G

Gait Recognition

Sudeep Sarkar[1], Zongyi Liu[2]
[1]Computer Science and Engineering, University of South Florida, Tampa, FL, USA
[2]Amazon.com, Seattle, WA, USA

Definition

Gait recognition refers to the use of video of human gait, processed by computer vision methods, to recognize or to identify persons based on their body shape and walking styles. This is a passive mode of accquiring biometric data from a distance.

Background

The gait of a person is a periodic activity with each gait cycle covering two strides – the left foot forward and right foot forward strides. Each stride spans the double-support stance to the legs-together stance as the legs swing past each other and back to the double-support stance (see Fig. 1). A large portion of the musculo-skeletal system is involved in the production of gait, making it a likely source of discriminating information between persons. There are two potential sources of biometric information in human gait as captured in video: shape and dynamics. Shape refers to the configuration or shape of the persons as they perform different gait phases. Dynamics refer to the *rates* of transition between these phases. This is usually the aspect one refers to when one talks about gait in other contexts, such as biomechanics or human motion recognition. In this respect, gait biometrics research has to synthesize both shape and human motion research. Designing well-performing gait recognition algorithms does not appear to be a straightforward application of existing methods developed in shape research or those in human motion research, independently of each other. The challenge is to overcome gait motion variations due to conditions, other than identity, such as footwear, clothing, walking surface, carrying objects, over elapsed time, walking speed, indoor versus outdoors, and so on. Given the lack of theoretical understanding or modeling of how these factors effect gait shape and dynamics, gait recognition research, like most biometrics research, is reliant on datasets.

Theory

Gait recognition approaches are basically of three types: (a) temporal alignment based, (b) silhouette shape based, and (c) static parameter based approaches.

Temporal-Alignment-Based Approaches

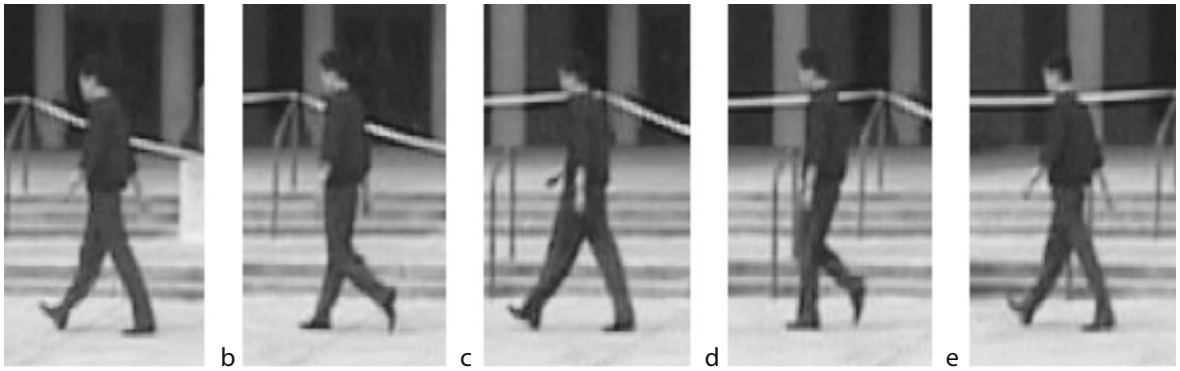
The temporal-alignment-based approach treats the gait sequence as a time series and involves a classic three-stage processing framework. The first stage is the extraction of features such as whole silhouettes or 2D shape features such as Fourier descriptors, 2D projections, statistics of the overall shape, or various kinds of moment-based features. These features are modified to impart some invariance with respect to distance from camera, that is, scale invariance. The interested reader is pointed to works of Nixon and Carter, such as [6], who have experimented with many features.

The second step is the design of the distance or similarity measure used to quantify differences of two feature vectors, which can be Euclidean, dot product, or based on probabilistic models, or Procrustes distance, or derived based on rigorous manifold analysis.

The third step involves the alignment of sequences of these features, corresponding to the two given sequences to be compared. For this, one can use simple correlation, dynamic time warping, Fourier analysis, and hidden Markov models (HMM) [11].

A head to head study of this class of approaches was done by Boulgouris et al. [1]. Time warping methods seemed to result in better performance when matching across surface changes. A variation of the HMM approach, where the overall distance was the accumulation of the observation probability, ignoring the transition probabilities, resulted in almost similar performance as the full HMM. This points to the importance of silhouette shape information for recognition and leads one to a second class of approaches to gait recognition, which is discussed next.

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



Gait Recognition. Fig. 1 A gait cycle can be partitioned into four periods: (i) right stance period when the right foot is in contact with the floor, beginning from "right heel-strike" (**photo a**) and ending at "right toe-off" (**photo d**) (ii) left swing period when the left foot is not in contact with the floor, beginning from "left toe-off" (**photo b**) and ending at "left heel-strike" (**photo c**) (iii) left stance period when the left foot is in contact with the floor, beginning from "left heel-strike" (**photo c**) and ending at "left toe-off" (**photo e**) and (iv) right swing period when the right foot is not in contact with the floor, beginning from "right toe-off" (**photo d**) and ending at "right heel-strike" (**photo e**). Moreover, the time between these periods, that is, when both feet are in contact with the floor, is called "double limb support"

Shape-Based Approaches

This class of approaches emphasizes the silhouette shape similarity and underplays the temporal information. So far, they have resulted in the highest demonstrated performance. One direction involves the transformation of the silhouette sequence into a single image representation. The simplest such transformation is the averaged silhouette, computed by simply summing the silhouettes over one gait cycle [3]. Similarity is based on just the Euclidean distance between the average silhouettes from two silhouette sequences. A sophisticated version of this idea, with enhanced performance, was also proposed by Han and Bhanu [2].

Another way of using shape information preserves individual silhouettes but disregards the sequence ordering, and treats the sequences as just a collection of silhouette shapes [10]. In this approach, the silhouettes with similar shapes are clustered using the spectral partitioning framework, based on graph weights built out of the correlation of high variance areas in the shape, representing parts such as arms and legs. The power of this representation partly derives from the ability to identify and disregard low-quality silhouettes in a sequence; bad silhouettes form a separate cluster. Identification is made by comparing the collection of silhouette shapes from a given probe set to the gallery shape clusters.

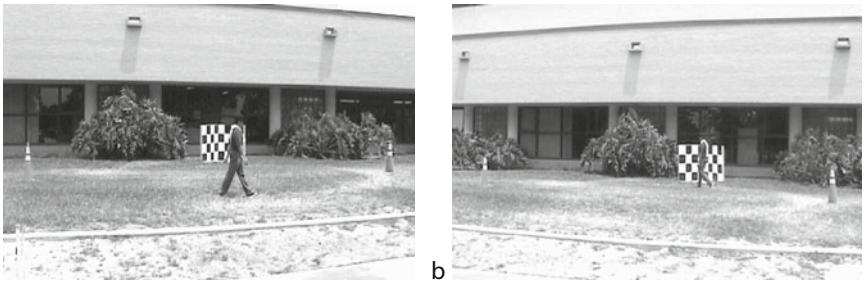
Liu and Sarkar's gait recognition approach normalizes the gait of each person to a standard gait model [5]. This standard gait model is a population-HMM model built using silhouette sequences from all persons in the gallery. One p-HMM represents the average gait over the population and is used to normalize any given gait sequence. A separate temporal alignment process is not needed. The silhouette shapes of each stance are then matched to arrive at a similarity measure.

Static Parameters

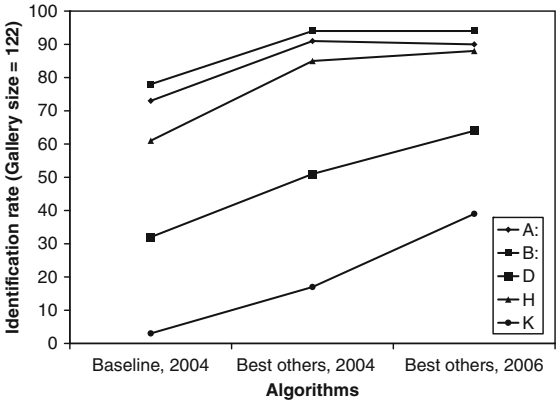
The third class of approaches opts for parameters that can be used to characterize gait dynamics, such as stride length, cadence, and stride speed along with static body parameters such as the ratio of sizes of various body parts [9]. However, these approaches have not yet reported high performances on common databases, partly due to their need for 3D calibration information.

Gait Recognition Performance

The HumanID gait challenge problem facilitates an objective, quantitative measurement of gait research progress on a large dataset [7] consisting of 1,870 sequences from 122 subjects. The problem definition has three components: a dataset, challenge experiments of different difficulty levels, and a simple gait recognition approach that is intended to set a baseline performance level to improve upon. Figure 2 shows some sample frames from this dataset. It was collected outdoors and each person in the data set was studied under combination of as many as five conditions, such as change in viewing angle, change in shoe type, change in walking surfaces (concrete and grass), carrying or not carrying a briefcase, and temporal differences. Along with the



Gait Recognition. Fig. 2 Samples from the HumanID gait challenge dataset: subject walking on grass. (**a**) along the frontal half of the elliptical path, and (**b**) along the back half of the elliptical path



Gait Recognition. Fig. 3 Improvement in gait recognition algorithms over time with respect to the baseline performance on the full dataset with 122 subjects for the key experiments. Experiments A, B, D, H, and K, involves gait matching across viewpoint, shoe types, walking surface type, carrying condition, and time. From 2004 to 2006, the best reported performances are better on all the experiments

dataset, the gait challenge problem includes a definition of a set of 12 challenge experiments (A through L), spanning different levels of difficulty.

In Fig. 3, the performance of the baseline performance and the best performance reported in the literature has been tracked. Performance is reported in terms of identification rate, that is, percentage of times the gait template with the correct identity is at the top rank in a 1 to N match. In 2002, when the gait challenge problem was released, the performance of the baseline algorithm was better than the best reported performance. By 2004, while the baseline algorithm performance improved as the algorithm was fine-tuned, the performance of the best performance improved significantly and continued to improve through 2006.

Open Problems

Study of gait recognition from video has made notable progress since 2001. Based on the results that have been reported on common datasets by multiple groups the following guidance for productive future work can be given [8]:

1. Gait dynamics, which has been the core focus of most study of human gait in computer vision, is susceptible to change, whereas, stance shape information is somewhat stable. However, the evidence for this is not conclusive. A better understanding of the change in dynamics for the same person under various conditions is needed.
2. Most recogniton algorithms use silhouettes as input, which raises the question of silhouette quality on recogniton. Of course, with very poor silhouettes, performance would be poor. However, studies [4] show that even with perfect, hand drawn silhouettes, the recognition performance is the same as that obtained with automated methods. This observation has implication for future work direction in gait recognition. Instead of searching for better methods for silhouette detection to improve recognition, it would be more productive to study and isolate components of gait that do not change under shoe, surface, or time.
3. Focused analysis of the study of the impact of a covariate on match-score distribution, suggests that shoe type has the least effect on performance, but the effect is nevertheless statistically significant [7]. This is followed by either a change in camera view or carrying a brief case. Carrying a brief case does not affect performance as much as one might expect. This effect is marginally larger than changing shoe type but is substantially smaller than a change in surface type. In future experiments, it may be interesting to investigate the effect of carrying a backpack rather than a briefcase, or to vary the object that is carried.

The other factor with large impact on gait recognition is walking surface. With the subject walking on grass in the gallery sequence and on concrete in the probe sequence, highest reported rank-one recognition is only 57% [5]. Performance degradation might be even larger for other surface types, such as sand or gravel that might reasonably be encountered in some applications. The large effect of surface type on performance suggests that an important future research topic might be to investigate whether the change in gait with surface type is predictable. For example, given a description of gait from walking on concrete, is it possible to predict the gait description that would be obtained from walking on grass or sand? Alternatively, is there some other description of gait that is not as sensitive to change in surface type?

4. Given the data-driven nature of biometrics research, the key to future progress is such data sets collected to explore issues not considered or raised by existing ones. One idea could be to collect a dataset that facilitates the better understanding of the variation of gait due to surface conditions and across elapsed time. This dataset should include a set of predefined experiments, designed to study the impact of each covariate, and a baseline performance to beat.

Recommended Reading

1. Boulgouris NV, Hatzinakos D, Plataniotis KN (2005) Gait recognition: a challenging signal processing technology for biometric identification. IEEE Signal Proc Mag 22(6):78–90
2. Han J, Bhanu B (2004) Statistical feature fusion for gait-based human recognition. In: IEEE conference on computer vision and pattern recognition, II, pp 842–847
3. Liu Z, Sarkar S (2004) Simplest representation yet for gait recognition: averaged silhouette. In: International conference on pattern recognition, 4, pp 211–214
4. Liu Z, Sarkar S (2005) Effect of silhouette quality on hard problems in gait recognition. IEEE T Syst Man Cy B 35(2):170–183
5. Liu Z, Sarkar S (2006) Improved gait recognition by gait dynamics normalization. IEEE T Pattern Anal 28(6):863–876
6. Nixon MS, Carter JN (2004) Advances in automatic gait recognition. In: International conference on automatic face and gesture recognition, pp 139–146
7. Sarkar S, Phillips PJ, Liu Z, Vega IR, Grother P, Bowyer KW (2005) The Human ID gait challenge problem: data sets, performance, and analysis. IEEE T Pattern Anal 27(2):162–177
8. Sarkar S, Liu Z (2008) Gait recognition. Handbook of biometrics, Springer, Heidelberg
9. Tanawongsuwan R, Bobick A (2001) Gait recognition from time-normalized joint-angle trajectories in the walking plane. In: IEEE conference on computer vision and pattern recognition, II, pp 726–731
10. Tolliver D, Collins R (2003) Gait shape estimation for identification. In: International conference on audio- and video-based biometric person authentication, pp 734–742
11. Veeraraghavan A, Roy Chowdhury A, Chellappa R (2004) Role of shape and kinematics in human movement analysis. In: IEEE conference on computer vision and pattern recognition, Washington, DC

Galois Counter Mode

David McGrew
Poolesville, MD, USA

Synonyms

GCM

Related Concepts

►Authenticated Encryption; ►Block Ciphers; ►MAC Algorithms; ►Modes of Operation of a Block Cipher; ►Symmetric Cryptosystem

Definition

Galois Counter Mode (GCM) is a block cipher mode of operation that provides authenticated encryption with associated data. It has a minimal computational cost and is widely used in practice, especially at higher data rates.

Background

A mode of operation of a block cipher uses a ►block cipher, along with other operations, to encrypt or authenticate a message (►Encryption and ►Authentication). A mode of operation that provides *authenticated encryption with associated data* (AEAD) will encrypt and authenticate a message, and at the same time, authenticate (but not encrypt) a string of associated data. In practice, a typical use of an AEAD algorithm is to protect a packet by encrypting and authenticating the body of a data packet, and authenticating (but not encrypting) the packet header.

The same block cipher used in different modes of operation can have widely different performance, cost, and security characteristics. Important performance goals for a block cipher mode of operation include minimizing the time needed to encrypt and/or authenticate a message and minimizing circuit size. GCM is an AEAD mode of operation that meets these goals.

The block cipher encryption of the value $X \in \{0,1\}^w$ with the key K is denoted as $E(K, X)$. An encryption mode of operation partitions the plaintext P into a sequence of n bit strings, in which the bit length of the last bit string is u, and the bit length of the other bit strings is w. The sequence is denoted as $P_1, P_2, \ldots, P_{n-1}, P_n^*$, and the bit strings are

called blocks; the last bit string, P_n^* , may not be a complete block. The ciphertext C is similarly partitioned as $C_1, C_2, \ldots, C_{n-1}, C_n^*$, where the number of bits in the final block C_n^* is u.

The Cipher Block Chaining (CBC) mode of operation is a widely used encryption mode, but it cannot easily be parallelized, and thus is unsuitable for high data rates. The CBC encryption operation computes the ciphertext blocks as $C_i = E(K, P_i \oplus C_{i-1})$. The values of the inputs in each block cipher evaluation depend on the ciphertext output of the previous block cipher evaluation. Thus, the block cipher evaluations must be performed serially, and not in parallel.

Counter mode (CTR) is an encryption mode in which block cipher evaluations can be computed in parallel, because each plaintext input to each block cipher evaluation, during an encryption and decryption, is independent of all block cipher outputs. The CTR encryption process determines the ciphertext as $C_i = P_i \oplus E(K, Y_i)$ where Y_i is a counter; $Y_i = \text{incr}(Y_{i-1})$ for $i > 1$, and Y_0 is the initial counter value. The increment function incr( ) updates the counter; this function and the choice of initial counter values ensure that all counters are unique, for each key K.

A typical block cipher consists of a round function that is iterated r times. CTR encryption can be efficiently implemented at high data rates by using a pipeline of r round functions. Each successive round function will have data corresponding to successive plaintext blocks. In this case, CTR can encrypt r times as much data as CBC with the same size circuit.

Theory

While CTR provides efficient encryption, it does not provide any message authentication. GCM addresses this shortcoming by combining CTR encryption with a message authentication code (►MAC Algorithms) based on a universal hash function U. After the ciphertext has been produced during an encryption operation, the hash of the associated data and the ciphertext is computed, then the result is encrypted with K to produce an authentication tag T.

The function U is a *polynomial hash* in a finite field of characteristic two. The message being hashed is partitioned into blocks, and the blocks are treated as the coefficients of a polynomial in the field $GF(2^w)$, and the polynomial is evaluated at the hash function key, using Horner's method. The multiplication of two elements $X, Y \in GF(2^w)$ is denoted as $X \cdot Y$, and the addition of X and Y is denoted as $X \oplus Y$. The function length(S) takes a bit string S with a length between zero and $2^{w/2} - 1$, inclusive, and returns a $w/2$-bit string containing the nonnegative integer describing the number of bits in its argument, with the least significant bit on the right. The block cipher width w must be an even number. The additional authenticated data A is partitioned as $A_1, A_2, \ldots, A_{m-1}, A_m^*$, where the last bit string A_m^* may be a partial block of length v, and m and v denote the unique pair of positive integers such that the total number of bits in A is $(m-1)w + v$ and $1 \le v \le w$, when length$(A) > 0$; otherwise $m = v = 0$. The input C is partitioned into w-bit blocks $C_1, C_2, \ldots, C_{n-1}, C_n^*$, where C_n^* has length v. The universal hash function is defined by $U(H, A, C) = X_{m+n+1}$, where the variables $X_i \in GF(2^w)$ for $i = 0, \ldots, m+n+1$ are defined recursively as

$$X_i = \begin{cases} 0 & \text{for } i = 0 \\ (X_{i-1} \oplus A_i) \cdot H & \text{for } i = 1, \ldots, m-1 \\ (X_{m-1} \oplus (A_m^* \| 0^{w-v})) \cdot H & \text{for } i = m \\ (X_{i-1} \oplus C_{i-m}) \cdot H & \text{for } i = m+1, \ldots, m+n-1 \\ (X_{m+n-1} \oplus (C_n^* \| 0^{w-u})) \cdot H & \text{for } i = m+n \\ (X_{m+n} \oplus (\text{length}(A) \| \text{length}(C))) \cdot H & \text{for } i = m+n+1. \end{cases}$$

Here the value H is the hash key. The expression 0^l denotes a string of l zero bits, and $A \| B$ denotes the concatenation of two bit strings A and B.

The GCM authenticated encryption operation takes as inputs a secret key K, initialization vector IV, a plaintext P, and additional authenticated data A, and gives as its outputs a ciphertext C and an authentication tag T. The operation is defined as

$$\begin{aligned} H &= E(K, 0^w) \\ Y_0 &= \begin{cases} IV \| 0^{31} 1 & \text{if length}(IV) = w - 32 \\ U(H, \{\}, IV) & \text{otherwise.} \end{cases} \\ Y_i &= \text{incr}(Y_{i-1}) \text{ for } i = 1, \ldots, n \\ C_i &= P_i \oplus E(K, Y_i) \text{ for } i = 1, \ldots, n-1 \\ C_n^* &= P_n^* \oplus \text{MSB}_u(E(K, Y_n)) \\ T &= \text{MSB}_t(U(H, A, C) \oplus E(K, Y_0)) \end{aligned} \quad (1)$$

The function $\text{MSB}_t(S)$ takes a bit string S and returns the bit string containing only the leftmost t bits of S, and the tag-length parameter t is fixed for each instance of the key. The hash key H is computed from K and can optionally be cached between invocations.

The initial counter Y_0 is computed from the IV; if the IV is not exactly $w - 32$ bits in length, then the hash U is applied to the IV. Performance at high data rates is better if the IV matches that length. The symbol {} denotes the bit string with zero length. The particular increment function incr( ) used in GCM treats the rightmost 32 bits of its argument as a nonnegative integer with the least significant bit

on the right, and increments this value modulo 2^{32}, that is, $\mathrm{incr}(F\|I)$ is $F\|(I+1 \bmod 2^{32})$.

The inputs P, A, IV, C, and T are bit strings with lengths that obey the following relations:

$$\begin{aligned} 0 \le \mathrm{length}(P) &\le (2^{32}-2)w \\ 0 \le \mathrm{length}(A) &\le 2^{w/2} \\ 0 < \mathrm{length}(IV) &\le 2^{w/2} \\ \mathrm{length}(C) &= \mathrm{length}(P) \\ \mathrm{length}(T) &= t \le w, \end{aligned} \tag{2}$$

where the secret key K has a length appropriate to the block cipher, and is only used as an input to that cipher. For each fixed value of K, it is essential for security that *each value of the IV must be distinct*, but those IV values need not have equal lengths.

The GCM authenticated decryption operation accepts the inputs K, IV, C, A, and T, as defined above. It outputs either the plaintext value P or the special symbol **FAIL** that indicates that its inputs are not authentic.

GCM is secure in the standard ►adaptive chosen plaintext and chosen ciphertext model of concrete security. If the block cipher cannot be distinguished from a random permutation, then an adversary will be unable to distinguish GCM ciphertexts from random strings of the same length, and will be unable to forge GCM authentication tags. This is true as long as the following bounds on the amount of data processed are respected, for a fixed value of the key:

$$(l_{\mathrm{P}}/w+2q)^2 2^{-w-1} - q((l_{\mathrm{P}}/w+2q)\lceil l_{\mathrm{IV}}/w+1\rceil 2^{1-w} + \lceil l/w+1\rceil 2^{-t}) \ll 1$$

where the number of encryption and/or decryption operations is q, the total number of plaintext or ciphertext bits processed is denoted l_{P} and where $\mathrm{length}(C)+\mathrm{length}(A) \le l$ and $\mathrm{length}(IV) \le l_{\mathrm{IV}}$ for each query.

Applications

GCM is used in a wide number of standards, including IEEE 802.1AE Ethernet Security, IETF Internet Protocol Security (IPsec) and Transport Layer Security (TLS), and IEEE 1619.1 storage encryption. The algorithm itself is recommended in NIST Special Publication 800-38D.

When GCM is used with zero-length plaintext input, it acts as a ►message authentication code (MAC). This use is called GMAC, and it is an *incremental* MAC: after computing the GMAC value of message M, the computational cost of computing the GMAC value of message M' is proportional to the hamming weight between those messages. If the messages differ in only a few bit locations then the MAC of M' can be easily computed from the MAC of M. This property of GMAC follows from the algebraic properties of its hash function; it is highly useful for protecting data at rest.

Open Problems

GCM has been criticized for being vulnerable to a *multiple forgery attacks*, in which an attacker who succeeds in crafting a single forgery will be able to use information gleaned from that success to perpetrate additional forgeries. This is not a significant security issue unless the authentication tag size t is small. Another criticism is that security degrades with the length of messages that are processed. These demerits are due to the choice of hash function used in GCM, which also bring low computational cost and low latency. It is an open problem to design an authenticated encryption algorithm with all of its benefits, but none of its demerits.

Recommended Reading

1. Dworkin M (Nov 2007) Recommendation for block cipher modes of operation: galois/counter mode (GCM) and GMAC. NIST Special Publication 800-38D, National Institute of Standards and Technology, Gaithersburg
2. McGrew D (Dec 2005) Efficient authentication of large, dynamic data sets using Galois/counter mode (GCM). In: Third international IEEE security in storage workshop, San Francisco
3. McGrew D, Viega J (Oct 2004) The security and performance of the Galois/counter mode (GCM) of operation. In: Proceedings of INDOCRYPT04, Chennai. Springer, Berlin. Full paper available from the IACR cryptology ePrint archive: Report 2004/193. http://eprint.iacr.org/2004/193/

Galois Message Authentication Code

►GMAC

Gap

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

Related Concepts

►Golomb's Randomness Postulates; ►Maximal-Length Linear Sequences; ►Pseudo-Noise Sequences (PN-Sequences); ►Run; ►Sequences

Definition

A gap of length k in a binary sequence is a set of k consecutive 0s flanked by 1s (►run).

Applications

The number of gaps in a sequence is important in ►Golomb's randomness postulates [1] to evaluate the randomness of a sequence.

Recommended Reading

1. Golomb SW (1967) Shift register sequences. Holden-Day series in information systems. Holden-Day, San Francisco. Revised ed., Aegean Park Press, Laguna Hills, 1982

GCD

►Greatest Common Divisor

GCM

►Galois Counter Mode

Gene

►DNA

Generalized Mersenne Prime

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

Definition

Generalized Mersenne Prime is a *prime number* of the form

$$p = f(2^m),$$

where $f(t)$ is a low-degree polynomial with small integer coefficients.

Applications

Generalized Mersenne primes are useful in *public-key cryptography* because reduction modulo these primes can be very fast using a generalization of the technique used for *Mersenne primes*. In particular, the integer division by p is replaced by a small number of additions and subtractions and some bit shifts.

In practice, m is taken to be a multiple of the word size of the machine in order to eliminate the need for shifting bits within words. The precise rule for modular reduction (►*modular arithmetic*) depends on $f(t)$.

The simplest example is

$$p = 2^{192} - 2^{64} - 1.$$

In this case, $f(t) = t^3 - t - 1$, and one has

$$\sum_{i=0}^{5} c_i t^i \equiv b_2 t^2 + b_1 t + b_0 \,(mod\, f(t)),$$

where

$$\begin{aligned} b_0 &= a_0 + a_3 + a_5 \\ b_1 &= a_1 + a_3 + a_4 + a_5 \\ b_2 &= a_2 + a_4 + a_5. \end{aligned}$$

This yields the following reduction algorithm for $p = f(2^{64})$. A positive integer modulo p^2 is represented as the concatenation of six 64-bit words

$$(c_5 \quad c_4 \quad c_3 \quad c_2 \quad c_1 \quad c_0),$$

which represents

$$c = \sum_{i=0}^{5} c_i \cdot 2^{64i}.$$

Then

$$c \equiv s_0 + s_1 + s_2 + s_3 \;\; (mod\; p),$$

where the s_i are the integers represented by the following concatenations of the six words:

$$\begin{aligned} s_0 &= (c_2 \quad c_1 \quad c_0) \\ s_1 &= (0 \quad c_3 \quad c_3) \\ s_2 &= (c_4 \quad c_4 \quad 0) \\ s_3 &= (c_5 \quad c_5 \quad c_5). \end{aligned}$$

In the general case, one derives the reduction formulae by writing down the reductions $(\mathrm{mod}\, f(t))$ of t^j for $d \le j < 2d$, where d is the degree of f. One then rearranges the terms to minimize the numbers of additions and subtractions. Details can be found in [2].

To find a generalized Mersenne prime of a given bit size, one lists the small-coefficient integer polynomials $f(t)$ of appropriate degrees and chooses the one for which $f(2^m)$ is prime and whose reduction rules require the smallest number of additions and subtractions.

The US National Institute for Standards and Technology (NIST) has standardized four generalized Mersenne primes, including the above example, in its *Digital Signature Standard* [1].

A useful generalization of generalized Mersenne primes is generalized Mersenne numbers. A near-prime (i.e., a small multiple of a large prime) of the generalized Mersenne form can be used in *elliptic curve cryptography* almost as easily as a prime modulus. If m is a generalized Mersenne number divisible by a large prime p, then one implements the cryptography over an elliptic curve over the *field* F_p. The arithmetic is carried out modulo m (using the appropriate reduction rules) and is additionally reduced modulo p at the end of the calculation. This can sometimes be advantageous if a generalized Mersenne near-prime exists requiring significantly fewer additions and subtractions than any available generalized Mersenne prime.

Recommended Reading

1. Federal Information Processing Standard (FIPS) 186-3 (June 2009) Digital signature standard. http://csrc.nist.gov/publications/PubsFIPS.html
2. Solinas JA (1999) Generalized Mersenne numbers. Centre for Applied Cryptographic Research (CACR) Tech. Report 99–39. http://www.cacr.math.uwaterloo.ca/

Generator

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

Synonyms

Base

Related Concepts

►Group; ►Order; ►Primitive Element; ►Subgroup

Definition

A *generator* is an element of a ►group from which all other elements of the group may be obtained by repeated application of the group operation.

Theory

An element g of a ►group is said to *generate* that group if the set of elements

$$g, g^2, g^3, \ldots$$

(where the group law is denoted here as multiplication) traverses all of the elements in the group. Such an element is a *generator*. In other words, g is a generator of a group if and only if every element y in the group can be expressed as

$$y = g^x$$

for some x. Every cyclic group has at least one generator, and a group that is generated by a single element is cyclic by definition.

When the group is a multiplicative group modulo an integer n, a generator is also called a *primitive root* of n.

Applications

A generator (sometimes also called the *base*) is one of the parameters in several cryptosystems, including ►Diffie–Hellman key agreement and the ►Digital Signature Standard.

Generic Attacks Against DLP

Dan Gordon
IDA Center for Communications Research, San Diego, CA, USA

Synonyms

Black box algorithms

Related Concepts

►Computational Diffie–Hellman Problem; ►Discrete Logarithm Problem; ►Elliptic Curve Discrete Logarithm Problem; ►Function Field Sieve; ►Index Calculus Method; ►Multi-threaded Implementation for Cryptography and Cryptanalysis; ►Number Field Sieve for DLP

Definition

Given elements g and y in a cyclic group G, the ►discrete logarithm problem (DLP) is to find an x such that $g^x = y$. An algorithm for solving the DLP that does not use any properties of G, but only composes, inverts, and compares elements g^a and y^b, is called a generic algorithm.

Background

Let G be a cyclic ►group of order n. Nechaev [2] and Shoup [6] Showed that generic algorithms against the DLP in G must take $\Omega(\sqrt{n})$ time (►O-notation). Shor showed that a quantum computer can solve a discrete logarithm problem in *any* group in polynomial time, but whether a sufficiently large quantum computer can be built is still an open problem, refer ►post-quantum cryptography for details.

A *black box group* is a group where the elements are encoded as bit strings, and group operations are done by

an oracle. For such a group, the only method of solving the DLP is a generic algorithm.

Theory

Shanks [5] gave the first generic algorithm better than a brute-force search. Let $m = \lceil \sqrt{n} \rceil$. He constructs two tables, one starting at 1 and taking "giant steps" of length m:

$$1, g^m, g^{2m}, \ldots, g^{(m-1)m}$$

and one of "baby steps" of length y:

$$y, yg, yg^2, \ldots, yg^{m-1}.$$

He then sorts these lists and looks for a match. If he finds $g^{im} = yg^j$, then $y = g^{im-j}$, and so $x = im - j$. Any $x \in [0, n-1]$ may be written in this form for $i, j \leq m$, so this algorithm will always succeed.

The time for this algorithm is $O(\sqrt{n})$ group operations, plus the time to find collisions in the two lists. This may be done either by sorting the lists or using hash tables.

The drawback to Shanks' *baby-step giant-step* algorithm is that it requires $O(\sqrt{n})$ space as well as time. Pollard [4] gave two methods that use negligible space and still run in $O(\sqrt{n})$ time: the rho method and the kangaroo method. They are not deterministic but depend on taking pseudorandom walks in G.

For the *Pollard rho method*, divide the elements of G into three subsets, S_1, S_2, and S_3, say by the value of a hash of the elements modulo three. Define a walk by $h_0 = 1$ and

$$h_{i+1} = \begin{cases} h_i y & \text{if } h_i \in S_1 \\ h_i^2 & \text{if } h_i \in S_2 \\ h_i g & \text{if } h_i \in S_3 \end{cases}$$

At each step

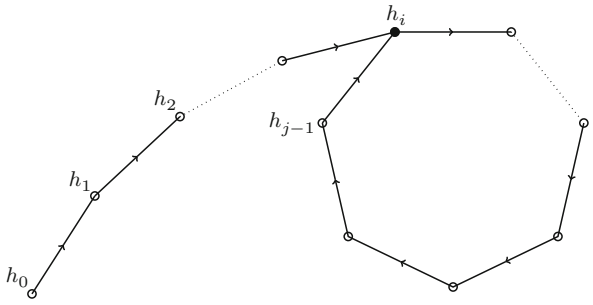
$$h_i = g^{a_i} y^{b_i} = g^{a_i + x b_i}$$

for some a_i, b_i. (In particular, $(a_0, b_0) = (0, 0)$ initially, and $(a_{i+1}, b_{i+1}) = (a_i, b_i + 1)$, $(2a_i, 2b_i)$, or $(a_i + 1, b_i)$, depending on the hash value.) Eventually, this walk must repeat. If $h_i = h_j$, then

$$x \equiv \frac{a_j - a_i}{b_i - b_j} \pmod{n}.$$

If $b_i - b_j$ is relatively prime to n (which is very likely if n is prime), this gives x. Figure 1 illustrates the rho method walk.

Rather than store all of the steps to detect a collision, one may simultaneously compute h_i and h_{2i}, and continue around the cycle until they agree. Assuming that this map behaves as a random walk, $O(\sqrt{n})$ steps will suffice to find a repeat.



Generic Attacks Against DLP. Fig. 1 Rho method walk, with a collision at $h_i = h_j$

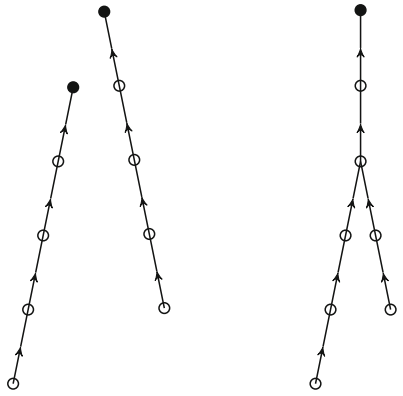
The rho method has two main drawbacks. One is that it is difficult to parallelize. Having k processors do random walks only results in an $O(\sqrt{k})$ speedup since the different walks are independent, and the probability of one of k cycles of length l having a collision is much less than one cycle of length kl. Another is that after the collision occurs, many more steps around the cycle are needed before the collision is detected. Parallelized collision search [3] is a variant of the rho method which fixes both problems.

Designate a small fraction of elements of G *distinguished points*, say ones for which the last several bits of the representation of the element are all zero. Then a walk will begin at a random point, proceed as for the rho method, and end when a distinguished point is hit. That point is saved, along with the starting point of the path, and then a new walk is begun at a new random point. When a distinguished point is hit for the second time, the collision will with high probability determine x.

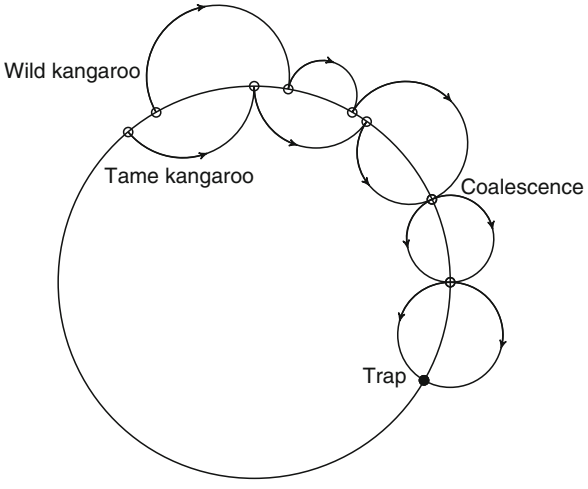
By picking the right fraction of elements of G to be distinguished points, one may ensure that not too much memory is needed to store the paths, and not much time is wasted after a collision occurs. Also, this algorithm may be trivially parallelized, with a linear speedup. For an overview of parameter choices see Teske [7]. Figure 2 illustrates this method.

Another method due to Pollard also uses a random walk in G. In the *kangaroo method*, the steps are limited: $h \longrightarrow hg^{s(h)}$, where the hop length $s(h)$ is a pseudorandom function of h with values between 1 and $\sqrt{n}$.

The idea is to start from two points, say g (the "tame" kangaroo, since its discrete logarithm is known at all times) and y (the "wild" kangaroo), and alternately take hops with length determined by $s(h)$. Set "traps" when a kangaroo hits a distinguished point. If the wild kangaroo and tame kangaroo paths meet or "coalesce" at any point, they will take the same hops from then on, so any traps encountered



Generic Attacks Against DLP. Fig. 2 Parallelized collision search paths, with three distinguished points and one collision



Generic Attacks Against DLP. Fig. 3 Kangaroo method paths, with one distinguished point and collision

by one after they coalesce will also so be encountered by the other. When one reaches a trap that the other one hit, the resulting collision determines x.

The main advantage of the kangaroo method is when x is known to be in a certain range, say $[0, L]$ for some $L \ll |G|$. In that case, one may start the tame kangaroo from $g^{L/2}$, and the wild kangaroo from y. A collision is likely before getting far out of $[0, L]$, and so this will take $O(\sqrt{L})$ time. See Fig. 3 for an illustration.

Experimental Results

In 1997, Certicom issued a series of ►ECC challenges. The ►elliptic curve discrete logarithm problems posed as challenges ranged from easy (curves over 79-bit fields) to very difficult (359-bit fields). The largest challenge problems solved to date are curves over 109-bit fields. A group lead by Monico solved the challenge for a curve over a prime field in 2002 and a curve over $GF(2^{109})$ in 2004, using parallelized collision search. Work has begun on an attack on a challenge curve over $GF(2^{131})$ [1].

Recommended Reading

1. Bailey DV, Baldwin B, Batina L, Bernstein DJ, Birkner P, Bos JW, van Damme G, de Meulenaer G, Fan J, Güneysu T, Gurkaynak F, Kleinjung T, Lange T, Mentens N, Paar C, Regazzoni F, Schwabe P, Uhsadel L (2009) The certicom challenges ECC2-X. In: Workshop record of SHARCS 2009: special purpose hardware for attacking cryptographic systems, pp 51–82
2. Nechaev VI (1994) On the complexity of a deterministic algorithm for a discrete logarithm. Math Zametki 55:91–101
3. van Oorschot PC, Wiener MJ (1999) Parallel collision search with cryptanalytic applications. J Cryptol 12:1–28
4. Pollard JM (1978) Monte Carlo methods for index computation (mod p). Math Comput 32:918–924
5. Shanks D (1969) Class number, a theory of factorization, and genera. In: 1969 Number Theory Institute, proceedings of symposia in pure mathematics, vol XX, State University of New York, Stony Brook, NY. American Mathematical Society, Providence, pp 415–440
6. Shoup V (1997) Lower bounds for discrete logarithms and related problems. In: Fumy W (ed) Advances in cryptology EUROCRYPT'97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 256–266
7. Teske E (2001) Square-root algorithms for the discrete logarithm problem (a survey). In: Public-key cryptography and computational number theory. de Gruyter, Berlin, pp 283–301

Generic Model

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

Related Concepts

►Random Oracle Model; ►Security Proofs; ►Standard Model

Definition

The generic group model is an idealized cryptographic model, where the adversary is only given access to a randomly chosen encoding of a group, instead of efficient encodings, such as those used by the finite field or elliptic curve groups used in practice. The model includes an oracle that executes the group operation. This oracle takes two encodings of group elements as input and outputs an encoding of a third element. The generic model is sometimes criticized as being an excessive idealization of cryptographic reality. The generic group model suffers from some of the same problems as the random oracle model.

Namely, it has been shown that there exist cryptographic schemes which are provable secure in the generic group model, but which are trivially insecure once the random group encoding is replaced with any efficiently computable instantiation of the encoding function.

Recommended Reading

1. Rogaway P (2004) On the role definitions in and beyond cryptography. In: Maher MJ (ed) ASIAN 2004. LNCS, vol 3321. Springer, Heidelberg, pp 13–32

Genetic Code

►DNA

Geometry of Numbers

►Lattice

GMAC

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

Synonyms

Galois message authentication code

Related Concepts

►Finite Fields; ►MAC Algorithms

Definition

GMAC is a ►MAC algorithm based on a universal hash function that uses polynomial evaluation in a finite field of characteristic two. The polynomial with as coefficients the message blocks is evaluated in a secret key. The result is encrypted using a block cipher (e.g., ►AES) in Counter Mode.

Background

Information theoretic authentication was developed in the 1970s by Simmons [10] and Carter and Wegman [3, 11]. The polynomial construction has been proposed by several authors including Bierbrauer et al. [2] and den Boer [4] in 1993. GMAC is a polynomial authentication code over the finite field $\mathrm{GF}(2^n)$; the specification of GMAC is part of the GCM mode for ►authenticated encryption that was standardized in 2007 by NIST [8, 9].

Theory

►GMAC is a randomized ►MAC algorithm. In contrast to ►CBC-MAC, GMAC computations can be parallelized.

The encryption with the block cipher E using the key K will be denoted with $E_K(.)$. An n-bit string consisting of zeros will be denoted with 0^n. The length of a string x in bits is denoted with $|x|$. The function $\mathrm{MSB}_m(.)$ selects the leftmost m bits of its input.

GMAC uses a block cipher with block length $n = 128$ bits and key length $k \geq 128$ bits. GMAC operates on messages of bitlength at most $2^{64} - 1$. It requires an IV with $1 \leq |IV| \leq 2^{64} - 1$. It is critically important for the security of GMAC that an IV value is never reused with the same key.

First the value H is computed as $H \leftarrow E_K(0^{128})$. If $|IV| = 96$, the masking key K_1 is defined as $E_K(IV|0^{31}|1)$ (for other lengths of the IV, see [9]). Subsequently the message x is right-padded with 0 bits until its length is a multiple of 128. Next $|x|$ is represented as a 64-bit string and concatenated with 0^{64}; the resulting value is appended to the message.

The padded message x' is now divided into t' blocks of 128 bits denoted with x'_i, $1 \leq i \leq t'$. All inputs to the GMAC function (H, K_1, and the x'_i) are now considered as elements of $\mathrm{GF}(2^{128})$. GMAC can be described as follows:

$$\mathrm{GMAC}_K(x) = \mathrm{MSB}_m\left(K_1 + \sum_{i=1}^{t'} x'_i \cdot H^{t'+1-i}\right).$$

If the block cipher is pseudorandom and the IV is never reused, the security of GMAC is well understood: The probability of sending an acceptable tag without having observed a text/tag pair (the impersonation probability) equals 2^{-m}, while the probability of forging a text/tag pair after having observed one pair (the substitution probability) is equal to $t'/2^{128}$; this imposes restrictions on the message length. These are general results for polynomial MAC algorithms [2, 4]. In GMAC however, the key H is reused many times; McGrew and Viega provide an analysis that take into account the reuse of H and the details of the GMAC construction (such as the padding). The GMAC standard imposes that a single key K can only be used with at most 2^{32} messages, in order to guarantee a forgery probability less than 2^{-32}.

Joux has pointed out that a repeated IV results in a trivial key recovery [7]. Ferguson has demonstrated that making m too small brings security risks. Handschuh and

Preneel discuss some further issues related to partial key leakage [6]. This shows that in spite of the fact that one can prove elegant bounds on the forgery probability, GMAC is perhaps less robust that one would hope for.

One way to make polynomial MAC algorithms more robust is to use a different key H for each message; this approach has been taken in the SNOW 3G specification [1].

Recommended Reading

1. 3GPP TS 35.216 (2006) Specification of the 3GPP confidentiality and integrity algorithms UEA2 & UIA2; Document 2: SNOW 3G specification, Mar 2006
2. Bierbrauer J, Johansson T, Kabatianskii G, Smeets B (1994) On families of hash functions via geometric codes and concatenation. In: Stinson D (ed) Advances in cryptology. Proceedings of the Crypto 1993. Lecture notes in computer science, vol 773. Springer, Berlin, pp 331–342
3. Carter JL, Wegman MN (1979) Universal classes of hash functions. J Comput Syst Sci 18:143–154
4. den Boer B (1993) A simple and key-economical unconditional authentication scheme. J Comput Sec 2:65–71
5. Ferguson N (2005) Authentication weaknesses in GCM. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/CWC-GCM/Ferguson2.pdf
6. Handschuh H, Preneel B (2008) Key-recovery attacks on universal hash function based MAC algorithms. In: Wagner D (ed) Advances in cryptology. Proceedings of the Crypto 2008. Lecture notes in computer science, vol 5157. Springer, Berlin, pp 144–161
7. Joux A (2006) Authentication failures in NIST version of GCM. http://csrc.nist.gov/CryptoToolkit/modes/
8. McGrew DA, Viega J (2004) The security and performance of the Galois/counter mode (GCM) of operation. In: Canteaut A (ed) Proceedings of the Indocrypt 2004. Lecture notes in computer science, vol 3348. Springer, Berlin, pp 343–355
9. National Institute of Standards and Technology (NIST) (2007) Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. SP 800-38D (earlier drafts published in May 2005, Apr 2006, June 2007). National Institute of Standards and Technology, Gaithersburg
10. Simmons GJ (1991) A Survey of information authentication. In: Simmons GJ (ed) Contemporary cryptology: the science of information integrity. IEEE Press, Piscataway, pp 381–419
11. Wegman MN, Carter JL (1981) New hash functions and their use in authentication and set equality. J Comput Syst Sci 22(3): 265–279

GMR Signature

Gerrit Bleumer
Research and Development, Francotyp Group,
Birkenwerder bei Berlin, Germany

Related Concepts

►Digital Signature; ►Forgery

Definition

The GMR Signature Scheme was invented by Goldwasser et al. [1, 2]. In their landmark publication [1], they presented a formal framework of security definitions of cryptographic signature schemes (►digital signature schemes and ►public key cryptography) and proposed the GMR Signature Scheme, which, under the assumption that ►integer factoring is hard, is provably secure by their strongest security definition, i.e., it resists ►existential forgery under an adaptive chosen message attack.

Background

The value of the GMR Signature Scheme is not in its practical use, but in showing the existence of a provably secure signature scheme. Other more efficient signature schemes such as ►RSA digital signature scheme, *DSA*, or *ECDSA* (►digital signature standard) are memoryless. In order to produce a signature, one does not need to memorize any previously produced signatures (in whole or in part). Also, the computational effort of signing and verifying and the length of signatures are independent of the number of signatures a signer has produced before. In contrast, the GMR Signature Scheme is not memoryless, i.e., each signature a signer produces depends on every other signature the signer has produced before. Even worse, the computation time to produce and to verify a signature and the length of a signature increase (logarithmically) with the number of signatures that a signer has produced before. At the cost of some memory, the average signing performance can be made independent of the number of signatures that a signer has produced before. In essence, the overall performance of the GMR Signature Scheme (in terms of time and memory) decreases steadily over the lifetime of each signing key pair.

Theory

In a nutshell, the GMR Signature Scheme works as follows: a signer who anticipates to produce 2^b signatures constructs a key pair for security parameter k as follows: The public verifying key consists of four components:

1. The maximum number $B = 2^b (b \in N_0)$ of messages to be signed.
2. Two randomly chosen ►Blum integers $n_1 = p_1 q_1$ and $n_2 = p_1 q_2$, i.e., each the product of two ►prime numbers both congruent to 3 modulo 4, but not congruent to each other modulo 8, such that n_1 and n_2 are each of length k-bit.
3. The two infinite families of pairwise ►claw-free trapdoor permutations (►trap-door one-way functions) $f_{i,n}(x) = \pm 4^{\mathrm{rev}(i)} x^{2^{\mathrm{len}(i)}} \bmod n$ for $n = n_1$ and $n = n_2$,

where the function rev$(\cdot)$ takes a bit string $i \in \{0,1\}^+$ and returns the integer represented by the bits of i in reversed order, and the function len$(\cdot)$ takes a bit string $i \in \{0,1\}^+$ and returns its number of bits. The sign plus or minus is selected such that the image of f_{inn} is positive and smaller than $n/2$.

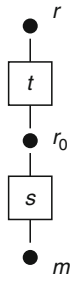
4. A randomly chosen integer r in the domain of $f_{i,n_1}(\cdot)$.

The private signing key consists of the primes *p1, q1, p2, q2*, which allow to efficiently compute the inverse permutations $f^{-1}_{i,n_1}(y)$ and $f^{-1}_{i,n_2}(y)$.

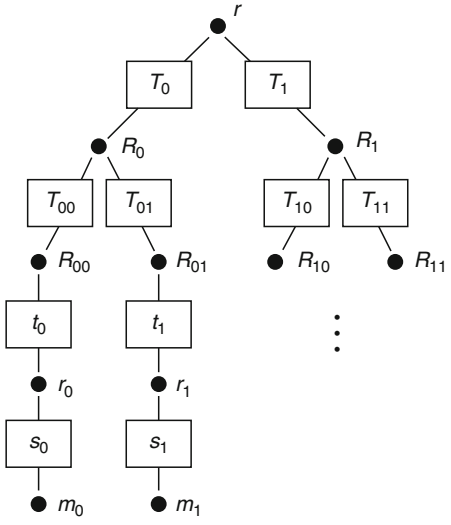
Let us first consider the simple case of signing only one message m, i.e., $B = 1$ and $b = 0$. The signer chooses a random element r_0 from the domain of $f_{i,n_1}(\cdot)$ and computes the signature $(t,\ r_0,\ s)$ for m such that:

$$t = f^{-1}_{r_0,n_1}(r) \qquad \text{and} \qquad s = f^{-1}_{m,n_2}(r_0). \tag{1}$$

The signature $(t,\ r_0,\ s)$ is displayed in Fig. 1 as a chain connecting the signer's root element r with the actual message m.



GMR Signature. Fig. 1 Signing one message



GMR Signature. Fig. 2 Signing more than one message

A verifier checks the signature by computing from the message m up to the signer's root element r as follows: First check that $f_{m,n_2}(s) = r_0$, and then check that $f_{r_0,n_1}(t) = r$.

If there are $B = 2^b > 1$ messages to be signed, the signer expands the root element r from the case $B = 1$ above into a binary authentication tree (Merkle [3]) with B leaves $r_0,\ r_1, \ldots,\ r_{B-1}$. All nodes of the authentication tree except for the root r are chosen uniformly at random from the domain of $f_{i,n_1}(\cdot)$ analogously to r_0 in (1) above. Each node R is unforgeably tied to both its children R_0 and R_1 by computing a tag $f^{-1}_{R_0 \| R_1, n_1}(R)$ analogously to how the tag t was computed in (1) above. Finally, the message $m_j (0 \le j \le B-1)$ is unforgeably tied to r_j, which hangs off of the jth leaf of the authentication tree by computing a tag $s_j = f^{-1}_{m_j,\, n_2}(r_j)$ analogously to how the tag s was computed in (1) above. The signature for message m_j then consists of (a) the sequence of b nodes from the root r down to item r_j, (b) the b tags associated to these nodes (analogously to t above), and (c) the tag s_j (analogously to s above). The authentication tree and associated tags for the case $B = 8,\ b = 3$ is depicted in Fig. 2 as a binary tree connecting the signer's root element r with the B messages $m_0,\ m_1, \ldots,\ m_{B-1}$.

A verifier can check the signature in reverse order beginning from the message m_j and working back up to the root element r of the signer.

Obviously, the authentication tree need not be precomputed in the first place, but can be built step by step as the need arises. For example, signing the first message requires one to compute the complete path from the root r down to the item r_0, but signing the second message only requires one to build one more leaf of the authentication tree and the next item r_1, while all previously built nodes of the authentication tree can be reused. So, on average, each signature requires one to build two new nodes of the authentication tree and one additional item. A complete security analysis is given in [2].

Recommended Reading

1. Goldwasser S, Micali S, Rivest RL (1984) A "Paradoxical" solution to the signature problem. 25th symposium on foundations of computer science (FOCS) 1984. IEEE Computer Society, Washington, DC, pp 441–448
2. Goldwasser S, Micali S, Rivest RL (1988) A digital signature scheme secure against adaptive chosen-message attacks. SIAM J Comput 17(2):281–308
3. Merkle R (1982) Secrecy authentication, and public-key systems. Ph.D. dissertation, Electrical Engineering Department, ISL SEL 79-017k, Stanford University, Stanford

Goldwasser–Micali Encryption Scheme

Kazue Sako
NEC, Kawasaki, Japan

Related Concepts

►Probabilistic Encryption; ►Public Key Cryptography; ►Public Key Encryption Scheme; ►Semantic Security

Definition

Goldwasser–Micali Encryption scheme is an encryption scheme that was proposed in the paper "probabilistic encryption" in 1984.

Background

The Goldwasser–Micali encryption scheme (►Public Key Cryptography) is the first encryption scheme that achieved ►Semantic Security against a passive adversary under the assumption that solving the quadratic residuosity problem is hard. The scheme encrypts 1 bit of information, and the length of the resulting ciphertext equals the length of the composite number n used in scheme, which is typically thousands of bits long.

Theory

In the Goldwasser–Micali encryption scheme, a public key is a number n, that is a product of two ►prime numbers, say p and q. Let Y be a quadratic nonresidue modulo n (►Quadratic Residue and ►Modular Arithmetic), whose ►Jacobi Symbol is 1. The decryption key is formed by the prime factors of n.

The Goldwasser–Micali encryption scheme encrypts a bit b as follows. One picks an integer $r(1 < r < n - 1)$ and outputs $c = Y^b r^2 \bmod n$ as ciphertext. That is, c is quadratic residue if and only if $b = 0$. Therefore a person knowing the prime factors of n can compute the quadratic residuosity of the ciphertext c, thus obtaining the value of b.

If the quadratic residuosity problem is hard, then guessing the message from the ciphertext is equivalently hard.

Recommended Reading

1. Goldwasser S, Micali S (1984) Probabilistic encryption. J Comp Syst Sci 28:270–299

Golomb's Randomness Postulates

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

Related Concepts

►Autocorrelation; ►Gap; ►Maximal-Length Linear Sequences; ►Pseudo-Noise Sequences (PN-Sequences); ►Run; ►Sequences

Definition

Golomb's randomness postulates are three properties that one expects to find in random sequences.

Theory and Applications

No finite sequence constructed by a ►linear feedback shift register is a truly random sequence. Golomb [1] introduced the notion of a *pseudo-random* sequence for a periodic binary sequence that satisfies three randomness postulates. These postulates reflect the properties one would expect to find in a random sequence.

A ►run in a binary sequence is a set of consecutive 0s or 1s. A run of 0s is denoted a ►gap and a run of 1s is denoted a *block*. A gap of length k is a set of k consecutive 0s flanked by 1s. A block of length k is a set of k consecutive 1s flanked by 0s. A run of length k is is a gap of length k or a block of length k. In this terminology, the three randomness postulates of a periodic sequence are as follows:

R-1 In a period of the sequence, the number of 1s and the number of 0s differ by at most one.

R-2 In every period, half the runs have length 1, one-fourth have length 2, one-eight has length 3, etc., as long as the number of runs so indicated exceeds 1. Moreover, for each of these lengths, there are equally many gaps and blocks.

R-3 The out-of-phase ►autocorrelation of the sequence always has the same value.

The R-2 postulate implies the R-1 postulate, but otherwise the postulates are independent, which follows from the observation that there exist sequences that satisfies some but not all of the postulates. Some examples are:

Example 1 *The m-sequence 00001001011001111100011011101 01 of period 31 is an R-1, R-2, and R-3 sequence.*

Example 2 *The sequence 00100011101 of period 11 is an R-1 sequence. It is also an R-3 sequence since the out-of-phase*

autocorrelation is equal to –1. However, the sequence is not an R-2 sequence since there are two blocks of length 1 but only one gap of length 1.

Example 3 *The sequence 0000010001101 of period 13 is an R-3 sequence with a constant out-of-phase autocorrelation being 1, but the sequence violates the R-1 and R-2 conditions.*

Example 4 *The sequence 01001101 of period 8 is an R-1 and R-2 sequence but not an R-3 sequence.*

The three randomness postulates are inspired by the properties obeyed by ►maximal-length linear sequences (*m*-sequences). Also, they can be interpreted as properties of flipping a perfect coin. R-1 says that heads and tails occur about equally often, R-2 says that after a run of *n* heads (tails) there is a probability 1/2 that the run will end with the next coin-flip. R-3 is the notion of independent trials. Knowing the outcome of a previous coin-flip gives no information of the current coin-flip.

Any sequence obeying both R-1 and R-3 can be shown to have a value of the out-of-phase autocorrelation being –1 and therefore the period must be odd. Sequences which obey the randomness postulates of Golomb are sometimes also called ►pseudo-noise sequences (PN-sequences).

For the extension of the Golomb's randomness postulates to nonbinary sequences, see [2].

Recommended Reading

1. Golomb SW (1967) Shift register sequences. Holden-Day series in information systems. Holden-Day, San Francisco. Revised ed., Aegean Park Press, Laguna Hills, 1982
2. Golomb SW, Gong G (2005) Signal design for good correlation – for wireless communication, cryptography, and radar. Cambridge University Press, Cambridge

GOST

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

Related Concepts

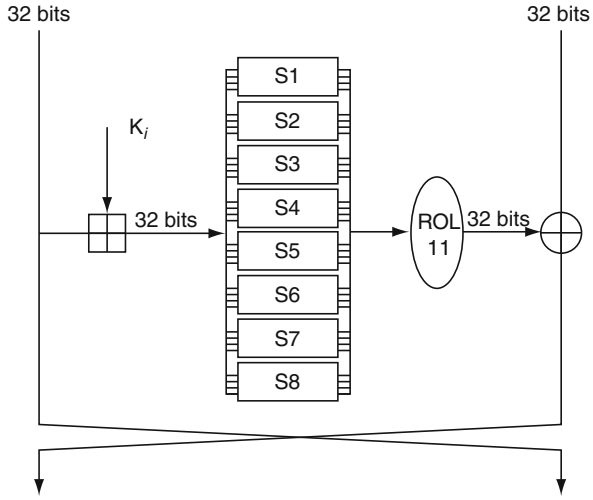
►Block Ciphers; ►Feistel Cipher

GOST is an encryption algorithm adopted as a standard by the former Soviet Union in 1989 [5]. The specifications, translated from Russian in 1993, describe a DES-like 64-bits ►block cipher (►Data Encryption Standard) and specify four ►modes of operation.

The GOST encryption algorithm is a very simple 32-round ►Feistel cipher. It encrypts data in blocks of 64 bits and uses a 256-bit secret key. The 32-bit *F*-function used in the Feistel construction consists of three transformations. First, a 32-bit subkey is mixed with the data using an addition modulo 2^{32}. The result is then split into 4-bit segments, fed in parallel to eight 4 × 4-bit S-boxes. Finally, the output values are merged again and rotated over 11 bits. The *key schedule* (►Block Cipher) of GOST is also particularly simple: the 256-bit secret key is divided into eight 32-bit words and directly used as subkeys in rounds 1–8, 9–16, and 17–24. The same eight subkeys are reused one more time in rounds 25–32, but in reverse order Fig. 1.

A remarkable property of the GOST standard is that the eight S-boxes are left unspecified. The content of these lookup tables is considered to be a secondary long-term secret key, common to a network of computers, and selected by some central authority. The set of S-boxes can be strong or weak, depending on the authority's intention. The S-boxes can even be hidden in an encryption chip, thus keeping them secret to the users of the device. As explained in a short note by Saarinen [3], recovering the secret S-boxes would not be very hard, however. If a user is allowed to select the 256-bit key of the cipher, he can easily mount a "chosen key attack" and efficiently derive the secret contents of the lookup tables.

The best attacks on GOST exploit the simplicity of its key schedule. The first attack in open literature is a ►related key attack by Kelsey et al. [2]. Biryukov and Wagner [1] have shown that the cipher is vulnerable to



GOST. Fig. 1 One round of GOST

►slide attacks. Their cryptanalysis breaks 20 rounds out of 32, but also reveals ►weak key classes for the full cipher. Finally, Seki and Kaneko [4] have applied ►differential cryptanalysis to reduced-round versions of GOST. Combined with a related-key approach, the attack breaks 21 rounds.

Recommended Reading

1. Biryukov A, Wagner D (2000) Advanced slide attacks. In: Preneel B (ed) Advances in cryptology – EUROCRYPT 2000. Lecture notes in computer science, vol 1807. Springer, Berlin, pp 589–606
2. Kelsey J, Schneier B, Wagner D (1996) Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. In: Koblitz N (ed) Advances in cryptology – CRYPTO 96. Lecture notes in computer science, vol 1109. Springer, Berlin, pp 237–251
3. Saarinen M-J (1998) A chosen key attack against the secret S-boxes of GOST (unpublished)
4. Seki H, Kaneko T (2001) Differential cryptanalysis of reduced rounds of GOST. In: Stinson DR, Tavares SE (eds) Selected areas in cryptography, SAC 2000. Lecture notes in computer science, vol 2012. Springer, Berlin, pp 315–323
5. Zabotin IA, Glazkov GP, Isaeva VB (1989) Cryptographic protection for information processing systems: cryptographic transformation algorithm. Technical Report, Government Standard of the USSR, GOST 28147–89 (trans: Malchik A, with editorial and typographic assistance of Diffie W)

Grant Option

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

Related Concepts

►Privileges in SQL; ►SQL Access Control Model

Definition

The *grant option* in SQL states that a subject (authorization identifier) receiving a privilege has the additional authority to grant such a privilege to others.

Theory

The `GRANT` and `REVOKE` commands used for managing ►privileges in SQL (►SQL access control model) can include the `WITH GRANT OPTION` and `GRANT OPTION FOR` clauses, respectively. If `WITH GRANT OPTION` is specified in a `GRANT` command, the recipient of the privilege can grant it to others. If `GRANT OPTION FOR` is specified in a `REVOKE` command, only the grant option for the privilege (and not the privilege itself) is revoked [1–3].

Recommended Reading

1. De Capitani di Vimercati S, Samarati P, Jajodia S (2001) Database security. In Marciniak J (ed) Wiley encyclopedia of software engineering. Wiley, New York
2. Samarati P, De Capitani di Vimercati S (2001) Access control: Policies, models, and mechanisms. In Focardi R, Gorrieri R (eds) Foundations of Security Analysis and Design. LNCS, vol 2171. Springer, Berlin
3. Database Language SQL (2008) ISO International Standard, ISO/IEC 9075–*:2008

Greatest Common Divisor

Scott Contini
Silverbrook Research, New South Wales, Australia

Synonyms

GCD; Greatest common factor

Related Concepts

►Euclidean Algorithm; ►Least Common Multiple; ►Number Theory; ►Relatively Prime

Definition

The *greatest common divisor* (gcd) of a set of positive integers $\{a_1, \ldots, a_k\}$ is the largest integer that divides every element of the set. This is denoted $\gcd(a_1, \ldots, a_k)$ or sometimes just $(a_1, \ldots, a_k)$.

Background

Approximately in 300 BC, the Greek mathematician Euclid described a method for computing the greatest common divisor in Book 7 of *Elements*. Known as the ►Euclidean algorithm, it is based upon the fact that the greatest common divisor of two integers is preserved under subtraction.

Theory

An important property of the greatest common divisor is that it can always be written as an integer linear combination of the elements of the set. In other words, there exist integers $x_1, \ldots, x_k$ such that $\sum_{i=1}^{k} a_i \cdot x_i = \gcd(a_1, \ldots, a_k)$.

For example, $\gcd(21, 91) = 7$ because 7 divides both 21 and 91, and no integer larger than 7 divides both of these values. A linear combination that represents the gcd is given by $x_1 = -4$ and $x_2 = 1$ since $21 \cdot (-4) + 91 \cdot 1 = 7$.

For the case of $k = 2$, the extended ►Euclidean algorithm can be used to efficiently find the integer linear

combination. If the gcd is 1, then the integers are said to be ►relatively prime. The greatest common divisor is related to the ►least common multiple (lcm) by the relation $\gcd(a_1, a_2) \cdot \mathrm{lcm}(a_1, a_2) = a_1 \cdot a_2$.

Greatest Common Factor

►Greatest Common Divisor

Gröbner Basis

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

Synonyms
Standard basis

Related Concepts
►Multi-variate Cryptography

Definition
A Gröbner (or a standard) basis G is a particular kind of generating subset of an ideal I in a polynomial ring R. A Gröbner basis can be seen as a multivariate, nonlinear generalization of the Euclidean algorithm for computation of univariate GCDs or as a generalization of Gaussian elimination.

A Gröbner basis G is characterized by the fact that any multivariate division of any $p \in I$ by G gives 0. Here, division is understood as being relative to some monomial order (the obtained basis depends on the monomial ordering chosen).

Gröbner bases always exist and can be effectively obtained for any I starting with a generating subset. The best-known algorithm for computing Gröbner bases is Buchberger's algorithm. Gröbner bases are useful in cryptanalysis, mostly for attacking multivariate cryptosystems.

Open Problems
Find more efficient algorithms to compute Gröbner bases.

Recommended Reading
1. Buchberger B, Winkler F (eds) (1998) Gröbner bases and applications. London mathematical society lecture note series 251. Cambridge University Press. Cambridge

Group

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

Related Concepts
►Field; ►Generator; ►Order; ►Subgroup

Definition
A *group* is a set of elements with a certain well-defined mathematical structure under a *group operation*.

Theory
A *group* $G = (S, \circ)$ is defined by a set of elements S and a group operation $\circ$ that satisfy the following *group axioms*:

- *Closure*: For all $x, y \in S, x \circ y \in S$.
- *Associativity*: For all $x, y, z \in S, (x \circ y) \circ z = x \circ (y \circ z)$.
- *Identity*: There exists an *identity element*, denoted I, such that for all $x \in S, x \circ I = I \circ x = x$.
- *Inverse*: For all $x \in S$, there exists an *inverse* y such that $x \circ y = y \circ x = I$.

A group is *commutative* (also called *Abelian*) if the group operation does not depend on the ordering of the elements, i.e., if for all $x, y \in S, x \circ y = y \circ x$. A group is *cyclic* if it has a single ►generator, i.e., an element g such that every element of the group can be obtained by repeated composition with g *and its inverse*, starting with the identity element. The ►order of a group G, denoted #G, is the number of elements in G.

Groups commonly employed in cryptography include the following:

- A *multiplicative group modulo a prime*, where S consists of the set of integers (i.e., residue classes) modulo a prime p, excluding 0, and the group operation is multiplication. This group is typically denoted by $\mathbf{Z}_p^*$, where $\mathbf{Z}_p$ denotes the integers modulo p, and * denotes that the group operation is multiplication and 0 is excluded. The order of $\mathbf{Z}_p^*$ is $p - 1$.

(This group is the same as the multiplicative group of the ►finite field F_p.)

- A ►subgroup of a multiplicative group modulo a prime, i.e., a subset of elements in $\mathbf{Z}_p^*$ that is closed under multiplication. Typically, the subgroup is selected so that its order is a prime number. For

instance, the Digital Signature Algorithm (►Digital Signature Standard) operates in a subgroup of order q of $\mathbf{Z}_p^*$, where p and q are large primes and q divides $p-1$.
- A subgroup of the multiplicative group of an ►extension field $\mathbf{F}_{q^d}$.
- A subgroup of an ►elliptic curve group. If the elliptic curve group has a prime order, then the subgroup is the same as the elliptic curve group.

All these examples are cyclic and commutative. An elliptic curve group itself may be noncyclic, but the subgroup of interest itself is typically cyclic.

A group is formally denoted by both the set and the group operation, i.e., $(S,^\circ)$, but in cryptography, sometimes only the set S is denoted and the group operation is implied. In cryptography, the group operation is typically either denoted by multiplication or addition. In the former case, repeated application of the group operation is denoted by *exponentiation* (e.g., g^a); in the latter, it is denoted by *scalar multiplication* (e.g., aP where P is the group element).

Applications

Groups are primarily associated with ►public-key cryptography, but they also have some applications to ►symmetric cryptosystems. For instance, several researchers investigated whether the set of keys in the ►Data Encryption Standard forms a group [2, 3], which could significantly weaken the standard; the set of keys does not.

Braid groups are a notable example of a noncommutative group in public-key cryptography [1, 4, 5].

Recommended Reading

1. Anshel I, Anshel M, Goldfeld D (1999) An algebraic method for public-key cryptography. Math Res Lett 6:287–291
2. Campbell KW, Wiener MJ (1992) DES is not a group. In: Brickell EF (ed) Advances in cryptology – CRYPTO'92. Lecture notes in computer science, vol 740. Springer, Berlin, pp 512–520
3. Kaliski Jr BS, Rivest RL, Sherman AT (1988) Is the data encryption standard a group? J Cryptol 1:3–36
4. Ko KH, Lee SJ, Cheon JH, Han JW, Kang J-S, Park C (2000) New public-key cryptosystem using braid groups. In: Bellare M (ed) Advances in cryptology – CRYPTO 2000. Lecture notes in computer science, vol 1880. Springer, Berlin, pp 166–183
5. Lee E, Je HP (2003) Cryptanalysis of the public-key encryption based on braid groups. In: Biham E (ed) Advances in cryptology – EUROCRYPT 2003. Lecture notes in computer science, vol 2656. Springer, Berlin, pp 477–490

Group Key Agreement

Mike Burmester
Department of Computer Science, Florida State University, Tallahassee, FL, USA

Synonyms

Conference key agreement; Conference keying; Group key distribution; Group key exchange

Related Concepts

►Entity Authentication; ►Key Agreement; ►Public-Key Encryption Scheme

Definition

Group Key Agreement (GKA) is an extension of two-party ►key agreement to groups of $n \geq 2$ parties: it allows a group consisting of several parties to establish a common *session key* (►Key) or *conference key* over an unprotected network.

Background

Ingemarsson, Tang, and Wong presented in 1982 the first GKA protocol [20] based on the two-party ►Diffie–Hellman key agreement protocol [19]. This was followed by the GKA protocols of Koyama and Ohta [24], Blundo et al. [6], and Burmester and Desmedt [15]. Since then a large number of research papers on GKA and on securing GKA protocols have been presented due mainly to the distributed and dynamic nature of GKA and to the security challenges that have to be resolved – see e.g., [1, 5, 7–13, 16, 18, 23, 26–31, 33, 34], and the tripartite Diffie–Helman key agreement protocol [21]. Most of these protocols combine aspects of the Diffie–Hellman key agreement with other cryptographic primitives to support security.

Some of these protocols are briefly described in the following, starting with protocols for small groups that are secure against passive attacks. Authenticated GKA is then considered, followed by a discussion on insider attacks and provable security. Finally, GKA for large groups is considered. To start with, the basic requirements for GKA are reviewed.

Requirements

Given the openness of most networking systems, it is important that key agreement is achieved efficiently and securely. Other requirements include, scalability, freshness of session keys, forward secrecy, rekeying and, of course, reliability.

Efficiency: The efficiency of a GKA protocol is measured in terms of its ►computational complexity (usually the number of modular exponentiations), its *communication* complexity (the number of communicated messages), and its *round* complexity (the number of rounds of communication exchanges). The complexity of a multiparty protocol involving n parties is *scalable* if it is bounded by $\mathcal{O}(\log n)$ (►$\mathcal{O}$-notation). Scalability is particularly important when the number of parties in the group is large.

Security: Security involves both the ►privacy of session keys and ►authentication. Privacy (*key secrecy*–►Secrecy) requires that in a passive attack it is hard to *distinguish* the session key from a random key. Passive attacks are eavesdropping attacks in which the adversary has access only to traffic communicated in public. There is a stronger version of privacy that requires that it is hard to *compute* the session key in a passive attack. Freshness of session keys is required to prevent *known-key attacks* (►Related Key Attack) in which the adversary succeeds in getting hold of session keys by analyzing large amounts of communication traffic encrypted with the same group key, or by non-cryptographic means. Session keys must therefore be regularly updated.

Forward secrecy: A compromised session key should only affect its session, not jeopardize earlier sessions.

Rekeying: Groups are dynamic, with new parties joining the group or current members leaving the group. Whenever a new group is formed, a new session key must be agreed: otherwise information regarding previous sessions (or future sessions) will become available to those joining (or leaving) the group.

Reliability: The communication channels are reliable (guarantee delivery).

Private Group Key Agreement for Small Groups

GKA protocols that are secure against passive attacks (eavesdropping) are first considered. These typically involve a group $G = \langle g \rangle$ generated by an element g whose order is a k-bit prime q. The group could be a subgroup of the multiplicative group Z_m^* (which consists of all positive integers less than m and relative prime to m), the group of an *elliptic curve* (►Elliptic Curves) or, more generally, any finite group whose operation can be efficiently computed and for which one can establish membership and select random elements efficiently. For convenience the elements of G are called numbers.

The parameters of the group G are selected by a *Trusted Center* (►Trusted Third Party); k is the security parameter. To start with, assume that the number of parties n involved in key agreement is small. That is, $n << q$ $(n = poly(k))$. The protocols described below will involve random selections of elements from G. The notation $x \in_R X$ is used to indicate that x is selected at random (independently) from the set X with uniform distribution.

1. *A basic centralized GKA protocol* [15]. Let $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$, $n << q$, be a group of n parties. A designated member of $\mathcal{U}$ called the *chair*, say U_1, selects the session key $sk \in_R G$.

Round 1. The chair exchanges a key $K_i \in G$ with every party $U_i \in \mathcal{U} \backslash U_1$ by using the ►Diffie–Hellman key agreement protocol.

Round 2. The chair sends to each party $U_i \in \mathcal{U} \backslash U_1$ the number: $X_i = sk/K_i \in G$.

Key Computation. Each party $U_i \in \mathcal{U} \backslash U_1$ computes the key: $sk = X_i \cdot K_i$.

Security and efficiency. Clearly one has forward secrecy: the session keys sk are independent (randomly selected in G). ►Privacy reduces to the ►Diffie–Hellman problem. More specifically, *distinguishing* the session key from a random key in a passive attack (*key secrecy*), is as hard as the ►Decisional Diffie–Hellman problem; *computing* the session key is as hard as the *Computational Diffie–Hellman problem*. The complexity of this protocol is unbalanced: whereas the chair has to exchange Diffie–Hellman keys K_i with each one of the (other) parties of the group and then send the X_i's, the other parties need only exchange a Diffie–Hellman key with the chair. It is possible [16] to share the burden evenly among the parties by arranging the group in a graph tree, with root the chair, and then exchange Diffie–Hellman keys only along root-to-leaf paths. In this case, all members of the group (including the chair) exchange a Diffie–Hellman key with adjacent nodes in the tree (their parent and children). Then the per-user complexity is shared evenly; however, there is now a time delay which is proportional to the height of the tree [16]. The per-user complexity of the tree-based protocol is $\mathcal{O}(\log n)$.

2. *The Ingemarsson–Tang–Wong GKA protocol* [8, 11, 20]. Let $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ be the group of parties arranged logically in a cycle. Here the n-th order protocol is presented (lower order protocols, and in particular the second-order protocol, are vulnerable to passive attacks in which the attacker eavesdrops on *all* communication

channels [20]). This uses subsets $X_i \subset G$, $i = 1, 2, \ldots, n$ defined as follows. Select $r_1, r_2, \ldots, r_n \in_R Z_q$. Define $X_1 = \{x_{1,0} = g, x_{1,1} = g^{r_1}\}$, and recursively $X_i = \{x_{i,0} = (x_{i-1,0})^{r_i}, \ldots, x_{i,i-2} = (x_{i-1,i-2})^{r_i}, x_{i,i-1} = x_{i-1,i-1}, x_{i,i} = (x_{i-1,i-1})^{r_i}\}$, for $1 < i \leq n-1$. Finally $X_n = \{x_{n,0} = (x_{n-1,0})^{r_n}, \ldots, x_{n,i-2} = (x_{n-1,n-2})^{r_n}, x_{n,n-1} = x_{n-1,n-1}\}$. These sets, for the case when $n = 4$, are illustrated in Fig. 1. The protocol has two phases: *up-flow* and *down-flow*. In the up-flow phase each party U_i, $1 \leq i \leq n-1$, sends to the next party U_{i+1} in the cycle the set X_i. In the down-flow phase, the last party U_n broadcasts X_n. The session key is $sk = g^{x_1 x_2 \cdots x_n} \in G$. More specifically:

Phase 1
For $i = 1$ to $n-1$
U_i sends to U_{i+1}: $X_i \subset G$, where the exponent $r_i \in_R Z_q$ is selected by U_i.

Phase 2
U_n broadcasts: $X_n \subset G$, where the exponent $r_n \in_R Z_q$ is selected by U_n.

Key Computation. Each party $U_i \in \mathcal{U}$ computes the key:

$$sk_i = \left(g^{x_1 x_2 \cdots x_{i-1} x_{i+1} \cdots x_n}\right)^{x_i} = g^{x_1 x_2 \cdots x_n} = sk,$$

where $g^{x_1 x_2 \cdots x_{i-1} x_{i+1} \cdots x_n}$ is obtained from X_n.

Remark If all parties adhere to the protocol then each will compute the same key. This protocol exploits the transitivity of the Diffie–Hellman operator: $DH(A, DH(B, C)) = DH(DH(A, B), C)$, for $A, B, C \in G$, where $DH(g^a, g^b) = g^{ab}$.

Security and efficiency. As in the previous protocol one has forward secrecy. Privacy is reduced to the *Group Decisional Diffie–Hellman* (G-DDH) assumption, which is described below.

G-DDH Assumption. Let: $G = \langle g \rangle$ be a group of order a k-bit prime q, $n = poly(k)$, $I_n = \{1, 2, \ldots, n\}$, 2^{I_n} be the powerset of I_n and $\Gamma \subset 2^{I_n} \backslash I_n$. Given the set:

$$\bigcup_{J \in \Gamma} \left(J, g^{\prod_{j \in J} x_j}\right), \quad \text{where } (x_1, \ldots, x_n) \in_R (Z_q)^n,$$

it is hard to distinguish the key $g^{x_1 x_2 \cdots x_n}$ from a random number in G.

X_1	g	g^{r1}		
X_2	g^{r2}	g^{r1}	g^{r1r2}	
X_3	g^{r2r3}	g^{r1r3}	g^{r1r2}	g^{r1r2r3}
X_4	g^{r2r3r4}	g^{r1r3r4}	g^{r1r2r4}	g^{r1r2r3}

Group Key Agreement. Fig. 1 The sets X_i for a group with four members

The complexity of this protocol is $\mathcal{O}(\log n)$. There is a time delay for the $n+1$ steps, and the computation of the sets X_i requires on average $n/2$ exponentiations.

3. *The Burmester–Desmedt GKA protocol* [15]. The group of parties $\mathcal{U} = (U_1, U_2, \ldots, U_n)$ is arranged logically in a cycle. The protocol uses a broadcast channel (this could be replaced by $\binom{n}{2}$ point-to-point channels), and exploits the distributivity of the Diffie–Hellman operator: $DH(A, B \cdot C) = DH(A, B) \cdot DH(A, C)$.

There are two rounds. In the first round, each party $U_i \in \mathcal{U}$ broadcasts a random number $z_i = g^{r_i} \in G$. In the second round, U_i broadcasts the number $X_i = (z_{i+1}/z_{i-1})^{r_i} \in G$. The session key is, $sk = g^{r_1 r_2 + r_2 r_3 + \cdots + r_n r_1} \in G$. More specifically:

Round 1. Each party $U_i \in \mathcal{U}$ selects an exponent $r_i \in_R Z_q$ and broadcasts: $z_i = g^{r_i} \in G$.

Round 2. Each party $U_i \in \mathcal{U}$ broadcasts $X_i = (z_{i+1}/z_{i-1})^{r_i} \in G$, where the indexes are taken in a cycle.

Key Computation. Each party $U_i \in \mathcal{U}$ computes the session key:

$$sk_i = (z_{i-1})^{nr_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i-2} \in G.$$

Remark If all parties adhere to the protocol then each party will compute the same key: $sk = g^{r_1 r_2 + r_2 r_3 + \cdots + r_n r_1}$. Indeed, let $A_{i-1} = (z_{i-1})^{r_i} = g^{r_{i-1} r_i}$, $A_i = (z_{i-1})^{r_i} \cdot X_i = g^{r_i r_{i+1}}$, $A_{i+1} = (z_{i-1})^{r_i} \cdot X_i \cdot X_{i+1} = g^{r_{i+1} r_{i+2}}$, and so on. Then $sk_i = A_{i-1} \cdot A_i \cdot A_{i+1} \cdots A_{i-2} = sk$.

In the special case when there are only two parties, one has $X_1 = X_2 = 1$ and $sk = g^{r_1 r_2 + r_2 r_1} = g^{2 r_1 r_2}$. This is essentially the ►Diffie–Hellman key agreement (in this case there is no need to broadcast X_1, X_2).

Security and efficiency. As in the previous protocol one has forward secrecy. Privacy is reduced to the Decisional Diffie–Hellman problem [23] (computing the session key, in a passive attack, is as hard as the Computational Diffie–Hellman problem [14]). The complexity per-user is $\mathcal{O}(1)$ (constant). This is roughly double the complexity of the two-party Diffie–Hellman protocol: there are two (broadcast) rounds, three exponentiations, and a few multiplications.

Re-keying. Because of it efficiency, to re-key one just needs to re-run the protocol.

Variants. By arranging the connectivity of group $\mathcal{U}$ in different ways to route messages, *e.g.*,in a star, a cycle, or a tree, one gets several GKA variants which can be used to meet specific requirements [15, 16].

Authenticated Group Key Agreement, Provable Security

Active adversaries (Active Attack) control the communication in the network and may impersonate the parties of a group and/or modify their messages. Authenticated GKA protects against such attacks. First consider the case when the adversaries are *outsiders*, and the parties of the group are not corrupted (adhere to the GKA protocol). Bresson, Chevassut, Pointcheval, and Quisquater introduced in 2001 a security model for GKA that addresses *active* (*outsider*) *adversary* attacks – the BCPQ model [11]. This model extends the ►Random Oracle model of Bellare and Rogaway [3, 4] to multiple parties. The BCPQ model has been used extensively for proving the security of GKA protocols. Examples are the protocols presented in [1, 11–13].

Katz and Yung describe [23] a compiler $\mathcal{C}$ which transforms any GKA protocol P that is secure against passive attacks into a protocol $P' = \mathcal{C}(P)$ that is secure against active (outsider) adversaries. The compiler adds only one communication round to protocol P and authenticates (►Authentication) all communication traffic. Security is in the BCPQ model. In this model, each party U is allowed to execute the protocol P many times with several groups of partners (and to interleave the rounds of the executions). Each such execution of P is called an *instance*. The i-th instance executed by U is denoted by Π_U^i.

The following assumptions regarding the inputs P to the compiler $\mathcal{C}$ are made: (a) P is a GKA protocol that is secure against passive attacks (key secrecy); (b) each message sent by instance Π_U^i includes the *identity* of the sender U and a *sequence number* j which starts at 0 and is incremented by 1 each time Π_U^i sends a new message m (i.e., Π_U^i sends $U||j||m$, the concatenation of U, j and m); (c) all messages are broadcast to the entire group $\mathcal{U}$ taking part in the execution of P.

It is easy to see that any GKA protocol $\widetilde{P}$ can readily be converted to a protocol P that satisfies these three assumptions for compiler $\mathcal{C}$. Moreover if $\widetilde{P}$ is secure against passive attacks then P will also be. Finally, the round complexity of P is the same as that of $\widetilde{P}$.

The compiler $\mathcal{C}$ uses a digital signature scheme that is secure against adaptive chosen message attacks. Let $\mathcal{P}$ be the set of potential users of protocol P. Each party $U \in \mathcal{P}$ should have a public/secret signature key pair (PK_U, SK_U). Given protocol P, the compiler now constructs protocol $P' = \mathcal{C}(P)$ as follows:

1. Let $\mathcal{U} = U_1, \ldots, U_n$ be a group of parties that wish to establish a session key and let $\mathcal{U}' = U_1||\cdots||U_n$ (concatenation). Each party U_i selects a random nonce $t_i \in \{0,1\}^k$ and broadcasts $U_i||0||t_i$, where 0 is the (initial) sequence number for this message. After receiving all the broadcast messages, each instance Π_U^i stores $\mathcal{U}'$ and $t' = t_1||\cdots||t_n$ as part of its state information.
2. The parties in group $\mathcal{U}$ now execute protocol P with the following changes:
   - Whenever instance Π_U^i has to broadcast $U||j||m$ in protocol P, instead the instance broadcasts $U||j||m||\sigma$, where σ is the signature of party U on $j||m||\mathcal{U}'||t'$ with the key SK_U.
   - Whenever instance Π_U^i receives message $V||j||m||\sigma$ it checks that: (i) party $V \in \mathcal{U}$ and message m is valid (for the protocols described earlier this means that $z_i, X_i \in G$, or $x_{ij} \in G$); (ii) j is the next expected sequence number for V, and (iii) σ is a valid signature of V on $j||m||\mathcal{U}'||t'$. If any of these fails, Π_U^i aborts the protocol.

Compiler $\mathcal{C}$ can thus be used to convert any GKA protocol that is secure against a passive adversary into a GKA protocol that is secure against active adversaries in the BCPQ model [11].

Authenticated Group Key Agreement: Robustness Against Insider Attacks

Desmedt et al. presented an Authenticated GKE protocol [18] that is robust against two types of malicious insider attacks:

1. *DoS attacks* in which insiders send "bad" messages to cause an honest party to abort, or to cause two honest parties to compute different session keys
2. *Malleability attacks* in which insiders send "bad" messages to bias the probability distribution of the session key, or force it to have some desired value

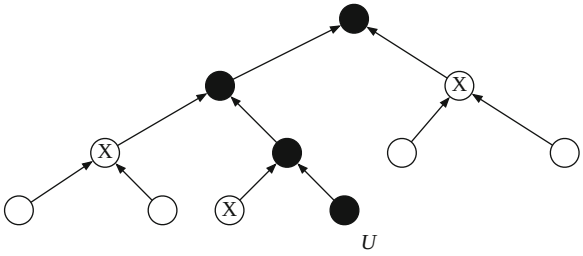
In their threat model, Desmedt et al. assume that the communication channels are reliable and that no more than half of the protocol parties are corrupted. Katz and Shin [22] assume a more general adversarial model in which the adversary has full control of the communication channels (communication is not reliable), and propose a compiler that transforms a secure Authenticated GKE protocol (with no insiders) into an Authenticated GKE protocol that is secure in the UC framework [17] against adversaries that include insiders. Bresson-Manulis [9, 10] extend this model to capture *key control and contributiveness* (guaranteeing *key confirmation*).

To conclude, the case when the number of parties n in the group G is large is considered.

Group Key Agreement for Large Groups

Many emerging applications such as teleconferencing, pay-per-view, and real-time information systems, are based on a group communication model in which data is

broadcast to large dynamically evolving groups. Securing such communication is particularly challenging because of the requirement for frequent rekeying as a result of members leaving the group or new members joining the group. Several GKA protocols for large groups have been proposed in the literature – see e.g., [2, 26–29, 32, 34], and also [31, 33]. These are centralized algorithms that offer scalable solutions to the group key management problem and employ appropriate key graphs that securely distribute rekeying information after members leave or join the group. Following a protocol that employs a key graph tree for scalable rekeying is briefly described.



Group Key Agreement. Fig. 2 An OFT tree: party U possesses the keys of the black nodes and the blinded keys of their siblings "⊗"

(1). *The McGrew–Sherman One-way Function Tree (OFT) Protocol* [29]. Let k be the security parameter and n the number of potential users of the system. Assume $n >> k$ (n is superpolynomial in k). A trusted manager is responsible for the selection and management of all the *symmetric* user-keys (►symmetric cryptography), as well as the maintenance of user-key relation. Each party U of the group $\mathcal{G}$ is given a set of keys $X_U \subset 2^{\{0,1\}^k}$, which includes the group session key. The sets X_U are constructed in a special way by using a ►one-way function [25] $g : \{0,1\}^k \to \{0,1\}^k$ and a ►pseudo-random function [25] $f : \{0,1\}^{2k} \to \{0,1\}^k$.

Structure of the OFT tree. This is a binary tree maintained by the manager. Nodes are assigned keys as follows. First, each leaf u of the tree is assigned a random key $k_u \in_R \{0,1\}^k$. Then starting from the bottom of the tree each internal node x is assigned a key which is computed by using the rule:

$$k_x = f(g(k_{left(x)}), g(k_{right(x)})), \tag{1}$$

where $left(x)$ is the left child of x and $right(x)$ is the right child of x. The key k_{root} of the root is the group session key.

Each party U who wishes to join the group is issued with a set of keys X_U which consists of the key k_u of a leaf u and all the "blinded" node keys $k'_x = g(k_x)$ that are siblings to nodes on its path to the root. From these keys, U can compute all the node keys on its path to the root by using rule (1). An illustration is given in Fig. 2.

Removing or adding members. Whenever a member U associated with leaf u is to be removed, the key k_u of U is invalidated and the group member V associated with the sibling of u is assigned to the parent of u and given a new leaf key k_v^{new} which is selected at random from $\{0,1\}^k$ by the manager. All internal node keys along the path of this node to the root are then assigned new values using rule (1).

The new key values must be communicated securely to all the members of the group who store the corresponding old values. For this purpose the manager broadcasts the new values encrypted in such a way that only the entitled members can decrypt them. For the member V associated with the sibling v of u, the manager encrypts the new leaf key k_v^{new} with the old key of V and broadcasts $E_{k_v^{old}}[k_v^{new}]$, where E is a *symmetric encryption* function [25] (►Symmetric Cryptography). For each of the internal nodes x on the path from u to the root, only the new blinded values $(k_x^{new})' = g(k_x^{new})$ are needed (members store one leaf key and the blinded keys of the siblings of the nodes on their paths to the root). Observe that group members that possess the old blinded value $(k_x^{old})'$ know the unblinded key value k_s of the sibling s of x. The manager therefore only needs to broadcast the encryption $E_{k_s}[(k_x^{new})']$.

When a new member V joins the group, an existing leaf node u is split: the member U associated with u is now associated with $left(u)$ and the new member V is associated with $right(u)$. Members U, V are given keys $k_u^{new} = k_{left(u)}$, $k_v^{new} = k_{right(u)}$, respectively, selected at random from $\{0,1\}^k$. All the keys of the internal nodes x along the path from u to the root are then assigned new values k_x^{new} using rule (1). The new value k_u^{new} and the blinded values of the new keys $(k_x^{new})'$ are then encrypted with the appropriate keys and broadcast as in the previous case. The key k_v^{new} is given privately to V.

Security and efficiency. The security of the OFT protocol reduces to that of the one-way function g, the pseudo-random function f, and the symmetric encryption function E. One has *forward* and *backward* security (►Group Signature): members who leave the group cannot read *future* messages and parties who join the group cannot read *previous* messages.

The communication complexity of adding or removing a member is bounded by $hk + h$ bits, where h is the height of the tree and k the length of the keys. This is because up to h keys must be broadcast, and h bits are needed to describe the member that joins or leaves the

group. If the tree is balanced then the operation of adding or removing a member with the OFT protocol is scalable (with complexity $O(\log n)$).

Applications of GKA

Many computer applications involve dynamic peer groups. Examples include audio/video conferencing, multiuser computations, multicast messaging, pay-per-view, distributed interactive simulations, real-time information systems, replication services and, generally, distributed applications. To protect such applications, and in particular to preserve privacy, one has to employ secure communication channels (Secure Channel) that link all the parties of a group. For privacy, one can establish such channels by using *symmetric encryption* (►Symmetric Cryptosystem, private key cryptosystem) with key, a *group session key*.

Recommended Reading

1. Abdalla M, Bresson E, Chevassut O, Pointcheval D (2006) Password-based group key exchange in a constant number of rounds. In: PKC 2006, New York. LNCS 3958, Springer, Berlin, pp 427–442
2. Balenson DM, McGrew DA, Sherman AT (1998) Key management for large dynamic groups: one-way function trees and amortized initialization. Advanced Sec Res J – NAI Labs 1(1): 27–46
3. Bellare M, Rogaway P (1994) Entity authentication and key distribution. In: Crypto '93, Santa Barbara, CA. LNCS 773, Springer, Berlin, pp 232–249
4. Bellare M, Rogaway P (1995) Provably secure session key distribution: the three party case. In: STOC 1995, Las Vegas, NV. ACM, New York, pp 57–66
5. Bellare M, Pointcheval D, Rogaway P (2000) Authenticated key exchange secure against dictionary attacks. In: Eurocrypt 2000, Bruges, Belgium. LNCS 1807, Springer, Berlin, pp 139–155
6. Blundo C, De Santis A, Herzberg A, Kutten S, Vaccaro U, Yung M (1992) Perfectly-secure key distribution for dynamic conferences. In: Crypto '92, Santa Barbara, CA. LNCS 740, Springer, Berlin, pp 471–486
7. Boyd C, Nieto JMG (2003) Round-optimal contributory conference key agreement. In: PKC 2003, Miami, FL. LNCS 2567, Springer, Berlin, pp 161–174
8. Bresson E, Chevassut O, Pointcheval D (2001) Provably authenticated group Diffie-Hellman key exchange – the dynamic case. In: Asiacrypt 2001, Gold Coast, Australia. LNCS 2248, Springer, Berlin, pp 290–300
9. Bresson E Manulis M (2007) Malicious participants in group key exchange: key control and contributiveness in the shadow of trust. In: ATC 2007, Hong Kong, China. LNCS 4610, Springer, Berlin, pp 395–409
10. Bresson E, Manulis M (2008) Securing Group Key Exchange against strong corruptions. In: ASIACCS 2008, Tokyo. ACM, New York, pp 249–260
11. Bresson E, Chevassut O, Pointcheval D, Quisquater J-J (2001) Provably authenticated group Diffie-Hellman key exchange. In: CCS 2001, Philadelphia, PA. ACM, New York, pp 255–264
12. Bresson E, Chevassut O, Pointcheval D (2002) Group Diffie-Hellman key exchange under standard assumptions. In: Eurocrypt 2002, Amsterdam, The Netherlands. LNCS 2332, Springer, Berlin, pp 321–336
13. Bresson E, Chevassut O, Pointcheval D (2002) Group Diffie-Hellman Key Exchange secure against dictionary attacks. In: Asiacrypt 2002, Queenstown, New Zealand. LNCS 2501, Springer, Berlin, pp 497–514
14. Burmester M, Desmedt Y (2005) A secure and scalable group key exchange system. Inform Process Lett 94(3): 137–143
15. Burmester M, Desmedt Y (1995) A secure and efficient conference key distribution system. In: Eurocrypt '94, Perugia, Italy. LNCS 950, Springer, Berlin pp 275–286 (Proofs are in: Preproceedings of Eurocrypt '94: Scuola Superiore Guglielmo Reiss Romoli (SSGRR), Perugia, Italy, May 9–12, pp 279–290)
16. Burmester M, Desmedt Y (1996) Efficient and secure conference key distribution. In: Security protocols, international workshop, Cambridge, UK. LNCS 1189, Springer, Berlin, pp 119–129
17. Canetti R, Krawczyk H (2002) Universally composable notions of key exchange and secure channels. In: Eurocrypt 2002, Amsterdam, The Netherlands. http://eprint.iacr.org/2002/059
18. Desmedt Y, Pieprzyk J, Steinfeld R, Wang H (2006) A non-malleable group key exchange protocol robust against active insiders. In: ISC 2006, Samos Island, Greece. LNCS 4176, Springer, Berlin, pp 459–475
19. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Inform Theory IT–22(6):644–654
20. Ingemarsson I, Tang DT Wong CK (1982) A conference key distribution system. IEEE Trans Inform Theory 28(5): 714–720
21. Joux A (2004) A one round protocol for tripartite Diffie-Hellman. J Cryptol 17(4):263–276
22. Katz J, Sun Shin J (2005) Modeling insider attacks on group key exchange protocols. In: CCS 2005, Alexandria, VA. ACM, New York, pp 180–189
23. Katz J, Yung M (2003) Authenticated group key exchange in constant rounds. In: Crypto 2003, Santa Barbara, CA. LNCS 2729, Springer, New York, pp 110–125
24. Koyama K, Ohta K (1988) Identity-based conference key distribution systems. In: Crypto '87, Santa Barbara, CA. LNCS 293, Springer, Berlin, pp 175–185
25. Menezes A, van Oorschot P, Vanstone AA (1997) Handbook of applied cryptography. CRC Press, Boca Raton, FL
26. Mittra S (1997) Iolus: a framework for scalable secure multicasting. In: Proceedings of the ACM SIGCOMM '97, Cannes, France. ACM, New York, pp 11–19
27. Perrig A, Song D, Tygar J (2001) ELK, a new protocol for efficient large-group key distribution. In: Proceedings, IEEE Security and Privacy Symposium, Oakland, California. IEEE Computer Society Press, Washington, DC
28. Setia S, Koussih S, Jajodia S (2000) Kronos: a scalable group re-keying approach for secure multicast. In: IEEE Symposium on Security and Privacy, 2000, IEEE Computer Society Press, Washington, DC, pp 215–228
29. Sherman AT, McGrew DA (2003) Key establishment in large dynamic groups using one-way function trees. IEEE Trans Software Eng 29(5):444–448
30. Steiner M, Tsudik G, Waidner M (1996) Diffie-Hellman key distribution extended to group communication. In: CCS 1996, New Delhi, ACM, New York, pp 31–37

31. Steiner M, Tsudik G, Waidner M (1998) CLIQUES: a new approach to group key agreement. In: ICDCS '98, Amsterdam, the Netherlands
32. Steiner M, Tsudik G, Waidner M (2000) Key agreement in dynamic peer groups. IEEE Trans Parallel Distribut Syst 11(8):769–780
33. Wallner DM, Harder EJ, Agee RC (1997) Key management for multicast: issues and architectures, Internet Engineering Task Force, 1 July 1997 ftp://ftp.ietf.org/internet-drafts/draft-wallner-key-arch-01.txt.
34. Wong CK, Gouda M, Lam SS (1998) Secure group communications using key graphs. In: Proceedings of SIGCOMM 98, Vancouver, ACM, pp 68–79

Group Key Distribution

►Group Key Agreement

Group Key Exchange

►Group Key Agreement

Group Signatures

Gerrit Bleumer
Research and Development, Francotyp Group, Birkenwerder bei Berlin, Germany

Related Concepts

►Anonymity; ►Digital Signature; ►Forgery; ►Threshold Signature

Definition

Group signatures are ►digital signatures where signers can establish groups such that each member of the group can produce signatures anonymously on behalf of the group. Each group can be managed by a *trusted group authority*, which oversees joining and leaving the group and can reidentify individual signers in case of disputes according to a clearly stated policy. Obviously, different groups can choose to be managed by the same trusted group authority, or a group can choose to fully distribute the group management among its members such that every member is involved in all management transactions.

Background

The concept of group signatures and first practical constructions were introduced by Chaum and van Heijst [1]. The term "group signatures" or "group-oriented signatures" is sometimes also used for another type of signature scheme where signers also form groups such that, for example, any *t*-out-of-*n* members of a group can together produce a signature. In this case, the capability of producing signatures is granted only to large enough coalitions within a group. This was first introduced by Boyd as *multisignatures*, but according to Desmedt [2], there is growing consensus to call them *threshold signatures*.

Theory

A group signature scheme has the following operations:

1. An operation for generating pairs of a private signing key and a public verifying key for an individual
2. An operation for generating pairs of a private group key and a public group key for a trusted group authority
3. Operations for group management such as joining and revoking group members
4. An operation for signing messages
5. An operation for verifying signatures against a public group key
6. An operation to identify (de-anonymize) a group member by one of her or his signatures

In a group signature scheme, each signing member of a group has an individual signing key pair. If individuals generate their key pairs without having to agree on common domain parameters, the group signature scheme is called *separable* [3]. An individual is registered for a group by presenting a suitable ID certificate to the respective trusted group authority and submitting her or his public verifying key. The trusted group authority constructs a group key pair, which consists of a private group key and a public group key, and publishes the public group key through one or more authentic channels such as a *public key infrastructure* (PKI). A member leaves a group by revoking her or his public verifying key from the trusted group authority. It is the responsibility of the trusted group authority to keep track of who belongs to the group at any point of time. A group signature scheme is called *static* if the public group key needs to be updated if members join or leave the group, or update their individual key pairs. Constructions were proposed in [1, 4, 5]. All of them suffer from the drawback that the size of a public group key and that of the signatures are proportional to the size of a group. If the public group key remains unchanged ►when members join or leave the group or update their individual key pairs, the group signature scheme is called *dynamic* [6]. A dynamic group signature scheme features a *sequential join* or a *concurrent join* depending on whether the trusted group authority can join two or more new members one after another or in

parallel. The first construction of a dynamic group signature scheme was proposed by Camenisch and Stadler [7]. Their paper sparked more work on dynamic group signature schemes [3, 6, 8, 9]. A formel model for group signatures with concurrent join and an efficient construction was proposed by Kiayias and Yung [10].

Everyone who has access to the public group key of group G can verify every signature produced by every member of group G. When a signature is disputed, the respective verifier can request the signer's identity from the respective trusted group authority. The trusted group authority then uses the private group key in order to recover the signer's identity. The conditions whether and when a signer's identity is recovered and released are controlled by the group management policy under which the trusted group authority operates.

Robust ►key management is a particularly important issue in group signatures. For example, if the public group key changes whenever members join or leave the group or update their private signing keys, then it becomes a burden for the trusted group authority to publish the public group keys for all recent time intervals in a timely fashion. Moreover, if a private signing key of a group member is compromised, then the attacker can freely produce signatures on behalf of the group until the respective public verifying key is revoked. Therefore, all the signatures of the victimized group member must be regarded invalid if there is no way of distinguishing the signatures produced by the honest group member from those produced by the attacker. These problems are addressed by an approach called *forward security*. Forward secure group signature schemes allow individual group members to join or leave a group or update their private signing keys without affecting the respective public group key. By dividing the lifetime of all individual private signing keys into discrete time intervals, and by tying all signatures to the time interval when they are produced, group members who are revoked in time interval i have their signing capability effectively stripped away in time interval $i + 1$, while all their signatures produced in time interval i or before (and, of course, the signatures of all other group members) remain verifiable and anonymous [6]. Forward security in group signature schemes is similar to *forward security* in ►threshold signature schemes.

The characteristic security requirements of a group signature scheme are [11]:

Full Traceability: Any coalition of cheating signers including a cheating group authority, who pool their private individual keys and the private group key, cannot produce a valid signature that identifies an originating group member who has in fact not produced the signature (false claim of origin).

Full Anonymity: Any adversary without access to the trusted group authority, who chooses a message m and two group members i and j and is then provided with the two group signatures of members i and j (in random order) for message m with respect to the public group key, cannot find out with non-negligibly better chance of success than pure guessing, which signature comes from which member.

Forward Security: Signers who leave the group can no longer sign messages on behalf of the group. Other additional security requirements are proposed by Song [6].

Full-traceability implies the unforgeability requirement of any signature scheme. It also implies other security requirements that have been stated in the literature such as coalition-resistance, exculpability, and framing-resistance. Full-anonymity can be expressed equivalently in terms of unlinkability as follows [11]:

Unlinkability: Any adversary without access to the trusted group authority, who is given two messages *m1, m2* and corresponding signatures *s1, s2* of two signers where *s1* and *s2* are valid for these messages with respect to the public group key, cannot decide with non-negligible probability better than pure guessing whether the two signatures have been produced by the same group member or two different group members.

Constructions have been based on groups, in which the ►discrete logarithm problem is hard [7], and on the ►RSA signature scheme [6, 8, 9]. It is shown by Bellare et al. [11] that secure group signatures exist if trapdoor permutations exist. They suggest a static group signature construction where the length of all keys of a group is logarithmic in the number of its members. Their construction is not efficient, but serves as a proof of existence of a construction that is provably secure in the standard model (without random oracles).

Applications

Group signatures are useful, for example, to build secure auction systems [5], where all the bidders form a group and authorize their tenders by group signatures. After the winning tender has been determined, the trusted group authority can reidentify the lucky bidder, while the other bidders remain anonymous. Group signature schemes are dually related to *identity escrow schemes* [12], i.e., group-member identification schemes with revocable ►anonymity.

Group signature schemes can be equipped with additional features: Cramer et al. [13] proposed to add threshold properties into a group signature scheme such that any subset of group members can be authorized to produce signatures on behalf of the group. Lysyanskaya and Ramzan [14] proposed blind group signatures based on the dynamic group signature scheme by Camenisch and Stadler [7]. Another blind group signature scheme based on [7] was proposed by Nguyen et al. [15].

Recommended Reading

1. Chaum D, van Heijst E (1991) Group signatures. In: Davies DW (ed) Advances in cryptology –EUROCRYPT'91. Lecture notes in computer science, vol 547. Springer, Berlin, pp 257–265
2. Desmedt Y (1993) Threshold cryptography. In: Seberry J, Zheng Y (eds) Advances in cryptology – AUSCRYPT'92. Lecture notes in computer science, vol 718. Springer, Berlin, pp 3–14
3. Camenisch J, Michels M (1999) Separability and efficiency of generic group signatures. In: Wiener J (ed) Advances in cryptology – CRYPTO'99. Lecture notes in computer science, vol 1666. Springer, Berlin, pp 413–430
4. Camenisch J (1997) Efficient and generalized group signatures. In: Fumy W (ed) Advances in cryptology – EUROCRYPT'97. Lecture notes in computer science, vol 1233. Springer, Berlin, pp 465–479
5. Chen L, Pedersen TP (1995) New group signature schemes. In: De Santis A (ed) Advances in cryptology – EUROCRYPT'94. Lecture notes in computer science, vol 950. Springer, Berlin, pp 171–181
6. Song DX (2001) Practical forward secure group signature schemes. 8th ACM conference on computer and communications security (CCS-8). ACM Press, New York
7. Camenisch J, Stadler M (1997) Efficient group signature schemes for large groups. In: Kaliski BS (ed) Advances in cryptology – CRYPTO'97. Lecture notes in computer science, vol 1294. Springer, Berlin, pp 410–424
8. Ateniese G, Camenisch J, Joye M, Tsudik G (2000) A practical and provably secure coalition-resistant group signature scheme. In: Bellare M (ed) Advances in cryptology – CRYPTO 2000. Lecture notes in computer science, vol 1880. Springer, Berlin, pp 255–270
9. Camenisch J, Michels M (1998) A group signature scheme with improved efficiency. In: Ohta K, Pei D (eds) Advances in cryptography – ASIACRYPT'98. Lecture notes in computer science, vol 1514. Springer, Berlin, pp 160–174
10. Kiayias A, Yung M (2005) Group signatures with efficient concurrent join. In: Cramer R (ed) Advances in cryptology – EUROCRYPT 2005. Lecture notes in computer science, vol 3494. Springer, Berlin, pp 198–214
11. Bellare M, Micciancio D, Warinschi B (2003) Foundations of group signatures: format definitions, simplified requirements, and a construction based on general assumptions. In: Biham E (ed) Advances in cryptology – EUROCRYPT'03. Lecture notes in computer science, vol 2656. Springer, Berlin, pp 614–629
12. Kilian J, Petrank E (1998) Identity escrow. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1642. Springer, Berlin, pp 169–185
13. Cramer R, Damgård I, Schoenmakers B (1994) Proofs of partial knowledge and simplified design of witness hiding Protocols. In: Desmedt YG (ed) Advances in cryptology –CRYPTO'94. Lecture notes in computer science, vol 839. Springer, Berlin, pp 174–187, CWI Quart J, 8(2):111–128, 1995
14. Lysyanskaya A, Ramzan Z (1998) Group blind digital signatures: a scalable solution to electronic cash. In: Goldschlag DM, Stubblebine SG (eds) Financial cryptography "98". Lecture notes in computer science, vol 1465. Springer, Berlin, pp 184–197
15. Nguyen KQ, Mu Y, Varadharajan V (1999) Divertible zero-knowledge proof of polynomial relations and blind group signature. In: Pieprzyk J, Safari-Naini R, Seberry J (eds) Australasian conference on information security and privacy, ACISP'99. Lecture notes in computer science, vol 1587. Springer, Berlin, pp 117–128