L

ℓ-Diversity

Johannes Gehrke[1], Daniel Kifer[2], Ashwin Machanavajjhala[3]
[1] Department of Computer Science, Cornell University, Ithaca, NY, USA
[2] Department of Computer Science and Engineering, Penn State University, University Park, PA, USA
[3] Yahoo! Research, Santa Clara, CA, USA

Related Concepts

►DeFinetti Attack; ►k-Anonymity; ►Statistical Disclosure Limitation

Definition

ℓ-diversity is a method for publishing data about individuals while limiting the amount of sensitive information disclosed about them.

Background

Many organizations are increasingly publishing microdata – tables that contain information about individuals that is not aggregated in any way. Examples include medical, voter registration, census, and customer data. Microdata is a valuable source of information for subsequent data analysis – medical research, the allocation of public funds, or trend analysis, just to name a few. However, if individuals can be uniquely identified in the microdata, then their private information is disclosed, and this is unacceptable.

There is a long history of research on limiting disclosure in data publishing, starting with work by Stanley Warner on randomized response [9]. Much progress has been made over the following decades, but the area is still full of new developments; these two surveys show some of the recent progress and excitement in this area [1, 3].

Consider the database of patients' data shown in Fig. 1. There is a set of attributes (like {Zip Code, Age, Nationality, gender, and date of birth} in the example) that can be linked with external data to uniquely identify individuals in the population; these are called *quasi-identifiers*. To counter linking attacks using quasi-identifiers, Samarati and Sweeney proposed ►*k-anonymity* [6–8]. A table satisfies k-anonymity if every record in the table is indistinguishable from at least $k-1$ other records for the values of the quasi-identifier attributes; such a table is called a *k-anonymous* table. For every combination of values of the quasi-identifiers in the k-anonymous table, there are at least k records that share those values. This ensures that individuals cannot be uniquely matched to an individual record in the published microdata by linking attacks.

Figure 2 shows a 4-anonymous table derived from the table in Fig. 1 (here "*" denotes a suppressed value so, for example, "zip code = 1485*" means that the zip code is in the range $[14850 - 14859]$ and "age=3*" means the age is in the range $[30 - 39]$). Note that in the 4-anonymous table, each tuple has the same values for the quasi-identifier as at least three other tuples in the table.

K-anonymity was designed to protect against associating respondents' identities with released tuples without specifically addressing the association of sensitive information. First, when there is no diversity in the values of the sensitive attributes, an attacker can easily discover the sensitive value of an individual through a *homogeneity attack*. For example, an attacker who knows that her 32-year-old neighbor's record is in Fig. 2 and who knows that the neighbor lives in zip code 13053 can conclude from the table that the neighbor has cancer. Second, attackers have background knowledge, and k-anonymity does not limit disclosure against such background knowledge attackers. For example, an attacker who knows that her 22-year-old Japanese friend lives in zip code 13068 can, with near certainty, conclude that the friend has a viral infection since it is well known that Japanese have a very low rate of heart disease. ℓ-Diversity was developed to address these two issues [5].

Theory

Let $T = \{t_1, t_2, \ldots, t_n\}$ be a table with tuples that form a subset of some larger population Ω. Without loss of generality, we may combine all of the quasi-identifier attributes into one single attribute. Thus in this discussion, each tuple has two attributes: a *quasi-identifier* and a *sensitive attribute*. The value of the quasi-identifier can often be linked with external data to uniquely associate tuples with

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

	Nonsensitive			Sensitive
	Zip code	Age	Nationality	Condition
1	13053	28	Russian	Heart disease
2	13068	29	American	Heart disease
3	13068	21	Japanese	Viral infection
4	13053	23	American	Viral infection
5	14853	50	Indian	Cancer
6	14853	55	Russian	Heart disease
7	14850	47	American	Viral infection
8	14850	49	American	Viral infection
9	13053	31	American	Cancer
10	13053	37	Indian	Cancer
11	13068	36	Japanese	Cancer
12	13068	35	American	Cancer

ℓ-Diversity. Fig. 1 Patient microdata

	Nonsensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	130**	< 30	*	Heart disease
2	130**	< 30	*	Heart disease
3	130**	< 30	*	Viral infection
4	130**	< 30	*	Viral infection
5	1485*	≥ 40	*	Cancer
6	1485*	≥ 40	*	Heart disease
7	1485*	≥ 40	*	Viral infection
8	1485*	≥ 40	*	Viral infection
9	130**	3*	*	Cancer
10	130**	3*	*	Cancer
11	130**	3*	*	Cancer
12	130**	3*	*	Cancer

ℓ-Diversity. Fig. 2 **4-anonymous** patient microdata

individuals in the population Ω. The value of the sensitive attribute contains information about the individual only known to the data publisher. The goal of privacy-preserving data publishing is to publish as much statistical information about T as possible while limiting the amount of disclosure about the association of the sensitive attribute with individuals.

In ℓ-diversity, a generalization $T^\star$ of T is published and is defined as follows. A domain $D^\star = \{P_1, P_2, \dots\}$ is a *generalization* (partition) of a domain D if $\bigcup P_i = D$ and $P_i \cap P_j = \varnothing$ for $i \neq j$. For $q \in D$, let $\phi_{D^\star}(q)$ denote the element $P \in D^\star$ that contains q. Given a table $T = \{t_1, \dots, t_n\}$ with a quasi-identifier Q and a generalization $D_N^\star$ of the domain D of Q, a table $T^\star = \{t_1^\star, \dots, t_n^\star\}$ can now be constructed by replacing q_i, the value of the quasi-identifier of t_i, with its generalized value $\phi_{D^\star}(q_i)$.

It is not possible for a data publisher to guard against attacks employing arbitrary amounts of background knowledge [2]. The goal of microdata publishing is to allow a data publisher to guard against many reasonable attacks without having access to the attacker's background knowledge. It seems that background knowledge such as "men do not have breast cancer" or "Japanese have a very low incidence of heart disease" is very powerful as it enables the adversary to eliminate sensitive values from a given data-block in the generalized table.

However, if there are ℓ "well represented" sensitive values in a $q^\star$-block (i.e., the set of tuples whose quasi-identifier has been generalized to $q^\star$), then the attacker needs $\ell - 1$ damaging pieces of background knowledge to eliminate $\ell - 1$ possible sensitive values and infer a positive disclosure. Thus, by setting the parameter ℓ, the data publisher can determine how much protection is provided against background knowledge – even if this background knowledge is unknown to the publisher.

This intuition can be made formal through the ℓ-diversity principle as follows.

Principle 1 (ℓ-Diversity Principle [5]) *Let q be a value of the attribute Q in the base table T; let $q^\star$ be the generalized value of q in the published table $T^\star$. A $q^\star$-block is ℓ-diverse if *it* contains at least ℓ "well-represented" values for the sensitive attribute S. A table is ℓ-diverse if every $q^\star$-block (for all $q^\star$ that appear in $T^\star$) is ℓ-diverse.*

There are two instantiations of the ℓ-diversity principle that both result in diversity in the sensitive attribute.

Definition 1 (Entropy ℓ-Diversity) *A table is* Entropy ℓ-Diverse *if for every $q^\star$-block*

$$-\sum_{s \in S} p_{(q^\star, s)} \log(p_{(q^\star, s')}) \geq \log(\ell)$$

where $p_{(q^\star,s)} = \dfrac{n_{(q^\star,s)}}{\sum_{s' \in S} n_{(q^\star,s')}}$ *is the fraction of tuples in the $q^\star$-block with sensitive attribute value equal to s.*

Definition 2 (Recursive (c, ℓ)-Diversity) *In a given $q^\star$-block, let r_i denote the number of times the i^{th} most frequent sensitive value appears in that $q^\star$-block. Given a constant c, the $q^\star$-block satisfies* recursive (c, ℓ)-diversity *if $r_1 < c(r_\ell + r_{\ell+1} + \cdots + r_m)$. A table $T^\star$ satisfies recursive (c, ℓ)-diversity if every $q^\star$-block satisfies recursive ℓ-diversity; 1-diversity is always satisfied.*

ℓ-Diversity has several advantages. It does not require the data publisher to have as much information as the adversary. The parameter ℓ protects against more knowledgeable adversaries; the larger the value of ℓ, the more

	Nonsensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	1305*	≤ 40	*	Heart disease
4	1305*	≤ 40	*	Viral infection
9	1305*	≤ 40	*	Cancer
10	1305*	≤ 40	*	Cancer
5	1485*	> 40	*	Cancer
6	1485*	> 40	*	Heart disease
7	1485*	> 40	*	Viral infection
8	1485*	> 40	*	Viral infection
2	1306*	≤ 40	*	Heart disease
3	1306*	≤ 40	*	Viral infection
11	1306*	≤ 40	*	Cancer
12	1306*	≤ 40	*	Cancer

ℓ-Diversity. Fig. 3 3-Diverse inpatient microdata

information is needed to rule out possible values of the sensitive attribute. Instance-level knowledge (such as "my neighbor does not have diabetes") is also automatically covered. It is treated as just another way of ruling out possible values of the sensitive attribute. Different adversaries can have different background knowledge leading to different inferences. ℓ-Diversity simultaneously protects against all of them without the need for checking which inferences can be made with which levels of background knowledge (Fig. 3).

Open Problems

Daniel Kifer has recently shown that a statistical model more sophisticated than the one used by ℓ-diversity can be used to improve estimates of the probability of a tuple being associated with a sensitive value. This model takes advantage of information that q^* groups provide about each other [4]. Thus the exact level of protection that ℓ-diversity provides in general is an open problem.

It is also known that the largest value that entropy can take depends on the size of the domain (of the sensitive attribute). Thus when the domain is large, it may be necessary to use thresholds larger than $\log(\ell)$ when applying Entropy ℓ-diversity.

Recommended Reading

1. Chen B-C, Kifer D, LeFevre K, Machanavajjhala A (2009) Privacy-preserving data publishing. Found Trend Databases 2(1–2):1–167
2. Dwork C (2006) Differential privacy. In: 33rd international colloquium on automata, languages and programming (ICALP). Springer, Berlin, pp 1–12
3. Fung BCM, Wang K, Chen R, Yu PS (2010) Privacy-preserving data publishing: a survey of recent developments. ACM Comput Surv 42(14):1–53
4. Kifer D (2009) Attacks on privacy and definetti's theorem. In: SIGMOD. ACM, New York, pp 127–138
5. Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M (2007) L-diversity: privacy beyond k-anonymity. TKDD, 1(1)
6. Samarati P (2001) Protecting respondents' identities in microdata release. TKDE, 13(6):1010–1027
7. Samarati P, Sweeney L (1998) Generalizing data to provide anonymity when disclosing information. In: PODS. ACM, New York, p 188
8. Sweeney L (2002) Achieving k-anonymity privacy protection using generalization and suppression. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10(5):571–588
9. Warner SL (1965) Randomized response: a survey technique for eliminating evasive answer bias. J Amer Stat Assoc 60(309):63–69

L Notation

Arjen K. Lenstra
Laboratory for cryptologic algorithms - LACAL, School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne, Switzerland

Related Concepts

►Exponential Time; ►*O*-Notation; ►Polynomial Time; ►Subexponential Time

Definition

For $t, \gamma \in \mathbf{R}$ with $1 \le t \le 1$, the notation $L_x[t, \gamma]$ is used for any function of x that equals

$$e^{(\gamma+o(1))(\log x)^t(\log\log x)^{1-t}}, \text{ for } x \to \infty,$$

where logarithms are natural and where $o(1)$ denotes any function of x that goes to 0 as $x \to \infty$ (►*O notation*).

Theory

This function has the following properties:

- $L_x[t, \gamma] + L_x[t, \delta] = L_x[t, \max(\gamma, \delta)]$
- $L_x[t, \gamma] \cdot L_x[t, \delta] = L_x[t, \gamma + \delta]$
- $L_x[t, \gamma] \cdot L_x[s, \delta] = L_x[t, \gamma]$ if $t > s$
- For any fixed k:
  - $L_x[t, \gamma]^k = L_x[t, k\gamma]$
  - If $\gamma > 0$ then $(\log x)^k L_x[t, \gamma] = L_x[t, \gamma]$
- $\pi(L_x[t, \gamma]) = L_x[t, \gamma]$ where $\pi(y)$ is the number of primes $\le y$

When used to indicate runtimes and for γ fixed, $L_x[t, \gamma]$ for t ranging from 0 to 1 ranges from ►*polynomial time* to ►*exponential time* in $\log(x)$:

- Runtime
$$L_x[0,\gamma] = e^{(\gamma+o(1))\log\log x} = (\log x)^{\gamma+o(1)}$$
is polynomial in $\log(x)$.
- Runtimes $L_x[t,\gamma]$ with $0 < t < 1$ are examples of runtimes that are ►*subexponential time* in $\log(x)$, i.e., asymptotically greater than polynomial and less than exponential.
- Runtime
$$L_x[1,\gamma] = e^{(\gamma+o(1))\log x} = x^{\gamma+o(1)}$$
is exponential in $\log(x)$.

Lamport One-Time Signatures

►Hash-Based Signatures

Late Launch

►Dynamic Root of Trust

Lattice

Phong Nguyen
Département d'informatique, Ecole normale supérieure, Paris, Cedex 05, France

Synonyms

Euclidean lattice; Geometry of numbers

Related Concepts

►Closest Vector Problem; ►Lattice Reduction; ►Lattice-Based Cryptography; ►Shortest Vector Problem

Definition

In mathematics, the term *lattice* is used for two very different kinds of objects, arising, respectively, in order theory and number theory. Here, lattice always means a number-theoretical lattice. Informally speaking, a lattice is a regular infinite arrangement of points in n-dimensional space. More formally, a lattice is a discrete ►subgroup of $\mathbb{R}^n$.

Background

Lattices appeared in the nineteenth century in both crystallography and number theory. But in some sense, their study goes back to that of quadratic forms: Gauss [3] made a connection between quadratic forms and lattices, which was further developed by Dirichlet [2] and especially Minkowski [7]. Lattice theory is usually called *geometry of numbers* [1, 5, 10], a name due to Minkowski [7].

Theory

A lattice can be defined in many equivalent ways. To be precise, a few definitions need to be recalled. Let $\vec{x}, \vec{y} \in \mathbb{R}^n$ denote two row vectors $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ where the x_i's and the y_i's are real numbers (►Vector Space). Let $\langle \vec{x}, \vec{y} \rangle$ denote the Euclidean inner product of $\vec{x}$ with $\vec{y}$: $\langle \vec{x}, \vec{y} \rangle = \sum_{i=1}^{n} x_i y_i$. Let $\|\vec{x}\|$ denote the Euclidean norm of $\vec{x}$: $\|\vec{x}\| = \langle \vec{x}, \vec{x} \rangle^{1/2}$. A set of vectors $\{\vec{b}_1, \ldots, \vec{b}_d\}$ are said to be $\mathbb{R}$-linearly independent if and only if any equality of the form $\mu_1 \vec{b}_1 + \cdots + \mu_d \vec{b}_d = 0$, where the μ_i's are real numbers, implies that the μ_i's are all zero. Then the two most usual definitions of a lattice are the following ones:

– A lattice is a discrete (additive) ►subgroup of $\mathbb{R}^n$, that is, a non-empty subset $L \subseteq \mathbb{R}^n$ such that $\vec{x} - \vec{y} \in L$ whenever $(\vec{x}, \vec{y}) \in L^2$ (i.e., the group axiom), and where there exists a real $\rho > 0$ such that the simultaneous conditions $\vec{x} \in L$ and $\|\vec{x}\| \leq \rho$ imply that $\vec{x}$ be zero. With this definition, it is obvious that $\mathbb{Z}^n$ is a lattice (the group axiom is satisfied, and $\rho = 1/2$ works), and that any subgroup of a lattice is a lattice.
– A lattice is the set of all integer linear combinations of some set of $\mathbb{R}$-linearly independent vectors of $\mathbb{R}^n$, that is if $\vec{b}_1, \ldots, \vec{b}_d$ are linearly independent, then $L = \left\{ \sum_{i=1}^{d} n_i \vec{b}_i \mid n_i \in \mathbb{Z} \right\}$ is a lattice, and $[\vec{b}_1, \ldots, \vec{b}_d]$ is said to be a *basis* of L. With this definition, it is still obvious that $\mathbb{Z}^n$ is a lattice, but it is not clear that a subgroup of a lattice is still a lattice.

It is not difficult to prove that the above definitions are in fact equivalent (see [10]). To decide at first sight whether or not a given subset L of $\mathbb{R}^n$ is a lattice, the second definition is useful only when one already knows a potential basis, which is not necessary with the first definition. Both definitions suggest that lattices are discrete analogues of vector spaces: as a result, lattice theory bears much resemblance to linear algebra.

Lattice bases are not unique, but they all have the same number of elements, called the *dimension* or the *rank* of the lattice. Any lattice L of rank $d \geq 2$ has infinitely many bases. Indeed, one can see that to transform a lattice basis into another lattice basis, it is necessary and sufficient to apply a unimodular transformation, that is, a linear transformation represented by an integer matrix with determinant ± 1. This implies that the d-dimensional volume of the parallelepiped spanned by a lattice basis only depends on the

lattice, and not on the choice of the basis: it is called the *volume* or *determinant* of the lattice, denoted by $\mathrm{vol}(L)$ or $\det(L)$. By definition, it is equal to the square root of the following $d \times d$ determinant, where $(\vec{b}_1, \ldots, \vec{b}_d)$ is any basis of L:

$$\det(\langle \vec{b}_i, \vec{b}_j \rangle)_{1 \le i,j \le d} = \begin{vmatrix} \langle \vec{b}_1, \vec{b}_1 \rangle & \langle \vec{b}_1, \vec{b}_2 \rangle & \ldots & \langle \vec{b}_1, \vec{b}_d \rangle \\ \langle \vec{b}_2, \vec{b}_1 \rangle & \langle \vec{b}_2, \vec{b}_2 \rangle & \ldots & \langle \vec{b}_2, \vec{b}_d \rangle \\ \vdots & & \ddots & \vdots \\ \langle \vec{b}_d, \vec{b}_1 \rangle & \langle \vec{b}_d, \vec{b}_2 \rangle & \ldots & \langle \vec{b}_d, \vec{b}_d \rangle \end{vmatrix}.$$

The volume is useful to estimate the norm of lattice short vectors. More precisely, a classical theorem of Minkowski states that in any d-rank lattice L of $\mathbb{R}^n$, there is a nonzero vector $\vec{v} \in L$ such that $\|\vec{v}\| \le \sqrt{d}\mathrm{vol}(L)^{1/d}$. And this upper bound is optimal (up to a constant) in the worst case in the following sense: there exists $C > 0$ such that for all integer $d \ge 1$, there exists a d-rank lattice L of $\mathbb{R}^d$ such that for all nonzero $\vec{v} \in L$, the inequality $\|\vec{v}\| \ge C\sqrt{d}\mathrm{vol}(L)^{1/d}$ holds.

Applications

Lattices are classical objects of number theory, which have many applications in mathematics and computer science: see [4, 9]. In particular, they are widely used in public-key cryptology, both in cryptanalysis and cryptographic design: see [6, 8, 9] and the entries on ►lattice-based cryptography, ►NTRU, and ►lattice reduction.

By definition, any subgroup of $\mathbb{Z}^n$ is a lattice, and such lattices are the main ones used in computer science.

Recommended Reading

1. Cassels JWS (1971) An introduction to the geometry of numbers. Springer, Berlin
2. Dirichlet JPGL (1850) Über die Reduction der positiven quadratischen Formen in drei unbestimmten ganzen Zahlen. J Reine Angew Math 40:209–227
3. Gauss CF (1840) Recension der "Untersuchungen über die Eigenschaften der positiven ternären quadratischen Formen von Ludwig August Seeber." Göttingische Gelehrte Anzeigen, July 9, 1065ff, 1831. Repr. J Reine Angew Math 20:312–320. http://gdz.sub.uni-goettingen.de/dms/load/toc/?PPN=PPN23599524X&DMDID=dmdlog22
4. Grötschel M, Lovász L, Schrijver A (1993) Geometric algorithms and combinatorial optimization. Springer, Berlin
5. Gruber M, Lekkerkerker CG (1987) Geometry of numbers. North-Holland, Groningen
6. Micciancio D, Goldwasser S (2002) Complexity of lattice problems: a cryptographic perspective. The Kluwer international series in engineering and computer science, vol 671. Kluwer, Boston
7. Minkowski H (1896) Geometrie der Zahlen. Teubner, Leipzig
8. Nguyen PQ, Stern J (2001) The two faces of lattices in cryptology. In: Cryptography and lattices – proceedings of CALC'01, Providence. Lecture notes in computer science, vol 2146. Springer, Berlin, pp 146–180
9. Nguyen PQ, Vallée B (2009) The LLL algorithm: survey and applications. Information security and cryptography. Springer, Heidelberg
10. Siegel CL (1989) Lectures on the geometry of numbers. Springer, Berlin

Lattice Basis Reduction

►Lattice Reduction

Lattice Reduction

Phong Nguyen
Département d'informatique, Ecole normale supérieure, Paris, Cedex 05, France

Synonyms

Lattice basis reduction

Related Concepts

►Closest Vector Problem; ►Lattice; ►Lattice-Based Cryptography; ►Shortest Vector Problem

Definition

Among all the bases of a ►lattice, some are more useful than others. The goal of *lattice reduction* (also known as *lattice basis reduction*) is to find interesting bases, such as bases consisting of vectors which are relatively short and almost orthogonal. From a mathematical point of view, one is interested in proving the existence of at least one basis (in an arbitrary lattice) satisfying strong properties. From a computational point of view, one is rather interested in computing such bases in a reasonable time, given an arbitrary basis. In practice, one often has to settle for a trade-off between the quality of the basis and the running time.

Background

Lattice reduction goes back to the reduction theory of quadratic forms, initiated by Lagrange [11], Gauss [6], and Hermite [10]. Indeed, there is a natural relationship between lattices and positive definite quadratic forms, as first noted by Gauss [7], and later developed by Dirichlet [3] and especially Minkowski [15].

A classical fact of bilinear algebra states that any finite-dimensional Euclidean space has an orthogonal basis, that is, a basis consisting of vectors which are pairwise orthogonal. Such bases are very useful, so it is natural to ask

whether such bases also exist for lattices. Unfortunately, a lattice does not have in general an orthogonal basis. The goal of lattice reduction is to circumvent this problem.

Theory

Interesting lattice bases are called *reduced*, but there are many different notions of reduction, such as those of *Minkowski*, *Hermite–Korkine–Zolotarev*, *Lenstra–Lenstra–Lovász*, *etc.* Typically, a reduced basis is made of vectors which are relatively short and almost orthogonal. To explain what relatively short means, the so-called *successive minima* of a lattice are now defined.

The intersection of a d-dimensional lattice $L \subseteq \mathbb{R}^n$ with any bounded subset of $\mathbb{R}^n$ is always finite. It follows that there is a shortest nonzero vector in L, that is, there is $\mathbf{v} \in L\backslash\{0\}$ such that $\|\mathbf{u}\| \geq \|\mathbf{v}\|$ for all $\mathbf{u} \in L\backslash\{0\}$. Such a vector is not unique, but all such vectors must have the same norm. The first minimum of L is thus defined as $\lambda_1(L) = \|\mathbf{v}\|$. Note that if $\mathbf{v}$ is a shortest vector, then $-\mathbf{v}$ is also short but is not very interesting. To avoid such problems, one defines the successive minima as follows. For any integer k such that $1 \leq k \leq d$, the k-th successive minimum of L, denoted by $\lambda_k(L)$, is the radius of the smallest hyperball centered at the origin and containing at least k linearly independent vectors of L. The successive minima can be defined with respect to any norm, but the Euclidean norm is the most common.

One can show that there are linearly independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$ in L such that $\|\mathbf{v}_i\| = \lambda_i(L)$ for all $1 \leq i \leq d$. Surprisingly, as soon as $d \geq 4$, such vectors may not form a basis of L: the integral linear combinations of the $\mathbf{v}_i$'s may span a strict subset of L. Furthermore, as soon as $d \geq 5$, there may not exist a basis reaching simultaneously all the minima: there exist d-dimensional lattices such that for all bases $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$, $\|\mathbf{b}_i\| \neq \lambda_i(L)$ for at least some i. This is one of the reasons why there is no definition of reduction which is obviously better than all the others: for instance, one basis may minimize the maximum of the vector norms, while another minimizes the product of the norms, and it is not clear if one is better than the other. A reduced basis should have relatively short vectors in the sense that the i-th vector of the basis is not far away from the i-th minimum $\lambda_i(L)$, that is, $\|\mathbf{b}_i\|/\lambda_i(L)$ can be upper bounded.

The orthogonality of a basis is often measured by the product $\prod_{i=1}^{d} \|\mathbf{b}_i\|$ of the norms of the basis vectors divided by the volume of the lattice: this ratio is always ≥ 1, with equality if and only if the basis is orthogonal. Minkowski's second theorem states that for all $1 \leq k \leq d$, the geometric mean of the first k minima $\left(\prod_{i=1}^{k} \lambda_i(L)\right)^{1/k}$ is at most $\sqrt{\gamma_d}\mathrm{vol}(L)^{1/d}$, where γ_d is Hermite's constant in dimension d and $\mathrm{vol}(L)$ is the volume of the lattice (see the entry ►lattice for a definition). Hermite's constant is asymptotically linear in the dimension: $\gamma_d = \Theta(d)$. In particular, $\lambda_1(L) \leq \sqrt{\gamma_d}\mathrm{vol}(L)^{1/d}$, but this upper bound may not hold for the other minima: in fact, one can easily construct lattices such that the first minimum is arbitrarily small, while the other minima are large. In a random lattice, however, all the minima are asymptotically equivalent to $(\mathrm{vol}(L)/v_d)^{1/d} \sim \sqrt{d/(2\pi e)}\mathrm{vol}(L)^{1/d}$, where v_d denotes the volume of the d-dimensional unit ball.

Hermite, Korkine, Zolotarev, and Minkowski introduced strong notions of reduction: the corresponding reduced bases have very good properties but are very difficult to compute. For instance, bases reduced in the sense of Minkowski or of Hermite–Korkine–Zolotarev both include a shortest lattice vector, therefore finding such bases is already an NP-hard problem under randomized reductions as the lattice dimension increases (►Shortest Vector Problem). Lenstra, Lenstra, and Lovász [12] introduced the first notion of reduction to be interesting from both a mathematical point of view and a computational point of view, in the sense that such reduced bases are provably made of relatively short vectors (but not as short as, say, a Minkowski-reduced basis) and can be computed efficiently. More precisely, the celebrated LLL algorithm, given as input an arbitrary basis of a d-dimensional lattice L in $\mathbb{Q}^n$, outputs (in time polynomial in the size of the basis) a lattice basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ such that $\|\mathbf{b}_i\| = O((2/\sqrt{3})^d)\lambda_i(L)$ for all i. Smaller (slightly ►subexponential) approximation factors can be achieved in ►polynomial time using blockwise algorithms like Schnorr's reduction [20] and Gama-Nguyen's reduction [5].

Applications

Lattice reduction algorithms are useful because they enable to solve various lattice problems: approximating the ►Shortest Vector Problem and the ►Closest Vector Problem (see [1, 6, 8]), finding many short lattice vectors. This has proved invaluable in many areas in computer science (see [8]), notably in cryptology (see the survey [19]). Lattice reduction algorithms have arguably become the most popular tool in public-key cryptanalysis (see the survey [17]): they have been used to break various public-key cryptosystems, including many ►knapsack cryptographic schemes and ►lattice-based cryptography, but also certain settings of discrete-log signature schemes. Interestingly, they are also used in the most sophisticated attacks known against ►RSA (see [13, 17]): RSA with small secret exponent, chosen-message attacks on RSA signatures with peculiar

paddings, certain settings of RSA encryption with small public exponent, etc. In particular, Coppersmith opened in [2] a new avenue for cryptanalytic applications of lattice reduction when he revisited the connection between lattices and small solutions of polynomial equations. For instance, it can be shown using the LLL algorithm that, given an integer polynomial $f(X) \in \mathbb{Z}[X]$ of degree d such that the gcd of all the coefficients of f is coprime with a public integer N, one can find in time polynomial in $(d, \log N)$ all the integers $x_0 \in \mathbb{Z}$ such that $f(x_0) \equiv 0 \bmod N$ and $|x_0| \leq N^{1/d}$.

Open Problems

Lattice reduction is a very active research area: A lot of work is required to deepen our understanding of lattice reduction algorithms and to invent new lattice reduction algorithms.

Experimental Results

It should be emphasized that lattice reduction algorithms typically perform much better than their worst-case theoretical bounds would suggest: see [4] for an experimental assessment of the performances of the best lattice reduction algorithms in practice. For instance, in low dimension (say, less than 30), the LLL algorithm often outputs a shortest nonzero vector, while in high dimension, the approximation factor appears to be exponential on the average, but with a smaller constant than in the worst-case theoretical analysis, namely, the approximation factor is of the form c^d where c is significantly smaller than $2/\sqrt{3}$. This phenomenon has yet to be explained: from a theoretical point of view, the average-case behavior of lattice reduction algorithms is mostly unknown. The effectiveness of lattice reduction algorithm is another reason why lattice reduction has been so popular in cryptanalysis.

Recommended Reading

1. Babai L (1986) On Lovász lattice reduction and the nearest lattice point problem. Combinatorica 6:1–13
2. Coppersmith D (1997) Small solutions to polynomial equations, and low exponent RSA vulnerabilities. J Cryptol 10(4):233–260
3. Dirichlet JPGL (1850) Über die Reduction der positiven quadratischen Formen in drei unbestimmten ganzen Zahlen. J Reine Angew Math 40:209–227
4. Gama N, Nguyen PQ (2008) Predicting lattice reduction. In: Proceedings of EUROCRYPT'08, Istanbul. LNCS, vol 4965. Springer, Berlin
5. Gama N, Nguyen PQ (2008) Finding short lattice vectors within Mordell's inequality. In STOC'08: Proceedings of the 40th annual ACM symposium on theory of computing, Victoria. ACM, New York
6. Gauss CF (1801) Disquisitiones Arithmeticæ. Apud G. Fleischer, Leipzig
7. Gauss CF (1840) Recension der "Untersuchungen über die Eigenschaften der positiven tern ären quadratischen Formen von Ludwig August Seeber." Göttingische Gelehrte Anzeigen, July 9, 1065ff, 1831. Repr J Reine Angew Math 20:312–320. http://gdz.sub.uni-goettingen.de/dms/load/toc/?PPN=PPN23599524X&DMDID=dmdlog22
8. Grötschel M, Lovász L, Schrijver A (1993) Geometric algorithms and combinatorial optimization. Springer, Berlin
9. Gruber M, Lekkerkerker CG (1987) Geometry of numbers. North-Holland, Groningen
10. Hermite C (1850) Extraits de lettres de M. Hermite à M. Jacobi sur différents objets de la théorie des nombres. J Reine Angew Math 40:279–290
11. Lagrange JL (1773) Recherches d'arithmétique. Nouv Mém Acad Roy Soc Belles Lett (Berlin):265–312
12. Lenstra AK, Lenstra Jr HW, Lovász L (1982) Factoring polynomials with rational coefficients. Math Ann 261: 513–534
13. May A (2009) Using LLL-reduction for solving RSA and factorization problems: a survey. In: Nguyen PQ, Vallée B (eds) The LLL algorithm: survey and applications. Information security and cryptography. Springer, Heidelberg
14. Micciancio D, Goldwasser S (2002) Complexity of lattice problems: a cryptographic perspective. The Kluwer international series in engineering and computer science, vol 671. Kluwer, Boston
15. Minkowski H (1896) Geometrie der Zahlen. Teubner, Leipzig
16. Nguyen PQ (2009) Hermite's constant and lattice algorithms. In: Nguyen PQ, Vallée B (eds) The LLL algorithm: survey and applications. Information security and cryptography. Springer, Heidelberg
17. Nguyen PQ (2009) Public-key cryptanalysis. In: Recent trends in cryptography. Contemporary mathematics series, vol 477. AMS–RME, Providence
18. Nguyen PQ, Vallée B (2010) The LLL algorithm: survey and applications. Information security and cryptography. Springer, Heidelberg
19. Nguyen PQ, Stern J (2001) The two faces of lattices in cryptology. In: Cryptography and lattices – proceedings of CALC '01, Providence. LNCS, vol 2146. Springer, Berlin, pp 146–180
20. Schnorr CP (1987) A hierarchy of polynomial lattice basis reduction algorithms. Theor Comput Sci 53:201–224

Lattice-Based Cryptography

Daniele Micciancio
Department of Computer Science & Engineering, University of California, San Diego, CA, USA

Related Concepts

►Closest Vector Problem; ►Lattice; ►Lattice Reduction; ►NTRU; ►Post-quantum Cryptography; ►Public Key Cryptography; ►Shortest Vector Problem

Definition

Lattice-based cryptography is a generic term used to encompass a wide range of cryptographic functions whose security is based on the conjectured intractability of ►Lattice problems, like (variants of) the ►Shortest Vector Problem and the ►Closest Vector Problems.

For applications of lattices in cryptanalysis, ►Lattice Reduction.

Background

The study of lattice-based cryptography was pioneered by Ajtai in 1996 [1], who proved that certain variants of the ►knapsack cryptographic schemes are at least as hard to break *on the average* as approximating (the length estimation variant of) the ►Shortest Vector Problem (GapSVP) within factors that grow only polynomially in the dimension n of the lattice. Two distinguishing features of Ajtai's result, and lattice-based cryptography in general, are that:

- Breaking the cryptographic functions is provably at least as hard as solving certain lattice problems in the *worst-case*.
- the underlying lattice problems are not known to be efficiently solvable by quantum computers

This should be contrasted with mainstream cryptographic functions based on ►number theory, which can be broken by quantum algorithms, and, even classically, rely on *average-case* complexity assumptions, a qualitatively much stronger requirement than worst-case hardness (Refer the entry ►Computational Complexity for further discussion on complexity assumptions and the entry on ►Post-Quantum Cryptography for a summary of attacks using quantum computers and systems that are unaffected).

Shortly after [1], Ajtai and Dwork [2] proposed a public-key encryption scheme also based on worst-case complexity of lattice problems. Since then, many more lattice-based cryptographic functions have been discovered. (See section on applications.)

Theory

The best currently known ►polynomial-time algorithms to (approximately) solve GapSVP and other lattice problems only produce solutions within an approximation factor which is almost exponential in the dimension n of the ►Lattice. (Refer entries on ►Lattice Reduction and the ►Shortest Vector Problem.) So, Ajtai's conjecture that no polynomial time algorithm can approximate these problems within any polynomial factor n^c is quite reasonable and supported by both theoretical and experimental results. However, lower-degree polynomials (i.e., smaller values of c) are more desirable because they give stronger security guarantees. The strongest result along these lines known to date [8] shows that Ajtai's function can be based on the inapproximability of GapSVP (as well as other problems) within factors $n^{1+o(1)}$ essentially linear in the lattice dimension.

The security of lattice-based public-key encryption was originally based on the conjectured intractability of a special version of SVP, called the "unique" SVP [2, 13]. This problem has been subsequently shown to be equivalent (up to small polynomial factors) to the standard GapSVP [5].

One of the less satisfactory aspects of lattice-based cryptography is that it typically yields cryptographic functions with very large keys. This is due to the fact that an n-dimensional lattice is described by an $n \times n$ matrix, which requires at least n^2 space to store. This obstacle can be overcome using lattices with a special structure which admit a much more compact representation, e.g., cyclic lattices where a single vector can be used to represent an n-dimensional lattice by cyclically rotating its coordinates. Proposals along these lines first appeared in the ►NTRU cryptosystem, but without the support of a security proof. The theoretical study of efficient cryptographic functions based on structured lattices was initiated in 2002 by Micciancio [7], who exhibited a one-way function with key size and computation time essentially linear in the lattice dimension n, and still provably secure based on the worst-case inapproximability of the ►Shortest Vector Problem over cyclic lattices.

Another major step in the development of lattice-based cryptography was the introduction of the "learning with errors" (LWE) problem (an average-case version of bounded distance decoding, Refer the entry ►Closest Vector Problem) by Regev [14], who first gave evidence that the problem is hard assuming lattice problems are hard even for *quantum* computers. The LWE problem was used by Peikert et al. [3, 10–12] to considerably expand the applicability of lattice-based cryptography to a wide range of cryptographic primitives.

Applications

The techniques of [1, 7, 8, 14] have been extended in a number of ways to obtain a wide range of cryptographic primitives, including public-key encryption secure against ►adaptive chosen ciphertext attack [12], ►digital signatures schemes [3, 4], ►oblivious transfer protocols [11], noninteractive ►zero-knowledge proof systems [10], very efficient collision-resistant ►hash functions [6], ►identity-based encryption [3], and more.

Open Problems

The main open problem in the area of lattice-based cryptography is probably to gain confidence in their security. While these functions are supported by asymptotic security proofs, the proofs do not offer much guidance regarding concrete values of the security parameters that should be used in practice. This is both because the proofs are not tight (i.e., there is a substantial gap between the best known attack and the strongest provable security guarantee) and also because they start from worst-case problems whose exact complexity is still not very well understood.

Another important open problem is to make lattice-based cryptography more efficient and attractive in practice. The use of cyclic, or similarly structured, lattices promises to give substantial efficiency improvements over cryptography based on general lattices. (See [6] for an illustrative example.) Still, for many cryptographic primitives, it is still not known how to take full advantage of structured lattices to achieve efficiency gains.

The reader is referred to the chapter [9] for an introduction to lattice-based cryptography, and the papers in the references for more recent developments.

Recommended Reading

1. Ajtai M (1996) Generating hard instances of lattice problems (extended abstract). In: Proceedings of the twenty-eighth annual ACM Symposium on the Theory of Computing (STOC'96), Philadelphia, 22–24 May 1996. ACM Press, New York, pp 99–108
2. Ajtai M, Dwork C (1997) A public-key cryptosystem with worstcase/average-case equivalence. In: Proceedings of the twenty-ninth annual ACM Symposium on Theory of Computing (STOC '97), El Paso, 4–6 May 1997. ACM Press, New York, pp 284–293
3. Gentry C, Peikert C, Vaikuntanathan V (2008) Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of STOC '08, Victoria, 17–20 May 2008. ACM Press, New York, pp 197–206
4. Lyubashevsky V, Micciancio D (2008) Asymptotically efficient lattice-based digital signatures. In: Proceedings of TCC '08, New York, 19–21 March 2008. Lecture notes in computer science, vol 4948. Springer, Berlin, pp 37–54
5. Lyubashevsky V, Micciancio D (2009) On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In: Proceedings of CRYPTO 2009, Santa Barbara, 16–20 August 2009. Lecture notes in computer science, vol 5677. Springer, Berlin, pp 577–594
6. Lyubashevsky V, Micciancio D, Peikert C, Rosen A (2008) Swifft: a modest proposal for FFT hashing. In: Proceedings of FSE '08, Lausanne, 10–13 February 2008. Lecture notes in computer science, vol 5086. Springer, Berlin, pp 54–72
7. Micciancio D (2007) Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. Comput Complex 16(4):365–411
8. Micciancio D, Regev O (2007) Worst-case to average-case reductions based on Gaussian measure. SIAM J Comput 37(1):267–302
9. Micciancio D, Regev O (2008) Lattice-based cryptography. In: Bernstein DJ, Buchmann J, Dahmén E (eds) Post-quantum cryptography. Springer, Berlin
10. Peikert C, Vaikuntanathan V (2008) Noninteractive statistical zeroknowledge proofs for lattice problems. In: Proceedings of CRYPTO '08, Santa Barbara, 17–21 August 2008. Lecture notes in computer science, vol 5157. Springer, Berlin, pp 536–553
11. Peikert C, Vaikuntanathan V, Waters B (2008) A framework for efficient and composable oblivious transfer. In: Proceedings of CRYPTO '08, Santa Barbara, 17–21 August 2008. Lecture notes in computer science, vol 5157. Springer, Berlin, pp 554–571
12. Peikert C, Waters B (2008) Lossy trapdoor functions and their applications. In: Proceedings of STOC '08, Victoria, 17–20 May 2008. ACM Press, New York, pp 187–196
13. Regev O (2004) New lattice based cryptographic constructions. J ACM 51(6):899–942
14. Regev O (2005) On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of STOC '05, Baltimore, 22–24 May 2005. ACM Press, New York, pp 84–93

LCM

►Least Common Multiple

Least Common Multiple

Scott Contini
Silverbrook Research, New South Wales, Australia

Synonyms

LCM

Related Concepts

►Greatest Common Divisor; ►Number Theory

Definition

The *least common multiple* (lcm) of a set of positive integers $\{a_1, \ldots, a_k\}$ is the smallest positive integer that is an integer multiple of every element of the set. This is denoted $\mathrm{lcm}(a_1, \ldots, a_k)$, or sometimes just $[a_1, \ldots, a_k]$.

Theory

For example, $\mathrm{lcm}(21, 91) = 273$ because 273 is a multiple of both 21 and 91 as $273 = 13 \cdot 21$ and $273 = 3 \cdot 91$, and no positive integer smaller than 273 has this property.

For a pair of integers, the least common multiple is related to the ►greatest common divisor by the relation $\gcd(a_1, a_2) \cdot \mathrm{lcm}(a_1, a_2) = a_1 \cdot a_2$.

Least Privilege

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

Synonyms
Minimal privilege

Related Concepts
►Access Control Policies, Models, and Mechanisms

Definition
The *least privilege principle* states that a subject (user or program) should be given only those privileges it actually needs to perform its job.

Theory
The least privilege principle was defined by Jerry Saltzer and Mike Schroeder [1] as follows.

- Every program and every user of the system should operate using the least set of privileges necessary to complete the job.

The importance of the least privilege principle is widely recognized since it minimizes the danger of damage due to inadvertent errors, Trojan Horses, or intruders masquerading as legitimate users. Although the least privilege principle is by itself a simple and fundamental design principle, in real practice its enforcement is not straighforward. The main motivation is that it may be difficult to determine the least amount of privileges a user/process will ever need to perform its job.

Recommended Reading
1. Saltzer JH, Schroeder MD (1975) The protection of information in computer systems. Proc. IEEE, 63(9):1278–1308

Legendre Symbol

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

Related Concepts
►Jacobi Symbol; ►Prime Number; ►Quadratic Residue

Background
The Legendre symbol was introduced by A.M. Legendre in 1798.

Definition
The *Legendre symbol* of an integer x modulo a prime p is 0 if x is divisible by p, and otherwise +1 if x has a square root modulo p, and −1 if not.

Theory
Let p be an odd ►prime number and let x be an integer. If x is a ►quadratic residue, i.e., if x is relatively prime to p and the equation (►Modular arithmetic)

$$x \equiv y^2 \bmod p$$

has an integer solution y, then the Jacobi symbol of x modulo p, written as (x/p) or $\left(\frac{x}{p}\right)$, is +1. If x is a *quadratic nonresidue* – i.e., x is relatively prime to p and has no square roots – then its Legendre symbol is −1. If x is not relatively prime to p then $\left(\frac{x}{p}\right) = 0$.

The Legendre symbol may be efficiently computed using the *Quadratic Reciprocity Theorem* or by modular exponentiation (►Exponentiation Algorithms) as

$$\left(\frac{x}{p}\right) = x^{\frac{p-1}{2}} \bmod p$$

Levels of Trust

Stephen M. Papa, William D. Casper
High Assurance Computing and Networking Labs (HACNet), Department of Computer Science and Engineering, Bobby B. Lyle School of Engineering, Southern Methodist University, Houston, TX, USA

Synonyms
Commercial off-the-shelf; Information assurance; Integrated circuit; Intellectual property

Related Concepts
►PKI Trust Models; ►Root of Trust; ►Trusted Boot; ►Trusted Computing; ►Trust Management

Definition
Trust Level: An appropriate level of hardware and software protection mechanisms in a computer system based on its intended use, and is established based on a risk analysis

that includes a probability and consequence of occurrence of an attack on the system.

Background

Computer security is reliant on trust. This trust is composed of several fundamental principles, including confidence that the targeted system is configured as expected, will operate as intended, and has not already been compromised or exploited. A verification strategy with appropriate methodology should be used to validate this trust. The validation can be done with a combination of hardware attestation and software integrity verification.

A trust level is a useful method to identify the required hardware and software security protection mechanisms that a system must include to protect the data confidentiality, availability, and integrity once this data (software or other information) is present in an operational system. The level selected should be based on the desired level of trust.

Theory

As a system is designed, developed, integrated, and deployed a "weak link" in any security protection mechanism has the potential for allowing an attacker to gain access to the data being protected. Protection mechanisms are counter-measures against these attacks and are often needed to provide trust that a system will protect the data it is storing, sending, or processing. These mechanisms may include information assurance (IA), cryptography, tamper protection, third-party certificate authorities, shared secrets/keys, etc. Protection mechanisms are designed into the system to ensure that the confidentiality, integrity, and availability of the data are maintained while it is stored, being transferred, or being used within the deployed system.

Mechanisms to create trust in a system may include cryptographic (hard) and non-cryptographic (soft) trust mechanisms. Hard trust mechanisms may include trusted boot, authentication between roots of trust in the system or network, methods and protocols to establish and maintain secure communication channels or exchange key material, digital signature for verification of data, third-party certificates, and the use of secure processors to protect data during run-time operation. Soft trust mechanisms may include verification of the hardware and software configuration to ensure proper system components are present, trusted operating systems, software designed to protect against known attacks, and run-time checks for attacks on the system. Trusted components are often the basis for installing trust in an overall design, but verifying the true trustworthiness of trusted components is not a trivial task [1].

The specific mechanisms selected to establish trust needs to be based on some required level of system trust.

The first step to establishing a system's required trust level is assessing the relative risk the data will experience once the computing system is deployed. Risk of attack to deployed data may come from one or more sources. At the root the risks are people who have malicious intents, and the expected access they may have to the system. Typical usage scenarios that would put the data at risk may include one or more of the following:

1. An authorized malicious user with system usage rights and physical system access
2. An authorized malicious user with system usage rights but no physical system access
3. An unauthorized malicious insider or outsider with physical access to the system
4. An unauthorized malicious outsider with network access to the system
5. Data contained in nonvolatile storage within the computer system
6. Data contained in a server or other storage device not physically part of the computer system

Users may be authorized to use a system, application, or service, however, use does not imply rights to the actual data processed within the system. An example of this is a video game, where the user can interact and play the game, but where the video game creator may want to prevent piracy of their IP in the game.

Once the expected usage scenarios are understood, the probability and consequence of data compromise can be quantified, and based on this analysis the risk of data compromise can be understood. Often the consequence of loss may be stated in terms of potential monetary loss, loss of competitive advantage, loss of the enterprise's ability to be profitable, or even in terms of national security.

Ultimately the data owner, an organization or person who created or owns the data, must assess the relative importance of the data requiring protection and the risk associated with its loss or compromise. The data owner must then decide if the system that will contain the data is designed with sufficient protection mechanisms so that the risk is acceptable.

Once a required level of trust has been identified, a trusted development environment (facilities, networks, and people), development tools, and hardware and software should be selected or developed to provide a consistent level of protection so that an appropriate level will exist in the deployed system. Discrete levels of trust provide a useful framework for identifying the design, development, and verification requirements for a given system. In any

deployed system a level of trust can be established using a mix of IA, trust, and tamper protection requirements.

A framework based on levels of trust is established below to provide data owners and system designers a method of understanding the required protection mechanisms. A trust level criteria can be used by both the data owner and system designers to come to a common understanding of the protection requirements for the system. This criteria does not formally exist in many commercial environments, but has existed in many forms within many national government security organizations. When the data owner is a government agency, the criteria are based on data sensitivity or classification level, and the required protection level is established based on this level and the expected environment the system is expected to exist in.

Based on the sensitivity of the information within the system and the intended use of the product, there are varying levels of trust required. A definition of the levels of trust that can be used as a criterion for establishing the required level of trust has its roots based on the five levels identified by FIPS-140-3 draft [2] for cryptographic devices. A proposal on levels of software trust is discussed in [3]. These levels are referred to as classes in that particular paper and focused specifically on software trust and the software development process. The definitions discussed below extend these concepts to included hardware and system design elements, hardware and software acquisition choices, maintenance processes, and even the development teams themselves.

Applications

It is insufficient to just have trust in the developed software or at the network interface. For any system to be trusted there are requirements for establishing trust during the development, deployment, and maintenance of the system. Trust in a computing system can be established and maintained if there is an appropriate and consistent level of trust in each of the following aspects of the product life cycle.

Based on the FIPS-based security levels the following five trust levels are defined below, and provide an overview of trust and protection requirements based on the data's risk of compromise throughout the computing system's life cycle.

Level 1 Trust (Very Low Risk of Compromise)

No trust is required. No special, sensitive, high value, or significant data (information or software) will exist in the deployed system.

Establishing and maintaining trust throughout the development process or in the deployed systems is not cost effective or relevant in the system design and is not required.

Level 2 Trust (Low Risk of Compromise)

Minimal trust required. Data in the deployed system is limited to low-value proprietary data or personal data readily available in public information. Its protection is not critical to long-term success of an organization or individual. Trust in the development environment and protection measures within the deployed system is optional.

Establishing and maintaining trust throughout the development process and in the deployed system may not be cost effective or relevant in the system design. Deployed system trust includes software only designs to protect the data and software. Software support for trust may include software decryption and integrity verification of software and data. FIPS140-3 Security Level 2 crypto requirements may be applicable.

Level 3 Trust (Medium Risk of Compromise)

Medium trust required. Data includes important proprietary or personal data. Trust in the development environment and protection measures within the deployed system are required. The level of trust and protection mechanisms designed into the system must be commensurate with the expected risk of exposure in the deployed system.

Establishing and maintaining trust throughout the development process is cost effective and relevant in the system design. Efforts to establish trust in the deployed system include software designs to protect the data and software, and if required hardware support of these designs. Purchased software is selected and integrated with regard to trust. Preference to software that has been evaluated or well tested for security flaws may be a design consideration. Software is developed with processes that include security criteria. Code inspections with security checklists are used, and engineers are trained in software protection methods. All security-relevant software is verified using code integrity checking tools. Hardware may be COTS systems, boards, and components. Selection is based on performance to functional requirements, and support of the security requirements. Modifications for trust support in hardware are required if needed by software to secure the system.

System or software installation or upgrades are performed by trusted personnel, tools, or applications. In deployed systems hardware supports for trust may be integral to the design. Operational software support for trust

includes use of decryption and integrity verification of software and data. Further checks on system configuration and operation may be performed to improve system trust. FIPS140-3 Security Level 3 crypto requirements are applicable.

Level 4 Trust (High Risk of Compromise)

High-level trust required. Data is critical to long-term success of the organization or individuals. Trust in the development environment and protection measures within the deployed system are required. The level of trust and protection mechanisms designed into the system must be commensurate with the expected risk of exposure in the deployed system.

Establishing and maintaining trust throughout the development process is cost effective and relevant in the system design. Efforts to establish trust in the deployed system include software designs to protect the data and software, and if available hardware support of these designs.

The development environment should not be connected to the Internet and strong security measures are in place. Lead engineers and managers must control product configuration. Access controls are in place to keep developers from accessing specific data or resources. Physical access to development areas may be required. Security screening or background checks of all key personnel involved in development or deployment of the product is required.

Hardware and software development tools must have a high level of assurance from a security perspective. Only open or closed source tools that come from reputable companies are selected. Tools' outputs are to be verified for Trojans and other security flaws.

Purchased software is selected and integrated with regard to trust. Selected software must have been evaluated or well tested for security flaws. Developed software is done using processes that include security criteria. Code inspections with security checklists are used, and engineers are trained in secure software development standards. All security-relevant software is verified using code integrity checking tools.

Hardware may be COTS systems, boards, and components. Selection is based on performance to functional requirements, and must include support of the security requirements.

System and software installation or upgrades are performed by trusted personnel, tools, or applications. Specific measures are in place to ensure data and software confidentiality and integrity during upgrades. These measures may include attestation of the new configuration by trusted third parties, or with digital signatures.

In deployed systems, hardware supports for trust is integral to the design. Modifications to hardware to support establishment of trust is required to help secure the system. This support starts with digital signature verification and decryption of boot software as part of the boot process. Additional hardware crypto engines to support cryptographic functions required by software are needed in the hardware. Operational software support for trust includes use of decryption and integrity verification of key system elements, of software and data. Further checks on system configuration and operation may be performed.

Deployed system configuration control and release is performed by trusted personnel, configuration tools, or applications. FIPS140-3 Security Level 4 crypto requirements are applicable.

Level 5 Trust (Very High Risk of Compromise)

Highest level of trust required. Loss of data will result in long-term and permanent damage to the organization, country, individuals, or groups of individuals. The level of trust and protection mechanisms designed into the system must be commensurate with the expected risk of exposure in the deployed system.

Establishing and maintaining trust throughout the development process is cost effective and relevant in the system design. Efforts to establish trust in the system include software, hardware, and firmware designs to protect the data and software.

Development environment is not connected to the Internet and strong security measures are in place. Lead engineers or managers control product configuration and are trusted. Access controls are in place to keep developers from accessing specific data or resources. Physical access controls to development area is required. Security screening or background checks of all personnel involved in the project is required.

SW and hardware development tools must have a high level of assurance from a security perspective. Only open or closed source tools that come from reputable companies are selected. Tools outputs are be verified for Trojans and other security flaws.

Purchased software is selected and integrated with regard to trust. Selected software must have been evaluated or well tested for security flaws. Developed software is done with processes that include security criteria. Code inspections with security checklists are used; engineers are trained in secure software development. All

security-relevant software is verified using code integrity checking tools.

Hardware must be designed and developed to meet trust requirements. Selection is based on performance to functional requirements, and must include support of the security requirements. Specific hardware designs are used to create trust and are required to secure and verify the system. In some systems the individual ICs must be trusted (to prove lack of Trojans or other malicious circuitry).

System installation and upgrades are performed by trusted personnel, tools, or applications. Specific measures are in place to ensure data and software confidentiality and integrity during upgrades. Deployed system configuration control and release is performed by trusted personnel, configuration tools, or applications. FIPS 140-3 Security Level 5 crypto requirements are applicable.

Open Problems

A methodology to assess attacks risk (probability of an attack and consequence of the occurrence of a successful attack) and quantifying the trust level/protection mechanisms to reduce the probability of occurrence still remain to be developed.

Recommended Reading

1. Bertrand M (2003) The grand challenge of trusted components. In: Proceedings of the 25th international conference on software engineering (ICSE'03), Portland, Oregon, IEEE
2. FIPS PUB 140-3 (DRAFT) Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, 20899-8900
3. Amoroso E, Nguyen T, Weiss J, Watson J, Lapiska P, Starr T (1991) Toward an approach to measuring software trust. In: Proceedings of the 1991 IEEE computer society symposium on research in security and privacy, Oakland, IEEE

LFSR

►Linear Feedback Shift Register

Linear Complexity

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

Related Concepts

►Berlekamp–Massey Algorithm; ►Combination Generator; ►Filter Generator; ►Linear Feedback Shift Register; ►Minimal Polynomial; ►Stream Cipher

Definition

The *linear complexity* of a semi-infinite sequence $\mathbf{s} = (s_t)_{t\geq 0}$ of elements of $\mathbf{F}_q$, $\Lambda(\mathbf{s})$, is the smallest integer Λ such that $\mathbf{s}$ can be generated by a ►linear feedback shift register (LFSR) of length Λ over $\mathbf{F}_q$, and is ∞ if no such LFSR exists. By way of convention, the linear complexity of the all-zero sequence is equal to 0. The linear complexity of a linear recurring sequence corresponds to the degree of its ►minimal polynomial.

The linear complexity $\Lambda(\mathbf{s}^n)$ of a finite sequence $\mathbf{s}^n = s_0 s_1 \ldots s_{n-1}$ of n elements of $\mathbf{F}_q$ is the length of the shortest LFSR which produces $\mathbf{s}^n$ as its first n output terms for some initial state. The linear complexity of any finite sequence can be determined by the ►Berlekamp–Massey algorithm. An important result due to Massey (1969) is that, for any finite sequence $\mathbf{s}^n$ of length n, the LFSR of length $\Lambda(\mathbf{s}^n)$ which generates $\mathbf{s}^n$ is unique if and only if $n \geq 2\Lambda(\mathbf{s}^n)$.

Theory

The linear complexity of an infinite linear recurring sequence $\mathbf{s}$ and the linear complexity of the finite sequence $\mathbf{s}^n$ composed of the first n digits of $\mathbf{s}$ are related by the following property: if $\mathbf{s}$ is an infinite linear recurring sequence with linear complexity Λ, then the finite sequence $\mathbf{s}^n$ has linear complexity Λ for any $n \geq 2\Lambda$. Moreover, the unique LFSR of length Λ that generates $\mathbf{s}$ is the unique LFSR of length Λ that generates $\mathbf{s}^n$ for every $n \geq 2\Lambda$.

For a sequence $\mathbf{s} = s_0 s_1 \ldots$, the sequence of the linear complexities $(\Lambda(\mathbf{s}^n))_{n\geq 1}$ of all subsequences $\mathbf{s}^n = s_0 \ldots s_{n-1}$ composed of the first n terms of $\mathbf{s}$ is called the *linear complexity profile* of $\mathbf{s}$.

The expected linear complexity of a binary sequence $\mathbf{s}^n = s_0 \ldots s_{n-1}$ of n independent and uniformly distributed binary random variables is

$$E[\Lambda(\mathbf{s}^n)] = \frac{n}{2} + \frac{4+\varepsilon(n)}{18} + 2^{-n}\left(\frac{n}{3} + \frac{2}{9}\right),$$

where $\varepsilon(n) = n \bmod 2$.

If $\mathbf{s}$ is an infinite binary sequence of period 2^n which is obtained by repeating a sequence $s_0 \ldots s_{2^n-1}$ of 2^n independent and uniformly distributed binary random variables, its expected linear complexity is

$$E[\Lambda(\mathbf{s})] = 2^n - 1 + 2^{-2^n}.$$

Further results on the linear complexity and on the linear complexity profile of random sequences can be found in [1].

Recommended Reading

1. Rueppel RA (1986) Analysis and design of stream ciphers. Springer-Verlag, New York

Linear Congruential Generator

Caroline Fontaine
Lab-STICC/CID and Telecom Bretagne/ITI,
CNRS/Lab-STICC/CID and Telecom Bretagne,
Brest Cedex 3, France

Related Concepts

►Pseudorandom Generator; ►Stream Cipher

Definition

A linear congruential generator is a pseudorandom generator that produces a sequence of numbers $x_1, x_2, x_3, \ldots$ according to the following linear recurrence:

$$x_t = ax_{t-1} + b \mod n$$

for $t \geq 1$ (modular arithmetic); integers a, b, and n characterize entirely the generator, and the seed is x_0.

Example

Considering for example $a = 3$, $b = 5$, $n = 17$, and $x_0 = 2$, the sequence produced by the linear congruential generator will be 11, 4, 0, 5, 3, 14, 13, 10, 1, 8, 12, 7, 9, 15, 16, ...

Background

Pseudorandom generators are very useful in cryptography, in protocols, but also in the generation of keystreams in stream ciphers. In this case, they have to present strong properties to face cryptanalysis.

Applications

Such generators are easy to implement and pass the following statistical tests: Golomb's randomness postulates, frequency test, serial test, poker test, runs test, autocorrelation test, Maurer's universal statistical test. Hence, it can be considered as a good candidate for generating strong pseudorandom sequences. However, there is an important drawback: the sequence is *predictable*: given a piece of the sequence, it is easy to reconstruct the whole rest of it, even if the attacker does not know the exact values of a, b, and n [1, 2]. So, it would be very dangerous to use it in a cryptographic purpose. Some variants have been considered, using either several terms in the linear recurrence equation,

$$x_t = a_1 x_{t-1} + a_2 x_{t-2} + \ldots + a_\ell x_{t-\ell} + b \mod n,$$

or a quadratic recurrence relation,

$$x_t = ax_{t-1}^2 + bx_{t-1} + c \mod n.$$

In both cases, it can be shown that the sequence remains predictable [3, 4]. Another variant has also been studied, considering that some least significant bits of the produced integers are discarded; but such sequences still are predictable [3, 5, 6]. A more precise state of the art about cryptanalytic attacks of such generators can be found in [7].

Recommended Reading

1. Plumstead JB (1982) Inferring a sequence generated by a linear congruence. In: Proceedings of the IEEE 23rd annual symposium on foundations of computer science, IEEE, pp 153–159
2. Plumstead JB (1983) Inferring a sequence produced by a linear congruence. Advances in Cryptology – Crypto'82, Plenum Press, New York, pp 317–319
3. Boyar J (1989) Inferring sequences produced by a linear congruential generator missing low-order bits. J Cryptol 1:177–184
4. Krawczyk H (1992) How to predict congruential generators. J Algorithms 13:527–545
5. Frieze AM, Hastad J, Kannan R, Lagarias JC, Shamir A (1988) Reconstructing truncated integer variables satisfying linear congruence. SIAM J Comput 17:262–280
6. Stern J (1987) Secret linear congruential generators are not cryptographically secure. In: Proceedings of the IEEE 28th annual symposium on foundations of computer science, IEEE, pp 421–426
7. Brickell EF, Odlyzko AM (1992) Cryptanalysis: a survey of recent results. Contemporary Cryptology: The Science of Information Integrity, IEEE-Press, New York, pp 501–540

Linear Consistency Attack

Anne Canteaut
Project-team SECRET, INRIA Paris-Rocquencourt,
Le Chesnay, France

Related Concepts

►Linear Cryptanalysis for Stream Ciphers; ►Stream Cipher

Definition

The *linear consistency attack* is a divide-and-conquer technique which provides a ►known plaintext attack on ►stream ciphers. It was introduced by Zeng, Yang, and Rao in 1989. It has been applied to various keystream generators, like the *Jenning generator* [2], the *stop-and-go generator* [2], and the ►E0 cipher used in Bluetooth [1].

Theory

The linear consistency attack applies as soon as it is possible to single out a portion K_1 of the secret key and to form a system $Ax = b$ of linear equations, where the matrix A only

depends on K_1 and the right-side vector b is determined by the known keystream bits. Then, an exhaustive search for K_1 can be performed. The correct value of K_1 can be distinguished from a wrong one by checking whether the linear system is consistent or not. Once K_1 has been recovered, the solution x of the system may provide some additional bits of the secret key.

Recommended Reading

1. Fluhrer SR, Lucks S (2001) Analysis of the E0 encryption system. In: Selected areas in cryptography – SAC 2001, Lecture notes in computer science, vol 2259. Springer, Berlin, pp 38–48
2. Zeng K, Yang CH, Rao TRN (1989) On the linear consistency test (LCT) in cryptanalysis with applications. In: Advances in cryptology – CRYPTO'89. Lecture notes in computer science, vol 435. Springer, Berlin, pp 164–174

Linear Cryptanalysis for Block Ciphers

Alex Biryukov[1], Christophe De Cannière[2]
[1] FDEF, Campus Limpertsberg, University of Luxembourg, Luxembourg
[2] Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven-Heverlee, Belgium

Related Concepts

►Block Ciphers; ►FEAL

Definition

Linear cryptanalysis is a ►known plaintext attack in which the attacker studies probabilistic linear relations (called *linear approximations*) between parity bits of the plaintext, the ciphertext, and the secret ►key. Given an approximation with high probability, the attacker obtains an estimate for the parity bit of the secret key by analyzing the parity bits of the known plaintexts and ciphertexts. Using auxiliary techniques, he or she can usually extend the attack to find more bits of the secret key.

Background

Linear cryptanalysis is a powerful method of ►cryptanalysis of block ciphers introduced by Matsui in 1993 [13]. The attack in its current form was first applied to the ►Data Encryption Standard (DES), but an early variant of linear cryptanalysis, developed by Matsui and Yamagishi, was already successfully used to attack ►FEAL in 1992 [12].

Theory

The next section provides some more details about the attack algorithm. Sections "Piling-up Lemma" to "Provable security against linear cryptanalysis" discuss a number of practical and theoretical aspects which play a role in linear cryptanalysis. Section "Comparison with differential cryptanalysis" points out analogies between linear and differential cryptanalysis, and Section "Extensions" concludes with some extended variants of linear cryptanalysis.

Outline of a Linear Attack

Following Matsui's notation, we denote by $A[i]$ the ith bit of A and by $A[i_1, i_2, \ldots, i_k]$ the parity bit $A[i_1] \oplus A[i_2] \oplus \cdots \oplus A[i_k]$. The first task of the attacker is to find a suitable linear approximation. For simple linear operations such as an XOR with the key or a permutation of bits, very simple linear expressions can be written which hold with probability one. For nonlinear elements of a cipher such as S-boxes, one tries to find linear approximations with probability p that maximizes $\left|p - \frac{1}{2}\right|$. Approximations for single operations inside a cipher are then further combined into approximations that hold for a single round of a cipher. By appropriate concatenation of one-round approximations, the attacker eventually obtains an approximation for the whole cipher of the type:

$$P[i_1, i_2 \ldots, i_a] \oplus C[j_1, j_2, \ldots, j_b] = K[k_1, k_2, \ldots, k_c], \quad (1)$$

where $i_1, i_2, \ldots, i_a$, $j_1, j_2, \ldots, j_b$, and $k_1, k_2, \ldots, k_c$ denote fixed bit locations. Note that such approximation is interesting only if it holds with a probability $p \neq \frac{1}{2}$ (how this probability is calculated is explained in the next section). For DES, Matsui found such an approximation with probability $\frac{1}{2} + 2^{-24}$. Using this approximation, a simple algorithm based on the maximum likelihood method can be used to find one parity bit $K[k_1, k_2, \ldots, k_c]$ of the key:

Given a pool of N random known plaintexts, let T be the number of plaintexts such that the left side of the Eq. 1 is 0.

if $(T - N/2) \cdot (p - 1/2) > 0$ then
    $K[k_1, \ldots, k_c] = 0$
else
    $K[[k_1, \ldots, k_c]] = 1$
end if

In order for the parity bit $K[k_1, k_2, \ldots, k_c]$ to be recovered correctly with a reasonable probability, Matsui demonstrated that the amount of plaintext N needs to be in

the order of $\left|p - \frac{1}{2}\right|^{-2}$. More efficient algorithms for linear cryptanalysis, which find more key bits, are described in [12].

Piling-up Lemma

The first stage in linear cryptanalysis consists in finding useful approximations for a given cipher (or in demonstrating that no useful approximations exist, which is usually much more difficult). Although the most biased linear approximation can easily be found in an exhaustive way for a simple component such as an S-box, a number of practical problems arise when trying to extrapolate this method to full-size ciphers. The first problem concerns the computation of the probability of a linear approximation. In principle, this would require the cryptanalyst to run through all possible combinations of plaintexts and keys, which is clearly infeasible for any practical cipher. The solution to this problem is to make a number of assumptions and to approximate the probability using the so-called *Piling-up Lemma*.

Lemma 1 *Given n independent random variables $X_1, X_2, \ldots, X_n$ taking on values from {0,1}, then the bias $\epsilon = p - 1/2$ of the sum $X = X_1 \oplus X_2 \oplus \ldots \oplus X_n$ is given by:*

$$\epsilon = 2^{n-1} \prod_{j=1}^{n} \epsilon_j, \tag{2}$$

where $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ are the biases of the terms $X_1, X_2, \ldots, X_n$.

Notice that the lemma can be further simplified by defining $c = 2\epsilon$, known as the imbalance or the (correlation) of an expression. With this notation, Eq. 2 reduces to

$$c = \prod_{j=1}^{n} c_j.$$

In order to estimate the probability of a linear approximation using the Piling-up Lemma, the approximation is written as a chain of connected linear approximations, each spanning a small part of the cipher. Such a chain is called a *linear characteristic*. Assuming that the biases of these partial approximations are statistically independent and easy to compute, the total bias can be computed using Eq. 2.

Although the Piling-up Lemma produces very good estimations in many practical cases, even when the approximations are not strictly independent, it should be stressed that unexpected effects can occur when the independence assumption is not fulfilled. In general, the actual bias in these cases can be both much smaller and much larger than predicted by the lemma.

Matsui's Search for the Best Approximations

The Piling-up Lemma in the previous section provides a useful tool to estimate the strength of a given approximation, but the problem remains how to find the strongest approximations for a given cipher. For DES, this open problem was solved by Matsui in 1994 [15]. In his second paper, he proposes a practical search algorithm based on a recursive reasoning. Given the probabilities of the best i-round characteristic with $1 \le i \le n-1$, the algorithm efficiently derives the best characteristic for n rounds. This is done by traversing a tree where branches are cut as soon as it is clear that the probability of a partially constructed approximation cannot possibly exceed some initial estimation of the best n-round characteristic.

Matsui's algorithm can be applied to many other ►block ciphers, but its efficiency varies. In the first place, the running time strongly depends on the accuracy of the initial estimation. Small estimations increase the size of the search tree. On the other hand, if the estimation is too large, the algorithm will not return any characteristic at all. For DES, good estimations can be easily obtained by first performing a restricted search over all characteristics which only cross a single S-box in each round. This does not work as nicely for other ciphers, however. The specific properties of the S-boxes also affect the efficiency of the algorithm. In particular, if the maximum bias of the S-box is attained by many different approximations (as opposed to the distinct peaks in the DES S-boxes), this will slow down the algorithm.

Linear Hulls

Estimating the bias of approximations by constructing linear characteristics is very convenient, but in some cases, the value derived in this way diverges significantly from the actual bias. The most important cause for this difference is the so-called *linear hull* effect, first described by Nyberg in 1994 [16]. The effect takes place when the correlation between plaintext and ciphertext bits, described by a specific linear approximation, can be explained by multiple linear characteristics, each with a non-negligible bias, and each involving a different set of key bits. Such a set of linear characteristics with identical input and output masks is called a *linear hull*. Depending on the value of the key, the different characteristics will interfere constructively or destructively, or even cancel out completely. If the sets of keys used in the different linear characteristics are independent, then this effect might considerably reduce the average bias of expression (1), and thus the success rate of the simple attack described above. Nyberg's paper shows, however, that the more efficient attacks described

in [12], which only use the linear approximations as a distinguisher, will typically benefit from the linear hull effect.

Provable Security Against Linear Cryptanalysis

The existence of a single sufficiently biased linear characteristic suffices for a successful linear attack against a block cipher. A designer's first objective is therefore to ensure that such characteristic cannot possibly exist. This is usually done by choosing highly nonlinear S-boxes and then arguing that the diffusion in the cipher forces all characteristics to cross a sufficiently high minimal number of "active" S-boxes.

The above approach provides good heuristic arguments for the strength of a cipher, but in order to rigorously prove the security against linear cryptanalysis, the designer also needs to take into account more complex phenomena such as the linear hull effect. For DES-like ciphers, such security proofs were studied by Knudsen and Nyberg, first with respect to differential cryptanalysis [17], and then also applied to linear cryptanalysis [16]. The results inspired the design of a number of practical block ciphers such as MISTY (or its variant KASUMI; ►KASUMI/MISTYI), ►Rijndael/AES, ►Camellia, and others. Later, similar proofs were formulated for ciphers based on SP-networks [5, 8].

A somewhat more general theory for provable security against a class of attacks, including basic linear cryptanalysis, is based on the notion of decorrelation, introduced by Vaudenay [23]. The theory suggests constructions were a so-called Decorrelation Module that effectively blocks the propagation of all traditional linear and differential characteristics.

An important remark with respect to the previous notions of provable security, however, is that ciphers which are provably optimal against some restricted class of attacks often tend to be weak when subject to other types of attacks [21, 24].

Comparison with Differential Cryptanalysis

Linear cryptanalysis has many methodological similarities with ►differential cryptanalysis as is noted in [1]. *Differential characteristics* correspond to *linear approximations*. *Difference distribution tables* are replaced by *linear approximation tables*. Concatenation rule for differential characteristics: "match the differences, multiply the probabilities" corresponds to concatenation rule for linear approximations (the piling-up lemma): "match the masks, multiply the imbalances." The algorithms that search for the best characteristic or the best linear approximation are essentially the same. The notion of *differentials* has a corresponding notion of *linear hulls*. Together with striking methodological similarity between the two techniques, there is also *duality* [15] of operations: "XOR branch" and "three-forked branch" are mutually dual regarding their action on differences and masks, respectively. An important distinction between the two methods is that differential cryptanalysis works with blocks of bits, while linear cryptanalysis typically works with a single bit. The bias of the linear approximation has a sign. Thus, given two approximations with the same input and output masks and equal probability but opposite signs, the resulting approximation will have zero bias, due to the cancellation of the two approximations by each other.

Extensions

The linear cryptanalysis technique has received much attention since its invention and has enjoyed several extensions. One technique is a combined ►differential–linear approach proposed by Langford and Hellman. Other extensions include *key-ranking* which allows for a tradeoff between data and time of analysis [6, 14, 19]; *partitioning cryptanalysis* [4] which studies correlation between partitions of the plaintext and ciphertext spaces (no practical cipher has been broken via this technique so far); X^2 *cryptanalysis* [10, 22] has been applied successfully against several ciphers, including round-reduced versions of ►RC6; the use of nonlinear approximations was suggested [11, 20], but so far it provided only small improvements over the linear cryptanalysis. A full nonlinear generalization still remains evasive. The idea to use multiple approximations has been proposed in [7] though the problem of estimating the attacker's gain as well as information extraction from such approximations largely remained opened. In [2] by using a maximal likelihood framework, explicit gain formulas have been derived. Define *capacity* $\overline{c}^2$ of a system of m approximations as $\overline{c}^2 = 4 \cdot \sum_{j=1}^{m} \epsilon_j^2$, where ϵ_j – are the biases of individual approximations. For a fixed attacker's gain over the ►exhaustive search, the data complexity N of the multiple linear attack is proportional to $\overline{c}^2$ – the *capacity* of the "information channel" provided by multiple approximations. The paper also describes several algorithms which provide such gains. A conversion of a known plaintext linear attack to a chosen plaintext linear attack has been proposed in [9]. Finally note that similar techniques have been applied to stream ciphers (►Linear Cryptanalysis for Stream Ciphers).

Recommended Reading

1. Biham E (1995) On Matsui's linear cryptanalysis. In: De Santis A (ed) Advances in cryptology – eurocryrt'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 341–355
2. Biryukov A, De Cannière C, Quisquater M (2004) On multiple linear approximations In: Franklin M (ed) Advances in cryptology, proceedings of crypto 2004. Lecture notes in computer science, vol 3152. Springer, pp 1–22
3. Desmedt Y (ed) (1994). In: Desmedt YG (ed) Advances in cryptology – crypto'94. Lecture notes in computer science, vol 839. Springer, Berlin
4. Harpes C, Massey JL (1997) Partitioning cryptanalysis. In: Biham E (1997) Fast software encryption, FSE'97. Lecture notes in computer science, vol 1267. Springer, Berlin, pp 13–27
5. Hong S, Lee S, Lim J, Sung J, Cheon D, Cho I (2000) Provable security against differential and linear cryptanalysis for the SPN structure. In: Schneier B (ed) Proceedings of fast software encryption – FSE 2000. Lecture notes in computer science, vol 1978. Springer-Verlag, Berlin, pp 273–283
6. Junod P, Vaudenay S (2003) Optimal key ranking procedures in a statistical cryptanalysis. In: Johansson T (ed) Fast software encryption, FSE 2003. Lecture notes in computer science, vol 2887. Springer, Berlin, pp 1–15
7. Kaliski BS, Robshaw MJ (1994) Linear cryptanalysis using multiple approximations. In: Desmedt Y (ed) Advances in cryptography – crypto'94. Lecture notes in computer science, vol 839. Springer, Berlin, pp 26–39
8. Keliher L, Meijer H, Tavares SE (2001) New method for upper bounding the maximum average linear hull probability for SPNs. In: Pfitzmann B (ed) eurocrypt 2001. Lecture notes in computer science, vol 2045. Springer, Berlin, pp 420–436
9. Knudsen LR, Mathiassen JE (2001) A chosen-plaintext linear attack on DES. In: Schneier B (ed) Fast software encryption, FSE 2000. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 262–272
10. Knudsen LR, Meier W (2000) Correlations in RC6 with a reduced number of rounds. In: Schneier B (ed) Proceedings of fast software encryption – FSE 2000. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 94–108
11. Knudsen LR, Robshaw MJB (1996) Non-linear approximations in linear cryptanalysis. In: Maurer U (ed) Advances in cryptology – eurocrypt'96. Lecture notes in computer science, vol 1070. Springer, Berlin, pp 224–236
12. Matsui M (1993) Linear cryptanalysis method for DES cipher. In: Helleseth T (ed) Advances in cryptology – eurocrypt'93. Lecture notes in computer science, vol 765. Springer, Berlin, pp 386–397
13. Matsui M, Yamagishi A (1993) A new method for known plaintext attack of FEAL cipher. In: Rueppel RA (ed) Advances in cryptography – eurocrypt'92. Lecture notes in computer science, vol 658. Springer, Berlin, pp 81–91
14. Matsui M (1994) The first experimental cryptanalysis of the data encryption standard. In: Desmedt YG (ed) Advances in cryptography – crypto'94. Lecture notes in computer science, vol 839. Springer, Berlin, pp 1–11
15. Matsui M () On correlation between the order of S-boxes and the strength of DES. In: De Santis S (ed) Advances in cryptology – eurocrypt'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 366–375
16. Nyberg K (1994) Linear approximations of block ciphers. In: De Santis (ed) Advances in cryptography – eurocrypt'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 439–444
17. Nyberg K, Knudsen LR (1995) Provable security against a differential attack. J Cryptol 8(1):27–38
18. Santis AD (ed) (1995). In: De Santis A (ed) Advances in cryptology – eurocrypt'94. Lecture notes in computer science, vol 950. Springer, Berlin
19. Selcuk AA (2002) On probability of success in differential and linear cryptanalysis. Technical report, network systems lab, department of computer science, Purdue University, 2002. Previously published at SCN 2002
20. Shimoyama T, Kaneko T (1998) Quadratic relation of S-box and its application to the linear attack of full round des. In: Krawczyk H (ed) Advances in cryptology – crypto'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 200–211
21. Shimoyama T, Moriai S, Kaneko T, Tsujii S (1999) Improved higher order differential attack and its application to Nyberg-Knudsen's designed block cipher. IEICE Trans Fundament E82-A(9):1971–1980 http://search.ieice.or.jp/1999/files/e000a09.htm#e82-a,9,1971
22. Vaudenay S (1996) On the weak keys of blowfish. In: Gollmann D (ed) Fast software encryption, FSE'96. Lecture notes in computer science, vol 1039. Springer, Berlin, pp 27–32
23. Vaudenay S (2003) Decorrelation: a theory for block cipher security. Journal of Cryptology 16(4):249–286
24. Wagner D (1999) The boomerang attack. In: Knudsen LR (ed) Fast software encryption, FSE'99. Lecture notes in computer science, vol 1636. Springer, Berlin, pp 156–170

Linear Cryptanalysis for Stream Ciphers

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

Related Concepts

►Fast Correlation Attack; ►Linear Cryptanalysis; ►Stream Cipher

Definition

Linear cryptanalysis for stream ciphers relies on the same basic principles as the ►linear cryptanalysis for block ciphers introduced by Matsui. It exploits the existence of biased linear relations between some keystream bits and some key bits. The linear cryptanalysis provides a ►known plaintext attack on various ►stream ciphers, which allows to distinguish the keystream from a truly random sequence. Such a *distinguishing attack* can be used for reducing the uncertainty of unknown plaintexts, or for recovering the unknown structure of the keystream generator. It may also be extended to a key-recovery attack in some cases. It might be mounted in the context of a ►resynchronization attack, when several keystream segments corresponding to different initial values are available to the attacker.

Background

In the context of stream ciphers, linear cryptanalysis is a terminology introduced by Golić in 1994 [7]. However, linear attacks against stream ciphers were known before the introduction of linear cryptanalysis by Matsui: for instance, the ►correlation attack on the ►combination generator presented by Siegenthaler in 1985 [9] exploits a biased linear relation between the keystream and the bits of the initial state of a constituent register.

Theory

The linear cryptanalysis consists in finding some linear functions of the keystream bits which are not *balanced*, i.e., which are not uniformly distributed. Such linear correlations are used for distinguishing the keystream sequence from a random sequence by a classical statistical test.

Biased linear relations are usually found by replacing the nonlinear components in the cipher by appropriate linear approximations. General methods for exhibiting such relations include the ►correlation attack [9] against the ►combination generator and on the ►filter generator, the *linear sequential circuit approximation* due to Golić [6, 7] and some variants used in [2–4] against several LFSR-based generators.

Linear cryptanalysis has led to successful attacks on several stream ciphers, including SOBER [4], SNOW [3, 8, 10], ►E0 [5], and the original version of Grain [1].

Recommended Reading

1. Berbain C, Gilbert H, Maximov A (2006) Cryptanalysis of Grain. In: Fast software encryption – FSE 2006. Lecture notes in computer science, vol 4047. Springer, Berlin, pp 15–29
2. Canteaut A, Filiol E (2001) Ciphertext only reconstruction of stream ciphers based on combination generators. In: Fast software encryption – FSE 2000. Lecture notes in computer science, vol 1978. Springer, Berlin, pp 165–180
3. Coppersmith D, Halevi S, Jutla C (2002) Cryptanalysis of stream ciphers with linear masking. In: Advances in cryptology – CRYPTO 2002. Lecture notes in computer science, vol 2442. Springer, Berlin, pp 515–532
4. Ekdahl P, Johansson T (2002) Distinguishing attacks on SOBER-t16 and t32. In: Fast software encryption – FSE 2002. Lecture notes in computer science, vol 2365. Springer, Berlin, pp 210–224
5. Golić JDj, Bagini V, Morgari G (2002) Linear cryptanalysis of Bluetooth stream cipher. In: Advances in cryptology – EUROCRYPT 2002. Lecture notes in computer science, vol 2332. Springer, Berlin, pp 238–255
6. Golić JDj (1992) Correlation via linear sequential circuit approximation of combiners with memory. In: Advances in cryptology – EUROCRYPT'92. Lecture notes in computer science, vol 658. Springer, Berlin, pp 113–123
7. Golić JDj (1994) Linear cryptanalysis of stream ciphers. In: Fast software encryption – FSE'94. Lecture notes in computer science, vol 1008. Springer, Berlin, pp 154–169
8. Nyberg K, Wallén J (2006) Improved linear distinguishers for SNOW 2.0. In: Fast software encryption – FSE 2006. Lecture notes in computer science, vol 4047. Springer, Berlin, pp 144–162
9. Siegenthaler T (1985) Decrypting a class of stream ciphers using ciphertext only. IEEE Trans Comput C-34(1):81–84
10. Watanabe D, Biryukov A, De Cannière C (2003) A distinguishing attack of SNOW 2.0 with linear masking method. In: Selected areas in cryptography – SAC 2003. Lecture notes in computer science, vol 3006. Springer, Berlin, pp 222–233

Linear Feedback Shift Register

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

Synonyms

LFSR

Related Concepts

►Berlekamp–Massey Algorithm; ►Combination Generator; ►Filter Generator; ►Linear Complexity; ►Minimal Polynomial; ►Stream Cipher

Definition

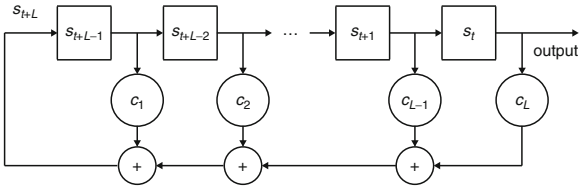
Linear Feedback Shift Registers (LFSRs) are the basic components of many ►running-key generators for ►stream cipher applications, because they are appropriate to hardware implementation and they produce sequences with good statistical properties. LFSR refers to a feedback shift register with a linear feedback function (►Nonlinear Feedback Shift Register).

An LFSR of length L over $\mathbf{F}_q$ is a finite state automaton which produces a semi-infinite sequence of elements of $\mathbf{F}_q$, $\mathbf{s} = (s_t)_{t\geq 0} = s_0 s_1 \ldots$, satisfying a linear recurrence relation of degree L over $\mathbf{F}_q$

$$s_{t+L} = \sum_{i=1}^{L} c_i s_{t+L-i}, \qquad \forall t \geq 0.$$

The L coefficients $c_1, \ldots, c_L$ are elements of $\mathbf{F}_q$. They are called the *feedback coefficients* of the LFSR.

An LFSR of length L over $\mathbf{F}_q$ has the following form:



The register consists of L delay cells, called *stages*, each containing an element of $\mathbf{F}_q$. The contents of the L stages, $s_t, \ldots, s_{t+L-1}$, form the *state* of the LFSR. The L stages are initially loaded with L elements, $s_0, \ldots, s_{L-1}$, which can be arbitrary chosen in $\mathbf{F}_q$; they form the *initial state* of the register.

The shift register is controlled by an external clock. At each time unit, each digit is shifted one stage to the right. The content of the rightmost stage s_t is output. The new content of the leftmost stage is the *feedback bit*, s_{t+L}. It is obtained by a linear combination of the contents of the register stages, where the coefficients of the linear combination are given by the feedback coefficients of the LFSR:

$$s_{t+L} = \sum_{i=1}^{L} c_i s_{t+L-i}.$$

Therefore, the LFSR implements the linear recurrence relation of degree L:

$$s_{t+L} = \sum_{i=1}^{L} c_i s_{t+L-i}, \qquad \forall t \geq 0.$$

Example. Table 1 gives the successive states of the binary LFSR of length 4 with feedback coefficients $c_1 = c_2 = 0$, $c_3 = c_4 = 1$ and with initial state $(s_0, s_1, s_2, s_3) = (1, 0, 1, 1)$. This LFSR is depicted in Fig. 1. It corresponds to the linear recurrence relation

$$s_{t+4} = s_{t+1} + s_t \bmod 2.$$

The output sequence $s_0 s_1 \ldots$ generated by this LFSR is $1011100\ldots$.

Theory

Feedback polynomial and characteristic polynomial. The output sequence of an LFSR is uniquely determined by its feedback coefficients and its initial state. The feedback coefficients $c_1, \ldots, c_L$ of an LFSR of length L are usually represented by the LFSR *feedback polynomial* (or *connection polynomial*) defined by

$$P(X) = 1 - \sum_{i=1}^{L} c_i X^i.$$

Alternatively, one can use the *characteristic polynomial* [3], which is the reciprocal polynomial of the feedback polynomial:

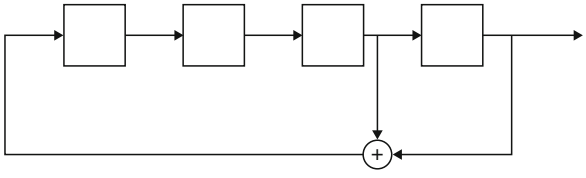
$$P^\star(X) = X^L P(1/X) = X^L - \sum_{i=1}^{L} c_i X^{L-i}.$$

For instance, the feedback polynomial of the binary LFSR shown in Fig. 1 is $P(X) = 1 + X^3 + X^4$ and its characteristic polynomial is $P^\star(X) = 1 + X + X^4$.

An LFSR is said to be *non-singular* if the degree of its feedback polynomial is equal to the LFSR length (i.e., if the feedback coefficient c_L differs from 0). Any sequence generated by a non-singular LFSR of length L is periodic, and its period does not exceed $q^L - 1$. Indeed, the LFSR has at most q^L different states and the all-zero state is always followed by the all-zero state. Moreover, if the LFSR is singular, all generated sequences are *ultimately periodic*, that is, the sequences obtained by ignoring a certain number of elements at the beginning are periodic [2].

Characterization of LFSR output sequences. A given LFSR of length L over $\mathbf{F}_q$ can generate q^L different sequences corresponding to the q^L different initial states and these sequences form a vector space over $\mathbf{F}_q$. The set of all sequences generated by an LFSR with feedback polynomial P is characterized by the following property: a sequence $(s_t)_{t\geq 0}$ is generated by an LFSR of length L over $\mathbf{F}_q$ with feedback polynomial P if and only if there exists a polynomial $Q \in \mathbf{F}_q[X]$ with $\deg(Q) < L$ such that the generating function of $(s_t)_{t\geq 0}$ satisfies

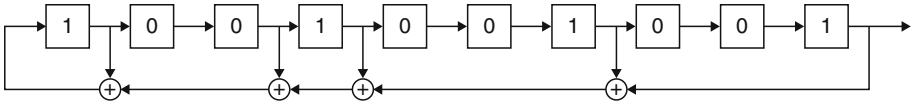
$$\sum_{t\geq 0} s_t X^t = \frac{Q(X)}{P(X)}.$$



Linear Feedback Shift Register. Fig. 1 Binary LFSR with feedback coefficients $(c_1, c_2, c_3, c_4) = (0, 0, 1, 1)$

Linear Feedback Shift Register. Table 1 Successive states of the LFSR with feedback coefficients $(c_1, c_2, c_3, c_4) = (0, 0, 1, 1)$ and with initial state $(s_0, s_1, s_2, s_3) = (1, 0, 1, 1)$

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s_t	1	0	1	1	1	1	0	0	0	1	0	0	1	1	0	1
s_{t+1}	0	1	1	1	1	0	0	0	1	0	0	1	1	0	1	0
s_{t+2}	1	1	1	1	0	0	0	1	0	0	1	1	0	1	0	1
s_{t+3}	1	1	1	0	0	0	1	0	0	1	1	0	1	0	1	1



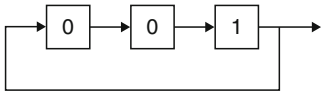
Linear Feedback Shift Register. Fig. 2 Example of a LFSR of length 10

Moreover, the polynomial Q is completely determined by the coefficients of P and by the initial state of the LFSR:

$$Q(X) = -\sum_{i=0}^{L-1} X^i \left(\sum_{j=0}^{i} c_{i-j} s_j \right),$$

where $P(X) = \sum_{i=0}^{L} c_i X^i$. This result, which is called the fundamental identity of formal power series of linear recurring sequences, means that there is a one-to-one correspondence between the sequences generated by an LFSR of length L with feedback polynomial P and the fractions $Q(X)/P(X)$ with $\deg(Q) < L$. It has two major consequences. On the first hand, any sequence generated by an LFSR with feedback polynomial P is also generated by any LFSR whose feedback polynomial is a multiple of P. This property is used in some attacks on keystream generators based on LFSRs (►Fast Correlation attack). On the other hand, a sequence generated by an LFSR with feedback polynomial P is also generated by a shorter LFSR with feedback polynomial P' if the corresponding fraction $Q(X)/P(X)$ is such that $\gcd(P, Q) \neq 1$. Thus, amongst all sequences generated by the LFSR with feedback polynomial P, there is one which can be generated by a shorter LFSR if and only if P is not ►irreducible over $\mathbf{F}_q$.

Moreover, for any linear recurring sequence $(s_t)_{t\geq 0}$, there exists a unique polynomial P_0 with constant term equal to 1, such that the generating function of $(s_t)_{t\geq 0}$ is given by $Q_0(X)/P_0(X)$, where P_0 and Q_0 are relatively prime. Then, the shortest LFSR which generates $(s_t)_{t\geq 0}$ has length $L = \max(\deg(P_0), \deg(Q_0) + 1)$, and its feedback polynomial is equal to P_0. The reciprocal polynomial of P_0, $X^L P_0(1/X)$, is the characteristic polynomial of the shortest LFSR which generates $(s_t)_{t\geq 0}$; it is called the ►minimal polynomial of the sequence. It determines the linear recurrence relation of least degree satisfied by the sequence. The degree of the minimal polynomial of a linear recurring sequence is the ►linear complexity of the sequence. It corresponds to the length of the shortest LFSR which generates it. The minimal polynomial of a sequence $\mathbf{s} = (s_t)_{t\geq 0}$ of linear complexity $\Lambda(\mathbf{s})$ can be determined from the knowledge of at least $2\Lambda(\mathbf{s})$ consecutive bits of $\mathbf{s}$ by the ►Berlekamp–Massey algorithm.



Linear Feedback Shift Register. Fig. 3 LFSR of length 3 which generates the same sequence as the LFSR of Fig. 2

Example. The binary LFSR of length 10 depicted in Fig. 2 has feedback polynomial

$$P(X) = 1 + X + X^3 + X^4 + X^7 + X^{10},$$

and its initial state $s_0 \ldots s_9$ is 1001001001.

The generating function of the sequence produced by this LFSR is given by

$$\sum_{t\geq 0} s_t X^t = \frac{Q(X)}{P(X)}$$

where Q is deduced from the coefficients of P and from the initial state:

$$Q(X) = 1 + X + X^7.$$

Therefore, we have

$$\sum_{t\geq 0} s_t X^t = \frac{1 + X + X^7}{1 + X + X^3 + X^4 + X^7 + X^{10}} = \frac{1}{1 + X^3},$$

since $1 + X + X^3 + X^4 + X^7 + X^{10} = (1 + X + X^7)(1 + X^3)$ in $\mathbf{F}_2[X]$. This implies that $(s_t)_{t\geq 0}$ is also generated by the LFSR with feedback polynomial $P_0(X) = 1 + X^3$ depicted in Fig. 3. The minimal polynomial of the sequence is then $1 + X^3$ and its linear complexity is equal to 3.

Period of an LFSR sequence. The minimal polynomial of a linear recurring sequence plays a major role since it completely determines the linear complexity and the least period of the sequence. Actually, the least period of a linear recurring sequence is equal to the period of its minimal polynomial. The *period* (also called the *order*) of a polynomial P in $\mathbf{F}_q[X]$, where $P(0) \neq 0$, is the least positive integer e for which $P(X)$ divides $X^e - 1$. Then, $\mathbf{s}$ has maximal period $q^{\Lambda(\mathbf{s})} - 1$ if and only if its minimal polynomial is a primitive polynomial (i.e., if the period of its minimal polynomial is maximal). For instance, the sequence generated by the LFSR shown in Fig. 3 has period 3 because its minimal polynomial $1 + X^3$ has period 3. This sequence is 100100100 . . .. On the other hand, any nonzero sequence

generated by the LFSR of length 4 depicted in Fig. 1 has period $2^4 - 1 = 15$. Actually, the minimal polynomial of any such sequence corresponds to its characteristic polynomial $P^\star(X) = 1+X+X^4$, because $P^\star$ is irreducible. Moreover, $P^\star$ is a primitive polynomial. Any sequence $\mathbf{s} = (s_t)_{t\geq 0}$ generated by an LFSR of length L which has a primitive feedback polynomial has the highest possible linear complexity $\Lambda(\mathbf{s}) = L$ and the highest possible period $q^L - 1$. Such sequences are called ►maximal-length linear sequences (m-sequences). Because of the previous optimal properties, the linear recurring sequences used in cryptography are always chosen to be m-sequences. Moreover, they possess good statistical properties [1] (►maximal-length linear sequences for further details). In other terms, the feedback polynomial of a LFSR should always be chosen to be a primitive polynomial.

Keystream generators based on LFSRs. It is clear that an LFSR should never be used by itself as a keystream generator. If the feedback coefficients of the LFSR are public, the entire keystream can obviously be recovered from the knowledge of any Λ consecutive bits of the keystream, where Λ is the linear complexity of the running-key (which does not exceed the LFSR length). If the feedback coefficients are kept secret, the entire keystream can be recovered from any 2Λ consecutive bits of the keystream by the ►Berlekamp–Massey algorithm. Therefore, a commonly used technique to produce a pseudorandom sequence which can be used as a running-key is to combine several LFSRs in different ways in order to generate a linear recurring sequence which has a high linear complexity (e.g., ►combination generator, ►filter generator...).

Recommended Reading

1. Golomb SW (1982) Shift register sequences. Revised edition, Aegean Park Press, Laguna Hills, CA
2. Lidl R, Niederreiter H (1983) Finite fields. Cambridge University Press, Cambridge
3. Rueppel RA (1986) Analysis and design of stream ciphers. Springer-Verlag, New York

Linear Syndrome Attack

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

Related Concepts

►Fast Correlation Attack; ►Stream Cipher

Definition

The *linear syndrome attack* is an attack on ►LFSR-based keystream generators, which was presented by Zeng and Huang in 1988 [1] (see also [3]). It is a weak version of the ►fast correlation attack, which was independently proposed by Meier and Staffelbach [2].

Recommended Reading

1. Meier W, Staffelbach O (1988) Fast correlation attacks on stream ciphers. In: Advances in cryptology – EUROCRYPT'88. Lecture notes in computer science, vol 330. Springer, Berlin, pp 301–314
2. Zeng K, Huang M (1988) On the linear syndrome method in cryptanalysis. In: Advances in cryptology – CRYPTO'88. Lecture notes in computer science, vol 403. Springer, Berlin, pp 469–478
3. Zeng K, Yang CH, Rao TRN (1990) An improved linear syndrome algorithm in cryptanalysis with applications. In: Advances in cryptology – CRYPTO'90. Lecture notes in computer science, vol 537. Springer, Berlin, pp 34–47

List Decoding

►Decoding Algorithms

Location Information (Privacy of)

Claudio A. Ardagna
Dipartimento di Tecnologie dell'Informazione (DTI), Università degli Studi di Milano, Crema (CR), Italy

Synonyms

Location privacy

Related Concepts

►Anonymity

Definition

Location information ►privacy is the right of mobile individuals to decide how, when, and for which purposes their location information could be released to and managed by other parties.

Background

The rapid growth of mobile technologies and the widespread adoption of mobile communication devices have fostered the development of new applications that exploit the physical position of the users to offer Location-Based Services (LBSs) for business, social, or informational purposes. Today, several commercial and

enterprise-oriented LBSs are already available and are gaining popularity. In general, LBSs can be partitioned into the following categories [1].

- *Locate-me services.* They provide information about the position of the users. They should be used when authorized third parties need to know the position of the users for performing their tasks. A locate-me service is at the basis of all the others LBS categories.
- *Nearby-information services.* They provide information about the environment surrounding the location of a user (e.g., point of interest, context-aware tourist guides, or weather and traffic alerts). A user subscribes to these services and receives real-time information through her mobile device.
- *Locate-friends and nearby-friends services.* They provide information to subscribers about the real-time location or proximity of other subscribers. They can be used, for example, to provide services in the context of social networks or as industrial applications to coordinate workforces.
- *Tracking services.* They allow monitoring movements of the users and include telemetric services (i.e., the observation of parameters of mobile objects such as speed, direction of movement, and so on). They can be used by online services that provide tracking of children, employees, or vehicles, and warning about dangerous areas.
- *Personal-navigation services.* They provide information about the path that has to be followed to reach a target location from the current location of the user. These services rely on tracking services to gather the position of a user moving on the field.

While these applications offer great benefits to the users, they also exhibit significant potential for privacy abuses since positioning and tracking systems are collecting a huge amount of location information. Recent security incidents have revealed faulty data management practices and unauthorized trading of personal (including location) information of the users. In this scenario, the improper exposure of location information could result in abuses, such as stalking or physical harassment.

Theory

Geolocation solutions measure the position of mobile devices by using several mobile technologies (e.g., GSM/3G, GPS, WiFi) that have been developed and can be exploited to compute location information. The boost in terms of accuracy and reliability enjoyed by geolocation solutions in the recent years and the widespread adoption of GSM/3G, GPS, and WiFi devices (e.g., cellular phones, laptops, PDAs) enable the delivery of services that use the physical locations of the users and call for an urgent and careful consideration of privacy issues. Privacy concerns become more critical since mobile devices are unable to enforce restrictions on the location data scattering or to avoid the data flow (unless the mobile devices are switched off). The worst-case scenario that some analysts have foreseen as a consequence of an unrestricted and unregulated availability of location technologies recalls the well-known "Big Brother" stereotype: a society where the secondary effect of location technologies (whose primary effect is to enable the development of innovative and useful services) is a form of implicit total surveillance of individuals.

►Location privacy can be defined as the right of individuals to decide how, when, and for which purposes their location information could be released to or managed by other parties. The lack of location privacy protection could be exploited by adversaries and result in different types of attacks: *unsolicited advertising*, when the location of a user could be exploited, without her consent, to provide advertisements of products and services available nearby the user position; *physical attacks or harassment*, when the location of a user could allow criminals to carry physical assaults to specific individuals; *users profiling*, when the location of a user could be used to infer other sensitive information, such as state of health, personal habits, or professional duties, by correlating visited places or paths; *denial of service*, when the location of a user could motivate an access denial to services under some circumstances.

The concept of location privacy can assume several meanings and pursue different objectives, depending on the scenario in which users are moving and on the services users are interacting with. Location privacy solutions can be aimed at protecting users by making their location information anonymous or keeping explicit identification, but perturbing their location information to decrease the accuracy. Different categories of location privacy can then be defined [1].

- *Identity privacy.* The main goal is to protect users' identities associated with or inferable from location information. In this case, accurate location measurements can be provided to location-based services, but the identity of the users must be kept hidden.
- *Position privacy.* The main goal is to perturb the location of the users as a way to protect their actual position. In particular, this type of location privacy is suitable when users' identities are required for the successful provisioning of a service.

- *Path privacy.* The main goal is to protect the privacy of those users that are continuously monitored during a certain period of time. In this case, location-based services will no longer receive a single location measurement, rather they will gather a flow of position samples that permit them to track the users and to infer sensitive areas they have visited.

Based on the above categories, three main classes of location privacy techniques have been introduced: anonymity-based, obfuscation-based, and policy-based. *Anonymity-based* techniques provide a class of solutions for the protection of identity and path privacy. In particular, this class includes all solutions based on the notion of ►anonymity, which is aimed at making an individual (i.e., her identity or personal information) not identifiable. Anonymity-based techniques [1, 3] are suitable for all those contexts that do not need knowledge of the identity of the users, and their effectiveness depends on the number of users physically located in the same area. *Obfuscation-based* techniques provide a class of solutions that perturb the location information still maintaining a binding with the identity of the users. Obfuscation degrades the accuracy of the location information to provide privacy protection. Obfuscation-based techniques [1, 2] are suitable for all those contexts that need knowledge of the identity of the users. Finally, *policy-based* techniques provide a class of solutions based on the definition of privacy policies and for the protection of all privacy categories.

In general, anonymity-based and obfuscation-based techniques are dual categories. While anonymity-based techniques have been primarily defined to protect identity privacy and are less suitable for protecting position privacy, obfuscation-based techniques are well suited for position privacy protection and unrelated with identity privacy protection. As for path privacy, both anonymity-based and obfuscation-based techniques are well suited and able to provide the required degree of protection. Policy-based techniques are flexible and in general well suited for all location privacy categories, whereas their management complexity could easily become overwhelming for the users.

Applications

Many mobile network providers offer a variety of location-based services, such as point of interest proximity, friend-finder, or location information transfer in case of an accident (e.g., 911 emergency service). Such services naturally raise privacy concerns. Users consider their physical location and movements as highly privacy sensitive, and demand for solutions able to protect such an information in a variety of environments. Also, although privacy is currently seen as an optional add-on by the LBS providers, in the near future, it will represent one of the key aspects to the success of the LBSs and a fundamental parameter in their selection by the users. In this context, solutions for the protection of location information privacy might be integrated with LBSs to protect the privacy of the users, still preserving the overall quality of the services.

Open Problems and Future Directions

Some open problems and interesting research directions that need to be tackled by future research in the context of location information privacy are as follows.

- *Untrusted mobile network operator.* Current approaches usually assume untrusted location-based services, while they consider the mobile network operator as a trusted powerful entity able to know and observe all the traffic in the network. All the requests and responses in a communication are mediated by the network operator that knows, for all its users, which users access which servers. In other words, it can reconstruct exactly all the pairs $\langle user,LBS \rangle$ describing communications of its users. A critical problem, and an interesting future research direction, is to provide a means for users to communicate with LBSs without giving the operator the ability to observe the communication profiles. The mobile operator, while considered trustworthy with respect to the availability and working of the network, should be restricted in terms of the view and traffic it can reconstruct.
- *Path protection.* Future work should extend current solutions to better protect the privacy of the users that are monitored during a certain period of time. This research area is particularly relevant given the ever-increasing interest in offering applications for tracking users. Data about users moving in a particular area are collected by external services that use them to provide their services effectively. In such a scenario, the need for privacy techniques aimed at protecting path privacy becomes urgent.
- *Map constraints.* Current privacy solutions do not consider map constraints as a way to better protect the location privacy of the users. Topological information, however, could help adversaries in reducing location privacy by guessing identity of users and by producing more accurate location information. An interesting research direction is to enrich existing privacy techniques with Geographical Information System (GIS) maps, providing more robust solutions that take advantage from map information.

Recommended Reading

1. Ardagna CA, Cremonini M, Damiani E, De Capitani di Vimercati S, Samarati P (2007) Privacy-enhanced location services information. In: Acquisti A, De Capitani di Vimercati S, Gritzalis S, Lambrinoudakis C (eds) Digital privacy: theory, technologies and practices. Auerbach (Taylor and Francis Group), New York
2. Ardagna CA, Cremonini M, De Capitani di Vimercati S, Samarati P (2011) An obfuscation-based approach for protecting location privacy. IEEE Transactions on Dependable and Secure Computing, 8(1):13–27, January-March 2011
3. Ardagna CA, Jajodia S, Samarati P, Stavrou A (2010) Providing mobile users' anonymity in hybrid networks. In: Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS 2010), Athens, Greece, September 2010

Location Privacy

►Location Information (Privacy of)

Location Privacy in Wireless Networks

Marco Gruteser
Wireless Information Network Laboratory, Department of Electrical and Computer Engineering, Rutgers University, North Brunswick, NJ, USA

Synonyms

Wireless locational privacy

Related Concepts

►Access Control; ►Anonymity; ►Entropy

Definition

The ability of a user or owner of a wireless device to control to which party, to what degree, and at what times information about the device's geographic location is revealed.

Background

This definition of location privacy is derived from the more general concept of information privacy. It is commonly characterized as the claim for informational self-determination, originally defined by Alan Westin as "the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others" [1]. Other more restrictive definitions have also been proposed, for example, "the ability [...] to move through public space with the expectation that [...] location will not be systematically and secretly recorded for later use" [2]. While some location information is also available on wired networks, location privacy is particularly relevant in wireless networks due to user's significantly higher degree of mobility. Examples of the broad range of wireless location privacy concerns are:

- A cellular phone user may want to make a phone call without the phone's position recorded by the cellular phone network.
- The owner of a sensor network deployed for target tracking in a hostile environment may want to conceal the location of sensors, the location of detected events, and the location of the data sinks which collect information.
- The user of an automotive navigation service may want to share location traces for traffic congestion monitoring without revealing identity and exact places visited.

These examples illustrate that in different situations device owners may want to control the release of location information with respect to wireless service providers, cellular service providers, application service providers, or eavesdroppers on the wireless channel. A common challenge is that location information is often implicitly revealed by network usage or activity. For example, the cellular base station through which a phone call originates reveals the approximate location of the caller.

The potential risks associated with uncontrolled revealing of location information were enumerated in the Location Privacy Protection Act of 2001 considered in the United States Congress [3]. Examples are the drawing of inferences about user's medical condition, nightlife, or political activities from the places users visit.

Theory

A breach of location privacy requires that personally identifiable information is revealed with the location data, that is, the data can be uniquely linked to an individual person. Such personally identifiable information can take many forms, such as names and addresses of persons or device/network identifiers (e.g., the Global System for Mobile Communications International Mobile Subscriber Identifier or an Internet Protocol address), which can be easily linked to a person. It is also possible, however, that

the location information itself can be linked to individual persons and thus can be considered personally identifiable information. In actual usage, often *some* information about location or identity is revealed, implying that location privacy should be understood in terms of a degree of privacy, rather than absolute privacy. The definition of metrics for this degree of location privacy remains an active area of research.

This understanding gives rise to two approaches for the design of privacy-enhancing technologies: controlling the release of personally identifiable data or filtering out personally identifiable information to render the data truly anonymous.

The first approach resembles access control and the main challenge is designing usable mechanisms. Standard web-tools such as privacy policy preference mechanisms that can provide guidance to users and make automatic decisions are also applicable to wireless applications. One common difference of access control mechanisms specifically developed for location information, however, is that the rules governing access frequently depend on location and time of access. They also differ, in that they may reduce the fidelity of location information, for example, providing only city-level location data although a precise GPS location is available.

The second approach allows sharing of some location information without revealing personally identifiable records and is referred to as anonymization [4]. In this model, a set of location records L are filtered or perturbed by an anonymizer A, resulting in an anonymous dataset $L_a = A(L)$, which can be shared with others. It is assumed that the adversary, say Eve, obtains access to L_a and seeks to reidentify users whose records are contained in L_a.

In one model focused on tracking, $L_a = (l_1, l_2, \ldots, l_k)$ with each l_i = (lat, lon, time, heading, speed) from one of m users U. The dataset does not identify which location record l_i belongs to which user u_j in U and does not identify for any two location updates l_i, l_j whether they were generated by the same user (i.e., the order in the dataset does not allow any such conclusion). Eve has an external location dataset O with location observations or restricted spaces for some of the users, where each observation has the form (u_j, l_i). The adversary also has a model of human movement that allows the adversary to assign a likelihood and can be used to reconstruct paths of individual users from the location dataset (i.e., link two location updates l_i, l_j in L_a to the same user). The adversary can compromise location privacy if any of the observations in O can be uniquely linked to a location record in L to reidentify a location record and the adversary can learn significant additional information about the user's travels from L (beyond what is already known in O). A privacy metric that quantifies this additional information is the time-to-confusion metric [5]. Intuitively, it measures the duration that an adversary can follow a user in the dataset L_a from the observation where the user was reidentified. Strong anonymization requires that time to confusion is bounded and small. Techniques for stronger anonymization include path cloaking [5] and mix zones [6].

In a related model [7] also considering query privacy, $L_a = (l_1, l_2, \ldots, l_k)$ but with each l_i = (query, cloaking area, time) from one of m users U. Again the adversary has a dataset of location observations, but the adversary should not be able to reidentify any of the location records, since this would compromise query privacy in addition to the tracking concerns described above. The anonymizer can replace the exact location coordinates with a cloaking area, that is, an uncertainty region for the user location, to satisfy the k-anonymity criterion.

Applications

Typical designs of wireless communication networks generate personally identifiable information at many levels of the network stack. At the lower levels of the network stack any message transmitted can be localized and generate a location record. Users can therefore only control the release of location information toward the service provider by controlling where and when they communicate (and deactivate the device at other times) or by using specialized privacy-enhancing technologies to mask location or identifiers (e.g., identifier-free protocols or protocols that frequently change network and devices identifiers).

At the physical layer, radio waveforms carry a fingerprint of the analog radio front end that transmitted the signal. On a set of identical wireless LAN radios, modulation level characteristics such as center frequency offset have been shown to allow distinguish one out of more than 100 transmitters. Any signal transmitted can also be located using location fingerprints (of the signal), or techniques based on triangulation and trilateration. Privacy-enhancing techniques at this layer include reducing the frequency of transmissions, and decreasing transmission power (or increasing directionality) to reduce the chance of signal observation and localization.

At the medium access layer, devices usually carry a unique address and their approximate location can be determined based on proximity to the wireless access point or cell tower that they associate with. A key privacy-enhancing technique at the medium access control layer is

the frequent switching of medium access control addresses to reduce the chance of identification and tracking of the device [8].

At the network layer, Internet protocol addresses act as an identifier that sometimes can be linked to an individual and they also describe the location of the device in the network topology. This topological location can often be mapped to geographic location with about city-level granularity. The routes on which packets are sent can also reveal information about the approximate location of the source or destination node [9]. Privacy-enhancing technologies at the network layer include route randomization, onion routing, and reducing the granularity of location information maintained deep inside the network.

Finally, location-based applications, which are frequently used over wireless networks, can collect location information together with application-level identifiers. Location is frequently obtained from Global Positioning System receivers, which can make it especially precise.

While policy-based access control and anonymity are most commonly used at the application layer, these techniques can be adapted for use at all layers of the network stack. For example, the time-to-confusion criterion can be used to evaluate tracking risks from several wireless messages.

Recommended Reading

1. Westin A (1967) Privacy and freedom. Atheneum, New York
2. Blumberg A, Eckersley P (2009) On locational privacy, and how to avoid losing it forever, Electronic Frontier Foundation Whitepaper, August 2009
3. S1164, Location Privacy Protection Act of 2001. United States congressional record
4. Krumm J (2009) A survey of computational location privacy. Pers Ubiquit Comput 13(6):391–399
5. Hoh B, Gruteser M, Xiong H, Alrabady A (2007) Preserving privacy in GPS traces via density-aware path cloaking. In: Proceedings of the 14th ACM conference on Computer and communications security (CCS), Alexandria, 2007
6. Beresford AR, Stajano F (2003) Location privacy in pervasive computing. IEEE Pervasive Comput 2(1):46–55
7. Gruteser M, Grunwald D (2003) Anonymous usage of location-based services through spatial and temporal cloaking. In: Proceedings of First ACM/USENIX international conference on mobile systems, applications, and services (MobiSys), San Francisco, CA, May 2003
8. Greenstein B, McCoy D, Pang J, Kohno T, Seshan S, Wetherall D (2008) Improving wireless privacy with an identifier-free link layer protocol. In: Proceeding of the 6th international conference on mobile systems, applications, and services, Breckenridge, CO, 17–20 June 2008
9. Kamat P, Zhang Y, Trappe W, Ozturk C (2005) Enhancing source-location privacy in sensor network routing. In: ICDCS 2005 Proceedings of 25th IEEE international conference, Columbus

Logic Bomb

Sean W. Smith
Department of Computer Science, Dartmouth College, Hanover, NH, USA

Related Concepts

►Insider Threat

Definition

The term *logic bomb* refers to an attack by an inside adversary who plants code to automatically trigger some negate action at some point later in time.

Background

The term *insider threat* refers to the general problem posed when the adversary may be internal to an organization, already within the security perimeter and possessing privileges that may be abused.

Theory and Applications

Within this problem space, one type of adversary is a disgruntled employee angry about termination. Such an adversary may carry out an attack by (while still having appropriate privileges) planting code to automatically trigger some negate action at some point later in time (perhaps when the adversary's privileges have been removed). The term *logic bomb* refers to such attacks (and has also been used to describe similar attacks carried by parties other than disgruntled insiders, such as national intelligence operators).

Claburn [1] discusses a recent logic bomb incident that made it into court.

Recommended Reading

1. Claburn T (2009) Fannie Mae contractor indicted for logic bomb. Information Week. January 29, 2009

Logic-Based Authorization Languages

Piero A. Bonatti
Dipartimento di Scienze Fisiche, Università di Napoli Federico II, Napoli, Italy

Related Concepts

►Flexible Authorization Framework (FAF); ►Logic-Based Authorization Languages

Definition

A logic-based authorization language is an executable specification language for expressing access control policies by means of axioms written in a formal logic.

Background

Logic-based authorization languages have been introduced by Woo and Lam [20] with two main goals in mind: giving authorization policies a well-defined, unambiguous semantics, and enhancing the expressiveness of policy languages to match the flexibility needs of application domains. In the ►Flexible Authorization Framework [12] these ideas have been further elaborated by placing more structure on policies; the syntactic restrictions adopted in this work provide policy authoring guidelines and guarantee good semantic and computational properties. Subsequently, a rich literature has been extending the expressiveness of logic-based authorization languages to address the specificities of trust negotiation and usage control.

Theory

Logic-based authorization languages typically adopt a vocabulary of distinguished predicates that denote security-related concepts such as authorizations, digital credentials, etc. Such a reserved vocabulary can be extended with application-dependent symbols to model policy evaluation contexts, including, for example, user profile information, groups, roles, object hierarchies, data models, and histories. Role-based and attribute-based access control can be modeled as special cases.

In this framework, an authorization A (encoded as a logical atom) is granted by a policy P and a context C (both encoded as a set of axioms) if, and only if, A is entailed by $P \cup C$. The precise notion of entailment adopted depends on the formal logic underlying the authorization language; a common requirement is that entailment – as well as the other reasoning tasks mentioned below – should be efficiently decidable.

Often the underlying logic is nonmonotonic, as required to model default policies – such as open and closed policies – and authorization inheritance with overriding. Further nonclassical features include constructs for distributed evaluation [2, 11], temporal reasoning [13] and dynamic logics [18] (for time-dependent and usage control policies), paraconsistent semantics [10] (for conflict resolution), and deontic modalities [3] (to associate obligations to authorizations).

Most logic-based authorization languages are based on logic programming languages that nicely match expressiveness requirements and enjoy low asymptotic complexity (e.g., quadratic time in the case of stratified logic programs under the stable model semantics). A few approaches are based on description logics [16, 19].

Entailment is not the only reasoning task relevant to logic-based authorization languages. In some trust negotiation frameworks [5, 8, 17], agents have to find out a set of elements E in their portfolio of credentials such that $P \cup C \cup E$ entails a desired authorization A, where P is a given policy and C a given context. This kind of inference is called *abduction* in the automated reasoning jargon. Abduction has been proposed also as a technique for policy analysis and explanation [1, 7]. Another relevant reasoning task is *policy comparison*, a variant of a query containment problem useful for compliance checking (as in ►P3P) and policy validation [4].

The interested reader may find more details in a survey on logic-based access control languages [9] and a tutorial on rule-based policies [6].

Implementations

Several logic-based authorization languages have actually been implemented and deployed, including Cassandra [2], KAoS [19], PeerTrust [11], Protune [5], Rei [14] and TrustBuilder2 [17]. In perspective, rule-based trust negotiation frameworks may take advantage of the Rule Interchange Format W3C standard (www.w3.org/2005/rules/) to publish and exchange policies. The frameworks based on Description Logics typically rely on the W3C standard OWL (www.w3.org/2004/OWL/).

Open Problems and Future Directions

Some of the major open issues concern usability and usage control. Both issues are shared by all policy languages, including those that are not based on any formal logic. End users are typically not good at writing their policies (see, e.g., [15]). Issues such as the trade-off between expressiveness and usability are still largely unresolved, and authoring and validation tools currently provide little help. Concerning usage control, there is currently no general reliable and scalable technique for enforcing and monitoring the usage restrictions prescribed by a policy on a piece of information after it has been disclosed.

Recommended Reading

1. Becker MY, Nanz S (2008) The role of abduction in declarative authorization policies. In: Hudak P, Warren DS (eds) PADL, Lecture notes in computer science, vol 4902, Springer, Heidelberg, pp 84–99
2. Becker MY, Sewell P (2004) Cassandra: distributed access control policies with tunable expressiveness. In: 5th IEEE international workshop on policies for distributed systems and networks (POLICY 2004), IEEE Computer Society, Yorktown Heights, pp 159–168

3. Bieber P, Cuppens F (1993) Expression of confidentiality policies with deontic logic. In: Deontic logic in computer science: normative system specification, Wiley, Chichester, pp 103–123
4. Bonatti PA, Mogavero F (2008) Comparing rule-based policies. In: 9th IEEE international workshop on policies for distributed systems and networks (POLICY 2008), IEEE Computer Society, Palisades, New York, pp 11–18
5. Bonatti PA, Olmedilla D (2005) Driving and monitoring provisional trust negotiation with metapolicies. In: 6th IEEE international workshop on policies for distributed systems and networks (POLICY 2005), IEEE Computer Society, Stockholm, Sweden, pp 14–23
6. Bonatti PA, Olmedilla D (2007) Rule-based policy representation and reasoning for the semantic web. In: Antoniou D, Assmann U, Baroglio C, Decker S, Henze N, Patranjan PL, Tolksdorf R (eds) Reasoning web, Lecture notes in computer science, vol 4636, Springer, Heidelberg, pp 240–268
7. Bonatti PA, Olmedilla D, Peer J (2006) Advanced policy explanations on the web. In: Brewka G, Coradeschi S, Perini A, Traverso P (eds) ECAI, Frontiers in artificial intelligence and applications, vol 141, IOS Press, Amsterdam, pp 200–204
8. Bonatti PA, Samarati P (2002) A uniform framework for regulating service access and information release on the web. J Computer Sec 10(3):241–272
9. Bonatti PA, Samarati P (2003) Logics for authorization and security. In: Chomicki J, van der Meyden R, Saake G (eds) Logics for emerging applications of databases, Springer, Berlin, pp 277–323
10. Bruns G, Huth M (2008) Access-control policies via Belnap logic: Elective and efficient composition and analysis. In: CSF, IEEE Computer Society, Pittsburg, PA (USA), pp 163–176
11. Gavriloaie R, Nejdl W, Olmedilla D, Seamons KE, Winslett M (2004) No registration needed: how to use declarative policies and negotiation to access sensitive resources on the semantic web. In: 1st European semantic web symposium (ESWS 2004), Lecture notes in computer science, vol 3053, Heraklion, Crete. Springer, Heidelberg, pp 342–356
12. Jajodia S, Samarati P, Sapino ML, Subrahmanian VS (2001) Flexible support for multiple access control policies. ACM Trans Data Sys 26(2):214–260
13. Joshi JB, Bertino E, Latif U, Ghafoor A (2005) A generalized temporal role-based access control model. IEEE Trans Knowl Data Eng 17(1):4–23
14. Kagal L, Finin TW, Joshi A (2003) A policy language for a pervasive computing environment. In: 4th IEEE international workshop on policies for distributed systems and networks (POLICY 2003), IEEE Computer Society, Lake Como, Italy, pp 63
15. Kelley PG, Hankes Drielsma P, Sadeh N, Cranor LF (2008) User-controllable learning of security and privacy policies. In: 1st workshop on artificial intelligence and security (AISec 2008), ACM, Rotterdam, The Netherlands, pp 11–18
16. Kolovski V, Parsia B, Katz Y, Hendler JA (2005) Representing web service policies in OWL-DL. In: Gil Y, Motta E, Benjamins VR, Musen MA (eds) International semantic web conference, Lecture notes in computer science, vol 3729, Springer, Heidelberg, pp 461–475
17. Lee AJ, Winslett M, Perano KJ (2009) Trustbuilder2: a reconfigurable framework for trust negotiation. In: IFIP Advances in information and communication technology (IFIP AICT 300), Springer, West Lafayette, IN (USA), pp 176–195
18. Pretschner A, Hilty M, Basin D (2006) Distributed usage control. Commun ACM, 49(9):39–44
19. Uszok A, Bradshaw JM, Johnson M, Jeers R, Tate A, Dalton J, Aitken JS (2004) KAoS policy management for semantic web services. IEEE Intel Sys 19(4):32–41
20. Woo TYC, Lam SS (1993) Authorizations in distributed systems: a new approach. J Comput Sec 2(2–3):107–136

Longhand

►Handwriting Analysis

Luby-Rackoff Ciphers

Lars R. Knudsen
Department of Mathematics, Technical University of Denmark, Lyngby, Denmark

Related Concepts

►Pseudorandom Permutation; ►Secret Key Encryption; ►Symmetric Cryptography

Definition

A Luby-Rackoff cipher is a Feistel cipher where in each round the nonlinear function used is assumed to be chosen uniformly at random from the set of all such functions. These ciphers are mainly of theoretical interest.

Background

In their celebrated paper [2] Luby and Rackoff showed how to construct $2n$-bit ►Pseudorandom Permutations from n-bit random functions. The constructions use three and four rounds in Feistel networks with randomly chosen functions in the round functions. Let L and R be the left, respectively, the right n-bit halves of a $2n$-bit input. Then one round of a Feistel network is defined as follows:

$$F(L,R) = (R, L \oplus f(R)),$$

where $f : \{0,1\}^n \to \{0,1\}^n$ is a randomly chosen function. In order to make the encryption and decryption routines similar, it is custom to swap the halves of the output of the last round in an r-round Feistel network. The entry on *Feistel ciphers* provides an overview of practical designs.

Theory

Luby and Rackoff's result says that in order to be able to distinguish the three-round construction from a randomly chosen $2n$-bit function with probability close to one, an attacker needs at least $2^{n/2}$ chosen plaintexts and their

corresponding ciphertexts. Such a permutation is called *pseudorandom* [2]. However, if an attacker can mount a chosen plaintext and a chosen ciphertext attack, he is able to distinguish the construction from a randomly chosen $2n$-bit function using two chosen plaintexts and one chosen ciphertext. To see this, choose two plaintexts with left halves L_1 and L_2, where $L_1 \neq L_2$ and with equal right halves R. From the corresponding ciphertexts (T_1, S_1) and (T_2, S_2) compute the ciphertext $(T_1 \oplus L_1 \oplus L_2, S_1)$ and get the corresponding plaintext. Then the right half of this plaintext equals $R \oplus S_1 \oplus S_2$, whereas this would be the case only with probability 2^{-n} in the random case. Luby and Rackoff also showed that in a combined chosen plaintext and chosen ciphertext attack for the four-round construction, an attacker will need roughly $2^{n/2}$ chosen texts to win with probability close to one. Such a permutation is called *super pseudorandom*.

With q chosen plaintexts one can distinguish the three-round construction from a random function with probability

$$p = 1 - e^{-q(q-1)/2^{n+1}},$$

which is close to one for $q \simeq 2^{n/2}$ [4]. Choose plaintexts (L_i, R) for $i = 1 \ldots, q$, where the L_is are (pair-wise) distinct and R is a fixed, arbitrary value. Denote by (T_i, S_i) the corresponding ciphertexts. Then for the three-round construction with probability p one finds at least one pair (i, j) for which $i \neq j$, $L_i \oplus L_j = T_i \oplus T_j$ and $S_i = S_j$. For a random $2n$-bit function and with $q \simeq 2^{n/2}$ this happens with only very small probability. Also, with roughly $2^{n/2}$ chosen plaintexts one can distinguish the four-round construction from a random function. Choose plaintexts (L_i, R) for $i = 1 \ldots, c2^{n/2}$, where c is an integer, the L_is are (pairwise) distinct and R is a fixed, arbitrary value. Denote by (T_i, S_i) the corresponding ciphertexts. Then for the four-round construction one expects to find c pairs of plaintexts for which $L_i \oplus L_j = S_i \oplus S_j$, whereas for a random $2n$-bit function one expects to find only $c/2$ such pairs [4]. These results show that the inequalities by Luby and Rackoff are tight, that is, to distinguish the three-round and four-round constructions from a randomly chosen function with probability close to one, an attacker needs at least but not much more than $2^{n/2}$ chosen plaintexts and their corresponding ciphertexts.

The Luby-Rackoff result has spawned a lot of research in this area, and many different constructions have been proposed, of which only a few are mentioned here. In [5] it was shown that four-round super Pseudorandom Permutations can be constructed from only one or two (pseudo)random n-bit functions. In the four-round construction, the first and fourth functions can be replaced by simpler "combinatorial" constructions achieving the same level of security as the original construction as shown in [3], which is also a good reference for a survey of this area.

Coppersmith [1] analyzed the four-round construction. It was shown that with $n2^n$ chosen plaintexts the round functions can be identified up to symmetry. With 8×2^n texts 99.9% of the functions are identified.

There is a trivial upper bound of $O(2^n)$ for distinguishing constructions with r rounds for any r from a randomly chosen $2n$-bit function. This follows from the fact that the Luby-Rackoff constructions are permutations and with 2^n chosen distinct plaintexts, the resulting ciphertexts will all be distinct, whereas a collision is likely to occur for a truly random function [4]. It has been studied how to distinguish the Luby-Rackoff constructions from randomly chosen $2n$-bit permutations (bijective mappings). However, in the cases using $O(2^{n/2})$ inputs this does not make much of a difference, since in these cases the probability to distinguish a $2n$-bit randomly chosen permutation from a $2n$-bit randomly chosen function is small. Also, for a fixed number of rounds, r, it has been shown that there is an upper bound of $O(2^n)$ for distinguishing the r-round construction from a randomly chosen $2n$-bit permutation [4]. More recent results indicate that with a larger number of rounds, the lower bound for the security of the Luby-Rackoff constructions approaches 2^n.

Recommended Reading

1. Coppersmith D (1996) Luby-Rackoff: four rounds is not enough. Technical report RC 20674. IBM, Yorktown Heights
2. Luby M, Rackoff C (1998) How to construct pseudorandom permutations from pseudorandom functions. SIAM J Comput 17(2):373–386
3. Naor M, Reingold O (1999) On the construction of pseudorandom permutations: Luby–Rackoff revisited. J Cryptol 12(1):29–66
4. Patarin J (1992) New results on pseudorandom permutations generators based on the DES scheme. In: Feigenbaum J (ed) Advances in cryptology – CRYPTO '91: proceedings. Lecture notes in computer science, vol 576. Springer, Berlin, pp 301–312
5. Patarin J (1993) How to construct pseudorandom and super pseudorandom permutations from one single pseudorandom function. 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 256–266