 and covariance matrix $\alpha\Sigma$; this preserves means and additionally allows preservation of correlation coefficients. In noise addition and linear transformation, it is ensured by additional transformations that the sample covariance matrix of the masked attributes is an unbiased estimator of the covariance matrix of the original attributes.

  – *Microaggregation* is a family of masking methods for statistical disclosure control of numerical microdata (although variants for categorical data exist). The rationale behind microaggregation is that confidentiality rules in use allow publication of microdata sets if records correspond to groups of k or more individuals, where no individual dominates (i.e., contributes too much to) the group and k is a threshold value. Strict application of such confidentiality rules leads to replacing individual values with values computed on small aggregates (microaggregates) prior to publication. This is the basic principle of microaggregation. To obtain microaggregates in a microdata set with n records, these are combined to form g groups of size at least k. For each attribute, the average value over each group is computed and is used to replace each of the original averaged values. Groups are formed using a criterion of maximal similarity. Once the procedure has been completed, the resulting (modified) records can be published. The optimal k-partition (from the information loss point of view) is defined to be the one that maximizes within-group homogeneity; the higher the within-group homogeneity, the lower the information loss, since microaggregation replaces values in

a group by the group centroid. The sum of squares criterion is common to measure homogeneity in clustering. The within-groups sum of squares SSE is defined as $SSE = \sum_{i=1}^{g} \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)'(x_{ij} - \bar{x}_i)$. The lower the SSE, the higher the within-group homogeneity. Thus, in terms of sums of squares, the optimal k-partition is the one that minimizes SSE. See [2] for an approximation algorithm to optimal microaggregation. If microaggregation is applied to key attributes (i.e., quasi-identifiers) in the original dataset, then microaggregation yields *k-anonymity*; a protected dataset is said to satisfy k-anonymity [4] for $k > 1$ if, for each combination of key attribute values, at least k records exist in the dataset sharing that combination.

- *Non-perturbative masking*. These methods do not alter data; rather, they produce partial suppressions or reductions of detail in the original dataset. Sampling, global recoding, top and bottom coding, and local suppression are examples of non-perturbative masking methods.
  - *Sampling* is a non-perturbative masking method for statistical disclosure control of microdata. Instead of publishing the original microdata file, what is published is a sample S of the original set of records. Sampling methods are suitable for categorical microdata, but for continuous microdata they should probably be combined with other masking methods. The reason is that sampling alone leaves a continuous attribute V_i unperturbed for all records in S. Thus, if attribute V_i is present in an external administrative public file, unique matches with the published sample are very likely: Indeed, given a continuous attribute V_i and two respondents o_1 and o_2, it is highly unlikely that V_i will take the same value for both o_1 and o_2 unless $o_1 = o_2$ (this is true even if V_i has been truncated to represent it digitally).
  - *Global recoding* or *generalization* is a masking method for statistical disclosure control of microdata. For a categorical attribute V_i, several categories are combined to form new (less specific) categories, thus resulting in a new V_i' with $\left|D\left(V_i'\right)\right| < |D(V_i)|$ where $|\cdot|$ is the cardinality operator. For a numerical attribute, global recoding means replacing V_i by another attribute V_i' which is a discretized version of V_i. In other words, a potentially infinite range $D(V_i)$ is mapped onto a finite range $D\left(V_i'\right)$. This technique is more appropriate for categorical microdata, where it helps disguise records with strange combinations of categorical attributes. Global recoding is used heavily by statistical offices. Global recoding is implemented in the μ-Argus package [3]. In combination with local suppression, it can be used to achieve k-anonymity [4].
  - *Top coding* and *bottom coding* are special cases of the global recoding masking method for statistical disclosure control of microdata. Their operating principle is that top values (those above a certain threshold), respectively bottom values (those below a certain threshold), are lumped together to form a new category. Top and bottom coding can be used on attributes that can be ranked, i.e., numerical or categorical ordinal.
  - *Suppression* or *blanking* is a masking method for statistical disclosure control of microdata. Certain values of individual attributes are suppressed with the aim of increasing the set of records agreeing on a combination of key values. Local suppression is implemented in the μ-Argus package. The combination of local suppression and global recoding can be used to attain k-anonymity. If a numerical attribute V_i is part of a set of key attributes, then each combination of key values is probably unique. Since it does not make sense to systematically suppress the values of V_i, it can be asserted that local suppression is rather oriented to categorical attributes.

Microdata masking faces an inherent trade-off between loss of information (i.e., analytical utility) and disclosure risk (►Microdata Protection).

Applications

There are several areas of application of SDC and microdata masking, which include, but are not limited to the following:

- *Official statistics*. Most countries have legislation which compels national statistical agencies to guarantee statistical confidentiality when they release data collected from citizens or companies. This justifies the research on SDC undertaken by several countries, among them the European Union (e.g., the FP5 CASC project) and the United States.
- *Health information*. This is one of the most sensitive areas regarding privacy. For example, in the US, the Privacy Rule of the Health Insurance Portability and Accountability Act (HIPAA) requires strict protection/masking of health information for use in medical research. In most western countries, the situation is similar.

- *E-commerce*. Electronic commerce results in the automated collection of large amounts of consumer data. This wealth of information is very useful to companies, which are often interested in sharing it with their subsidiaries or partners. Such consumer information transfer should not result in public profiling of individuals and is subject to strict regulation, especially in the European Union and the United States. Sharing masked data rather than original data is a good option.

Recommended Reading

1. Ciriani V, De Capitani di Vimercati S, Foresti S, Samarati P (2007) Microdata protection. In: Yu T, Jajodia S (eds) Secure data management in decentralized systems. Springer, Heidelberg
2. Domingo-Ferrer J, Sebé F, Solanas S (2008) A polynomial-time approximation to optimal multivariate microaggregation. Comput Math Appl 55(4):714–732
3. Hundepool A, Domingo-Ferrer J, Franconi L, Giessing S, Lenz R, Longhurst J, Schulte-Nordholt E, Seri G, DeWolf P-P (2009) Handbook on statistical disclosure control (version 1.2). Eurostat (ESSNET SDC project deliverable). http://neon.vb.cbs.nl/casc/..%5Ccasc%5Chandbook.htm
4. Samarati P (2001) Protecting respondents' identities in microdata release. IEEE Trans Knowl Data Eng 13(6):1010–1027
5. Willenborg L, DeWaal T (2001) Elements of statistical disclosure control. Springer, New York

Microdata Protection

Sara Foresti
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Synonyms

Microdata disclosure limitation; Microdata disclosure protection

Related Concepts

►k-Anonymity; ►Macrodata protection; ►Microdata masking techniques

Definition

Microdata protection techniques ensure protection of respondents' identities and/or their related information when data are released or published in detailed (in contrast to aggregated) form.

Background

Today's globally networked society places great demand on the dissemination and sharing of information, which is probably becoming the most important and demanded resource. While in the past released information was mostly in tabular and statistical form (*macrodata*), many situations call today for the release of specific data (*microdata*). Microdata, in contrast to macrodata reporting pre-computed statistics, provide the convenience of allowing the final recipient to perform different analysis as needed on the data. The protection of microdata against improper disclosure is therefore an issue that has become increasingly important. Disclosure can be categorized as: *identity disclosure*, *attribute disclosure*, and *inferential disclosure*. Identity disclosure occurs when using a combination of identifying attributes (e.g., Social Security number, name, and address), an individual's identity can be reconstructed. Attribute disclosure occurs when using a combination of indirect identifying attributes, a given attribute value (or restricted set thereof) can be associated with an individual. Inferential disclosure occurs when information can be inferred with high probability from statistical properties of the released data.

A first step in protecting the privacy of the *respondents* (i.e., individuals, organizations, and so on) to whom the data refer, consists in removing all explicit identifiers. *De-identified* data may however contain *quasi-identifiers* (e.g., race, birth date, sex, and ZIP code) that uniquely, or almost uniquely, pertain to specific respondents and make them stand out from others [2]. Microdata protection techniques are applied to protect sensitive de-identified data from identity, attribute, and inferential disclosure.

Theory

Intuitively, *microdata* contain a set of attributes relating to single respondents in a sample or in a population. Microdata can be represented as tables, composed of tuples (records) with values from a set of attributes. The attributes in a microdata table are usually classified as follows:

- *Identifiers*. Attributes that uniquely identify respondents. For instance, attribute `SSN` uniquely identifies the person with which each tuple is associated.
- *Quasi-identifiers*. Attributes that, in combination, can be linked with external information to reidentify all or some of the respondents to whom information refers or reduce the uncertainty over their identities. For instance, attributes `Sex`, `Marital Status`, `ZIP`, and `DoB`.
- *Confidential attributes*. Attributes of the microdata table that contain sensitive information. For instance, attribute `Disease` can be considered sensitive.

- *Non confidential attributes.* Attributes that the respondents do not consider sensitive and whose release does not cause disclosure.

In general, to limit the disclosure risk of a microdata table it is first necessary to suppress explicit and implicit identifiers. This process is also known as *de-identification*. Figure 1 illustrates an example of microdata table with five tuples and with attributes `SSN` (social security number), `Name`, `Sex`, `Marital Status`, `ZIP` code, `DoB` (Date of Birth), and `Disease`, where data have been de-identified by suppressing names and Social Security numbers. De-identifying data, however, provide no guarantee of anonymity. Released information often contains other data, such as marital status, birth date, sex, and ZIP code, that can be linked to publicly available information to re-identify respondents and to infer information that was not intended for release [3].

Classification of Microdata Disclosure Protection Techniques

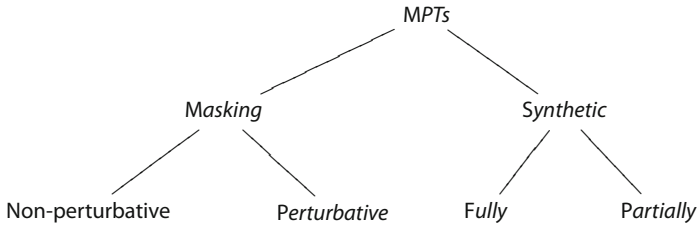
Several microdata disclosure protection techniques have been proposed in the literature [1]. Basically, these techniques are based on the principle that reidentification can be counteracted by reducing the amount of released information, masking the data (e.g., by not releasing or by perturbing their values), or by releasing plausible but made up values instead of the real ones. Microdata protection techniques can then be classified into two main categories: *masking techniques*, and *synthetic data generation techniques* (see Fig. 2).

- *Masking techniques*. The original data are transformed to produce new data that are valid for statistical analysis and such that they preserve the confidentiality of respondents. Masking techniques can be classified as:
  - *Non-perturbative*, the original data are not modified, but some data are suppressed and/or some details are removed.
  - *Perturbative*, the original data are modified.
- *Synthetic data generation techniques*. The original set of tuples in a microdata table is replaced with a new set of tuples generated in such a way to preserve the key statistical properties of the original data. The generation process is usually based on a statistical model, and the key statistical properties that are not included in the model will not be necessarily respected by the synthetic data. Since the released microdata table contains synthetic data, the reidentification risks is reduced. Note that the released microdata table can be entirely synthetic (i.e., *fully* synthetic) or mixed with the original data (i.e., *partially* synthetic).

Another important feature of microdata protection techniques is that they can operate on *continuous* and/or *categorical* data. An attribute is said to be continuous if it is numerical and arithmetic operations are defined on it. For instance, attribute `Age` is a continuous attribute. An attribute is said to be categorical if it can assume a limited and specified set of values and arithmetic operations do not have sense on it. Note that an order relationship can be defined over a categorical attribute. For instance,

SSN	Name	Sex	Marital status	ZIP	DoB	Disease
		M	Married	22020	1975/12/04	Hypertension
		M	Married	22020	1964/10/16	Short breath
		M	Divorced	22024	1978/04/12	Hypertension
		F	Widow	22024	1925/08/25	Chest pain
		F	Single	22024	1978/04/12	Chest pain

Microdata Protection. Fig. 1 An example of a de-identified microdata table



Microdata Protection. Fig. 2 Classification of microdata protection techniques (MPTs)

attributes `Marital Status` and `Sex` are categorical attributes.

Assessing Microdata Confidentiality and Utility

Microdata protection techniques should balance two contrasting needs: the need for confidentiality protection and the need for data. Protection technique can then be evaluated with respect to *disclosure risk* and *data utility*.

Disclosure risk is the risk that identity and/or attribute disclosure will be encountered if protected microdata are released. In general, two factors may have an impact on identity disclosure.

- *Population uniqueness* means that the probability of identifying a respondent who is the unique respondent with a specific combination of attributes is high if those attributes are present in the microdata table.
- *Reidentification* means that the released microdata is linked to another published table, where the identifiers have not been removed.

The main methods for measuring the risk of identity disclosure therefore quantify the risk of having population *uniqueness* and the risk of *record* or *data linkage* (i.e., the risk of finding a matching between a tuple in the microdata table and a tuple in a public non anonymous dataset). The main method to measure attribute disclosure quantifies instead the risk of *interval disclosure*, that is, the risk of reducing to a small interval the set of values that the sensitive attribute can assume for a given respondent.

Data utility is measured as the amount of information that is preserved by the protection technique. The measure of data utility is strongly connected to the *purpose* for which the information will be used. Since the purpose may be different and not known a priori, it is not possible to establish a general data utility measure based on purpose. The methods used are therefore based on the concepts of *analytically valid* and *analytically interesting*. A protected microdata table is *analytically valid* if it approximately preserves statistical analyzes (e.g., mean and co-variance) that can be produced with the original microdata. A protected microdata table is *analytically interesting* if it contains a sufficient number of attributes that can be validly analyzed.

In general, there are two strategies for computing data utility: (i) directly comparing the tuples of the protected microdata with the tuples in the original microdata; (ii) comparing the statistics computed on the protected microdata with the same statistics evaluated on the original microdata.

Disclosure risk and data utility combination. Microdata protection techniques have a different impact on disclosure risk and data utility. To be able to assess alternative microdata protection techniques, it is necessary to define a framework for assessing how good a protection technique is. Different proposals have been introduced in the literature (e.g., *k-anonymity* [3] and *ℓ-diversity*) that establish an upper bound to the disclosure risk of a released table. Among the tables, obtained by applying microdata protection techniques, that satisfy the disclosure risk threshold, the one with the highest disclosure risk guarantees higher data utility. In fact, such a table has been obtained by removing from the original microdata table only the information necessary to satisfy the confidentiality requirement.

The disclosure risk and data utility measures should be used before releasing the data to verify whether the protection is adequate to the respondents' requests of confidentiality and to the data recipients' needs of information.

Applications

Microdata protection techniques can be applied whenever the publication of information directly referred to respondents is required. This may happen, for example, when a private organization makes available various data regarding its business (products, sales, and so on) and, at the same time, needs to protect sensitive information such as the identity of its customers or plans for future products. As another example, this may also happen when government agencies release historical data for statistical analysis and, at the same time, need to apply a sanitization process to "blank out" information considered sensitive, either directly or because of the sensitive information it would allow the recipient to infer.

Open Problems

The problem of preserving the privacy of the respondents in microdata publication has been widely studied. However, there are still different open issues that need to be further investigated such as multiple data releases, external knowledge of recipients, and definition of simple privacy and utility measures.

Recommended Reading

1. Ciriani V, De Capitani di Vimercati S, Foresti S, Samarati P (2007) Microdata protection. In: Yu T, Jajodia S (eds) Security in decentralized data management. Springer, Berlin Heidelberg
2. Federal committee on statistical methodology. Statistical policy working paper 22. Report on statistical disclosure limitation methodology. Washington, DC, May 1994
3. Samarati P (2001) Protecting respondents' identities in microdata release. IEEE Trans Knowl Data Eng 13(6):1010–1027

Microdata Statistical Disclosure Control

►Microdata Masking Techniques

Miller–Rabin Probabilistic Primality Test

Moses Liskov
Department of Computer Science, The College of William and Mary, Williamsburg, VA, USA

Synonyms

Miller-Rabin test

Related Concepts

►Fermat Primality Test; ►Fermat's Little Theorem; ►Modular Arithmetic; ►Primality Test; ►Prime Number

Definition

The Miller–Rabin ►probabilistic primality test is a probabilistic algorithm for testing whether a number is a ►prime number using modular exponentiation, ►Fermat's little theorem, and the fact that the only square roots of 1 modulo a prime are ±1.

Background

The Miller–Rabin test was described initially by Miller [2]. Rabin provided further analysis [3], hence the name.

Theory and Applications

One property of primes is that any number whose square is congruent to 1 modulo a prime p must itself be congruent to 1 or −1. This is not true of composite numbers. If a number n is the product of k distinct odd prime powers, then there will be 2^k distinct "square roots" of 1 modulo n. For example, there are four square roots of 1 modulo 77. The roots must be either 1 or −1 modulo 7, and either 1 or −1 modulo 11, since 7 and 11 divide 77. In order to solve these square roots, one must solve sets of equations like these:

$$a \equiv 1 \bmod 7, a \equiv 1 \bmod 11$$

$$b \equiv -1 \bmod 7, b \equiv 1 \bmod 11$$

$$c \equiv 1 \bmod 7, c \equiv -1 \bmod 11$$

$$d \equiv -1 \bmod 7, d \equiv -1 \bmod 11$$

The solutions in this case are $a \equiv 1 \bmod 77$, $b \equiv 34 \bmod 77$, $c \equiv 43 \bmod 77$, and $d \equiv -1 \equiv 76 \bmod 77$. (This is a simple application of the ►Chinese remainder theorem.)

The Miller–Rabin test uses this fact about composite numbers to test if a number is composite. A single round of Miller–Rabin tests where a given base a is a "witness" to the compositeness of n, by computing $a^{n-1} \bmod n$. This is the same computation as in the ►Fermat primality test, but that the test fails to detect compositeness of n if n is a Carmichael number. The Miller–Rabin test performs this computation as a series of squarings and checks after each squaring whether a square root of 1 other than 1 or −1 mod n is found. If so, the number n is composite; these additional checks also catch Carmichael numbers.

The procedure is the following, where s, the number of rounds, is a parameter:

1. On input n, output PRIME if $n = 2$ and COMPOSITE if $n > 2$ but n is even. Otherwise, find k such that $n-1 = q2^k$ where q is odd.
2. For $j = 1$ to s, do:
   (a) Generate a random base a between 2 and $n - 2$ with $\gcd(a, n) = 1$.
   (b) Compute $b = a^q \bmod n$.
   (c) For $i = 1$ to k:
      (i) Compute $b' = b^2 \bmod n$.
      (ii) If $b' \equiv 1 \bmod n$ and $b \not\equiv \pm 1 \bmod n$, output COMPOSITE.
      (iii) Set $b = b'$.
   (d) If $b \not\equiv 1 \bmod n$, output COMPOSITE.
3. Output PROBABLY PRIME.

Note that when this algorithm outputs PRIME or COMPOSITE, the answer is always correct. When this algorithm outputs PROBABLY PRIME, there is a chance of failure.

For any odd composite n, at least $\frac{3(n-1)}{4}$ of the bases of a are Miller–Rabin witnesses that n is composite (a good presentation of a proof for the simpler bound $\frac{n-1}{2}$ is in [1]). Each round of the Miller–Rabin test thus gives at least a 3/4 probability of finding a witness, if n is composite. These probabilities are independent, so if we run the Miller–Rabin test with s rounds, then the probability that n is composite and we never find a compositeness witness is at most 2^{-2s}.

The running time of the Miller–Rabin primality test, for failure threshold ϵ, on an input n, is $\Theta((-\log \epsilon) \log n)$ modular multiplications.

Recommended Reading

1. Cormen TH, Leiserson CE, Rivest RL, Stein C (2001) Introduction to algorithms. MIT Press, Cambridge

2. Miller GL (1976) Riemann's hypothesis and tests for primality. J Comput Syst Sci 13:300–317
3. Rabin MO (1980) Probabilistic algorithm for testing primality. J Number Theory 12:128–138

Miller-Rabin Test

►Miller–Rabin Probabilistic Primality Test

MILS

►Multiple Independent Levels of Security

Minimal Polynomial

Anne Canteaut
Project-Team SECRET, INRIA Paris-Rocquencout, Le Chesnay, France

Related Concepts

►Berlekamp–Massey Algorithm; ►Linear Complexity; ►Linear Feedback Shift Register; ►Stream Cipher

Definition

The *minimal polynomial* of a linear recurring sequence $\mathbf{s} = (s_t)_{t\geq 0}$ of elements of $\mathbf{F}_q$ is the polynomial P in $\mathbf{F}_q[X]$ of lowest degree such that $(s_t)_{t\geq 0}$ is generated by the ►linear feedback shift register (LFSR) with characteristic polynomial P. In other terms, $P = \sum_{i=0}^{L-1} p_i X^i + X^L$ is the characteristic polynomial of the linear recurrence relation of least degree satisfied by the sequence:

$$s_{t+L} + \sum_{i=0}^{L-1} p_i s_{t+i} = 0, \; t \geq 0.$$

The minimal polynomial of a linear recurring sequence **s** is monic and unique; it divides the characteristic polynomial of any LFSR which generates **s**. The degree of the minimal polynomial of **s** is called its ►linear complexity. The period of the minimal polynomial of **s** is equal to the least period of **s**. (►linear feedback shift register for further details).

The minimal polynomial of a linear recurring sequence with linear complexity Λ can be recovered from any 2Λ consecutive terms of the sequence by the ►Berlekamp–Massey algorithm.

Minimal Privilege

►Least Privilege

MIPS-Year

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Moore's Law; ►RSA Factoring Challenge

Definition

A *MIPS-year* is the amount of work performed in one year by a computer operating at the rate of one million operations per second, or approximately 2^{45} operations.

Background

The term "MIPS" is fairly old in the computer industry; Digital Equipment Corporation's VAX-11/780 is often considered the benchmark of a 1-MIPS machine. An early use of the term "MIPS-year" in cryptography may be found in a 1991 letter from Rivest to NIST regarding the security of the then-proposed ►Digital Signature Standard, a revised version of which is published as part of [1].

Theory

A *MIPS-year* is perhaps the "standard" measure of computational effort in cryptography: it refers to the amount of work performed, in one year, by a computer operating at the rate of one million operations per second (1 MIPS). The actual type of operation is undefined but assumed to be a "typical" computer operation. A MIPS-year is thus approximately 2^{45} operations.

The MIPS-year is a convenient measurement, but not a perfect one. In practice, there is no "typical" computer operation, and the actual difficulty of an effort must also consider other factors such as the cost of hardware and the amount of memory required. (See Silverman [2] for discussion of some of these issues.) Research into the difficulty of factoring 1024-bit RSA moduli has taken a more precise approach by giving a specific hardware design and estimating both the cost of the hardware involved and the number of operations (►Factoring Circuits, ►TWIRL). The MIPS-year nevertheless remains a helpful guideline.

Applications

The difficulty of solutions to the cryptographic problems (as illustrated, for instance, ►RSA Factoring Challenge as well as challenges involving the ►Data Encryption Standard) is usually given in MIPS-years, as a rough measure of effort independent of the actual computers involved. For instance, the RSA-512 benchmark took about 8000 MIPS-years, or approximately 2^{58} operations, distributed across a large number of computers. (This is somewhat less than the number of operations to search for a 56-bit DES key, as multiple operations are required to test each DES key.)

Recommended Reading

1. Rivest RL, Hellman ME, Anderson JC, Lyons JW (1992) Responses to NIST's proposal. Commun ACM 35(7):41–54
2. Silverman R (1999) Exposing the mythical MIPS-year. IEEE Comput 32(8):22–26

Miss-in-the-Middle Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers

Definition

Following the idea behind the ►meet-in-the-middle approach, the miss-in-the-middle attack is one of the techniques to construct *distinguishers* for the ►impossible differential attack. The idea is that one finds two events that propagate half way through the cipher top and bottom with certainty, but which do not match in the middle. This results in an event which is impossible for the full cipher, i.e., has zero probability. A typical tool for constructing such events would be *truncated differentials*. Note that it is sufficient that events contradict each other in a single bit in the middle.

Background

This technique was first introduced in the papers by Biham et al. [1, 2] to cryptanalyze round-reduced versions of ►Skipjack, IDEA, and Khufu.

Recommended Reading

1. Biham E, Biryukov A, Shamir A (1999) Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In: Stern J (ed) Advances in cryptology – eurocrypt'99. Lecture notes in computer science, vol 1593. Springer, Berlin, pp 12–23
2. Biham E, Biryukov A, Shamir A (1999) Miss in the middle attacks on IDEA and Khufu. In: Knudsen LR (ed) Fast software encryption, FSE'99. Lecture notes in computer science, vol 1636. Springer, Berlin, pp 124–138

Mix Networks

Matthew K. Franklin
Computer Science Department, University of California, Davis, CA, USA

Definition

The basic functionality of a mix network is to provide *sender* ►*anonymity*, i.e., the identity of the originator of a message is difficult or impossible to discern for any given message delivered to any given recipient. A mix network can also provide *receiver anonymity*, i.e., the identity of the intended recipient of a message is difficult or impossible to discern for any given message originating from any given sender.

Background

It is not difficult to imagine scenarios in which message secrecy in communication is desirable. In such scenarios, ►encryption is a crucial tool. Unfortunately, encryption of message contents by itself may not be sufficient. Even if the contents of sensitive messages are protected, much can be inferred merely by the fact that one party is sending a message to another party. If an authoritarian regime already suspects that one party to a communication is a dissident, then the other parties to the communication become suspect as well. Accessing a crisis hotline or a patent database is a strong clue about the intentions of the user, even if the exact wording of the query remains secret. In settings where encryption is rare, the mere fact that certain messages are encrypted may cause increased scrutiny. In 1981, Chaum [3] published a beautifully simple and elegant method to protect the identities of communicating parties: "mix networks."

Theory and Applications

Chaum's idea can be described using the simplest mix network, which consists of a single mix. If Alice wants to send a message M to Bob, she first encrypts it using Bob's public key (►Publickey Cryptography): $C_1 = E(M, PK_{Bob})$. Then she re-encrypts this ciphertext (together with Bob's

name!) using the mix's public key: $C_2 = E$ (C_1 "Bob," PK_{Mix}). Then the following steps occur:

1. Alice sends C_2 to the mix.
2. The mix decrypts C_2 to recover C_1 and "Bob."
3. The mix sends C_1 to Bob.
4. Bob decrypts C_1 to recover the message M.

The identity of the originator (Alice) is protected if many senders are using the same mix at the same time. The identity of the recipient (Bob) is protected if many receivers have messages routed to them through the same mix at the same time. Of course, the mix knows who is communicating to whom. That is, there is perfect linkability by the mix of originator to recipient for every message that passes through the mix.

A general mix network is constructed similarly, except that Alice re-encrypts several times using a "chain" (or "cascade") of mixes. For example, a chain of three mixes would have Alice compute:

$$\begin{aligned} C_1 &= E(M, PK_{Bob}), \\ C_2 &= E(C_1 \cdot \text{"Bob"}, PK_{Mix1}), \\ C_3 &= E(C_2 \cdot \text{"Mix1"}, PK_{Mix2}), \\ C_4 &= E(C_3 \cdot \text{"Mix2"}, PK_{Mix3}), \end{aligned} \tag{1}$$

Then Alice would send C_4 to Mix3, and Mix3 would send C_3 to Mix2, and Mix2 would send C_2 to Mix1, and Mix1 would send C_1 to Bob. Notice that now the mixes would have to collude to know who is communicating to whom.

Of course, this is just the high-level description of a mix network. In practice, a variety of attacks on ►anonymity are possible, and various design countermeasures would have to be incorporated (see also [1]):

►*Timing Attacks*: Careful observation of the timing of inputs and outputs of a single mix could enable an eavesdropper to link specific inbound and outbound messages. One defense against this is for a mix to delay the forwarding of messages until a certain number of them can be sent at the same time ("batching").

Message Length Attacks: Encryption by itself does not necessarily conceal the size of the plaintext message. If certain messages passing through the mix network differ in size, then these might be distinguishable by an outside attacker. Defenses include splitting large messages ("fragmentation") and lengthening short messages ("padding").

Absence of Communication Attacks: Certain suspects can be eliminated from consideration as potential senders of a message simply because they were idle during some relevant time period, i.e., sent no encrypted messages to a particular mix. One defense against this is for senders to continue to send null ("dummy") messages when they have nothing to communicate.

Abundance of Communication Attacks: If an adversary can inject a lot of messages into the mix network, then only a small amount of legitimate messages can be routed through the mixes at the same time. The adversary can choose his messages so that they are easily recognized as they pass through the network, and thus he may be able to infer quite a lot about the origin and destination of the few remaining messages. This is closely related to a denial of service attack, and similar defenses are possible. For example, one could limit and balance the rate at which any given sender can route messages through any given mix ("fair allocation").

Chaum's original work included several other interesting extensions. There was a technique for the receiver to reply to an anonymously transmitted message. He also showed how a mix network can be combined with pseudonyms to achieve a "general purpose" untraceable mail system. Furthermore, there was a method for the sender to specify a different chain of mixes for each message.

Chaum's mix network ideas have been implemented numerous times. Notable instances include Freedom [2], Onion Routing [5], Babel [6], and the Cypherpunk remailer system. Serjantov et al. [11] provide a good taxonomy of mix implementations and their properties.

Open Problems

The Crowds system of Reiter and Rubin [8] is a kind of mix network with on-the-fly randomized decisions for how many mixes a message should pass through. In fact, parties in their scheme act as both message initiators and the mixes themselves. In the Crowds protocol, some set of parties form a group called a "jondo." When a sender initiates a message, it is mixed by passing it along a path of other parties in the jondo. The length and constituency of this path is chosen randomly. Specifically, each party in the path makes a randomized decision whether to end the path. If the path is not to be ended, a second randomized decision chooses another jondo member to receive the message next. Perfect concealment is not possible for the Crowds system, but a level of "probable innocence" can be achieved.

Crowds is also vulnerable to a "predecessor attack" [8, 12]. An attacker joins a jondo and keeps track of how often any other jondo member immediately precedes the attacker in a path. Suppose that a single sender is involved in a number of related transmissions, e.g., if a jondo member is using Crowds to anonymously surf the web and often returns to the same website. Then the attacker will be included in some of the paths for these related

transmissions. The attacker's immediate predecessor in each of these paths could be any member of the jondo, but the most frequent occupant of this position will be the original sender.

Mix networks can simplify the design of a cryptographic ►protocol for conducting a secret ballot election [4]. The basic idea is that each eligible voter encrypts his ballot using the public key of the tallying authority. All of the encrypted ballots from eligible voters pass through a mix network to the tallying authority. The tallying authority decrypts all of the received messages, throws away the ones that are not well-formed ballots, and then determines the outcome of the election. The privacy of each individual voter's choices is ensured by the proper functioning of the anonymizing mixes.

For applications such as a secret ballot election, it is reasonable to assume that all of the mix network inputs are available for processing at the same time. This means that the mix network design can be "synchronous." By contrast, Chaum's mix network design is asynchronous, since this is a more reasonable assumption for his motivating application of hiding traffic patterns in ongoing communication. The "re-encryption mix net" [7] is a synchronous design that makes use of more sophisticated cryptographic tools to achieve robustness despite the failure of some of the mix servers. Some of the fastest mix-based election schemes are based on this design.

Verifiable mix protocols [9] are a variant of the basic mix protocol in which correct functioning of any given mix can be publicly verified by any external observer. The primary motivation of verifiable mix protocols is to enhance the security of mix-based election protocols. Typically, the mix generates a short "proof" of correctness that can be checked against the encrypted inputs and encrypted outputs of the mix. Another recent approach to verifiable mixes is based on a "cut-and-choose" (challenge/response) approach [7].

Recommended Reading

1. Back A, Moller U, Stiglic A (2001) Traffic analysis and trade-offs in anonymity providing systems. In: Moskowitz IS (ed) Proceedings of the 4th information hiding workshop. Lecture notes in computer science, vol 2137. Springer, Berlin, pp 243–254
2. Boucher P, Shostack A, Goldberg I (2000) Freedom 2.0 system architecture. Zero knowledge systems (white paper)
3. Chaum D (1981) Untraceable electronic mail, return addresses, and digital pseudonyms. Commun ACM 24(2):84–88
4. Fujioka A, Okamoto T, Ohta K (1992) A practical secret voting scheme for large scale elections. In: Seberry J, Zheng Y (eds) Advances in cryptology – AUSCRYPT'92. Lecture notes in computer science, Gold Coast, Queensland, vol 718. Springer, Berlin, pp 244–251
5. Goldschlag D, Reed M, Syverson P (1999) Onion routing. Commun ACM 42(2):39–41
6. Gulcu C, Tsudik G (1996) Mixing e-mail with BABEL. In: Proceedings of symposium on network and distributed system security. IEEE Computer Society, Washington, DC
7. Jakobsson M, Juels A, Rivest R (2002) Making mix nets robust for electronic voting by partial checking. In: Proceedings of 11th USENIX security symposium, Berkeley, 2002
8. Park C, Itoh K, Kurosawa K (1993) Efficient anonymous channel and all/nothing election scheme. In: Helleseth T (ed) Advances in cryptology – EUROCRYPT'93. Lecture notes in computer science, vol 765. Springer, Berlin, pp 248–259
9. Reiter M, Rubin A (1998) Crowds: anonymity for web transactions. ACM Trans Inform Syst Secur 1(1):66–92
10. Sako K, Kilian J (1995) Receipt-free mix-type voting scheme – a practical solution to the implementation of a voting booth. In: Guillou L, Quisquater J-J (eds) Advances in cryptology – EUROCRYPT'95. Lecture notes in computer science, vol 021. Springer, Berlin, pp 393–403
11. Serjantov A, Dingledine R, Syverson P (2002) From a trickle to a flood: active attacks on several mix types. In: Petitcolas FAP (ed) Proceedings of information hiding workshop, Lecture notes in computer science, vol 2578. Springer, Berlin, pp 36–52
12. Wright M, Adler M, Levine B, Schields C (2002) An analysis of the degradation of anonymous protocols. In: Proceedings of symposium on network and distributed system security (NDSS), San Diego

ML-Sequence

►Maximal-Length Sequences

Mobile Payments

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Definition

Mobile payments are any type of payments which are enabled by the usage of a mobile device, e.g., a mobile phone, a smart phone, a PDA, etc. . .

Background

The mobile device is becoming a universal platform with which people will manage their financial services such as mobile banking, mobile payments, brokerage, direct mobile billing, mobile remittances, etc. . . Mobile payments are payment transactions which are initiated through the usage of a mobile phone which are basically of three types: card payments, debit, or credit transfers. This means that the transfer of funds from a payer to a payee is initiated by using a mobile device. As such the mobile device is only to

be considered as an access channel to a "classic" payment transaction such as a card payment or a debit/credit transfer.

Theory

Basically two main classes of mobile payments can be considered: mobile contactless payments, also referred to as proximity payments, and mobile remote payments.

In the case of proximity payments, it is a mobile device initiated payment where the payer and payee (and/or his/her equipment) are in the same location and communicate directly with each other using contactless radio technologies, such as NFC (RFID), Bluetooth or infrared for data transfer.

For remote payments, it is a mobile device initiated payment where the transaction is conducted over telecommunication networks such as GSM or Internet, and can be made independently from the payer's location (and/or his/her equipment).

In order to guarantee the security of the mobile payment, most banks require that the so-called mobile payment application and the personalisation data are stored on a Secure Element in the mobile device (see [1]). Also specific (security) requirements for the mobile devices used for executing payments apply (see [4]).

Applications

A variety of applications of mobile payments exist. A description of use cases may be found in documents provided by [1] and [3].

Open Problems and Future Directions

Mobile payments are to be considered as a rapidly adopting alternative payment method to cash and cheques to pay for a wide range of services and digital or hard goods.

Recommended Reading

1. http://www.mobeyforum.org/
2. http://www.emvco.com/mobile.aspx
3. http://www.europeanpaymentscouncil.eu/content.cfm?page=sepa_mobile_payments
4. http://www.gsmworld.com/our-work/mobile_lifestyle/mobile_money/index.htm

Mobile Wallet

Marijke De Soete
Security4Biz, Oostkamp, Belgium

Related Concepts

►Electronic Wallet

Definition

A mobile wallet is a service allowing the wallet holder to securely store, access, manage, and use identification and payment instruments related information in order to initiate payments from any mobile device.

Theory

The mobile wallet may be considered as the electronic equivalent of a physical wallet which carries identification cards, payment cards, money, and personal items of value. This means that the mobile wallet may store identification information, virtual cards, and other personal items of value. It consists of hardware and software that ensures the secure storage and management of the data and wallet applications and the secure usage of the wallet applications including the authentication of the wallet holder.

Two basic implementations are possible. The wallet (hardware, software, data) is implemented in the end-user mobile device used by the wallet holder, or the wallet is implemented as a service on a remote server and is accessible from any (mobile) device used by the wallet holder.

Applications

The mobile wallet aggregates payment methods and makes them available as one-click payment options on a user's mobile device (e.g., a mobile phone). When the user proceeds to pay for goods or service, the mobile wallet will present the registered payment methods as payment options for completing the payment transaction. Not only does it allow users on the move to access financial services/accounts, but also it plays an integral part in the development of digital commerce.

Recommended Reading

1. http://www.gsmworld.com

Modes of Operation of a Block Cipher

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC,
Katholieke Universiteit Leuven and IBBT,
Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers

A n-bit ►block cipher with a k-bit ►key is a set of 2^k bijections on n-bit strings. A block cipher is a flexible building block; it can be used for encryption and ►authenticated

encryption to construct ►MAC algorithms and ►hash functions [2].

When a block cipher is used for confidentiality protection, the security goal is to prevent a passive ►eavesdropper with limited computational power to learn any information on the plaintext (except for maybe its length). This eavesdropper can apply the following attacks: ►known plaintext attacks, ►chosen plaintext attacks, and ►chosen ciphertext attacks.

Applications need to protect the confidentiality of strings of arbitrary length. A mode of operation of a block cipher is an algorithm which specifies how one has to apply an n-bit block cipher to achieve this. One approach is to pad the data with a padding algorithm such that the bit-length of the padded string is a multiple t of n bits, and to define a mode which works on t n-bit blocks. For example, one always appends a "1"-bit followed by as many "0" bits as necessary to make the length of the resulting string a multiple of n. An alternative is to define a mode of operation that can process data in blocks of $j \le n$ bits.

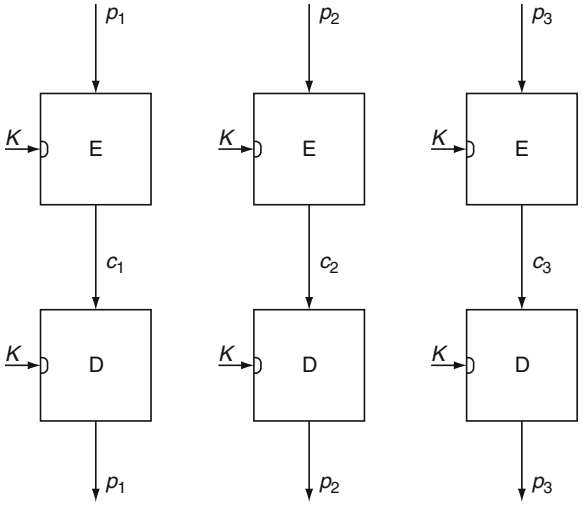
We first discuss the five modes of operation which have been defined in the FIPS [12] (also [22]) and ISO/IEC [16] standards: the ECB mode, the CBC mode, the OFB mode, the CTR mode, and the CFB mode. Next, we discuss some alternative modes that have been defined for ►triple-DES and modes which allow to encrypt values from finite sets.

We use the following notation: $E_K(p_i)$ denotes the encryption with a block cipher of the n-bit plaintext block p_i with the key K; similarly, $D_K(c_i)$ denotes the decryption of the iphertext c_i. The operation $rchop_j(s)$ returns the rightmost j bits of the string s, and the operation $lchop_j(s)$ returns the leftmost j bits. The symbol $\|$ denotes concatenation of strings and $\oplus$ denotes addition modulo 2 (exor).

The Electronic Code Book (ECB) Mode: The simplest mode is the ECB (Electronic Code Book) mode. After padding, the plaintext p is divided into t n-bit blocks p_i and the block cipher is applied to each block; the decryption also operates on individual blocks (Fig. 1):

$$c_i = E_K(p_i) \quad \text{and} \quad p_i = D_K(c_i), \quad i = 1, \ldots, t.$$

Errors in the ciphertext do not propagate beyond the block boundaries (as long as these can be recovered). However, the ECB mode is the only mode covered in this entry which does not hide patterns (such as repetitions) in the plaintext. Usage of this mode should be strongly discouraged. In the past, the ECB mode was sometimes recommended for the encryption of keys; however, ►authenticated encryption would be much better for this application (or the ►Rijndael/AES key wrapping algorithm proposed by NIST).



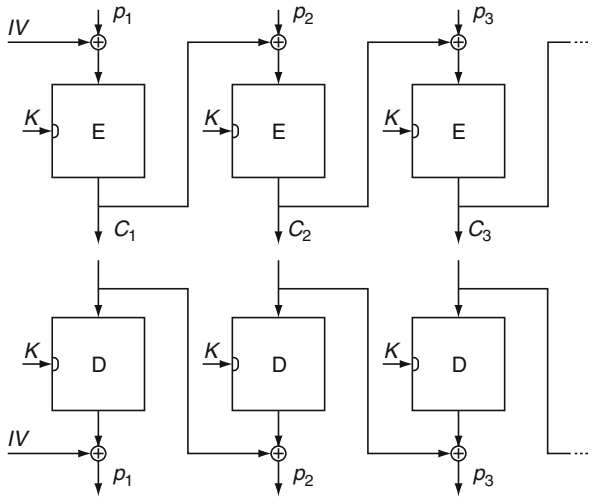
Modes of Operation of a Block Cipher. Fig. 1 The ECB mode of a block cipher

The Cipher Block Chaining (CBC) mode: The most popular mode of operation of a block cipher is the CBC (Cipher Block Chaining) mode. The plaintext p is divided into t n-bit blocks p_i. This mode adds (modulo 2) to a plaintext block the previous ciphertext block and applies the block cipher to this result (Fig. 2):

$$c_i = E_K(p_i \oplus c_{i-1}),$$
$$p_i = D_K(c_i \oplus c_{i-1}), \qquad i = 1, \ldots, t.$$

Note that in the CBC mode, the value c_{i-1} is used to randomize the plaintext; this couples the blocks and hides patterns and repetitions. To enable the encryption of the first plaintext block ($i = 1$), one defines c_0 as the initial value IV, which should be randomly chosen and transmitted securely to the recipient. By varying this IV, one can ensure that the same plaintext is encrypted into a different ciphertext under the same key, which is essential for secure encryption. The IV plays a similar role in the OFB, CTR, and CFB modes.

The CBC decryption has a limited error propagation: Errors in the ith ciphertext block will garble the ith plaintext block completely, and will be copied into the next plaintext block. The CBC decryption allows for parallelism and random access: If necessary, one can decrypt only a small part of the ciphertext. However, the encryption mode is a serial operation. To overcome this restriction, ISO/IEC 10116 [16] has defined a variant of the CBC mode which divides the plaintext into r parallel streams and applies



Modes of Operation of a Block Cipher. Fig. 2 The CBC mode of a block cipher

the CBC mode to each of these streams. This requires, however, r different IV values.

A security proof of the CBC mode (with random and secret IV) against an adversary who has access to chosen plaintexts has been provided by Bellare et al. [3]; it shows that if the block cipher is secure in the sense that it is hard to distinguish it from a random permutation, the CBC mode offers secure encryption in the sense that the ciphertext is random (which implies that it does not provide the opponent additional information on the plaintext). The security result breaks down if the opponent can obtain approximately $q = 2^{n/2}$ plaintext/ciphertext pairs due to a matching ciphertext attack [18]. This can be seen as follows. Note that the ciphertext blocks c_i are random n-bit strings. After observing q n-bit ciphertext blocks, one expects to find approximately $q^2/2^{n+1}$ pairs of matching ciphertexts, that is, indices (v, w) with $c_v = c_w$ (also the ▶birthday paradox). As a block cipher is a permutation, this implies that the corresponding plaintexts are equal, or $p_v \oplus c_{v-1} = p_w \oplus c_{w-1}$ which can be rewritten as $p_v \oplus p_w = c_{v-1} \oplus c_{w-1}$. Hence, each pair of matching ciphertexts leaks the sum of two plaintext blocks. To preclude such a leakage, one needs to impose that $q \ll 2^{(n+1)/2}$ or $q = \alpha \cdot 2^{n/2}$ where α is a small constant (say 10^{-3}, which leads to a collision probability of 1 in two million). If this limit is reached, one needs to change the key. Note that the proof only considers security against chosen plaintext attacks; the CBC mode is not secure if chosen ciphertext attacks are allowed. The security against these attacks can be obtained by using ▶authenticated encryption.

For some applications, the ciphertext should have exactly the same length as the plaintext; hence padding, methods cannot be used. Two heuristic constructions have been proposed to address this problem; they are not without problems (both leak information in a ▶chosen plaintext setting). A first solution encrypts the last incomplete block p_t (of $j < n$ bits) in OFB mode (cf. Section "The output mode"):

$$c_t = p_t \oplus \mathrm{rchop}_j(E_K(c_{t-1})).$$

A second solution is known as *ciphertext stealing* [21]: One appends the rightmost $n-j$ bits of c_{t-1} to the last block of j bits p_t, to obtain a new n-bit block:

$$c_{t-1} = E_K(p_{t-1} \oplus c_{t-2}),$$
$$c_t = E_K(p_t \parallel \mathrm{rchop}_{n-j}(c_{t-1})).$$

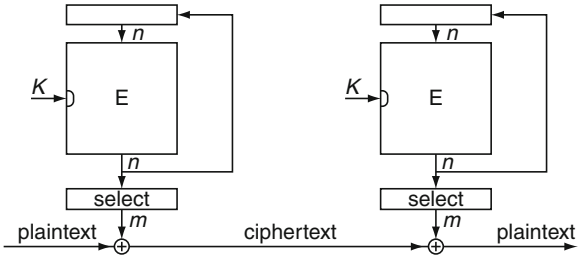
For the last two blocks of the ciphertext, one keeps only the leftmost j bits of c_{t-1} and n bits of c_t. This variant has the disadvantage that the last block needs to be decrypted before the one but last block.

It turns out that the common padding methods are vulnerable to side channel attacks that require chosen ciphertexts: An attacker who can submit ciphertexts of her or his choice to a decryption oracle can obtain information on the plaintext by noting whether or not an error message is returned stating that the padding is incorrect. This was first pointed out for symmetric encryption by Vaudenay in [24]; further results on concrete padding schemes can be found in [8, 9, 23]. The specific choice of the padding rule makes a difference: For example, the simple padding rule described in the introduction seems less vulnerable. Moreover, the implementation can to some extent preclude these attacks, for example, by interrupting the session after a few padding errors. However, the preferred solution is the use of ▶authenticated encryption.

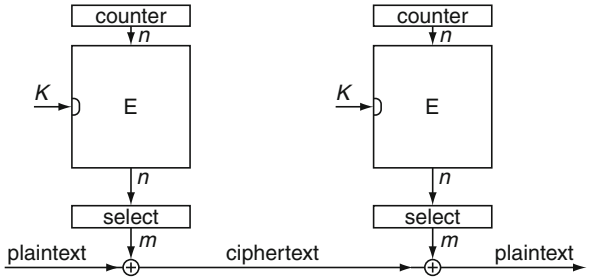
The Output FeedBack (OFB) Mode: The OFB mode transforms a block cipher into a ▶synchronous stream cipher. This mode uses only the encryption operation of the block cipher. It consists of a finite state machine, which is initialized with an n-bit initial value or $s_0 = IV$. The state is encrypted and the encryption result is used as key stream and fed back to the state (also Fig. 3):

$$s_i = E_K(s_{i-1}) \quad \text{and} \quad c_i = p_i \oplus s_i, \quad i = 1, 2, \ldots$$

Treating an incomplete last block in the OFB mode is very simple: One selects the leftmost m bits of the last key word. The OFB mode can also be applied when the strings p_i and c_i consist of $m < n$ bits; in that case, one uses only the m leftmost bits of each key word s_i. This results in a performance penalty with a factor n/m.



Modes of Operation of a Block Cipher. Fig. 3 The m-bit OFB mode of an n-bit block cipher



Modes of Operation of a Block Cipher. Fig. 4 The m-bit CTR mode of an n-bit block cipher

It is essential for the security of the OFB mode that the key stream does not repeat. It can be shown that the average period equals $n \cdot 2^{n-1}$ bits [14] and that the probability that an n-bit state lies on a cycle of length $< c$ is equal to $c/2^n$. This implies that after $2^{n/2}$ n-bit blocks, one can distinguish the output of the OFB mode from a random string (in a random string one expects to see repetitions of n-bit blocks after $2^{n/2}$ blocks as a consequence of the ►birthday paradox, but it is highly unlikely that such repetitions occur in an OFB key stream). This suggests that one should rekey the OFB mode after $\alpha \cdot 2^{n/2}$ n-bit blocks for a small constant α. A repetition could also be induced in a different way: If IV is chosen uniformly at random for every message, the ►birthday paradox implies that IV values will repeat with high probability after approximately $2^{n/2}$ messages. The impact of such a repetition is dramatic, since it will leak the sum of all the plaintext blocks of the two messages encrypted with this IV value (for simplicity, it is assumed here that all messages have equal length).

The main advantage of the OFB mode is that it has no error propagation: Errors in the ith ciphertext bit will only affect the ith plaintext bit. The OFB mode does not allow for parallelism or random access.

It can be shown that the OFB mode is secure against ►chosen plaintext attacks if the block cipher is secure in the sense that it is hard to distinguish it from a random permutation. The proof requires that one changes the key after $\alpha \cdot 2^{n/2}$ n-bit blocks for small α (say 10^{-3}).

Note that an early draft of [12] included a variant of the OFB mode where only $m < n$ bits were fed back to the state, which acted as a shift register. However, this variant of the OFB mode has an average period of about $n \cdot 2^{n/2}$ bits [11]. This variant was removed because of this weakness.

The CounTeR (CTR) Mode: The CTR mode is another way to transform a block cipher into a ►synchronous stream cipher. As the OFB mode, this mode only uses the encryption operation of the ►block cipher. It consists of a finite state machine, which is initialized with an n-bit integer IV. The state is encrypted to obtain the key stream; the state is updated as a counter mod 2^n (also Fig. 4):

$$c_i = p_i \oplus E_K(< (IV + i) \bmod 2^n >), \quad i = 1, 2, \ldots$$

The mapping $<\cdot>$ converts an n-bit integer to an n-bit string. The processing of an incomplete final block or of shorter blocks is the same as for the OFB mode.

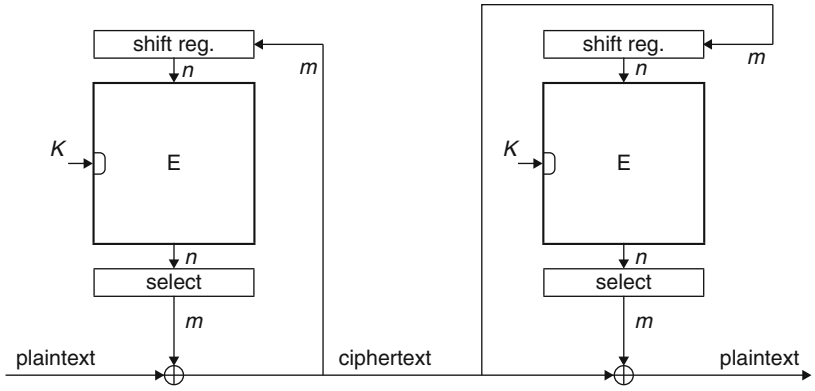
The period of the key stream is exactly $n \cdot 2n$ bits. This implies that after $2^{n/2}$ n-bit blocks, one can distinguish the output of the CTR mode from a random string (as for the OFB mode). This suggests that one should rekey the CTR mode after $\alpha \cdot 2^{n/2}$ n-bit blocks for a small constant α. A repeating value of IV has the same risks as for the OFB mode.

As the OFB mode, the CTR mode has no error propagation. Moreover, the CTR mode allows for parallelism and for random access in both encryption and decryption.

It can be shown that the CTR mode is secure against ►chosen plaintext attacks if the block cipher is secure in the sense that it is hard to distinguish it from a random permutation [3]. Again it is recommended to change the key after $\alpha \cdot 2^{n/2}$ n-bit blocks for small α (say 10^{-3}).

The Cipher FeedBack (CFB) Mode: The CFB mode transforms a block cipher into a ►self-synchronizing stream cipher. As the OFB and CTR mode, this mode only uses the encryption operation of the block cipher. It consists of a finite state machine, which is initialized with an n-bit initial value $s_0 = IV$. The state is encrypted and the leftmost m bits of the result are added to the m-bit plaintext block; the resulting ciphertext is fed back to the state (also Fig. 5):

$$c_i = p_i \oplus \mathrm{lchop}_m(E_K(s_{i-1})),$$
$$s_i = \mathrm{lchop}_{n-m}(s_{i-1}) \parallel c_i, \quad i = 1, 2, \ldots$$



Modes of Operation of a Block Cipher. Fig. 5 The m-bit CFB mode of an n-bit block cipher

Treating an incomplete last block in the CFB mode is very simple: One selects the required number of bits from the output of the block cipher. The CFB mode is a factor n/m times slower than the CBC mode, since only m bits are used per encryption operation. In practice, one often uses $m = 1$ and $m = 8$; this results in a significant speed penalty.

It can be shown that the CFB mode is secure against chosen plaintext attacks if the block cipher is secure in the sense that it is hard to distinguish it from a random permutation. A matching ciphertext attack also applies to the CFB mode (cf. Section "The cipher . . . mode") [19]; the analysis is more complex since one can now consider n-bit ciphertext blocks which are shifted over m positions. To preclude leakage of information on the plaintexts, one needs to impose that the number q of m-bit ciphertext blocks to which an opponent has access satisfies $q \ll 2^{(n+1)/2}$ or $q = \alpha \cdot 2^{n/2}$, where α is a small constant (say 10^{-3}). If this limit is reached, one needs to change the key.

The CFB decryption has a limited error propagation: Errors in the ith ciphertext block will be copied into the ith plaintext block; about n subsequent plaintext bits will be completely garbled, since the error will stay for n/m steps in the state register s. From then on, the decryption will recover. Moreover, if a multiple of m bits of the ciphertext are lost, synchronization will return as soon as n consecutive correct ciphertext bits have been received. Particularly when $m = 1$, this is very attractive, since this allows for a recovery after loss of an arbitrary number of bits. The CFB decryption allows for random access and parallel processing, but the encryption process is serial.

ISO/IEC 10116 [16] specifies two extensions of the CFB mode: A first extension allows to encrypt plaintext blocks of length $m' < m; m - m'$ "1" bits are then prepended to the ciphertext c_i before feeding it back to the state. This mode offers a better speed, but increases the risk of a matching ciphertext attack. For example, if $n = 64$, $m = 8$, and $m' = 7$, on expects repetitions of the state after 2^{28} blocks, since the 64-bit state always contains eight "1" bits. A second extension allows for a larger state s (for example of $r \cdot n$ bits). This allows for parallel processing (with r processors) in the CFB encryption, at the cost of r IVs, a delayed error propagation and a slower synchronization.

Yet another variant of the CFB mode [1] improves the efficiency by using all the bits of $E_K(s_{i-1})$. A new encryption is only calculated if all bits of the n-bit block have been used or if a specific pattern of fixed length is observed in the ciphertext. The latter property allows resynchronization: the shorter the pattern, the faster the resynchronization, but the slower the performance.

Other Modes of Operation: In the early 1990s, modes for ►multiple encryption of DES (►Data Encryption Standard) were analyzed. The simplest solution is to replace DES by ►triple-DES and to use triple-DES in one of the five modes discussed above. For triple-DES, these solutions are known as the *outer modes* [17]. However, their disadvantage is that one can only encrypt $\alpha \cdot 2^{n/2}$ blocks with a single key for small α (for example, due to matching ciphertext attacks on CBC and CFB mode). This motivated research on *inner modes*, also known as interleaved or combined modes, where the modes themselves are considered as primitives (e.g., inner-CBC for triple-DES consists of three layers of single-DES in CBC mode). Biham has analyzed all the 36 double and 216 triple interleaved modes [4, 5], where each layer consists of ECB, OFB, CBC, CFB, and the inverses of CBC and CFB. His goal is to recover the secret key (total break). He notes that by allowing chosen

plaintext and chosen ciphertext attacks, "all triple modes of operation are theoretically not much more secure than a single encryption." The most secure schemes in this class require for DES 2^{67} chosen plaintexts or ciphertexts, 2^{75} encryptions, and 2^{66} storage. Biham also proposes a small set of triple modes, where a single key stream is generated in OFB mode and exored before every encryption and after the last encryption, and a few quadruple modes [4] with a higher conjectured security. However, Wagner has shown that if one allows chosen ciphertext/chosen *IV* attacks, the security of all but two of these improved modes with DES can be reduced to 2^{56} encryptions and between 2 and 2^{32} chosen chosen-*IV* texts [25]. A further analysis of the influence of the constraints on the *IV*s has been provided by Handschuh and Preneel [15]. The ANSI X9.52 standard [2] has opted for the outer modes of triple-DES. Coppersmith et al. propose the CBCM mode [10], which is a quadruple mode; this mode has also been included in ANSI X9.52. Biham and Knudsen present a certificational attack on this mode with DES requiring 2^{65} chosen ciphertexts and memory that requires 2^{58} encryptions [6]. In conclusion, one can state that it seems possible to improve significantly over the matching ciphertext attacks. However, the security results strongly depend on the model, security proofs have not been found so far and the resulting schemes are rather slow. It seems more appropriate to upgrade DES to ►Rijndael/AES [13].

A second area of research is on how to encrypt plaintexts from finite sets, which are not necessarily of size 2^n; this problem is partially addressed by Davies and Price in [11]; a formal treatment has been developed by Black and Rogaway in [7].

Recommended Reading

1. Alkassar A, Geraldy A, Pfitzmann B, Sadeghi A-R (2002) Optimized self-synchronizing mode of operation. In: Mastsui M (ed) Fast software encryption. Lecture notes in computer science, vol 2355. Springer-Verlag, Berlin, pp 78–91
2. ANSI X9.52 (1998) Triple data encryption algorithm modes of operation
3. Bellare M, Desai A, Jokipii E, Rogaway P (1997) A concrete security treatment of symmetric encryption. Proceedings 38th annual symposium on foundations of computer science, FOCS'97. IEEE Computer Society, 394–403
4. Biham E (1996) Cryptanalysis of triple-modes of operation. Technion Technical Report CS0885
5. Biham E (1999) Cryptanalysis of triple modes of operation. J Cryptol 12(3):161–184
6. Biham E, Knudsen LR (2002) Cryptanalysis of the ANSI X9.52 CBCM mode. J Cryptol 15(1):47–59
7. Black J, Rogaway P (2002) Ciphers with arbitrary finite domains. In: Preneel B (ed) Topics in cryptology CT—RSA 2002. Lecture notes in computer science, vol 2271. Springer-Verlag, Berlin, pp 114–130
8. Black J, Urtubia H (2002) Sidechannel attacks on symmetric encryption schemes: the case for authenticated encryption. Proceedings of the 11th USENIX Security Symposium. pp 327–338
9. Canvel B, Hiltgen AP, Vaudenay S, Vuagnoux M (2003) Password interception in a SSL/TLS channel. In: Boneh D (ed) Advances in cryptology—CRYPTO 2003. Lecture notes in computer science, vol 2729. Springer-Verlag, Berlin, pp 583–599
10. Coppersmith D, Johnson DB, Matyas SM (1996) A proposed mode for triple-DES encryption. IBM J Res Dev 40(2): 253–262
11. Davies DW, Price WL (1989) Security for computer networks. An introduction to data security in teleprocessing and electronic funds transfer, 2nd edn. Wiley, New York
12. FIPS 81 (1980) DES modes of operation. Federal information processing standard (FIPS), Publication 81. National Bureau of Standards, US Department of Commerce, Washington, DC
13. FIPS 197 (2001) Advanced encryption standard (AES). Federal Information Processing Standard (FIPS), Publication 197. National Institute of Standards and Technology, US Department of Commerce, Washington, DC
14. Flajolet P, Odlyzko AM (1999) Random mapping statistics. In: Stern J (ed) Advances in Cryptlogy—EUROCRYPT'99. Lecture notes in computer science, vol 1592. Springer-Verlag, Berlin, pp 329–354
15. Handschuh H, Preneel B (1999) On the security of double and 2-key triple modes of operation. In: Knudsen LR (ed) Fast software encryption. Lecture notes in computer science, vol 1636. Springer-Verlag, Berlin, pp 215–230
16. ISO/IEC 10116 (1991) Information technology–security techniques—modes of operation of an *n*-bit block cipher algorithm. IS 10116
17. Kaliski BS, Robshaw MJB (1996) Multiple encryption: weighing security and performance. Dr DOBB'S J 243:123–127
18. Knudsen L (1994) Block ciphers—analysis, design and applications. PhD Thesis, Aarhus University, Denmark
19. Maurer UM (1991) New approaches to the design of self-synchronizing stream ciphers. In: Davies DW (ed) Advances in Cryptology—EUROCRYPT'91. Lecture notes in computer science, vol 547. Springer-Verlag, Berlin, pp 458–471
20. Menezes A, van Oorschot PC, Vanstone S (1998) Handbook of applied cryptography. CRC Press, Boca Raton, FL
21. Meyer CH, Matyas SM (1982) Cryptography: a new dimension in data security. Wiley & Sons, New York
22. NIST (2001) SP 800–38A recommendation for block cipher modes of operation—methods and techniques
23. Paterson KG, Yau A (2004) Padding oracle attacks on the ISO CBC mode encryption standard. In: Okamoto T (ed) Topics in cryptology—the cryptographers' track at the RSA conference. Lecture notes in computer science, vol 2964. Springer-Verlag, Berlin, pp 305–323
24. Vaudenay S (2002) Security flaws induced by CBC padding—applications to SSL, IPSEC, WTLS.... In: Knudsen L (ed) Advances in cryptology—EUROCRYPT 2002. Lecture notes in computer science, vol 2332. Springer-Verlag, Berlin, pp 534–546
25. Wagner D (1998) Cryptanalysis of some recently-proposed multiple modes of operation. In: Vaudenay S (ed) Fast software encryption. Lecture notes in computer science, vol 1372. Springer-Verlag, Berlin, pp 254–269

Modular Arithmetic

Scott Contini[1], Çetin Kaya Koç[2], Colin D. Walter[3]
[1]Silverbrook Research, New South Wales, Australia
[2]College of Engineering and Natural Sciences, Istanbul Sehir University, Uskudar, Istanbul, Turkey
[3]Information Security Group, Royal Holloway University of London, Surrey, UK

Synonyms

Residue arithmetic

Related Concepts

►Finite Field; ►Prime Fields; ►Residue; ►Rings

Definition

Modular arithmetic is almost the same as the usual arithmetic of whole numbers. The main difference is that operations involve remainders after division by a specified number (the *modulus*) rather than the integers themselves.

Background

Modular arithmetic is a key ingredient of many public key cryptosystems. It provides finite structures (called "rings") which have all the usual arithmetic operations of the integers and which can be implemented without difficulty using existing computer hardware. An important property of these structures is that they appear to be randomly permuted by operations such as exponentiation, but the permutation is often easily reversed by another exponentiation. For suitably chosen cases, these operations enable *encryption* and decryption or *signature generation* and verification. Direct applications include *RSA public-key encryption* and the *RSA digital signature scheme* [17], *ElGamal public key encryption* and the *ElGamal digital signature scheme* [3], the *Fiat-Shamir signature scheme* [4], the *Schnorr Identification Protocol* [18], and *Diffie-Hellman key agreement* [2].

Modular arithmetic is also used to construct *finite fields* and in tests during *prime generation* [7] (►Probabilistic Primality Test). Several copies of the modular structures form higher dimensional objects in which lines, planes, and curves can be constructed. These can be used to perform *elliptic curve cryptography* (ECC) [6, 12] and to construct *threshold schemes* [19]. Additionally, modular arithmetic is used in some *hash functions* and *symmetric key* primitives. In many such cases, the modulus is implied by the computer word size, but other times the modulus is explicitly stated.

Theory

Introduction

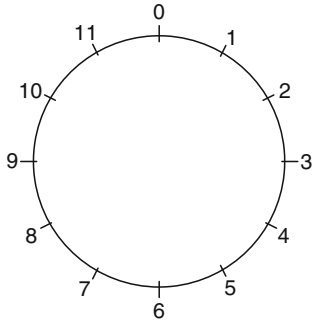
There are many examples of modular arithmetic in everyday life. It is applicable to almost any measurement of a repeated, circular or cyclic process. Clock time is a typical example: seconds range from 0 to 59 and just keep repeating, hours run from 0 to 11 (or 23) and also keep repeating, days run from Sunday (0, say) to Saturday (6, say). These are examples of arithmetic modulo 60, 12 (or 24), and 7, respectively. Measuring angles in degrees uses arithmetic modulo 360.

To understand arithmetic in modulus N, imagine a line of length N units, where the whole number points $0, \ldots, N-1$ are labelled. Now connect the two end points of the line so that it forms a circle of circumference N. Performing modular arithmetic with respect to *modulus* N is equivalent to arithmetic with the marked units on this circle.

An example for $N = 12$ is shown in Fig. 1. If one starts at number 0 and moves 14 units forward, the number 2 is reached. This is written $14 = 2 \pmod{12}$. Similarly, one can walk backwards 15 units from 0 and end up at 9. Hence, $-15 = 9 \pmod{12}$. In this arithmetic, every 12 is discarded. Equivalently, for any two numbers A and B such that $A = B \pmod{12}$, 12 divides the difference $A - B$.

Modular addition is the same as addition of units on this circle. For example, if $N = 12$ and the numbers 10 and 4 are added on this circle, the result is 2. This is because if one starts at position 10 and moves ahead 4 units, position 2 is reached. So four hours after 10 o'clock is 2 o'clock. This is written $10+4 = 2 \pmod{12}$. The result is the remainder (or "residue") after division by 12, i.e., $10 + 4 = 14$ becomes 14–12, namely 2.

The notation for modular arithmetic is almost identical to that for ordinary (integer) arithmetic. The main difference is that most expressions and equations specify



Modular Arithmetic. Fig. 1 Geometric view of arithmetic modulo 12

the modulus. Thus,

$$14 = 2 \pmod{12}$$

states that 14 and 2 represent the same element in a set which is called the *ring of residues mod 12*. When the modulus is clear, it may be omitted, as in

$$14 \equiv 2$$

The different symbol ≡ is needed because 14 and 2 are not equal as integers. The equation (or "congruence") is read as "14 is congruent to 2." All the integers in the set $\{\ldots, -22, -10, 2, 14, 26, \ldots\}$ represent the same *residue class* (or *congruence class*) modulo 12 because they all give the same remainder on division by 12, *i.e* the difference between any two of them is a multiple of 12. In general, the numbers $A, A+N, A+2N, A+3N, \ldots$ and $A-N, A-2N, A-3N, \ldots$ are all *equivalent* modulo N. Normally one works with the least nonnegative representative of a class, 2 in this case, because of the convenience of the unique choice when equality is tested, and because it takes up the least space. (Note that some programming languages incorrectly implement the modular reduction of negative numbers by failing to take proper account of the sign. The Microsoft Windows calculator correctly reduces negatives, but gives the greatest nonpositive value, namely, −10 in the above example.)

Modular Arithmetic Operations

Addition, subtraction, and multiplication are performed in exactly the same way as for integer arithmetic. Strictly speaking, the arithmetic is performed on the residue classes but, in practice, integers are picked from the respective classes, and they are worked with instead. Thus,

$$7 \times 11 + 3 = 80 = 8 \pmod{12}$$

In the expression on the left, the least nonnegative residues have been selected for working with. The result, 80, then requires a modular reduction to obtain a least nonnegative residue. Any representatives could be selected to perform the arithmetic. The answer would always differ by at most a multiple of the modulus, and so it would always reduce to the same value.

Hardware usually performs such reductions as frequently as possible in order to stop results from overflowing. Optimising integer arithmetic to perform modular arithmetic is the subject of much research. Modular multiplication is one of the most important areas of value to those implementing cryptographic functions; another is modular exponentiation. Montgomery [13] and Barrett [1] have created the most widely used methods for modular multiplication (Montgomery Modular Arithmetic and Barrett Reduction). Such operations make data-dependent use of power. This makes their use in embedded cryptosystems (e.g., smart cards) susceptible to attack through timing variations [8], compromising emanations [15], and differential power analysis [9] (Timing Attack, RF Attack and Smartcard Tamper Resistance). Secure implementation of modular arithmetic is therefore at least as important as efficiency in such systems.

Addition, subtraction, and multiplication behave in the same way for residues as for integer arithmetic. The usual identity, commutative and distributive laws hold, so that the set of residue classes form a "ring" in the mathematical sense, denoted $\mathbb{Z}_N$ for modulus N. Thus,

- $N \equiv 0 \pmod{N}$.
- $A + 0 \equiv A \pmod{N}$.
- $1 \times A \equiv A \pmod{N}$.
- if $A \equiv B \pmod{N}$, then $B \equiv A \pmod{N}$.
- if $A \equiv B \pmod{N}$ and $B \equiv C \pmod{N}$, then $A \equiv C \pmod{N}$.
- if $A \equiv B \pmod{N}$ and $C \equiv d \pmod{N}$, then $A + C \equiv B + d \pmod{N}$.
- if $A \equiv B \pmod{N}$ and $C \equiv d \pmod{N}$, then $A \times C \equiv B \times d \pmod{N}$.
- $A + B \equiv B + A \pmod{N}$.
- $A \times B \equiv B \times A \pmod{N}$.
- $A + (B + C) \equiv (A + B) + C \pmod{N}$.
- $A \times (B \times C) \equiv (A \times B) \times C \pmod{N}$.
- $A \times (B + C) \equiv (A \times B) + (A \times C) \pmod{N}$.

However, division is generally a problem unless the modulus is a prime. Since

$$10 = 2 \times 5 = 2 \times 11 \pmod{12}$$

it is clear that division by 2 (mod 12) can produce more than one answer; it is not uniquely defined. In fact, division by 2(mod 12) is not possible in some cases: $2x \pmod{12}$ always gives an even residue, so 3(mod 12) cannot be divided by 2. It can be shown that division by $A \pmod{N}$ is always well-defined precisely when A and N share no common factor, i.e., when they are *co-prime*. Thus, division by 7 is possible in modulo 12, but not division by 2 or 3.

If 1 is divided by 7(mod 12), the result is the *multiplicative inverse* of 7. Since $7 \times 7 = 1 \pmod{12}$, 7 is its own inverse. Following the usual notation of real numbers, this inverse is written 7^{-1}. For large numbers, the extended *Euclidean algorithm* [5] is used to compute multiplicative inverses. More precisely, to find the inverse of $A \pmod{N}$, one inputs the pair A, N into the algorithm, and it outputs X, Y such that $A \times X + N \times Y = \gcd(A, N)$, where gcd is the *greatest common divisor*. If the gcd is 1, then X is the inverse of $A \pmod{N}$. Otherwise, no such inverse exists.

Modular exponentiation (►Exponentiation Algorithms) is the main process in many of the cryptographic applications of this arithmetic. The notation is identical to that for integers and real numbers. $C^D \pmod N$ is D copies of C all multiplied together and reduced modulo N. As mentioned, the multiplicative inverse is denoted by an exponent $^{-1}$. Then the usual power laws, such as $x^A \times x^B = x^{A+B} \pmod N$, hold in the expected way.

When a composite modulus is involved, say N, it is often easier to work modulo its factors. Usually a set of co-prime factors of N is chosen such that the product is N. Solutions to the problem for each of these factors can then be pieced together into a solution modulo N using the *Chinese Remainder Theorem* (CRT) [14]. Implementations of the RSA cryptosystem which store the private key can use CRT to reduce the workload of decryption by a factor of 4.

An interesting aside is that the ring of integers modulo 0, i.e., $\mathbb{Z}_0$, is just the usual set of whole numbers with its normal operations of addition and multiplication: two whole numbers which belong to the same residue class must differ by a multiple of 0, and so have to be equal.

Multiplicative Groups and Euler's ϕ Function

The numbers which are *relatively prime* to (or just "prime to" for short) the modulus N have multiplicative inverses, as noted above. So they form a group under multiplication. Consequently, each number X which is prime to N has an *order* mod N which is the smallest positive integer n such that $X^n = 1 \pmod N$. The Euler phi function ϕ gives the number of elements in this group, and it is a multiple of the order of each element. So $X^{\phi(N)} = 1 \pmod N$ for X prime to N, and, indeed, $X^{k\phi(N)+1} = X \pmod N$ for such X and any k. This last is essentially what is known as Euler's Theorem. As an example, $\{1, 5, 7, 11\}$ is the set of residues prime to 12. So these form a multiplicative group of order $\phi(12) = 4$ and $1^4 = 5^4 = 7^4 = 11^4 = 1 \pmod{12}$. A special case of this result is *Fermat's "little" theorem* which states that $X^{P-1} = 1 \pmod P$ for a prime P and integer X which is not divisible by P. These are really the main properties that are used in reducing the cost of exponentiation in cryptosystems and in probabilistic primality testing (►Miller-Rabin Probabilistic Primality Test) [11, 16].

When $N = PQ$ is the product of two distinct primes P and Q, $\phi(N) = (P-1)(Q-1)$. RSA encryption on plaintext M is performed with a public exponent E to give ciphertext C defined by $C = M^E \pmod N$. Illustrating this with $N = 35$, $M = 17$ and $E = 5$, the computation is $C \equiv 17^5 \equiv (17^2)^2 \times 17 \equiv 289^2 \times 17 \equiv 9^2 \times 17 \equiv 81 \times 17 \equiv 11 \times 17 \equiv 187 \equiv 12 \pmod{35}$. The private decryption exponent D must have the property that $M = C^D \pmod N$, i.e., $M^{DE} = M \pmod N$. From the above, the value of D must satisfy $DE = k\phi(N) + 1$ for some k, i.e., D is a solution to $DE \equiv 1 \bmod (P-1)(Q-1)$. A solution is obtained using the *Euclidean algorithm* [5]. For the example, $D = 5$ since $\phi(35) = 24$ and $DE \equiv 5 \times 5 \equiv 1 \pmod{24}$. So $M \equiv 12^5 \equiv (12^2)^2 \times 12 \equiv 144^2 \times 12 \equiv 4^2 \times 12 \equiv 192 \equiv 17 \pmod{35}$, as expected. RSA chooses moduli which are products of two (large) primes so that decryption works also for texts which are not prime to the modulus. A nice exercise for the reader is to prove that this is really true. CRT is useful in the proof.

Prime Fields

When the modulus is a prime P, every residue except 0 is prime to the modulus. Hence, every nonzero number has a multiplicative inverse. So residues mod P form a *field* with P elements, written $\mathbb{F}_P$ or $GF(P)$. These *prime* fields are examples of *finite fields* [10]. The smallest such field is $\mathbb{F}_2$ which contains the two values 0 and 1. Because every nonzero has an inverse, the arithmetic of these fields is similar in many ways to that of the real numbers, and it is possible to perform similar geometric constructions. They already form a very rich source for cryptography, such as *Diffie-Hellman key agreement* [2] and *elliptic curve cryptography* [6, 12], and will undoubtedly form the basis for many more cryptographic primitives in the future.

Recommended Reading

1. Barrett P (1987) Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. In: Odlyzko AM (ed) Advances in Cryptology – CRYPTO '86, Lecture Notes in Computer Science, vol 263. Springer, New york, pp 311–323. http://www.springerlink.com/content/c4f3rqbt5dxxyad4/
2. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Inform Theory 22(6):644–654. http://citeseer.ist.psu.edu/diffie76new.html
3. ElGamal T (1985) A public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inform Theory 31(4):469–472. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1057074
4. Fiat A, Shamir A (1987) How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko AM (ed) Advances in Cryptology – CRYPTO '86, Lecture Notes in Computer Science, vol 263. Springer, Berlin, pp 186–194. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.8796
5. Knuth DE (1998) The art of computer programming, vol 2, Seminumerical Algorithms, 3rd edn. Addison-Wesley, Reading, ISBN 0-201-89684-2. http://www.informit.com/title/0201896842
6. Koblitz N (1987) Elliptic curve cryptosystems. Math Comput 48(177):203–209. http://www.jstor.org/pss/2007884
7. Koblitz N (1994) A course in number theory and cryptography, Graduate Texts in Mathematics, vol 114, 2nd edn. Springer, New York, ISBN 978-0-387-94293-3. http://www.springer.com/math/numbers/book/978-0-387-94293-3
8. Kocher P (1996) Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz N (ed)

Advances in Cryptology – CRYPTO '96, Lecture Notes in Computer Science, vol 1109, Springer, Berlin, 104–113. http://www.springerlink.com/content/4el17cvre3gxt4gd/
9. Kocher P, Jaffe J, Jun B (1999) Differential power analysis. In: Wiener M (ed) Advances in Cryptology – CRYPTO '99, Lecture Notes in Computer Science, vol 1666, Springer, Berlin, pp 388–397. http://www.springerlink.com/content/kx35ub53vtrkh2nx/
10. Lidl R, Niederreiter H (1994) Introduction to finite fields and their applications, 2nd edn. Cambridge University Press, Cambridge, ISBN 9780521460941. http://www.cambridgeuniversitypress.com/catalogue/catalogue.asp?isbn=9780521460941
11. Miller GL (1976) Riemann's hypothesis and tests for primality. J Comput Syst Sci 13(3):300–317. http://www.cs.cmu.edu/~glmiller/Publications/b2hd-Mi76.html
12. Miller V (1986) Uses of elliptic curves in cryptography. In: Williams HC (ed) Advances in Cryptology – CRYPTO '85: Proceedings, Lecture Notes in Computer Science, vol 218, Springer, Berlin, pp 417–426. http://www.springerlink.com/content/4lfhkd08684v3wyl/
13. Montgomery PL (1985) Modular multiplication without trial division. Math Comput 44(170):519–521. http://www.jstor.org/pss/2007970
14. Quisquater J-J, Couvreur C (1982) Fast decipherment algorithm for RSA Publickey cryptosystem. Electron Lett 18(21):905–907. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4246955
15. Quisquater J-J, Samyde D (2001) ElectroMagnetic analysis (EMA): measures and counter-measures for smart cards. In: Attali I, Jensen T (eds) Smart card programming and security (e-Smart 2001), Lecture Notes in Computer Science, vol 2140, Springer, Cannes, pp 200–210. http://www.springerlink.com/content/chmydkq8x5tgdrce/
16. Rabin MO (1980) Probabilistic algorithm for testing primality. J Number Theory 12(1):128–138. http://dx.doi.org/10.1016/0022-314X(80)90084-0
17. Rivest RL, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.5588
18. Schnorr CP (1991) Efficient signature generation by smart cards. J Cryptol 4:161–174. http://www.springerlink.com/content/w037127811042441/
19. Stinson DR (2005) Cryptography: theory and practice, 3rd edn. CRC Press, New York, ISBN 9781584885085. http://www.crcpress.com/product/isbn/9781584885085

Modular Root

Scott Contini
Silverbrook Research, New South Wales, Australia

Related Concepts

►Euler's Totient Function; ►Integer Factoring; ►Modular Arithmetic; ►Number Theory; ►Quadratic Residue; ►RSA Problem

Definition

In the congruence $x^e \equiv y \bmod n$, x is said to be the e^{th} *modular root* of y with respect to modulus n.

Background

The cases that are of interest to cryptography have $\gcd(x, n) = \gcd(y, n) = 1$. Algorithms for finding modular roots are relevant to the security of the ►RSA cryptosystem.

Theory

Computing modular roots is no more difficult than finding the order of the multiplicative group modulo n. In number theoretic terminology, this value is known as ►Euler's totient function, $\phi(n)$, which is defined to be the number of integers in $\{1, 2, \ldots, n-1\}$ that are ►relatively prime to n. If $\gcd(e, \phi(n)) = 1$, then there is either one or zero solutions, depending upon whether y is in the multiplicative ►subgroup generated by x. Assuming it is, the solution is obtained by raising both sides of the congruence to the power $e^{-1} \bmod \phi(n)$. If the gcd condition is not 1, then there may be more than one solution. For example, consider the special case of $e = 2$ and n an odd integer larger than 1. The congruence can have solutions only if y is a ►quadratic residue modulo n. Furthermore, if x is one solution, then $-x$ is another, implying that there are at least two distinct solutions.

Open Problems

Computing modular roots is easy when n is prime since then $\phi(n) = n - 1$. The more interesting case is when n is composite, where it is known as the ►RSA problem. An important open question is whether a method exists for computing modular roots faster than ►integer factoring. Note that any method which finds $\phi(n)$ cannot be faster than factoring since determining $\phi(n)$ is provably as difficult as factoring n.

Modulus

Scott Contini
Silverbrook Research, New South Wales, Australia

Related Concepts

►Modular Arithmetic; ►Number Theory

Definition

In ►modular arithmetic, the operand that the mod operation is computed with respect to is known as the *modulus*.

Background

In the congruence $a \equiv b \bmod n$, the value n is the modulus.

Refer the more general entry on ►modular arithmetic.

Monitoring

►Eavesdropping

Monotone Signatures

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Related Concepts

►Blackmailing Attacks; ►Digital Signatures

Definition

A monotone signature is a process allowing to resist, to some extent, blackmailing attacks. A monotone signature admits ℓ keys-pairs $\{pk_i, sk_i\}$. The scheme is such that a signature s generated with sk_i is verifiable with respect to all pk_j for $j \leq i$. Hence in case of blackmailing, the signer can reveal the key sk_i and inform users that pk_i is obsolete (switch to pk_{i+1}). The legitimate signer always uses sk_ℓ to sign messages.

Recommended Reading

1. Naccache D, Pointcheval D, Tymen C (2001) Monotone signatures. In: Syverson PF (ed) Financial cryptography. 5th International conference, FC 2001, Grand Cayman, British West Indies, 19–22 Feb 2002, Proceedings. Volume 2339 of Lecture notes in computer science, pp 295–308, Springer

Montgomery Arithmetic

Çetin Kaya Koç[1], Colin D. Walter[2]
[1]College of Engineering and Natural Sciences, Istanbul Sehir University, Uskudar, Istanbul, Turkey
[2]Information Security Group, Royal Holloway University of London, Surrey, UK

Related Concepts

►Modular Arithmetic; ►Modular Exponentiation

Definition

Suppose a machine performs arithmetic on words of w bits. Let a, b, and n be cryptographically sized integers represented using s such words. Then the Montgomery modular product of a and b modulo n is $abr^{-1} \pmod n$ where $r = 2^{sw}$. This is computed at a word level using a particularly straightforward and efficient algorithm. Compared with the normal "school book" method, for each word of the multiplier the reduction modulo n is performed by adding rather than subtracting a multiple of n, only a single digit is used to decide on this multiple, and the accumulating product is shifted down rather than up.

Background

The modular reduction $u \pmod n$ is typically computed on a word-based machine by repeatedly taking several leading digits from u and n, obtaining the leading digit of their quotient, and using that multiple of n to reduce u. This takes a number of clock cycles on a general processor, and the machine has to wait for carries to propagate from lowest to highest word before the next iteration can take place. Peter Montgomery designed his algorithm [5] to simplify or avoid these bottlenecks so that the modular exponentiations typical of public key cryptography could be significantly speeded up. The consequent initial and final scalings by a power of r are relatively cheap. Resource-constrained environments such as those in a smart card or RFID device benefit particularly from the choice of this modular multiplication algorithm.

Theory

Introduction

In 1985, P. L. Montgomery introduced an efficient algorithm [5] for computing $u = a \cdot b \pmod n$, where a, b, and n are k-bit binary numbers. The algorithm is particularly suitable for implementation on general-purpose computers (signal processors or microprocessors) which are capable of performing fast arithmetic modulo a power of 2. The Montgomery reduction algorithm computes the resulting k-bit number u without performing a division by the modulus n. Via an ingenious representation of the residue class modulo n, this algorithm replaces division by n with division by a power of 2. The latter operation is easily accomplished on a computer since the numbers are represented in binary form. Assuming the modulus n is a k-bit number, i.e., $2^{k-1} \leq n < 2^k$, let r be 2^k. The Montgomery reduction algorithm requires that r and

n be relatively prime, i.e., $\gcd(r,n) = \gcd(2^k,n) = 1$. This requirement is satisfied if n is odd. In the following, the basic idea behind the Montgomery reduction algorithm is summarized.

Given an integer $a < n$, define its *n-residue* or *Montgomery representation* with respect to r as

$$\overline{a} = a \cdot r \pmod{n}.$$

It is straightforward to show that the set

$$\{i \cdot r \pmod{n} \mid 0 \le i \le n-1\}$$

is a complete residue system, i.e., it contains all numbers between 0 and $n-1$. Thus, there is a one-to-one correspondence between the numbers in the range 0 and n–1 and the numbers in the above set. The Montgomery reduction algorithm exploits this property by introducing a much faster multiplication routine which computes the n-residue of the product of the two integers whose n-residues are given. Given two n-residues $\overline{a}$ and $\overline{b}$, the *Montgomery product* is defined as the scaled product

$$\overline{u} = \overline{a} \cdot \overline{b} \cdot r^{-1} \pmod{n}$$

where r^{-1} is the (multiplicative) inverse of r modulo n (see ►Modular Arithmetic), i.e., it is the number with the property

$$r^{-1} \cdot r = 1 \pmod{n}.$$

As the notation implies, the resulting number $\overline{u}$ is indeed the n-residue of the product

$$u = a \cdot b \pmod{n}$$

since

$$\begin{aligned}\overline{u} &= \overline{a} \cdot \overline{b} \cdot r^{-1} \pmod{n} \\ &= (a \cdot r) \cdot (b \cdot r) \cdot r^{-1} \pmod{n} \\ &= (a \cdot b) \cdot r \pmod{n}.\end{aligned}$$

In order to describe the Montgomery reduction algorithm, an additional quantity n' is needed. This is the integer with the property

$$r \cdot r^{-1} - n \cdot n' = 1.$$

The integers r^{-1} and n' can both be computed by the extended *Euclidean algorithm* [2]. The Montgomery product algorithm, which computes

$$\overline{u} = \overline{a} \cdot \overline{b} \cdot r^{-1} \pmod{n}$$

given $\overline{a}$ and $\overline{b}$, is given below:

function MonPro($\overline{a}, \overline{b}$)

Step 1. $t := \overline{a} \cdot \overline{b}$
Step 2. $m := t \cdot n' \pmod{r}$
Step 3. $\overline{u} := (t + m \cdot n)/r$
Step 4. **if** $\overline{u} \ge n$ **then return** $\overline{u} - n$
else return $\overline{u}$

The most important feature of the Montgomery product algorithm is that the operations involved are multiplications modulo r and divisions by r, both of which are intrinsically fast operations since r is a power 2. The MonPro algorithm can be used to compute the (normal) product u of a and b modulo n, provided that n is odd:

function ModMul(a, b, n) {n is an odd number}

Step 1. Compute n' using the extended Euclidean algorithm.
Step 2. $\overline{a} := a \cdot r \pmod{n}$
Step 3. $\overline{b} := b \cdot r \pmod{n}$
Step 4. $\overline{u} :=$ MonPro($\overline{a}, \overline{b}$)
Step 5. $u :=$ MonPro($\overline{u}, 1$)
Step 6. **return** u

A better algorithm can be given by observing the property

$$\text{MonPro}(\overline{a}, b) = (a \cdot r) \cdot b \cdot r^{-1} = a \cdot b \pmod{n},$$

which modifies the above algorithm to:

function ModMul(a, b, n) {n is an odd number}

Step 1. Compute n' using the extended Euclidean algorithm.
Step 2. $\overline{a} := a \cdot r \pmod{n}$
Step 3. $u :=$ MonPro($\overline{a}, b$)
Step 4. **return** u

However, the preprocessing operations, namely, steps (1) and (2), are rather time-consuming, especially the first. Since r is a power of 2, the second step can be done using k repeated shift and subtract operations. Thus, it is not a good idea to use the Montgomery product computation algorithm when a single modular multiplication is to be performed.

Montgomery Exponentiation

The Montgomery product algorithm is more suitable when several modular multiplications are needed with respect to the same modulus. Such is the case when one needs to

compute a modular exponentiation, i.e., the computation of $M^e \pmod n$. Algorithms for modular exponentiation decompose the operation into a sequence of squarings and multiplications using a common modulus n. This is where the Montgomery product operation MonPro finds its best use. In the following, modular exponentiation is exemplified using the standard "square-and-multiply" method, i.e., the left-to-right *binary exponentiation method*, with e_i being the bit of index i in the k-bit exponent e:

function ModExp(M, e, n) {n is an odd number}

Step 1. Compute n' using the extended Euclidean algorithm.
Step 2. $\overline{M} := M \cdot r \pmod n$
Step 3. $\overline{x} := 1 \cdot r \pmod n$
Step 4. **for** $i = k - 1$ **down to** 0 **do**
Step 5. $\quad \overline{x} := \text{MonPro}(\overline{x}, \overline{x})$
Step 6. $\quad$ **if** $e_i = 1$ **then** $\overline{x} := \text{MonPro}(\overline{M}, \overline{x})$
Step 7. $x := \text{MonPro}(\overline{x}, 1)$
Step 8. **return** x

Thus, the process starts with obtaining the n-residues $\overline{M}$ and $\overline{1}$ from the ordinary residues M and 1 using division-like operations, as described above. However, once this preprocessing has been completed, the inner loop of the binary exponentiation method uses the Montgomery product operation, which performs only multiplications modulo 2^k and divisions by 2^k. When the loop terminates, the n-residue $\overline{x}$ of the quantity $x = M^e \pmod n$ has been obtained. The ordinary residue number x is recovered from the n-residue by executing the MonPro function with arguments $\overline{x}$ and 1. This is easily shown to be correct since

$$\overline{x} = x \cdot r \pmod n$$

immediately implies that

$$x = \overline{x} \cdot r^{-1} \pmod n = \overline{x} \cdot 1 \cdot r^{-1} \pmod n := \text{MonPro}(\overline{x}, 1).$$

The resulting algorithm is quite fast, as was demonstrated by many researchers and engineers who have implemented it; for example, see [1, 4]. However, this algorithm can be refined and made more efficient, particularly when the numbers involved are multi-precision integers. For example, Dussé and Kaliski [1] gave improved algorithms, including a simple and efficient method for computing n'. In fact, any exponentiation algorithm can be modified in the same way to make use of MonPro: simply append the illustrated pre- and postprocessing (steps 1–3 and 7) and replace the normal modular multiplication operations in the iterative loop with applications of MonPro to the corresponding n-residues (steps 4–6 in the above).

Here, as an example, the computation of $x = m7^{10} \pmod{13}$ is illustrated using the Montgomery binary exponentiation algorithm.

- Since $n = 13$, the value for r is taken to be $r = 2^4 = 16 > n$.
- Step 1 of the ModExp routine: Computation of n':
  The extended Euclidean algorithm is used to determine that $16 \cdot 9 - 13 \cdot 11 = 1$, and thus $r^{-1} = 9$ and $n' = 11$.
- Step 2: Computation of $\overline{M}$:
  Since $M = 7$, $\overline{M} := M \cdot r \pmod n = 7 \cdot 16 \pmod{13} = 8$.
- Step 3: Computation of $\overline{x}$ for $x = 1$:
  $\overline{x} := x \cdot r \pmod n = 1 \cdot 16 \pmod{13} = 3$.
- Step 4: The loop of ModExp:

e_i	Step 5	Step 6
1	MonPro(3,3) = 3	MonPro(8,3) = 8
0	MonPro(8,8) = 4	
1	MonPro(4,4) = 1	MonPro(8,1) = 7
0	MonPro(7,7) = 12	

  - Step 5: Computation of MonPro$(3,3) = 3$:
    $t := 3 \cdot 3 = 9$
    $m := 9 \cdot 11 \pmod{16} = 3$
    $u := (9 + 3 \cdot 13)/16 = 48/16 = 3$
  - Step 6: Computation of MonPro$(8,3) = 8$:
    $t := 8 \cdot 3 = 24$
    $m := 24 \cdot 11 \pmod{16} = 8$
    $u := (24 + 8 \cdot 13)/16 = 128/16 = 8$
  - Step 5: Computation of MonPro$(8,8) = 4$:
    $t := 8 \cdot 8 = 64$
    $m := 64 \cdot 11 \pmod{16} = 0$
    $u := (64 + 0 \cdot 13)/16 = 64/16 = 4$
  - …
- Step 7 of the ModExp routine: $x = \text{MonPro}(12,1) = 4$
  $t := 12 \cdot 1 = 12$
  $m := 12 \cdot 11 \pmod{16} = 4$
  $u := (12 + 4 \cdot 13)/16 = 64/16 = 4$

Thus, $x = 4$ is obtained as the result of the operation $7^{10} \pmod{13}$.

Efficient Montgomery Multiplication

The previous algorithm for Montgomery multiplication is not efficient on a general purpose processor in its stated form, and so perhaps only has didactic value. Since the Montgomery multiplication algorithm computes

$$\text{MonPro}(a, b) = abr^{-1} \pmod n$$

and $r = 2^k$, it is possible to give a more efficient bit-level algorithm which computes exactly the same value

$$\mathrm{MonPro}(a,b) = ab2^{-k} \pmod{n}$$

as follows:

function MonPro(a,b) {n is odd and $a,b,n < 2^k$}

Step 1. $u := 0$
Step 2. **for** $i = 0$ *to* $k-1$
Step 3. $\quad u := u + a_i b$
Step 4. $\quad u := u + u_0 n$
Step 5. $\quad u := u/2$
Step 6. **if** $u \geq n$ **then return** $u - n$
$\quad$ **else return** u

where u_0 is the least significant bit of u and a_i is the bit with index i in the binary representation of a. The oddness of n guarantees that the division in step (5) is exact. This algorithm avoids the computation of n' since it proceeds bit-by-bit: it needs only the least significant bit of n', which is always 1 since n' is odd because n is odd.

The equivalent word-level algorithm only needs the least significant word n'_0 (w bits) of n', which can also be easily computed since

$$2^k \cdot 2^{-k} - n \cdot n' = 1$$

implies

$$-n_0 \cdot n'_0 = 1 \pmod{2^w}.$$

Therefore, n'_0 is equal to $-n_0^{-1} \pmod{2^w}$, and it can be quickly computed by the extended Euclidean algorithm or table lookup since it is only w bits (1 word) long. For the words (digits) a_i of a with index i and $k = sw$, the word-level Montgomery algorithm is as follows:

function MonPro(a,b) {n is odd and $a,b,n < 2^{sw}$}

Step 1. $u := 0$
Step 2. **for** $i = 0$ *to* $s-1$
Step 3. $\quad u := u + a_i b$
Step 4. $\quad u := u + (-n_0^{-1}) \cdot u_0 \cdot n$
Step 5. $\quad u := u/2^w$
Step 6. **if** $u \geq n$ **then return** $u - n$
$\quad$ **else return** u

This version of Montgomery multiplication is the algorithm of choice for systolic array modular multipliers [6] because, unlike classical modular multiplication, completion of the carry propagation required in Step 3 does not prevent the start of Step 4, which needs u_0 from Step 3. Such systolic arrays are extremely useful for fast *SSL*/*TLS* servers.

Application to Finite Fields

Since the integers modulo p form the *finite field* $GF(p)$, these algorithms are directly applicable for performing multiplication in $GF(p)$ by taking $n = p$. Similar algorithms are also applicable for multiplication in $GF(2^k)$, which is the finite field of polynomials with coefficients in $GF(2)$ modulo an irreducible polynomial of degree k [3].

Montgomery squaring (required for exponentiation) just uses MonPro with the arguments a and b being the same. However, in fields of characteristic 2, this is rather inefficient: all the bit products $a_i a_j$ for $i \neq j$ cancel, leaving just the terms a_i^2 to deal with. Then it may be appropriate to implement a modular operation ab^2 for use in exponentiation.

Secure Montgomery Multiplication

As a result of the data-dependent conditional subtraction in the last step of MonPro, embedded cryptosystems which make use of the above algorithms can be subject to a *timing attack* which reveals the secret key [9]. In the context of modular exponentiation, the final subtraction of each MonPro should then be avoided [7]. With this step omitted, all I/O to/from MonPro simply becomes bounded by $2n$ instead of n, but an extra loop iteration may be required on account of the larger arguments [8].

Recommended Reading

1. Dussé SR, Kaliski BS Jr (1991) A cryptographic library for the motorola DSP56000. In: Damgård IB (ed) Advances in cryptology – EUROCRYPT '90. Lecture notes in computer science, vol 473, Springer, Berlin, pp 230–244. http://www.springerlink.com/content/07h8eyfk4jnafy5c/
2. Knuth DE (1998) The art of computer programming, 3rd edn. Semi-numerical algorithms, vol 2. Addison-Wesley, Reading. ISBN 0-201-89684-2. http://www.informit.com/title/0201896842
3. Koç ÇK, Acar T (1998) Montgomery multiplication in GF(2k). Design Code Cryptogr 14(1):57–69. http://www.springerlink.com/content/g25q57w02h21jv71/
4. Laurichesse D, Blain L (1991) Optimized implementation of RSA cryptosystem. Comput Secur 10(3):263–267. http://dx.doi.org/10.1016/0167-4048(91)90042-C
5. Montgomery PL (1985) Modular multiplication without trial division, Math Comput 44(170):519–521. http://www.jstor.org/pss/2007970
6. Walter CD (1993) Systolic modular multiplication. IEEE Trans Comput 42(3):376–378. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=210181
7. Walter CD (1999) Montgomery exponentiation needs no final subtractions. Electron Lett 35(21):1831–1832. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=810000
8. Walter CD (2002) Precise bounds for montgomery modular multiplication and some potentially insecure RSA moduli. In:

Preneel B (ed) Topics in cryptology – CT-RSA 2002. Lecture notes in computer science, vol 2271. Springer, Berlin, pp 30–39. http://www.springerlink.com/content/3p1qw48b1vu84gya/
9. Walter CD, Thompson S (2001) Distinguishing exponent digits by observing modular subtractions. In: Naccache D (ed) Topics in cryptology – CT-RSA 2001. Lecture notes in computer science, vol 2020. Springer, Berlin, pp 192–207. http://www.springerlink.com/content/8h6fn41pfj8uluuu/

Moore's Law

Burt Kaliski
Office of the CTO, EMC Corporation, Hopkinton, MA, USA

Related Concepts

►Exhaustive Key Search; ►Exponential Time; ►Polynomial Time

Definition

Moore's Law states that the amount of computing power available for a given cost will increase by a factor of two every 18 months to 2 years.

Background

Moore's Law was articulated in 1965 by Gordon Moore of Intel [1].

Theory

The phenomenal rise in computing power over the past half century – which has driven the increasing need for cryptography and security as covered in this work – is due to an intense research and development effort that has produced an essentially exponential increase in the number of transistors than can fit on a chip, while maintaining a constant chip cost.

Roughly speaking, the amount of computing power available for a given cost has increased and continues to increase by a factor of 2 every 18 months to 2 years, a pattern called *Moore's Law* after Gordon Moore of Intel, who first articulated this exponential model. More specifically, the amount of computing power $P(t)$ available for a given cost at time t may be estimated as

$$P(t) = P(t_0)\, 2^{(t-t_0)/T}$$

where $P(t_0)$ is the amount of computing power available for the same cost at a reference time t_0, and T is the interval between doublings in computing power (e.g., 1.5 or 2 years). Lenstra and Verheul have formalized the treatment such growth rates in their model for estimating the strength of cryptographic key sizes over time [2].

Applications

The implications of Moore's Law to cryptography are twofold. First, the resources available to users are continually growing, so that users can readily employ stronger and more complex cryptography. Second, the resources available to opponents are also growing. Effectively, the strength of any cryptosystem decreases by the equivalent of one symmetric-key bit every 18 months – or 8 bits every 12 years – posing a challenge to long-term security. This long-term perspective on advances in (classical) computing is one motivation for the large key sizes currently being proposed for many cryptosystems, such as the *Advanced Encryption Standard* (►Rijndael/AES), which has a 128-bit symmetric key.

The benefit of Moore's Law to users of cryptography is much greater than the benefit to opponents, because even a modest increase in computing power has a much greater impact on the key sizes that can be used, than on the key sizes that can be broken. This is a consequence of the fact that the methods available for using cryptosystems are generally ►polynomial time, while the fastest methods known for breaking ►symmetric cryptosystems and several asymmetric cryptosystems are ►exponential time.

This contrast between using and breaking algorithms may well be limited to classical computing for current algorithms. Quantum computers pose a more substantial potential threat in the future, because methods have been discovered for breaking ►public-key cryptosystems based on integer factorization or the discrete logarithm problem in ►polynomial time on such computers [3]. Quantum computers themselves are still in the research phase, and it is not clear if and when a sufficiently large quantum computer could be built. But if one were built (perhaps sometime in the next 30 years?), the impact on cryptography and security would be even more dramatic than the one Moore's Law has had so far.

Recommended Reading

1. Moore G (1965) Cramming more components onto integrated circuits. Electronics 38(8):114
2. Lenstra AK, Verheul ER (2000) Selecting cryptographic key sizes, In: Imai H, Zheng Y (eds) Public Key Cryptography, PKC 2000, Lecture Notes in Computer Science, vol 1751. Springer, Berlin, pp 446–465
3. Shor PW (1994) Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings of the 35th Annual IEEE Symposium on the Foundations of Computer Science, Santa Fe, pp 124–134

MPKC

►Multivariate Cryptography

MQ or Multivariate Quadratic Public-Key Cryptosystem (MQPKC)

►Multivariate Cryptography

Multibiometrics

Arun Ross
Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV, USA

Synonyms

Biometric fusion

Related Concepts

►DNA; ►Fingerprint; ►Gait; ►Hand Geometry; ►Handwriting; ►Iris; ►Keystroke; ►Palmprint; ►Speaker Recognition; ►Vascular Patterns

Definition

Multibiometrics refers to the use of multiple sources of biometric information in order to establish the identity of an individual. Multibiometric systems combine the biometric evidence offered by multiple biometric sensors (e.g., 2D and 3D face sensors), algorithms (e.g., minutia-based and ridge-based fingerprint matchers), samples (e.g., frontal and profile face images), units (e.g., left and right irises), or traits (e.g., face and iris) in order to enhance the recognition accuracy of a biometric system. Information fusion can be accomplished at several different levels in a biometric system, including the sensor-level, feature-level, score-level, rank-level, or decision-level. The challenge is to design an effective fusion scheme to consolidate multiple pieces of evidence in order to generate a decision about an individual's identity.

Background

Most biometric systems that are presently in use, typically use a single biometric trait to establish identity (i.e., they are unibiometric systems). Some of the challenges commonly encountered by biometric systems include:

1. Noise in sensed data: The biometric data being presented to the system may be contaminated by noise due to imperfect acquisition conditions or subtle variations in the biometric itself.
2. Nonuniversality: The biometric system may not be able to acquire meaningful biometric data from a subset of individuals resulting in a failure-to-enroll (FTE) error.
3. Upper bound on identification accuracy: The matching performance of a unibiometric system cannot be indefinitely improved by tuning the feature extraction and matching modules. There is an implicit upper bound on the number of distinguishable patterns (i.e., the number of distinct biometric feature sets) that can be represented using a template.
4. Spoof attacks: Behavioral traits such as voice and signature are vulnerable to spoof attacks by an impostor attempting to mimic the traits corresponding to legitimately enrolled subjects. Physical traits such as fingerprints can also be spoofed by inscribing ridge-like structures on synthetic material such as gelatin and play-doh. Targeted spoof attacks can undermine the security afforded by the biometric system and, consequently, mitigate its benefits.

Some of the limitations of a unibiometric system can be addressed by designing a system that consolidates (or fuses) *multiple* sources of biometric information. This can be accomplished by fusing, for example, multiple traits of an individual, or multiple feature extraction and matching algorithms operating on the same biometric trait. Such systems, known as multibiometric systems can improve the matching accuracy of a biometric system while increasing population coverage and deterring spoof attacks. Fusion in biometrics relies on principles in the information fusion and multiple classifier system (MCS) literature.

Besides enhancing matching accuracy, the other advantages of multibiometric systems over traditional unibiometric systems are enumerated below.

1. Multibiometric systems address the issue of nonuniversality (i.e., limited population coverage) encountered by unibiometric systems. If a subject's dry finger prevents her from successfully enrolling into a fingerprint system, then the availability of another biometric trait, say iris, can aid in the inclusion of the individual in the biometric system. A certain degree of flexibility is achieved when a user enrolls into the system using several different traits (e.g., face, voice, fingerprint, iris,

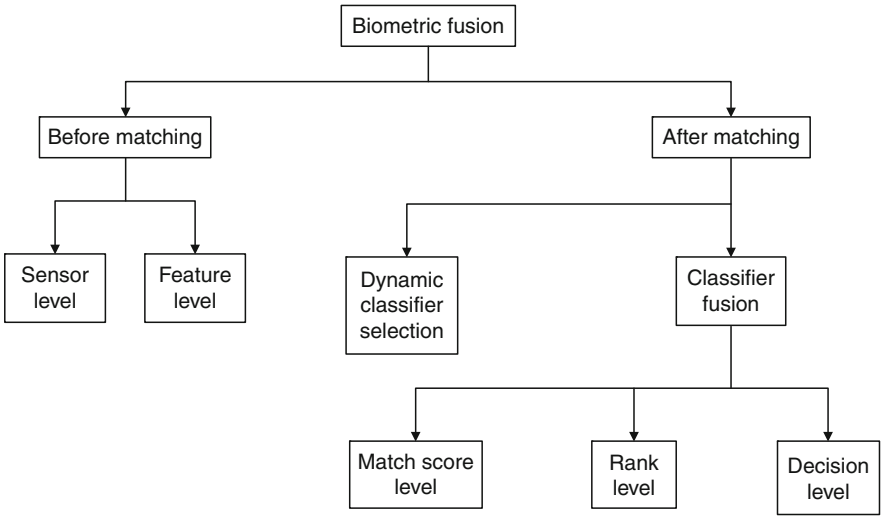
hand) while only a subset of these traits (e.g., face and voice) is requested during authentication based on the nature of the application under consideration and the convenience of the user.

2. Multibiometric systems can facilitate the filtering or indexing of large-scale biometric databases. For example, in a bimodal system consisting of face and fingerprint, the face feature set may be used to compute an index value for extracting a candidate list of potential identities from a large database of subjects. The fingerprint modality can then determine the final identity from this limited candidate list.
3. It becomes increasingly difficult (if not impossible) for an impostor to spoof multiple biometric traits of a legitimately enrolled individual. If each subsystem indicates the probability that a particular trait is a "spoof," then appropriate fusion schemes can be employed to determine if the user, in fact, is an impostor. Furthermore, by asking the user to present a random subset of traits at the point of acquisition, a multibiometric system facilitates a challenge-response type of mechanism, thereby ensuring that the system is interacting with a *live* user. Note that a challenge-response mechanism can be initiated in unibiometric systems also (e.g., system prompts "Please say 1-2-5-7," "Blink twice and move your eyes to the right," "Change your facial expression by smiling," etc.).
4. Multibiometric systems also effectively address the problem of noisy data. When the biometric signal acquired from a single trait is corrupted with noise, the availability of other (less noisy) traits may aid in the reliable determination of identity. Some systems take into account the *quality* of the individual biometric signals during the fusion process. This is especially important when recognition has to take place in adverse conditions where certain biometric traits cannot be reliably extracted. For example, in the presence of ambient acoustic noise, when an individual's voice characteristics cannot be accurately measured, the facial characteristics may be used by the multibiometric system to perform authentication. Estimating the quality of the acquired data is in itself a challenging problem but, when appropriately done, can reap significant benefits in a multibiometric system.
5. These systems also help in the *continuous* monitoring or tracking of an individual in situations when a single trait is not sufficient. Consider a biometric system that uses a 2D camera to procure the face and gait information of a person walking down a crowded aisle. Depending upon the distance and pose of the subject with respect to the camera, both these characteristics may or may not be simultaneously available. Therefore, either (or both) of these traits can be used depending upon the location of the individual with respect to the acquisition system thereby permitting the continuous monitoring of the individual.
6. A multibiometric system may also be viewed as a fault tolerant system which continues to operate even when certain biometric sources become unreliable due to sensor or software malfunction, or deliberate user manipulation. The notion of fault tolerance is especially useful in large-scale authentication systems involving a large number of subjects (such as a border control application).

Theory

Based on the type of information available in a certain module, different levels of fusion may be defined. The various levels of fusion can be categorized into two broad categories: pre-classification or fusion *before* matching, and post-classification or fusion *after* matching (see Fig. 1). Such a categorization is necessary since the amount of information available for fusion reduces drastically once the matcher has been invoked. Pre-classification fusion schemes typically require the development of new matching techniques (since the matchers used by the individual sources may no longer be relevant), thereby introducing additional challenges. Pre-classification schemes include fusion at the sensor (or raw data) and the feature levels while post-classification schemes include fusion at the match score, rank, and decision levels.

1. Sensor-level fusion: The raw biometric data (e.g., a face image) acquired from an individual represents the richest source of information although it is expected to be contaminated by noise (e.g., nonuniform illumination, background clutter, etc.). Sensor-level fusion refers to the consolidation of (a) raw data obtained using multiple sensors, or (b) multiple snapshots of a biometric using a single sensor.
2. Feature-level fusion: In feature-level fusion, the feature sets originating from multiple biometric algorithms are consolidated into a single feature set by the application of appropriate feature normalization, transformation, and reduction schemes. The primary benefit of feature-level fusion is the detection of correlated feature values generated by different biometric algorithms and, in the process, identifying a salient set of features that can improve recognition accuracy. Eliciting this feature set typically requires the use of dimensionality reduction methods and, therefore, feature-level fusion assumes the availability of a large number of training data. Also,



Multibiometrics. Fig. 1 Fusion can be accomplished at various levels in a biometric system

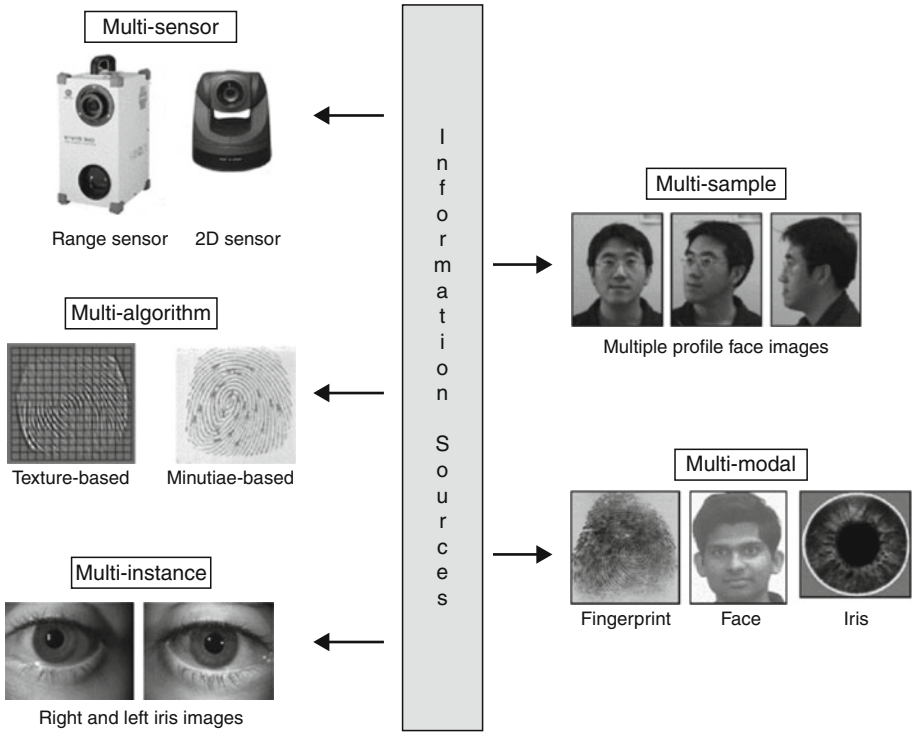
the feature sets being fused are typically expected to reside in commensurate vector space in order to permit the application of a suitable matching technique upon consolidating the feature sets.

3. Score-level fusion: In score-level fusion the match scores output by multiple biometric matchers are combined to generate a new match score (a scalar) that can be subsequently used by the verification or identification modules for rendering an identity decision. Fusion at this level is the most commonly discussed approach in the biometric literature primarily due to the ease of accessing and processing match scores (compared to the raw biometric data or the feature set extracted from the data). Fusion methods at this level can be broadly classified into three categories: density-based schemes, transformation-based schemes, and classifier-based schemes.
4. Rank-level fusion: When a biometric system operates in the identification mode, the output of the system can be viewed as a ranking of the enrolled identities. In this case, the output indicates the set of possible matching identities sorted in decreasing order of confidence. The goal of rank-level fusion schemes is to consolidate the ranks output by the individual biometric subsystems in order to derive a consensus rank for each identity. Ranks provide more insight into the decision-making process of the matcher compared to just the identity of the best match, but they reveal less information than match scores. However, unlike match scores, the rankings output by multiple biometric systems are comparable. As a result, no normalization is needed and this makes rank-level fusion schemes simpler to implement compared to the score level fusion techniques.
5. Decision-level fusion: Many commercial off-the-shelf (COTS) biometric matchers provide access only to the final recognition decision. When such COTS matchers are used to build a multibiometric system, only decision level fusion is feasible. Methods proposed in the literature for decision level fusion include "AND" and "OR" rules, majority voting, weighted majority voting, Bayesian decision fusion, the Dempster-Shafer theory of evidence, and behavior knowledge space.

Applications

A multibiometric system relies on the evidence presented by multiple sources of biometric information. Based on the nature of these sources, a multibiometric system can be classified into one of the following six categories: multi-sensor, multi-algorithm, multi-instance, multi-sample, multimodal, and hybrid (see Fig. 2).

1. Multi-sensor systems: Multi-sensor systems employ multiple sensors to capture a single biometric trait of an individual. For example, a face recognition system may deploy multiple 2D cameras to acquire the face image of a subject; an infrared sensor may be used in conjunction with a visible-light sensor to acquire the subsurface information of a person's face; a multi-spectral camera may be used to acquire images of the iris, face or finger; or an optical as well as a capacitive



Multibiometrics. Fig. 2 Sources of information for biometric fusion

sensor may be used to image the fingerprint of a subject. The use of multiple sensors, in some instances, can result in the acquisition of complementary information that can enhance the recognition ability of the system. For example, based on the nature of illumination due to ambient lighting, the infrared and visible-light images of a person's face can present different levels of information resulting in enhanced matching accuracy. Similarly, the performance of a 2D face matching system can be improved by utilizing the shape information presented by 3D range images.

2. Multi-algorithm systems: In some cases, invoking multiple feature extraction and/or matching algorithms on the same biometric data can result in improved matching performance. Multi-algorithm systems consolidate the output of multiple feature extraction algorithms, or that of multiple matchers operating on the same feature set. These systems do not necessitate the deployment of new sensors and, hence, are cost-effective compared to other types of multibiometric systems. But on the other hand, the introduction of new feature extraction and matching modules can increase the computational complexity of these systems.
3. Multi-instance systems: These systems use multiple instances of the same body trait and have also been referred to as multiunit systems in the literature. For example, the left and right index fingers, or the left and right irises of an individual, may be used to verify an individual's identity. The FBI's IAFIS (integrated automated fingerprint identification system) service combines the evidence of all ten fingers to determine a matching identity in the database. These systems can be cost-effective if a single sensor is used to acquire the multiunit data in a sequential fashion. However, in some instances, it may be desirable to obtain the multiunit data simultaneously, thereby demanding the design of an effective (and possibly more expensive) acquisition device.
4. Multi-sample systems: A single sensor may be used to acquire multiple samples of the same biometric trait in order to account for the variations that can occur in the trait, or to obtain a more complete representation of the underlying trait. A face system, for example, may capture (and store) the frontal profile of a person's face along with the left and right profiles in order to account for variations in the facial pose. Similarly, a fingerprint system equipped with a small size sensor

may acquire multiple dab prints of an individual's finger in order to obtain images of various regions of the fingerprint. A mosaicing scheme may then be used to stitch the multiple impressions and create a composite image. One of the key issues in a multi-sample system is determining the *number* of samples that have to be acquired from an individual. It is important that the procured samples represent the *variability* as well as the *typicality* of the individual's biometric data. To this end, the desired relationship between the samples has to be established beforehand in order to optimize the benefits of the integration strategy. For example, a face recognition system utilizing both the frontal- and side-profile images of an individual may stipulate that the side-profile image should be a three-quarter view of the face. Alternately, given a set of biometric samples, the system should be able to automatically select the "optimal" subset that would best represent the individual's variability.

5. Multimodal systems: Multimodal systems establish identity based on the evidence of multiple biometric traits. For example, some of the earliest multimodal biometric systems utilized face and voice features to establish the identity of an individual. Physically uncorrelated traits (e.g., fingerprint and iris) are expected to result in better *improvement* in performance than correlated traits (e.g., voice and lip movement). The cost of deploying these systems is substantially more due to the requirement of new sensors and, consequently, the development of appropriate user interfaces. The identification accuracy can be significantly improved by utilizing an increasing number of traits although the **curse-of-dimensionality** phenomenon would impose a bound on this number. The number of traits used in a specific application will also be restricted by practical considerations such as the cost of deployment, enrollment time, throughput time, expected error rate, user habituation issues, etc.
6. Hybrid systems: The term *hybrid* is used to describe systems that integrate a subset of the five scenarios discussed above. For example, consider an audio-visual multibiometric system in which multiple speaker recognition algorithms are combined with multiple face recognition algorithms at the match score or rank levels. Such a system would be multi-algorithmic as well as multimodal in its design.

Open Problems

Some topics of research in multibiometrics include (a) protecting multibiometric templates; (b) indexing multimodal databases; (c) consolidating biometric sources in highly unconstrained nonideal environments; (d) designing dynamic fusion algorithms to address the problem of incomplete input data; (e) quality-based fusion; and (e) predicting the matching performance of a multibiometric system.

Recommended Reading

1. Brunelli R, Falavigna D (1995) Person identification using multiple cues. IEEE T Pattern Anal 17(10):955–966
2. Chang KI, Bowyer KW, Flynn PJ (2005) An evaluation of multimodal 2D + 3D face biometrics. IEEE T Pattern Anal 27(4):619–624
3. Jain AK, Nandakumar K, Ross A (2005) Score normalization in multimodal biometric systems. Pattern Recogn 38(12): 2270–2285
4. Kittler J, Hatef M, Duin RP, Matas JG (1998) On combining classifiers. IEEE T Pattern Anal 20(3):226–239
5. Kuncheva LI (2004) Combining pattern classifiers - methods and algorithms. Wiley
6. Ross A, Nandakumar K, Jain AK (2006) Handbook of multibiometrics, 1st edn. Springer, New York

Multicast Authentication

►Broadcast Authentication from a Conditional Perspective
►Broadcast Authentication from an Information Theoretic

Multicast Stream Authentication

►Stream and Multicast Authentication

Multidimensional Databases

►Statistical Databases

Multi-Exponentiation

►Simultaneous Exponentiation

Multifactor Authentication

►Three-Factor Authentication

Multilevel Database

Pierangela Samarati, Giovanni Livraga
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Related Concepts

►Bell LaPadula; ►Inference Control; ►Polyinstantiation

Definition

A multilevel database system (MDBMS) supports the application of a multilevel policy for regulating access to the database objects.

Theory

The first formulation of multilevel mandatory policies and the ►Bell LaPadula model, simply assumed the existence of objects (information containers) to which a classification is assigned. This assumption works well in the operating system context, where objects to be protected are essentially files containing the data. Later studies investigated the extension of mandatory policies to database systems. While in operating systems security classes are assigned to files, database systems can afford a finer-grained classification. Classification can in fact be considered at the level of relations (equivalent to file-level classification in OS), at the level of columns (different properties can have a different classification), at the level of rows (properties referred to a given real-world entity or association have the same classification), or at the level of single cells (each data element, meaning the value assigned to a property for a given entity or association, can have a different classification), this latter being the finest possible classification. Element-level classification is appealing since it allows the assignment of a security class to each single real-world fact that needs to be represented. For instance, a patient's name can be labeled Unclassified, while her disease can be labeled Secret; also the disease of different patients can assume different classifications. However, the support of fine-grained classifications together with the obvious constraint of maintaining secrecy in the system operation introduces complications. The major complication is represented by the so-called ►polyinstantiation problem.

Let us start giving the definition of multilevel relational database. A relational database is composed of a finite set of relations, each defined over a set $A_1, \ldots, A_n$ of attributes (columns of the relation). Each relation is composed of a set $t_1, \ldots, t_k$ of tuples (rows of the relation) mapping attributes to values over their domain. A subset of the attributes, called key attributes, are used to uniquely identify each tuple in the relation, and the following *key constraints* are imposed: (1) no two tuples can have the same values for the key attributes, and (2) key attributes cannot be null.

In a multilevel relational database supporting element-level labeling (eg., [5, 7]), an access class $\lambda(t[A])$ is associated with each element $t[A]$ in a relation. An example of multilevel relation is illustrated in Fig. 1a. Note that the classification associated with a value usually does not represent the absolute sensitivity of the value as such, but rather the sensitivity of the fact that the attribute takes on that value for a specific entity in the real world. For instance, classification Secret associated with value `H1N1` of the last tuple is not the classification of value `H1N1` by itself, but of the fact that it is the disease suffered by `David`. Note that this is not meant to say that the classification of an element is independent of its value. As a matter of fact it can depend on the value. For instance a classification rule may state that all *infectious diseases* must be classified as Secret.

An alternative proposal to element-level classification applies classification at the level of key or nonkey attributes. For each tuple, one classification is specified for the key attributes (AK) and one classification for the nonkey attributes. Such granularity of classification has been proposed in [8] with the aim of eliminating possible semantic ambiguities caused by uncontrolled fine-grained classifications. As well as providing a clean view of the database at different clearance levels, the proposal introduces a theory of *belief*: a subject *sees* all the database instances dominated by its own classification, but *believes* only the database at its own classification.

Access control in multilevel DBMSs applies the two basic ►Bell LaPadula principles, although the no-write-up restriction is usually reduced to the principle of "write at your own level." In fact, while write-up operations can make sense in operating systems, where a file is seen as an information container and subjects may need to append low-level data in a high-level container, element-level classification nullifies this reasoning. Subjects at different levels have different views on a relation, which is the view composed only of elements they are cleared to see (i.e., whose classification they dominate). As an example, Fig. 1b reports the view of an Unclassified subject on the multilevel relation in Fig. 1a. Note that, in principle, to not convey information, the Unclassified subject should see no difference between values that are actually `null` in the database and those that are `null` since they have a higher classification. However, some proposals do not

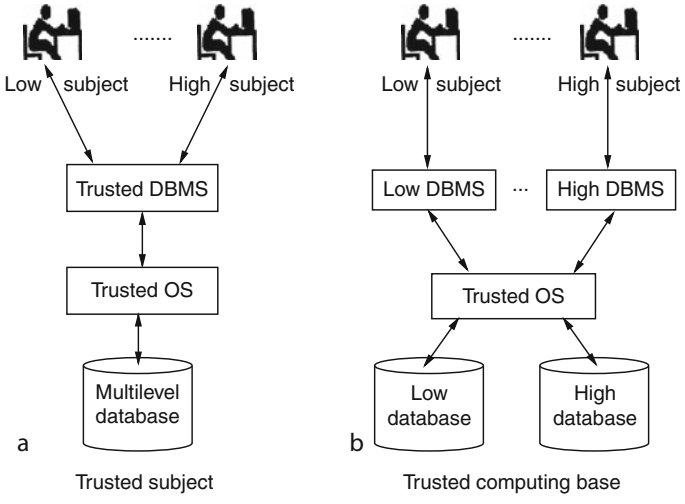
a

Name	λ_N	Age	λ_A	Illness	λ_I
Alice	U	45	U	Gastritis	U
Fred	S	23	S	HIV	S
Carol	U	57	U	Ulcera	U
David	U	79	U	H1N1	S

b

Name	λ_N	Age	λ_A	Illness	λ_I
Alice	U	45	U	Gastritis	U
Carol	U	57	U	Ulcera	U
David	U	79	U	–	U

Multilevel Database. Fig. 1 An example of multilevel relation (**a**) and the Unclassified view on it (**b**)



Multilevel Database. Fig. 2 Multilevel DBMS architecture

adopt this assumption: for example, in LDV [3], a special value "restricted" appears in a subject's view to denote the existence of values not visible to the subject.

To produce a view consistent with the relational database constraints, the classification needs to satisfy at least the following two basic constraints: (1) the key attributes must be uniformly classified, and (2) the classifications of nonkey attributes must dominate that of key attributes. If it were not so, the view at some levels would contain a null value for some or all key attributes (and therefore would not satisfy the key constraints). Further constraints need to be enforced to ensure correctness of the database and, in particular, control possible side-effect of ►polyinstantiation (e.g., proliferation of polyinstantiated tuples) in the execution of update operations. In addition, all relational operations need to be redefined to operate on multilevel databases.

A complicating aspect in the support of a mandatory policy in a database system is that the definition of the access class to be associated with each piece of data is not always easy [1]. This is the case, for example, of *association* and *aggregation* requirements, where the classification of a set of values (properties, resp.) is higher than the classification of each of the values singularly taken. As an example, while patients' names and diseases in a hospital may be considered Unclassified, the association of a specific disease with a patient's name can be considered Secret (association constraint). Similarly, while the location of a single military ship can be Unclassified, the location of all the ships of a fleet can be Secret (aggregation constraint), as by knowing it one could infer that some military operations are being planned. Proper data classification assignment is also complicated by the need to take into account possible inference channels [1, 4, 6]. There is an inference channel between a set of data x and a set of data y if, by knowing x, a user can infer some information on y (e.g., an inference channel can exist between a patient's disease and the treatment she is being cared with). Inference-aware classification requires that no information x be classified at a level lower (or incomparable) than the level of the information y that can be inferred from it. Capturing and blocking all inference channels is a complex process, also because of the intrinsic difficulty of detecting all the semantics relationships between the data that can cause inference channels [1].

An interesting point that must be taken into account in multilevel database systems is the system architecture, which is concerned with the need of confining subjects accessing a multilevel database to the data that can be made visible to them. This problem comes out in any data system where classification has a finer granularity than the stored objects (e.g., multilevel object-oriented systems). Two possible approaches are:

– *Trusted subject*: data at different levels are stored in a single database (Fig. 2a). The DBMS itself must be *trusted* to ensure obedience of the mandatory policy (i.e., subjects will not gain access to data whose classification they do not dominate).
– *Trusted computing base*: data are partitioned in different databases, one for each level (Fig. 2b). Only the operating system needs to be trusted since every DBMS will be confined to access data whose classification is dominated by that of the requesting subject. Decomposition and recovery algorithms must be carefully constructed to be correct and efficient [2].

Applications

Multilevel database systems find application in context where databases contain data that have different sensitivity, and therefore need to be classified at different classifications. MDBMSs give the possibility of associating security classifications with the database objects and ensure that each subject gains access – directly or indirectly – only to those data for which it has proper clearance. Commercial database management systems (e.g., Oracle label security) typically support the specification of multilevel security labels at the level of tuple.

Recommended Reading

1. Dawson S, De Capitani di Vimercati S, Lincoln P, Samarati P (1999) Minimal data upgrading to prevent inference and association attacks. In: Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), Philadelphia
2. Denning DE (Apr 1985) Commutative filters for reducing inference threats in multilevel database systems. In: Proceedings of the 1985 IEEE Symposium on Security and Privacy, Oakland, pp 134–146
3. Haigh JT, O'Brien RC, Thomsen DJ (1991) The LDV secure relational DBMS model. In: Jajodia S, Landwehr CE (eds) Database security, IV: Status and Prospects. Elsevier, North-Holland, pp 265–279
4. Jajodia S, Meadows C (1995) Inference problems in multilevel secure database management systems. In: Abrams MD, Jajodia S, Podell HJ (eds) Information Security: An Integrated Collection of Essays. IEEE Computer Society Press, Los Aamitos, pp 570–584
5. Jajodia S, Sandhu RS (May 1991) Toward a multilevel secure relational data model. In: Proceedings of ACM SIGMOD International Conference on Management of Data, Denver, pp 50–59
6. Lunt TF (1989) Aggregation and inference: facts and fallacies. In: Proceedings of IEEE Symposium on Security and Privacy, Oakland, pp 102–109
7. Lunt TF, Denning DE, Schell RR, Heckman M, Shockley WR (Jun 1990) The SeaView security model. IEEE Trans Softw Eng 16(6):593–607
8. Smith K, Winslett M (1992) Entity modeling in the MLS relational model. In: Proceedings of the 18th International Conference on Very Large Data Bases (VLDB '92), Vancouver

Multilevel Security Policies

Frédéric Cuppens, Nora Cuppens-Boulahia
LUSSI Department, TELECOM Bretagne, Cesson Sévigné CEDEX, France

Related Concepts

►Bell and LaPadula Model; ►The Biba Model; ►Mandatory Access Control Policy (MAC)

Definition

A multilevel security (MLS) policy assigns security levels, called classification, to information and security levels, called clearance, to users. The set of security levels is generally partially ordered and the objective of multilevel security policies is to prevent users from accessing information for which they lack ►authorization.

Background

Since the beginning of the 1970s, many works have been undertaken to design information systems that support MLS policies. These include implementation of MLS policies in networks, operating systems, and applications like database management systems. Most of these projects implement the Bell and LaPadula (BLP) model [1] which suggests an operational interpretation of MLS policies for information systems. See the ►Bell and LaPadula model entry for more information.

Theory

There are actually two main classes of MLS policies: MLS policies for managing confidentiality and MLS policies for managing integrity. In both cases, security levels, called classification and clearance, are respectively assigned to information and users.

So to define MLS policy, it is necessary to first specify a set of security levels. This set of security levels is associated with a partial order relation denoted <. If l_1 and l_2 are two security levels then $l_1 < l_2$ is read l_1 is lower than l_2. When < is a partial order relation, it may happen that both $l_1 < l_2$

and $l_2 < l_1$ and $l_1 = l_2$ are false. In this case, one says that l_1 and l_2 are incomparable, which is denoted $l_1 <> l_2$.

In the case of MLS policies for managing confidentiality, security levels are called confidential levels. A typical example is to consider that a confidential level has two parts which respectively correspond to a sensitivity level and a set of compartments. Examples of sensitivity are confidential, secret and top secret. Generally, sensitivities are associated with a total order relation defined as follows: confidential < secret < top secret. Compartments may represent application-dependent categories or domains, for example, nuclear, chemical, or transport.

A security level l is defined as a pair (s,c), where s is a sensitivity level and c is a set of compartments. If $l_1 = (s_1, c_1)$ and $l_2 = (s_2, c_2)$ are two security levels, then $l_1 < l_2$ is defined as follows:

- $l_1 < l_2$ if and only if $s_1 < s_2$ and $c_1 \subseteq c_2$.

For example, (confidential, {transport}) < (secret, {transport,chemical}) and (confidential, {chemical, transport}) <> (secret, {chemical}).

In addition to the sensitivity and compartments, classified information may also bear various caveats to restrict the access to specific nationality groups. Examples of caveats are "UK eyes only" used in the UK or "Special France" used in France.

The classification level assigned to information represents how confidential this information must be considered in the organization. This means that the organization must define precise security rules to classify their information. Most countries have defined such rules to manage classification of governmental, health care, or military information.

The clearance level assigned to a user represents how trusted the user may be considered when he or she manages classified information. For this purpose and for the different security levels, organizations defined investigation procedures to assign clearance level to users. If a given user successfully complies with the procedures associated with some security level, then he or she can become cleared at this level after being briefed and signing some nondisclosure agreement form.

Once some user becomes cleared at some security level, two different conditions must be satisfied in order this user may get an access to read some information: (1) the classification level of this information must be lower than or equal to the clearance level of this user; (2) this user must need to know this information. The first condition is generally called "no read up" and corresponds to ►mandatory access control (MAC) [3]. The second condition is implemented by a control performed by some user, generally the owner of the information, who will evaluate if the user requesting the access actually needs to know the requested information. This second condition is generally called ►discretionary access control (DAC) [3]. For example, a user may need to know some document classified (secret, {transport,chemical}) because he or she has a position at the Transport ministry and is working on a regulation of chemical transport.

Regarding MLS policies for managing integrity, one needs to consider a set of security levels called integrity levels which are generally distinct from confidentiality levels. Similar to MLS confidentiality policies, in MLS integrity policies, classification and clearance levels are respectively assigned to information and users with the following interpretation. A classification level assigned to some information represents the degree of confidence that may be placed in the information. A clearance level assigned to some user represents the degree of trustworthiness that may be placed in the information inserted or updated by this user.

In MLS integrity policies, a user may get an access to write (i.e., insert, update, or delete) some information only if the classification of this information must be lower than or equal to the clearance level of this user ("no write up").

Most of information systems that support MLS confidentiality policies also support MLS integrity policies. In that case, these information systems generally implement the ►Biba model [2]. See the Biba model entry for more information.

Recommended Reading

1. Bell D, LaPadula L (1975) Secure computer systems: unified exposition and multics interpretation, Technical Report ESD-TR-75-306, MTR-2997. The MITRE Corporation, Bedford
2. Biba K (1976) Integrity considerations for computer systems, Technical Report, ESD-TR-76-372. The MITRE Corporation, Bedford
3. Department of Defense (1983) Trusted computer systems evaluation criteria, Technical Report CSC-STD-001-83

Multiparty Computation

Berry Schoenmakers
Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven, Eindhoven,
The Netherlands

Synonyms

Secure computation; Secure function evaluation; Secure multiparty computation

Related Concepts

►Secure Multiparty Computation (SMC); ►Threshold Homomorphic Cryptosystems; ►Verifiable Secret Sharing

Definition

Let f denote a given n-ary function, and suppose parties $P_1, \ldots, P_n$ each hold an input value $x_1, \ldots, x_n$, respectively. A *secure multiparty computation* for f is a joint ►protocol between parties $P_1, \ldots, P_n$ for computing $y = f(x_1, \ldots, x_n)$ *securely*. That is, even when a certain fraction of the parties is corrupted, (1) each party obtains the correct output value y and (2) no information leaks on the input values of the honest parties beyond what is implied logically by the value of y and the values of the inputs of the corrupted parties.

Background

Conceptually, a secure multiparty computation for function f can be viewed as an implementation of a ►trusted party T, which, upon receipt of the input values $x_1, \ldots, x_n$ from parties $P_1, \ldots, P_n$, respectively, produces the output value $y = f(x_1, \ldots, x_n)$. Party T is trusted for (1) providing the correct value for y and (2) not revealing any further information to parties $P_1, \ldots, P_n$.

A classical example is Yao's millionaires problem $(n = 2)$. Parties P_1 and P_2 are two millionaires who want to see who is richer: writing x_1, x_2 for their respective wealths, they want to evaluate the function $f(x_1, x_2) = x_1 > x_2$. They could simply do so by telling each other the values of x_1 and x_2 but obviously this way much more information than the value of $x_1 > x_2$ is revealed. A secure two-party protocol allows them to compute the value of $x_1 > x_2$ without leaking any further information on x_1 and x_2. ►Electronic voting is another example of a secure multiparty computation, where $f(x_1, \ldots, x_n) = x_1 + \cdots + x_n$ and $x_i \in \{0,1\}$ represent each party P_i's yes–no vote.

More generally, a secure multiparty computation may be *reactive* in the sense that parties $P_1, \ldots, P_n$ may interact with the trusted party T in more than one round. Each round starts with one or more parties providing input values and ends with one or more parties receiving output values. The special case in which the entire computation consists of a single round is also called *secure function evaluation*, and in this case the trusted party does not need to keep any state. In some settings secure function evaluation is strictly easier to achieve than (reactive) secure multiparty computation, see [10] and references therein.

Theory

The theory of secure multiparty computation shows that a protocol for evaluating a given function f securely can be found, as long as f is a *computable* function, while imposing certain restrictions on the power of the corrupted parties, who are collectively called the *adversary*. A first distinction is whether the adversary is assumed to be computationally restricted, or not. In the *cryptographic model*, the adversary is assumed to be *polynomially* restricted (i.e., the adversary is viewed as a probabilistic ►polynomial-time Turing machine). In the ►*information-theoretic model* no such restriction is assumed for the adversary. For the cryptographic model, it suffices to assume *authentic* channels for each pair of parties: the messages exchanged over authentic channels cannot be changed by other (corrupted) parties; using ►encryption it is possible to hide the content of the messages. For the information-theoretic model one needs to assume a *private* (or *secure*) channel is available to each pair of parties: the messages exchanged over private channels cannot be seen at all by other (corrupted) parties.

The adversary is called *passive* (or *honest-but-curious*, or *semi-honest*) if it only tries to deduce information on the inputs of the honest parties by inspecting all the information available to the corrupted parties; the adversary is called *active* (or *malicious*) if it is also allowed to let the corrupted parties deviate from the protocol in arbitrary ways. A further distinction is whether the adversary is allowed to choose which parties to corrupt adaptively. A *static* adversary must decide at the start of the protocol which parties it chooses to corrupt. An *adaptive* (or *dynamic*) adversary may decide during the protocol which parties it chooses to corrupt; once corrupted, however, a party remains so for the entire duration of the protocol.

A threshold (►Threshold Scheme) parameter $t, 1 \le t \le n$, is used to indicate the maximum number of corrupted parties tolerated by a protocol for secure multiparty computation. As long as the number of corrupted parties does not exceed t, the protocol protects the interests of the honest parties. In terms of t, the main results for secure multiparty computation are as follows, where in each case the adversary may be adaptive. For the cryptographic model, any $t < n/2$ is achievable for an active adversary; for a passive adversary, this can be improved to $t < n$. For the information-theoretic model, any $t < n/3$ is achievable for an active adversary; for a passive adversary this can be improved to $t < n/2$.

It is important to note that in case of an active adversary, the condition saying that each party obtains the correct output value y, can be split into two further conditions: the condition that each party actually receives an output

value and the condition that, if an output value is received, then the output value is correct. If a protocol guarantees that each party receives an output value, the protocol is said to be *robust*.

The results above show that an honest majority is required to deal with an active adversary (if robustness is required). Therefore, these results are not useful for the special case of two-party computation: for two parties, an honest majority comprises *all* of the parties ($t = 0,\ n = 2$). For secure two-party computation, the property of robustness is thus replaced by the property of *fairness*. A protocol for two-party computation is said to be fair, if neither party can gain an advantage over the other party by quitting the protocol prematurely. For instance, a two-party protocol in which party P_1 learns the result first, and needs to send it to party P_2 is not fair, as P_1 may simply skip sending the result to P_2. Solutions to address this problem typically use a form of gradual release of a secret value, where the output value is released bit by bit and quitting by either party gives an advantage of at most one bit over the other party. Solutions achieving that neither party gets any advantage at all over the other party are impossible, as proved by Cleve [7].

There exists a vast body of literature on secure multiparty computation. The paper by Yao [13] (and also [14]) and subsequent papers [1–3, 5, 6, 9, 12] build foundations for general secure multiparty computation, yielding the results mentioned above for various settings. (Refer also ►Verifiable Secret Sharing). The strength of ►oblivious transfer is stressed by the result of [11]. In a similar direction, the results of [4, 8] show that threshold ►homomorphic cryptosystems provide a basis for efficient general secure multiparty computation as well.

A quick way to see why secure multiparty computation is possible at all, runs as follows, following [4, 8]. The basic primitive is a (probabilistic) threshold ►homomorphic cryptosystems E, where the private key is shared among parties $P_1, \ldots, P_n$. Such a public key cryptosystem is homomorphic in the sense that the product of two ciphertexts $E(x)$ and $E(y)$ results in a ciphertext $E(x + y)$, containing the sum of the values x and y. It is a threshold cryptosystem in the sense that decryption of a ciphertext $E(x)$ is done by a joint protocol between $P_1, \ldots, P_n$, resulting in the value of x, and as long as a majority of the parties is honest, ciphertexts will only be decrypted if a majority of parties agrees to do so.

Suppose function f is to be evaluated at $x_1, \ldots, x_n$, where x_i is the private input supplied by party P_i, for $i = 1, \ldots, n$. One assumes that function f is represented as an arithmetic circuit consisting of addition gates and multiplication gates (where additions and multiplications are defined over $\mathbb{Z}_N$, the integers modulo N, for a fixed integer $N \geq 2$). The protocol for secure computation of f then proceeds as follows: First, each party encrypts its private input value, yielding ciphertexts $E(x_1), \ldots, E(x_n)$. The circuit is then evaluated gate by gate, as described below, ultimately producing $E(f(x_1, \ldots, x_n))$ as encrypted output, from which the value $f(x_1, \ldots, x_n)$ is obtained, using threshold decryption.

The gates are evaluated as follows. An addition gate takes as input two ciphertexts $E(x)$ and $E(y)$ and produces as output a ciphertext $E(x+y)$, simply using the homomorphic property of E. A multiplication gate also takes as input two ciphertexts $E(x)$ and $E(y)$, but this time a protocol is required to produce $E(xy)$ as output value. The protocol for the passive (semi-honest) case runs as follows, omitting the ►zero-knowledge proofs to stop active adversaries:

1. Each party P_i, $1 \leq i \leq n$, picks a random value d_i and broadcasts ciphertexts $E(d_i)$ as well as $E(d_i y)$, where $E(d_i \mathrm{y})$ can be computed easily from d_i and $E(y)$.
2. Let $d = \sum_{i=1}^{n} d_i$. Using the homomorphic property of E, the parties compute ciphertext $E(x + d) = E(x) \prod_{i=1}^{n} E(d_i)$, from which they subsequently determine $x + d$, using threshold decryption. From $x + d$ and $E(y)$, one may then compute $E((x + d)y)$. Finally, using $E(dy) = \prod_{i=1}^{n} E(d_i y)$, one obtains $E(xy) = E((x + d)y)/E(dy)$, which is the desired output.

Note that all computations on x, y, and d_i's are done modulo N. Intuitively, the protocol is secure because the only values ever decrypted – apart from the output value $f(x_1, \ldots, x_n)$ – are values $x + d$, where d is distributed uniformly at random and chosen jointly by $P_1, \ldots, P_n$.

Recommended Reading

1. Ben-Or M, Goldwasser S, Wigderson A (1988) Completeness theorems for noncryptographic fault-tolerant distributed computation. In: Proceedings of 20th symposium on theory of computing (STOC'88). ACM Press, New York, pp 1–10
2. Beaver D, Haber S (1992) Cryptographic protocols provably secure against dynamic adversaries. In: Rueppel RA (ed) Advances in cryptology – eurocrypt'92. Lecture notes in computer science, vol 658. Springer, Berlin, pp 307–323
3. Chaum D, Crépeau C, Damgård I (1988) Multiparty unconditionally secure protocols. In: Proceedings of 20th symposium on theory of computing (STOC'88). ACM Press, New York, pp 11–19
4. Cramer R, Damgård I, Nielsen JB (2001) Multiparty computation from threshold homomorphic encrytpion. In: Pfitzmann B (ed) Advances in cryptology – eurocrypt 2001. Lecture notes in computer science, vol 2045. Springer, Berlin, pp 280–300
5. Canetti R, Feige U, Goldreich O, Naor M (1996) Adaptively secure multi-party computation. Proceedings of 28th

symposium on theory of computing (STOC'96). ACM Press, New York, pp 639–648
6. Canetti R, Lindell Y, Ostrovsky R, Sahai A (2002) Adaptively secure multi-party computation. In: Proceedings of 34th symposium on theory of computing (STOC 2002). ACM Press, New York, pp 494–503
7. Cleve R (1986) Limits on the security of coin flips when half of the processors are faulty. In: Proceedings of the 18th symposium on theory of computing (STOC'86). ACM Press, New York, pp 364–369
8. Damgård I, Nielsen JB (2003) Universally composable efficient multiparty computation from threshold homomorphic encryption. In: Boneh D (ed) Advances in Cryptology—CRYPTO 2003. Lecture notes in computer science, vol 2729. Springer, Berlin, pp 247–264
9. Goldreich O, Micali S, Wigderson A (1987) How to play any mental game – or – a completeness theorem for protocols with honest majority. Proceedings of 19th symposium on theory of computing (STOC'87). ACM Press, New York, pp 218–229
10. Hirt M, Maurer U, Zikas V (2008) MPC vs. SFE: Unconditional and computational security. In: Pieprzyk J (ed) Advances in cryptology – asiacrypt 2008. Lecture notes in computer science, vol 5350. Springer, Berlin, pp 1–18
11. Kilian J (1988) Basing crpytography on oblivious transfer. In: Proceedings of 20th symposium on theory of computing (STOC'88). ACM Press, New York, pp 20–31
12. Rabin T, Ben-Or M (1989) Verifiable secret sharing and multiparty protocols with honest majority. In: Proceedings of 21st symposium on theory of computing (STOC'89). ACM Press, New York, pp 73–85
13. Yao A (1982) Protocols for secure computations. Proceedings of 23rd IEEE symposium on foundations of computer science (FOCS'82). IEEE Computer Society, pp 160–164
14. Yao A (1986) How to generate and exchange secrets. Proceedings of 27th IEEE symposium on foundations of computer science (FOCS'86). IEEE Computer Society, pp 162–167

Multiparty Computation (MPC)

►Secure Multiparty Computation (SMC)

Multiple Encryption

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Triple DES

Definition

Composition of several ciphers is called *multiple encryption* or *cascade cipher*. Refer also ►product cipher.

Multiple Independent Levels of Security

Jim Alves-Foss
Department of Computer Science, University of Idaho, Moscow, ID, USA

Synonyms

MILS

Related Concepts

►Mandatory Access Control; ►Multilevel Security Policies; ►Reference Monitor; ►Security Architecture; ►Trusted Computing Base

Definition

The Multiple Independent Levels of Security architecture

Background

In the 1970s, there was active research into the development of secure operating systems. Out of that work came the concept of a security kernel as discussed by Lampson and Sturgis [1], and by Popek and Kline [2]. A security kernel provides the basic operating system functionality needed to run services and user applications, and provides the basic security mechanisms of the system. The portion of the kernel that enforces security is often referred to as the reference monitor, a concept described separately by Anderson [3] and Lampson [4]. The reference monitor determines if access should be permitted and can be part of the security kernel, a separate guard a firewall or other security mechanisms. The reference monitor should be tamperproof, non-bypassable, and evaluatable.

In the 1980s, John Rushby [5, 6] expanded on the concepts of a security kernel and reference monitor by introducing the concept of a separation kernel. The idea of a separation kernel was based on a layering concept where the kernel enforced controlled separation and well-defined interfaces between individual processes, including services and user applications.

In the early 2000s, microprocessor technology was advanced enough that the concept of full virtual machines, running on a separation kernel was realizable on a large number of commercial microprocessor platforms. The National Security Agency with support from industry, academia, and the Air Force Research Lab decided to expand upon Rushby's ideas and advances of the intervening 20 years to develop a separation-based architecture that was evaluatable under the common criteria. The result of this effort was the MILS architecture and the

US Government Protection Profile for Separation Kernels in Environments Requiring High Robustness [7].

Applications

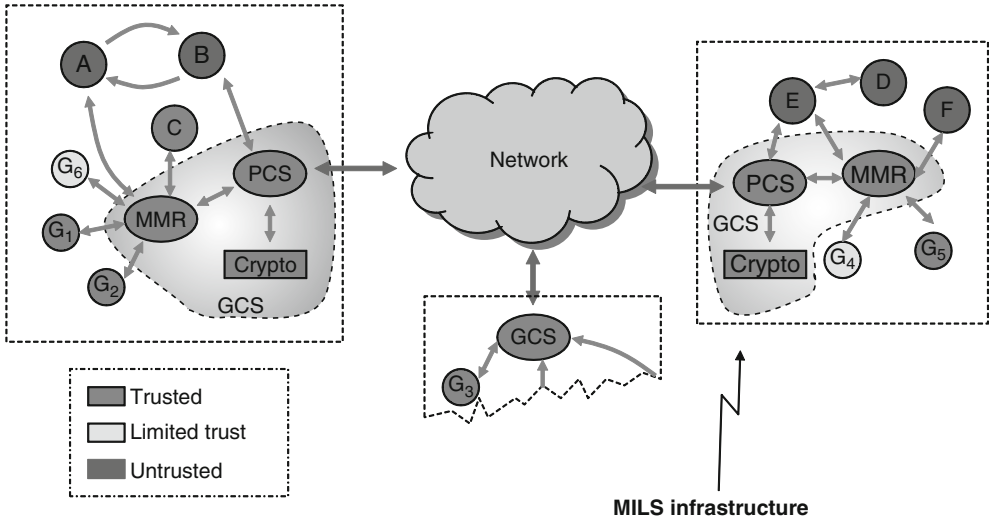
Modern net-centric concepts are based upon ubiquitous connectivity and standards-based services. This is in fact the basis upon which the Department of Defense Information Enterprise Architecture is founded. However to realize the objective of secured availability requires that users and process with various levels of trust and access share a common infrastructure. The emerging state of the art on this type of Multiple Level Security is based upon the Multiple Independent Levels of Security (MILS) architecture [8, 9]. This architecture consists of a number of high robustness components that when combined appropriately can be trusted to enforce two primary principles of Information Assurance.

- Information is only shared with those processes and users allowed by policy.
- Information is not shared with those processes and users disallow by policy.

In the Multiple Independent Level Security (MILS) architecture, multi-level secure systems are implemented through separation and controlled information flow. The system is built on a foundation consisting of a separation-based infrastructure, the separation kernel (i.e., hypervisor), and secure inter-processor communication (i.e., the partitioned communication service, PCS). These components isolate individual applications and services and provide the pathways for secure communication. Supporting these pathways are additional components (i.e., guards, cross-domain services, encryption engines, and routers) that implement the system security policy. One approach to the MILS architecture is to develop a *guarded communication subsystem* (GCS), which is responsible for the "routing" of messages between applications, sending them through the appropriate filters, guards, and access decision points.

Consider the exemplary system depicted in Fig. 1. In this figure, we have three processors, each hosting a number of processes. Some of the processes (A, B, D, E, and F) are applications that have not been fully analyzed with respect to security, and are thus considered untrusted. Assume application A needs to utilize services provided by application F. Requests from A are passed through the MMR (MILS message router [10]) that first sends the message through the appropriate guards (e.g., G_1 or G_2) and then passes it on to the PCS, which securely transmits the message to Processor 2 and its MMR. Processor 2 MMR may pass the message through Guard G_5 first, and then to the F service engine. Along the way the guards may accept the message, modify it (e.g., add additional metadata, filter contents), or reject it.

The security policy of the system depicted in Fig. 1 includes all of the specifications of authorized requests and communication between the untrusted applications. For example, the policy can specify the content and format of requests from A to F (A may be running a Secret level application).



Multiple Independent Levels of Security. Fig. 1 Secure multi-level communication system, implemented using guarded communication subsystems

The security policy can be specified as a conjunction of predicates/operations performed on the messages as they travel between the processes [11]. Each guard can be responsible for enforcement of one or more aspect of the policy. The system as a whole enforces the totality of the policy.

MILS Principles

The MILS approach to secure system architecture is based on a set of design principles that can be summarized with the following [8, 9]:

Time and Space Separation: The MILS architecture requires that the system be architected as a set of functional units, called "partitions" supported by one or more separation mechanisms (e.g., separation kernel, partitioned communication system). Each partition represents a well-defined set of resources and functionality. The MILS separation mechanisms ensure that private resources (e.g., memory, I/O devices) of a partition are kept isolated from other partitions, including residual data in shared resources, hence space separation. In addition, the execution behavior of one partition should not unduly influence the execution of another partition, hence time separation.

Controlled Information Flow: The concept of design modularity requires the development of multiple partitions with communication between the partitions. The MILS architecture supports this with the concept of controlled information flow. The MILS separation mechanisms will allow information to flow only along defined communication paths – allowing a controlled exception to full data separation. With this controlled flow, system architects can require that messages be processed by access guards, information re-graders, or other security enforcing components.

Separation Security Policy (*TIME*): The MILS separation mechanisms enforce policies of *type-safety, infiltration*, *mediation*, and *exfiltration*. Type safety specifies that the data types of the information flow mechanisms are preserved (e.g., the controlled information flow will not allow overwriting of a bounded buffer). Infiltration specifies that an executing partition is not able to read or otherwise be influenced by private data of another partition (or the separation mechanism). Exfiltration specifies that private data of executing partition cannot be written to, modify or otherwise influence the private data of another partition. Mediation specifies that an executing partition cannot use private data from one partition to modify or otherwise influence private data of another partition.

Reference Monitor (*NEAT*): MILS separation mechanisms implement the reference monitor concepts such that they are *non-bypassable*, *evaluatable*, *always invoked*, and *tamperproof*. The separation mechanism will always be invoked to control information flow and manage access of private and shared data. In addition, the system is implemented such that there is no other way to provide information flow or access to a partition's data, except through the separation mechanism, hence non-bypassable. To provide high levels of assurance that the system correctly implements the TIME security policy, the system must be designed in a manner that prevents tampering with the separation mechanism and the mechanism must be simple enough to allow for full evaluation.

As can be seen from these principles, the MILS architecture can be used for an operating system running on a single microprocessor. However, it is not limited to that environment. MILS can support one or more processor cores, either as a multicore processor or as separate parallel processors or individual machines. The architectural concepts can be abstracted to many different levels of granularity treating each "partition" as a single virtual machine on a single processor up to a collection of physical machines. The required separation between partitions can be physical, cryptographic, or logical; they can be enforced through microprocessor virtualization technology, memory management units, cryptography, or physical placement and wiring.

Recommended Reading

1. Lampson B, Sturgis H (May 1976) Reflections on an operating system design, Commun Assoc Comput Mach 19(5): 251–266
2. Popek G, Kline C (Jan 1978) Design issues for secure computer networks. In: Operating systems: an advanced course. Lecture Notes in Computer Science, vol 60. Springer, pp 517–546
3. Anderson J (1972) Computer security technology planning study, ESD-TR-73-51, ESD/AFSC, Hanscom AFB, Bedford. http://csrc.nist.gov/publications/history/ande72.pdf
4. Lampson B (Jan 1974) Protection, In: Proceedings of Princeton symposium 1971. Reprinted in Oper Syst Rev 8(1):18–24
5. Rushby J (1981) Design and verification of secure systems. In: Proceedings of eighth ACM symposium on operating system principles, Asilomar, pp 12–21. http://www.csl.sri.com/papers/sosp81/sosp81.pdf
6. Rushby J (1999) Partitioning for safety and security: requirements, mechanisms, and assurance. Tech Report: CR-1999-209347, NASA Langley Research Center
7. U.S. Government protection profiles for separation kernels in environments requiring high robustness. Version 1.03, July 2007
8. Alves-Foss J, Harrison WS, Oman P, Taylor C (2007) The MILS architecture for high assurance embedded systems. Int J Embed Syst 2:239–247
9. Harrison WS, Hanebutte N, Oman P, Alves-Foss J (Oct 2005) The MILS architecture for a secure global information grid.

CrossTalk 18(10):20–24. http://www.stsc.hill.af.mil/CrossTalk/2005/10/0510Harrisonetal.html
10. Rossebo B, Oman P, Alves-Foss J, Blue R, Jaszkowiak P (2006) Using spark-ada to model and verify a MILS message router. In: Proceedings of international symposium on secure software engineering, Washington, DC
11. Zhou J, Alves-Foss J (2008) Security policy refinement and enforcement in secure computer systems design. J Comput Secur 16(2):107–131

Multiplicative Knapsack Cryptosystem

David Naccache
Département d'informatique, Groupe de cryptographie, École normale supérieure, Paris, France

Related Concepts

►Gröbner Bases Algorithms; ►Public Key Encryption

Definition

The multiplicative knapsack public-key cryptosystem was proposed by Naccache and Stern. The scheme's general outline is the following: Let p_i denote the ith prime number (starting from $p_1 = 2$), fix a message bit length $k \in \mathbb{N}$ and let $p > \prod_{i=1}^{k} p_i$ be a large public prime. Alice selects a random key $1 < s < p - 1$, and publishes the public key $\{v_1, \ldots, v_k\}$ where $v_i^s = p_i \bmod p$. Let $\{m_1, \ldots, m_k\}$ be a binary message. Bob sends to Alice the ciphertext:

$$c = \prod_{i=1}^{k} v_i^{m_i} \bmod p$$

To decrypt c, Alice computes the quantity:

$$u = c^s \bmod p = \left(\prod_{i=1}^{k} v_i^{m_i} \right)^s \bmod p = \prod_{i=1}^{k} p_i^{m_i} \bmod p$$

Since $p > u$, Alice can factor u in $\mathbb{Z}$ and recover $\{m_1, \ldots, m_k\}$.

Open Problems

Prove or cryptanalyze this scheme.

Recommended Reading

1. Naccache D, Stern J (1997) A new public key cryptosystem. In: Fumy W (ed) Advances in cryptology – EUROCRYPT'97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 27–36

Multiprecision Multiplication

Berk Sunar
Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, USA

Synonyms

Big number multiplication

Related Concepts

►Modular Multiplication

Definition

Multiprecision multiplication is the fundamental arithmetic operation of computing the product of two integers where the numbers are represented using multiple computer words.

Background

Many cryptographic primitives require the multiplication of integers that do not fit into the standard precision of common microprocessors, e.g., 32 or 64 bits. Thus there is a rich literature of interesting and sometimes exotic multiprecision mutliplication algorithms.

Theory

The integer multiplication operation lies at the very heart of many cryptographic algorithms, especially public-key algorithms. Naturally, much effort went into developing efficient multiplication algorithms. The simplest of such algorithms is the classical "grammar school" multiplication method given as follows:

Multiprecision Multiplication Algorithm
Input: positive integers $u = (u_{m-1}u_{m-2}\ldots u_1u_0)_B$ and $v = (v_{n-1}v_{n-2}\ldots v_1v_0)_B$
Output: The integer product $t = u \cdot v$
For $i = 0$ to $m + n + 1$ do $t_i \leftarrow 0$;
$c \leftarrow 0$;
For $i = 0$ to m do
  For $j = 0$ to n do
    $(cs)_B \leftarrow t_{i+j} + u_i \cdot v_j + c$;
    $t_{i+j} \leftarrow s$;
  End For
  $t_{i+n+1} \leftarrow c$;
End For
Return (t)

The algorithm proceeds in a *row-wise* manner. That is, it takes one digit of one operand and multiplies it with the digits of the other operand, in turn appropriately shifting

and accumulating the product. The two digit intermediary result $(cs)_B$ holds the value of the digit t_{i+j} and the carry digit that will be propagated to the digit t_{i+j+1} in the next iteration of the inner loop. The subscript B indicates that the digits are represented in radix-B. As can be easily seen the number of digit multiplications performed in the overall execution of the algorithm is $m \cdot n$. Hence, the time complexity of the classical multiplication algorithm grows with $O(n^2)$, where n denotes the size of the operands.

There is a wealth of multiplication algorithms beyond the grammar school methods. One of the most popular ones in practice is the ►Karatsuba algorithm. Many other algorithms attempt to exploit a connection between the linear convolution and multiplication operations. Consider the multiplication of two polynomials. When two sequences are constructed using the polynomial coefficients and when their linear convolution is computed, the elements of the resulting sequence give the coefficients of the product of the two polynomials. Hence, any linear convolution algorithm may easily be adapted to compute polynomial multiplications. Furthermore, evaluating the operand and the product polynomials at $x = B = 2^w$, where w is the digit size, yields an algorithm for integer multiplication.

For instance, the elements of a 3-point convolution of the sequences $\{u_0, u_1, u_2\}$ and $\{v_0, v_1, v_2\}$ are given as

$$\begin{aligned} w_0 &= u_0 v_0 \\ w_1 &= u_1 v_0 + u_0 v_1 \\ w_2 &= u_2 v_0 + u_1 v_1 + u_0 v_2 \\ w_3 &= u_2 v_1 + u_1 v_2 \\ w_4 &= u_2 v_2 \,. \end{aligned}$$

These expressions are exactly in the form of the coefficients of the product of the two polynomials $U(x) = u_0 + u_1 x + u_2 x^2$ and $V(x) = v_0 + v_1 x + v_2 x^2$. Evaluating the product polynomial for a particular *radix* size $B = 2^w$ gives an algorithm for integer multiplication where polynomial coefficients represent the digits of the two integer operands. Note that the coefficients w_i may not exactly fit into a digit. Therefore, a digit carry-over operation needs to be performed through the entire length of the product in the final integer conversion step.

A well-known convolution algorithm was introduced by Toom and Cook [1]. The Toom–Cook algorithm works by treating the two operands as polynomials $U(x)$ and $V(x)$ of maximum degree $k-1$. This is done by partitioning the integer representations into k digits. Both polynomials are evaluated at $2k-1$ points and multiplied together.

$$W(x_i) = U(x_i) \cdot V(x_i), \quad i = 0, 1, \ldots, 2k-2 \,.$$

This gives the evaluation of the product $U(x)V(x)$ at $2k-1$ points which are used to form $2k-1$ equations with the coefficients of $W(x)$ as unknowns. Solving the linear system of equations gives the coefficients of the product $W(x) = U(x) \cdot V(x)$. Finally, $W(x)$ is evaluated at $B = 2^w$ and the product is obtained in integer form.

Example 1 (Toom–Cook multiplication) A 3-point Toom–Cook multiplication algorithm is derived. Let

$$U(x) = u_0 + u_1 x + u_2 x^2 \quad \text{and} \quad V(x) = v_0 + v_1 x + v_2 x^2 .$$

Arbitrarily pick a sequence S of $2k - 1 = 5$ points. Let $S = \{0, 1, 2, -1, -2\}$. Then evaluate $U(x)$ and $V(x)$ for each element in S and compute their products as follows:

$$\begin{aligned} W(0) &= U(0)V(0) = u_0 v_0 \\ W(1) &= U(1)V(1) = (u_0 + u_1 + u_2)(v_0 + v_1 + v_2) \\ W(2) &= U(2)V(2) = (u_0 + 2u_1 + 4u_2)(v_0 + 2v_1 + 4v_2) \\ W(-1) &= U(-1)V(-1) = (u_0 - u_1 + u_2)(v_0 - v_1 + v_2) \\ W(-2) &= U(-2)V(-2) = (u_0 - 2u_1 + 4u_2) \\ &\qquad (v_0 - 2v_1 + 4v_2) \end{aligned}$$

These give us the evaluations of the product polynomial

$$W(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 \,.$$

Note that when all computations are done symbolicly, the evaluations of $W(x)$ are products of linear expressions of the coefficients of $U(x)$ and $V(x)$. In the final step these products are related to the coefficients of $W(x)$ by forming the following system of equations:

$$\begin{aligned} W(0) &= w_0 \\ W(1) &= w_0 + w_1 + w_2 + w_3 + w_4 \\ W(2) &= w_0 + 2w_1 + 4w_2 + 8w_3 + 16w_4 \\ W(-1) &= w_0 - w_1 + w_2 - w_3 + w_4 \\ W(-2) &= w_0 - 2w_1 + 4w_2 - 8w_3 + 16w_4 \end{aligned}$$

By solving the equations the coefficients are obtained as follows:

$$\begin{aligned} w_0 &= W(0) \\ w_1 &= \frac{2}{3}W(1) - \frac{1}{12}W(2) - \frac{2}{3}W(-1) + \frac{1}{12}W(-2) \\ w_2 &= -\frac{5}{4}W(0) + \frac{2}{3}W(1) - \frac{1}{24}W(2) + \frac{2}{3}W(-1) \\ &\quad - \frac{1}{24}W(-2) \\ w_3 &= -\frac{1}{6}W(1) + \frac{1}{12}W(2) + \frac{1}{6}W(-1) - \frac{1}{12}W(-2) \\ w_4 &= \frac{1}{4}W(0) - \frac{1}{6}W(1) + \frac{1}{24}W(2) - \frac{1}{6}W(-1) \\ &\quad + \frac{1}{24}W(-2) \end{aligned}$$

With the equations relating the coefficients of the product $W(x) = U(x)V(x)$ to the coefficients of the input operands, the multiplication algorithm is obtained.

As seen in the example above only in the second step of Toom–Cook's algorithm multiplications are computed. The number of multiplications is fixed as $2n-1$. In the initial polynomial evaluation step multiplications with small integers are performed which are ignored here, since they may be implemented with inexpensive shifts and additions. In the final step where the coefficients of $W(x)$ are computed divisions by small integers are required. In fact as the length of the convolution grows the fractions grow radically. Considering a recursive implementation of the Toom–Cook algorithm the complexity can be shown to be $O(n^{\log_k(2k-1)})$ [2]. By choosing appropriately large k, the complexity can be brought close to $O(n^{1+\epsilon})$ for any $\epsilon > 0$ value. It should be noted, however, that this complexity figure ignores the additions as well multiplications and divisions with small constant integers. The number of such operations becomes more serious as k grows (and ϵ decreases).

As noted before, the multiplication operation is equivalent to linear convolution. This immediately suggests a Fourier Transform based approach for multiplication. One advantage of this technique over the Toom–Cook method and other direct convolution methods is in the lower number of additions and constant multiplications which were ignored in the complexity figure. But more importantly, it allows one to utilize Fast Fourier Transform techniques and achieve $O(n \log n)$ speed.

The Discrete Fourier Transform of a sequence is defined as follows.

Definition 1 (*Discrete Fourier Transform (DFT)*). *Let s be a sequence of length d consisting of elements from an algebraic domain. Let g be a primitive d-th root of unity in that domain, i.e., $g^d = 1$ and let d be invertible in the domain. Then the Discrete Fourier Transform of s is defined as the sequence* $\mathbf{S}$ *whose elements are given as*

$$\mathbf{S}_k = \sum_{i=0}^{d-1} s_i g^{ik}.$$

The inverse transform is defined as

$$s_k = \frac{1}{d} \sum_{i=0}^{d-1} \mathbf{S}_i g^{-ik}.$$

There are many choices for the domain of the transformation such as a complex field, a ►finite field, or an integer ring. If the domain is a complex field, floating point operations may be needed and special attention must be given to handle rounding errors. If the domain is chosen as a finite field, then modular reductions become necessary. The third choice, an integer ring, gives more flexibility in choosing the modulus. In practice, special moduli of form $2^k \pm 1$ may be chosen to eliminate costly reductions.

After a domain is chosen and a DFT is setup as defined above, the following outlines an integer multiplication algorithm:

1. Partition both integer operands into equal sized blocks treating them as sequence elements.
2. Compute the DFT of both sequences.
3. Compute the componentwise product of the DFT of the two sequences.
4. Compute the inverse DFT of the product sequence.
5. Treat the sequence elements as the digits of the integer product.

The asssociated algorithm is given below.

DFT Based Integer Multiplication Algorithm

Input: positive integers $u = (u_{m-1}u_{m-2}\ldots u_1u_0)_B$ and $v = (v_{n-1}v_{n-2}\ldots v_1v_0)_B$

Output: The integer product $w = u \cdot v$

For $k = 0$ to $d-1$ do
  $\mathbf{U}_k \leftarrow \sum_{i=0}^{d-1} u_i g^{ik}$;
  $\mathbf{V}_k \leftarrow \sum_{i=0}^{d-1} v_i g^{ik}$;
End For
For $k = 0$ to $d-1$ do
  $\mathbf{W}_k \leftarrow \mathbf{U}_k \cdot \mathbf{V}_k$;
End For
For $k = 0$ to $d-1$ do
  $w_k = \frac{1}{d} \sum_{i=0}^{d-1} \mathbf{W}_i g^{-ik}$;
End For
Return (w)

While the overall method is quite simple, an efficient algorithm is obtained only if the parameters are carefully chosen and a particular Fast Fourier Transform can be applied for computing the two forward transforms and the final inverse transform.

In an early work Schönhage and Strassen [3] introduced a DFT based integer multiplication method that achieves an exciting asymptotic complexity of $O(n \log n \log\log n)$. The Schönhage and Strassen method is based on the Fermat number transform where the domain of the DFT is the integer ring Z_{2^m+1}. In this method the DFT is recursively turned into shorter DFT's of the same kind. Due to the special ►pseudo-Mersenne structure of the modulus 2^m+1, the method requires no multiplications for implementing the reductions. There are many other methods derived from special DFT and convolution

algorithms. For an excellent survey on DFT based multiplication algorithms the reader is referred to [4].

Despite the tremendous improvement in the asymptotic complexity, the majority of DFT based algorithms have a large computational overhead associated with the forward and inverse transformations. Therefore they only become effective when the operands are longer than several thousand bits, which is often not the case, even for asymmetric cryptographic schemes.

Finally, it is worth recognizing the relationship between the multiplication and squaring operations. Although highly redundant, a multiplication algorithm may be used in a trivial manner to accomplish a squaring computation. On the other hand, an integer multiplication may be achieved via two squarings by using the following simple trick:

$$u \cdot v = \frac{1}{4}\left[(u+v)^2 - (u-v)^2\right]$$

This identity may be useful when a fast squaring algorithm is available.

Applications

The multiprecision multiplication operation is a crucial in the implementation of many cryptographic primitives. Multiplication of large numbers plays a central role in ►public key cryptography where operations such as exponentiation require a large number of integer multiplications to be computed.

Recommended Reading

1. Cook SA (1966) On the minimum computation time of functions. Master's thesis, Harvard University, Boston
2. Knuth DE (1997) The art of computer programming, volume 2: seminumerical algorithms, 3rd edn. Addison-Wesley, Reading
3. Schönhage A, Strassen V (1971) Schnelle multiplikation großer zahlen. Computing 7:281–292
4. Crandall R, Pomerance C (2001) Prime numbers: a computational perspective. Springer, New York
5. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton
6. Karatsuba A, Ofman Y (1963) Multiplication of multidigit numbers on automata. Sov Phys Dokl 7(7):595–596 (English translation)

Multiprecision Squaring

Berk Sunar
Department Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, USA

Synonyms

Big number squaring

Related Concepts

►Multiprecision Multiplication

Definition

Multiprecision squaring is the fundamental arithmetic operation of multiplying a number with itself where the number is represented using multiple computer words.

Background

Many cryptographic primitives require arithmetic operations of integers that do not fit into the precision supported by common microprocessors, for example, 32 or 64 bits. Therefore it becomes essential to build arithmetic algorithms that support arithmetic with multiprecision numbers. One such operation is multiprecision squaring.

Theory

The squaring operation can be thought of as a special case of multiplication, where both operands are the same. In this case, the multiplication algorithm has symmetries which are exploited. Consider a "grammar-school" multiplication performed with four-digit operands $u = (u_3u_2u_1u_0)_B$ and $v = (v_3v_2v_1v_0)_B$ in radix $B = 2^w$ notation. For $u = v$ and $v_i = u_i$, the partial product array is as follows.

$$\begin{array}{cccccccc}
 & & & u_3 & u_2 & u_1 & u_0 \\
\times & & & u_3 & u_2 & u_1 & u_0 \\
\hline
 & & & u_3u_0 & u_2u_0 & u_1u_0 & u_0^2 \\
 & & u_3u_1 & u_2u_1 & u_1^2 & u_0u_1 & \\
 & u_3u_2 & u_2^2 & u_1u_2 & u_0u_2 & & \\
u_3^2 & u_2u_3 & u_1u_3 & u_0u_3 & & &
\end{array}$$

Observe that the array is symmetric across the diagonal, and that the odd-numbered columns have a squared middle term in their diagonal. Hence, in a column only about half of the multiplications need to be computed. The result is multiplied by 2, and if the column is odd numbered the squared term is added. In the following, a multiprecision squaring algorithm that is based on the algorithms in [1, 2] is given.

Multiprecision Squaring Algorithm

Input: positive integer $u = (u_{m-1}u_{m-2}\ldots u_1u_0)_B$
Output: The integer $t = u^2$

```
For i = 0 to 2m − 1 do t_i ← 0 ;
For i = 0 to m − 1 do
    (cs)_B ← t_{2i} + u_i^2 ;
    t_{2i} ← s ;
    For j = i + 1 to m − 1 do
        (cs)_B ← t_{i+j} + 2u_i · u_j + c ;
        t_{i+j} ← s ;
    End For
    t_{i+m} ← s ;
```

End For
$(cs)_B \leftarrow t_{2m-2} + u_{m-1}^2$;
$t_{2m-2} \leftarrow s$; $t_{2m-1} \leftarrow c$;
Return (t)

Note that the product and sum operations performed in the inner loop may not fit into a double digit $(cs)_B$ and require an extra bit due to the term $2u_iu_j$. The total number of multiplications performed in the algorithm is easily seen to be $(m^2 + m)/2$. The multiplication by the constant 2 in the inner loop of the algorithm may be implemented by a simple shift and hence is not counted as a multiplication.

Applications

The integer squaring operation is crucial in the implementation of many cryptographic primitives. Squaring plays a central role in ►public key cryptography where operations such as exponentiation require a large number of integer squarings to be computed.

Recommended Reading

1. Knuth DE (1997) The art of computer programming, vol 2: seminumerical algorithms, 3rd edn. Addison-Wesley, Reading
2. Menezes AJ, van Oorschot PC, Vanstone SA (1997) Handbook of applied cryptography. CRC Press, Boca Raton

Multiset Attack

Alex Biryukov
FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg

Related Concepts

►Block Ciphers; ►Structural Cryptanalysis

Definition

Multiset attack is a generic class of attacks which covers several recently designed (typically ►chosen plaintext attacks), which appeared in the literature under three different names: the *Square attack* [1], *the saturation attack* [4], the *integral cryptanalysis* [3].

Background

The first such attack was discovered by Knudsen during analysis of the cipher Square [1] and was thus called "Square attack." A similar attack was used by Lucks [4] against the cipher ►Rijndael/AES and called "saturation" attack. Later Biryukov and Shamir have shown an attack of similar type breaking arbitrary three round SPN (refer also ►Substitution–Permutation (SP) Network) with secret components (the so-called *SASAS* scheme, which consists of five layers of substitutions and affine transforms). Gilbert–Minier's "collision" attack [2] on 7-rounds of Rijndael as well as Knudsen–Wagner's [3] "integral" cryptanalysis of 5-rounds of ►MISTY1 (see also [5]) fall into the same class.

Theory

The main feature behind these attacks is that unlike a differential attack in which the attacker studies the behavior of pairs of encryptions, in a multiset attack, the attacker looks at a larger, carefully chosen set of encryptions, in which parts of the input text forms a multiset. A multiset is different from a regular notion of a set, since it allows the same element to appear multiple times. The element of a multiset is thus a pair (*value, multiplicity*), where *value* is the value of the element and *multiplicity* counts the number of times this value appears in the multiset. The attacker then studies the propagation of multisets through the cipher. The effect of the cipher on a multiset is in the changing of values of the elements but preserving some of the multiset properties like: multiplicity; or "integral" (i.e., sum of all the components); or causing a reduced set of values which would increase the probability of birthday-like events inside the cipher.

Multiset attacks are among the best attacks on AES-128 (►Rijndael/AES) due to its byte-wise structure. This type of attacks is a promising direction for future research.

Recommended Reading

1. Daemaen J, Knudsen LR, Rijmen V (1997) The block cipher Squar. In: Biham E (ed) Proceedings of fast software encryption – FSE'97. Lecture notes in computer science, vol 1267. Springer, Berlin, pp 149–165
2. Gilbert H, Minier M (2000) A collision attack on seven rounds of Rijindael. In: Proceedings of the third AES candidate conference, pp 230–241
3. Knudsen LR, Wagner D (2002) Integral cryptanalysis (extended abstract). In: Daemen J, Rijmen V (eds) Fast software encryption, FSE 2002. Lecture notes in computer science, vol 2365. Springer, Berlin, pp 112–127
4. Lucks S (2000) Attacking seven rounds of Rijndael under 192-bit and 256-bit keys. In: Proceedings of the third AES candidate conference, pp 215–229
5. Sun X, Lai X (2009) Improved Integral Attacks on MISTY1. In: Selected areas in cryptography. Lecture notes in computer science, vol 5867. Springer, Berlin, pp 266–280

Multi-Threaded Implementation for Cryptography and Cryptanalysis

Chen-Mou Cheng
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

Related Concepts

►Exhaustive Key Search; ►FPGA in Cryptography; ►Moore's Law; ►Special-Purpose Cryptanalytical Hardware

Definition

Multi-threaded implementation for cryptography and cryptanalysis is a way to implement cryptographic and cryptanalytic algorithms by mapping the inherent or carefully constructed parallelism in the algorithms onto multi-core or many-core computer architectures in order to exploit the immense computational power provided by modern multi-core or many-core computers.

Background

After decades of remarkable growth, the performance increase in single-threaded processors has slowed down in many areas of application. The most cited reasons include the *power wall*, the physical limit related to power and heat dissipation that prevents clock frequency from increasing indefinitely, and the *ILP (instruction-level parallelism) wall*, the increasing difficulty and cost to find and exploit instruction-level parallelism in general workload. To respond, the semiconductor industry is moving to the *multi-core* or *many-core* paradigm, a more scalable way of taking advantage of the exponentially growing transistor budget offered by the ►Moore's law in semiconductor fabrication technology. It is hence envisioned that in the future, all processors will be multi-core or many-core, all computers will be [massively] parallel, and all programs will be parallel programs. This posts challenges and new opportunities to implementers of cryptographic and cryptanalytic algorithms to develop software that can effectively leverage the increasing number of processor cores, as well as software that can scale with future generations of multi-core or many-core computers.

Theory

As in any parallel programming, a key to efficient multi-threaded implementation for cryptography and cryptanalysis is to find concurrency in the computation [11]. In most cryptanalytic and some cryptographic algorithms, there is inherent concurrency that the implementer can easily exploit, but in the cases where there is little parallelism in the algorithm to implement, one might need to artificially introduce concurrency to the system. This is typically done via batching, that is, grouping several units of data before processing them together, which brings parallelism in a natural way. However, batching is not always a viable option for all kinds of applications, especially when there are delay constraints such as in providing confidentiality to real-time communication.

With exploitable concurrency in the target algorithm, the actual structure of a particular implementation is then largely determined by the architecture of the parallel computer on which the implementation runs. For example, one should adopt different implementation strategies for SIMD (single instruction, multiple data) and MIMD (multiple instruction, multiple data) machines. For an MIMD machine, the choice of implementation strategies is further complicated by the memory model of the machine, which can be shared or distributed memory, or a hybrid between the two. For the shared memory model, a widely used parallel programming environment is OpenMP [5], whereas the counterpart for the distributed memory model is MPI (message-passing interface) [8].

Graphics processing units (GPUs) represent another class of many-core architectures that are cost-effective for achieving high arithmetic throughput. The success of GPUs has mainly been driven by the economy of scale in the video-game industry. Currently, the most widely used GPU development toolchain is NVIDIA's CUDA [10, 13]. At the core of CUDA are three key abstractions, namely, a hierarchy of thread groups, shared memories, and barrier synchronization, that are exposed to the programmers as a set of extensions to the C programming language. At the system level, the GPU is used as a coprocessor to the host processor for massively data-parallel computations, each of which is executed by a grid of GPU threads that must run the same program (the *kernel*). This is the SPMD (single program, multiple data) programming model, similar to SIMD but with more flexibility such as in changing of data size on a per-kernel-launch basis, or deviation from SIMD to MIMD at a performance penalty.

It should be noted that in multi-threaded implementation for cryptography, one needs to use caution to provision for defenses against ►side-channel attacks such as ►fault attacks, ►timing attacks, (differential) power analysis, and ►electromagnetic attacks.

Applications

There have been an array of efficient multi-threaded implementations of cryptographic and cryptanalytic algorithms.

On cryptography side, there have been implementations of ►symmetric cryptosystems [6] as well as ►asymmetric cryptosystems [12, 14]. On cryptanalysis side, people have attacked the problems of ►integer factoring [3, 4], the ►discrete logarithm problem [2], and ►lattice reduction [1, 7, 9].

Open Problems

Implementing complex cryptographic and cryptanalytic algorithms on today's multi-threaded computers is painfully labor-intensive. It often involves a lot of hand-tuning of code in low-level, C or C-like programming languages such as OpenMP, MPI, CUDA, and OpenCL. This development overhead can be reduced by use of better design-automation tools such as domain-specific programming languages and software libraries, as well as optimizing compilers that automatically generate efficient code for various multi-threaded hardware platforms. However, generating efficient multi-threaded code from high-level algorithmic description is still an open problem in general.

Recommended Reading

1. Backes W, Wetzel S (2009) Parallel lattice basis reduction using a multithreaded Schnorr-Euchner LLL algorithm. In: Proceedings of the 15th International European Conference on Parallel and Distributed Computing (Euro-Par), Delft, pp 960–973
2. Bailey DV, Baldwin B, Batina L, Bernstein DJ, Birkner P, Bos JW, van Damme G, de Meulenaer G, Fan J, Guneysu J, Gurkaynak F, Kleinjung T, Lange T, Mentens N, Paar C, Regazzoni F, Schwabe P, Uhsadel L (2009) The Certicom challenges ECC2-X, Special-purpose hardware for attacking cryptographic systems (SHARCS'09). Lausanne, Switzerland
3. Bernstein DJ, Chen HC, Chen MS, Cheng CM, Hsiao CH, Lange T, Lin ZC, Yang BY (2009) The billion-mulmod-per-second PC, Special-purpose Hardware for Attacking Cryptographic Systems (SHARCS'09). Lausanne, Switzerland
4. Bernstein DJ, Chen TR, Cheng CM, Lange T, Yang BY (2009) ECM on graphics cards. In: Proceedings of the 28th annual international conference on advances in cryptology: the theory and applications of cryptographic techniques (EUROCRYPT), Cologne, Germany, pp 483–501
5. Chandra R, Menon R, Dagum L, Kohr D, Maydan D, McDonald J (2000) Parallel programming in OpenMP. Morgan Kaufmann, San Francisco
6. Cook D, Keromytis A (2006) CryptoGraphics: exploiting graphics cards for security. Springer, New York
7. Dagdelen O (2009) Parallelization of lattice basis reduction. Diploma thesis, Technische Universität Darmstadt
8. Gropp W, Lusk E, Skjellum A (1994) Using MPI: Portable parallel programming with the message-passing interface. MIT Press, Cambridge
9. Hermans J, Schneider M, Buchmann J, Vercauteren F, Preneel B (2009) Shortest lattice vector enumeration on graphics cards, special-purpose hardware for attacking cryptographic systems (SHARCS'09). Lausanne, Switzerland
10. Kirk DB, Hwu WM (2010) Programming massively parallel processors: a hands-on approach. Morgan Kaufmann, San Francisco
11. Mattson TG, Sanders BA, Massingill BL (2004) Patterns for parallel programming. Addison-Wesley Professional, Reading
12. Moss A, Page D, Smart NP (2007) Toward acceleration of RSA using 3D graphics hardware. In: Proceedings of the 11th IMA International Conference on Cryptography and Coding, Cirencester, pp 364–383
13. NVIDIA CUDA programming guide, version 2.3.1, August 2009
14. Szerwinski R, Güneysu T (2008) Exploiting the power of GPUs for asymmetric cryptography. In: Proceedings of 10th International Workshop on Cryptographic Hardware and Embedded Systems (CHES), Washington, pp 79–99

Multivariate Cryptography

Louis Goubin[1], Jacques Patarin[1], Bo-Yin Yang[2]
[1]Versailles St-Quentin-en-Yvelines University, France
[2]Research Fellow, Nankang, Taipei, Taiwan

Synonyms

MPKC; $\mathcal{MQ}$; Multivariate quadratic public-key Cryptosystem (MQPKC)

Related Concepts

►Differential Cryptanalysis; ►Digital Signature Schemes; ►Post-quantum Cryptography; ►Public-key Cryptography

Definition

A *Multivariate Public-Key Cryptosystem* (MPKC) is a ►public-key cryptosystem where the public map $\mathcal{P}$, or ►trapdoor one-way function, is given as a set of m polynomial equations of a small degree d over n variables in a ►finite field F. Usually $d = 2$, hence the alternate name "Multivariate Quadratic" (MQ).

To decrypt, authenticate, or sign digitally, a user must, for a given m-tuple $\mathbf{z} = (z_1, \ldots, z_m)$, find a solution $\mathbf{w} = (w_1, \ldots, w_n)$ of the system

$$(\mathcal{P}) \begin{cases} p_1(w_1, \ldots, w_n) = z_1 \\ \cdots \\ p_m(w_1, \ldots, w_n) = z_m \end{cases}.$$

For a *digital signature*, a challenge–response *authentication* scheme, and an *encryption* scheme, $\mathbf{z} = (z_1, \ldots, z_m)$

is respectively the hash of the message to be signed, the challenge, and the ciphertext, while $\mathbf{w} = (w_1, \ldots, w_n)$ is respectively the signature of this message, the response, and the cleartext.

Anyone can easily verify whether a given pair $(\mathbf{w}, \mathbf{z})$ satisfies the system $(\mathcal{P})$, or compute $\mathbf{z} = \mathcal{P}(\mathbf{w})$ from any given $\mathbf{w}$. However, finding at least one solution $\mathbf{w} = (w_1, \ldots, w_n)$ of the system $\mathcal{P}$ should be difficult for most $\mathbf{z} = (z_1, \ldots, z_m)$ without the knowledge of a secret key, but easy with the knowledge of a secret key. In other words, without the knowledge of a secret key, it should be infeasible to decrypt a message, forge a signature successfully, or pass an authentication.

Background

In general, solving a set of quadratic equations over a finite field is NP-hard (►Computational Complexity) for any finite field. This problem, known as $\mathcal{MQ}$, is even conjectured to be probabilistically hard, i.e., as $m, n \to \infty$, $\forall \varepsilon > 0$, and for any probabilistic Turing Machine $\mathcal{A}$, $\Pr(\mathcal{P}(x) = y$ be solved by $\mathcal{A}$ in $\text{poly}(m, n)) < \varepsilon$.

However, it is difficult to obtain a proof of security for multivariate schemes: since the system $(\mathcal{Q})$ has to be easy to solve, it is never random, and neither is $(\mathcal{P})$ which is obtained by linear changes of variables from $(\mathcal{Q})$.

One can always try to solve the system of equations directly using system solvers based on ►Gröbner bases (XL, $\mathbf{F_4}$, $\mathbf{F_5}$). For more or less "generic" or "random" equations, most current methods take ►exponential time when m and n are of the same size. Some systems appear very nonrandom under Gröbner basis methods. Recently differential analysis techniques were applied to MPKCs resulting in the break of the SFLASH system [6, 12, 23].

The first proposals of cryptosystems based on $\mathcal{MQ}$ date back to the early 1980s, see Imai et al. [18], Matsumoto and Imai [19], Tsujii et al. [24], Matsumoto and Imai [20].

Shor's algorithm cannot be used to speed up solving $\mathcal{MQ}$, so unlike the RSA cryptosystem, ►elliptic curve cryptography and the ►digital signature standard which are known to be breakable by quantum computers, MPKCs are candidates for ►post-quantum cryptography.

There are variants in which more general systems of polynomials are used with the $\mathbf{w}$ and $\mathbf{z}$ variables mixed ("implicit MPKC"), or where not all the equations of the system $(\mathcal{P})$ have to be valid, but only a given fixed percentage ("probabilistic MPKC"). Some authors also include in the category of multivariate cryptography other schemes like the zero-knowledge schemes IP [22] or MinRank [4]. These variants are not considered in this entry.

Theory

Typically, a multivariate scheme is built from an easy-to-solve system $\mathcal{Q}(\mathbf{x}) = \mathbf{y}$, the *central map*, which is then "hidden" by two secret random linear (or affine) transformations $S : \mathbf{w} \mapsto \mathbf{x}$ and $T : \mathbf{y} \mapsto \mathbf{z}$ to obtain a new system $(\mathcal{P})$, by composing them (on the left and on the right) with $\mathcal{Q}$. The system $(\mathcal{P})$ is the public key, the original system $(\mathcal{Q})$ and the transformations S and T form the secret key.

More precisely, if $(\mathcal{Q})$ is given by m polynomials $(q_1, \ldots, q_m)$ in $(x_1, \ldots, x_n)$, the new system $(\mathcal{P})$ is given by m polynomials $(p_1, \ldots, p_m)$ in $(w_1, \ldots, w_n)$ as in:

$$(p_1, \ldots, p_m)(w_1, \ldots, w_n) = T(q_1(S(w_1, \ldots, w_n)), \ldots, q_m(S(w_1, \ldots, w_n)))$$

where S and T are secret random linear (or affine) bijective changes of variables.

The new system $\mathcal{P} := T \circ \mathcal{Q} \circ S$ is also quadratic and finding a solution of

$$(\mathcal{P}) \begin{cases} p_1(w_1, \ldots, w_n) = z_1 \\ \cdots \\ p_m(w_1, \ldots, w_n) = z_m \end{cases}$$

for a given m-tuple $(z_1, \ldots, z_m)$ is expected to be difficult without the knowledge of S and T. In the following, the system $(\mathcal{P})$ will also be denoted by $\mathcal{P}(\mathbf{w}) = \mathbf{z}$.

There exist several families of multivariate schemes, corresponding to several choices for the system $\mathcal{Q}$ (some examples are given below). Some schemes have been broken; for others, only generic Gröbner-basis attacks apply and the parameters are chosen large enough to make those infeasible.

Variants (Perturbations)

In order to increase the security of an MPKC or to repair a broken scheme, some variants or "perturbations" of the polynomials are often introduced, either on the system $(\mathcal{P})$ or on the system $(\mathcal{Q})$. The most classical variants are denoted $+$, $\oplus$, $-$, V, and F.

$+$: The public key of a "plus" variant is not given by the system of polynomials $(p_1(\mathbf{w}), \ldots, p_m(\mathbf{w}))$ but by $(p_1(\mathbf{w}) + L_1(R_1(\mathbf{w}), \ldots, R_\alpha(\mathbf{w})), \ldots, p_m(\mathbf{w}) + L_m(R_1(\mathbf{w}), \ldots, R_\alpha(\mathbf{w})), R_1(\mathbf{w}), \ldots, R_\alpha(\mathbf{w}))$, where $L_1, \ldots, L_m$ are secret linear forms and $R_1, \ldots, R_\alpha$ are α truly random secret quadratic polynomials in $(w_1, \ldots, w_n)$, with $\alpha \ll m$.

$\oplus$: The public key is not given by the system of polynomials $(p_1(\mathbf{w}), \ldots, p_m(\mathbf{w}))$ but by $(p_1(\mathbf{w}) + R_1(L_1(\mathbf{w}), \ldots, L_\alpha(\mathbf{w})), \ldots, p_m(\mathbf{w}) + R_m(L_1(\mathbf{w}), \ldots, L_\alpha(\mathbf{w})))$, where $R_1, \ldots, R_m$ are m truly random secret

quadratic polynomials, and $L_1, \ldots, L_\alpha$ are secret linear forms, with $\alpha \ll m$.

– : The "minus" variant consists in keeping secret some of the m polynomials $(p_1, \ldots, p_m)$. Only $m - \beta$ polynomials are made public and will be used to check the validity of the equations.

V : In the "vinegar" variant – also called "extra variables perturbation" – α new variables are introduced in the system $(\mathcal{Q})$, and secretly combined with the usual variables $(w_1, \ldots, w_n)$. This mixing is not always linear, but is such that the new obtained system is still quadratic.

F : In the "fix" variant, the public-key polynomials $(p'_1, \ldots, p'_m)$ are formed by replacing α of the variables in $(p_1(\mathbf{w}), \ldots, p_m(\mathbf{w}))$ by arbitrary linear combinations in $\mathbf{w} = (w_1, \ldots, w_n)$.

Applications

Among the most famous multivariate public-key schemes are C^* by Matsumoto and Imai [19, 20]; HFE (Hidden Field Equations) by Patarin [22]; UOV (Unbalanced Oil and Vinegar) by Kipnis, Patarin, and Goubin [16]; Rainbow/TTS by Ding and Schmidt [7] and modified by Ding et al. [10]; and IFS (Intermediate Field System) by Billet, Patarin, Seurin [2].

Many other schemes have been presented by various authors from around the world, but most of them have been broken. For a recent overview article see Ding and Yang [11].

C^* and SFLASH

C^* [20] was originally designed by Matsumoto and Imai in 1985. The idea is to create a quadratic system $(\mathcal{Q})$ from a monomial transformation $\mathbf{x} \mapsto \mathbf{y} = \mathbf{x}^{1+q^\theta}$, for some θ, over an extension $F = \mathrm{GF}(q^n)$ of the finite field $K = \mathrm{GF}(q)$, where the elements $\mathbf{x}$ and $\mathbf{y}$ of F correspond to the usual vectors over K by using an explicit basis (►Vector Space).

It was broken by Patarin in 1995 [21], using the following idea: if $\mathbf{y} = \mathbf{x}^{1+q^\theta}$ then $\mathbf{x}\mathbf{y}^{q^\theta} = \mathbf{x}^{q^{2\theta}}\mathbf{y}$. As $\mathbf{v} \mapsto \mathbf{v}^q$ is K-linear, regardless of S and T, the variables $\mathbf{v}$ and $\mathbf{w}$ satisfy a system of equations of the form $\sum \alpha_{ij} w_i z_j + \sum \beta_i w_i + \sum \gamma_j z_j + \delta = 0$. The coefficients can be determined by evaluating the original system at many random $\mathbf{w}$s and inserting the resulting pairs $(\mathbf{w}, \mathbf{z})$. Finally, for a target $\mathbf{z}$ this linear system can be solved for $\mathbf{w}$ by Gaussian elimination.

Many ways to repair the original scheme have been suggested. Most famous among these is the SFLASH scheme [4], which is a $(C^*)^-$ instance (i.e., a "minus" variant of the C^* scheme). It has been broken in 2007 by Dubois, Fouque, Shamir, and Stern [12]. Some variants of C^* are still under investigation, for example in characteristic 3 instead of characteristic 2.

HFE and QUARTZ

Instead of using a monomial over an extension $F = \mathrm{GF}(q^n)$ of the finite field $\mathrm{GF}(q)$, HFE [22] uses a *polynomial* transformation

$$F \to F; \mathbf{x} \mapsto \mathbf{y} = \sum_{0 \le i \le j < r} a_{ij} \mathbf{x}^{q^i + q^j} + \sum_{0 \le i < r} b_i \mathbf{x}^{q^i} + c.$$

and builds $(\mathcal{Q})$ by using an explicit basis of F over $\mathrm{GF}(q)$. Operations involving the secret key work in F while the public key is given over $\mathrm{GF}(q)$. By construction $(\mathcal{Q})$ is quadratic in $\mathbf{x}$. Finding $\mathbf{w}$ given $\mathbf{z}$ corresponds to finding roots of the polynomial e.g., using Berlekamp's Algorithm (►Berlekamp Q Matrix).

When the maximum degree D is fixed, ►polynomial time attacks exist, as mentioned in the original paper. When D increases with n, no polynomial attack is known. Nevertheless, subexponential and superpolynomial-time attacks have been found by Faugère and Joux: The basic HFE central map has (in some sense) a rank which is decided by D, which in turn bounds the operating degree in a direct algebraic attack if D is too small. Therefore, it is recommended to use HFE with some perturbations. For example, only exponential attacks are known against HFE$^-$. QUARTZ [5] is an HFE^{V-} scheme with $q = 2$, $d = 129$, $n = 103$, $r = 8$, $v = 3$ and the central polynomial is of the form below with four polynomials removed from the system:

$$\sum_{\substack{0 \le i,j < r \\ q^i + q^j \le d}} a_{ij} \mathbf{x}^{q^i + q^j} + \sum_{\substack{0 \le i,j < r \\ q^i + q^j \le d}} b_{ij} \mathbf{x}^{q^i} \overline{\mathbf{x}}^{q^j} + \sum_{\substack{0 \le i,j < r \\ q^i + q^j \le d}} \alpha_{ij} \overline{\mathbf{x}}^{q^i + q^j} + \sum_{i=0}^{r-1} b_i \mathbf{x}^{q^i} + \sum_{i=0}^{r-1} \beta'_i \overline{\mathbf{x}}^{q^i} + c,$$

where $\overline{\mathbf{x}}$ denotes the vinegar variables.

UOV

In UOV [16], the system $(\mathcal{Q})$ is given by m quadratic polynomials in $n = m + v$ variables $(w_1, \ldots, w_m, w_{m+1}, \ldots, w_{m+v})$. The first m variables $w_1, \ldots, w_m$ are called "oil variables" and the v other variables $w_{m+1}, \ldots, w_{m+v}$ are called "vinegar" variables. Each polynomial may contain quadratic terms of the form "oil×vinegar" (i.e., $w_i w_j$ with $1 \le i \le m$ and $m + 1 \le i \le m + v$) or "vinegar×vinegar" (i.e., $w_i w_j$ with $m + 1 \le i, j \le m + v$), but must not contain quadratic terms of the form "oil×oil" (i.e., $w_i w_j$ with $1 \le i, j \le m$). As a consequence, the system $(\mathcal{Q})$ is easy to solve, by fixing arbitrary values for the vinegar variables, and solving the

obtained system (in the m oil variables) by Gaussian elimination, however $(\mathcal{P})$ should hide the distinction between oil and vinegar variables. An early version of this scheme ("Oil and Vinegar," corresponding to the particular choice $v = m$ of the parameters) was broken by Kipnis and Shamir in 1998 [17]. However the "unbalanced" version (UOV) remains unbroken as long as $v > 2m$ (i.e., $n > 3m$).

Rainbow/TTS

The idea of the Rainbow scheme [7, 10] is to use u UOV instances, in an iterative way. The first UOV instance contains $v_2 - v_1$ polynomials in $v_2 - v_1$ oil variables and v_1 vinegar variables. The second UOV instance contains $v_3 - v_2$ polynomials in $v_3 - v_2$ oil variables and v_2 vinegar variables. The last UOV instance contains $v_{u+1} - v_u$ polynomials in $v_{u+1} - v_u$ oil variables and v_u vinegar variables. Denoting v_{u+1} by n, the obtained system $(\mathcal{Q})$ thus contains $m = n - v_1$ polynomials in n variables, and is easily solved by recursively applying the UOV principle: fix arbitrarily the v_1 vinegar variables of the first UOV instance, solve this instance in the $v_2 - v_1$ oil variables, then consider all these $v_1 + (v_2 - v_1) = v_2$ variables as the vinegar variables of the second UOV instance, and so on. This means that the legitimate signer has to solve more but smaller systems compared to UOV. For carefully chosen parameters, the Rainbow scheme remains unbroken. TTS [25], a predecessor of Rainbow, can be considered an aggressive variant of the latter with sparse coefficients. This makes it faster in exchange of opening more possible pitfalls.

"Medium-Sized" Extension Fields and IFS

Multivariate systems can also be constructed over medium-sized extension fields, e.g., by constructing $(\mathcal{Q})$ using a system $(\mathcal{Q}')$ of k polynomial equations in k variables over an algebraic extension field of F. This system $(\mathcal{Q}')$ is chosen to be solvable using tailored Gröbner-basis algorithms, and such that, when rewritten in terms of variables $\mathbf{x}$ using a basis over the "small" field F, it is quadratic. The system just described is the "Intermediate Field System" (IFS, [2]) which remains unbroken for carefully chosen parameters. Other attempts to use several variables in an extension field include [8, 9].

Implementations

Multivariate digital signature schemes can have very short signatures as in QUARTZ [4], which allows approximately 100-bit-long signatures.

Multivariate cryptography is in general quite fast both for the public and private maps. For instance, Rainbow/TTS are among the fastest digital signature schemes and can be implemented cheaply on ASICs.

The eBACS benchmarking framework [1] has the following performance data for multivariate systems benchmarked on `hydra2`, an Intel Xeon E5620, running at 2,400 MHz:

System	Signature in cycles	Verification in cycles	Key pair in cycles	Secret key in bytes	Public key in bytes
3icp	1,392,924	34,724	10,416,528	12,768	35,712
pflash1	1,633,668	337,476	78,627,188	5,550	72,124
rainbow 6440	138,284	53,220	193,386,868	150,512	57,600
rainbowbinary 16242020	38,948	26,128	62,545,576	94,384	102,912
tts6440	47,236	56,452	38,462,532	16,608	57,600

The messages considered are short (59 bytes long).

Another advantage of multivariate schemes is their flexibility in the design of various schemes, with ad hoc properties.

Open Problems

The main drawbacks of MPKCs are large keys and the uncertainty about their security. It is difficult to obtain (relative) security proofs as in schemes based on factorization of discrete logarithm problem. The confidence in a given scheme relies on its resistance to all known attacks that have been developed against multivariate systems. Most needed are some provable security results; also useful are management of large keys and continuing optimizations on current hardware.

Recommended Reading

1. Bernstein DJ, Lange T (eds) (2011) eBACS: ECRYPT benchmarking of cryptographic systems. http://bench.cr.yp.to. Accessed 10 June 2011
2. Billet O, Patarin J, Seurin Y (2008) Analysis of intermediate field systems. In: Proceedings of SCC 2008, Beijing
3. Chen AI-T, Chen M-S, Chen T-R, Cheng C-M, Ding J, Kuo EL-H, Lee FY-S, Yang B-Y (2009) SSE Implementation of multivariate PKCs on modern x86 CPUs. In: Proceedings of CHES 2009. Lecture notes in computer science, vol 5747, pp 33–48
4. Courtois NT (2001) Efficient zero-knowledge authentication based on a linear algebra problem MinRank. In: Proceedings of ASIACRYPT 2001. Lecture notes in computer science, vol 2248. Springer, pp 402–421
5. Courtois NT, Goubin L, Patarin J (2001) QUARTZ, 128-bit long digital signatures. In: Proceedings of CT-RSA 2011. Lecture notes in computer science, vol 2020. Springer, pp 282–297
6. Ding J, Dubois V, Yang B-Y, Chen C-H, Cheng C-M (2008) Can SFLASH be repaired. In: Proceedings of ICALP 2008. Lecture notes in computer science, vol 5126. Springer, pp 691–701
7. Ding J, Schmidt D (2005) Rainbow, a new multivariable polynomial signature scheme. In: Proceedings of ACNS 2005. Lecture notes in computer science, vol 3531. Springer, pp 164–175
8. Ding J, Werner F, Yang B-Y, Chen C-H, Chen M-S (2008) Odd-char multivariate hidden field equations, Cryptology eprint archive report 2008/543 version 20081229:161921

9. Ding J, Wolf C, Yang B-Y (2007) ℓ-Invertible Cycles for $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic ($\mathcal{MQ}$) Public Key Cryptography. In: Proceedings of PKC 2007. Lecture notes in computer science, vol 4450, pp 266–281
10. Ding J, Yang B-Y, Chen C-H, Chen M-S, Cheng C-M (2008) New differential-algebraic attacks and reparametrization of rainbow. In: Proceedings of ACNS 2008. Lecture notes in computer science, vol 5037, pp 242–257
11. Ding J, Yang B-Y (2009) Multivariate public-key cryptography. In: Bernstein DJ, Buchmann J, Dahmen E (eds) Post-quantum cryptography. Springer, ISBN: 978-3-540-88701-0, e-ISBN: 978-3-540-88702-7
12. Dubois V, Fouque P-A, Shamir A, Stern J (2007) Practical cryptanalysis of SFLASH. In: Proceedings of Crypto 2007. Lecture notes in computer science, vol 4622, pp 1–12
13. Faugère J-C, Joux A (2003) Algebraic cryptanalysis of hidden field equation (HFE) Cryptosystems using Gröbner bases. In: Proceedings of Crypto 2003. Lecture notes in computer science, vol 2729, pp 44–60
14. Faugère J-C, Perret L (2006) Polynomial equivalence problems – algorithmic and theoretical aspects. In: Proceedings of Eurocrypt 2006. Lecture notes in computer science, vol 4004. Springer, pp 30–47
15. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W.H. Freeman and Company, New York
16. Kipnis A, Patarin J, Goubin L (1999) Unbalanced oil and vinegar signature schemes. In: Proceedings of Eurocrypt'99. Lecture notes in computer science, vol 1592. Springer, pp 206–222
17. Kipnis A, Shamir A (1998) Cryptanalysis of the oil and vinegar signature scheme. In: Proceedings of CRYPTO'98. Lecture notes in computer science, vol 1462, pp 257–266
18. Matsumoto T, Imai H, Harashima H, Miyakawa H (1983) A class of asymmetric cryptosystems using obscure representations of enciphering functions. In: Proceedings of the 1983 national convention record on information systems, IECE Japan
19. Matsumoto M, Imai H (1986) Algebraic methods for constructing asymmetric cryptosystems. In: Proceedings of the 3rd international conference on Algebraic Algorithms and Error-Correcting Codes (AAECC-3), Grenoble, France, 15-19 July 1985. Lecture notes in computer science, vol 229. Springer, pp 108–119
20. Matsumoto M, Imai H (1988) Public quadratic polynomial-tuples for efficient signature verification and message-encryption. In: Proceedings of Eurocrypt'88. Lecture notes in computer science, vol 330. Springer, pp 419–545
21. Patarin J (1995) Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88. In: Proceedings of CRYPTO'95. Lecture notes in computer science, vol 963. Springer, pp 248–261
22. Patarin J (1996) Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In: Proceedings of Eurocrypt'96. Lecture notes in computer science, vol 1070. Springer, pp 33–48
23. Patarin J, Courtois N, Goubin L (2001) FLASH, a Fast Multivariate Signature Algorithm. In: Proceedings of the conference on topics in cryptology: the cryptographer's track at RSA. Lecture notes in computer science, vol 2020, Springer, pp 298–307
24. Tsujii S, Itoh T, Fujioka A, Kurosawa K, Matsumoto T (1988) A public-key cryptosystem based on the difficulty of solving a system of nonlinear equations. Syst Comput Jpn 19:10–18
25. Yang B-Y, Chen J-M, Chen Y-H (2004) TTS: high-speed signatures on a low-cost smart card. In: Proceedings of CHES 2004. Lecture notes in computer science, vol 3156. Springer, pp 371–385