S

SAFE PRIME

A *safe prime* is a prime number p of the form $p = 2q + 1$ where q is also prime. In such a case, q is called a *Sophie Germain prime*. Safe primes are used in some implementations of the Diffie–Hellman key exchange protocol, for example, to protect against certain types of attacks.

Anton Stiglic

SALT

A *salt* is a t-bit random string that may be prepended or appended to a user's password prior to application of a one-way function in order to make dictionary attacks less effective. Both the salt and the hash (or encryption) of the augmented password are stored in the password file on the system. When the user subsequently enters a password, the system looks up the salt associated with that user, augments the password with the salt, applies the one-way function to the augmented password, and compares the result with the stored value.

It is important to note that the work factor for finding a particular user's password is unchanged by salting because the salt is stored in cleartext in the password file. However, it can substantially increase the work factor for generating random passwords and comparing them with the entire password file, since each possible password could be augmented with any possible salt. The effort required to find the password associated with an entry in the password file is multiplied by the smaller of {the number of passwords, 2^t} compared with a password file containing hashes (or encryptions) of unsalted passwords.

Another benefit of salting is that two users who choose the same password will have different entries in the system password file; therefore, simply reading the file will not reveal that the passwords are the same.

Carlisle Adams

References

[1] Denning, D. (1982). *Cryptography and Data Security*. Addison-Wesley, Reading, MA.

[2] Kaufman, C., R. Perlman, and M. Speciner (1995). *Network Security: Private Communication in a Public World*. Prentice-Hall, Englewood Cliffs, NJ.

[3] Menezes, A., P. van Oorschot, and S. Vanstone (1997). *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL.

SCHNORR DIGITAL SIGNATURE SCHEME

The Schnorr signature scheme [6] is derived from Schnorr's identification protocol using the Fiat–Shamir heuristic [2]. The resulting digital signature scheme is related to the Digital Signature Standard (DSS). As in DSS, the system works in a subgroup of the group $\mathbb{Z}_p^*$ for some prime number p. The resulting signatures have the same length as DSS signatures. The signature scheme works as follows:

Key Generation. Same as in the DSS system. Given two security parameters $\tau, \lambda \in \mathbb{Z}$ $(\tau > \lambda)$ as input do the following:

1. Generate a random λ-bit prime q.
2. Generate a random τ-bit prime prime p such that q divides $p - 1$.
3. Pick an element $g \in \mathbb{Z}_p^*$ of order q.
4. Pick a random integer $\alpha \in [1, q]$ and compute $y = g^\alpha \in \mathbb{Z}_p^*$.
5. Let H be a hash function $H : \{0, 1\}^* \to \mathbb{Z}_q$.

The resulting public key is (p, q, g, y, H). The private key is (p, q, g, α, H).

Signing. To sign a message $m \in \{0, 1\}^*$ using the private key (p, q, g, α, H) do:

1. Pick a random $k \in \mathbb{Z}_p^*$.
2. Compute $r = g^k \in \mathbb{Z}_p^*$. Set $c = H(m \| r) \in \mathbb{Z}_q$ and $s = \alpha c + k \in \mathbb{Z}_q$.
3. Output the pair $(s, c) \in \mathbb{Z}_q^2$ as the signature on m.

Verifying. To verify a message/signature pair $(m,\ (s, c))$ using the public key (p, q, g, y, H) do:

1. Compute $v = g^s y^{-c} \in \mathbb{Z}p$.
2. Accept the signature if $c = H(m \| v)$. Otherwise, reject.

We first check that the verification algorithm accepts all valid message/signature pairs. For a valid message/signature pair we have

$$v = g^s y^{-c} = g^{\alpha c + k} y^{-c} = (y^c g^k) y^{-c} = g^k \in \mathbb{Z}p$$

and therefore $H(m \| v) = H(m \| g^k) = c$. It follows that a valid message/signature is always accepted.

The signature can be shown to be existentially unforgeable (see existential forgery) under a chosen message attack in the random oracle model, assuming the discrete logarithm problem in the group generated by g is intractable. This proof of security is a special case of a general result that shows how to convert a public-coin authentication protocol (a protocol in which the verifier only contributes randomness) into a secure signature scheme in the random oracle model [1, 5]. In the proof of security, the function H is assumed to be a random oracle. In practice, one derives H from some cryptographic hash function such as SHA-1.

To discuss signature length we fix concrete security parameters. At the present time the discrete-log problem in the cyclic group $\mathbb{Z}_p^*$ where p is a 1024-bit prime is considered intractable [3] except for a very well funded organization. Schnorr signatures use a subgroup of order q of $\mathbb{Z}_p^*$. When q is a 160-bit prime, the discrete log problem in this subgroup is believed to be as hard as discrete-log in all of $\mathbb{Z}_p^*$, although proving this is currently an open problem. Hence, for the present discussion we assume p is a 1024-bit prime and q is a 160-bit prime. Since a Schnorr signature contains two elements in $\mathbb{Z}_q$ we see that, with these parameters, its length is 320-bits.

Schnorr signatures are efficient and practical. The time to compute a signature is dominated by one exponentiation and this exponentiation can be done offline, i.e. before the message is given. Verifying a signature is dominated by the time to compute a multi-exponentiation of the form $g^a h^b$ for some $g, h \in \mathbb{Z}_p$ and $a, b \in \mathbb{Z}_q$. Multi-exponentiations of this type can be done at approximately the cost of a single exponentiation [4, p. 617].

Dan Boneh

References

[1] Abdalla, M., J. An, M. Bellare, and C. Namprempre (2002). "From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security." *Advances in Cryptology—EUROCRYPT 2004*, Lecture Notes in Computer Science, vol. 2332, ed. Lars Knudsen. Springer-Verlag, Berlin, 418–33.

[2] Fiat, Amos and Adi Shamir (1986). "How to prove yourself: Practical solutions to identification and signature problems." *Advances in Cryptology—CRYPTO'86*, Lecture Notes in Computer Science, vol. 263, ed. Andrew M. Odlyzko. Springer-Verlag, Berlin, 186–194.

[3] Lenstra, Arjen and Eric Verheul (2001). "Selecting cryptographic key sizes." *Journal of Cryptology*, 14 (4), 255–293.

[4] Menezes, Alfred J., Paul C. van Oorschot, and Scott A. Vanstone (1997). *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL.

[5] Ohta, Kazuo and Tatsuaki Okamoto (1998). "On concrete security treatment of signatures derived from identification." *Advances in Cryptology—CRYPTO'98*, Lecture Notes in Computer Science, vol. 1462, ed. H. Krawczyk. Springer-Verlag, Berlin, 354–369.

[6] Schnorr, C. (1991). "Efficient signature generation by smart cards." *Journal of Cryptology*, 4 (3), 161–174.

SCHNORR INDENTIFICATION SCHEME

In its simplest form, an identification protocol involves the presentation or submission of some information (a "secret value") from a claimant to a verifier (see Identification). Challenge-response identification is an extension in which the information submitted by the claimant is the function of both a secret value known to the claimant (sometimes called a "prover"), and a challenge value received from the verifier (or "challenger").

Such a challenge-response protocol proceeds as follows. A verifier V generates and sends a challenge value c to the claimant C. Using his/her secret value s and appropriate function $f()$, C computes the response value $v = f(c, s)$, and returns v to V. V verifies the response value v, and if successful, the claim is accepted. Choices for the challenge value c, and additionally options for the function $f()$ and secret s are discussed below.

Challenge-response identification is an improvement over simpler identification because it offers protection against replay attacks. This is achieved by using a challenge value that is time-varying. Referring to the above protocol, there are three general types of challenge values that might be used. The property of each is that the challenge value is not repeatedly sent to multiple claimants. Such a value is sometimes referred to as a *nonce*, since it is a value that is "*n*ot used more than *once*." The challenge value could be a randomly generated value (see Random bit generation), in which case V would send a random value c to C. Alternatively, the challenge value might be a sequence number, in which case the verifier V would maintain a sequence value corresponding to each challenger. At each challenge, the stored sequence number would be increased by (at least) one before sending to the claimant. Finally, the challenge value might be a function of the current time. In this case, a challenge value need not be sent

from V to C, but could be sent by C, along with the computed verifier. As long as the time chosen was within an accepted threshold, V would accept.

There are three general classes of functions and secret values that might be used as part of a challenge-response protocol. The first is symmetric-key based in which the claimant C and verifier V *a priori* share a *secret* key K. The function $f()$ is a symmetric encryption function (see Symmetric Cryptosystem), a hash function, or a *Message Authentication Code* (MAC algorithms). Both Kerberos (see Kerberos authentication protocol) and the Needham–Schroeder protocol are examples of symmetric-key based challenge-response identification. In addition, the protocols of ISO/IEC 9798-2 perform identification using symmetric key techniques.

Alternatively, a public key based solution may be used. In this case, the claimant C has the private key in a public key cryptosystem (see Public Key Cryptography). The verifier V possesses a public key that allows validation of the public key corresponding to C's private key. In general, C uses public key techniques (generally based on number-theoretic security problems) to produce a value v, using knowledge of his/her private key. For example, V might encrypt a challenge value and send the encrypted text. C would decrypt the encrypted text and return the value (i.e., the recovered plaintext) to V (note that in this case it would only be secure to use a random challenge, and not a sequence number or time-based value). Alternatively, V might send a challenge value to C and ask C to digitally sign and return the challenge (see Digital Signature Schemes). The Schnorr identification protocol is another example of public key based challenge-response identification.

Finally, a zero-knowledge protocol can be used. In this case, the challenger demonstrates knowledge of his/her secret value without revealing any information (in an information theoretic sense—see "information theoretic security" in glossary) about this value. Such protocols typically require a number of "rounds" (each with its own challenge value) to be executed before a claimant may be successfully verified.

Mike Just

References

[1] Menezes, A., P. van Oorschot, and S. Vanstone (1997). *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL.

[2] Stinson, D.R. (1995). *Cryptography: Theory and Practice*. CRC Press, Boca Raton, FL.

SEAL

SEAL stands for Software-optimized Encryption ALgorithm. It is a binary additive stream cipher (see the entry concerning synchronous stream ciphers). It has been proposed in 1993, and several versions have been published: SEAL 1.0, SEAL 2.0 and SEAL 3.0 [3, 4]. Some attacks have been published that show how SEAL 1.0, SEAL 2.0 [2] and later SEAL 3.0 [1] can be distinguished from a true random function. But there is no really practical attack for the moment.

SEAL has been designed to be really efficient in its software implementation, mainly for 32-bit processors. It is a length-increasing pseudorandom function that maps a 32-bit sequence number n to an L-bit keystream, under control of a 160-bit secret key. a more precise description can be found in the original papers.

Caroline Fontaine

References

[1] Fluhrer, S.R. (2001). "Cryptanalysis of the SEAL 3.0 pseudorandom function family." *FSE'01*, Lecture Notes in Computer Science, vol. 2355, ed. M. Matsui. Springer-Verlag, Berlin, 135-ff.

[2] Handschuh, H. and H. Gilbert (1997). "χ^2-cryptanalysis of the SEAL encryption algorithm." *FSE'97*, Lecture Notes in Computer Science, vol. 1687, eds. O. Nierstrasz and M. Lemoine. Springer-Verlag, Berlin, 1–12.

[3] Rogaway, P. and D. Coppersmith (1994). "A software-optimized encryption algorithm." *FSE'94*, Lecture Notes in Computer Science, vol. 809, ed. R.J. Anderson. Springer-Verlag, Berlin, 56–63.

[4] Rogaway, P. and D. Coppersmith (1998). "A software-optimized encryption algorithm." *Journal of Cryptology*, 11 (4), 273–287.

SECOND PREIMAGE RESISTANCE

Second preimage resistance is the property of a hash function that it is computationally infeasible to find any second input that has the same output as a given input. This property is related to preimage resistance and one-wayness; however, the later concept is typically used for functions with input and output domain of similar size (see one-way function). Second preimage resistance is also known as weak collision resistance. A minimal requirement for a hash function to be second preimage resistant is that the length of its result should be at least 80 bits (in 2004). A hash

function is said to be a one-way hash function (OWHF) if it is both preimage resistant and second preimage resistant. The relation between collision resistance, second preimage resistance and preimage resistance is rather subtle, and it depends on the formalization of the definition: it is shown in [6] that under certain conditions, collision resistance implies second preimage resistance and second preimage resistance implies preimage resistance.

In order to formalize the definition, one needs to specify according to which distribution the first element in the domain is selected and one needs to express the probability of finding a second preimage for this element. Moreover, one often introduces a class of functions indexed by a public parameter, which is called a key. One could then distinguish between three cases: the probability can be taken over the random choice of elements in the range, over the random choice of the parameter, or over both simultaneously. As most practical hash functions have a fixed specification, the first approach is more relevant to applications. The second case is known as a Universal One-Way Hash Function or UOWHF.

The definition of a one-way function was given in 1976 by Diffie and Hellman [1]. Second preimage resistance of hash functions has been introduced by Rabin in [5]; further work on this topic can be found in [2–4, 7, 8]. For a complete formalization and a discussion of the relation between the variants and between hash functions properties, the reader is referred to Rogaway and Shrimpton [6].

B. Preneel

References

[1] Diffie, W. and M.E. Hellman (1976). "New directions in cryptography." *IEEE Transactions on Information Theory*, IT-22 (6), 644–654.

[2] Merkle, R. (1979). *Secrecy, Authentication, and Public Key Systems*. UMI Research Press.

[3] Preneel, B. (1993). "Analysis and Design of Cryptographic Hash Functions." *Doctoral Dissertation*, Katholieke Universiteit Leuven.

[4] Preneel, B. (1999). "The state of cryptographic hash functions." *Lectures on Data Security*, Lecture Notes in Computer Science, vol. 1561, ed. I. Damgård. Springer-Verlag, Berlin, 158–182.

[5] Rabin, M.O. (1978). "Digitalized signatures." *Foundations of Secure Computation*, eds. R. Lipton and R. DeMillo. Academic Press, New York, 155–166.

[6] Rogaway, P. and T. Shrimpton (2004). "Cryptographic hash function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance." *Fast Software Encryption*, Lecture Notes in Computer Science, vol. 3017, eds. B.K. Roy and W. Meier. Springer-Verlag, Berlin, 371–388.

[7] Stinson, D. (2001). "Some observations on the theory of cryptographic hash functions." Technical Report 2001/020, University of Waterloo.

[8] Zheng, Y., T. Matsumoto, and H. Imai (1990). "Connections between several versions of one-way hash functions." *Proceedings SCIS90, The 1990 Symposium on Cryptography and Information Security, Nihondaira, Japan, January 31–February 2*.

SECRET SHARING SCHEMES

Informally speaking, a secret sharing scheme (SSS, for short) allows one to share a secret among n participants in a such a way that some sets of participants called *allowed* coalitions can recover the secret exactly, while any other sets of participants (*non-allowed* coalitions) cannot get any additional (i.e., a posteriori) information about the possible value of the secret. the SSS with the last property is called *perfect*. The set Γ of all allowed coalitions is called an access structure.

The history of SSS began in 1979 when this problem was introduced and partially solved by Blakley [1] and Shamir [2] for the case of (n, k)-threshold schemes where the access structure consists of all sets of k or more participants. Consider the simplest example of (n, n)-threshold scheme. There is a dealer who wants to distribute a secret s_0 among n participants. Let s_0 be an element of some finite additive group G. For instance, G is the group of binary strings of length m with addition by modulo 2, i.e., $G = GF(2)^m$ (see finite field). The dealer generates a random sequence $s_1, \ldots, s_n$ such that $\sum_{i=1}^{n} s_i = s_0$ (for instance, by generating independently elements $s_1, \ldots, s_{n-1} \in G$ and then putting $s_n := s_0 - \sum_{i=1}^{n-1} s_i$). Then the dealer sends *privately* to each ith participant the elements s_i called share, i.e., other participants have no information about the value of s_i. It is easy to see that any coalition of less then n participants has no information except of a priori information about s_0 and all participants together recover the value of the secret as $\sum_{i=1}^{n} s_i$. These simple schemes appear to be enough for the realization of arbitrary monotone (i.e., if $A \in \Gamma$ and $A \subset B$ then $B \in \Gamma$) access structure Γ. Namely, for any allowed coalition $A \in \Gamma$ let the above realize (independently) an $(|A|, |A|)$-threshold scheme, i.e., send to the ith, participant as many shares s_i^A) as the number of allowed coalitions to which this participant

belongs it is enough to consider only maximum allowed coalitions).

The probabilistic model of an SSS for the general case is the following (see [4, 5]). There are $n+1$ sets $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_n$ and the probability distribution P on their Cartesian product $\mathcal{S} = \mathcal{S}_0 \times \cdots \times \mathcal{S}_n$. A pair $(P, \mathcal{S})$ is called a perfect probabilistic SSS realizing the access structure Γ if the following two properties hold:

- participants of an allowed set A (i.e., $A \in \Gamma$) together can recover the secret exactly (formally, $P(S_0 = c_0 \mid S_i = c_i, i \in A) \in \{0, 1\}$ if $A \in \Gamma$);
- participants forming a non-allowed set A ($A \notin \Gamma$) cannot get additional information beyond their *a priori* information about s_0, i.e., $P(S_0 = c_0 \mid S_i = c_i, i \in A) = P(S_0 = c_o)$ if $A \notin \Gamma$. These conditions can be reformulated in the language of entropy (see information theory) as $H(S_i, i \in A \cup 0) = H(S_i, i \in A) + \delta_\Gamma(A) H(S_0)$, where $\delta_\Gamma(A) = 0$ if $A \in \Gamma$, and $\delta_\Gamma A) = 1$, otherwise.

There are also combinatorial models of SSSs. An arbitrary set $V \subset \mathcal{S}$ is called the "code" of the combinatorial SSS, and its codewords called "sharing rules". The simplest combinatorial model demands that at first, for every set $A \in \Gamma$ the 0th coordinate of any codeword from V is uniquely determined by the values of the coordinates from the set A and, secondly, for every set $A \notin \Gamma$ and for any given selection of values of the coordinates from the set A the number of codewords with given value of 0th coordinate does not depend on this value. This model is a particular case of the probabilistic model, namely, when all nonzero values of P are equal. The most general definition of combinatorial models, which contain probabilistic ones as a particular case, was given in [6, 7].

For both types of models the "size" of share, provided to any participant, cannot be smaller than the "size" of the secret, where "size" is defined as $log|S_i|$ or $H(S_i)$ respectively for combinatorial and probabilistic statements of the problem. Special attention has been paid to so-called *ideal* SSSs, where the size of any share coincides with the size of the secret. For example, any (n, k)-threshold scheme can be realized as an ideal perfect SSS (see threshold cryptography). It was shown in a chain of papers [6–9] that the access structures of ideal perfect SSS correspond to a special class of matroids [10]. On the other hand, any access structure can be realized as a perfect SSS but probably not very efficient (ideal). At least the above given realization demands for some access structures to distribute shares which size is exponentially (in n) larger than the secret's size. An infinite family of access structures was constructed such that for any perfect realization the size of the shares is at least $n/\ln n$ times larger than the size of the secret [11].

To generate shares the dealer of an SSS has to use some source of randomness, say $r \in X$, where X is in some probabilistic space, and any share s_i is a function of s_0 and r, i.e., $s_i = f_i(s_0, r)$. A linear realization of SSS (or, *linear* SSS) means that all functions $f_i(\cdot)$ are linear. To make it formal: let $s_0, \ldots, s_n$ be elements in m_i-dimensional vector spaces ($i = 0, 1, \ldots, n$) over some finite field $GF(q)$ of q elements, let r be an element of the l-dimensional vector space over the same field. Then a linear SSS is generated by some $(m_0 + l) \times m$ matrix G according to the formula $s = (s_0, \ldots, s_n) = xG$, where $m = \sum_{i=0}^{n} m_i$ and vector x is the concatenation of vectors s_0 and r. Consider vector spaces $V_0, \ldots, V_n$, where V_i is the linear subspace generated by the columns of G that correspond to s_i, i.e., by columns g_j, where $j = m_0 + \cdots + m_{i-1} + 1, \ldots, m_0 + \cdots + m_i$. Then matrix G realizes the access structure Γ perfectly if and only if [12, 13]:

- for any set $A \in \Gamma$ the linear span of subspaces $\{V_a : a \in A\}$ (i.e., the minimal vector subspace containg all these subspaces V_a) contains the subspace V_0;
- for any set $A \notin \Gamma$ the linear span of subspaces $\{V_a : a \in A\}$ intersects with the linear subspace V_0 only by the vector $\mathbf{0}$.

All aforementioned examples of SSS are linear. Note that if all dimensions m_i are equal to 1 then the matrix G can be considered as a generator matrix of a linear error-correcting code (see cyclic codes). In particular, it gives another description of Shamir's threshold schemes via Reed–Solomon codes [14]. This "coding theory approach" was further developed and generalized to the case of arbitrary linear codes and their minimal words as only possible access structures [16]. Surely, a linear SSS with all dimensions $m_i = 1$ is ideal, but multidimensional linear SSSs with all $m_i = m_0$ give a larger class of ideal SSS [15]. It is an open question if any ideal SSS can be realized as a (multidimensional) linear ideal SSS?

Modifications of assumptions of the secret sharing problem's statement such as perfectness of the scheme, honesty of the dealer and participants, sending shares via secure, private channels and so on lead to many variations of secret sharing problem. Among them: *ramp schemes*, publicly verifiable secret schemes, SSS with cheaters, SSS with public reconstruction [17], and visual secret sharing schemes.

Robert Blakley
Gregory Kabatiansky

References

[1] Blakley, R. (1979). "Safeguarding cryptographic keys." *Proceedings of AFIPS 1979 National Computer Conference*, vol. 48, Va. Arlington, *New York*, 313–317.

[2] Shamir, A. (1979). "How to share a secret." *Communications of the ACM*, 22 (1), 612–613.

[3] Ito, M., A. Saito, and T. Nishizeki (1993). "Multiple assignment scheme for sharing secret." *Journal of Cryptology*, 6, 15–20.

[4] Karnin, E.D., J.W. Greene, and M.E. Hellman (1983). "On secret sharing systems." *IEEE Transactions on Informatiom Theory*, 29 (1), 231–241.

[5] Capocelli, R.M., A. De Santis, L. Gargano, and U. Vaccaro (1993). "On the size of shares for secret sharing schemes." *Journal of Cryptology*, 6, 157–167.

[6] Brickell, E.F. and D.M. Davenport (1991). "On the classification of ideal secret sharing schemes." *Journal of Cryptology*, 4, 123–134.

[7] Blakley, G.R. and G.A. Kabatianski (1997). "Generalized ideal secret sharing schemes and matroids." *Problems of Information Transmission*, 33 (3), 102–110.

[8] Kurosawa, K., K. Okada, K. Sakano, W. Ogata, and S. Tsujii (1993). "Nonperfect secret sharing schemes and matroids." *Advances in Cryptology—EUROCRYPT'93*, Lecture Notes in Computer Science, vol. 765, ed. T. Helleseth. Springer-Verlag, Berlin, 126–141.

[9] Jackson, W.-A. and K.M. Martin (1998). "Combinatorial models for perfect secret sharing schemes." *Journal of Combinatorial Mathematics and Combinatorial Computing*, 28, 249–265.

[10] Welsh, D.J.A. (1976). *Matroid Theory*. Academic Press, New York.

[11] Csirmaz, L. (1995). "The size of a share must be large." *Advances in Cryptology—EUROCRYPT'94*, Lecture Notes in Computer Science, vol. 950, ed. A. De Santis. Springer-Verlag, Berlin, 13–22.

[12] Blakley, G.R. and G.A. Kabatianski (1994). "Linear algebra approach to secret sharing schemes." *Error Control, Cryptology and Speech Compression*, Lecture Notes in Computer Science, vol. 829, eds. A. Chmora, and S.B. Wicker. Springer, Berlin, 33–40.

[13] van Dijk, M. (1995). "A linear construction of perfect secret sharing schemes." *Advances in Cryptology—EUROCRYPT'94*, Lecture Notes in Computer Science, vol. 950, ed. A. De Santis. Springer-Verlag, Berlin, 23–34.

[14] McEliece, R.J. and D.V. Sarwate (1981). "On secret sharing and Reed–Solomon codes." *Communications of the ACM*, 24, 583–584.

[15] Ashihmin, A. and J. Simonis (1998). "Almost affine codes." *Designs, Codes and Cryptography*, 14 (2), 179–197.

[16] Massey, J. (1993). "Minilal codewords and secret sharing." *Proceedings of Sixth Joint Swedish–Russian Workshop on Information Theory, Molle, Szweden*, 246–249.

[17] Beimel, A. and B. Chor (1998). "Secret sharing with public reconstruction." *IEEE Transactions on Informatiom Theory*, 44 (5), 1887–1896.

SECURE SIGNATURES FROM THE "STRONG RSA" ASSUMPTION

In the late 1990's it was realized that by making a somewhat stronger intractability assumption than RSA (see RSA problem), it is possible to devise digital signature schemes that are fairly efficient, and at the same time have a rigorous proof of security (without resorting to the random-oracle heuristic). The intractability assumption states that given a modulus n (see modular arithmetic) of unknown factorization and an element x in the ring Z_n^*, it is hard to come up with an exponent $e \geq 2$ and an element y in Z_n^*, such that $y^e = x \pmod{n}$. This assumption, first used by Barić and Pfitzmann in the context of fail-stop signatures [1], is called the *strong RSA assumption* (or the *flexible RSA assumption*).

A simple way of using this assumption for signatures was described by Gennaro et al. [7]. In their construction, the public key (see public key cryptography) is a triple (n, x, H), where n is product of two "quasi-safe primes", x is a random element in Z_n^*, and H is a hash function, mapping strings to odd integers. The secret key consists of the factorization of n. To sign a message m, the signer picks a random string r, computes $e \leftarrow H(m, r)$, and then, using the factorization of n, finds an element $y \in Z_n^*$ such that $y^e = x \pmod{n}$. To verify a signature (r, y) on message m, the verifier checks that $y^{H(m,r)} = x \pmod{n}$.

Gennaro et al. proved that this scheme is secur—in the sense of existential unforgeability (see also existential forgery) under an adaptive chosen message attack (EU-CMA)—under some conditions on the function H. The first condition is that H is *division intractable*. This means that it is hard to come up with a list of pairs, $(m_i, r_i), i = 1, \ldots, t$, where the integer $H(m_t, r_t)$ divides the product $\prod_{i=1}^{t=1} H(m_i, r_i)$. The other condition on H means, informally, that one cannot reduce breaking the strong RSA assumption to "breaking the hash function H". It is shown in [7] that hash functions satisfying these conditions exist if the strong RSA assumption holds. (It is also shown in [7] that if H is modeled as a random oracle, then it satisfies these conditions, provided

that its output is sufficiently long. However, Coron and Naccache showed in [2] that the output of H in this case should be at least as long as the modulus n.)

Cramer and Shoup [4], followed by Fischlin [6], proposed more efficient signatures based on the strong RSA assumption. In the Cramer–Shoup public key scheme, the public key is a 5-tuple (n, h, x, e', H), where n is a product of two "safe primes", h, x are random quadratic residues in Z_n^*, e' is an odd prime, and H is a hash function, mapping strings to integers. If ℓ denotes the output length of H, then the prime e' must be at least $(\ell+1)$-bit long. The secret key consists of the factorization of n. To sign a message m, the signer picks a new $(\ell+1)$-bit prime $e \neq e'$ and a random quadratic residue $y \in Z_n^*$, sets $x' \leftarrow (y')^{e'} h^{-H(m)} \pmod{n}$, and then, using the factorization of n, finds an element $y \in Z_n^*$ such that $y^e = x h^{H(x')} \pmod{n}$. To verify a signature (e, y, y') on message m, the verifier checks that e is an odd $(\ell+1)$-bit number, $e \neq e'$, sets $x' \leftarrow (y')^{e'} h^{-H(m)} \pmod{n}$, and checks that $y^e = x h^{H(x')} \pmod{n}$.

In the scheme of Fischlin, the public key is a 5-tuple (n, g, h, x, H), where n, h, x, H are as in the Cramer–Shoup scheme, and g is yet another random quadratic residue in Z_n^*. Again, the secret key is the factorization of n, and we use ℓ to denote the output length of H. To sign a message m, the signer picks a new $(\ell+1)$-bit prime e and a random ℓ-bit string α, and then, using the factorization of n, finds an element $y \in Z_n^*$ such that $y^e = x g^{\alpha} h^{\alpha \oplus H(m)} \pmod{n}$. To verify a signature (e, α, y) on message m, the verifier checks that e is an odd $(\ell+1)$-bit number, that α is an ℓ-bit string, and that $y^e = x g^{\alpha} h^{\alpha \oplus H(m)} \pmod{n}$. Fischlin also proposed other variations of this scheme, where the prime e can be chosen even shorter than $\ell+1$ bits (and the computation made slightly more efficient), at the price of a longer public key.

For all of these schemes, it is proved that they are secure (in the sense of EU-CMA), assuming the strong RSA assumption and the collision-intractability of the hash function H, and as long as the signer never uses the same prime e for two different signatures. The Cramer-Shoup signature scheme was generalized by Damgård and Koprowski to any group where extracting roots is hard [5]. The same generalization can be applied to the schemes described by Fischlin.

It is interesting to note that the Cramer-Shoup signature scheme can be viewed as a simple variation on an earlier scheme by Cramer and Damgård [3]. The Cramer-Damgård scheme is based on a tree construction, and one of its parameters is the depth of that tree. For a tree of depth one, the scheme maintains in the public key a list of t odd primes $e_1, \ldots, e_t$, and can be used to generate upto t signatures, using a different prime each time. The Cramer-Shoup scheme is obtained essentially by letting the signer choose the odd primes "on the fly" instead of committing to them ahead of time in the public key. One pays for this flexibility by having to make a stronger intractability assumption. Since the attacker can now choose the exponent e relative to which it issues the forgery, one has to assume the strong RSA assumption (whereas the standard RSA assumption suffices for the Cramer–Damgård scheme.)

A curious feature of the Cramer-Shoup and Fischlin schemes (as well as some instances of the Gennaro–Halevi–Rabin scheme) is that when there is an a-priori polynomial bound on the total number of signatures to be made, the signer can generate its public/private key pair even without knowing the factorization of the modulus n. (This is done using the same strategy as the simulator in the security proof of these schemes.) That makes it possible in some cases to have many users in a system, all sharing the same modulus n.

Dan Boneh

References

[1] Barić, N. and B. Pfitzmann (1997). "Collision-free accumulators and fail-stop signature schemes without trees." *Advances in Cryptology—EUROCRYPT'97*, Lecture Notes in Computer Science, vol. 1233, ed. W. Fumy. Springer-Verlag, Berlin, 480–494.

[2] Coron, J.S. and D. Naccache (2000). "Security analysis of the Gennaro–Halevi–Rabin signature scheme." *Advances in Cryptology—EUROCRYPT 2000*, Lecture Notes in Computer Science, vol. 1807, ed. B. Preneel. Springer-Verlag, Berlin, 91–101.

[3] Cramer, R. and I.B. Damgård (1996). "New generations of secure and practical RSA-based signatures." *Advances in Cryptology—CRYPTO'96*, Lecture Notes in Computer Science, vol. 1109, ed. N. Koblitz. Springer-Verlag, Berlin, 173–186.

[4] Cramer, R. and V. Shoup (2000). "Signature schemes based on the strong RSA assumption." *ACM Transactions on Information System Security (ACM-TISSEC)*, 3 (3), 161–185.

[5] Damgård, I.B. and M. Koprowski (2002). "Generic lower bounds for root extraction and signature schemes in general groups." *Advances in Cryptology—EUROCRYPT 2002*, Lecture Notes in Computer Science, vol. 2332, ed. L. Knudsen. Springer-Verlag, Berlin, 256–271.

[6] Fischlin, M. (2003). "The Cramer–Shoup strong-RSA signature scheme revisited." *Public Key Cryptography—PKC 2003*, Lecture Notes in Computer Science, vol. 2567, ed. Y.G. Desmedt. Springer-Verlag, Berlin, 116–129.

[7] Gennaro, R., S. Halevi, and T. Rabin (1999). "Secure hash-and-sign signatures without the random oracle." *Advances in Cryptology—EUROCRYPT'99*, Lecture Notes in Computer Science, vol. 1592, ed. J. Stem. Springer-Verlag, Berlin, 123–139.

SECURE SOCKET LAYER (SSL)

GENERAL: *Secure Socket Layer* (SSL) denotes the predominant security protocol of the Internet for World Wide Web (WWW) services relating to electronic commerce or home banking. The majority of web servers and browsers support SSL as the defacto standard for secure client-server communication. The Secure Socket Layer protocol builds up point-to-point connections that allow private and unimpaired message exchange between strongly authenticated parties.

CLASSIFICATION: In the ISO/OSI reference model [8], SSL resides in the session layer between the transport layer (4) and the application layer (7); with respect to the Internet family of protocols this corresponds to the range between TCP/IP and application protocols such as HTTP, FTP, Telnet, etc. SSL provides no intrinsic synchronization mechanism; it relies on the data link layer below.

Netscape developed the first specification of SSL in 1994, but only publicly released and deployed the next version, SSLv2, in the same year [6]. With respect to public key cryptography, it relies mainly on RSA encryption (RSA public key cryptosystem) and X.509-compliant certificates. Block ciphers, such as DES (see Data Encryption Standard), Triple DES (3DES), and RC4, along with hash functions like MD5 and SHA, complement the suite of algorithms. SSLv3 followed in 1995, adding cryptographic methods such as Diffie–Hellman key agreement (DH), support for the FORTEZZA key token, and the Digital Signature Standard (DSS) scheme [5].

This article focuses on SSL version 3.0 and its designated successor protocol Transport Layer Security (TLS) 1.0, which the *Internet Engineering Task Force* (IETF) published for the first time in 1999 [3]. The IETF published the most recent Internet-Draft for TLS 1.1 in October 2002 [4].

LAYER STRUCTURE: SSL splits into distinct layers and message types (see Figure 1). The *handshake* message sequence initiates the communication, establishes a set of common parameters like the protocol version, applicable cryptographic algorithms (*cipher suites*), and assures the validity of the message sequence. During the handshake, the participants accomplish the negotiated authentication and derive the session key material.

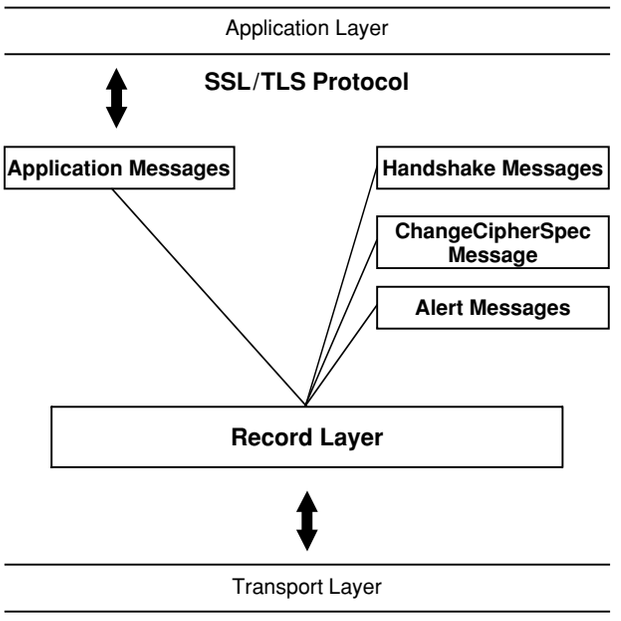


Fig. 1. SSL layer and message structure

The *record layer* fragments the full data stream into records with a maximum size of 2^{14} bytes and envelopes them cryptographically under the current session keys. Records contain a keyed message authentication code (HMAC). The initial handshake presupposes a NULL cipher suite applying no encryption and no HMAC. The record layer fully provides the use of compression. However, for patent reasons the core specifications name no method explicitly, except for the mandatory NULL algorithm, which practically makes compression an incompatible, implementation-dependent feature.

Additional *alert* messages inform on exceptional protocol conditions or one participant's demand to end the communication (*closure* alert).

BASIC PROTOCOL SEQUENCE: The SSL handshake accomplishes three goals. First, both parties agree on a *cipher suite*, i.e. the set of cryptographic algorithms that they intend to use for application data protection. Second, they establish a common *master_secret* in order to derive their session key material. Third, the participant's identities are authenticated. Although the SSL specification permits anonymous, server-only and mutual authentication, it is customary to only assert the server's identity.

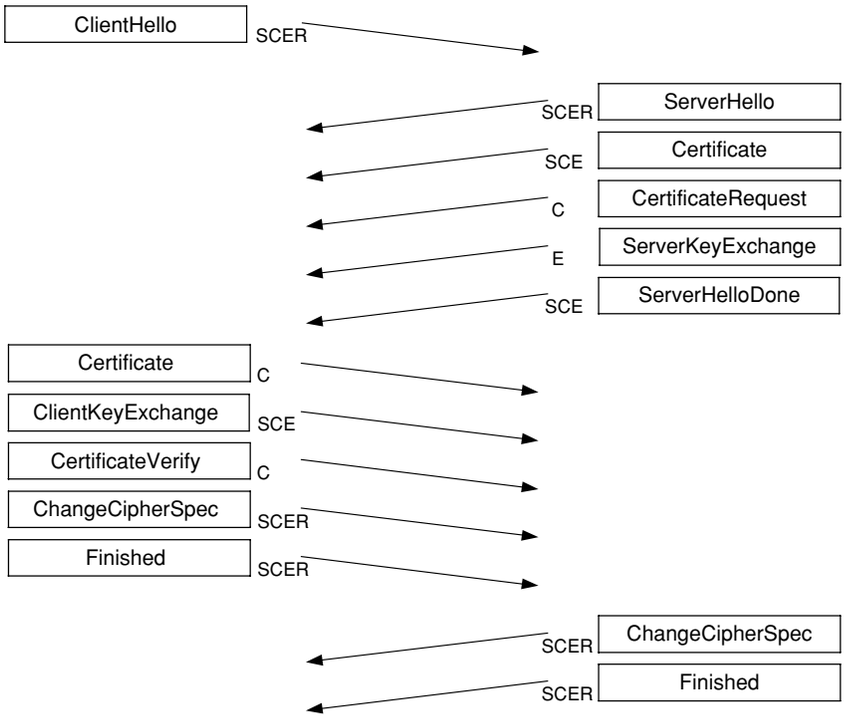


Fig. 2. SSL protocol sequence

Figure 2 gives an overview of the SSL protocol variants. It comprises four different handshake sequences, each identified by a capital letter:

- ***S*** denotes the server-authenticated message flow.
- ***C*** marks the sequence with additional client authentication.
- ***E*** shows the handshake variant with ephemeral Diffie–Hellman key agreement.
- ***R*** stands for the handshake of resumed sessions. Note, that the message pairs *ChangeCipherSpec*/*Finished* of client and server are drawn in reverse order; the server message pair follows *ServerHello* immediately.

The client opens the connection with a *ClientHello* message, which contains its selection of acceptable cipher suites and a random number.

The server chooses a supported cipher suite and adds another random number, which together builds the *ServerHello* response. Later, these two random numbers become part of the session's *master_secret*.

SERVER AUTHENTICATION: The server appends a *Certificate* message, which holds a X.509 certificate bearing its identity and public key. Most often RSA keys (see RSA digital signature scheme) are used. DSS signed certificates usually carry a long term DH public key. If multiple levels of the public key hierarchy separate the server certificate from the root authority certificate present in the client browser, then the server is required to deliver an ordered list of all dependent certificates. The empty *ServerHelloDone* message finishes this sequence.

The client confirms the validity of the certificate chain up to one of its built-in root certificates. It generates another random number and encrypts it under the server's RSA public key. This encrypted *pre_master_secret* forms the *ClientKeyExchange* message.

DH/DSS cipher suites might demand the client to create (ephemeral) DH keys matching the server's domain parameters for the *ClientKeyExchange* message. Note, that if both parties own certificates with group-compatible, fixed DH public keys, every connection generates the identical *pre_master_secret*.

Both sides derive the shared session key material independently in two steps. First, the *key derivation function* (KDF) transforms the client's *pre_master_secret* and both exchanged random numbers into the *master_secret*. Afterwards, the KDF is re-applied to the *master_secret* and both random values to compute the final key block. With respect to the chosen cipher suite, the key block is broken up into a maximum of six segments used as directional encryption keys (with initialization vectors) and directional HMAC keys.

CLIENT AUTHENTICATION: SSL servers can demand client authentication through a *CertificateRequest* message. It contains the permitted certificate types, i.e. signature algorithms, and a list of trusted certification authorities identified by their respective distinguished name.

The client answers with a *Certificate* message requiring either a single certificate or the full certification chain. In addition, it creates a *CertificateVerify* message that contains a digest of the previous handshake messages, signed by the private key corresponding to its certificate.

EPHEMERAL DIFFIE–HELLMAN: The ephemeral Diffie–Hellman key agreement (DH) embeds into the *ServerKeyExchange* and the *ClientKeyExchange* messages. Both sides send their DH public keys and, together with their own DH private keys, calculate a shared *pre_master_secret*. Note, that anonymous DH cipher suites are susceptible to man-in-the-middle attacks and protect only against passive eavesdropping (eavesdropper).

Both sides end the handshake sequence with two further messages: *ChangeCipherSpec* indicates the shift to the newly negotiated cipher parameters. *Finished* is the first message encrypted under the new keys and declares the handshake sequence complete. It holds an HMAC digest over the whole handshake sequence and the negotiated *master_secret* in order to ensure that no message tampering remains undetected.

The cryptographic state is established and confirmed on both sides and the record layer now encrypts and authenticates application data under its new session keys.

The alert message *CloseAlert* indicates the protocol end, followed by the TCP layer's closing FIN packet.

PROTOCOL RESUMPTION: SSL permits the resumption of formerly established sessions, in order to shortcut the handshake, and preserve CPU cycles by avoiding, for instance, the computationally expensive *pre_master_secret* decryption.

Depending on whether the server supports this feature, it sends a session identification string enclosed in the *ServerHello* message. After establishing a valid cryptographic state, this ID refers to the stored *master_secret*. Subsequent client connections indicate their intention to resume a session by specifying the ID in the *ClientHello* message.

Resumed sessions possess unique key blocks, because the key generation process recombines the stored *master_secret* with fresh random nonce out of both *Hello* messages.

ADDITIONAL INFORMATION: SSL permits the re-negotiation of its cipher suites during the course of the application protocol through a simple repetition of the handshake sequence.

The Internet Assigned Numbers Authority (IANA) assigns unique TCP port numbers to SSL/TLS-enabled protocols, which are marked with the appended letter *s*; for example port 443 for HTTPS or 989/990 for FTPS [7].

SECURITY ANALYSIS, BUGS: Several authors have analysed the SSL protocol suite, stating in consensus that, beginning with v3.0, it is mature and without major design flaws [11, 12, 14].

Wagner and Schneier conclude in their analysis that "*In general SSL 3.0 provides excellent security against eavesdropping and other passive attacks. Although export-weakened modes offer only minimal confidentiality protection, there is nothing SSL can do to improve that fact.*" [14]

Some minor attacks are known, however, cautious implementation seems to prevent their applicability [13].

The man-in-the-middle *version rollback* attack tries to establish a SSL v2.0 handshake protocol between SSLv3/TLS-capable parties in compatibility mode. Due to a serious flaw in SSLv2, an active adversary is capable to enforce an export weakened cipher suite, and *brute-force attack* (see cryptanalysis) to session keys directly. The SSLv2 attack is called *cipher suite rollback*. Reference [3] gives recommendations on how to detect downgrade attempts by embedding a short, well defined pattern into the PKCS#1 padding data (PKCS) of RSA encryptions, instead of using of purely random bytes. If an SSLv3/TLS-capable server finds the pattern, it will recognize that the other party operates in backwards compatibility mode although a later protocol version is supported.

Bleichenbacher published an attack against PKCS#1 (version 1) formatted RSA encryptions known as the *million message attack* [1]. Probing an SSL/TLS server with chosen *ClientKeyExchange* messages might reveal the encrypted *pre_master_secret* after about 2^{20} attempts, if the server issues error alerts that allow to distinguish between correctly and incorrectly formatted messages (chosen ciphertext attack). The TLS specification recommends as a countermeasure to treat PKCS#1 formatting errors by simply continuing the handshake protocol with a randomly generated *pre_master_secret*. In this case, the server

behaves in the same way, whether or not the formatting is correct [4].

The general method of timing cryptanalysis [10] is applicable against SSL/TLS servers, if a large number of unbiased measurements of private key operations is available. Practical attacks were shown for example by Brumley and Boneh and Klime, et al. Countermeasures against timing cryptanalysis usually degrade performance, for instance by randomly delaying cryptographic operations or by holding a constant response time without dependence of the algorithms' execution paths.

Clemens Heinrich

References

[1] Bleichenbacher, Daniel (1998). "Chosen ciphertext attacks against protocols based on RSA encryption standard PKCS#1." *Advances in Cryptology—CRYPTO'98*, Lecture Notes in Computer Science, vol. 1462, ed. H. Krawczyk. Springer-Verlag, Berlin.

[2] Brumley, David and Dan Boneh (2003). "Remote Timing Attacks are Practical." Proceedings of the 12th USENIX Security Symposium, Washington, D.C.

[3] Dierks, Tim and Christopher Allen (1999). The TLS Protocol Version 1.0, IETF Internet-Draft RFC 2246, January (expired), http://www.ietf.org/rfc/rfc2246.txt

[4] Dierks, Tim and Eric Rescorla (2002). The TLS Protocol Version 1.1; IETF Internet-Draft, http://www.ietf.org/

[5] Freier, Alan, Philip Karlton, and Paul Kocher (1996). The SSL 3.0 Protocol; Netscape Communications Corp.

[6] Hickman, Kipp (1995). The SSL Protocol; Netscape Communications Corp.

[7] Internet Assigned Numbers Authority (IANA). http://www.iana.org

[8] ISO 7498-2. "Information processing systems—open systems interconnection—basic reference model—Part 2: Security Architecture." ISO International Standard 7498-2; First edition 1989-02-15.

[9] Klima, Vlastimil, Ondrej Pokorny, and Tomas Rosa (2003). "Attacking RSA-based sessions in SSL/TLS." http://eprint.iacr.org/2003/053.

[10] Kocher, Paul (1997). *Timing Attacks on Implementations of Diffie–Hellman, RSA, DSS, and Other Systems. Advances in Cryptology—CRYPTO'97*, Lecture Notes in Computer Science, vol. 1109, ed. B.S. Kaliski Jr. Springer, Berlin, 104–113.

[11] Mitchell, John C., Vitaly Shmatikov, and Ulrich Stern (1998). "Finite state analysis of SSL 3.0." *Proceedings of the 7th USENIX Security Symposium, San Antonio, Texas*.

[12] Paulson, Lawrence C. (1999). "Inductive analysis of the Internet protocol TLS." *Security Protocols: 6th International Workshop 1998*, Lecture Notes in Computer Science, vol. 1550, eds. Bruce Christianson, Bruno Crispo, William S. Harbison, and Michael Roe. Springer, Berlin.

[13] Rescorla, Eric (2000). *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley, Reading, MA.

[14] Wagner, David and Bruce Schneier (1997). Analysis of the SSL 3.0 Protocol (revised).

SECURITY

The security of encryption against unauthorized decryption, unauthorized changing of the data, etc. Security should depend completely on the key. One distinguishes between the following two types of security:

Computational security: quantitative security against unauthorized decryption, based on particular (usually mathematical) assumptions like the inherent difficulty of factoring sufficiently long numbers. Often a *security parameter* denotes the computational level of the security.

Unconditional security: security against unauthorized decryption assuming that the cryptanalyst has unlimited computing facilities (so, security in an information theoretic sense).

Friedrich L. Bauer

Reference

[1] Menezes, A.J., P.C. van Oorschot, and S.A. Vanstone (1997). *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL.

SECURITY ARCHITECTURE

Given any system characterized by a number of devices and/or users communicating with specific communication protocols, the (related) security architecture refers to the enhancing security solution based on cryptographic techniques, protocols, and secure storage, as well as protection of keys and messages.

Examples of security architectures based on Public key techniques include X.509 and EMV. As part of the security architecture, a number of Trusted Third Parties may be defined, such as Registration Authorities, Certification Authorities, and Time Stamping Authorities. These are entities that are not part of the original system as such, but are introduced as part of the security architecture.

Other examples are Kerberos, based on conventional cryptography and bespoke key management architectures e.g. to handle online PIN-code (see Personal Identification Number) verification, which is characterized by key hierarchies, starting with session keys, or data keys at the bottom, which are protected or exchanged by key encryption keys, perhaps comprising several layers, and the top layer consisting of the so-called master keys.

Peter Landrock

SECURITY EVALUATION CRITERIA

Security Evaluation Criteria are usually presented as a set of parameter thresholds that must be met for a system to be evaluated and deemed acceptable. These criteria are established based on a Threat Assessment to establish the extent of the data sensitivity, the security policy, and the system characteristics. The system is evaluated, the evaluation is measured against the criteria, and then an assessment is made of whether or not the system security characteristics meet the requirements as specified by the Security Evaluation Criteria. The criteria is typically unique to each system, the environment it is in and how it is used.

Important past frameworks of security evaluation criteria have been the following:

TCSEC by US Department of Defense (1985): The Trusted Computer System Evaluation Criteria (TCSEC) is a collection of criteria that was previously used to grade or rate the security offered by a computer system product. No new evaluations are being conducted using the TCSEC although there are some still ongoing at this time. The TCSEC is sometimes referred to as the "*Orange Book*" because of its orange cover. The current version is dated 1985 (DOD 5200.28-STD, Library No. S225,711). The TCSEC, its interpretations, and guidelines all have different color covers and are sometimes known as the "*Rainbow Series*" [1]. It is available at http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html.

***ITSEC* by the European Commission (1991):** The Information Technology Security Evaluation Criteria (ITSEC) is a European-developed criteria filling a role roughly equivalent to the TCSEC. Although the ITSEC and TCSEC have many similar requirements, there are some important distinctions. The ITSEC places increased emphasis on integrity and availability, and attempts to provide a uniform approach to the evaluation of both products and systems. The ITSEC also introduces a distinction between doing the right job (effectiveness) and doing the job right (correctness). In so doing, the ITSEC allows less restricted collections of requirements for a system at the expense of more complex and less comparable ratings and the need for effectiveness analysis of the features claimed for the evaluation.

***CTCPEC* by CSE Canada (1993):** The Canadian Trusted Computer Product Evaluation Criteria is the Canadian equivalent of the TCSEC. It is somewhat more flexible than the TCSEC (along the lines of the ITSEC) while maintaining fairly close compatibility with individual TCSEC requirements.

Common Criteria ISO 15408 (2001): In 1990, the Organization for Standardization (ISO) sought to develop a set of international standard evaluation criteria for general use. The CC project was started in 1993 in order to bring all these (and other) efforts together into a single international standard for IT security evaluation. The new criteria was to be responsive to the need for mutual recognition of standardized security evaluation results in a global IT market. The common criteria combine the best aspects of TCSEC and ITSEC and aims to supersede both of them [2].

Tom Caddy
Gerrit Bleumer

References

[1] Trusted Product Evaluation Program: Computer Security Evaluation FAQ. http://www.radium.ncsc.mil/tpep/process/faq.html

[2] www.commoncriteria.org, http://csrc.nist.gov/cc

SECURITY STANDARDS ACTIVITIES

This article describes a number of highly visible security standards activities. It cannot be exhaustive, but it does include many standards bodies that are influencing the security industry and product development. Many of the standards are interrelated; for example, X.509 public key certificates have been profiled for use in the Internet by the PKIX working group of the Internet Engineering Task Force (IETF), and that profile has been augmented for Qualified Certificates, which are used to identify human beings involved in electronic commerce.

X.509: ITU-T Recommendation X.509 defines public key certificates and attribute certificates. ITU-T was previously known as CCITT, which has been developing telecommunications standards for decades. X.509 [40, 41] is part of a joint effort between ITU-T and the International Organization for Standardization (called ISO), which developed the X.500 series of standards. The documents have numbers assigned by both standards bodies, but the numbers assigned by ITU-T are traditionally used to refer to the documents. Within this series, X.509 defines the public key certificate to provide authentication in a ubiquitous global directory environment. While the envisioned directory deployment has never materialized, the certificate format has been used in small, closed, networks as well as large, open, deployments. The public key certificate enables secure communication between entities that were unknown to each other prior to the communication. Deployments of these certificates are known as public key infrastructure (PKI). To bring PKI to large multinational corporations or to millions of Internet users, a common certificate format is necessary. X.509 defines a general, flexible certificate format. The widespread adoption of X.509 is due to two factors. First, X.509 is technically suitable for many environments. Second, it was developed at an important time. It became an international standard at a time when a number of vendors were ready to begin implementing certificate-based products.

X.509 includes a powerful extension mechanism. It was defined for Version 3 certificates and Version 2 CRLs (Certificate Revocation Lists). Arbitrary extensions can be defined and incorporated into a certificate or CRL, and a criticality flag indicates whether or not a certificate using system needs to understand and examine this extension as part of the verification process. Thus, certificate and CRL contents can readily be tailored to specific environments. The inclusion of a particular critical extension can restrict use of the certificate to a particular application environment.

X.509 also specifies the format of the attribute certificate. The attribute certificate is used in conjunction with a public key certificate to provide additional information about the named entity. Attribute certificates are most often used to express authorization information.

Although X.509 is an international standard, the ITU-T continues to maintain the document and develop enhancements. Most of the maintenance takes the form of clarifying text, and most of the enhancements take the forms of new standard extensions. Any problems found through operational experience are addressed in the standard through a formal defect reporting and resolution process.

PKIX: The Internet Engineering Task Force (IETF) is responsible for creating standards for the Internet. For the most part, the IETF develops protocols. This work is carried out by a number of working groups, which are organized into Areas. Within the Security Area, the PKIX (Public Key Infrastructure using X.509) working group was formed at the end of 1995 with the explicit intention of tailoring the X.509 public key certificate to the Internet environment. Essentially, the group set out to define an Internet PKI. Quickly, the group realized that defining an Internet PKI was more extensive than profiling the X.509 certificate. Thus, the PKIX charter was written to encompass four major activities:

1. X.509 certificate and certificate revocation list (CRL) profile;
2. Certificate management protocols;
3. Operational protocols; and
4. Certificate Policy (CP) and Certification Practice Statement (CPS) framework.

The first activity was the original motivating task. The profile includes detailed specification of the mandatory, optional, critical, and non-critical extensions in a *PKIX-compliant* certificate or CRL. The profile was published in January 1999 [15], and updated in April 2002 [37]. The profile is likely to be refined to provide guidance on the use of international character sets [39]. Also, the qualified certificate profile was developed in January 2001 [34], and the attribute certificate profile was developed in April 2002 [38].

The second activity was to specify the protocols for management operations required in the Internet PKI, including certification of entities and their key pairs, certificate revocation, key backup and recovery (see key management), Certification Authority (CA) key rollover, and cross-certification. Two competing protocols were developed: CMP [16] and CMC [29].

The third activity, operational protocols, was to specify the protocols for day-to-day Internet PKI operation, such as certificate retrieval, CRL retrieval, and on-line certificate status checking. The results, to date, include several important specifications. It tells how to use FTP and HTTP to access repositories [20]. Others tell how to use an LDAPv2 directory as a repository [18,21]. Another specification defines the Online Certificate Status Protocol (OCSP) [19]. And, others are being developed to address the use of LDAPv3 directories.

Finally, the fourth activity, guidance to CP and CPS authors, provides topics and formats for these documents. The guidance was originally published

in March 1999 [17]. Since that time, the American Bar Association's Information Security Committee has reviewed it. With the assistance of these lawyers, an update is in progress.

PKIX has played an essential role in bringing PKI concepts to the Internet. The protocols and functions it has defined make a PKI possible, even in the diverse Internet, because their flexibility and generality can satisfy the requirements of greatly differing environments. The PKIX work continues to evolve, and the charter was expanded in 1999 to include additional work items, including time-stamping protocols. The Time-Stamp Protocol (TSP) [36] was published in August 2001.

LDAP: The Lightweight Directory Access Protocol (LDAP) [2] was originally conceived as a simple to describe and simple to implement subset of the capability of the X.500 Directory Access Protocol (DAP). Over time, the subset of functions and features has expanded. Today, it is used as the access protocol for many repositories, some of which are based on X.500 directories, but many are not. As part of this evolution, the "lightweight" aspect of the protocol has diminished. Nevertheless, many vendors worldwide use LDAPv2 [6] and LDAPv3 [8]. The IETF LDAPext Working Group has been formed to specify useful extensions for LDAPv3, such as an authentication and access control mechanism.

An LDAPv2 schema [21] has been specified for LDAP-compliant repositories that contain certificate and CRL information. This facilitates interoperability between PKI products from different vendors in an LDAP environment. In a joint effort between the LDAPext and PKIX working groups, a similar schema is being developed for LDAPv3.

S/MIME: In 1995, a consortium of industry vendors led by RSA Data Security, Inc., developed a companion security solution to the Multipurpose Internet Mail Extensions (MIME) specifications, which are the basis for any email message that goes beyond simple text. For example, an email message that includes bold text or includes an attachment makes use of MIME. Secure MIME (S/MIME) specifies encryption and digital signatures for MIME messages. While a formal standards body did not develop the original S/MIME specifications, many important product vendors embraced S/MIME. To build on and expand this success, the consortium released change control of the S/MIME Version 2 documents [9, 10] to the IETF in 1997.

The IETF S/MIME Working Group developed significant enhancements, resulting in S/MIME Version 3 [22–26]. The primary focus of the S/MIME Working Group was to develop an algorithm independent protocol and incorporate a number of new security features into the specifications, while preserving compatibility with the earlier specification whenever possible. In particular, the S/MIME Version 3 specifications include support for sending encrypted messages to large mail lists, security labels on messages (for example, "company proprietary," "secret," or "top secret"), and signed message receipts. These signed receipts provide proof that the intended recipient received a signed message that contained a request for a receipt.

The S/MIME Version 3 specifications include discussion of PKI concepts such as certificate format, certificate processing, and CRLs. These specifications are compatible with the X.509 profile developed by the PKIX Working Group, and they provide additional details for the use of X.509 certificates in the email environment. Further, provision is made in the message envelope to carry an arbitrary numbers of certificates and CRLs to assist the recipient with the task of path construction and certificate validation.

IPSEC: IPsec is designed to provide interoperable, high quality, cryptographically-based security the Internet Protocol (both Version 4 (IPv4) and Version 6 (IPv6)). The security services offered include access control, connectionless integrity, data origin authentication, protection against replays, confidentiality, and limited traffic flow confidentiality. The services are provided by the use of two traffic security protocols, the Authentication Header (AH) [11] and the Encapsulating Security Payload (ESP) [12], and through the use of cryptographic key management procedures and protocols.

The Internet Key Exchange (IKE) protocol [13] is used to establish the symmetric keying material needed by AH and ESP. IKE provides for strong, X.509-certificate-based authentication of the IP layer entities, and it is compatible with the certificate profile developed by the PKIX Working Group. However, a companion document is being developed to describe details of certificate usage in the IPsec environment.

The IPsec Working Group is working on the second version of IKE. The primary goal of the update is to simplify the protocol. The simplification is targeted at increased interoperability.

TLS: The Transport Layer Security (TLS) specification [7] is the IETF standards-track version of the Secure Sockets Layer Version 3.0 (SSLv3.0)

protocol found in millions of Web browsers and Web servers. The history has many parallels to S/MIME. The original specification was developed outside of any standards body, and then it was released to the IETF, who took over configuration control and made enhancements.

TLS creates a secure channel between two transport layer entities, providing certificate-based authentication, information integrity, and data confidentiality. The TLS specification discusses X.509 certificates, and it is mostly compatible with the profile developed by the PKIX Working Group. The few conflicts are associated with compatibility with SSLv3.0 (and earlier) implementations that were developed well in advance of the PKIX profile. The PKIX X.509 certificate profile appears to meet the goals of the Internet community. Interestingly, non-IETF standards groups are also using the PKIX certificate profile.

AAA: In 1997, the IETF created the first standard Authentication, Authorization, and Accounting (AAA) protocol, called RADIUS (Remote Access Dial-In User Service) [30–32]. As the name implies, RADIUS is designed for use with Dial-In Access Servers. RADIUS has been a big success, displacing many proprietary protocols. RADIUS is widely implemented as Network Access Servers (NASs) serving analog and digital dial-in Point-to-Point Protocol (PPP) service, and it is the prevalent Internet Service Provider (ISP) access model. RADIUS has been adapted for use with DSL (using PPPOE) and cable access (using DOCSIS). RADIUS has been successful because it offers a simple and flexible model for client-server exchanges. However, this simple model does not have sufficient security for some new applications, and it also lacks support for server-initiated control.

The IETF AAA Working Group is responsible for building a more secure and capable AAA protocol. A number of proposals were evaluated in June 2000, and the working group selected the Diameter protocol [35]. Diameter is designed to be upwards compatible with RADIUS, but many of the messaging underpinnings have been upgraded to be more secure. Security is provided using CMS and IPsec. For better response time, the SCTP (Stream Control Transmission Protocol) transport is supported as an alternative.

Diameter explicitly supports server-to-client requests and message forwarding. These capabilities have previously been forced into RADIUS [33]. Diameter also includes explicit support for application suite additions. Application designs have been drafted for Mobile IP authentication and third generation wireless telecommunications [1] sessions.

SPKI: The IETF formed the Simple Public Key Infrastructure (SPKI) Working Group in 1996. In many ways, it is an alternative to PKIX. One fundamental premise of SPKI is that X.509 is a complicated and bulky certificate format that, by explicitly binding a key pair to an identity, rests upon an inherently flawed foundation. SPKI proponents argue that the concept of a globally unique identity will never be realized. Instead, they advocate the use of the public key as an identity. Where necessary and meaningful, a name or other identifying information may be associated with a public key, but this is optional and, it is only intended to have local significance.

The SPKI specifications [27,28] discuss the concepts and philosophy behind this approach to an Internet PKI. A detailed certificate format and processing rules are included. SPKI explicitly encompasses authorization as well as authentication. The sophisticated certificate format makes it possible to express, in a general way, the permitted uses of the certified public key. This capability (not surprisingly) diminishes the intended simplicity of the Simple Public Key Infrastructure. Although SPKI embodies a number of interesting ideas and research contributions, it has not gained widespread support.

OPENPGP: As with the S/MIME and TLS Working Groups, the IETF OpenPGP Working Group was formed to develop a standard based on a protocol that was developed outside of any standards body. The popular Pretty Good Privacy (PGP) email security package was brought to the IETF so that interoperable implementations from different vendors could be developed. OpenPGP [14] defines email message protection and the PGP certificate format (an alternative to both X.509 and SPKI). Despite a loyal installed base, OpenPGP has not seen corporate or government adoption. OpenPGP is viewed as an individual-to-individual solution. The user-centric trust model cannot easily be centrally controlled by an organization.

XML SECURITY: Prominent standards bodies are actively developing XML (eXtensible Markup Language) security specifications, including the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS).

The W3C is developing specifications for the XML syntax with respect to encryption (XML

Encryption) and digital signature (XML Signature), as well as XML protocols for key management (XML Key Management Specification) that allow a client to obtain key information (including values, certificates, management, or trust data) from a Web service.

The OASIS Security Services Technical Committee is developing the Security Assertion Markup Language (SAML), an XML framework for exchanging authentication and authorization information. The underlying authentication mechanism may be PKI-based, but SAML encompasses a number of other authentication technologies as well. A number of other OASIS technical committees are likely to build upon SAML, as well as the W3C specifications mentioned above, to provide security; such committees include Business Transaction Processing (BTP), electronic business XML (ebXML), Provisioning Services Markup Language (PSML), eXtensible Access Control Markup Language (XACML), Web Services Security (WSS), and Digital Signature Services (DSS).

IEEE P802: Local Area Network (LAN) and Metropolitan Area Network (MAN) standards encompass a number of data communications technologies and the applications of these technologies. There is no single technology that is applicable to all applications. Correspondingly, no single local or metropolitan area network standard is adequate for all applications. As a result, the Institute of Electrical and Electronics Engineers (IEEE) Standards Association sponsors several working groups and technical advisory groups within Project 802. Security is the focus of IEEE 802.10, which has seen little market adoption. However, other working groups have also developed security relevant standards.

IEEE 802.1X specifies port-based access controls. It provides a means of authenticating and authorizing devices attached to a LAN port, preventing access when authentication and authorization fails.

IEEE 802.11 includes the ability to encrypt wireless LAN traffic using the Wired Equivalent Privacy (WEP) protocol. Unfortunately, WEP has many flaws. IEEE 802.11 is presently working on a short-term and a long-term replacement for WEP, called TKIP and CCMP, respectively. The Temporal Key Integrity Protocol (TKIP) is intended to replace WEP on current hardware, and it is implemented by firmware and driver software upgrades. The Counter and CBC-MAC Protocol (CCMP) is intended for future generations of product. Future product generations will likely implement both TKIP and CCMP for compatibility with currently fielded devices.

IEEE 802.15 is developing security solutions for personal area networks, and IEEE 802.3 is developing security solutions for some uses of Ethernet. Clearly, more customers are demanding security solutions. Other working groups are likely to have security initiatives in the near future.

IEEE P1363: IEEE Project 1363 is developing standard specifications for public key cryptography, which includes mathematical primitives for key derivation, public-key encryption, and digital signatures. P1363 has been adopted as an IEEE standard, although work continues on a companion document, called IEEE P1363a, which will specify additional techniques. A study group is investigating newer schemes and protocols not considered in P1363 and P1363a; such specifications will appear over time as P1363–1, P1363–2, and so on.

ANSI X9F: The American National Standards Institute (ANSI) committee X9 (Financial Services) develops and publishes standards for the financial services industry. These standards facilitate delivery of financial products and services. Subcommittee X9F is responsible for security, and it includes working groups that focus on cryptographic tools (X9F1), security protocols (X9F3), and digital signature and certification policy (X9F5), among others. X9F has published many standards (the X9 on-line catalog can be found at http://www.x9.org), and many of its standards become international standards through a close working relationship with ISO TC68.

INFLUENTIAL ACTIVITIES: Some activities that are not part of any formal security standards body are influencing security standards development, the security industry, and product development. Again, this discussion cannot be exhaustive, but a number of the highly visible security standards influencing activities are discussed.

U.S. FPKI

The U.S. Federal Public-Key Infrastructure (FPKI) is an initiative by the U.S. Government to define a PKI suitable for its own use. One focus is the production of an acceptable profile for X.509 certificates and CRLs, where there is significant harmonization with the PKIX certificate profile, but the ultimate goal is a full PKI specification. This specification will encompasses all relevant PKI entities, including end entities, registration

authorities (RAs), certification authorities (CAs), and Bridge CAs. It also includes the security-relevant protocols between these entities, as well as the operational policies and procedures required for the PKI.

The U.S. FPKI specifications impose compliance requirements on vendors wanting to sell PKI products to the U.S. Government. To the greatest extent possible, commercial standards have been referenced and profiled. The hope is that the FPKI is sufficiently similar to PKIs for other environments that compliance will not unduly restrict vendors.

The Minimum Interoperability Specifications for PKI Components (MISPC) [5] is one component of the full U.S. FPKI vision. The goal in MISPC is to understand and to specify the minimum functionality required of PKI entities that will still enable them to interoperate usefully with other PKI entities. Thus, for example, the certificate and CRL profile portion of MISPC identifies which of the many optional fields in the X.509 and PKIX specifications must be implemented. Interestingly, MISPC is more than a detailed specification; a CD containing a complete reference implementation compliant with the specification is also available. Thus, vendors have a straightforward way of testing whether their products are MISPC compliant.

GOC PKI

The Government of Canada Public-Key Infrastructure (GOCPKI) is the first large-scale governmental PKI initiative in the world. Its goal similar to the U.S. FPKI, but it defines a PKI suitable for Canadian federal government use. It is a full PKI specification, including certificate and CRL profiles, entity functionality and characteristics, communications protocols, and operational policies and procedures. The GOC PKI will impose compliance requirements on vendors, but it is hoped that this will not preclude Commercial Off-the-Shelf (COTS) products.

JCP

The Java Community Process (JCP) is an open organization of international Java developers and licensees whose charter is to develop and revise Java technology specifications, reference implementations, and technology compatibility kits. This group publishes Java Specification Requests (JSRs), and several are related to security and PKI. For example, JSR 55 discusses certification path creation, building, and verification; JSR 74 discusses many of the Public Key Cryptography Standards (PKCS) published by RSA Laboratories; JSR 104 discusses XML trust services; JSR 105 discusses XML Digital Signature services; JSR 106 discusses XML Digital Encryption services; and JSR 155 discusses Web Services Security Assertions based on the OASIS SAML specification. These and related efforts are expected to eventually be included in future versions of the Java 2 Micro Edition (J2ME), Java 2 Standard Edition (J2SE), and Java 2 Enterprise Edition (J2EE) platforms. Further information can be found at [4].

ICE-CAR

The Interworking Certification Infrastructure for Commerce, Administration and Research (ICE-CAR) project, a successor to the ICE-TEL project, began in January 1999. The objective of this project is to provide all of the technology components that are needed to support the secure use of the Internet for commercial and administrative applications in Europe. These applications include e-commerce, intra-organizational communication, health-care applications, and research. An additional goal was to promote the availability of technically compatible and interconnectable PKIs, which guarantee the authenticity and validity of public keys used in these environments. The project has produced numerous technical reports that are available for download from the *Deliverables* section of the main Web site; see [3] for further details.

Russ Housley

References

[1] See http://www.3gpp2.org/

[2] Howes, T. and M. Smith (1997). *LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol*. Macmillan Technical Publishing, Indianapolis.

[3] See http://ice-car.darmstadt.gmd.de/ice-car-home.html

[4] See http://www.jcp.org/

[5] National Institute of Standards and Technology. Minimum Interperability Specification for PKI Components, Version 1, June 1997.

[6] Yeong, W., T. Howes, and S. Kille (1995). Lightweight Directory Access Protocol. RFC 1777.

[7] Dierks, T. and C. Allen (1999). The TLS Protocol Version 1.0. RFC 2246.

[8] Wahl, M., T. Howes, and S. Kille (1997). Lightweight Directory Access Protocol (v3). RFC 2251.

[9] Dusse, S., P. Hoffman, B. Ramsdell, L. Lundblade, and L. Repka (1998). S/MIME Version 2 Message Specification. RFC 2311.

[10] Dusse, S., P. Hoffman, B. Ramsdell, and J. Weinstein (1998). S/MIME Version 2 Certificate Handling. RFC 2312.
[11] Kent, S. and R. Atkinson (1998). IP Authentication Header. RFC 2402.
[12] Kent, S. and R. Atkinson (1998). IP Encapsulating Security Payload (ESP). RFC 2406.
[13] Harkins, D. and D. Carrel (1998). The Internet Key Exchange (IKE). RFC 2409.
[14] Callas, J., L. Donnerhacke, H. Finney, and R. Thayer (1998). OpenPGP Message Format. RFC 2440.
[15] Housley, R., W. Ford, W. Polk, and D. Solo (1999). Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 2459.
[16] Adams, C. and S. Farrell (1999). Internet X.509. Public Key Infrastructure Certificate Management Protocols. RFC 2510.
[17] Chokhani, S. and W. Ford (1999). Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. RFC 2527.
[18] Boeyen, S., T. Howes, and P. Richard (1999). Internet X.509 Public Key Infrastructure Operational Protocols—LDAPv2. RFC 2559.
[19] Myers, M., R. Ankney, A. Malpani, S. Galperin, and C. Adams (1999). X.509 Internet Public Key Infrastructure Online Certificate Status Protocol—OCSP. RFC 2560.
[20] Housley, R. and P. Hoffman (1999). Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP. RFC 2585.
[21] Boeyen, S., T. Howes, and P. Richard (1999). Internet X.509 Public Key Infrastructure LDAPv2 Schema. RFC 2587.
[22] Housley, R. (1999). Cryptographic Message Syntax. RFC 2630.
[23] Rescorla, E. (1999). Diffie-Hellman Key Agreement Method. RFC 2631.
[24] Ramsdell, B. (ed.). (1999). S/MIME Version 3 Certificate Handling. RFC 2632.
[25] Ramsdell, B. (ed.). (1999). S/MIME Version 3 Message Specification. RFC 2633.
[26] Hoffman, P. (ed.). (1999). Enhanced Security Services for S/MIME. RFC 2634.
[27] Ellison, C. (1999). SPKI Requirements. RFC 2692.
[28] Ellison, C., B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen (1999). SPKI Certificate Theory. RFC 2693.
[29] Myers, M., X. Liu, J. Schaad, and J. Weinstein (2000). Certificate Management Messages over CMS. RFC 2797.
[30] Rigney, C., S. Willens, A. Rubens, and W. Simpson (2000). Remote Authentication Dial in User Service (RADIUS). RFC 2865.
[31] Rigney, C. (2000). RADIUS Accounting. RFC 2866.
[32] Rigney, C., W. Willats, and P. Calhoun (2000). RADIUS Extensions. RFC 2869.
[33] Mitton, D. (2000). Network Access Servers Requirements: Extended RADIUS Practices. RFC 2882.
[34] Santesson, S., W. Polk, P. Barzin, and M. Nystrom (2001). Internet X.509 Public Key Infrastructure Qualified Certificates Profile. RFC 3039.
[35] Mitton, D., M. St.Johns, S. Barkley, D. Nelson, B. Patil, M. Stevens, and B. Wolff (2001). Authentication, Authorization, and Accounting: Protocol Evaluation. RFC 3127.
[36] Adams, C., P. Cain, D. Pinkas, and R. Zuccherato (2001). Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161.
[37] Housley, R., W. Polk, W. Ford, and D. Solo (2002). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280.
[38] Farrell, S. and R. Housley (2002). An Internet Attribute Certificate Profile for Authorization. RFC 3281.
[39] Yergeau, F. (1998). UTF-8, a Transformation Format of ISO 10646. RFC 2279.
[40] ITU-T. (1997). Recommendation X.509: The Directory—Authentication Framework.
[41] ITU-T. (2000). Recommendation X.509: The Directory—Public Key and Attribute Certificate Frameworks.

SELECTIVE FORGERY

Selective forgery is a message related forgery against a cryptographic digital signature scheme. Given a victim's verifying key, a selective forgery is successful if the attacker finds a signature s for a message m selected by the attacker prior to the attack, such that the signature s is valid for m with respect to the victim's verifying key.

Gerrit Bleumer

SELF-SHRINKING GENERATOR

The self-shrinking generator is a clock-controlled generator that has been proposed in [1]; it is strongly related to the shrinking generator, but uses only one Linear Feedback Shift Register (LFSR) R, producing a maximum-length linear sequence.

Its principle is really easy to get: the output sequence of the LFSR is partitioned into pairs of bits. According to the value of the pair, one bit is added to the keystream, and then the pair is discarded and we go to the next one. More precisely:

Pair	Bit added
10	0
11	1
01	no bit added
00	no bit added

EXAMPLE. Let us consider that R has length four, and that its feedback is given by $s_{t+1} = s_t + s_{t-3}$. If the initial state is $s_0 s_1 s_2 s_3 = 1010$, then the output of the LFSR is 10101100100011110101100100011110101100100011110 . . . This gives the following output for the whole scheme: 00101101001011 . . .

A recent survey on the possible attacks is [2].

Caroline Fontaine

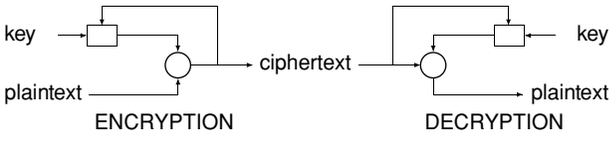
References

[1] Meier, W. and O. Staffelbach (1995). "The self-shrinking generator." *Advances in Cryptology—EUROCRYPT'94*, Lecture Notes in Computer Science, vol. 950, ed. A. De Santis. Springer-Verlag, Berlin, 205–214.

[2] Zenner, E., M. Krause, and S. Lucks (2001). "Improved cryptanalysis of the self-shrinking generator." ACIPS 2001, Lecture Notes in Computer Science, vol. 2119, eds. V. Varadharajan and Y. Mu. Springer-Verlag, Berlin, 21–35.

SELF-SYNCHRONIZING STREAM CIPHER

In a self-synchronizing, or asynchronous, stream cipher, the keystream depends on the secret key of the scheme, but also of a fixed number, say t, of ciphertext digits (that have already been produced, or read; this distinguishes it from a synchronous stream cipher). It can be viewed as follows:



According to its design, such a scheme is able to resynchronize the keystream with the message with just a few correct bits of ciphertext. This means that if some bits are inserted or deleted in the ciphertext, just a small part of the plaintext will not be obtained correctly; the next set of t consecutive correct bits in the ciphertext will be sufficient to resynchronize the keystream and produce the following bits of the plaintext correctly.

Let us now consider that one bit of the ciphertext has been altered during the transmission. This will induce some errors in the decryption of the next t bits; after this, decryption will go on correctly.

What can an active attacker do with such a scheme? According to the propagation of each error in a ciphertext on about t bits of plaintext, it is more difficult for an attacker to forge a plaintext of its choice than in a synchronous stream cipher. Moreover, it is also more difficult for him to desynchronize the keystream, since the scheme is able to resynchronize it by itself. If the attacker wants to desynchronize all the keystream, he has to do a lot of modifications on the ciphertext. Nevertheless, some complementary mechanisms, that can ensure authentication or integrity of the ciphertext are welcome to help the receiver check that all is going well.

At last, since each plaintext digit influences the whole ciphertext (through the feedback of the ciphertext on the keystream generation), the statistical properties of the plaintext are dispersed in the ciphertext, and such a scheme may be more resistant against attacks based on plaintext redundancy, than synchronous stream ciphers.

Good references are [1] and [2].

Caroline Fontaine

References

[1] Maurer, U.M. (1991). "New approaches to the design of self-synchronizing stream ciphers." *Advances in Cryptology—EUROCRYPT'91*, Lecture Notes in Computer Science, vol. 547, ed. D.W. Davies. Springer-Verlag, Berlin, 458–471.

[2] Rueppel, R.A. (1986). *Analysis and Design of Stream Ciphers*. Springer-Verlag, Berlin.

SEMANTIC SECURITY

Semantic security is a notion to describe the security of an encryption scheme.

An adversary is allowed to choose between two plaintexts, m_0 and m_1, and he receives an encryption of either one of the plaintexts. An encryption scheme is semantically secure, if an adversary cannot guess with better probability than 1/2 whether the given ciphertext is an encryption of message m_0 or m_1. The notion is also referred to as *indistinguishability of encryptions* and noted as IND. Historically the word "semantic" came from the definition that the encryption reveals no information no matter what kind of semantics are embedded in the encryption. It has been proven that the definition describing this requirement is equivalent to the indistinguishability of encryptions. The notion of semantic security can be further distinguished by the power of adversary. More specifically, a powerful adversary may have access

to an encryption *oracle* and/or decryption oracle at various stages of the guessing game. Here, an encryption oracle is an oracle that provides an encryption of a queried plaintext, and a decryption oracle provides the decryption of a queried ciphertext (see also random oracle model).

The notion of semantic security can be applied to both symmetric cryptosystems and public key cryptosystems. But since the concrete security analysis of a public key encryption scheme is more tractable, the term is more frequently used to discuss the security of public key encryption schemes.

In a public key encryption scheme, the adversary can always access the encryption oracle, because he can encrypt by himself. Therefore the semantic security must be achieved against such an adversary. Such security is called "semantically secure against chosen plaintext attack" and written IND-CPA. The threat of adversary who has access to decryption oracle is called chosen ciphertext attack (CCA). If a public-key scheme is semantically secure against an adversary who has access to a decryption oracle before determining the pair of plaintexts m_0 and m_1, it is called IND–CCA1. If a public-key scheme is semantically secure against an adversary who has access to a decryption oracle not only before receiving a target ciphertext but also during the guessing stage, then it is defined as IND–CCA2. It is regarded that this type of adversary is the most powerful. Therefore the scheme achieving IND–CCA2 is considered most secure. (There is a restriction on this type of adversary, namely that he cannot receive an answer of the target ciphertext from decryption oracle.)

Besides semantic security, there are related notions such as non-malleability and *plaintext awareness*.

Kazue Sako

Reference

[1] Bellare, M., A. Desai, D. Pointcheval, and P. Rogaway (1998). "Relations among notions of security for public-key encryption schemes." *Advances in Cryptography—CRYPTO'98*, Lecture Notes in Computer Science, vol. 1462, ed. H. Krwawczyk. Springer-Verlag, Berlin, 26–45.

SENDER ANONYMITY

Sender anonymity is achieved in a messaging system if an eavesdropper who picks up messages from the communication line of a recipient cannot tell with better probability than pure guessing who sent the messages. During the attack, the eavesdropper may also listen on all communication lines of the network including those that connect the potential senders to the network and he may send his own messages. It is clear that all messages in such network must be encrypted to the same length in order to keep the attacker from distinguishing different messages by their content or length. The *anonymity* set for any particular message attacked by the eavesdropper is the set of all network participants that have sent message within a certain time window before the attacked message was received. This time window ofcourse depends on latency characteristics and node configurations of the network itself.

Sender anonymity can be achieved against computationally restricted eavesdroppers by MIX networks [1] and against computationally unrestricted eavesdroppers by DC networks [2, 3].

Note that sender anonymity is weaker than sender unobservability, where the attacker cannot even determine whether or not a participant sends a message. Sender unobservability can be achieved with MIX networks and DC networks by adding dummy traffic.

Gerrit Bleumer

References

[1] Chaum, David (1981). "Untraceable electronic mail, return addresses, and digital pseudonyms." *Communications of the ACM*, 24 (2), 84–88.

[2] Chaum, David (1985). "Security without identification: Transaction systems to make big brother obsolete." *Communications of the ACM*, 28 (10), 1030–1044.

[3] Chaum, David (1988). "The dining cryptographers problem: Unconditional sender and recipient untraceability." *Journal of Cryptology*, 1 (1), 65–75.

SEQUENCES

Sequences have many applications in modern communication systems, including signal synchronization, navigation, radar ranging, Code-Division Multiple-Access (CDMA) systems, random number generation, spread-spectrum communications, cryptography, in particular in stream cipher systems.

In stream cipher systems it is essential to construct sequences with good random properties, long periods, and large linear complexity. To achieve many of these goals one often generates

sequences using linear recurrence relations. The *period* of a sequence $\{s_t\}$ is the smallest integer ε such that $s_{t+\varepsilon} = s_t$ for all t. We will explain how the period of a generated sequence is completely determined by the characteristic polynomial of the sequence. For linear sequences the period of the sequences generated can easily be controlled, which makes them good building blocks in stream cipher systems.

A linear recursion of degree n with binary coefficients is given by

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

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

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

The initial state $(s_0, s_1, \ldots, s_{n-1})$ and the given recursion uniquely determines the generated sequence. A linear shift register with a characteristic polynomial $f(x)$ of degree n generates 2^n different sequences corresponding to the 2^n different initial states and these form a vector space over $GF(2)$ which is denoted $\Omega(f)$.

The maximum period of a sequence generated by a linear shift register is at most $2^n - 1$. This follows since a sequence is completely determined by n-successive bits in the sequence and period 2^n is impossible since n successive zeros implies the all zero sequence. Sequences with the maximal period $2^n - 1$ are called m-sequences. For example, with initial state $(s_0, s_1, s_2) = (001)$, then $f(x) = x^3 + x + 1$ generates the m-sequence 0010111.

Example 1. Let the recursion be

$$s_{t+4} + s_{t+3} + s_{t+2} + s_{t+1} + s_t = 0 \pmod 2$$

with characteristic polynomial $f(x) = x^4 + x^3 + x^2 + x + 1$. The sequences in $\Omega(f)$ consists of the $2^4 = 16$ sequences corresponding to the sequences $\{(0), (00011), (00101), (01111)\}$ and their cyclic shifts.

To analyze properties of linear sequences, we associate a generating function $G(x)$ to the sequence $\{s_t\}$, and let

$$G(x) = \sum_{t=0}^{\infty} s_t x^t.$$

Let $f^*(x) = \sum_{i=0}^{n} f_{n-i} x^i$ be the *reciprocal polynomial* of the characteristic polynomial of $f(x)$ of the sequence. Then, we can compute the product

$$\begin{aligned} G(x) f^*(x) &= (s_0 + s_1 x + s_2 x^2 + \cdots) \\ &\quad \times (1 + f_{n-1} x + \cdots + f_1 x^{n-1} + x^n) \\ &= \sum_{t=0}^{\infty} c_t x^t. \end{aligned}$$

The coefficient c_{t+n} of x^{t+n} for any $t \geq 0$ becomes

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

as a consequence of the recurrence relation. Hence,

$$G(x) f^*(x) = \phi^*(x)$$

for some polynomial $\phi^*(x)$ of degree at most $n - 1$. Its reciprocal polynomial $\phi(x)$ is given by

$$\begin{aligned} \phi(x) &= s_0 x^{n-1} + (s_1 + f_{n-1} s_0) x^{n-2} + \cdots \\ &\quad + (s_{n-1} + f_{n-1} s_{n-2} + \cdots + f_1 s_0) \\ &= \sum_{i=0}^{n-1} \left(\sum_{j=0}^{n-1-i} f_{i+j+1} s_j \right) x^i. \end{aligned}$$

There is a one-to-one correspondence between any sequence $\{s_t\}$ in $\Omega(f)$ and any polynomial $\phi^*(x)$ of degree $\leq n - 1$. All sequences generated by $f(x)$ can therefore be described by

$$\Omega(f) = \left\{ \frac{\phi^*(x)}{f^*(x)} \;\middle|\; deg(\phi^*(x)) < deg(f) = n \right\}.$$

For example, the m-sequence 0010111 in $\Omega(x^3 + x + 1)$ can be written

$$\begin{aligned} &\frac{x^2}{1 + x^2 + x^3} \\ &\quad = x^2 + x^4 + x^5 + x^6 + x^9 + x^{11} + x^{12} + \cdots \\ &\quad = (x^2 + x^4 + x^5 + x^6)(1 + x^7 + x^{14} + \cdots \end{aligned}$$

In particular, a simple consequence of the above description of $\Omega(f)$ is that $\Omega(f) \subset \Omega(g)$ if and only if $f(x)$ divides $g(x)$.

The generating function $G(x)$ for a periodic sequence of period ε can be written as

$$\begin{aligned} G(x) &= \left(s_0 + s_1 x + \cdots + s_{\varepsilon-1} x^{\varepsilon-1}\right) \\ &\quad \times (1 + x^{\varepsilon} + x^{2\varepsilon} + \cdots) \\ &= \frac{s_0 + s_1 x + \cdots + s_{\varepsilon-1} x^{\varepsilon}}{1 - x^{\varepsilon}}. \end{aligned}$$

Combining the two expressions for $G(x)$, we obtain the identity

$$(x^{\varepsilon} - 1)\phi(x) = \sigma(x) f(x),$$

where $\sigma(x) = s_0 x^{\varepsilon-1} + s_1 x^{\varepsilon-2} + \cdots + s_{\varepsilon-1}$, contains all the information of a period of the sequence.

The *period of the polynomial* $f(x)$ is the smallest positive integer e such that $f(x)$ divides $x^e - 1$. The

importance of the period e of $f(x)$, is that in order to find the period of all the sequences in $\Omega(f)$, it is enough to find the period of $f(x)$.

Since $f(x)$ divides $x^e - 1$ it follows that $\Omega(f) \subset \Omega(x^e - 1)$, the set of sequences where $s_{t+e} = s_t$, i.e., of period dividing e. Hence, all sequences generated by $f(x)$ has period dividing e. Let the sequence $\{s_t\}$ correspond to the polynomial $\phi(x)$. If $gcd(f(x), \phi(x)) = 1$ then as a consequence of the identity $(x^\varepsilon - 1)\phi(x) = \sigma(x)f(x)$, it follows that $\{s_t\}$ has *smallest* period e, since in this case $f(x)$ must divide $x^\varepsilon - 1$ and thus $\varepsilon \geq e$ which implies that $\varepsilon = e$.

In particular, when $f(x)$ is an irreducible polynomial of period e, then all the nonzero sequences in $\Omega(f)$ have period e. For example the polynomial $f(x) = x^4 + x^3 + x^2 + x + 1$ in Example 1 is irreducible and divides $x^5 + 1$ and has period 5, and therefore all nonzero sequences in $\Omega(f)$ have period 5.

To determine the cycle structure of $\Omega(f)$ for an arbitrary polynomial $f(x)$ that can be factored as $f(x) = \prod f_i(x)^{k_i}$, $f_i(x)$ irreducible, one first needs to determine the cycle structure of $\Omega(f_i^{k_i})$ and then the cycle structure for $\Omega(gh)$ when $gcd(g,h) = 1$.

Cycle structure of $\Omega(f^r)$

Let $f(x)$ be an irreducible polynomial of period e. Let k be defined such that $2^k < r \leq 2^{k+1}$. Then $\Omega(f^r) \backslash \Omega(f)$ contains

$$2^{n2^j} - 2^{n2^{j-1}}$$

sequences of period $e2^j$ for $j = 1, 2, \ldots, k$ and

$$2^{nr} - 2^{n2^k}$$

sequences of period $e2^{k+1}$.

Example 2. Let $f(x) = x^3 + x + 1$ be the characteristic polynomial with $e = 7$, that generates an m-sequence. The number of sequences of each period in $\Omega(f^3)$ is therefore

Number	1	7	56	448
Period	1	7	14	28

Cycle structure of $\Omega(gh)$ *when* $gcd(g, h) = 1$.

In this case it can be shown that each sequence $\{s_t\}$ in $\Omega(gh)$ can be written uniquely

$$\{s_t\} = \{u_t\} + \{v_t\}$$

where $\{u_t\} \in \Omega(g)$ and $\{v_t\} \in \Omega(h)$. Further, the period of the sum $\{u_t\} + \{v_t\}$ is equal to the least common multiple of the period of the two sequences, i.e.,

$$per(s_t) = lcm(per(u_t), per(v_t)).$$

To find the cycle structure of $\Omega(gh)$, suppose $\Omega(g)$ contains d_1 cycles of length λ_1 and $\Omega(h)$ contain d_2 cycles of length λ_2. Add in all possible ways the corresponding $d_1\lambda_1$ sequences from $\Omega(g)$ and the $d_2\lambda_2$ sequences from $\Omega(h)$. This gives $d_1\lambda_1 d_2\lambda_2$ distinct sequences all of period $lcm(\lambda_1, \lambda_2)$. Formally we can write this as $[d_1(\lambda_1)][d_2(\lambda_2)] = [d(\lambda)]$ where $d = d_1 d_2 gcd(\lambda_1, \lambda_2)$ and $\lambda = lcm(\lambda_1, \lambda_2)$.

Example 3. Let $f_1(x) = x^3 + x + 1$ and $f_2(x) = x^4 + x^3 + x^2 + x + 1$. The cycle structure of $\Omega(f_1)$ can be written $[1(1) + 1(7)]$, and similarly for $\Omega(f_1)$ as $[1(1) + 3(5)]$. Combining the cycle structure as described above, gives the cycle structure $[1(1) + 1(7) + 3(5) + 3(35)]$.

The discussion above shows that the period of all sequences in $\Omega(f)$ is completely determined from the periods of the divisors of $f(x)$. This way of controlling the periods is one of the main reasons for using linear recursions as building blocks in stream ciphers.

The sequence $\{s_t\}$ can be expressed in terms of the zeros of its characteristic polynomial $f(x)$ of degree n. In the case when the zeros of $f(x)$ are simple, which is the case when the sequence has odd period, then $\{s_t\}$ has a unique expansion in the form

$$s_t = \sum_{i=1}^{n} a_i \alpha_i^t$$

for some constants a_i and where α_i, $1 \leq i \leq n$ are the zeros of $f(x)$.

The main problem with linear recursions in cryptography is that it is easy to reconstruct a sequence $\{s_t\}$ generated by a characteristic polynomial $f(x)$ of degree n from the knowledge of $2n$ consecutive bits in $\{s_t\}$, since this gives a system of n equations for determining the unknown coefficients of $f(x)$. The Berlekamp–Massey algorithm is an efficient method for finding $f(x)$ in this way. Several methods exist to increase the linear span, i.e, the smallest degree of the linear recursion that generates the sequence. We just mention a few simple ones obtained by multiplying sequences.

Let $\{u_t\}$ and $\{v_t\}$ be two sequences of odd period. Then, $u_t = \sum a_i \alpha_i^t$ where α_i, $1 \leq i \leq n$, are the zeros of the characteristic polynomial of $\{u_t\}$ and $v_t = \sum b_j \beta_j^t$ where β_j, $1 \leq j \leq m$, are the zeros of the characteristic polynomial of $\{v_t\}$. Then the sequence $\{w_t\} = \{u_t v_t\}$ can be written as

$$w_t = \sum a_i b_j (\alpha_i \beta_j)^t.$$

This shows that $\{w_t\}$ is generated by the polynomial with the, at most, nm different zeros $\alpha_i \beta_j$ for

$1 \le i \le n,\ 1 \le j \le n$. If $gcd(per(u_t), per(v_t)) = 1$, then it can be shown that $per(w_t) = per(u_t)per(v_t)$. However, the number of zeros and ones in sequence $\{w_t\}$ will in general not be *balanced* even if this is the case for $\{u_t\}$ and $\{v_t\}$. This follows since $w_t = 1$ if and only if $u_t = v_t = 1$, and thus only 1/4 of the elements in $\{w_t\}$ will be 1's when $\{u_t\}$ and $\{v_t\}$ are balanced.

Often one considers sequences of the form

$$w_t = s_{t+\tau_1} s_{t+\tau_2} \dots s_{t+\tau_k},$$

where s_t is an m-sequence. A closer study of the zeros of the characteristic polynomial of $\{w_t\}$ shows that the linear span is at most $\sum_{i=1}^{k} \binom{n}{i}$ and frequently the equality holds.

Every Boolean function in n variables, $f(x_1, x_2, \dots, x_n)$, can be written uniquely as the sum

$$f(x_1, x_2, \dots, x_n) = u_0 + \sum_{i=1}^{n} u_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} u_{ij} x_i x_j + \dots + u_{12\dots n} x_1 x_2 \cdots x_n$$

with binary coefficients (see *algebraic normal* form in Boolean functions).

One can determine the linear span obtained by combining n different m-sequences $\{a_t\}, \{b_t\}, \dots, \{c_t\}$ with characteristic polynomials of pair-wise relative prime degrees $e_1, e_2, \dots, e_n$ using a Boolean combining function. From the Boolean function $f(x_1, x_2, \dots, x_n)$ we construct a sequence $w_t = f(a_t, b_t, \dots, c_t)$. Then the linear span of the combined sequence is equal to $f(e_1, e_2, \dots, e_n)$, evaluated over the integers.

It is important in applications of sequences in communication systems as well as in stream cipher systems to generate sequences with good auto- and cross-correlation properties.

Let $\{u(t)\}$ and $\{v(t)\}$ be two binary sequences of period e. The cross-correlation of the sequences $\{u(t)\}$ and $\{v(t)\}$ at shift τ is defined as

$$C_{u,v}(\tau) = \sum_{t=0}^{e-1} (-1)^{u_{t+\tau} - v_t}$$

where the sum $t + \tau$ is computed modulo e. In the case when the two sequences are the same, we denote this by the auto-correlation at shift τ.

For synchronization purposes one prefers sequences with low absolute values of the maximal out-of-phase auto-correlation, i.e., $|C_{u,u}(\tau)|$ should be small for all values of $\tau \neq 0 \pmod e$.

Let $\mathcal{F}$ be a family consisting of M sequences

$$\mathcal{F} = \{s_i(t) : i = 1, 2, \cdots, M\},$$

where each sequence $\{s_i(t)\}$ has period e.

The cross-correlation between two sequences $\{s_i(t)\}$ and $\{s_j(t)\}$ at shift τ is denoted by $C_{i,j}(\tau)$. In Code-Division Multiple-Access (CDMA) applications it is desirable to have a family of sequences with certain properties. To facilitate synchronization, it is desirable that all the out-of-phase auto-correlation values ($i = j, \tau \neq 0$) are small. To minimize the interference due to the other users in a multiple access situation, the cross-correlation values ($i \neq j$) must also be kept small. For this reason the family of sequences should be designed to minimize

$$C_{max} = max\{|C_{i,j}| : 1 \le i, j \le M, \text{ and either } i \neq j \text{ or } \tau \neq 0\}.$$

For practical applications in communication systems one needs a family $\mathcal{F}$ of sequences of period e, such that the number of users $M = |\mathcal{F}|$ is large and simultaneously $C_{\max}$ is small. Also in stream ciphers it is of importance that the generated sequences have good auto-correlation and cross-correlation properties.

Tor Helleseth

References

[1] Golomb, S.W. (1982). *Shift Register Sequences*. Aegean Park Press, Laguna Hills, CA.

[2] Helleseth, T. and P.V. Kumar (1999). "Pseudonoise sequences." *The Mobile Communications Handbook*, ed. J.D. Gibson. Chapter 8. CRC/IEEE Press, Boca Raton, FL/Piscataway, NJ.

[3] Helleseth, T. and P.V. Kumar (1998). "Sequences with low correlation." *Handbook of Coding Theory*, eds. V.S. Pless and W.C. Huffman. Chapter 21. Elsevier, Amsterdam.

[4] Selmer, E.S. (1966). *Linear Recurrence Relations over Finite Fields*. Lecture Notes, Department of Mathematics, University of Bergen, Norway.

[5] Zierler, N. (1959). "Linear recurring sequences." *J. Soc. Indust. Appl. Math.*, 7, 31–48.

SERPENT

Serpent is a 128-bit block cipher designed by Anderson et al. and first published in 1998 [1]. Later that year the cipher was slightly modified [2] and proposed as a candidate for the *Advanced Encryption Standard* (Rijndael/AES). In 1999 it was selected as one of the five finalists of the AES competition.

Serpent is a 32-round substitution–permutation (SP) network operating on 128-bit blocks. Each round consists of a key mixing operation, a layer of 32 copies of a 4×4-bit S-box, and (except in the last round) a linear transformation. The replicated

S-box differs from round to round and is selected from a set of eight different S-boxes. The last (incomplete) round is followed by a final key mixing operation. An additional bit permutation before the first round and after the last key mixing layer is applied to all data entering and leaving the SP network. The 128-bit subkeys mixed with the data in each round are generated by linearly expanding a 128-bit, 192-bit, or 256-bit secret key, and passing the result through a layer of S-boxes.

The initial and final permutations, the S-boxes, and the linear transformation have all been designed in order to allow an optimized implementation in software using the "bitslice" technique [3]. The idea is to construct a complete description of the cipher using only logical bit-operations (as in hardware) and then execute 32 (or 64) operations in parallel on a 32-bit (or 64-bit) processor.

Serpent is considered to have a rather high security margin. The best attacks published so far break about 1/3 of the rounds. Kelsey, Kohno, and Schneier [7] presented a first attack breaking 9 rounds with a time complexity slightly faster than exhaustive key search. This amplified boomerang attack was improved and extended by one round by Biham et al. [5]. The best attacks so far are the linear and the differential-linear attacks presented in [4] and [6]. Both break 11 rounds out of 32.

Christophe De Cannière

References

[1] Anderson, R.J., E. Biham, and L.R. Knudsen (1998). "Serpent: A new block cipher proposal." *Fast Software Encryption, FSE'98*, Lecture Notes in Computer Science, vol. 1372, ed. S. Vaudenay. Springer-Verlag, Berlin, 222–238.

[2] Anderson, R.J., E. Biham, and L.R. Knudsen (1998). "Serpent: A proposal for the advanced encryption standard." *Proceedings of the First AES Candidate Conference. National Institute of Standards and Technology, August*.

[3] Biham, E. (1997). "A fast new DES implementation in software." *Fast Software Encryption, FSE'97*, Lecture Notes in Computer Science, vol. 1267, ed. E. Biham. Springer-Verlag, Berlin, 260–272.

[4] Biham, E., O. Dunkelman, and N. Keller (2002). "Linear cryptanalysis of reduced round Serpent." *Fast Software Encryption, FSE 2001*, Lecture Notes in Computer Science, vol. 2355, ed. M. Matsui. Springer-Verlag, Berlin, 16–27.

[5] Biham, E., O. Dunkelman, and N. Keller (2002). "New results on boomerang and rectangle attacks." *Fast Software Encryption, FSE 2002*, Lecture Notes in Computer Science, vol. 2365, eds. J. Daemen and V. Rijmen. Springer-Verlag, Berlin, 1–16.

[6] Biham, E., O. Dunkelman, and N. Keller (2003). "Differential-linear cryptanalysis of Serpent." *Fast Software Encryption, FSE 2003*, Lecture Notes in Computer Science, vol. 2887, ed. T. Johansson. Springer-Verlag, Berlin, 9–21.

[7] Kelsey, J., T. Kohno, and B. Schneier (2001). "Amplified boomerang attacks against reduced-round MARS and Serpent." *Fast Software Encryption, FSE 2000*, Lecture Notes in Computer Science, vol. 1978, ed. B. Schneier. Springer-Verlag, Berlin, 75–93.

SET

SET (Secure Electronic Transactions) is a standard for a payment protocol for credit card payments over the Internet and was developed in 1996–97 as a joint initiative of MasterCard, VISA, IBM, Microsoft, Netscape and others as a more secure alternative to Secure Socket Layer SSL, which never really caught on.

SET assumes the existence of appropriate infrastructure within the card organisation, and entails the communication between the registered Payer (cardholder), Payee (merchant) and the Payment Gateway Provider, i.e. the Acquirer or a Payment Service Provider. The main purpose of the protocol is to secure this communication in such a way that neither the Payee, nor the Payment Gateway Provider can access all purchase transaction details. Thus the Payee has access to the order information only and not the credit card details, while the Payment Gateway Provider has access to the payment information only.

SET is a PKI-solution (see public key infrastructure). The Certificate Authority (CA) hierarchy consists of a Root CA that signs the certificates of each of the credit card brand CA's. These CA's sign certificates for the Cardholder CA (the Card Issuer), Merchant CA (the Customer Acquirer) and the Payment CA. These CA's then in turn sign the certificates for the cardholder, merchant, and payment gateway provider, respectively, using the X.509 v3 format. Neither cardholder's name, nor card number are shown in the certificates. Rather a number is used that has been computed from the credit card number and other input by the Issuer.

In short, the protocol works as follows: the cardholder indicates that he wants to initiate payment for his order. The merchant then identifies himself with his certificate and provides the cardholder with the public key of the payment gateway provider. The cardholder encrypts the payment

information using this key, thus ensuring the merchant cannot access this information and signs the payment instruction. The merchant forwards the payment information to the payment gateway provider in an authorization request, and the payment gateway provider verifies the content and authorizes accordingly.

Peter Landrock

SHA FAMILY (SECURE HASH ALGORITHM)

The SHA (Secure Hash Algorithm) Family designates a family of six different hash functions: SHA-0, SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 [7, 8]. They take variable length input messages and hash them to fixed-length outputs. The first four operate on 512-bit message blocks divided into 32-bit words and the last two on 1024-bit blocks divided into 64-bit words. SHA-0 (the first version of SHA since replaced by SHA-1) and SHA-1 produce a message digest of 160 bits, SHA-224 of 224 bits, SHA-256 of 256 bits, SHA-384 of 384 bits and SHA-512 of 512 bits respectively. All six functions start by padding the message according to the so-called Merkle-Damgård strengthening technique. Next, the message is processed block by block by the underlying compression function. This function initializes an appropriate number of chaining variables to a fixed value to hash the first message block, and to the current hash value for the following message blocks. Each step i of the compression function updates in turn one of the chaining variables according to one message word W_i. As there are more steps in the compression function than words in a message block, an additional message schedule is applied to expand the message block. In the last step, the initial value of the chaining variable is added to each variable to form the current hash value (or the final one if no more message blocks are available). The following provides an overview of SHA-1, SHA-256, and SHA-512. SHA-0 is almost identical to SHA-1, SHA-224 to SHA-256 and SHA-384 to SHA-512.

PADDING: The message is appended with a binary one and right-padded with a variable number of zeros followed by the length of the original message coded over two binary words. The total padded message length must be a multiple of the message block size.

SHA-1 Compression Function

Five 32-bit chaining variables *A, B, C, D, E* are either initialized to

$$A \leftarrow IV_1 = \texttt{67452301}_x$$
$$B \leftarrow IV_2 = \texttt{EFCDAB89}_x$$
$$C \leftarrow IV_3 = \texttt{98BADCFE}_x$$
$$D \leftarrow IV_4 = \texttt{10325476}_x$$
$$E \leftarrow IV_5 = \texttt{C3D2E1F0}_x$$

for the first 512-bit message block or to the current hash value for the following message blocks. The first sixteen words of the message schedule are initialized to input message words. The following 64 message schedule words W_i are computed as

$$W_i \leftarrow (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \lll 1, \quad 16 \le i \le 79$$

where "$\oplus$" represents bit-wise exclusive-or, and "$X \lll n$" is the cyclic rotation of X to the left by n bits. Then the compression function works as follows:

for $i = 0$ to 79 do
$T \leftarrow W_i + A \lll 5 + f_i(B, C, D) + E + K_i \mod 2^{32}$
$B \leftarrow A$
$C \leftarrow B \lll 30$
$D \leftarrow C$
$E \leftarrow D$
$A \leftarrow T$

where the nonlinear functions f_i are defined by

$$f_{if}(X, Y, Z) = (X \wedge Y) | (\neg X \wedge Z), \quad 0 \le i \le 19$$
$$f_{xor}(X, Y, Z) = (X \oplus Y \oplus Z), \quad 20 \le i \le 39, \quad 60 \le i \le 79$$
$$f_{maj}(X, Y, Z) = ((X \wedge Y) | (X \wedge Z) | (Y \wedge Z), \quad 40 \le i \le 59$$

and the constants K_i by

$$K_i \leftarrow \texttt{5A827999}_x, 0 \le i \le 19$$
$$K_i \leftarrow \texttt{6ED9EBA1}_x, 20 \le i \le 39$$
$$K_i \leftarrow \texttt{8F1BBCDC}_x, 40 \le i \le 59$$
$$K_i \leftarrow \texttt{CA62C1D6}_x, 60 \le i \le 79.$$

After 80 steps, the output value of each chaining variable is added to the previous intermediate hash value according to the Davies–Meyer construction to give the new intermediate hash value. When all consecutive message blocks have been

hashed, the last intermediate hash value is the final overall hash value.

SHA-0

The only difference between SHA-1 and SHA-0 is the fact that there is no left rotation by one bit in the message schedule of SHA-0. In other words, the 64 message schedule words W_i for SHA-0 are computed as

$$W_i \leftarrow W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}, \ \ 16 \leq i \leq 79.$$

SHA-256 and SHA-512 Compression Functions

Eight chaining variables A, B, C, D, E, F, G, H are initialized to fixed values H_0 to H_7 for the first message block, and to the current intermediate hash value for the following blocks. The first sixteen w-bit words (where $w = 32$ for SHA-256 and $w = 64$ for SHA-512) of the message schedule are initialized to the input message words. The following $r - 16$ (where $r = 64$ for SHA-256 and $r = 80$ for SHA-512) message schedule words W_i are computed as

$$W_i = \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} \bmod 2^w, \qquad 16 \leq i \leq r-1$$

where σ_0 and σ_1 represent linear combinations of three rotated values of the input variable. Then the compression function works as follows:

```
for i = 0 to r do
  T1 ← H + Σ1(E) + fif(E, F, G) + Ki
       + Wi  mod 2^w
  T2 ← Σ0 (A) + fmaj(A, B, C)  mod 2^w
  H ← G
  G ← F
  F ← E
  E ← D + T1  mod 2^w
  D ← C
  C ← B
  B ← A
  A ← T1 + T2  mod 2^w
```

where Σ_0 and Σ_1 again represent linear combinations of three rotated values of the input variable and K_i is a different w-bit constant for each step i. Finally, the output value of each chaining variable is added to the previous intermediate hash value according to the Davies–Meyer construction to give the new intermediate hash value. When all consecutive message blocks have been hashed, the last intermediate hash value is the final overall hash value.

SHA-224

The SHA-224 hash computations are exactly the same as those of SHA-256, up to the following two differences: the constants H_0 to H_7 used in SHA-224 are not the same as those used in SHA-256, and the SHA-224 output is obtained by truncating the final overall hash value to its 224 leftmost bits.

SHA-384

The SHA-384 hash computations are exactly the same as those of SHA-512, up to the following two differences: the constants H_0 to H_7 used in SHA-384 are not the same as those used in SHA-512, and the SHA-384 output is obtained by truncating the final overall hash value to its 6 leftmost words.

SECURITY CONSIDERATION: All six SHA functions belong to the MD4 type hash functions and were introduced by the American National Institute for Standards and Technology (NIST). SHA was published as a Federal Information Processing Standard (FIPS) in 1993. This early version is known as SHA-0. In 1994, a minor change to SHA-0 was made, and published as SHA-1 [7]. SHA-1 was subsequently standardized by ISO [5]. The following generation of SHA functions with much larger message digest sizes, namely 256, 384, and 512 bits, was introduced in 2000 and adopted as a FIPS standard in 2002 [8] as well as an ISO standard in 2003 [5]. The latest member of the family, namely SHA-224, was adopted in a Change Notice to FIPS 180-2 in 2004. This latter generation of hash functions provides theoretical security levels against collision search attacks which are consistent with the security levels expected from the three standard key sizes of the *Advanced Encryption Standard* (see Rijndael/AES) (128, 192, and 256 bits). The first attack known on SHA-0 is by Chabaud and Joux [2]. They show that in about 2^{61} evaluations of the compression function it is possible to find two messages hashing to the same value whereas a brute-force attack exploiting the birthday paradox requires about 2^{80} evaluations in theory. In 2004, Biham and Chen introduce the neutral bit technique and find near-collisions on the compression function of SHA-0 [1] as well as collisions on reduced-round versions of SHA-1. In August 2004, Joux, Carribault, Jalby and Lemuet [6] first provide a full collision on SHA-0 using two four-block messages and requiring a complexity of 2^{51} compression function computations. In February 2005, Wang, Yin and Yu [10] announce full collisions on SHA-0 in 2^{39} hash operations and report that collisions on SHA-1 can be obtained in less than 2^{69} hash operations. Saarinen [9] addresses

the existence of slid pairs in SHA-1. The first security analysis on SHA-256, SHA-384 and SHA-512 in 2003 is by Gilbert and Handschuh [3]. They show that collisions can be found with a reduced work factor for weakened variants of these functions. Subsequently, Hawkes and Rose show that second pre-image attacks are far easier than expected on SHA-256 [4]. However these observations do not lead to actual attacks in 2004.

Helena Handschuh

References

[1] Biham, E. and R. Chen (2004). "Near-collisions of SHA-0." *Advances in Cryptology—CRYPTO 2004*, Lecture Notes in Computer Science, vol. 3152, ed. M. Franklin. Springer-Verlag, Berlin, 290–305.

[2] Chabaud, F. and A. Joux (1998). "Differential collisions in SHA-0." *Advances in Cryptology—CRYPTO'98*, Lecture Notes in Computer Science, vol. 1462, ed. H. Krawczyk. Springer-Verlag, Berlin, 56–71.

[3] Gilbert, H. and H. Handschuh (2004). "Security analysis of SHA-256 and sisters." *Selected Areas in Cryptography—SAC 2003*, Lecture Notes in Computer Science, vol. 3006, eds. M. Matsui and R. Zuccherato. Springer-Verlag, Berlin, 175–193.

[4] Hawkes P. and G. Rose (2004). On Corrective Patterns for the SHA-2 Family. http://eprint.iacr.org/2004/207.

[5] ISO/IEC 10118-3 (2003). "Information technology—security techniques—hash-functions—Part 3: Dedicated hash-functions."

[6] Joux, A., P. Carribault, W. Jalby, and C. Lemuet (2004). "Collisions in SHA-0." Presented at the rump session of *CRYPTO 2004*, August.

[7] National Institute of Standards and Technology (NIST). (1995). FIPS Publication 180-1: Secure Hash Standard.

[8] National Institute of Standards and Technology (NIST). (2002). FIPS Publication 180-2: Secure Hash Standard.

[9] Saarinen, M.-J.O. (2003). "Cryptanalysis of block ciphers based on SHA-1 and MD5." *Fast Software Encryption—FSE 2003*, Lecture Notes in Computer Science, vol. 2887, ed. Thomas Johansson. Springer-Verlag, Berlin, 36–44.

[10] Wang, X., Y.-L. Yin, and H. Yu (2005). Collision Search Attacks on SHA-1. Unpublished manuscript.

SHAMIR'S THRESHOLD SCHEME

In [1], Shamir proposed an elegant "polynomial" construction of a perfect threshold schemes (see threshold cryptography). An (n, k)-threshold scheme is a particular case of secret sharing scheme when any set of k or more participants can recover the secret exactly while any set of less than k particiants gains no additional, i.e. a posteriori, information about the secret. Such threshold schemes are called *perfect* and they were constructed in [2] and [1]. Shamir's construction is the following.

Assume that the set S_0 of secrets is some finite field $GF(q)$ of q elements (q should be prime power) and that the number of participants of SSS $n < q$. The dealer chooses n different nonzero elements (points) $x_1, \ldots, x_n \in GF(q)$, which are publicly known. To distribute a secret s_0 the dealer generates randomly coefficients $g_1, \ldots, g_{k-1} \in GF(q)$, forms the polynomial $g(x) = s_0 + g_1 x + \cdots + g_{k-1}x^{k-1}$ of degree less than k and sends to the i-th articipant the share $s_i = g(x_i)$. Clearly any k participants can recover the whole polynomial $g(x)$ and, in particular, its zero coefficient (or $g(0)$), since any polynomial of degree l is uniquely determined by its values in $l + 1$ points and Lagrange interpolation formula shows how to determine it. On the other hand, the point 0 can be considered as an "evaluation point" x_0, corresponding to the dealer, since $s_0 = g(0)$. Then the above consideration shows that for any given shares $s_1 = g(x_1), \ldots, s_{k-1} = g(x_i)$ all possible values of s_0 are equally probable, hence the scheme is perfect.

For some applications it is convenient to have the maximal possible number n of participants equal to q, especially for $q = 2^m$. For Shamir's scheme $n < q$ but the following simple modification allows to have $n = q$. Namely, the dealer generates a random polynomial of the form $f(x) = f_0 + f_1 x + \cdots + f_{k-2}x^{k-2} + s_0 x^{k-1}$ and distribute shares $s_i = f(x_i)$, where the x_i are different but not necessary nonzero elements of $GF(q)$. The perfectness of this scheme can be proved either directly (along the line of the above proof by considering the polynomial $h(x) = f(x) - s_0 x^{k-1}$ of degree at most $k - 2$), or as an application of established in [3] the relationship between perfect (n, k)-threshold schemes and $(n + 1, k)$ Reed–Solomon codes (see cyclic codes), since the above construction is equivalent to so-called 2-lengthening of Reed–Solomon codes.

Robert Blakley
Gregory Kabatiansky

References

[1] Shamir, A. (1979). "How to share a secret." *Communications of the ACM*, 22 (1), 612–613.

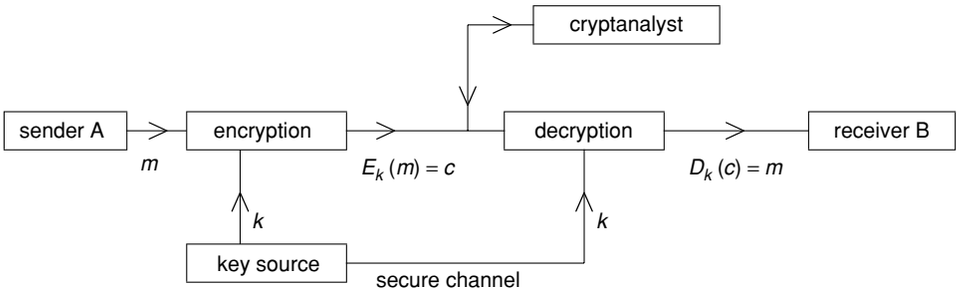


Fig. 1. The conventional cryptosystem

[2] Blakley, R. (1979). "Safeguarding cryptographic keys." *Proceedings of AFIPS 1979 National Computer Conference*, 48, 313–317.
[3] McEliece, R.J. and D.V. Sarwate (1981). "On secret haring and Reed–Solomon codes." *Communications of the ACM,* 24, 583–584.

SHANNON'S MODEL

Although symmetric cryptosystems have been around for at least two thousand years (see for instance Caesar cipher), it was only in 1949 that Claude Shannon gave a formal mathematical description of these systems [1].

In his description, a sender A (often called Alice) wants to send a message m to a receiver B (who is called Bob). The message is called a *plaintext* and is taken from a finite set, called plaintext space $\mathcal{M}$. Of course, Alice may send more messages.

Since the transmission channel is insecure (a person called Eve is also connected to the channel), Alice applies a mapping E_k to m. The result c is called the *ciphertext* and is an element of a set $\mathcal{C}$, the ciphertext space. The mapping E_k is called the encryption function. It is c that Alice sends to Bob and so it will be c that is intercepted by Eve.

Clearly, the encryption function E_k must be a one-to-one mapping, since Bob must be able to retrieve the plaintext/message m from the ciphertext c by means of the *decryption* function D_k. In formula: $D_k(c) = m$.

Since more people may want to use the same cryptosystem and since Alice and Bob do not want to use the same mapping too long for security reasons, their function is taken from a large set $\mathcal{E}$ of one-to-one mappings from $\mathcal{M}$ to $\mathcal{C}$. It is for this reason that the encryption and decryption functions carry a label k. This k is called the key and is taken from the so-called *key-space* $\mathcal{K}$. It is the set $\mathcal{E} = \{E_k \mid k \in \mathcal{K}\}$ that describes the cryptosystem. Quite clearly Alice and Bob must use the same key k. To this end, they use a *secure channel*, a communication line without any eavesdroppers. A possibility is that they agreed beforehand on the key, another possibility is that one has sent the key by means of a courier to the other. Nowadays public key cryptography is often used for this purpose.

Normally, the same cryptosystem $\mathcal{E}$ will be used for a long time and by many people, so it is reasonable to assume that $\mathcal{E}$ is also known to the cryptanalyst. It is the frequent changing of the key that has to provide the security of the data. This principle was already clearly stated by the Dutchman Auguste Kerckhoff (see maxims) in the 19th century.

Often $\mathcal{M} = \mathcal{C}$ in which case one wants the number of plaintexts that are mapped to a particular ciphertext (under different keys) to be the same. In that case the ciphertext does not give any information about the plaintext (see information theory).

The cryptanalyst who is connected to the transmission line can be:

Passive (eavesdropping): The cryptanalyst tries to find m (or even better k) from c.

Active (tampering): The cryptanalyst tries to actively manipulate the data that are being transmitted. For instance, she alters a transmitted ciphertext.

Henk van Tilborg

Reference

[1] Shannon, C.E. (1949). "Communication theory and secrecy systems." *Bell Systems Technical Journal*, 28, 656–715.

SHARE

Share is a portion of information distributed by a secret sharing scheme (SSS) to a given user. In the standard definition of SSS, shares are distributed

via secure, private channels in such a way that each participant only knows his own share [1, 2]. We note that it is also possible to organize SSS in case of public channels [3].

Robert Blakley
Gregory Kabatiansky

References

[1] Shamir, A. (1979). "How to share a secret." *Communications of the ACM*, 22 (1), 612–613.
[2] Blakley, R. (1979). "Safeguarding cryptographic keys." *Proceedings of AFIPS 1979 National Computer Conference*, 48, 313–317.
[3] Beimel, A. and B. Chor (1998). "Secret sharing with public reconstruction." *IEEE Transactions on Informatiom Theory*, 44 (5), 1887–1896.

SHORTEST VECTOR PROBLEM

The Shortest Vector Problem (SVP) is the most famous and widely studied computational problem on lattices. Given a lattice $\mathcal{L}$ (typically represented by a basis), SVP asks to find the shortest nonzero vector in $\mathcal{L}$. The problem can be defined with respect to any norm, but the Euclidean norm is the most common (see the entry lattice for a definition). A variant of SVP (commonly studied in computational complexity theory) only asks to compute the length (denoted $\lambda(\mathcal{L})$) of the shortest nonzero vector in $\mathcal{L}$, without necessarily finding the vector.

SVP has been studied by mathematicians (in the equivalent language of quadratic forms) since the 19th century because of its connection to many problems in the number theory. One of the earliest references to SVP in the computer science literature is [7], where the problem is conjectured to be NP-hard.

A cornerstone result about SVP is *Minkowski's first theorem*, which states that the shortest nonzero vector in any n-dimentional lattice has length at most $\gamma_n \det(L)^{1/n}$, where γ_n is an absolute constant (approximately equal to $\sqrt{n}$) that depends only of the dimension n, and $\det(L)$ is the determinant of the lattice (see the entry lattice for a definition).

The upper bound provided by Minkowski's theorem is tight, i.e., there are lattices such that the shortest nonzero vector has length $\gamma_n \det(L)^{1/n}$. However, general lattices may contain vectors much shorter than that. Moreover, Minkowski's theorem only proves that short vectors exist, i.e., it does not give an efficient algorithmic procedure to find such vectors. An algorithm to find the shortest nonzero vector in two-dimensional lattices was already known to Gauss in the 19th century, but no general methods to efficiently find (approximately) shortest vectors in n-dimentional lattices were known until the early 1980s. A g-approximation algorithm for SVP is an algorithm that on input a lattice $\mathcal{L}$, outputs a nonzero lattice vector of length at most g times the length of the shortest vector in the lattice. The LLL lattice reduction algorithm ([4], see lattice reduction) can be used to approximate SVP within a factor $g = O((2/\sqrt{3})^n)$ where n is the dimension of the lattice. Smaller approximation factors (slightly subexponential in n—see subexponential time for a definition) can be achieved in polynomial time using more complex algorithms like Schnorr's *Block Korkine–Zolotarev* reduction [6].

No efficient (polynomial time) algorithm to compute the length of the shortest vector in a lattice is known to date (leave alone actually finding the shortest vector). The NP-hardness of SVP (in the Euclidean norm) was conjectured by van Emde Boas in 1981 [7]. The conjecture remained wide open until 1997, when Ajtai proved that SVP is NP-hard to solve exactly under randomized reductions [1]. The strongest NP-hardness result for SVP known to date is due to Micciancio [5], who showed that SVP is NP-hard even to approximate within any factor less than $\sqrt{2}$. Stronger (but still subpolynomial) inapproximability results are known for SVP in the ℓ_∞ norm [2]. On the other hand, Goldreich and Goldwasser [3] have shown that (under standard complexity assumptions) SVP cannot be NP-hard to approximate within small polynomial factors $g = O(\sqrt{n/\log n})$.

As is the case with the related Closest Vector Problem, finding a good approximation algorithm (i.e., a polynomial-time algorithm with polynomial approximation factors) is one of the most important open questions in the area. Indeed, the hardness of approximating SVP within certain polynomial factors can be used as the basis for the construction of provably secure cryptographic functions (see lattice based cryptography).

Daniele Micciancio

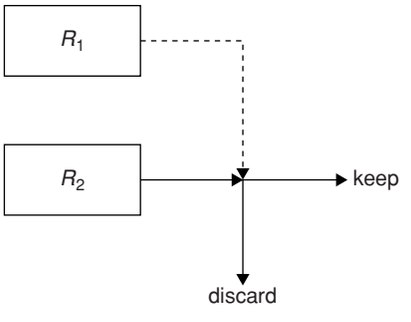
References

[1] Ajtai, M. (1998). "The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract)." *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing—STOC'98, Dallas, TX*. ACM Press, New York, 10–19.

[2] Dinur, I. (2000). "Approximating SVP_∞ to within almost-polynomial factors is NP-hard." *Proceedings of the 4th Italian Conference on Algorithms and Complexity—CIAC 2000, Rome*, Lecture Notes in Computer Science, vol. 1767, eds. G. Bongiovanni, G. Gambosi, and R. Petreschi. Springer, Berlin, 263–276.
[3] Goldreich, O. and S. Goldwasser (2000). "On the limits of nonapproximability of lattice problems." *Journal of Computer and System Sciences*, 60 (3), 540–563. Preliminary version in *STOC'98*.
[4] Lenstra, A.K., H.W. Lenstra, Jr., and L. Lovász (1982). "Factoring polynomials with rational coefficients." *Mathematische Annalen*, 261, 513–534.
[5] Micciancio, D. "The shortest vector problem is NP-hard to approximate to within some constant." *SIAM Journal on Computing*, 30 (6), 2008–2035. Preliminary version in *FOCS'98*.
[6] Schnorr, C.-P. (1987). "A hierarchy of polynomial time lattice basis reduction algorithms." *Theoretical Computer Science*, 53 (2–3), 201–224.
[7] van Emde Boas, P. (1981). "Another NP-complete problem and the complexity of computing short vectors in a lattice." Technical Report 81-04, Mathematische Instituut, University of Amsterdam, Available on-line at URL http://turing.wins.uva.nl/peter/

SHRINKING GENERATOR

The shrinking generator is a clock-controlled generator that has been proposed in 1993 [1]. It is based on two Linear Feedback Shift Registers (LFSRs), say R_1 and R_2. The idea is that R_1's output will *decimate* R_2's output. At each step, both are clocked; if R_1 output a 1, then R_2's output bit is included in the keystream, else (if R_1 outputs a 0) R_2's output bit is discarded.



EXAMPLE. Let us consider R_1 of length three, with the feedback relation $s_{t+1} = s_t + s_{t-2}$, and R_2 of lenth four, with the feedback relation $s_{t+1} = s_t + s_{t-3}$. Then the following happens (the first row concerns only the initialization; the internal states are of the form $s_t s_{t-1} s_{t-2}$ or $s_t s_{t-1} s_{t-2} s_{t-3}$):

R_1		R_2		
State	Output	State	Output	Output
010		0101		
001	0	1010	1	
100	1	1101	0	0
110	0	0110	1	
111	0	0011	0	
011	1	1001	1	1
101	1	0100	1	1
010	1	0010	0	0
001	0	0001	0	
100	1	1000	1	1
110	0	1100	0	
111	0	1110	0	
011	1	1111	0	0
101	1	0111	1	1
⋮	⋮	⋮	⋮	⋮

The inventors discussed some security points in their paper. More recent results have been given in [2, 5]. A discussion about the implementation and the use of a buffer (in order to avoid the irregular rate of the output) is presented in [3] and [4].

Caroline Fontaine

References

[1] Coppersmith, D., H. Krawczyk, and Y. Mansour (1994). "The shrinking generator." *Advances in Cryptology—CRYPTO'93*, Lecture Notes in Computer Science, vol. 773, ed. D.R. Stinson. Springer-Verlag, Berlin, 22–39.
[2] Golic, J. (1995). "Intrinsic statistical weakness of keystream generators". *Advances in Cryptology—ASIACRYPT'94*, Lecture Notes in Computer Science, vol. 917, eds. J. Pieprzyk and R. Safari-Naini. Springer-Verlag, Berlin, 91–103.
[3] Kessler, I. and H. Krawczyk (1995). "Minimum buffer length and clock rate for the shrinking generator cryptosystem." IBM Research Report RC19938, IBM T.J. Watson Research Center, Yorktown Heights, NY, USA.
[4] Krawczyk, H. (1994). "The shrinking generator: Some practical considerations." *Fast Software Encryption*, Lecture Notes in Computer Science, vol. 809, ed. R.J. Anderson. Springer-Verlag, Berlin, 45–46.

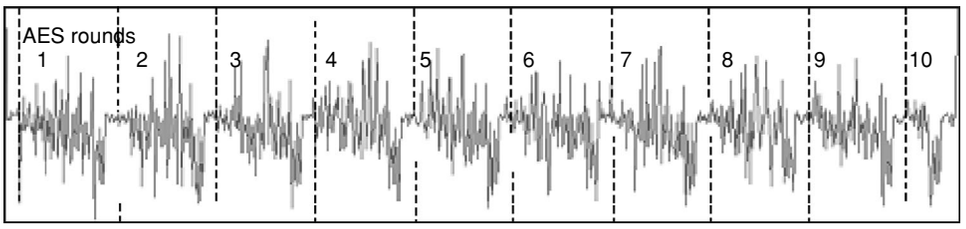


Fig. 1. AES power trace

[5] Johansson, T. (1998). "Reduced complexity correlation attack on two clock-controlled generators." *Advances in Crptology—ASIACRYPT'98*, Lecture Notes in Computer Science, vol. 1514, eds. K. Ohta and D. Pei. Springer-Verlag, Berlin, 342–356.

SIDE-CHANNEL ANALYSIS

INTRODUCTION: Electronic devices have to comply with consumption constraints especially on autonomous equipments, like mobile phones. Power analysis has been included into most certification processes regarding products dealing with information security such as smart cards.

The electrical consumption of any electronic device can be measured with a resistor inserted between the ground or Vcc pins and the actualground in order to transform the supplied current into a voltage easily monitored with an oscilloscope.

Within a micro-controller the peripherals consume differently. For instance writing into non-volatile memory requires more energy than reading. Certain chips for smart cards enclose a crypto-processor, i.e., a particular device dedicated to specific cryptographic operations, which generally entails a consumption increase. The consumption trace of a program running inside a micro-controller or a microprocessor is full of information. The signal analysis may disclose lots of things about the used resources or about the process itself. This illustrates the notion of *side channel* as a source of additional information.

Basically a power consumption trace exhibits large scale patterns most often related to the structure of the executed code. The picture below (Figure 1) shows the power trace of a smart-card chip ciphering a message with the Advanced Encryption Standard (AES). The ten rounds are easily recognised with nine almost regular patterns first followed by a shorter one.

Zooming into a power signal exhibits a local behaviour in close relationship with the silicon technology. At the cycle scale, the consumption curve looks roughly like a capacitive charge and discharge response.

A careful study of several traces of a same code with various input data shows certain locations where power trace patterns have different heights. The concerned cycles indicate some data dependence also called *information leakage*. They may be magnified by a variance analysis over a large number of executions with random data. For instance, by ciphering many random plaintexts with a secret-key algorithm, it is possible to distinguish the areas sensitive to input messages from the constant areas that correspond to the key schedule.

INFORMATION LEAKAGE MODEL: The characterisation of data leakage (namely, finding the relationships between the data and the variability of consumption) has been investigated by several researchers. The most common model consists in correlating these variations to the Hamming weight of the handled data, i.e., the number of nonzero bits. Such a model is valid for a large number of

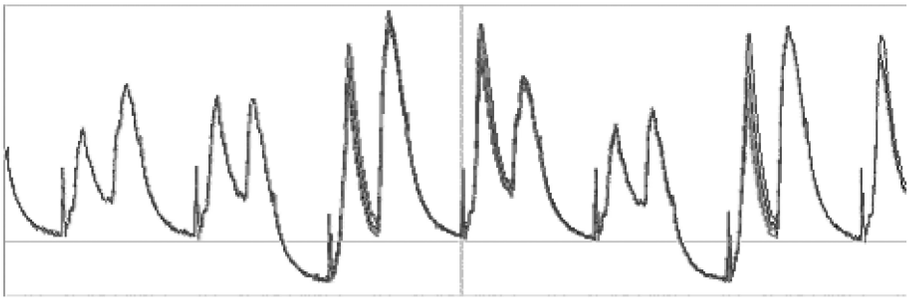


Fig. 2. Information leakage

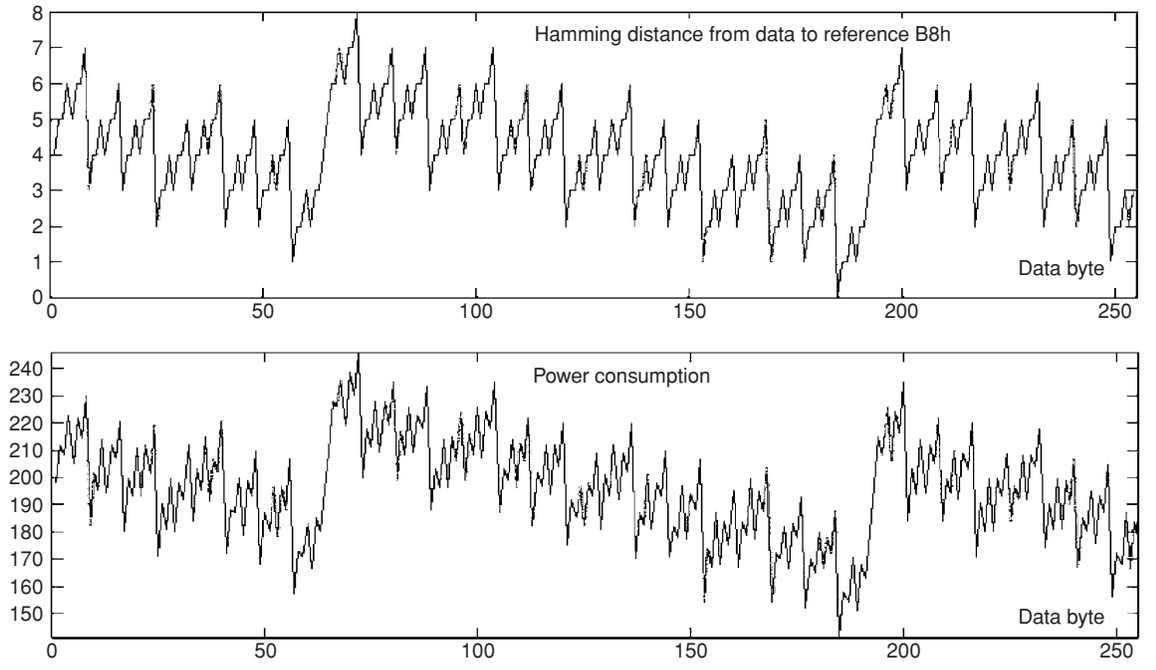


Fig. 3. Transition model

devices. However it can be considered as a special case of the transition model which assumes that the energy is consumed according to the number of bits switched for going from one state to the next one. This behaviour is represented by the Hamming distance between the data and some *a priori* unknown constant, i.e., the Hamming weight of the data XOR-ed with this constant.

As shown in the next picture (Figure 3), for an 8-bit micro-controller, the transition model may seem rough but it suffices to explain many situations, provided that the reference constant state is known. In most microprocessors this state is either an address or an operating code. Each of them has a specific binary representation and therefore a different impact in the power consumption: this is why each cycle pattern is most often different from its neighbours.

Some technologies systematically go through a clear "all-zeros" state that explains the simpler Hamming-weight model.

STATISTICAL ANALYSES: With information leakage models in mind, it is possible to designstatistical methods in order to analyse the data leakage. They require a large amount of power traces assigned to many executions of the same-code with varying data, generally at random, and make use of statistical estimators such as averages, variances and correlations. The most famous method is due to Paul Kocher et al. and is called Differential Power Analysis (DPA).

Basically the purpose of DPA is to magnify the effect of a single bit inside a machine word. Suppose that a random word in a Ω-bit processor is known and uniformly distributed. Suppose further that the associated power consumption obeys the Hamming-weight model. On average the Hamming weight of this word is $\Omega/2$. Given N words, two populations can be distinguished according to an arbitrary selection bit: the first population, $\mathcal{S}_0$, is the set of t words whose selection bit is 0 and the second population, $\mathcal{S}_1$, is the set

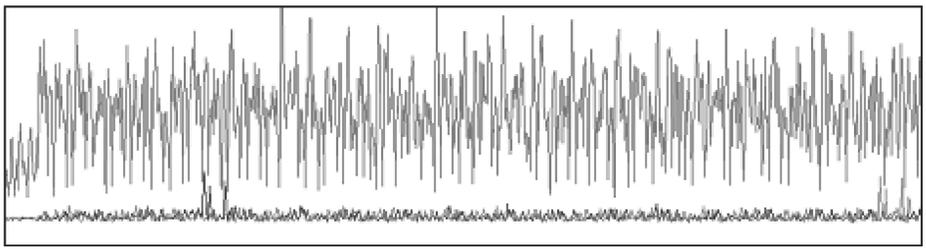


Fig. 4. Bit tracing (upper curve: power consumption of a single execution; two lower curves: DPA curves respectively tracing the first and last data bit of a targetted process)

of N–t words whose selection bit is 1. On average, the words of set $\mathcal{S}_0$ will have a Hamming weight of $(\Omega - 1)/2$ whereas the words of set $\mathcal{S}_1$ will have a Hamming weight of $(\Omega + 1)/2$. The same bias can be seen through the corresponding power consumption traces since it is supposed to be correlated with the Hamming weight of the data. Let C_0 and C_1 respectively denote the averaged power consumption traces of the blue curvesets $\mathcal{S}_0$ and $\mathcal{S}_1$. The *DPA trace* is defined as the difference $C_0 - C_1$.

The resulting DPA curve has the property of erecting bias peaks at moments when the selection bit is handled. It looks like noise everywhere else: indeed, the constant components of the signal are cancelled by the subtraction whereas dynamic ones are faded by averaging, because they are not coherent with the selection bit.

This approach is very generic and applies to many situations. It works similarly with the transition model. Of course the weight of a single selection bit is relatively more important in processors with short words like 8-bit chips. If the machine word is larger, the same DPA bias can be obtained by increasing the number of trials.

A first application of DPA is called *bit tracing*. It is a useful reverse engineering tool for monitoring a predictable bit during the course of a process. In principle a DPA peak rises up each time it is processed. This brings a lot of information about an algorithm implementation. To achieve the same goal Paul Fahn and Peter Pearson proposed another statistical approach called *Inferential Power Analysis* (IPA). The bits are inferred from the deviation between a single trace and an average trace possibly resulting from the same execution: for instance the average trace of a DES round (see Data Encryption Standard) can be computed over its sixteen instances taken from a single execution. IPA does not require the knowledge of the random data to make a prediction on a bit value. But as counterpart it is less easy to implement and the interpretation is less obvious.

After Paul Kocher, Thomas Messerges et al. have proposed to extend DPA by considering multiple selection bits in order to increase the signal to noise ratio (SNR). If the whole machine word is taken into account, a global approach consists in considering the transition model as suggested by Jean-Sébastien Coron et al.

FROM POWER ANALYSIS TO POWER ATTACKS: Obviously, if the power consumption is sensitive to the executed code or handled data, critical information may leak through power analysis. This section explains how to turn a side-channel analysis into an attack.

```
k ← bitsize(d)
y ← x
for i = k – 2 downto 0 do
    y ← y² (mod n)
    if (bit i of d is 1) then y ← y · x (mod n)
endfor
return y
```

Fig. 5. Square-and-multiply exponentiation algorithm

SPA-Type Attacks

A first type of power attacks is based on *Simple Power Analysis* (SPA). For example, when applied to an *unprotected* implementation of an RSA public key encryption scheme, such an attack may recover the whole private key (i.e., signing or decryption key) from a single power trace.

Suppose that a private RSA exponentiation, $y = x^d \bmod n$ (see modular arithmetic), is carried out with the square-and-multiply algorithm (see also exponentiation algorithms). This algorithm processes the exponent bits from left to right. At each step there is a squaring, and when the processed bit is 1 there is an additional multiplication. A straightforward (i.e., unprotected) implementation of the square-and-multiply algorithm is given in Figure 5.

The corresponding power curve exhibits a sequence of consumption patterns among which some have a low level and some have a high level. These calculation units are assigned to a cryptoprocessor handling n-bit arithmetic. Knowing that a low level corresponds to a squaring and that a high level corresponds to a multiplication, it is fairly easy to read the exponent value from the power trace:

- a low-level pattern followed by another low-level pattern indicates that the exponent bit is 0, and
- a low-level pattern followed by a high-level pattern indicates that the exponent bit is 1.

This previous picture also illustrates why the Hamming weight of exponent d can be disclosed by a timing measurement.

DPA-Type Attacks

Historically, DPA-type attacks—that is, power attacks based on *Differential Power Analysis* (DPA)—were presented as a means to retrieve the bits of a DES key.

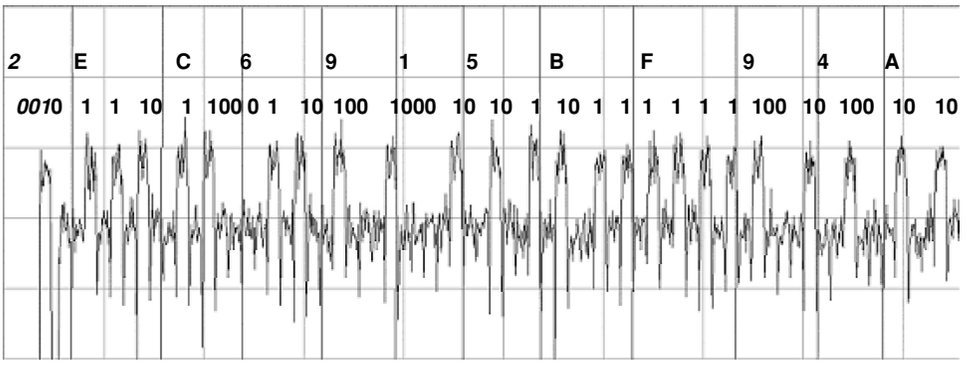


Fig. 6. SPA trace of the basic square-and-multiply algorithm

At the first round of DES, the output nibble of the ith S-box ($1 \leq i \leq 8$) can be written as $\mathrm{S}_i(M \oplus K)$ where

- M is made of 6 bits constructed from the input message after IP- and E-permutations: it has to be chosen at random but is perfectly known and predictable, and
- K is a 6-bit sub-key derived from the key scheduling.

Rising up a DPA bias would require the knowledge of the output nibble. As K is unknown to the adversary this is not possible. But sub-key K can be easily exhausted as it can take only $2^6 = 64$ possible values. Therefore the procedure consists in reiterating the following process for $0 \leq \widehat{K} \leq 63$:

1. form sets $\mathcal{S}_0 = \left\{M \mid g(\text{S-box}_i(M \oplus \widehat{K})) = 0\right\}$ and $\mathcal{S}_1 = \left\{M \mid g(\text{S-box}_i(M \oplus \widehat{K})) = 1\right\}$ where selection function g returns the value of a given bit in the output nibble; and
2. compute the corresponding DPA curve.

In principle the bias peak should be maximised when the guess $\widehat{K}$ is equal to the real sub-key K. Then inverting the key schedule permutation leads to the value of 6 key bits. In other words the DPA operator is used to validate sub-key hypotheses. The same procedure applies to the 7 other S-boxes of the DES. Therefore the whole procedure yields $8 \times 6 = 48$ key bits. The 8 remaining key bits can be recovered either by exhaustion or by conducting a similar attack on the second round.

The main feature of a DPA-type attack resides in its genericity. Indeed it can be adapted to many cryptographic routines as soon as varying and known data are combined with secret data through logical or arithmetic operations.

A similar attack can be mounted against the first round of Rijndael/AES; the difference being that there are 16 byte-wise bijective substitutions and therefore 256 guesses for each. Finally, we note that DPA-type attacks are not limited to symmetric algorithms, they also apply to certain (implementations of) asymmetric algorithms, albeit in a less direct manner.

Other Attacks

Amongst the other statistical attacks, IPA is more difficult and less efficient. Its purpose is to retrieve key bits without knowing the processed data. It proceeds by comparing the power trace of a DES round with an average power trace computed for instance over the 16 rounds. In principle, key bits could be inferred this way because the differential curve should magnify the bits deviation where they are manipulated.

Dictionary (or template) attacks can be considered as a generalisation of IPA to very comfortable but realistic situations. They have been widely studied in the field of smart cards where information on secret key or personal identification numbers (PIN) could potentially be extracted. They consist in building a complete dictionary of all possible secret values together with the corresponding side-channel behaviour (e.g., power trace) when processed by the device (e.g., for authentication purpose). Then a secret value embedded in a twin device taken from the field can be retrieved by comparing its trace and the entries of the dictionary.

In practice, things do not happen that easily for statistical reasons and application restrictions. Only part of the secret is disclosed and the information leakage remains difficult to exploit fully.

Finally, in addition to power consumption, other side channels can be considered; possible sources of information leakage include running time or electro-magnetic radiation.

COUNTERMEASURES: The aforementioned attacks have all been published during the second half of the 1990s. In view of this new threat the manufacturers of cryptographic tokens have

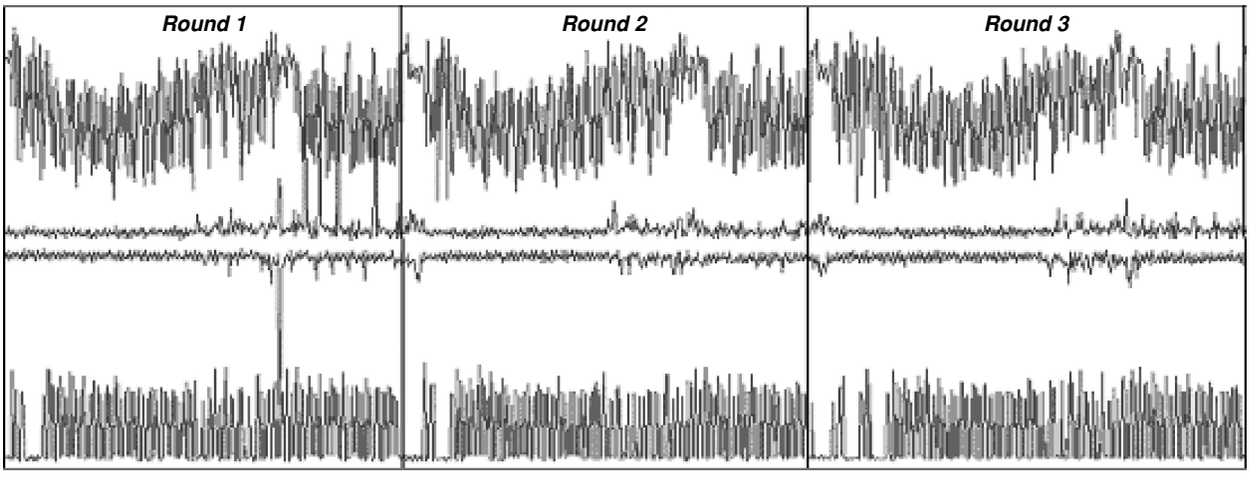


Fig. 7. DPA trace of the three first rounds of DES (two upper (respectively lower) curves: power consumption curve of maxima (respectively minima) of a single execution and DPA curve of maxima (respectively minima))

designed a large set of dedicated countermeasures especially to thwart statistical attacks like DPA. All the related research activity has now resulted in tamper resistant devices widely available in the market. It has given rise to the new concept of "secure implementation" which states that information leakage is not only due to the specification of an application (cryptographic processing or whatever), but also to the way it is implemented.

If information leaks through a physical side-channel there are two defensive strategies. The first consists in decorrelating the secret data from the side-channel. The second consists in decorrelating the side-channel from the secret data. The borderline between both is sometimes fuzzy but roughly speaking, the former is rather software oriented and intends to mask the data since they have to leak anyway, whereas the latter is more hardware oriented and intends to shut the side-channel physically in order to make the device tamper-resistant.

Chip manufacturers have introduced into their hardware designs many security features against power attacks. They are stricto sensu countermeasures since they aim at impeding the power measurement and make the recorded signal unworkable.

- Some countermeasures consist in blurring the signal using smoothing techniques, additive noise or desynchronisation effects. These countermeasures are poorly efficient against SPA working on broad scale traces. They are rather designed against statistical attacks. They may require some complementary circuits to generate parasitic components into the consumed current. Desynchronisation aims at misaligning a set of power traces by the means of unstable clocking or the insertion of dummy cycles at random, making the statistical combination of several curves ineffective.
- Other countermeasures rather intend to decrease or cancel the signal at the source. Reduction is a natural consequence of the shrinking trend in the silicon industry that diminishes the power consumption of each elementary gate. More interesting (and expensive) is the emerging technology called "precharged dual rail logic" where each bit is represented by a double circuitry. At a given time a logical 0 is represented physically by a 01, and a logical 1 by 10. The transition to the next time unit goes through a physical 00 or 11 state so that the same amount of switching occurs whatever the subsequent state is. Consequently if both rails are perfectly balanced, the overall consumption of a microprocessor does not depend on the data anymore.

Software countermeasures enclose a large variety of techniques going from the application level to the most specific algorithmic tricks. One can classify them into three categories: application constraints, timing counter-measures and data masking.

- Application constraints represent an obvious but often forgotten means to thwart statistical analyses. For instance DPA requires known data with a high variability. An application wherein an input challenge (or an output cryptogram) would be strictly formatted, partially visible and constrained to vary within hard limits (like a counter) would resist DPA fairly well.
- Timing countermeasures mean the usage of empirical programming tricks in order to tune the time progress of a process. A critical instruction

may have its execution instant randomised by software: if it never occurs at the same time, statistical analysis becomes more difficult. Conversely other situations require the code to be executed in a constant time, in order to protect it from SPA or timing analysis. For instance a conditional branch may be compensated with a piece of fake code with similar duration and electrical appearance.

- Data masking (also known as whitening or randomization), covers a large set of numerical techniques designed by cryptographers and declined in various manners according to the algorithm they apply to. Their purpose is to prevent the data from being handled in clear and to disable any prediction regarding their behavior when seen through the side channel. For example, the modular exponentiation $y = x^d \bmod n$ (as used in the RSA public key cryptosystem) can be evaluated as:

$$y = \{(x + r_1 n)^{d + r_2 \varphi(n)} \bmod r_3 n\} \bmod n$$

for randoms r_i and where ϕ denotes Euler totient function.

To illustrate how fuzzy the borderline between hardware and software countermeasures can be, we have mentioned that for instance desynchronisation can be implemented by hardware or software means. The same remark applies to data masking for which some manufacturers have designed dedicated hardware tokens or mechanisms such as bus encryption or wired fast implementations of symmetric algorithms.

The experience shows that combined countermeasures act in synergy and increase the complexity in a much larger proportion than the sum of both. For instance the simple combination of desynchronisation tricks and data masking makes DPA (or more sophisticated variants thereof) quite harmless. In the same way, new hardware designs resist the most state-of-the-art and best equipped experts.

Marc Joye
Francis Olivier

References

[1] Chari, Suresh, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi (1999). "Towards sound approaches to counteract power-analysis attacks." *Advances in Cryptology—CRYPTO'99*, Lecture Notes in Computer Science, vol. 1666, ed. M. Wiener. Springer-Verlag, Berlin, 398–412.

[2] Coron, Jean-Sébastien, Paul Kocher, and David Naccache (2001). "Statistics and secret leakage." *Financial Cryptography (FC 2000)*, Lecture Notes in Computer Science, vol. 1962, ed. Y. Frankel. Springer-Verlag, Berlin, 157–173.

[3] Fahn, Paul N. and Peter K. Pearson (1999). "IPA: A new class of power attacks." *Cryptographic Hardware and Embedded Systems (CHES'99)*, Lecture Notes in Computer Science, vol. 1717, eds. Ç.K. Koç and C. Paar. Springer-Verlag, Berlin, 173–186.

[4] Gandolfi, Karine, Christophe Mourtel, and Francis Olivier (2001). "Electromagnetic analysis: Concrete results." *Cryptographic Hardware and Embedded Systems—CHES 2001*, Lecture Notes in Computer Science, vol. 2162, eds. Ç.K. Koç, D. Naccache, and C. Paar. Springer-Verlag, Berlin, 251–261.

[5] Kocher, Paul (1996). "Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems." *Advances in Cryptology—CRYPTO'96*, Lecture Notes in Computer Science, vol. 1109, ed. N. Koblitz. Springer-Verlag, Berlin, 104–113.

[6] Kocher, Paul, Joshua Jaffe, and Benjamin Jun (1999). "Differential power analysis." *Advances in Cryptology—CRYPTO'99*, Lecture Notes in Computer Science, vol. 1666, ed. M. Wiener. Springer-Verlag, Berlin, 388–397.

[7] Messerges, Thomas S. (2000). "Using second-order power analysis to attack DPA resistant software." *Cryptographic Hardware and Embedded Systems—CHES 2000*, Lecture Notes in Computer Science, vol. 1965, eds. Ç.K. Koç and C. Paar. Springer-Verlag, Berlin, 238–251.

[8] Messerges, Thomas S., Ezzy A. Dabbish, and Robert H. Sloan (2002). "Examining smart-card security under the threat of power analysis attacks." *IEEE Transactions on Computers*, 51 (5), 541–552.

SIDE-CHANNEL ATTACKS

Side-Channel Attacks or Environmental Attacks of cryptographic modules exploit characteristic information extracted from the implementation of the cryptographic primitives and protocols. This characteristic information can be extracted from timing, power consumption, or electromagnetic radiation features (see tempest). Other forms of side-channel information can be a result of hardware or software faults, computational errors, and changes in frequency or temperature. Side-channel attacks make use of the characteristics of the hardware and software elements as well as the implementation structure of the cryptographic primitive. Therefore, in contrast to analyzing the mathematical structure and properties of the cryptographic primitives only, side-channel analysis also includes the implementation. Some implementations are more vulnerable to specific side-channel attacks than others. Examples of attacks based on side-channel analysis are Differential Power Attacks examining power traces (see Differential Power Analysis), Timing Attacks

measuring the amount of time used to complete cryptographic operations (see Timing Attack), and Fault Induction Attacks exploiting errors in the computation process of cryptographic primitives (see Fault Attacks).

Tom Caddy

SIEVING

Sieving refers to a process for selecting candidates for further processing among a set of elements. The "sieve" is the test that an element must pass to be considered further.

In many cases, by employing arithmetic progressions, it is possible to identify multiple candidates from the set more efficiently than if each element were tested separately. (Indeed, sometimes the term "sieving" refers only to this speedup.) For instance, in the *Sieve of Eratosthenes* (see prime number), candidate primes are selected from a range of integers by crossing off elements divisible by small primes 2, 3, 5, 7, 11, 13, ... Crossing off every second, third, fifth, seventh element and so on is generally faster than testing each element separately for divisibility by small primes.

Sieving is the first and major phase of the fastest general algorithms for integer factoring and for solving the discrete logarithm problem. Here, the candidates sought are those that are divisible only by small primes or their equivalent (see smoothness and factor base). Specific examples of sieving are described further in the entries Number Field Sieve, Quadratic Sieve, sieving in function fields, and index calculus. See also TWIRL for a recent design for efficient sieving in hardware.

Burt Kaliski

SIEVING IN FUNCTION FIELDS

Function fields are analogous constructions to number fields, where the role of the integers is replaced by polynomials. The coefficients of these polynomials are elements of finite fields for all cryptographically relevant applications. But in contrast to number fields, function fields over finite fields (so they are called) have interesting properties, notably concerning smoothness of elements, an important topic for sieving.

Notably, there exists a provable bound for the necessary size of a *factor base*, a set of elements generating a larger, targeted set of elements to be factored. This is due to the fact that the analog of the Riemann hypothesis has been proven in the function field case. The most important application to cryptography (although a theoretical result) is the existence of a provable subexponential-time algorithm by Adleman et al. in 1992 [1] for solving the discrete logarithm problem in the Jacobian of a hyperelliptic curve (in short called hyperelliptic cryptosystems), a generalization of the group of points of an elliptic curve, and an analog to the ideal class group of a quadratic number field in function fields. The result is mostly of a theoretical nature, since hyperelliptic cryptosystems, as proposed by Koblitz in 1989 [8], are considered to be not practical enough because of their complicated arithmetic. Yet there exist some implementations in the group around Frey showing this performance is not as bad as expected, especially because the size of the elements is considerably smaller than for elliptic curves which might make them even more suitable for small computing devices such as smart cards.

The first implementation actually solving hyperelliptic cryptosystems has been done by R. Flassenberg and the author in 1997 [4]. They applied a sieving technique to accelerate a variant of the Hafner–McCurley algorithm [6] (known to solve discrete logarithms (see discrete logarithm problem) in ideal class groups of quadratic number fields). The basic idea of sieving in function fields is to find a good representation of polynomials by integers, which allows one to "jump" from one polynomial to another, and to increment the exponent in the cell of a three dimensional matrix. All the other optimizations known from the number field sieve could then be applied. Later, Smart [9] compared the Hafner-McCurley variant with the original, theoretically faster, Adleman–De Marrais–Huang variant which allows one to construct sparse systems of linear equations, therefore being better suited for curves of larger genus (the genus basically being the size of the discriminant of the function field). It turned out that the size of the cryptosystems N.P. Smart experienced with was still too small. Later on, N.P. Smart implemented the sieving technique for superelliptic cryptosystems (where the degree of the corresponding function field is at least 3) based on a joint work with Galbraith and the author [5]. None of these implementations was ever even close to the size real cryptosystems would use, but no one ever started a massively parallel project as for the number field sieve for these types of cryptosystems. As a consequence, one cannot sincerely decide about the practical usefulness of hyperelliptic cryptosystems.

This is very different to the other application of sieving in function fields: namely, to compute

the discrete logarithm in a finite field. This can be done by constructing for a given finite field a function field with the following property: there is an embedding of subgroups of the multiplicative group of the finite field into the Jacobian of a curve corresponding to the field. This mapping has been used for applying the Adleman-De Marrais-Huang result to finite fields with small characteristic and high degree, resulting in a subexponential algorithm for discrete logarithms in this type of field [2]. Since solving discrete logarithms in finite fields is of general interest, especially for finite fields with characteristic 2, there are a few implementations of these algorithms.

An important point is how the function field is constructed. Whereas Adleman and Huang [2] were looking for the most simple representation for an optimal performance of the necessary function field arithmetic, Joux and Lercier in 2001 [7] generalized this approach to get better asympotic running times. They proved their theoretical result by solving a discrete logarithm in the finite field of size 2^{521} in approximately one month on one machine. Moreover, they showed that the specialized algorithm of Coppersmith [3], which holds the actual discrete logarithm record (in a field of size 2^{607}), is a special case of their algorithm in the case of characteristic 2. But since the record computation, done by Thome in 2001 [10], was performed using massively parallel computations for collecting relations in the sieving part of the algorithm, there is still room for practical improvements of the computation of discrete logarithms by using Joux' and Lercier's ideas. Especially, there are no known practically relevant results for characteristics different from 2 as of this writing.

Sachar Paulus

References

[1] Adleman, L.M., J. De Marrais, and M.-D. Huang (1994). "A subexponential algorithm for discrete logarithms over the rational subgroup of the Jacobian of large genus hyperelliptic curves over finite fields." *ANTS-1: Algorithmic Number Theory*, Lecture Notes in Computer Science, vol. 877, eds. L.M. Adleman and M.-D. Huang. Springer-Verlag, Berlin, 28–40.

[2] Adleman, L.M. and M.-D. Huang (1999). "Function field sieve method for discrete logarithms over finite fields." *Information and Computation*, 151, 5–16.

[3] Coppersmith, Don (1984). "Fast evaluation of Logarithms in Fields of Characteristic Two, IEEE. Transaction on Information Theory, vol. IT-30 (4), 587–594.

[4] Flassenberg, R. and S. Paulus (1999). "Sieving in function fields." *Experimental Mathematics*, 8, 339–349.

[5] Galbraith, S.D., S. Paulus, and N.P. Smart (2001). "Arithmetic on superelliptic curves." *Mathematics of Computation*, 71, 393–405.

[6] Hafner, James L. and Kevin S. McCurley (1991). "Asymptotically fast triangularization of matrices over rings." *SIAM Journal of Computer*, 20 (6), 1068–1083.

[7] Joux, A. and R. Lercier (2002). "The function field sieve is quite special." *Proceedings of ANTS-V*, Lecture Notes in Computer Science, vol. 2369, eds. C. Fieker and D.R. Kohel. Springer-Verlag, Berlin, 431–445.

[8] Koblitz, N. (1989). "Hyperelliptic cryptosystems." *Journal of Cryptology*, 1 (3), 139–150.

[9] Smart, N.P. (1999). "On the performance of hyperelliptic cryptosystems." *Advances in Cryptology—EUROCRYPT'99*, Lecture Notes in Computer Science, vol. 1592. Springer-Verlag, Berlin, 165–175.

[10] Thome, E. (2001). "Computation of discrete logarithms in GF(2^{607})." *Advances in Cryptology—ASIACRYPT 2001*, Lecture Notes in Computer Science, vol. 2248, ed. C. Boyd. Springer-Verlag, Berlin, 107–124.

SIGNCRYPTION

INTRODUCTION: Encryption and digital signature schemes are fundamental cryptographic tools for providing privacy and authenticity, respectively, in the public-key setting. Traditionally, these two important building-blocks of public-key cryptography have been considered as *distinct* entities that may be *composed* in various ways to ensure *simultaneous* message privacy and authentication. However, in the last few years a new, separate primitive—called *signcryption* [14]—has emerged to model a process simultaneously achieving privacy and authenticity. This emergence was caused by many related reasons. The obvious one is the fact that given that *both* privacy and authenticity are simultaneously needed in so many applications, it makes a lot of sense to invest special effort into designing a tailored, more efficient solution than a mere composition of signature and encryption. Another reason is that viewing authenticated encryption as a separate *primitive* may conceptually simplify the design of complex protocols which require both privacy and authenticity, as signcryption could now be viewed as an "indivisible" atomic operation. Perhaps most importantly, it was noticed by [2,3] (following some previous work in the symmetric-key setting [4,10]) that proper modeling of signcryption is not so obvious. For example, a straightforward composition of

signature and encryption might not always work; at least, unless some special care is applied [2]. The main reason for such difficulties is the fact that signcryption is a complex *multi-user* primitive, which opens a possibility for some subtle attacks (discussed below), not present in the settings of stand-alone signature and encryption.

Defining Signcryption

Syntactically, a signcryption scheme consists of the three efficient algorithms (Gen, SC, DSC). The key generation algorithm $\mathsf{Gen}(1^\lambda)$ generates the key-pair (SDK_U,VEK_U) for user U, where λ is the security parameter, SDK_U is the signing/decryption key that is kept private, and VEK_U is the verification/encryption key that is made public. The randomized signcryption algorithm SC for user U implicitly takes as input the user's secret key SDK_U, and explicitly takes as input the message m and the identity of the recipient ID_R, in order to compute and output the *signcryptext* on Π. For simplicity, we consider this identity ID_R, to be a public key VEK_R of the recipient R, although ID's could generally include more convoluted information (as long as users can easily obtain VEK from ID). Thus, we write $\mathsf{SC}_{\mathsf{SDK}_U}(M, \mathsf{ID}_R)$ as $\mathsf{SC}_{\mathsf{SDK}_U}(m, \mathsf{VEK}_R)$, or simply $\mathsf{SC}_U(m, \mathsf{VEK}_R)$. Similarly, user U's deterministic designcryption algorithm DSC implicitly takes the user's private SDK_U and explicitly takes as input the signcryptext $\tilde{\Pi}$ and the senders' identity ID_S. Again, we assume $\mathsf{ID}_S = \mathsf{VEK}_R$ and write $\mathsf{DSC}_{\mathsf{SDK}_U}(\Pi, \mathsf{VEK}_S)$, or simply $\mathsf{DSC}_U(\Pi, \mathsf{VEK}_S)$. The algorithm outputs some message $\tilde{m}$, or $\perp$ if the signcryption does not verify or decrypt successfully. Correctness of property ensures that for any users S, R, and message m, we have $\mathsf{DSC}_R(\mathsf{SC}_S(m, \mathsf{VEK}_R), \mathsf{VEK}_S) = m$.

We also remark that it is often useful to add another optional parameter to both SC and DSC algorithms: a *label* L (also termed *associated data* [11]). This label can be viewed as a public identifier which is "inseparably bound" to the message m inside the signcryptext. Intuitively, designcrypting the signcryptext Π of m with the wrong label should be impossible, as well as changing Π into a valid signcryptext $\tilde{\Pi}$ of the same m under a different label.

Security of Signcryption

Security of signcryption consists of two distinct components: one ensuring privacy, and the other—authenticity. On a high level, privacy is defined somewhat analogously to the privacy of an ordinary encryption, while authenticity—to that of an ordinary digital signature. For example, one can talk about indistinguishability of signcryptexts under chosen ciphertext attack, or existential unforgeability of signcryptexts under chosen message attack, among others. For concreteness, we concentrate on the above two forms of security too, since they are the strongest.

However, several new issues come up due to the fact that signcryption/designcryption take as an extra argument the identity of the sender/recipient. Below, we semiformally introduce some of those issues (see [2] for in-depth technical discussion, as well as formal definitions of signcryption).

- *Simultaneous Attacks.* Since the user U utilizes its secret key SDK_U to both send and receive the data, it is reasonable to allow the adversary $\mathcal{A}$ oracle access to both the signcryption and the designcryption oracle for user U, irrespective of whether $\mathcal{A}$ is attacking privacy or authenticity of U.
- *Two- vs. Multi-user Setting*. In the simplistic two-user setting, where there are only two users S and R in the network, the explicit identities become redundant. This considerably simplifies the design of secure signcryption schemes (see below), while providing a very useful intermediate step towards general, multi-user constructions (which are often obtained by adding a simple twist to the basic two-user construction). Intuitively, the security in the two-user model already ensures that there are no weaknesses in the way the message is encapsulated inside the signcryptext, but does not ensure that the message is bound to the identities of the sender and/or recipient. In particular, it might still allow the adversary a large class of so called *identity fraud* attacks, where the adversary can "mess up" correct user identities without affecting the hidden message.
- *Public* NonRepudiation? In a regular digital signature scheme, anybody can verify the validity of the signature, and unforgeability of the signature ensures that a signer S indeed certified the message. Thus, we say that a signcryption scheme provides *nonrepudiation* if the recipient can extract a regular (publicly verifiable) digital signature from the corresponding signcryptext. In general, however, it is a-priori only clear that the *recipient* R is sure that S sent the message. Indeed, without R's secret key SDK_R others might not be able to verify the authenticity of the message, and it might not be possible for R to extract a regular signature of m. Thus, signcryption does not necessarily provide nonrepudiation. In fact, for some

applications we might explicitly want *not* to have nonrepudiation. For example, S might be willing to send some confidential information to R only under the condition that R cannot convince others of this fact. To summarize, nonrepudiation is an optional feature which some schemes support, others do not, and others explicitly avoid!

- *Insider vs. Outsider Security.* In fact, even with R's secret key SDK_R it might be unclear to an observer whether S indeed sent the message m to R, as opposed to R "making it up" with the help of SDK_R. This forms the main basis for distinction between *insider-* and *outsider-secure* signcryption. Intuitively, in an outsider-secure scheme the adversary must compromise communication between two honest users (whose keys he does not know). Insider-secure signcryption protects a given user U even if his partner might be malicious. For example, without U's key, one cannot forge signcryptext from U to any other user R, even with R's secret key. Similarly, if honest S sent $\Pi = \mathsf{SC}_S(m, \mathsf{VEK}_U)$ to U and later exposed his key SDK_S to the adversary, the latter still cannot decrypt Π. Clearly, insider-security is stronger than outsider-security, but might not be needed in a given application. In fact, for applications supporting message repudiation, one typically does not want to have insider-security.

Supporting Long Inputs

Sometimes, it is easier to design natural signcryption schemes supporting short inputs. Below we give a general method how to create signcryption SC' supporting arbitrarily long inputs from SC which only supports fixed-length (and much shorter) inputs. The method was suggested by [8] and uses a new primitive called *concealment*. A concealment is a publicly known randomized transformation, which, on input m, outputs a *hider* h and a *binder* b. Together, h and b allow one to recover m, but separately, (1) the hider h reveals "no information" about m, while (2) the binder b can be "meaningfully opened" by at most one hider h. Further, we require $|b| \ll |m|$ (otherwise, one could trivially set $b = m$, $h = \emptyset$). Now, we let $\mathsf{SC}'(m) = \langle \mathsf{SC}(b), h \rangle$ (and DSC' is similar). It was shown in [8] that the above method yields a secure signcryption SC'. Further, a simple construction of concealment was given: set $h = E_\tau(m), b = \langle \tau, H(h) \rangle$, where E is a symmetric-key one-time secure encryption (with short key τ) and H is a collision-resistant hash function (with short output).

CURRENT SIGNCRYPTION SCHEMES: We now survey several signcryption schemes achieving various levels of provable security.

Generic Composition Schemes

The two natural composition paradigms are "encrypt-then-sign" ($\mathcal{E}t\mathcal{S}$) and "sign-then-encrypt" ($\mathcal{S}t\mathcal{E}$). More specifically, assume Enc is a semantically secure encryption against chosen ciphertext attack, and Sig is an existentially unforgeable signature (with message recovery) against chosen message attack. Each user U has a key for for Sig and Enc. Then the "basic" $\mathcal{E}t\mathcal{S}$ from S to R outputs $\mathsf{Sig}_S(\mathsf{Enc}_R(m))$, while $\mathcal{S}t\mathcal{E}$—$\mathsf{Enc}_R(\mathsf{Sig}_S(m))$. Additionally, [2] introduced a novel generic composition paradigm for *parallel* signcryption. Namely, assume we have a secure commitment scheme, which on input m, outputs a commitment c and a decommitment d (where c is both hiding and binding). Then "commit-then-encrypt-and-sign" ($\mathcal{C}t\mathcal{E}\&\mathcal{S}$) outputs a pair $\langle \mathsf{Enc}_R(d), \mathsf{Sig}_S(c) \rangle$. Intuitively, the scheme is private as public c reveals no information about m (while d is encrypted), and authentic since c binds one to m. The advantage of the above scheme over the sequential $\mathcal{E}t\mathcal{S}$ and $\mathcal{S}t\mathcal{E}$ variants is the fact that expensive signature and encryption operations are performed in parallel. In fact, by using trapdoor commitments in place or regular commitments, most computation in $\mathcal{C}t\mathcal{E}\&\mathcal{S}$—including the expensive computation of both public-key signature and encryption—can be done off-line, even before the message m is known!

It was shown by [2] that all three basic composition paradigms yield an insider-secure signcryption in the two-user model. Moreover, $\mathcal{E}t\mathcal{S}$ is outsider-secure even if Enc is secure only against the chosen plaintext attack, and $\mathcal{S}t\mathcal{E}$ is outsider-secure even if Sig is only secure against no message attack. Clearly, all three paradigms are insecure in the multiuser model, since no effort is made to bind the message m to the identities of the sender/recipient. For example, intercepting a signcryptext of the form $\mathsf{Sig}_S(e)$ from S to R, an adversary $\mathcal{A}$ can produce $\mathsf{Sig}_{\mathcal{A}}(e)$, which is a valid signcryptext from $\mathcal{A}$ to R of the same message m, even though m is *unknown* to $\mathcal{A}$. [2] suggest a simple solution: when encrypting, always append the identity of the sender to the message, and when signing, of the recipient. For example, a multi-user secure variant of $\mathcal{E}t\mathcal{S}$ is $\mathsf{Sig}_S(\mathsf{Enc}_R(m, \mathsf{VEK}_S), \mathsf{VEK}_R)$. Notice, if Enc and/or Sig support labels, these identities can be part of the label rather than the message.

Finally, we remark that $\mathcal{S}t\mathcal{E}$ and $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ always support nonrepudiation, while $\mathcal{S}t\mathcal{E}$ might or might not.

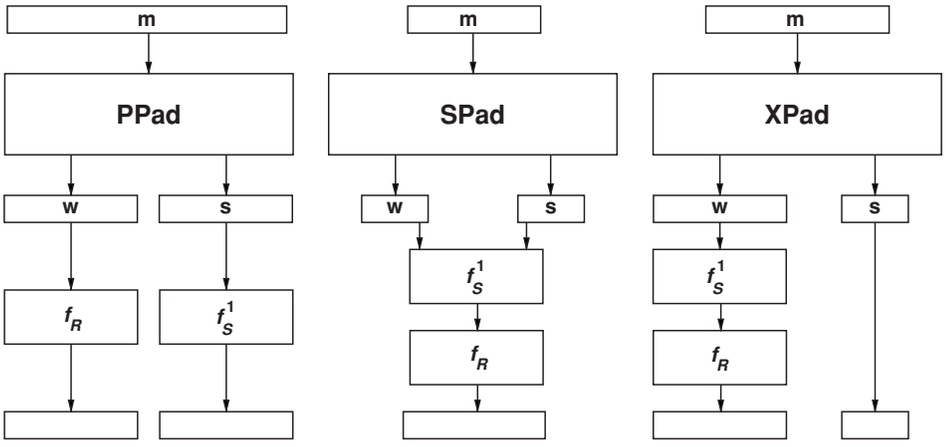


Fig. 1. Generalized paddings as used by signcryption

Schemes from Trapdoor Permutations

The generic schemes above validate the fact that signcryption can be built from ordinary signature and encryption, but will be inefficient unless the latter are efficiently implemented. In practice, efficient signature and encryption schemes, such as OAEP [5], OAEPP+ [13], PSS-R [6], are built from trapdoor permutations, such as RSA, and are analyzed in the random oracle model. Even with these efficient implementations, however, the generic schemes will have several drawbacks. For example, users have to store two independent keys, the message bandwidth is suboptimal and the "exact security" of the scheme is not as good as one might expect. Thus, given that practical schemes are anyway built from trapdoor permutations, it is natural to have highly optimized *direct* signcryption constructions from trapdoor permutations (in the random oracle model).

This is the approach of [9]. In their model, each user U independently picks a trapdoor permutation f_U (together with its trapdoor, denoted f_U^{-1}) and publishes f_U as its public key (see also trapdoor one-way function and substitutions and permutations). (Notice, only a *single* key is chosen, unlike what is needed for the generic schemes.) Then, [9] considers the following three paradigms termed **P**-Pad, **S**-Pad and **P**-Pad. Each paradigm proceeds by constructing a padding scheme produces $\pi(m) = w|s$, and then composing it with the corresponding permutations of the sender and the recipient as shown in Figure 1. Table 1 also shows how the corresponding approaches could be used for plain signature and encryption as well.

The convenience of each padding scheme depends on the application for which it is used. As was shown in [9], **P**-Pad signcryption provides parallel application of "signing" f_S^{-1} and "encrypting" f_R, which can result in efficiency improvements on parallel machines. However, the minimum ciphertext length is twice as large as compared to **S**-Pad, yet the exact security offered by **S**-Pad is not as tight as that of **P**-Pad. Finally, **X**-Pad regains the optimal exact security of **P**-Pad, while maintaining ciphertext length nearly equal to the length of the trapdoor permutation (by achieving quite short s).

It remains to describe secure padding schemes π for **P**-Pad, **S**-Pad and **X**-Pad. All constructions offered by [9] are quite similar. One starts with any *extractable commitment* (c, d), where c is the commitment and d is the decommitment. Such schemes are very easy to construct in the random oracle model. For example, if $|m| = n$, for any $0 \le a \le n$, the following scheme is an extractable commitment: split $m = m_1|m_2$, where $|m_1| = a$, $|m_2| = n - a$, and set

$$c = G(r) \oplus m_1 | H(m_2|r)$$
$$d = m_2|r$$

where G and H are random oracles (with appropriate input/output lengths) and r is a random salt.

Table 1. Signcryption Schemes Based on Trapdoor Permutations.

Padding Type	Encryption	Signature	Signcryption
P-Pad (**P**arallel Padding)	$f_R(w)\|s$	$w\|f_S^{-1}(s)$	$f_R(w)\|f_S^{-1}(s)$
S-Pad (**S**equential Padding)	$f_R(w\|s)$	$f_S^{-1}(w\|s)$	$f_R(f_S^{-1}(w\|s))$
X-Pad (e**X**tended sequential Padding)	$f_R(w)\|s$	$f_S^{-1}(w)\|s$	$f_R(f_S^{-1}(w))\|s$

To get a secure padding scheme for the **P**-Pad paradigm, one should then apply the Feistel Transform to the resulting pair (d, c), with yet another random oracle F as the round function. Namely, set $w = c$, $s = F(c) \oplus d$. For example, using the extractable commitment above with $a = n$, we get nothing else but the OAEP padding, while $a = 0$ would give the PSSR padding! For arbitrary a, [9] call the resulting hybrid between PSSR and OAEP *Probabilistic Signature-Encryption Padding* (PSEP).

To get the padding π sufficient for either **S**-Pad or **P**-Pad, one only needs to perform one more Feistel round to the construction above: $w' = s$, $s' = F'(s) \oplus w$, and set $\pi(m) = w'|s'$. Coincidentally, the resulting π also gives a very general construction of the so called *universal padding schemes* [7].

As described, the paddings π_1 and π_3 above would only give insider security in the two-user setting. To get multi-user security, all one needs to do is to prepend the pair $(\mathsf{VEK}_S, \mathsf{VEK}_R)$ to all the inputs to the random oracles F and F': namely, create effectively independent F and F' for every sender-recipient pairing! More generally, the paddings above also provide label support, if one sticks the label L as part of the inputs to F and F'.

Finally, we remark that **P**-Pad, **X**-Pad and **X**-Pad always support non-repudiation.

Schemes Based on Gap Diffie–Hellman

Finally, we present two very specific, but efficient schemes based on the so called *Gap Diffie–Hellman* assumption. Given a cyclic group G of prime order q, and a generator g of G, the assumption states that the computational Diffie–Hellman problem (CDH) is computationally hard, even if one is given oracle access to the decisional Diffie–Hellman (DDH) oracle. Specifically, it is hard to compute g^{ab} from g^a and g^b, even if one can test whether a tuple $\langle g^x, g^y, g^z \rangle$ satisfies $z = xy \bmod q$.

In both schemes, the user U chooses a random $x_U \in \mathbb{Z}_q$ as its secret key VEK_U, and sets its public key $\mathsf{SDK}_U = y_U = g^{x_U}$. The scheme of [1] is based on the following noninteractive key agreement between users S and R. Namely, both S and R can compute the quantity $Q_{SR} = g^{x_R x_S} = y_S^{x_R} = y_R^{x_S}$. They then set the key $K_{SR} = H(Q_{SR})$, where H is a random oracle, and then always use K_{SR} to perform symmetric-key authenticated encryption of the message m. For the latter, they can use any secure symmetric-key scheme, like "encrypt-then-mac" [4] or OCB [12]. The resulting signcryption scheme can be shown to be outsider-secure for both privacy and authenticity, in the multi-user setting. Clearly, it is not insider-secure, since both S and R know the key K_{SR}. In fact, the scheme is perfectly repudiable, since all the signcryptexts from S could have been easily faked by R.

To get insider-security for authenticity under the same assumption, one can instead consider the following scheme, originally due to [14], but formally analyzed by [3]. Below G and H are random oracles with appropriate domains, and E is a one-time secure symmetric-key encryption (e.g., one-time pad will do). To signcrypt a message from S to R, S chooses a random $x \in \mathbb{Z}_q$, computes $Q = y_R^x$, makes a symmetric key $K = H(Q)$, sets $c \leftarrow E_K(m)$, computes the "validation tag" $r = G(m, y_A, y_B, Q)$ and finally $t = x(r + x_S)^{-1} \bmod q$. Then S outputs $\langle c, r, t \rangle$ as the signcryption of m. To designcrypt $\langle c, r, t \rangle$, R first recovers g^x via $w = (y_S g^r)^t$, then recovers the Diffie–Hellman key $Q = w^{x_R}$, the encryption key $K = H(Q)$ and the message $m = D_K(c)$. Before outputting m, however, it double checks if $r = G(m, y_A, y_B, Q)$. While this scheme is insider-secure for authenticity, it is still not insider-secure.

We also mention that the scheme supports public nonrepudiation. All that R has to do is to reveal Q, m and a proof that $Q = w^{x_R}$ (which can be done noninteractively using the Fiat-Shamir heuristics, applied to the three-move proof that $\langle g, y_R, w, Q \rangle$ form a DDH-tuple).

Yevgeniy Dodis

References

[1] An, Jee Hea (2001). "Authenticated encryption in the public-key setting: Security notions and analyses." *Cryptology ePrint Archive*, Report 2001/079.

[2] An, Jee Hea, Yevgeniy Dodis, and Tal Rabin (2002). "On the security of joint signature and encrytion." *Advances in Cryptology—EUROCRYPT 2002, April 28–May 2*, Lecture Notes in Computer Science, vol. 2332, ed. L. Knudsen. Springer-Verlag, Berlin. Available from http://eprint.iacr.org/2002/046/

[3] Baek, Joonsang, Ron Steinfeld, and Yuliang Zheng (2002). "Formal proofs for the security of signcryption." *5th International Workshop on Practice and Theory in Public Key Cryptosystems—PKC 2002, February*, Lecture Notes in Computer Science, vol. 2274, eds. D. Naccache and P. Paillier. Springer-Verlag, Berlin, 80–98.

[4] Bellare, Mihir and Chanathip Namprempre (2000). "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm." *Advances in Cryptology—ASIACRYPT 2000, Kyoto, Japan, December 3–7*, Lecture Notes in Computer Science, vol. 1976, ed. T. Okamoto. Springer-Verlag, Berlin, 531–545.

[5] Bellare, Mihir and Phillip Rogaway (1995). "Optimal asymmetric encryption." *Advances in Cryptology—EUROCRYPT'94, May 9–12*, Lecture

Notes in Computer Science, vol. 950, ed. A. De Santis. Springer-Verlag, Berlin, 92–111. Revised version available from http://www-cse.ucsd.edu/users/mihir/

[6] Bellare, Mihir and Phillip Rogaway (1996). "The exact security of digital signatures: How to sign with RSA and Rabin." *Advances in Cryptology—EUROCRYPT'96, May 12–16*, Lecture Notes in Computer Science, vol. 1070, ed. U. Maurer. Springer-Verlag, Berlin, 399–416. Revised version appears in http://www-cse.ucsd.edu/users/mihir/papers/crypto-papers.html

[7] Coron, Jean-Sébastian, Marc Joye, David Naccache, and Pascal Pailler (2002). "Universal padding schemes for RSA." *Advances in Cryptology—CRYPTO 2002, August 18–22*, Lecture Notes in Computer Science, vol. 2442, ed. M. Yung. Springer-Verlag, Berlin. Available from http://eprint.iacr.org/2002/115/

[8] Dodis, Yevgeniy and Jee Hea An (2003). "Concealment and its applications to authenticated encryption." *Advances in Cryptology—EUROCRYPT 2003, May 4–8*, Lecture Notes in Computer Science, vol. 2656, ed. E. Biham. Springer-Verlag, Berlin, 312–329.

[9] Dodis, Yevgeniy, Michael J. Freedman, Stanislaw Jarecki, and Shabsi Walfish (2004). Versatile Padding Schemes for Joint Signature and Encryption. In Birgit Pfitzmann, editor, Eleventh ACM Conference on Computer and Communication Security pages 196–205. ACM, October 25–29, 2004.

[10] Krawczyk, Hugo (2001). "The order of encryption and authentication for protecting communications (or: How secure is ssl?)." *Advances in Cryptology—CRYPTO 2001, August 19–23*, Lecture Notes in Computer Science, vol. 2139, ed. J. Kilian. Springer-Verlag, Berlin, 310–331.

[11] Rogaway, Phillip (2002). "Authenticated-encryption with associated-data." *Ninth ACM Conference on Computer and Communication Security, November 17–21*, ed. Ravi Sandhu. ACM Press, New York, 98–107.

[12] Rogaway, Phillip, Mihir Bellare, John Black, and Ted Krovetz (2001). "OCB: A block-cipher mode of operation for efficient authenticated encryption." *Eighth ACM Conference on Computer and Communication Security, November 5–8*, ed. Pierangela Samarati. ACM Press, New York, 196–205. Full version available from http://www.cs.ucsdavis.edu/~rogaway

[13] Shoup, Victor (2001). "OAEP reconsidered." *Advances in Cryptology—CRYPTO 2001, August 19–23*, Lecture Notes in Computer Science, vol. 2139, ed. J. Kilian. Springer-Verlag, Berlin, 240–259.

[14] Zheng, Y. (1997). "Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption)." *Advances in Cryptology—CRYPTO'97, August 17–21*, Lecture Notes in Computer Science, vol. 1294, ed. B.S. Kaliski Jr. Springer-Verlag, Berlin, 65–179.

SIGNED DIGIT EXPONENTIATION

Signed digit exponentiation is an approach for computing powers in any group in which the inverse A^{-1} of any group element A can be computed quickly (such as the groups of points on an elliptic curve employed in elliptic curve cryptography). It is related to sliding window exponentiation: while in sliding window exponentiation each window corresponds to a positive digit value, signed digit exponentiation additionally makes use of the corresponding negative digit values, and the ease of inversion makes these extra digits available almost for free. This often makes signed digit exponentation faster when using the same amount of memory for storing group elements, and allows it to reach approximately the same speed with less memory.

Let $B_k = \{\pm 1, \pm 3, \ldots, \pm(2^k - 1)\}$ where k is a positive integer; and let a base-two representation of an exponent e be given using the digit set $\{0\} \cup B_k$, i.e.

$$e = \sum_{i=0}^{l-1} e_i 2^i, \quad e_i \in \{0\} \cup B_k.$$

Assuming that l is chosen such that $e_{l-1} \neq 0$, the *left-to-right signed digit exponentiation method* computes g^e as follows where g is any group element; cf. the left-to-right sliding window exponentiation method.

$G_1 \leftarrow g$
$A \leftarrow g \circ g$
for $d = 3$ to $2^k - 1$ step 2 **do**
  $G_d \leftarrow G_{d-2} \circ A$

if $e_{l-1} > 0$ **then**
  $A \leftarrow G_{e_{l-1}}$
else
  $A \leftarrow G_{-e_{l-1}}^{-1}$
for $i = l - 2$ down to 0 **do**
  $A \leftarrow A \circ A$
  if $e_i \neq 0$ then
    if $e_i > 0$ then
      $A \leftarrow A \circ G_{e_i}$
    else
      $A \leftarrow A \circ G_{-e_i}^{-1}$
return A

The *right-to-left signed digit exponentiation method* computes g^e as follows; cf. the right-to-left

sliding window exponentiation method. Note that the algorithm as written can be optimized similarly to the right-to-left 2^k-ary exponentiation or sliding window exponentiation methods to avoid (at least) 2^{k-1} applications of the group operation.

for $d = 1$ to $2^k - 1$ step 2 **do**
  $B_d \leftarrow$ identity element
$A \leftarrow g$

for $i = 0$ to $l - 1$ **do**
  if $e_i \neq 0$ **then**
    if $e_i > 0$ **then**
      $B_{e_i} \leftarrow B_{e_i} \circ A$
    else
      $B_{-e_i} \leftarrow B_{-e_i} \circ A^{-1}$
  if $i < l - 1$ **then**
    $A \leftarrow A \circ A$

{Now $g^e = \Pi_{d \in \{1,3,\ldots,2^k-1\}} B_d^d$.}

for $d = 2^k - 1$ to 3 step -2 **do**
  $B_{d-2} \leftarrow B_{d-2} \circ B_d$
  $B_1 \leftarrow B_1 \circ (B_d \circ B_d)$
return B_1

For both the left-to-right and the right-to-left variant, it remains to be considered how signed digit representations of exponents e using the digit set $\{0\} \cup B_k$ with $B_k = \{\pm 1, \pm 3, \ldots, \pm(2^k - 1)\}$ can be obtained. An algorithm for the simplest case $k = 1$ is due to Reitwiesner [1]; the representation obtained by it (using digits $\{-1, 0, 1\}$) is known as the *nonadjacent form (NAF)* of e. The generalization for an arbitrary parameter k was simultaneously suggested by multiple researchers; the following algorithm is from [2]:

$c \leftarrow e$
$i \leftarrow 0$
while $c > 0$ **do**
  if c is odd **then**
    $d \leftarrow c \bmod 2^{k+1}$
    if $d > 2^k$ **then**
      $d \leftarrow d - 2^{k+1}$
    $c \leftarrow c - d$
  else
    $d \leftarrow 0$
  $e_i \leftarrow d; i \leftarrow i + 1$
  $c \leftarrow c/2$
return $e_{i-1}, \ldots, e_0$

This algorithm is a variant of right-to-left scanning as used in sliding window exponentiation with an effective window size of $k + 1$. For efficiency considerations, if the cost of inverting groups elements and the additional cost for obtaining the appropriate representation of e can be neglected, signed digit exponentiation differs from sliding window exponentiation with the same parameter k in that the expected number of nonzero digits in the representation is approximately $l/(k + 2)$ instead of approximately $l/(k + 1)$ (but the maximum possible length of the signed digit representation is longer: while l cannot exceed the length of the binary representation of e for sliding window exponentiation, it can be said length plus 1 for signed digit exponentiation).

Bodo Möller

References

[1] Reitwiesner, G.W. (1960). "Binary arithmetic." *Advances in Computers*, 1, 231–308.
[2] Solinas, J.A. (2000). "Efficient arithmetic on Koblitz curves." *Designs, Codes and Cryptography*, 19, 195–249.

SIMULTANEOUS EXPONENTIATION

Various schemes for public-key cryptography involve computing power products in some commutative group (or commutative semigroup). A straightforward way to compute a power product

$$\prod_{j=1}^{n} g_j^{e_j}$$

is to compute the individual powers $g_j^{e_j}$ using binary exponentiation or some other exponentiation method, and perform $n - 1$ applications of the group operation to multiply these partial results. However, specialized algorithms for computing power products are often faster. The task of computing a power product is sometimes called *multi-exponentiation*, and performing a multiexponentiation by a procedure that does not involve computing the partial results $g_j^{e_j}$ is known as *simultaneous exponentiation*. Two methds for multiexponentiation that both generalize left-to-right sliding window exponentiation are *simultaneous sliding window exponentiation*, which is due to Yen, Laih. and Lenstra [3] (based on the simultaneous 2^k-ary exponentiation method from Straus [2]), and *interleaved sliding window*

exponentiation [1]. Like the sliding window method for single exponentiations, these methods use the binary representation of exponents, on which nonoverlapping windows are placed such that every nonzero bit is covered by one of the windows. Simultaneous sliding window exponentiation and interleaved sliding window exponentiation use different approaches for placing windows; sometimes the former is faster, sometimes the latter.

Simultaneous sliding window exponentiation uses windows up to some maximum width k that span across all n exponents; e.g., for exponents e_1, e_2, e_3 with binary representations 1011010, 0011001, and 1001011 and $k = 2$:

e_1	[1	0	1	1]	0	[1	0]
e_2	[0	0	1	1]	0	[0	1]
e_3	[1	0	0	1]	0	[1	1]

Such windows can be found by left-to-right scanning: look at the binary representations of the exponents simultaneously, going from left to right, starting a new window whenever a nonzero bit is encountered, choosing the maximum width up to k for this particular window such that one of the rightmost bits is also nonzero. The result of collapsing the window values into the right-most row of each window can be considered a base-two representation

$$(e_1, \ldots, e_n) = \sum_{i=0}^{l-1} (e_{1,i}, \ldots, e_{n,i}) 2^i$$

of the vector of exponents, e.g.

e_1	1 0 0 3 0 0 2
e_2	0 0 0 3 0 0 1
e_3	1 0 0 1 0 0 3

for the above example. Assume we have such a representation with l chosen minimal, i.e. $(e_{1,l}, \ldots, e_{n,l}) \neq (0, \ldots, 0)$. To perform a simultaneous sliding window exponentiation, first products

$$G_{(d_1,\ldots,d_n)} = \prod_{j=1}^{n} g_j^{d_j}$$

of small powers are computed and stored for all possible window values, namely for the tuples $(d_1, \ldots, d_n)$ with $d_j \in \{0, 1, \ldots, 2^k - 1\}$ for $j = 1, \ldots, n$ such that at least one of the d_j is odd. There are $2^{nk} - 2^{n(k-1)}$ such tuples, and computing the table of those products can be done with

$$2^{nk} - 2^{n(k-1)}$$

applications of the group operation, n of which are squarings ($g_1, \ldots, g_n$ appear in the table and are available without any computation; once $g_j \circ g_j$ for $j = 1, \ldots, n$ have been computed as temporary values, each of the $2^{nk} - 2^{n(k-1)} - n$ remaining table values can be obtained by using the group operation once). The multi-exponentiation result then is computed using the table of small powers:

$A \leftarrow G_{(e_{1,l-1},\ldots,e_{n,l-1})}$
for $i = l - 2$ down to 0 **do**
  $A \leftarrow A \circ A$
  if $(e_{1,l}, \ldots, e_{n,l}) \neq (0, \ldots, 0)$ **then**
    $A \leftarrow A \circ G_{(e_{1,i},\ldots,e_{n,i})}$
return A

For random b-bit exponents, this requires at most another $b - 1$ squaring operations and on average approximately another

$$b \cdot \frac{1}{k + \frac{1}{2^n - 1}}$$

general group operations. Note that in practice it is not necessary to completely derive the representation

$$(e_{1,l-1}, \ldots, e_{n,l-1}), \ldots, (e_{1,0}, \ldots, e_{n,0})$$

before starting the exponentiation; instead, left-to-right scanning can be used to determine it window by window when it is needed.

In *interleaved sliding window exponentiation*, each single exponent has independent windows up to some maximum width k; e.g., for exponents e_1, e_2, e_3 with binary representations 1011010, 0011001, and 1001011 and $k = 3$:

e_1	[1	0	1]	[1	0	1]	0
e_2	0	0	[1	1]	0	0	[1]
e_3	[1]	0	0	[1	0	1]	[1]

For each exponent, such windows can be found by left-to-right scanning: look at the binary representation of the respective exponent, going from left to right, starting a new window whenever a nonzero bit is encountered, choosing the maximum width up to k for this particular window such that the rightmost bit is also nonzero. To perform an interleaved sliding window exponentiation, first for each g_j, the powers for odd exponents 1 up to $2^k - 1$ are computed and stored:

for $j = 1$ to n **do**
  $G_{j,1} \leftarrow g$
  $A \leftarrow g \circ g$
  for $d = 3$ to $2^k - 1$ step 2 **do**
    $G_{j,d} \leftarrow G_{j,d-2} \circ A$

Rule A	Rule B
$w_1^{k+1} = G^k(w_1^k) \oplus w_4^k \oplus counter^k$	$w_1^{k+1} = w_4^k$
$w_2^{k+1} = G^k(w_1^k)$	$w_2^{k+1} = G^k(w_1^k)$
$w_3^{k+1} = w_2^k$	$w_3^{k+1} = w_1^k \oplus w_2^k \oplus counter^k$
$w_4^{k+1} = w_3^k$	$w_4^{k+1} = w_3^k$

Fig. 1. Rule A and Rule B

Then the multi-exponentiation result is computed using that table of powers. The following algorithm shows how this computation can be implemented including left-to-right scanning of exponents up to b bits. The algorithm accesses the bits $e_j[i]$ of the binary representations

$$e_j = \sum_{i=0}^{b-1} e_j[i]2^i, \quad e_j[i] \in \{0, 1\}$$

of the exponents; the notation $e_j[i \ldots h]$ is shorthand for $\Sigma_{v=h}^{i} e_j[v]2^{v-h}$.

$A \leftarrow$ identity element
for $j = 1$ to n **do**
  $window\_position_j \leftarrow -1$
for $i = b - 1$ down to 0 **do**
  $A \leftarrow A \circ A$
  for $j = 1$ to n **do**
    if $window\_position_j = -1$
    and $e_j[i] = 1$ **then**
      $h \leftarrow i - k + 1$
      if $h < 0$ **then**
        $h \leftarrow 0$
      while $e_j[h] = 0$ **do**
        $h \leftarrow h + 1$
      $window\_position_j \leftarrow h$
      $E_j \leftarrow e_j[i \ldots h]$
    if $window\_position_j = i$ **then**
      $A \leftarrow A \circ G_{j,E_j}$
      $window\_position_i \leftarrow -1$
return A

The algorithm as written can be improved by a simple optimization: while A still has its initial value, omit the statement $A \leftarrow A \circ A$, and use a direct assignment $A \leftarrow G_{j,E_j}$ instead of the first assignment $A \leftarrow A \circ G_{j,E_j}$. With this optimization, an interleaved sliding window exponentiation takes up to $n + b - 1$ squarings and on average about

$$n \cdot \left(2^{k-1} - 1 + \frac{b-1}{k+1}\right)$$

general group operations.

Interleaved sliding window exponentiation essentially interleaves the operations of n single exponentiations using left-to-right sliding window exponentiation, saving many of the squarings. In groups where computing inverses of elements is possible very quickly, it is possible to similarly interleave the operations of n single exponentiations using left-to-right signed digit exponentiation for faster multi-exponentiation.

Bodo Möller

References

[1] Moller, B. (2001). "Algorithms for multi-exponentiation." *Selected Area in Cryptography—SAC 2001*, Lecture Notes in Computer Science, vol. 2259, eds. S. Vaudenay and A.M. Youssef. Springer-Verlag, Berlin, 165–180.
[2] Straus, E.G. (1964). "Problems and solutions: Addition chains of vectors." *American Mathematical Monthly*, 71, 806–808.
[3] Yen, S.-M., C.-S. Laih, and A.K. Lenstra (1994). "Multi-exponentiation." *IEE Proceedings—Computers and Digital Techiques*, 141, 325–326.

SKIPJACK

Skipjack [6] is the secret key encryption algorithm (see symmetric cryptosystem) developed by the NSA for the Clipper chip initiative (including the Capstone chip and the Fortezza PC card). It was implemented in tamper-resistant hardware and its structure was kept secret since its introduction in 1993.

On June 24, 1998, Skipjack was declassified, and its description was made public on the web site of NIST [6]. It is an *iterative* block cipher with 64-bit block, 80-bit key and 32 rounds. It has two types of rounds, called Rule A and Rule B. Each round is described in the form of a linear feedback shift register with an additional nonlinear keyed G permutation. Rule B is basically the inverse of Rule A with minor positioning differences. Skipjack applies eight rounds of Rule A, followed by eight rounds of Rule B, followed by another eight rounds of Rule A, followed by another eight rounds of Rule B. The original definitions of Rule A and

Rule B are given in Figure 1, where *counter* is the round number (in the range 1 to 32), G is a four-round Feistel permutation whose F function is defined as an 8×8-bit S box, called *F Table*, and each round of G is keyed by eight bits of the key.

The key schedule of Skipjack takes a 10-byte key, and uses four of them at a time to key each G permutation. The first four bytes are used to key the first G permutation, and each additional G permutation is keyed by the next four bytes cyclically, with a cycle of five rounds.

Skipjack has been subject to intensive analysis [2–5]. For example, Skipjack reduced to (the first) 16 rounds can be attacked with 2^{17} chosen plaintexts and 2^{34} time of analysis [5], which may be reduced to 2^{14} texts and 2^{16} steps using the *yoyo-game* approach [1]. Attacking the middle 16 rounds of Skipjack requires only 3 chosen plaintexts and 2^{30} time of analysis. The currently most successfull attack against the cipher is the imposible differential attack which breaks 31 rounds out of 32, marginally faster than exhaustive search.

In addition, it is worth noting that Skipjack can be attacked by a generic time-memory tradeoff approach requiring 2^{80} steps of precomputation and 2^{54} 80-bit words (i.e., 2^{60} bits) of memory, but then each search for a key requires only 2^{54} steps of computation.

Alex Biryukov

References

[1] Biham, E., A. Biryukov, O. Dunkelman, E. Richardson, and A. Shamir (1999). "Initial observations on skipjack: Cryptanalysis of Skipjack-3XOR." *Selected Areas in Cryptography, SAC'98*, Lecture Notes in Computer Science, vol. 1556, eds. S.E. Tavares and H. Meijer. Springer-Verlag, Berlin, 362–376.

[2] Biham, E., A. Biryukov, and A. Shamir (1999). "Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials." *Advances in Cryptology—EUROCRYPT'99*, Lecture Notes in Computer Science, vol. 1592, ed. J. Stern. Springer-Verlag, Berlin, 12–23.

[3] Granboulan, L. (2001). "Flaws in differential cryptanalysis of Skipjack." *Proceedings of Fast Software Encryption—FSE 2001*, Lecture Notes in Computer Science, vol. 2335, ed. M. Matsui. Springer-Verlag, Berlin, 328–335.

[4] Hwang, K., W. Lee, S. Lee, S. Lee, and J. Kim (2002). "Saturation attacks on round Skipjack." *Fast Software Encryption—FSE 2002*, Lecture Notes in Computer Science, vol. 2365, eds. J. Daemen and V. Rijmen. Springer-Verlag, Berlin, 100–111.

[5] Knudsen, L.R., M. Robshaw, and D. Wagner (1999). "Truncated differentials and Skipjack." *Advances in Cryptology—CRYPTO'99*, Lecture Notes in Computer Science, vol. 1666, ed. M. Wiener. Springer-Verlag, Berlin, 165–180.

[6] NIST (1998). "SKIPJACK and KEA algorithm specification." Technical Report, http://csrc.nist.gov/CryptoToolkit/skipjack/skipjack-kea.htm. Version 2.0.

SLIDE ATTACK

Slide attack is generic attack designed by Biryukov and Wagner [1, 2]. It can be applied in both known plaintext or chosen plaintext scenarios. It can be viewed as a variant of a related key attack, in which a relation of the key with itself is exploited. The main feature of this attack is that it realizes a dream of cryptanalysts: if the cipher is vulnerable to such an attack, the complexity of the attack is independent of the number of rounds of the ciphel. A typical slide of one encryption against another by one round (under the same key) is shown in Figure 1. If the equation $F_1(P_0, K_1) = P_1$ holds, the pair is called a *slid pair*. The attacker would then obtain two equations:

$$F_1(P_0, K_1) = P_1, \qquad F_r(C_0, K_r) = C_1,$$

where the second equation would hold for free due to sliding. These equations involve only a single round function, and thus could be solved by the attacker for the secret subkeys K_1, K_r of these rounds. The attacker may create properly *slid pairs* (P_0, P_1) by birthday paradox or by careful construction. For an arbitrary cipher the attack has complexity of $2^{n/2}$ known-plaintexts, where n is the blocksize. For a Feistel cipher complexity is reduced to $2^{n/4}$ chosen plaintexts.

Several ciphers or slight modifications of existing ciphers have been shown vulnerable to such attacks: for example the Brown-Seberry variant of the Data Encryption Standard (DES) [3] (rotations in key-schedule are by seven positions, instead of varying 1, 2 rotations as in the original DES), DES-X, the *Even-Mansour scheme* [4], arbitrary Feistel ciphers with 4-round periodic key-schedule as well as round-reduced versions of GOST. The basic attack has been extended into a *slide-with a twist*, a technique where encryption is slid against decryption and *complementary slide*

$$\begin{aligned} P_0 &\to F_1\ F_2\ F_3\ \ldots\ F_r \to C_1 \\ P_1 &\to F_1\ F_2\ F_3\ \ldots\ F_r \to C_2. \end{aligned}$$

Fig. 1. A typical slide attack

technique [2], where the inputs to the rounds do not have to be identical but may have the difference which is canceled out by a difference in the keys. In the same paper another generalization of the technique for the case of a composition of strong round functions is given.

It is clear that slide-attack would apply to any *iterative* construction which has enough *self-similarity* in its rounds. It could be applied to block-ciphers as described above, to stream-ciphers (see for example resynchronization attack on *WAKE-ROFB* [1]) or to MAC and hash-functions (see for example a recent *slid pair* discovery for SHA-1 by Saarinen [5].

In practice the attack seems easy to avoid by breaking the similarity of the round transforms by applying round counters (as is done for example in Skipjack) or different random constants in each round (as in Rijndael/AES, SHA-256 and many other constructions). Whether such simple changes are indeed sufficient is a matter of further research.

Alex Biryukov

References

[1] Biryukov, A. and D. Wagner (1999). "Slide attacks." *Proceedings of Fast Software Encryption—FSE'99*, Lecture Notes in Computer Science, vol. 1636, ed. L.R. Knudsen. Springer-Verlag, Berlin, 245–259.

[2] Biryukov, A. and D. Wagner (2000). "Advanced slide attacks." *Advances in Cryptology—EUROCRYPT 2000*, Lecture Notes in Computer Science, vol. 1807, ed. B. Preneel. Springer-Verlag, Berlin, 589–606.

[3] Brown, L. and J. Seberry (1990). "Key scheduling in DES type cryptosystems." *Advances in Cryptology—AUSCRYPT'90*, Lecture Notes in Computer Science, vol. 453, eds. J. Seberry and J. Pieprzyk. Springer-Verlag, Berlin, 221–228.

[4] Even, S. and Y. Mansour (1997). "A construction of a cipher from a single pseudorandom permutation." *Journal of Cryptology*, 10 (3), 151–162.

[5] Saarinen, M.-J.O. (2003). "Cryptyanalysis of block ciphers based on SHA-1 and MD5." *Proceedings of Fast Software Encryption—FSE 2003*, Lecture Notes in Computer Science, vol. 2887, ed. T. Johansson. Springer-Verlag, Berlin, 36–44.

SLIDING WINDOW EXPONENTIATION

Sliding window exponentiation is an approach for computing powers in any group (or semigroup). Like 2^k-ary exponentiation, it generalizes binary exponentiation and is parameterized by a positive integer k, where the case $k = 1$ is the same as binary exponentiation. Sliding window exponentiation can be considered an improved variant of 2^k-ary exponentiation: with identical $k \geq 2$, sliding window exponentiation needs storage for fewer group elements and usually performs less applications of the group operation than 2^k-ary exponentiation. However, the algorithms for sliding window exponentiation are slightly more complicated. 2^k-ary exponentiation uses the 2^k-ary representation of exponents, which can be considered as looking at the binary representation through fixed windows of width k:

$$\boxed{0\ 0\ 1}\boxed{1\ 1\ 0}\boxed{1\ 0\ 0}\boxed{0\ 1\ 1}\boxed{0\ 0\ 1}\boxed{0\ 1\ 0}$$

The sliding window method is based on the observation that fewer windows of width up to k can suffice to cover all nonzero exponent bits if one allows the windows to take arbitrary positions. Also, one can arrange for all windows to be odd-valued (i.e., have a 1 as the rightmost bit). Then the bits covered by each single window correspond to a value in the set $B_k = \{1, 3, \ldots, 2^k - 1\}$, and the number of possible window values is less than with the 2^k-ary exponentiation method. Covering the binary representation of the exponent by such windows yields a base-two representation of the exponent that uses the digit set $\{0\} \cup B_k$. One possible way to determine windows for a given exponent is to look at the binary representation of the exponent from left to right, starting a new window whenever a nonzero bit is encountered, choosing the maximum width up to k for this particular window such that the rightmost bit is also nonzero:

$$\begin{array}{rl} & 0\ 0\ \boxed{1\ 1\ 1}\ 0\ \boxed{1}\ 0\ 0\ 0\ \boxed{1\ 1}\ 0\ 0\ \boxed{1\ 0\ 1}\ 0 \\ \Rightarrow & \qquad\quad 7\ 0\ 1\ 0\ 0\ 0\ 0\ 3\ 0\ 0\ 0\ 0\ 5\ 0 \end{array}$$

Another possibility is to look at the binary representation of the exponent from right to left, starting a new width-k window whenever a nonzero bit is encountered:

$$\begin{array}{rl} & 0\ \boxed{0\ 1\ 1}\boxed{1\ 0\ 1}\ 0\ 0\ \boxed{0\ 1\ 1}\ 0\ 0\ \boxed{1\ 0\ 1}\ 0 \\ \Rightarrow & \qquad\quad 3\ 0\ 0\ 5\ 0\ 0\ 0\ 0\ 3\ 0\ 0\ 0\ 0\ 5\ 0 \end{array}$$

Such left-to-right scanning or right-to-left scanning yields a representation

$$e = \sum_{i=0}^{l-1} e_i 2^i, \quad e_i \in \{0\} \cup B_k.$$

In the following we assume that we have such a representation of some positive integer e with l chosen minimal; thus, $e_{l-1} \neq 0$.

The *left-to-right sliding window exponentiation method* computes g^e, where g is an element of the group (or semigroup), as follows. First the powers for odd exponents 1 up to $2^k - 1$ are computed and stored:

$G_1 \leftarrow g$
$A \leftarrow g \circ g$
for $d = 3$ to $2^k - 1$ step 2 **do**
  $G_d \leftarrow G_{d-2} \circ A$

Then g^e is computed using the tables of powers $G_1 = g, G_3 = g^3, \ldots, G_{2^K-1} = g^{2^K-1}$:

$A \leftarrow G_{e_{l-1}}$
for $i = l - 2$ down to 0 **do**
  $A \leftarrow A \circ A$
  if $e_i \neq 0$ then
    $A \leftarrow A \circ G_{e_i}$
return A

Note that in practice it is not necessary to completely derive the representation $e_{l-1}, \ldots, e_0$ before starting the exponentiation; instead, left-to-right scanning can be used to determine it digit by digit when it is needed without storing it completely. Left-to-right sliding window exponentiation is a slight modification of the method described in [2, proof of Theorem 3]; the idea to use variable windows is from [2, p. 912])

Like binary exponentiation and 2^k-ary exponentiation, sliding window exponentiation has a variant that performs a right-to-left exponentiation:

for $d = 1$ to $2^k - 1$ step 2 **do**
  $B_d \leftarrow$ identity element
$A \leftarrow g$

for $i = 0$ to $l - 1$ **do**
  if $e_i \neq 0$ **then**
    $B_{e_i} \leftarrow B_{e_i} \circ A$
  if $i < l - 1$ **then**
    $A \leftarrow A \circ A$

{Now $g^e = \Pi_{d \in \{1,3,\ldots,2^k-1\}} B_d^d$; this can be computed as follow :}

for $d = 2^k - 1$ to 3 step -2 **do**
  $B_{d-2} \leftarrow B_{d-2} \circ B_d$
  $B_1 \leftarrow B_1 \circ (B_d \circ B_d)$
return B_1

Again, in practice it is not necessary to completely derive the representation $e_{l-1}, \ldots, e_0$ before starting the exponentiation; here, right-to-left scanning can be used to determine it digit by digit when it is needed. The algorithm as written can be optimized similarly to the right-to-left 2^k-ary exponentiation method to avoid (at least) 2^{k-1} applications of the group operation. The idea used to perform sliding window exponentiation in right-to-left fashion is due to Yao [3]; the sub-algorithm shown above for computing $\prod_{d \in \{1,3,\ldots,2^k-1\}} B_d^d$ is due to Knuth [1, answer to exercise 4.6.3–9].

The number of group operations performed during a sliding window exponentiation with maximum window width k depends on the length l of the sliding window representation and on the number of digits in the representation $e_{l-1}, \ldots, e_0$ that are non-zero. For any b-bit exponent ($2^{b-1} \leq e \leq 2^b$), the length l is bounded by $b - k < l \leq b$. Assume that left-to-right or right-to-left scanning is performed on a sequence of independently and uniformly random bits; then a new window will be started on average every $k + 1$ bits. For b-bit exponents, one bit is necessarily nonzero, and both scanning techniques will usually have an unused part in the final window when the end of the exponent is reached. The expected number of nonzero values among $e_{l-1}, \ldots, e_0$ for random b-bit exponents lies between $b/(k + 1)$ and $1 + (b - 1)/(k + 1)$.

Using the upper bounds to derive estimates for average performance that are on the safe side (i.e. slightly pessimistic) gives b squaring operations (one time $g \circ g$ and $b - 1$ times $A \circ A$) and

$$2^{k-1} - 1 + \frac{b-1}{k+1}$$

general group operations for left-to-right sliding window exponentiation, or

$$2^{k-1} - 2 + b$$

squaring operations ($b - 1$ times $A \circ A$ and $2^{k-1} - 1$ times $B_d \circ B_d$) and

$$\underbrace{1 + \frac{b-1}{k+1}}_{\text{loop over } i} + \underbrace{2 \cdot (2^{k-1} - 1)}_{\text{loop over } d} - \underbrace{2^{k-1}}_{\text{optimization}} = 2^{k-1} - 1 + \frac{b-1}{k+1}$$

general group operations for right-to-left sliding window exponentiation with the optimization explained above.

In some groups, such as those employed in elliptic curve cryptography, computing inverses of elements is a very fast operation. For such groups, better performance than with ordinary sliding window exponentiation can often be obtained by using signed digit exponentiation instead.

Bodo Möller

References

[1] Knuth, D.E. (1998). *The Art of Computer Programming—vol. 2: Seminumerical Algorithms* (3rd ed.). Addison-Wesley, Reading, MA.

[2] Thurber, E.G. (1973). "On addition chains $l(mn) \leq l(n) - b$ and lower bounds for $c(r)$." *Duke Mathematical Journal*, 40, 907–913.

[3] Yao, A.C.-C. (1976). "On the evaluation of powers." *SIAM Journal on Computing*, 5, 100–103.

SMARTCARD TAMPER RESISTANCE

Tamper-resistant cryptographic modules are devices intended for applications that need to protect stored cryptographic keys and intermediate results of algorithms against unauthorized access. The most popular portable form is the smartcard, which has the form of a banking plastic card with embedded microcontroller. The typical interfaces are either five visible electrical contacts (for ground, power supply, reset, clock, and a bi-directional serial port) or an induction loop. Typical smartcard processors are 8-bit microcontrollers with a few hundred bytes of RAM and 4–64 kilobytes of ROM or non-volatile writable memory (NVRAM). Battery-like small steel cans ("crypto buttons"), CardBus/PCMCIA modules, and various PCI plug-in cards for non-portable applications are other popular form factors for tamper-resistant modules.

Smartcards are used in applications with both tamper-resistance and tamper-evidence requirements. Tamper resistance means that stored information must remain protected, even when the attacker can work on several samples of the module undisturbed for weeks in a well-equipped laboratory. Tamper evidence is a weaker requirement in which the regular holder of the module must merely be protected against unnoticed access to information stored in the module.

One common application for tamper-resistant smartcards are pay-TV conditional-access systems, where operators hand out millions of cards to customers, each of which contains the key necessary to descramble some subscription TV service. Pirates who manage to extract the key from one single issued card can use it to produce and sell illicit clone cards. Most proposed forms of digital rights management (DRM) mechanisms are based on some form of tamper-resistant element in the user system.

Examples for smartcard applications where operators can rely more on just a tamper-evidence requirement are digital signature identity cards, banking cards, and GSM subscriber identity modules. Here, stored secrets are specific to a single card or cardholder and can be revoked, should the module get stolen.

There are four broad categories of attacks against tamper-resistant modules:

- *Software attacks* use the normal communication interface of the processor and exploit security vulnerabilities found in protocols, cryptographic algorithms, or the software implementation. Countermeasures involve very careful design and in-depth implementation reviews, possibly augmented by formal techniques.
- *Microprobing* techniques access the chip surface directly, such that the attacker is able to observe, manipulate, and interfere with the integrated circuit. This has been the dominant form of attack against pay-TV conditional-access cards since about 1993. Chemical depackaging (e.g., with fuming nitric acid) is used to dissolve conventional packaging materials without damaging the silicon chip. Microscopes with micromanipulators are then used to place fine tungsten hairs onto micrometer-wide on-chip bus lines, in order to establish an electrical contact between the chip circuits and recording equipment such as digital oscilloscopes. The glass passivation layer that covers the metal interconnects can be broken mechanically or removed with UV laser pulses. The content of the main memory can then be reconstructed from observed on-chip bus traffic, a process that can be simplified by damaging the instruction decoder to prevent the execution of jump commands. Attackers have also succeeded in accessing the memory with the help of circuitry placed on the chip by the manufacturer for post-production testing. Modern chips with smaller feature sizes require the use of focused ion-beam workstations. With these, the surface of a depackaged chip can be modified inside a vacuum chamber. A beam of accelerated gallium ions and various added processing gases remove chip material or deposit either conducting and insulating substances with a resolution of tens of nanometers. This not only allows attackers to modify the metal connections between the transistors, effectively to edit the processor design, but also helps in establishing larger probing pads for the connection of recording equipment. Countermeasures involve more difficult-to-remove packaging materials (e.g., silicon, silicon carbide), obfuscated circuits,

additional top-layer metal sensor meshes, the careful destruction of test circuitry before the chip is delivered to customers, and the design of instruction decoders that frustrate modifications aimed at simplifying access to all memory locations [1].

- *Fault generation* techniques or fault attacks use abnormal environmental conditions to generate malfunctions in the processor aimed at providing additional access. A simple example would be a deliberately caused and carefully timed glitch that disrupts the correct execution of a single security-critical machine instruction, such as the conditional branch at the end of a password comparison. Carefully placed, a single glitch can help to bypass many layers of cryptographic protection. Such glitches have been generated by increasing the provided clock frequency for a single cycle, by brief supply voltage fluctuations, by applying light flashes to the entire chip or single gates, and with the help of electromagnetic pulses. Another class of fault generation attacks attempts to reduce the entropy generated by hardware random-bit generators. For example, where multiple noisy oscillators are used to generate randomness, externally applied electromagnetic fields with carefully selected frequencies can result in a phase lock and more predictable output. Countermeasures against fault generation include adding filters into supply lines, regular statistical checks of random-bit generators, redundant consistency checks in the software, and new logic design techniques that lead to inherently glitch-resistant circuits.
- Eavesdropping or side-channel analysis techniques monitor with high time resolution the characteristics of all supply and interface connections and any other electromagnetic radiation produced by a processor. A simple example is the determination of the length of the correct prefix of an entered password from the runtime of the string-compare routine that rejects it. This can significantly reduce the average number of guesses needed to find the correct string. The nature of the executed instruction, as well as parts of the processed data, are evident in the power-supply current of a CPU. A conditional branch that takes effect can easily be distinguished from one that passes through by examining with an oscilloscope the voltage drop over a 10 Ω resistor inserted into the processor's ground connection line. The current consumed by the write operation into memory cells is often proportional to the number of bits that change their value. Even status register flags and Hamming weights of data processed in arithmetic units can show up in power consumption curves. The technique of differential power analysis determines secret-key bits by correlating measured current curves with externally simulated intermediate results of a symmetric cipher. It has been demonstrated as a practical attack technique, even in situations where there has not been a microprobing attack first to disassemble the software in the targeted smartcard. Countermeasures include the addition of filters and shields against compromising emanations, circuitry and routines for adding random noise and delays, new balanced or dual-rail logic design techniques that lead to inherently less information in the power signal, and algorithmic techniques for reducing the number of intermediate results useful for eavesdroppers.

Microprobing requires time and careful preparation in a laboratory environment and is therefore primarily a challenge of tamper resistance. The other three attack classes are noninvasive and can, with suitable preparation, be performed in just a few seconds with attack equipment that could be disguised as a regular smartcard reader. The holder of the card might not notice such an attack, and then even the tamper evidence would be lost.

Markus Kuhn

References

[1] Kömmerling, Oliver and Markus G. Kuhn (1999). "Design principles for tamper-resistant smartcard processors." *Proceedings of the USENIX Workshop on Smartcard Technology (Smartcard '99), Chicago, IL, USA, May 10–11*, Berkeley, CA, US. USENIX Association, 9–20, ISBN 1-880446-34-0.

[2] Weingart, Steve H. (1965). "Physical security devices for computer subsystems: A survey of attacks and defenses." *Workshop on Cryptographic Hardware and Embedded Systems (CHES 2000)*, Lecture Notes in Computer Science, vol. 1965. Springer-Verlag, Berlin, 302–317.

S/MIME

S/MIME (see also Security Standards Activities) is the IETF Internet Security Syntax for MIME (Multipurpose Internet Mail Extensions), currently available in version 3, under constant development and communicated in a range of RFCs (abbreviation for "Request for Comments"). It

basically specifies the syntax for the integration of various cryptographic mechanisms and algorithms within the MIME format scope.

The Cryptographic Message Syntax (CMS) (see RFC 3369) is cryptographic algorithm independent, but, typically, applying an actual algorithm is not entirely defined uniquely and requires some attendance and care for seamless interoperability.

As part of the specification update, a new suite of "mandatory to implement" algorithms are constantly being selected, reflected in updates to Certificate Handling (RFC 2632), and S/MIME v3 Message Specification (RFC 2633).

Building on the CMS Compressed Data content type specified in RFC 3274, the update to RFC specifies conventions for message compression as well as to message signature and encryption. Few are used in reality.

To aid implementers, documentation containing example output for CMS is made available, some of which for example, include structures and signed attributes defined in the Enhanced Security Services (ESS) (RFC 2634) document.

CMS, and thus S/MIME version 3 and later, permit the use of previously distributed symmetric key-encryption keys, and the underlying Public Key Infrastructure (PKI) is based on the PKIX standard, e.g. for certificates and CRLs (see certificate revocation), whilst the underlying syntax for cryptographic mechanisms rely on the PKCS standards.

Peter Landrock

Reference

[1] See http://www.networksorcery.com/enp/ for all RFCs.

SMOOTHNESS

A natural number n is called B-smooth if its factorization does not contain any prime factors larger than B, i.e.

$$n = \prod_{p \le B} p^{n_p}.$$

Analogously, we can also consider elements of the polynomial ring $\mathbb{F}_p[x]$. For a polynomial f of degree $\deg(f)$ define the norm of f to be $p^{\deg(f)}$. An element of $\mathbb{F}_p[x]$ is called B-smooth if its factorisation does not contain any irreducible polynomials of norm greater than B.

For $t, c \in \mathbb{R}$ such that $0 \le t \le 1$ the complexity-theoretic L-notation is defined by

$$L_x[t, \gamma] = e^{(\gamma + o(1))(\log x)^t (\log\log x)^{1-t}},$$

where $x \to \infty$. Note that for $t = 0$ this equals $(\log x)^\gamma$, while for $t = 1$ we obtain x^γ (neglecting the $o(1)$ term). Hence we see that for values of t between 0 and 1 the function L interpolates between polynomial time and exponential time behaviour. For these values we say that L is subexponential in x (see subexponential time).

The main observation about the distribution of smooth numbers in an interval $[0, a]$ is that if the smoothness bound B is chosen subexponentially in x, then the probability that a random integer in this interval is B–smooth (or more precisely the inverse of that probability) is also subexponential.

More precisely, set $a = L_x[r, \alpha]$ and $B = L_x[s, \beta]$, where $r, s, \alpha, \beta \in \mathbb{R}_{>0}$ and $s < r \le 1$, then the probability that a random number in $[0, a]$ is B–smooth is given (see smoothness probability) by

$$L_x[r - s, -\alpha(r - s)/\beta] \tag{1}$$

where $x \to \infty$.

A similar result holds for the polynomial case: Assume $r, s, \alpha, \beta \in \mathbb{R}_{>0}$ such that $r \le 1$ and essentially $s < r$. Then the probability that a random element of $\mathbb{F}_p[x]$ of norm bounded by $L_x[r, \alpha]$ is $L_x[s, \beta]$–smooth is given exactly by expression 1 (see [3] for details).

Smooth numbers or polynomials are used in the most effective methods to factor natural numbers (see integer factoring) and compute discrete logarithms in finite fields (see discrete logarithm problem). The overall subexponential complexity of these methods is a direct consequence of the fact that the number of smooth elements in a given interval grows subexponentially if the smoothness bound is chosen subexponential as well.

For further background, please see [1, 2, 4].

Kim Nguyen

References

[1] Canfield, E.R., P. Erdös, and C. Pomerance (1983). "On a problem of Oppenheim concerning 'Factorisatio Numenerorum.'" *Journal of Number Theory*, 17, 1–28.

[2] De Bruijn, N.G. (1966). "On the number of positive integers $\le x$ and free of prime factors $> y$." *Indag. Math.*, 38, 239–247.

[3] Odlyzko, A.M. (1985). "Discrete logarithms in finite fields and their cryptographic significance." *Advances in Cryptology—EUROCRYPT'84*, Lecture Notes in Computer Science, vol. 209, eds. T. Beth, N. Cot, and I. Ingemarsson. Springer-Verlag, Berlin.

[4] Ramaswami, V. (1949). "The number of positive integers $\leq x$ and free of prime divisors $< x^c$, and a problem of S.S. Pillai." *Duke Math. J.*, 16, 99–109.

SMOOTHNESS PROBABILITY

Let $\alpha, \beta, r, s \in \mathbf{R}_{>0}$ with $s < r \leq 1$. With L_x as in L-notation, it follows from [1, 2] that a random positive integer $\leq L_x[r, \alpha]$ is $L_x[s, \beta]$-smooth (see smoothness) with probability

$$L_x[r - s, -\alpha(r - s)/\beta], \quad \text{for } x \to \infty.$$

Arjen K. Lenstra

References

[1] Canfield, E.R., P. Erdös, and C. Pomerance (1983). "On a problem of Oppenheim concerning 'Factorisatio Numerorum.'" *Journal of Number Theory*, 17, 1–28.

[2] De Bruijn, N.G. (1966). "On the number of positive integers $\leq x$ and free of prime factors $> y$, II. *Indag. Math.*, 38, 239–247.

SOLITAIRE

Solitaire is a stream cipher designed to be implemented using a deck of cards. It was invented by Bruce Schneier for use in the novel *Cryptonomicon*, by Neal Stephenson [1], where it was called Pontifex. Solitaire gets its security from the inherent randomness in a shuffled deck of cards. By manipulating this deck, a communicant can create a string of "random" letters which he then combines with his message. Solitaire can be simulated on a computer, but it is designed to be used by hand.

Manual ciphers are intended to be used by spies in the field who do not want to be caught carrying evidence that they send and receive encrypted messages. In David Kahn's book *Kahn on Codes* [2], he describes a real pencil-and-paper cipher used by a Soviet spy. Both the Soviet algorithm and Solitaire take about the same amount of time to encrypt a message: most of an evening.

Solitaire, as described in the appendix to *Cryptonomicon*, has a cryptographic weakness. While this weakness does not affect the security of short messages, Solitaire is not recommended for actual use.[1]

Bruce Schneier

[1] See http://www.schneier.com/solitaire.html

References

[1] Stephenson, Neal (2002). *Cryptonomicom*. Avon Eos Books, Avon.

[2] Kahn, David (1984). *Kahn on Codes: Secrets of the New Cryptology*. Macmillan Publishing Co., London.

SPKI/SDSI

SPKI (Simple Public Key Infrastructure) [2, 1] was developed starting in 1995 to remedy shortcomings [3] in the existing ID certificate definitions: X.509 and PGP (see Pretty Good Privacy). It provided the first authorization certificate definition [4, 5]. Originally, SPKI used no names for keyholders but, after the merger with SDSI (Simple Distributed Security Infrastructure), now includes both named keyholders and named groups or roles—specifying authorization grants to names and definitions of names (membership in named groups).

In public-key security protocols, the remote party (the *prover*) in a transaction is authenticated via public key cryptography. Upon completion of that authentication, the *verifier* has established that the prover has control over a particular private key—the key that corresponds to the public key the verifier used. This public key is itself a good identifier for the prover. It is a byte string that is globally unique. It also has the advantages of not requiring a central ID creator or distributor and of being directly usable for authentication. However, since anyone can create a key pair at any time, a raw public key has no security value. It is the purpose of a certificate to give value or meaning to this public key.

ID certificate systems bind names to public keys. This is an attempt to directly answer the question "who is that other party?". The shortcomings of ID certificates that SPKI addresses are:

1. Because there is no single, global name source, names are not globally unique. Therefore mapping from public key to name can introduce nonuniqueness. In SPKI, the real identifier is a public key or its cryptographic hash—each of which is globally unambiguous.
2. Names have no special value to a computer, but are strongly preferred by people over raw keys or hash values. However, people have a limited ability to distinguish from among large numbers of names, so the use of names can introduce scaling problems. The original SPKI did not use names, but SDSI names are defined by

Table 1. Certificate name sources

Type	Source of names
X.509	Certificate Authority (CA)
PGP	End Entity (EE)
SDSI	Relying Party (RP)

the *Relying Party (RP)* and presumably limited to the set that the RP can distinguish.

3. Name assignments are made by some Certificate Authority *(CA)* and the introduction of that additional component reduces overall system security. In SPKI/SDSI there is no CA, in the X.509 sense.
4. The real job is to make a security decision and a name by itself does not give enough information to make that decision. SPKI carries authorization information.

There are certain characteristics of SPKI/SDSI that set it apart from other certificate systems: SDSI names, authorization algebra, threshold subjects, canonical S-expressions and certificate revocation (see authorization architecture, authorization management, and authorization policy).

SDSI NAMES: Keys, and by implication their keyholders, need to be identified. SPKI uses the public key itself or its cryptographic hash as the ID of the key and the keyholder. This ID is globally unique and requires no issuer, therefore no expense or added insecurity of an ID issuer. For computers and the protocols between them, this ID is nearly perfect: globally unique and directly authenticable. The hash of the key has the added advantage of being fixed length.

For humans, such IDs fail miserably. They have no mnemonic value. SPKI uses SDSI names for human interfaces. Each human in a system using SPKI/SDSI maintains his or her own dictionary mapping between that human's preferred name for a keyholder and the public key or hash. The human operator can see friendly and meaningful names displayed via a UI, while the underlying system uses the key or its hash as an ID.

Source of Names

There is sometimes confusion among *X.509*, *PGP*, and SDSI—all of which build name certificates. The best way to distinguish them is via the source of the names used (see Table 1).

X.509 started out planning to use globally unique assigned names from the one global X.500 directory. That single directory has never been created and is unlikely ever to be. This leaves X.509 names to be chosen by the CA that issues a certificate. PGP leaves choice of name up to the person generating the key. SDSI gives choice of name to the person who will need to use that name.

Advantage of SDSI Names. When a name is used by a human, the correctness of that use depends on whether the human calls the correct person, thing, or group, to mind on seeing the name. When SDSI names are used, the one who chose that name is the same person who must correctly understand the linkage between the name and the person, thing, or group.

For example, the RP might choose the SDSI name "John Smith", if the RP knows only one John Smith—but a global naming authority would form "John Smith 3751" or jsmith39@localisp.net and require the RP to somehow deduce from that name which John Smith was intended. If the RP has an offline channel to John Smith and can ask him what his global ID is, then the RP can keep a local mapping from his preferred "John Smith" to the global name—but that is exactly what happens with SDSI (the global name being the hash of a key). If the RP does not have off-line contact with this John Smith, then the RP is forced to guess which John Smith is behind the name—and that guess is a source of security error [6].

Group Names

Both X.509 and PGP assume that the name is of an individual. SDSI names are of groups or roles. A named individual is a group of one.

Globally Unique SDSI Names

There are times when a SDSI name needs to be included in a certificate: when rights are assigned to the name or the name is added to some other named group. Since SDSI names are inherently local, a global form must be constructed. For example:

```
(name (hash sha1
  #14dc6cb49900bdd6d67f03f91741cfefa2d26fa2#)
  Leanna)
```

stands for the name Leanna in the local dictionary of the keyholder of the key that hashes via SHA1 to 14dc6cb49900bdd6d67f03f91741cfefa2d26fa2.

This is an advantage of SPKI/SDSI over other ID certificate forms. SDSI knew that names were local and had to do something to make them globally unique while X.509 and PGP assumed names were global, even when they were not.

AUTHORIZATION ALGEBRA: SPKI carries authorization information in its certificates and (Access Control List) ACL entries. This authorization is constrained to be in a language defined by SPKI so that the SPKI library can perform set intersections over authorizations. Each authorization is a set of specific permissions. That set is expressed as an enumeration (a literal set) or in a closed form (e.g., ranges of strings or numbers). The language is defined by intersection rules [2] that were designed to permit the intersection of two authorization sets to be expressed in the same closed form (see also authorization architecture, authorization management, and authorization policy).

By contrast, X.509v3 certificate extensions can be used to carry permission information, but because the extension is completely free-form, custom code must be written to process each different extension type.

FORMAT: An SPKI certificate has five fields:

1. Issuer: the key of the certificate issuer.
2. Subject: a key, hash, SDSI name or threshold subject construct.
3. Delegation: a Boolean, indicating whether the Subject is allowed to delegate some or all of the rights granted here.
4. Tag: a canonical S-expression listing a set of rights granted by the issuer to the subject.
5. Validity: limits on validity: not-before or not-after dates, requirements to check online status or to get a revocation list, etc.

An SPKI ACL entry has fields 2.5 of the above since the authority (issuer) of an ACL entry is the machine that holds it.

A name membership certificate has four fields:

1. Issuer
2. Name being defined
3. Subject (key, hash or name)
4. Validity

There is one certificate for each member of a name.

A *threshold subject* is a list of N subjects (possibly including a subordinate threshold subject) and a parameter K. Only when K of the N subjects agree to delegate some rights or sign some document is that certificate or ACL entry considered valid. (The keys used by these subjects need not be in the same algorithm so, among other things, a threshold subject might tolerate the catastrophic break of one algorithm.)

Canonical S-expressions (CSEXP)

SPKI/SDSI certificates are expressed and communicated as canonical S-expressions. An S-expression is of power equivalent to XML (Extensible Markup Language). Canonical S-expressions are binary forms with only one possible encoding. S-expressions in SPKI/SDSI are constrained to have each list start with an atom (the equivalent of an XML element name). Atoms are binary strings, with an explicit length stated, so CSEXP creation is trivial. CSEXP parsing requires under 10KB of code, in the open-source implementation. If element names are kept small, CSEXP binary forms are smaller than equivalent ASN.1 forms.

CERTIFICATE REVOCATION: At the time SPKI was designed, X.509 used *Certificate Revocation Lists (CRL)* that were optional and were not dated. A new CRL could be issued at any time and would override any prior CRL. In SPKI, revocation is deterministic. Each certificate that could be subject to revocation includes the revocation/validation agent's key and URL and all validity instruments (CRLs, etc.) have contiguous, non-overlapping date ranges.

IMPLEMENTATIONS: SPKI certificates are used in HP's eSpeak and several prototype systems. It is available in open source code in two sourceforge.net projects: CDSA and JSDSI. SPKI's spiritual descendent XrML V.2 [5] is in use in Microsoft's Rights Management Services (RMS).

Carl Ellison

References

[1] Ellison, Carl. SPKI/SDSI Certificates; http://theworld.com/~cme/html/spki.html

[2] Ellison, Carl, Bill Frantz, Butler Lampson, Ronald Rivest, Brian Thomas, and Tatu Ylönen (1999). SPKI Certificate Theory, IETF RFC2693, September 1999, ftp://ftp.isi.edu/in-notes/rfc2693.txt

[3] Ellison, Carl (2002). "Improvements on conventional PKI wisdom." *1st Annual PKI Research Workshop*, April 2002, http://www.cs.dartmouth.edu/~pki02/Ellison/

[4] Blaze, Matt. KeyNote; http://www.crypto.com/trustmgt/kn.html

[5] ISO/IEC JTC1/SC29/WG11/N5231: XrML V.2 (MPEG-21 Rights Expression Language) http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm#_Toc23297977

[6] Dohrmann, Steve and Carl Ellison (2002). "Public-key support for collaborative groups." *1st Annual PKI Research Workshop, April 2002*, 139–148. http://www.cs.dartmouth.edu/~pki02/Dohrmann/

SSH

Secure Shell, a product from SSH Communications Security, allows the user to log into another machine over a network, to execute commands in a remote machine, and to move files from one machine to another. For sometime, it was developed as a standard under IEFT.

SSH basically provides strong authentication and secure communications over insecure channels. It was originally intended as a replacement for various UNIX commands such as telnet, rlogin, rsh, and rcp. For SSH2, there was in addition a replacement for FTP, namely sftp.

When the standardisation was terminated, there were two versions of Secure Shell available: SSH1 and SSH2, which unfortunately are quite different and incompatible. As for the use of cryptographic algorithms, SSH1 supported DES (the Data Encryption Standard) Triple-DES, IDEA, and Blowfish, for encryption, while SSH supports 3DES, Blowfish, Twofish, and a few others. For authentication, SSH1 supported RSA digital signature scheme, while SSH2 supported the Digital Signature Standard.

Peter Landrock

STATION-TO-STATION PROTOCOL

In a two-party *authenticated key exchange* the legitimate parties can compute a secret key, while at the same time being certain about the authenticity of the parties with whom they exchange a key. The scheme must, in particular, be secure against a man-in-the-middle attack.

A popular authenticated version of the Diffie–Hellman key exchange protocol is the *Station-to-Station protocol*. It was proposed by Diffie-van Oorschot-Wiener [1].

Let $\langle g \rangle$ be a suitable finite cyclic group of large enough order in which the computational Diffie–Hellman problem is (assumed to be) hard. We assume that q (not necessarily prime) is a multiple of the order of g and publicly known. Let $sign_A(m)$ indicate the digital signature of the bitstring m by party A. So, $sign_A(m)$ can be verified using the public key of A. Let $E_k(m)$ be a conventional encryption of the bitstring m using the conventional key k. If k is too long, one assumes it is hashed (see hash function). The corresponding decryption is written as $D_k(\cdot)$.

The protocol, in which Alice (A) and Bob (B) want to exchange a key, works as following:

Step 1. A sends B $\alpha := g^{r_A}$ computed in $\langle g \rangle$, where r_A is chosen uniformly random in Z_q.

Step 2. B chooses r_B uniformly random in Z_q and computes $\beta := g^{r_B}$ in $\langle g \rangle$, $k_B := \alpha^{r_B}$ in $\langle g \rangle$ and $\gamma_B := E_{k_B}(sign_B(\alpha, \beta))$, where α and β are concatenated. B sends A: β, γ_B.

Step 3. A computes $k_A := \beta^{r_A}$ and verifies whether the string $D_{k_A}(\gamma_B)$ is the digital signature of (α, β), signed by B. If so, she sends B: $\gamma_A := E_{k_A}(sign_A(\alpha, \beta))$ and views k_A as the authenticated key exchanged with B.

Step 4. B verifies whether the string $D_{k_B}(\gamma_A)$ is the digital signature of (α, β) signed by A. If so, B regards k_B as the authenticated key exchanged with A.

As in the Diffie-Hellman key agreement scheme, if there are no dishonest parties, Alice and Bob will exchange the same key, i.e. $k_A = k_B$.

Yvo Desmedt

Reference

[1] Diffie, W., P.C. van Oorschot, and M.J. Wiener (1992). "Authentication and authenticated key exchanges." *Designs, Codes and Cryptography*, 2, 107–125.

STREAM CIPHER

A *stream cipher* is a symmetric cryptosystem (see key) which operates with a time-varying transformation on individual plaintext digits. By contrast, block ciphers operate with a fixed transformation on large blocks of plaintext digits. More precisely, in a stream cipher a sequence of plaintext digits, $m_0 m_1 \ldots$, is encrypted into a sequence of ciphertext digits $c_0 c_1 \ldots$ as follows: a pseudo-random sequence $s_0 s_1 \ldots$, called the running-key or the *keystream*, is produced by a finite state automaton whose initial state is determined by a secret key. The ith keystream digit only depends on the secret key and on the $(i-1)$ previous plaintext digits. Then, the ith ciphertext digit is obtained by combining the ith plaintext digit with the ith keystream digit.

Stream ciphers are classified into two types: synchronous stream ciphers and *asynchronous stream ciphers*. The most famous stream cipher is the Vernam cipher, also called *one-time pad*, that leads to perfect secrecy (the ciphertext gives no information about the plaintext).

Stream ciphers have several advantages which make them suitable for some applications. Most notably, they are usually faster and have a lower hardware complexity than block ciphers. They are also appropriate when buffering is limited, since the digits are individually encrypted and decrypted. Moreover, synchronous stream ciphers are not affected by errorpropagation (see also nonlinear feedback shift register).

Anne Canteaut

References

[1] Rueppel, R.A. (1986). *Analysis and Design of Stream Ciphers.* Springer-Verlag, Berlin.

[2] Vernam, G.S. (1926). "Cipher printing telegraph systems for secret wire and radio telegraphic communications." *Journal of the American Institute of Electrical Engineers*, 55, 109–115.

STRONG PRIME

A strong prime [1] is an integer p such that

- p is a large prime.
- $p-1$ has a large prime number factor, denoted r.
- $p+1$ has a large prime factor.
- $r-1$ has a large prime factor.

The precise qualification of "large" depends on specific attacks the strong prime is intended to protect against. For a long time, strong primes were believed to be necessary in the cryptosystems based on the RSA problem in order to guard against two types of attacks: factoring of the RSA modulus by the $p+1$ and Pollard $p-1$ factoring methods, and "cycling" attacks. Rivest and Silverman [2] published a paper in 1999 arguing that strong primes are unnecessary in the RSA public key encryption system. There are two points in their argument. First, that the use of strong primes provides no additional protection against factoring attacks, because the Elliptic Curve Method for factoring is about as effective as the $p+1$ and the $p-1$ methods (though none is particularly likely to succeed for random, large primes) and is not prevented by the strong prime conditions. Furthermore, the Number Field Sieve can factor RSA modulus with near certainty in less time than these methods. (See integer factoring for a discussion on factoring methods.) Secondly, they argue that cycling attacks are extremely unlikely to be effective, as long as the primes used are large. This has recently been formally proven in [3]. Thus, in the current state of knowledge, there is no rationale for requiring strong primes in RSA. A new factoring method might once again make strong primes desirable for RSA, or on the contrary exploit the properties of strong primes in order to factor more efficiently and thus make strong primes appear to be dangerous.

Anton Stiglic

References

[1] Gordon, J. (1985). "Strong primes are easy to find." *Advances in Cryptology—EUROCRYPT'84*, Lecture Notes in Computer Science, vol. 209, eds. T. Beth, N. Cot, and I. Ingemarson. Springer-Verlag, Berlin, 216–223.

[2] Rivest, R.L. and R.D. Silverman (1998). "Are 'strong' primes needed for RSA?" Technical Report, RSA Data Security, Inc., Redwood City, CA, USA.

[3] Shparlinski, I., J.B. Friedlander, and C. Pomerance (2001). Period of the power generator and small values of Carmichael's function. Math. Comp., vol. 70, 1591–1605.

STRONG RSA ASSUMPTION

Let $1 < \tau \in \mathbb{Z}$ be a security parameter. Let $N = pq$ be a product of two random τ-bit primes and let s be an element of the group $\mathbb{Z}_N^*$ (see also modular arithmetic). The strong-RSA problem is defined as follows:

given (N, s) as input, output a pair $a, b \in \mathbb{Z}$ such that $a^b = s \bmod N$ and $b \neq \pm 1$.

Loosely speaking, the Strong-RSA assumption states that for a sufficiently large τ the strong RSA problem is intractable.

The Strong-RSA assumption was introduced by Baric and Pfitzman [2]. The assumption is used to construct efficient signature schemes that are existentially unforgeable under a chosen message attack *without* the random oracle model. One such system is described in [4] and another in [3]. The Strong-RSA assumption is also the basis of several efficient group signature schemes [1].

Dan Boneh

References

[1] Ateniese, Giuseppe, Jan Camenisch, Marc Joye, and Gene Tsudik (2000). "A practical and provably secure coalition-resistant group signature scheme." *Advances in Cryptology—CRYPTO 2000, August*, Lecture Notes in Computer Science, vol. 1880, ed. M. Bellare. Springer-Verlag, Berlin, 255–70.

[2] Baric, N. and B. Pfitzman (1997). "Collision-free accumulators and fail-stop signature schemes without trees." *Proceedings of Eurocrypt*, Lecture Notes in Computer Science, vol. 1233, ed. W. Fumy. Springer-Verlag, Berlin, 480–494.

[3] Cramer, Ronald and Victor Shoup (2000). "Signature schemes based on the strong RSA assumption." *ACM* Transactions on Information and System Security (ACM TISSEC), 3 (3), 161–185, extended abstract in *Proc. 6th ACM Conf. on Computer and Communications Security, 1999*.

[4] Gennaro, Rosario, Shai Halevi, and Tal Rabin (1999). "Secure hash-and-sign signatures without the random oracle." *Advances in Cryptology—EUROCRYPT'99*, Lecture Notes in Computer Science, vol. 1592, ed. J. Stern. Springer-Verlag, Berlin, 123–139.

STRUCTURAL CRYPTANALYSIS

Structural Cryptanalysis is a branch of *Cryptanalysis* which studies the security of cryptosystems described by generic block diagrams. It analyses the syntactic interaction between the various blocks, but ignores their semantic definition as particular functions. Typical examples include meet-in-the-middle attacks on multiple encryptions, the study of various chaining structures used in modes of operation, and the properties of Feistel structures or substitution–permutation networks with a small number of rounds.

Structural attacks are often weaker than actual attacks on given cryptosystems, since they cannot exploit particular weaknesses (such as bad differential cryptanalysis properties or weak *avalanche effect*) of concrete functions. The positive side of this is that they are applicable to large classes of cryptosystems, including those in which some of the internal functions are unknown or key dependent. Structural attacks often lead to deeper theoretical understanding of fundamental constructions, and thus they are very useful in establishing general design rules for strong cryptosystems.

Alex Biryukov

SUBEXPONENTIAL TIME

A *subexponential-time* algorithm is one whose running time as a function of the size k of its input grows more slowly than b^x for *every* base $b > 1$. That is, for every constant base $b > 1$, the running time $T(x)$ satisfies

$$T(x) < b^x$$

for all sufficiently large x. In O-notation, this would be written $T(x) = 2^{o(x)}$ or $e^{o(x)}$.

(In computational complexity, subexponential security sometimes refers to the related notion that for all $\epsilon > 0$, $T(x) < 2^{x^\epsilon}$, for all sufficiently large x.)

Subexponential-time algorithms occur in cryptography in connection with the discrete logarithm problem and integer factoring. The fastest algorithms known for those problems (i.e., the ones that grow most slowly as a function of input size) typically have running times of the form

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

for some constants $\gamma > 0$ and $0 < t < 1$, where x is the order of the finite field in which discrete logarithms are being computed, or the modulus to be factored. The size of the input to these algorithms is proportional to the length in bits of x, so the running time, being subexponential in $\log x$, is subexponential in the input size as well (see L-notation).

For further discussion, see exponential time and polynomial time.

Burt Kaliski

SUBGROUP

A subset of elements of a group that is itself a group, i.e., that follows the group axioms (closure, associativity, identity, inverse). For example, if $G = (S, \times)$ is a group, then for any $g \in S$, the set of elements

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

(together with the multiplication operation) is a subgroup of G. The order of any subgroup of a group G divides the order of the group G itself; this is known as *Lagrange's theorem*.

Burt Kaliski

SUBGROUP CRYPTOSYSTEMS

In cryptographic applications it is often advantageous to replace a generator of the multiplicative group $\mathbf{F}^*_{p^t}$ of a finite field $\mathbf{F}_{p^t}$ of characteristic p by a generator g of a subgroup of $\mathbf{F}^*_{p^t}$, as originally suggested by Schnorr [2]. The subgroup

$\langle g \rangle$ generated by g must be chosen in such a way that solving the discrete logarithm problem in $\langle g \rangle$ is not easier than computing discrete logarithms in $\mathbf{F}_{p^t}^*$.

Because of the Pohlig–Hellman algorithm (see discrete logarithm problem), the order of g must be chosen in such a way that it contains a sufficiently large prime factor. Usually, g is chosen in such a way that its order q is prime. Because q divides the order $p^t - 1$ of $\mathbf{F}_{p^t}^*$ and because $p^t - 1 = \prod_{s \text{ dividing } t} \Phi_s(p)$, where $\Phi_s(X)$ is the tth cyclotomic polynomial (as defined in the generalization of Pollard's $p-1$ method—see integer factoring), the prime order q of g divides $\Phi_s(p)$ for one of the s dividing t. However, if q divides $\Phi_s(p)$ for some $s < t$, then $\langle g \rangle$ can effectively be embedded in the proper subfield $\mathbf{F}_{p^s}$ of $\mathbf{F}_{p^t}$. This has the undesirable consequence that the discrete logarithm problem in $\langle g \rangle$ can be solved in the multiplicative group $\mathbf{F}_{p^s}^*$ of the substantially smaller field $\mathbf{F}_{p^s}$, which is easier than solving it in $\mathbf{F}_{p^t}^*$. Thus, in order not to affect the hardness of the discrete logarithm problem in $\langle g \rangle$, the order q of g must be chosen as a sufficiently large prime divisor of $\Phi_t(p)$. Given q, a proper g can be found as $g = h^{(p^t-1)/q}$ for any $h \in \mathbf{F}_{p^t}^*$ such that $g \neq 1$.

If $t = 1$, this implies that p must be chosen so that $\Phi_1(p) = p - 1$ has a large enough prime factor q. If $t = 2$, however, q must be a large prime factor of $\Phi_2(p) = p + 1$ and if $t = 6$ of $\Phi_6(p) = p^2 - p + 1$. The case $t = 1$ corresponds to the traditional and conceptually easiest choice of using the prime field $\mathbf{F}_{p^t} = \mathbf{F}_p$: for 1024-bit security, representation of elements of the subgroup $\langle g \rangle$ requires about 1024 bits. The latter two cases, $t = 2$ and $t = 6$ (or, more generally, t divisible by 2 or 6, respectively) are of interest because they allow a more efficient representation of the subgroup elements when LUC or XTR are used (where LUC [3] refers to 'Lucas' because of LUC's use of Lucas sequences, and XTR [1] is an abbreviation of ECSTR which stands for *efficient compact subgroup trace representation*). For 1024-bit security $1024/2 = 512$ bits suffice for even t when using LUC and $1024/3 \approx 342$ bits suffice for t divisible by 6 when using XTR. Let f be the factor indicating the improvement in representation size: $f = 2$ for LUC and $f = 3$ for XTR.

For any finite field $\mathbf{F}_u$, extension field $\mathbf{F}_{u^v}$, and $w \in \mathbf{F}_{u^v}$ the *trace* $Tr(w)$ of w over $\mathbf{F}_u$ is defined as the sum of the v conjugates of w over $\mathbf{F}_u$: $Tr(w) = \sum_{0 \leq i < v} w^{u^i} \in \mathbf{F}_u$ (the inclusion in $\mathbf{F}_u$ because $Tr(w)^u = Tr(w)$). LUC and XTR work by representing elements of $\langle g \rangle$ by their trace over the subfield $\mathbf{F}_{p^{t/f}}$. The resulting representation advantage of a factor f compared to the traditional representation applies in principle to any element of $\mathbf{F}_{p^t}$. When applied to the order-$\Phi_t(p)$ subgroup G of $\mathbf{F}_{p^t}^*$ with t as above, however, the trace representation has other important advantages: given $Tr(w) \in \mathbf{F}_{p^{t/f}}$ for any $w \in G$, it determines w and its conjugates uniquely and the trace of any power of w can be computed very efficiently. Since g was chosen in such a way that $\langle g \rangle \subset G$, this fast 'exponentiation' applies to the subgroup $\langle g \rangle$ as well. LUC with $t = 2$ and XTR with $t = 6$ allow very efficient methods to find proper p and q of cryptographically relevant sizes. For large choices of t parameter selection becomes more cumbersome. For details of the exponentiation and parameter selection methods, see [3] for LUC and [1] and [4] for XTR.

The fact that the distinction between subgroup elements and their $p^{t/f}$-th powers (i.e., their conjugates over $\mathbf{F}_{p^{t/f}}$) is lost, has been shown (see [1]) to have no negative impact on the security of LUC and XTR. A potential disadvantage of the trace-based systems is that they complicate ordinary multiplication of subgroup elements (represented by their traces).

Arjen K. Lenstra

References

[1] Lenstra, A.K. and E. Verheul (2000). "The XTR public key system." *Advances in Cryptology—CRYPTO 2000*, Lecture Notes in Computer Science, vol. 1880, ed. M. Bellare. Springer-Verlag, Berlin, 1–19.

[2] Schnorr, C.P. (1991). "Efficient signature generation by smart cards." *Journal of Cryptology*, 4, 161–174.

[3] Smith, P. and C. Skinner (1995). "A public-key cryptosystem and a digital signature system based on the Lucas function analogues to discrete logarithms." *Proceedings ASIACRYPT'94*, Lecture Notes in Computer Science, vol. 917, eds. J. Pieprzyk and R. Safari-Naini. Springer-Verlag, Berlin, 357–364.

[4] Stam, M. and A.K. Lenstra (2001). "Speeding up XTR." *Proceedings ASIACRYPT 2001*, Lecture Notes in Computer Science, vol. 2248, ed. C. Boyd. Springer-Verlag, Berlin, 125–143.

SUBSTITUTIONS AND PERMUTATIONS

A substitution cipher is usually described by a sequence or list of single substitutions, each of which is commonly denoted by an arrow, like $p \longmapsto \pi$.

Example: The Russian-English ISO transliteration (using diacritical marks) is a substitution.

A B V G D E · Z I Ϯ K L M N O P R S T U F H C Q X W _ Y ^ / Yu ″
↧ ↧
A B V G D E Ž Z I Ĭ K L M N O P R S T U F H C Č Š Šč ’ *Y* ” Ė Ju Ja

A substitution may have homophones (see encryption).

A *permutation* is a one-to-one mapping from an alphabet to itself.

A substitution may be described by two lines: the first one being the standard alphabet, the second one being a mixed alphabet (see alphabet). An example is a given below:

a b c d e f g h i j k l m n o p q r s t u v w x y z
↧ ↧
B E K P I R C H S Y T M O N F U A G J D X Q W Z L V

Note that we have used small letters for the plaintext and capital letters for the ciphertext.

In mathematics, there is a commonly used, simplified notation with two lines bracketed together:

$$\downarrow \begin{pmatrix} \mathrm{a\ b\ c\ d\ e\ f\ g\ h\ i\ j\ k\ l\ m\ n\ o\ p\ q\ r\ s\ t\ u\ v\ w\ x\ y\ z} \\ \mathrm{B\ E\ K\ P\ I\ R\ C\ H\ S\ Y\ T\ M\ O\ N\ F\ U\ A\ G\ J\ D\ X\ Q\ W\ Z\ L\ V} \end{pmatrix}$$

This is convenient for encryption. For decryption, it is worth while to rearrange the list:

$$\uparrow \begin{pmatrix} \mathrm{q\ a\ g\ t\ b\ o\ r\ h\ e\ s\ c\ y\ l\ n\ m\ d\ v\ f\ i\ k\ p\ z\ w\ u\ j\ x} \\ \mathrm{A\ B\ C\ D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ O\ P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z} \end{pmatrix}$$

or

$$\downarrow \begin{pmatrix} \mathrm{A\ B\ C\ D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ O\ P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z} \\ \mathrm{q\ a\ g\ t\ b\ o\ r\ h\ e\ s\ c\ y\ l\ n\ m\ d\ v\ f\ i\ k\ p\ z\ w\ u\ j\ x} \end{pmatrix}$$

There is also the cycle notation which is shorter

(a b e i s j y l m o f r g c k t d p u x z v q) (h) (n) (w)

but this notation is inconvenient both for encryption and decryption. The cycle is generated by iterating the substitution on a arbitrarily chosen starting letter; whenever a cycle is closed, a new starting letter is chosen until all letters are exhausted.

Self-reciprocal permutations are permutations that, when applied twice, restore the original. Put equivalently, they are their own inverse. Their cycle notation shows a decomposition in 2-cycles and 1-cycle, for example:

(a n) (b x) (d s) (e i) (f v) (g h) (k u) (l c) (m q) (o w)
(p y) (j) (r) (t) (z)

If a self-reciprocal permutation has no 1-cycle (so n is even) there is also the following notation

$$\updownarrow \begin{pmatrix} \mathrm{a\ b\ c\ d\ e\ f\ g\ h\ i\ j\ k\ l\ m} \\ \mathrm{n\ o\ p\ q\ r\ s\ t\ u\ v\ w\ x\ y\ z} \end{pmatrix}$$

The Enigma machine of the German *Wehrmacht* used a (properly) selfreciprocal permutation. This was thought to be particularly practical since the same machine could be used for encryption and decryption, disregarding the fact that this opened ways for a cryptanalytic attack (see noncoincidence exhaustion in Cryptanalysis).

A *substitution cipher* in general replaces certain groups of characters by certain other groups of characters. This may be described by a list, e.g.

for $\mathbb{Z}_2^3 \to \mathbb{Z}_2^3$:

$$(000) \longmapsto (001),\ (001) \longmapsto (010),\ (010) \longmapsto (011),\ (011) \longmapsto (100),$$
$$(100) \longmapsto (101),\ (101) \longmapsto (110),\ (110) \longmapsto (111),\ (111) \longmapsto (000).$$

We shall give some more terms that one may see in this context. A *monographic* substitution is a substitution *of* single characters, while a *unipartite* substitution is a substitution *by* single characters.

A *simple* substitution is a substitution of single characters by single characters, so it is a monographic, unipartite substitution.

A *digraphic* substitution is a substitution *of bigrams* (ordered pairs of characters). A *bipartite* substitution is a substitution *by* bigrams. Finally, a *bigram* substitution is a substitution of bigrams by bigrams, so a digraphic, bipartite substitution.

In general, an *n-graphic* substitution is a substitution *of* n-tuples of characters (n-grams) and an *n-partite* substitution is a substitution *by* n-tuples of characters. Similarly, a *polygraphic* substitution is an n-graphic substitution, $n \geq 2$, and a *multipartite* substitution is an n-partite substitution, $n \geq 2$.

A *linear substitution* is a block encryption $\mathbb{Z}_N^n \to \mathbb{Z}_N^m$ that is the composition of a translation t and an homogenous part φ which is additive with respect to addition modulo N (for all $x, y \in \mathbb{Z}_N^n$: $\varphi(x + y) = \varphi(x) + \varphi(y)$).

A *null* is meaningless ciphertext character, the encryption image of the empty plaintext word. It is used, e.g., for swamping the plaintext statistics or masking the occurrence of idle times.

A *straddling encryption* or *straddling cipher* is a substitution with encryption steps $V^{(l)} \to W^{(m)}$, where $Z^{(k)}$ denotes the set of all sequences of at most k characters from Z, in formula $\{\varepsilon\} \cup Z \cup Z^2 \cup Z^3 \ldots \cup Z^k$, where Z^n is the set of all words of length n over the alphabet Z, and ε denotes the empty word.

Example: $Z_{20}^{(3)} \to \mathbb{Z}^{(2)}$ with the homophonic substitution

$$\downarrow \begin{pmatrix} \text{che} & \text{con} & \text{non} & \text{et} & \text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} & \text{g} & \text{h} & \text{i} \\ 44 & 64 & 00 & 08 & 1 & 86 & 02 & 20 & 62 & 22 & 06 & 60 & 3 \\ & & & & & & & & 82 & & & & \end{pmatrix}$$

$$\downarrow \begin{pmatrix} \text{l} & \text{m} & \text{n} & \text{o} & \text{p} & \text{q} & \text{r} & \text{s} & \text{t} & \text{v} & \text{z} & \epsilon \\ 24 & 26 & 84 & 9 & 66 & 68 & 28 & 42 & 80 & 04 & 88 & 5 \\ & & & & & & & & 40 & & & 7 \end{pmatrix}$$

Both 5 and 7 are in this example (Matteo Argenti, 1590) nulls. Other elements of $Z_{20}^{(3)}$ have no image, except by composition of their individual letters.

Let a *block* be a text of predetermined length. Then a block cipher or *block encryption* is a substitution with encryption steps $V^n \to W^m$, i.e. without straddling. The block length is usually rather high (for instance, the Data Encryption Standard has a block length of $m = n = 64$, and the *Advanced Encryption Standard* (see Rijndael/AES) has a block length of $m = n = 128$, 192, or 256 bits). The same block encryption step with its key is repeated on and on, thus, each bit of ciphertext in a given block normally depends on the complete corresponding plaintext block, with as consequence the possibility of error propagation over the full block.

A stream cipher (also called *stream encryption*) is a substitution $(V^n)^* \to (W^m)^*$ between infinite series of blocks, controlled by a key generating algorithm. The generated key may have a finite period. Autokey or other cipher feedback is excluded.

A *transposition cipher* or *tranposition* does not substitute the characters of a message, but permutes their position: it may be considered as a special case of a polygraphic substitution $V^n \to V^n$ of the kind

$$(x_1, x_2, \ldots, x_n) \longmapsto (x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}),$$

where π is a permutation of the *subscripts* $\{1, 2, \ldots, n\}$. It can be performed by multiplication of $(x_1, x_2, \ldots, x_n)$ with a *permutation matrix*, i.e., an $n \times n$ $\{0, 1\}$-matrix such that in every row and in every column, one occurs just once. This extreme property makes cryptanalysis of transposition ciphers very different from cryptanalysis of normal substitution ciphers and explains why alternating composition of substitutions and transpositions (see "pastry dough mixing" below) is so effective.

A *grille* is a tool, usually in the form of punch cards, that can be rotated to perform a transposition of the letters.

Pastry dough mixing stands for a composition of alternating substitutions and transpositions. It was already recommended by Shannon in 1949 and used, e.g., in the DES cryptosystem. The expression 'pastry dough mixing' was introduced by Eberhard Hopf in the mathematical theory of compact spaces.

Friedrich L. Bauer

Reference

[1] Bauer, F.L. (1997). "Decrypted secrets." *Methods and Maxims of Cryptology*. Springer-Verlag, Berlin.

SUBSTITUTION–PERMUTATION (SP) NETWORK

Shannon [1] suggested to use several mixing layers interleaving substitutions and permutations to build strong block ciphers. Such design is called a *substitution–permutation sandwich* or a substitution-permutation network (SPN). Although weak on its own, a line of *substitutions* followed by a *permutation* has good "mixing" properties: substitutions add to local *confusion* and permutation "glues" them together and spreads (*diffuses*) the local confusion to the more distant subblocks (see also substitutions and permutations). If one considers flipping a single bit at the input of such a network, it effects the m output bits of particular S-box which in turn are sent to different S-boxes by a permutation. Thus inputs/outputs of up to m S-boxes would be effected by the *avalanche* of change. These are again permuted into different S-boxes, covering almost all the S-boxes of the network. On the output of such network about half of the bits are effected by change and are flipped and about half of the bits are not flipped. This makes an outcome of a single bit change at the input hard to predict, especially if secret key bits are mixed into the block between the layers of encryption. Without a secret key the SPN performs a complex but fully deterministic function of its inputs. Modern ciphers tend to use linear or affine mappings instead of permutations, which allows them to achieve better diffusion in fewer iterations. Such networks are called substitution-linear (SLN) or substitution-affine networks (SAN). The current block encryption standard Rijndael/AES is a SLN cipher.

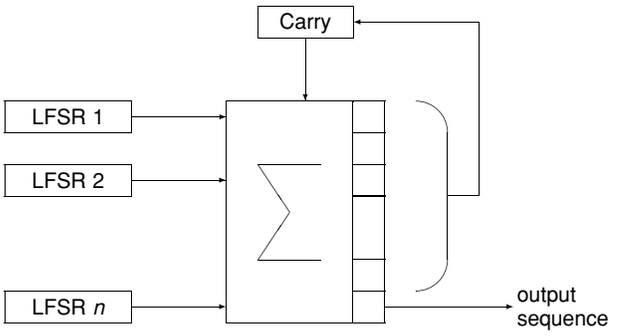
Alex Biryukov

Reference

[1] Shannon, C.E. (1949). "Communication theory of secrecy system." *Bell System Technical Journal*, 28, 656–715.

SUMMATION GENERATOR

The summation generator is based on a *combination* of n Linear Feedback Shift Registers (LFSRs) and was first proposed in [5, 6]. The combining function is an addition over the set of integers. From a binary point of view, it is a *nonlinear function*, with maximum correlation immunity. The output bit is the least significant bit of the integer sum.



Powerful attacks exist in the case $n = 2$ [1, 4]. Hence, it is better to use several LFSRs, with moderate lengths, than just a few large ones. But it has also been shown that this scheme is vulnerable if all the LFSRs are short [3]. A Fast Correlation Attack has recently been presented in [2]. (See also combination generator.)

Caroline Fontaine

References

[1] Dawson, E. (1993). "Cryptanalysis of summation generator." *Advances in Cryptology—ASIACRYPT'92*, Lecture Notes in Computer Science, vol. 718, eds. J. Seberry and Y. Zheng. Springer-Verlag, Berlin, 209–215.
[2] Golic, J., M. Salmasizadeh, and E. Dawson (2000). "Fast correlation attacks on the summation generator." *Journal of Cryptology*, 13, 245–262.
[3] Klapper, A. and M. Goresky (1995). "Cryptanalysis based on 2-adic rational approximation." *Advances in Cryptology—CRYPTO'95*, Lecture Notes in Computer Science, vol. 963, ed. D. Coppersmith. Springer-Verlag, Berlin, 262–273.
[4] Meier, W. and O. Staffelbach (1992). "Correlation properties of combiners with memory in stream ciphers." *Journal of Cryptology*, 5, 67–86.
[5] Rueppel, R.A. (1986). *Analysis and Design of Stream Ciphers*. Springer-Verlag, Berlin.
[6] Rueppel, R.A. (1986). "Correlation immunity and the summation generator." *Advances in Cryptology—CRYPTO'85*, Lecture Notes in Computer Science, vol. 218, ed. H.C. Williams. Springer-Verlag, Berlin, 260–272.

SYMMETRIC CRYPTOSYSTEM

The type of cryptography in which the same key is employed for each of the operations in the cryptosystem (e.g., encryption and decryption),

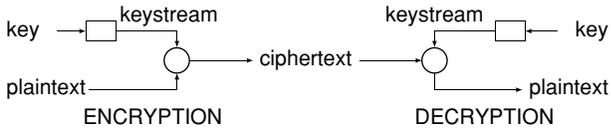
and thus that same key, typically a secret, must be shared by the parties performing the various operations. See also block cipher, stream cipher, MAC algorithms, and (for the contrasting notion) asymmetric cryptosystem.

Equivalent names are conventional cryptosystem, secret key cryptosystem, classical cryptosystem, and private key cryptosystem.

Burt Kaliski

SYNCHRONOUS STREAM CIPHER

A synchronous stream cipher consists of a cipher, in which the keystream is generated independently of the plaintext and of the ciphertext. It can be depicted as follows:



The keystream is usually produced by a pseudo-random generator, parameterized by a key, which is the secret key of the whole scheme.

This means that it is impossible to dynamically check the synchronization between the keystream and the message. The keystreams generated by the sender (encryption), and by the receiver (decryption) must be perfectly synchronized. If synchronization is lost, then decryption fails immediately. If we want to be able to resynchronize both signals, we need some additional techniques (through reinitialization, or by putting some marks in the message, ...).

Nevertheless, there is an advantage, in terms of errors of transmission. If the ciphertext is altered by some errors, then this will only affect the decryption of the wrong bits, but this will have no effect on the others.

These two properties (perfect synchronization needed, no propagation of errors) lead to some active attacks: the first one could be to modify the ciphertext in order to desynchronize the message and the keystream during decryption (this can easily be achieved by deleting or inserting some bits, for example); the second one consists in modifying the values of some bits, in order to modify the plaintext obtained after decryption (this can be powerful if the attacker knows sufficient information about the message in order to choose the meaning of the modified plaintext). This implies that it is important to use, at the same time as encryption, some integrity/authentication techniques in order to avoid such attacks.

Most of the stream ciphers used nowadays (see for example E0 and SEAL) are *binary additive stream ciphers*; they are synchronous stream ciphers, in which all the data (plaintext, keystream, and ciphertext) are binary, and that simply add (through the XOR function) the message (plaintext/ciphertext) to the keystream.

A good reference on the topic is [1].

Caroline Fontaine

Reference

[1] Rueppel, R.A. (1986). *Analysis and Design of Stream Ciphers*. Springer-Verlag, Berlin.