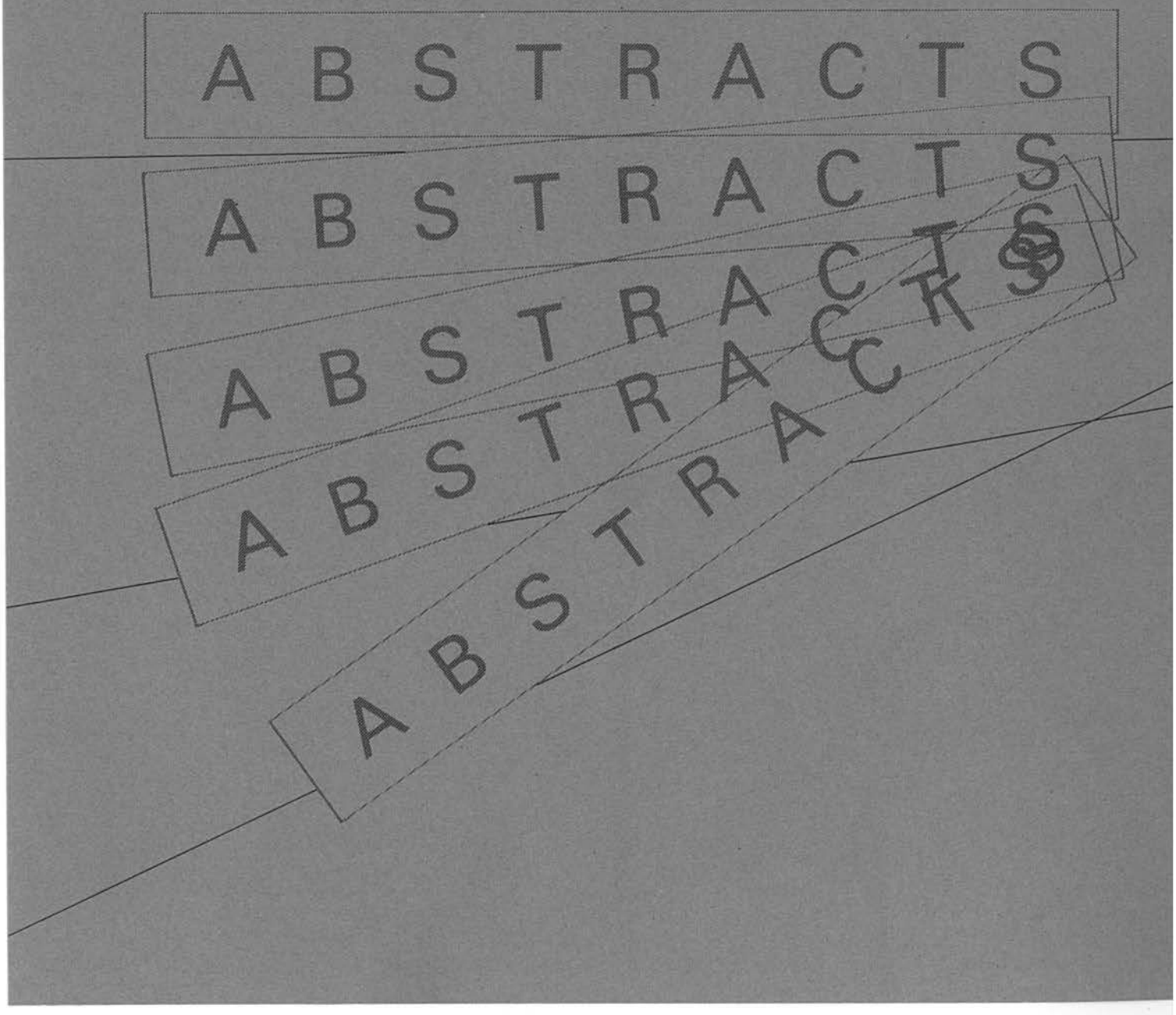
CWI

APRIL 13 - 15

EUROCRYPT 87 AMSTERDAM

ABSTRACTS



APRIL 13 - 15

EUROCRYPT 87 AMSTERDAM

ABSTRACTS

ORGANISATION

D. Chaum, CWI,
General Chairman

Organising Committee:
J. van de Graaf, CWI
C. Jansen, Philips USFA
G. Roelofsen, PTT-DNL
J. van Tilburg, PTT-DNL

Programme Committee:
W.L. Price, NPL, Chairman
T. Beth, U. Karlsruhe
J.-H. Evertse, CWI
L. Guillou, CCETT
T. Herlestam, U. Lund
F. Piper, U. London
J.J. Quisquater, Philips
C.P. Schnorr, JWG U.

CONTENTS

I

SEQUENCES AND LINEAR COMPLEXITY: A

A GENERATOR OF PSEUDORANDOM SEQUENCES WITH CLOCK CONTROLLED LINEAR FEEDBACK SHIFTREGISTERS

C.G. Günther

Brown Boveri Research Center

CH-5405 Baden, Switzerland

EXTENDED ABSTRACT*): During the last years two types of pseudorandom number generators based on linear feedback shift registers (LFSR's) have been investigated intensively. The first type of generators is based on the nonlinear correlation-immune feedforward combination of the output of one or several LFSR's [1]-[2], and the second one is based on a chain of LFSR's in which the clock of a given LFSR is controlled by the output of the preceeding LFSR in that chain [3]-[5]. The generators of the first type can be chosen to have a very large period and excellent statistical properties and to be secure against a correlation attack on one or several LFSR's. These generators, however, have a linear complexity that only depends algebraically on the length of the various LFSR's. The generators of the second type can be chosen to have a very large period and a linear complexing that is an exponential function of the length of one LFSR. Depending on the exact specifications of the generators one will, however, either have bad statistics (stop and go generator) [4] or a reduced bit rate (binary rate multiplier) [5]. Furthermore, in both cases the generators show some cryptographic weaknesses due to a lack of correlation immunity.

Considering the complementarity of the strengths and weaknesses of both schemes it is tempting to combine them to a new type of generators. These generators are composed of n+1 LFSR's with the clock of n LFSR's controlled by the output of the remaining one and with the output of the n LFSR's combined by a correlation immune feedforward function to give the output of the generator. One of the simplest such generator contains three LFSR's $\mathcal{K}$, $\mathcal{M}$ and $\overline{\mathcal{M}}$ which regularly clocked would generate the sequences κ, μ and $\bar{\mu}$. In the generators considered, the output κ of $\mathcal{K}$, however, clocks $\mathcal{M}$ and its complement clocks $\overline{\mathcal{M}}$. The outputs of $\mathcal{M}$ and $\overline{\mathcal{M}}$ are finally added to given the pseudorandom sequence ω:

$$\omega_t = \mu_{f_t} \oplus \bar{\mu}_{\bar{f}_t}$$

with $f_t := \sum_{s=0}^{t-1} \kappa_s$, $\bar{f}_t = t - f_t$ (see Fig. 1). Under certain simple assumptions these sequences have a period T which is the product of the periods of κ, μ and $\bar{\mu}$ and a linear complexity which is lower bounded by a quantity proportional to the period K of κ. The statistical properties of these sequences are described by two measures of randomness based on the autocorrelations

$$C_\omega^{(n)}(\tau_1 \ldots t_n) := \frac{1}{T} \sum_{t=0}^{T-1} (1-2\omega_{t+\tau_1}) \ldots (1-2\omega_{t+\tau_n})$$

and the frequency of patterns σ of length ℓ

$$\hat{C}_\omega^{(\ell)}(\sigma_0 \ldots \sigma_{\ell-1}) := \frac{1}{T} \operatorname{card}\{t \in \mathbb{Z}_T | \omega_{t+i} = \sigma_i, \forall\, i \in \mathbb{Z}_\ell\} \quad ,$$

respectively. These measures describe the deviations of the quantities C and $\hat{C}$ from the expectation of C and $\hat{C}$ for a sequence of statistically independent

equidistributed random variables. Under suitable conditions the first measure for ω is upper bounded by the product of the corresponding quantities for μ and $\bar{\mu}$ and the second measure is small for patterns of short lengths.

These results show that the new scheme can generate sequences which fulfil the classical requirements and which are generated at a rate equal to the maximal clock rate of the shift registers. Finally we note that the scheme is also immune against classical correlation attacks.

REFERENCES:

[1] T. Siegenthaler, "Correlation-immunity of nonlinear combining functions for cryptographic applications", IEEE Trans. on Inform. Theory, vol. IT-30, pp. 776-780, Sept. 1984.

[2] R.A. Rueppel, O.J. Staffelbach, "Products of linear recurring sequences with maximum complexity", IEEE Trans. on Inform. Theory, to appear.

[3] K. Kjeldsen, E. Andresen, "Some randomness properties of cascaded sequences", IEEE Trans. on Inform. Theory, vol. IT-26, pp. 227-232, March 1980.

[4] R. Vogel, "On the linear complexity of cascaded sequences", Advances in Cryptology - Proceedings of EUROCRYPT 84, Springer Lect. Notes in Computer Science, vol. 209, pp. 99-109.

[5] W.G. Chambers, S.M. Jennings, "Linear equivalence of certain BRM shift-register sequences", Elctronics Letters, vol. 20, pp. 1018-1019, Nov. 1984.

FIGURE:

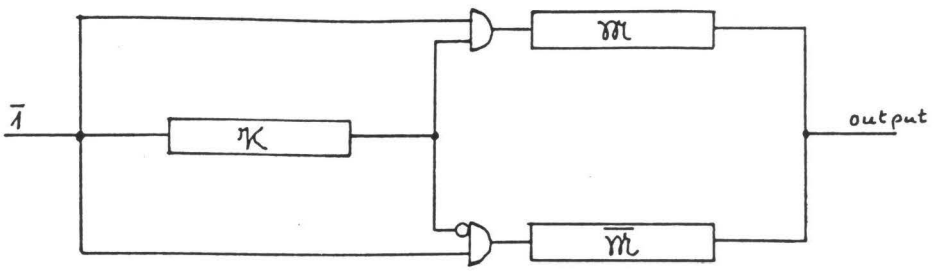


Fig. 1: A particularly simple generator of the new type. The feedback connections of the LFSR's are omitted, the inputs represent the clocks.

GENERATION OF BINARY SEQUENCES WITH CONTROLLABLE COMPLEXITY AND IDEAL r-TUPEL DISTRIBUTION

Thomas Siegenthaler
Amstein Walthert Kleiner
Information Systems Engineering AG
Leutschenbachstr. 45
8050 Zurich, Switzerland

Réjane Forré
Institute for Communications Technology
ETH Zentrum
8092 Zurich, Switzerland

Abstract:

A keystream generator is analyzed which consists of a single linear feedback shift register (LFSR) with a primitive connection polynomial and a nonlinear feedforward logic. It is shown, how, for arbitrary integers n and r and a binary LFSR of length $L = n \cdot r$ the linear complexity of the generated keystream can be determined for a large class of nonlinear feedforward logics. Moreover, a simple condition imposed on these logics ensures an ideal r-tupel distribution for these keystreams. Practically useful solutions exist where the keystream has linear complexity $n \cdot r^{n-1}$ together with an ideal r-tupel distribution.

I Introduction

A common type of keystream generator consists of a single binary linear feedback shift register (LFSR) and a feedforward logic (see Fig. 1).

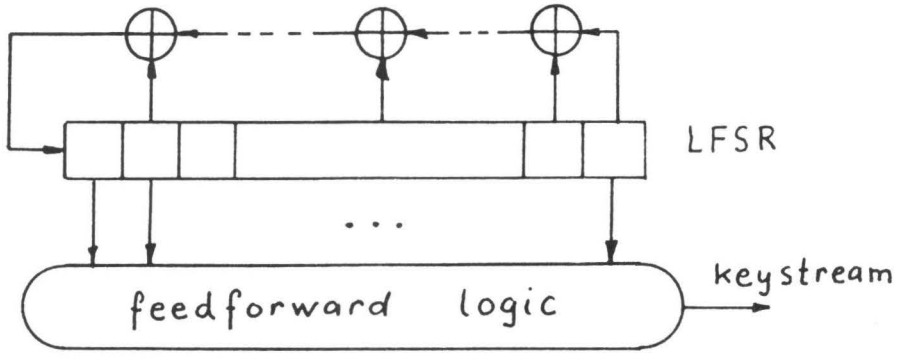


Fig. 1 A common type of keystream generator.

If the sequence produced by the LFSR has period p, all binary (keystream-) sequences of length p are generated by suitable feedforward logics. This makes the keystream generator of Fig. 1 attractive from the theoretical point of view. The type shown in Fig. 1 is also of considerable practical interest because it needs only a single (instead of several) LFSR. However, in the general case the analysis of this type of keystream generator has shown to be rather difficult [1]. Groth [2] proposed a layered structure for the feedforward logic to control the linear complexity of the generated keystream. This arrangement generates keystreams of large linear complexities, however, the statistics of these keystreams are hard to control. Rueppel suggested [3] a simple realisable and therefore practically useful class of feedforward logics such that a lower bound for the keystream's linear complexity is guaranteed. A closely related structure had independently been proposed by Günther/Bernasconi [4] which is also simple realisable and also guarantees a minimal linear complexity of the keystream. The latter two methods are based on the existence of one or several high order products in the corresponding algebraic normalform of the feedforward logic. A new approach [8] is proposed here. First, a number of "well choosen" delayed replicas (called "phases") of the seqence generated by the LFSR are picked then every nonlinear feedforward logic is allowed. The analysis uses the theory of finite fields GF(2^n). The approach is strongly based on an interpretation of two results recently obtained by Brynielsson. It is assumed that the LFSR of Fig. 1 has a primitive connection polynomial.

II Synthesis of keystream generators

In finite fields every function GF(q) --> GF(q) with x --> f(x) can be expressed as a polynomial [5]:

$$f(x) = \sum_{i=0}^{q-1} a_i x^i \quad ,$$

with coefficients (1)

$$a_i = \sum_{x \neq 0} [f(0) - f(x)] x^{-i}, \qquad a_i \in GF(q) \ .$$

Definition:

If the symbol y_k of the sequence $\{y_k\}$ over GF(q) is obtained as $y_k = f(x_k)$ where f denotes the polynomial in (1) and $\{x_k\}$ is a sequence over GF(q) then $\{y_k\}$ is called a **polynomial sequence**.

The following theorem is shown to be crucial for the computation of

the linear complexity of the keystream produced by a generator as given in Fig. 1.

Theorem: 1 (Brynielsson [6])
Let $\{x_k\}$ be a maximum length sequence over $GF(2^n)$ with a primitive characteristic polynomial of degree r and let $H(i)$ denote the Hamming weigth of the integer i. The polynomial sequence $\{y_k\}$ with $y_k = f(x_k)$ has linear complexity $LK(\{y_k\})$:

$$LK(\{y_k\}) = \sum_{a_i \neq 0} r^{H(i)} \quad , \quad a_i \in GF(2^n),$$

where the a_i's denote the coefficients in (1).

At a first glance polynomial sequences together with theorem 1 seem not to have any connection to the system of Fig. 1. Next, this connection is worked out with the help of Lemma 1 and Lemma 2. We consider a maximum length sequence $\{x_k\}$ over $GF(2^n)$. Symbols x_k from $GF(2^n)$ may be written as

$$x_k = x_{n-1,k} \cdot u^{n-1} + x_{n-2,k} \cdot u^{n-2} + \ldots + x_{1,k} \cdot u + x_{0,k} \quad , \qquad (2)$$

where the $x_{i,k}$'s belong to $GF(2)$ and where u denotes a primitive element of $GF(2^n)$. The n binary sequences $\{x_{i,k}\}$, $i = 0,1,\ldots,n-1$, in (2) are called the binary subsequences of $\{x_k\}$.

Lemma 1: (Brynielsson [7])
Let $\{x_k\}$ be a maximum length sequence over $GF(2^n)$ with (primitive) characteristic polynomial $p(x)$ of degree r. The binary subsequences $\{x_{i,k}\}$, $i = 0,1,\ldots,n-1$, of $\{x_k\}$ are linear independent and fulfil the **same linear recursion** with an associated (primitive) characteristic polynomial $q(x)$ of degree $L = r \cdot n$.

Therefore, the subsequences $\{x_{i,k}\}$ differ only by delays of each other. The polynomial $q(x)$ can be determined [7,8]. The following Lemma 2 is well known.

Lemma 2:
Let $\{z_k\}$ be a binary maximum length sequence with (primitive) characteristic polynomial $q(x)$ of degree L. Every delayed version $\{z_{k-d}\}$, where d denotes an integer in the range $[0,\ldots,2^L-1]$ of $\{z_k\}$ can be obtained by some linear combination of the sequences $\{z_{k-1}\},\{z_{k-2}\},\ldots,\{z_{k-L}\}$.

This means that every phase of the maximum length sequence generated by the LFSR of Fig. 1 can be obtained as a linear

combination of the sequences from the L stages of this LFSR. We are now ready to establish the connection between Theorem 1, Lemma 1, Lemma 2 and a system as given in Fig. 1. Consider a maximum length sequence $\{x_k\}$ over GF(2^n). Choose any of the binary subsequences $\{x_{i,k}\}$ mentioned in Lemma 1, say $\{x_{0,k}\}$. This binary subsequence is generated by a binary LFSR of length L, its feedback connections are known from q(x). The binary subsequences $\{x_{i,k}\}$, i =1,2, ..., n-1, are only phase shifts of $\{x_{0,k}\}$ (Lemma 1) and can be obtained as linear combinations of the sequences at the L stages of the LFSR that generates $\{x_{0,k}\}$ due to Lemma 2. (Instead of generating the maximum length sequence $\{x_k\}$ over GF(2^n) by a corresponding LFSR of length r with feedback connections due to p(x), $\{x_k\}$ is generated by a (binary) LFSR of length L with feedback connections due to q(x) and linear combinations of the sequences occuring at the L stages of this LFSR.) Every feedforward logic can now be applied to the n binary sequences $\{x_{0,k}\}$, $\{x_{1,k}\}$, ..., $\{x_{n-1,k}\}$ to produce the binary keystream $\{y_k\}$. This feedforward logic is then described as a polynomial f: GF(2^n) --> GF(2) with $y_k = f(x_k)$ as given in expression (1). The linear complexity of $\{y_k\}$ is computed by theorem 1. The corresponding system is shown in Fig.2.

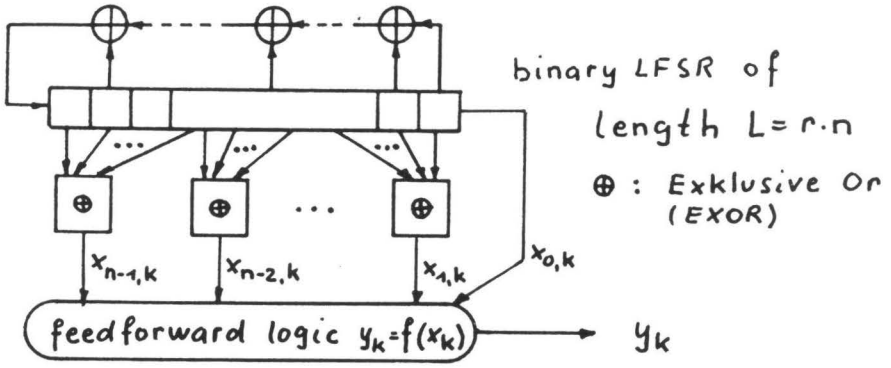


Fig.2 Synthesis of keystream generators.

Because of the required EXOR blocks, the system of Fig. 2 is a slightly restricted version of that shown in Fig. 1. But the linear complexity can be exactly determined for arbitrary feedforward logics as given in Fig. 2. So far we have not mentioned the r-tupel distribution of the keystream $\{y_k\}$. The r-tupel $\underline{y}_i$ is defined as a sequence $[y_i, y_{i+1}, \ldots, y_{i+r-1}]$ of successing symbols of $\{y_k\}$. The set $\underline{Y} = [y_0, y_1, \ldots, y_{p-1}]$ contains all r-tupels of the sequence $\{y_k\}$ of period p. The following definition is useful:

Definition:

A binary sequence $\{y_k\}$ of period $p = 2^L - 1$ exhibits an **ideal r-tupel distribution** $\underline{Y}$, $1 \leq r \leq L$, if exactly one of the 2^r possible and disjoint binary r-tuples occurs $2^{L-r} - 1$ times in a period of $\{y_k\}$ and each of the others occurs 2^{L-r} times.

Lemma 3:

An ideal r-tupel distribution of $\underline{Y}$ implies ideal r'-tupel distributions of $\underline{Y}$ for all r' with $1 \leq r' \leq r$.

Proof:

From an ideal r-tupel distribution follows that exactly one of the 2^r possible and disjoint binary r-tupels occurs $2^{L-r} - 1$ times and each of the others 2^{L-r} times. Therefore, exactly one r'-tupel, $1 \leq r' \leq r$, occurs $(2^{L-r} - 1) + 2^{L-r} \cdot (2^{r-r'} - 1) = 2^{L-r'} - 1$ times and each of the others occurs $2^{L-r} \cdot 2^{r-r'} = 2^{L-r'}$ times, as was to be shown.

Theorem 2:

Let $\{x_k\}$ denote a maximum length sequence over $GF(2^n)$ of period $2^{nr} - 1$ and f a polynomial $f: GF(2^n) \dashrightarrow GF(2)$. A polynomial sequence $\{y_k\} = \{f(x_k)\}$ exhibits an ideal r'-tupel distribution for all r' with $1 \leq r' \leq r$ for $x \in GF(2^n)$ if and only if

$$|\{x:f(x) = 1\}| = 2^{n-1}, \qquad (3)$$

where $|.|$ denotes the cardinality of the enclosed set $\{.\}$.

Proof:

Assume $|\{x:f(x) = 1\}| = b$ and $|\{x:f(x) = 0\}| = c$ with $b+c = 2^n$ and $f(0) = 0$. All r-tupels $\underline{x}_i = [x_i, x_{i+1}, \ldots, x_{i+r-1}]$ for $i = 0, 1, \ldots 2^{nr} - 1$ in the maximum length sequence $\{x_k\}$ are disjoint and every possible 2^n-ary nonzero r-tupel occurs exactly once. Binary r-tupels in $\{y_k\}$ occur from $\underline{x}_i = [x_i, x_{i+1}, \ldots, x_{i+r-1}]$ as $\underline{y}_i = [y_i, y_{i+1}, \ldots, y_{i+r-1}]$ with $y_i = f(x_i)$. First, we note that the 1-tupel distribution of $\underline{Y}$ is such that [1]-tupels occur b times and [0]-tupels occur c-1 times. Therefore, the 1-tupel distribution of $\underline{Y}$ is ideal iff $b=c=2^{n-1}$. Lemma 3 implies that none of the r'-tupel distributions for $1 \leq r'$ is ideal if the 1-tupel distribution of $\underline{Y}$ is not. Therefore, (3) is a necessary condition for an ideal r-tupel distrubution of $\underline{Y}$. This condition is also sufficient as is shown now. First, nonzero r-tupels $\underline{y}_i$ are considered. From the

assumption $f(0) = 0$ follows that for nonzero $\underline{y}_t$'s the involved $\underline{x}_t$'s are nonzero too. From (3) follows that there exist 2^{n-1} values x_t, such that $f(x_t) = y_t = 1$ (or 0). Hence, the number of r-tupels $\underline{x}_t$ which are mapped into the same nonzero binary r-tupel $\underline{y}_t$ is $(2^{n-1})^r$ or 2^{L-r} for $L = n \cdot r$. It remains to consider r-tupels $\underline{y}_t = \underline{0}$. From (3) follows that $(2^{n-1})^r - 1$ or $2^{L-r} - 1$ for $L = n \cdot r$ r-tupels $\underline{x}_t$ are mapped into $\underline{y}_t = \underline{0}$, where the -1 accounts for the missing r-tupel $\underline{x}_t = \underline{0}$ in the maximum length sequence $\{x_k\}$. This completes the proof. If $f(0) = 1$ is assumed, a similar proof exists.

From theorem 2 follows that a system as given in Fig. 2 generates a keystream $\{y_k\}$ with an ideal r-tupel distribution iff the polynomial $f:GF(2^n) \dashrightarrow GF(2)$ which describes the feedforward logic of Fig. 2 fulfils condition (3). The designer of such a system prefers polynomials f as given in (1) such that the following properties hold:

i) $f:GF(2^n) \dashrightarrow GF(2)$ (produces a **binary** sequence)
ii) f such that $|\{x:f(x) = 1\}| = |\{x:f(x) = 0\}|$ (ideal r-tupel distribution)
iii) f produces a keystream of **large** linear complexity
iv) f is easy to implement.

Solutions which fulfil all of the above requirements are described in [9] and will be discussed. For given integers r and n binary keystreams $\{y_k\}$ are produced which exhibit ideal r-tupel distribution together with a linear complexity $LK(\{y_k\}) = n \cdot r^{n-1}$.

III Conclusions

Keystream generators have been discussed which consist of a single binary LFSR and some feedforward logic. A new synthesis method has been proposed which is based on results recently obtained by Brynielsson. This method allows for a large class of feedforward logics to determine the linear complexity of the generated keystream. Moreover a simple necessary and sufficient condition has been derived to obtain keystreams which exhibit ideal r-tupel distributions.

References:

[1] T. Herlestam, "On Functions of Linear Shift Register Sequences", Advances in Cryptology - EUROCRYPT '85, Lecture Notes in Computer Science, No. 219, Springer Verlag, 1985, p. 119-129.

[2] E.J. Groth, "Generation of Binary Sequences with Controllable Complexity", IEEE Tr. on Inf. Theory, Vol. IT-17, No. 3, May 1971, p. 288-296.

[3] R.A. Rueppel, "New Approaches to Stream Ciphers", Diss. ETH, No. 7714, Zurich, 1984.

[4] J. Bernasconi, C.G. Günther, "Analysis of a Nonlinear Feedforward Logic for Binary Sequence Generators, Advances in Cryptology - EUROCRYPT '85, Lecture Notes in Computer Science, No. 219, Springer Verlag, 1985, p. 161-168.

[5] B. Benjauthrit, I.S. Reed, "Galois Switching Functions and their Applications", IEEE Tr. on Comp., Vol. C-25, No. 1, Jan. 1976, p. 78-86.

[6] L. Brynielsson, "On the linear Complexity of Combined Shift Register Sequences, Advances in Cryptology - EUROCRYPT '85, Lecture Notes in Computer Science, No. 219, Springer Verlag, 1985, p. 156-160.

[7] -, "Entwurf und Analyse eines Kryptosystems über GF(16)", Kryptologie Aufbauseminar, J. Kepler Universität, Linz, 1985.

[8] Th. Siegenthaler, "Methoden für den Entwurf von Stream Cipher -Systemen",Diss. ETH, No. 8185, Dec. 1986.

[9] R. Forré, "Analyse eines Chiffriergenerators", Diploma Project, Inst. for Communications Technology, ETH , Zurich, Dec. 1986.

Some Remarks on the

Cross Correlation Analysis of Pseudo Random Generators

S. Mund, D. Gollmann, T. Beth

Abstract

We consider pseudo random generators consisting of linear feedback shift registers and a coupling function. Siegenthaler has shown how cross correlation techniques can be used to identify the initial state of such a pseudo random generator. His algorithm takes time $O(R2^rN)$ to identify the initial state of one register. r denotes the length of the register, R the number of primitive polynomials of degree r, and N the number of bits one has to observe. Employing Walsh-Hadamard transform one can achieve a speed up to an $O(R(r2^r + N))$ algorithm.

We can show that there exists a trade-off between the dimension of the Hadamard matrix and the number of bits one has to observe. This may yield a further speed up to an $O(R(r2^{r-\delta} + 2^\delta N))$ algorithm. We then use the Multiplex-Generator as an example to demonstrate this algorithm.

Furthermore we examine the correlation immunity of the S-boxes used in the DES. Some of these show an "outlier" behaviour which is compared to the known irregularities of S-Boxes that have been reported.

II

SEQUENCES AND LINEAR COMPLEXITY: B

Sequences with almost perfect linear complexity profile

Harald Niederreiter (Vienna, Austria)

Extended abstract

For stream ciphers one needs pseudorandom sequences, i.e. deterministic sequences with acceptable randomness properties (see [2], [4]). A useful measure for randomness from the cryptographic viewpoint is the linear complexity of a sequence (see [1], [4]). Sequences with a linear complexity profile similar to that of truly random sequences may be viewed as pseudorandom sequences. We establish connections between the linear complexity profile of a sequence and the continued fraction expansion for the generating function of the sequence and we use these connections to analyze randomness properties of the sequence.

Let F_q be the finite field with q elements, where q is an arbitrary prime power. For a sequence $s_1,s_2,\ldots$ of elements of F_q and any positive integer n, the linear complexity $L(n)$ is defined as the least k such that the initial segment $s_1,s_2,\ldots,s_n$ of the sequence can be generated by a kth-order linear recursion (or equivalently by a feedback shift register with k delay elements), with the provision that $L(n) = 0$ if $s_i = 0$ for $1 \leq i \leq n$. The nondecreasing sequence $L(1),L(2),\ldots$ of nonnegative integers is called the linear complexity profile of the sequence $s_1,s_2,\ldots$. In the binary case $q = 2$, Rueppel [3] has shown that for random sequences the expected value of $L(n)$ is $\frac{n}{2} + c_n$ with $0 \leq c_n \leq \frac{5}{18}$. This has led to the following notion: a binary sequence has a perfect linear complexity profile if $L(n) = \lfloor (n+1)/2 \rfloor$ for all $n \geq 1$. Wang and Massey [5] proved that the binary sequence $s_1,s_2,\ldots$ has a perfect linear complexity profile if and only if it satisfies $s_1 = 1$ and $s_{2i+1} = s_{2i} + s_i$ for all $i \geq 1$.

Thus, every second term in a binary sequence with perfect linear complexity profile depends in a known manner on previous terms, an unsatisfactory state of affairs in a pseudorandom sequence.

Therefore we consider sequences with almost perfect linear complexity profile, i.e. sequences for which small deviations of $L(n)$ from its expected value are allowed. A relationship with continued fraction expansions is obtained in the following way. Let $s_1, s_2, \ldots$ be a sequence of elements of F_q and let $S = \sum_{i=1}^{\infty} s_i x^{-i}$ be its generating function, viewed as a formal power series in x^{-1}. Then S has a unique continued fraction expansion

$$S = 1/(a_1 + 1/(a_2 + \ldots)),$$

where $a_1, a_2, \ldots$ are polynomials over F_q of positive degree. We put

$$K(S) = \sup_{j \geq 1} \deg(a_j).$$

Theorem 1. If $s_1, s_2, \ldots$ is nonperiodic (or, equivalently, if S is not a rational function) and $K(S) < \infty$, then

$$\frac{1}{2}(n + 1 - K(S)) \leq L(n) \leq \frac{1}{2}(n + K(S)) \quad \text{for all } n \geq 1.$$

With an appropriate choice of the polynomials a_j we can therefore construct sequences for which $L(n)$ is always close to $(n+1)/2$. We have the following partial converse of Theorem 1.

Theorem 2. If $s_1, s_2, \ldots$ satisfies $L(n) \leq (n+1)/2$ for all $n \geq 1$ and $\lim_{n \to \infty} L(n) = \infty$, then $K(S) = 1$.

Corollary. The sequence $s_1, s_2, \ldots$ satisfies $L(n) = \lfloor (n+1)/2 \rfloor$ for all $n \geq 1$ if and only if it is nonperiodic and $K(S) = 1$.

In the case $q = 2$ we get in particular a new proof of the Wang-Massey characterization of binary sequences with a perfect linear complexity profile. Applications of these results to the construction of cryptographically useful pseudorandom sequences will also be discussed.

References

[1] T. Herlestam: On functions of linear shift register sequences, Advances in Cryptology - EUROCRYPT '85 (F. Pichler, ed.), Lecture Notes in Computer Science, Vol. 219, pp. 119-129, Springer-Verlag, Berlin, 1986.

[2] R. Lidl and H. Niederreiter: Introduction to Finite Fields and Their Applications, Cambridge Univ. Press, Cambridge, 1986.

[3] R. A. Rueppel: Linear complexity and random sequences, Advances in Cryptology - EUROCRYPT '85 (F. Pichler, ed.), Lecture Notes in Computer Science, Vol. 219, pp. 167-188, Springer-Verlag, Berlin, 1986.

[4] R. A. Rueppel: Analysis and Design of Stream Ciphers, Springer-Verlag, Berlin, 1986.

[5] M.-Z. Wang and J. L. Massey: The characterization of all binary sequences with a perfect linear complexity profile, Paper presented at EUROCRYPT '86, Linköping, May 1986.

Submitted to Eurocrypt 87, April 13-15, Amsterdam, The Netherlands.

When Shift Registers Clock Themselves

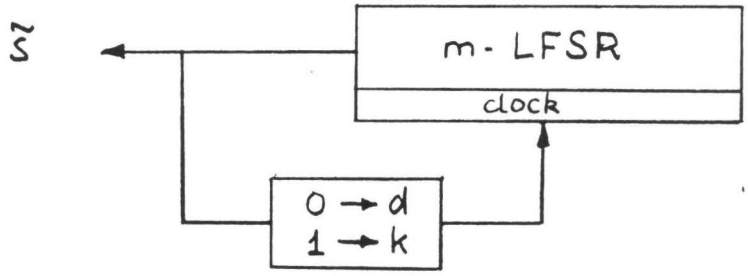
Rainer A. Rueppel
Crypto AG
6312 Steinhausen
Switzerland

Abstract:

A new class of sequences, which we term [d,k]-self-decimated sequences, is investigated. It is proved that when a binary m-sequence of degree L is [d,k]-self-decimated and [d,k]=t[1,2] with gcd(t,2^L-1), the resulting sequence is periodic with period $\lfloor(2/3)(2^L-1)\rfloor$. When $\tilde{s}$ denotes the semi-infinite periodic part of such a [d,k]-self-decimated m-sequence, it is shown that the short-term statistics of $\tilde{s}$ are close to being ideal. The number of ones within a period of $\tilde{s}$ is proved to be $\lfloor(1/3)(2^L-1)\rfloor$ which results in a perfectly balanced distribution when L is even. Tight upper and lower bounds for the frequencies of bit-pairs within a period of $\tilde{s}$ are derived which imply that the most frequent bit-pair can occur at most 3 times more often than the least frequent bit-pair, thereby demonstrating the flatness of the pair-distribution. Furthermore, exhaustive searches indicate that almost always the linear complexity (or linear span) of $\tilde{s}$ meets its theoretical maximum (which is the period length). Also empirically it is shown that the periodic autocorrelation function of $\tilde{s}$ has in general a low out-of-phase magnitude. These properties suggest that this new class of sequences may have some applications in cryptography and spread spectrum communications.

1. Introduction

Imagine we let the output sequence of a binary linear feedback shift register determine its own clock in the following way: whenever the output symbol is a '0', d clock pulses are applied to the LFSR, and, in case the output symbol is a '1', k clock pulses are applied to the LFSR. Figure 1 illustrates the system.

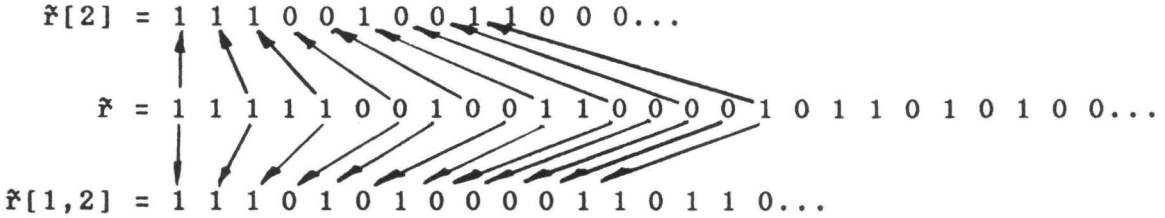


Suppose the above LFSR has a primitive connection polynomial $C(D)=1+D^2+D^3+D^4+D^5$ and is started in state [1 1 1 1 1]. When the self-clocking rule [d,k] is chosen to be [1,2] (i.e., for a '0', the LFSR is clocked once, and for a '1' the LFSR is clocked twice), then the following periodic sequence will appear at the output of the system:

$$\tilde{s} = (1\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1)$$

This sequence has remarkable properties: (1) the distribution of k-tuples is 'balanced' (to be more precise: for $1<k<3$, the frequencies of k-tuples differ by at most 2); (2) the linear complexity (or linear span) of $\tilde{s}$ is 20, which is the maximum possible for a sequence of period 20; (3) the periodic autocorrelation function of $\tilde{s}$ has a peak out-of-phase magnitude of 0.

This self-clocking operation can be interpreted as a generalization of the well-known and widely-studied decimation operation for LFSR-sequences. The conventional decimation of a sequence $\tilde{r}$ by a constant d is defined as the extraction of every d-th digit of $\tilde{r}$, usually denoted as $\tilde{r}[d]$. When a binary sequence $\tilde{r}$ is [d,k]-"self-clocked", then it is no longer decimated by a constant but by a function which depends on the value of the previous sequence digit; we will term the resulting sequence a **[d,k]-self-decimated sequence**. Let $\tilde{r}$ be the original m-sequence produced by the LFSR in Figure 1. Then the following figure compares decimation by 2, and the [1,2]-self-decimation of $\tilde{r}$ into $\tilde{s}$.



2. Theoretical Results

Let $\tilde{r}$ be a binary m-sequence whose characteristic polynomial has degree L, and let $\tilde{s}$ be the periodic part of $\tilde{r}[d,k]$ (which is equivalent to assuming that the phase of $\tilde{r}$ was chosen such that $\tilde{r}[d,k]$ has no preperiod. Then the following results can be proved.

Theorem 1:

The period of any [d,k]-self-decimated sequence $\tilde{s}=\tilde{r}[d,k]$, where $[d,k]=t[1,2] \bmod (2^L-1)$ with $\gcd(t,2^L-1)=1$, is

$$T = \lfloor (2/3)\,(2^L-1) \rfloor$$

Theorem 2:

The absolute frequencies of bits, N(b), within a period of any [d,k]-self-decimated sequence $\tilde{s}=\tilde{r}[d,k]$, where $[d,k]=t[1,2] \bmod (2^L-1)$ with $\gcd(t,2^L-1)=1$, are given as

$$N(1) = \lfloor (1/3)\,(2^L-1) \rfloor$$
$$N(0) = T-N(1)$$

Note that theorem 2 implies that when L is even the bit distribution is perfectly balanced, that is, $N(0)=N(1)=T/2$.

Theorem 3:

The absolute frequencies of bit-pairs, $N(b_1 b_2)$, within a period of any [d,k]-self-decimated sequence $\tilde{s}=\tilde{r}[d,k]$, where $[d,k]=t[1,2] \bmod (2^L-1)$ with $\gcd(t,2^L-1)=1$, are bound by

$$N_{min}(b_1 b_2) \leq N(b_1 b_2) \leq N_{min}(b_1 b_2) + 2$$

with

$$N_{min}(00) = \lfloor (1/6)(2^L-8) \rfloor$$
$$N_{min}(01) = \lfloor (1/6)(2^L-4) \rfloor$$
$$N_{min}(10) = N_{min}(11) = \lfloor (1/6)(2^L-2) \rfloor$$

Since the lower bounds on the absolute frequencies of the bit-pairs differ by at most 1, we conclude that the most frequent bit-pair can appear at most 3 times more often than the least frequent bit-pair within one period of $\tilde{s}$.

3. Simulation Results

In our simulations we concentrated so far on [1,2]-self-decimated m-sequences for degrees L=3,...,11. It showed that the pair-distributions, for a given L, where independent of the characteristic polynomial of the m-sequence. Exhaustive searches over all primitive polynomials of degree L=5,6,7,8 revealed the following averages and minimum values for the linear complexities of [1,2]-self-decimated sequences:

L	5	6	7	8
T	20	42	84	170
L_{avg}	19,3	38,7	82	169,3
L_{min}	16	33	78	166

The proximity of L_{avg} to the period length T and the largeness of the minimal encountered linear complexity L_{min} speak for themselves.
Another topic of interest is the periodic autocorrelation function. Exhaustive searches over all primitive polynomials of degrees L=4,5,6,7 revealed the following averages R_{avg} and minimum values R_{min} for the peak out-of-phase autocorrealtion magnitude of [1,2]-self-decimated sequences:

L	4	5	6	7
R_{avg}	4	4	9.3	20.5
R_{min}	2	0	6	12

4. Conclusion

Both the theoretical and the empirical results, as well as the ease of generation, indicate that the set of [d,k]-self-decimated m-sequences may have some applications in cryptography and spread spectrum communications.

Finite State Machine Modelling of Cryptographic Systems in LOOPS

Franz R Pichler
Systems Theory and Informations Engineering
Institute of Systems Science
Universisity of Linz
A-4040 Linz,Austria

Abstract

Finite State Machines constitute important mathematical objects for modelling electronic hardware specified above the register transfer level. Furthermore,by their recursiveness Finite State Machines are convenient means for realizing infinite wordfunctions built over finite alphabets. The related theory, the "finite automata theory" or also the "theory of sequential switching circuits" is well developed. However, by the poor standard of computer aided design tools in earlier years, engineers could not use this theory a lot in logic design. We believe that today, by the availability of modern workstations and object-oriented programming techniques, it is possible to enhance the design environment of an engineer by automata theory in an important way.

It is wellknown that many functions of cryptographic systems can be modelled by finite state machines. In that area of application they are appropriate means for model specification which give support to hardware-realization as well - and this is the more important part for our work here - to theoretical analysis tasks, such as machine-structure identification by experiments, investigation of controllability and observability properties or machine decomposition.In our lecture we want to deal which such tasks. For doing this we introduce the classes of VERNAM-machines,outputfeedback-machines,prime-machines and one-way machines to define important classes of finite automata which are used in stream-ciphering, block-ciphering and for the realization of one-way functions.

The lecture continues to investigate subclasses of finite state achine which have important properties for cryptographic applications. For example, in the class of outputfeedback-machines the subclass of machines with finite memory is important for the construction of selfsynchronizing cryptographic systems, as used for example in analog speech scrambling in mobil communication networks. A main goal in investigating subclasses is to find classes of machines which have a nice algebraic or combinatorical structure to allow by the existing theories efficient symbolic computation. In order that finite state machine theory becomes effective in cryptographic applications it is badly needed to know about methods to speed up existing algorithms and to be able to make some estimation on lower bounds of complexity measures.

For computer support of our research in that direction we use the AI programming system LOOPS and the programming language INTERLISP-D running on a SIEMENS/XEROX dandelion workstation. A goal for the future is to build an expertsystem for finite state machine theory which is applicable to cryptography. It is my hope that the lecture will convince the cryptographic research community that by the existing modern tools for explorative programming and Rapid-Prototyping the theory of finite state machines has high potentials for beeing used in cryptographic design.

III

HARDWARE TOPICS

Random Sources for Cryptographic Systems

G.B. Agnew
Dept. of Electrical Engineering
University of Waterloo
Waterloo, Ontario, Canada

Random and pseudorandom sources have long been of interest in the study of statistical processes. In these systems, pseudorandom sources (PRS) are preferred over true random sources (TRS) as repeatability (recreatability) of observations is important.

In cryptographic systems, true random sources are preferred in some instances to prevent penetration or influence of the system (some cryptographic systems in fact assume the availability of a true random source [1], [2]). It is observed that the strongest cryptographic system may be rendered insecure if an attacker can influence the generation of keys (presumably either by a PRS or TRS).

In this study we define a binary true random source (BTRS) as a device which generates an output pattern of 1's and 0's such that they are bitwise iid and all 2^n sequences of n bits are equally likely for any integer n. BTRS are generally based on measuring naturally occurring events such as sampling diode shot noise [3], time between radioactive particle decay, etc.. These devices are generally built external to the device using the random source. This is done for a number of reasons. First, the technology used for the random source is generally incompatible with the technology used for the cryptographic system (cryptodevice). Second, due to the nature of the process observed, temperature control or compensation is generally required. Third, shielding to prevent influence or observation may be required. Finally, if the random source is proven bias at a later date, it can be replaced by another source without redesign of the cryptodevice.

There are several drawbacks in using an external random source. If the system is subject to observation or influence along the path from the source to the cryptodevice, then the system may be insecure (in some cryptographic systems, it may not be necessary or desirable to expose the outcome of the random source to the external environment, e.g., Diffie-Hellman key exchange protocol [2]). To provide a cryptographically strong BTRS for implementation on a cryptodevice, several criteria must be met:

i) Compatibility with device technology
ii) Immunity to observation or influence
iii) Stability of source output and freedom from bias.

By including the BTRS as part of the cryptodevice, most observation attacks can be avoided. Outside influence on the other hand, can take on several forms. A good BTRS should provide immunity from outside influence due to:

i)Modulation of grounds, power, input or output lines
ii) electromagnetic fields or radiation
iii) temperature manipulation
iv) forced resetting of the device to a known starting state.

In this study we examine several techniques and structures for a VLSI implementation of a BTRS compatible within a cryptodevice. We model these structures, examine their output patterns and study how these designs meet the above requirements.

References

1. R. Rivest, A. Shamir, L. Adleman, 'On digital signatures and public key cryptosystems', Comm. of ACM, Vol. 21, Feb. 1978, pp.120-126

2. W. Diffie, M. Hellman, 'Privacy and authentication : An introduction to cryptography', Proc. of the IEEE, Vol. 67, March 1979, pp. 397-427.

3. D. Kahn, 'Cryptography and the origin of spread spectrum', IEEE Spectrum, Vol. 21, no. 9, Sept. 1984, pp. 70-80.

Title: **Physical Protection of Cryptographic Devices**

Author: Andrew J. Clark BSc. (Hons) A.M.I.E.E.
R & D Manager, Computer Security Limited

With the growth of user awareness for the need to protect sensitive computer data by cryptographic means, this paper explains the need to protect critical cryptographic variables (particularly cryptographic keys, and in some cases algorithms) in a secure environment within cryptographic equipment.

The principles of cryptographic device design are outlined, leading to the concept of tamper resistant and not tamper proof devices to protect key data, whether the data be retained within physically large devices or on small portable tokens.

Tamper resistant design criteria for the detection of attempts to gain access to sensitive data rather than attack prevention are outlined, together with two types of attack scenario - invasive and non-invasive.

Potential adversary's attack objectives are suggested together with a system designer's defence strategy; including design solutions to invasive and non-invasive attacks.

Typical detection mechanisms and sensor systems are discussed plus the design trade-offs that must be made in implementing tamper resistant cryptographic devices; in particular manufacturing and maintenance costs versus scope of attack protection.

Once an attack is detected, various data destruction mechanisms may be employed. The desirability of active data destruction by "intelligent" means is proposed, together with a discussion of alternative techniques with particular reference to the data storage device characteristics. These device characteristics (physical, electrical, electrostatic etc.) may well limit the practicality of reusable data storage and demand the use of a non-reversible destruction technique. Some potential problems of latent data retention in memory devices are explained with some alternative storage strategies.

Some experiences of tamper resistant research and development highlight the potential manufacturing problems - particularly in respect of quality assurance, product fault analysis and life-testing. Production engineering goals are identified both for current types of technology and future systems.

The desirability of tamper resistant standards and independent assessment facilities is expressed and a comprehensive check-list of design features is included which a prospective system builder may find useful when specifying requirements for cryptographic products. The applicability of such standards and large scale protection methods on intelligent tokens, in particular smart cards and personal authenticators, is discussed.

The presentation closes with a view of future trends in secure product design with particular reference to EFTPOS systems and distributed network security.

THE RSA CRYPTOGRAPHY PROCESSOR

The First Very High Speed One-Chip Solution

Holger Sedlak
Institut für Theoretische Informatik
Technische Universität Braunschweig
D-3300 Braunschweig
Federal Republic of Germany
(531) 391-2384

Extended Abstract

In commercial applications a minimum ciphering rate of 64 *Kbit/sec* is required which will be the transmission rate of public digital networks. In contrast, the RSA method has a very slow ciphering rate particular when using software implementations of the algorithm. The solution of this problem is a hardware implementation of the RSA algorithm. A cryptography processor, however, consisting of standard chips like bit slice processors again does not achieve the necessary speed. Moreover, in a multi-chip processor, the security of the key management system cannot be guaranteed. Therefore, a single-chip implementation of the RSA algorithm seems to be the only solution, for example the presented RSA Cryptography Processor.

The problem which now arises is the handling of very long numbers. If one wishes to cipher a 200 digit decimal number, then the length of each key is 660 bits. The RSA Cryptography Processor has to execute the following function

$$C = M^E \bmod N$$

C is the generated code, M is the data, and E and N together are the public encoding key. The same function is used for decoding, so the fast execution of this function is the heart of the problem. In this form, the function is not executable, but after some transformations the function is reduced to a sequence of additions and subtractions, i.e. inverse additions. In order to get a ciphering rate of 64 *Kbit/sec*, the RSA Cryptography Processor has to add two 660 bit numbers with a speed of 30 million operations per second. The time critical phase of each addition is the delay of the carry bit as long as the lowest bit position may influence the highest one.

Rather than using a normal arithmetic logic unit, a special "elementary" cell was designed which is only capable of calculating the RSA ciphering, i.e. high speed additions. A new carry look ahead logic (CLA) was designed reducing the time for carry bit calculation to a minimum. In this CLA, the addition is normally calculated in a single processor cycle. For long carry bit calculations, the CLA increases the calculation time for this addition to more than one cycle. The major advantage of this proceeding lies in the fact that the rate of addition is not determined by the longest lasting addition, but by the average adding time.

Furthermore, look ahead methods were developed which reduce the maximum number of additions. This reduction is achieved by auxiliary circuits operating in parallel with the actual addition circuits. The addition and the auxiliary circuits resulted in a relatively large elementary cell of 40 × 600 lambda.

These features enable the Ciphering Unit (CU) of the RSA Cryptography Processor to encode and decode data with a minimum rate of 64 *Kbit/sec*, even if, in the worst case, the keys have their maximum length of 660 bits. This rate of ciphering is about 10 000 times faster than any software solution of the RSA method. Compared with the hardware solution of Rivest [2], the CU is more than 50 times faster and the maximum length of the keys is about 30 percent higher. The acceleration of the CU is based on its specialized architecture. In contrast, the chip of Rivest has the same architecture as a general purpose arithmetic logic unit.

All the other RSA implementations known to the author suffer from one or more of the following restrictions: the ciphering rate is too low (e.g. the "Security Processor C.R.I.P.T." in [6] with a rate of less than 10 *Kbit/sec*); the key length is too low; or several chips are needed (e.g. the "NEC/Miyaguchi" design in [5] with a rate of 29 *Kbit/sec*, a key length of 660 bits and a solution of 333 chips).

In contrast to all of them is the RSA Cryptography Processor. The heart of it consists of two independent CUs with a size of 340 and 440 bits. For encoding, both CUs work together as one 780 bit sized CU, for decoding each of them work independently using the advantages of the Chinese Remainder Theorem. This results in an encoding rate of 3 *Mbit/sec* if using as exponent Fermat's 4th number and in a decoding rate of 0.2 *Mbit/sec* if the two primes do not exceed the size of the CUs.

Furthermore, the RSA Cryptography Processor has a shell around its kernel. So it is not only a very fast ciphering processor but a "Cryptograhic Area" which means that

- "Cryptographic Actions" like "Load a new key and accept it only if the certificate is correct" can only be started but not manipulated from outside the chip,
- signatures (certificates) are automatically generated respectively checked,
- secret memory is neither readable nor writeable from outside the chip, and
- it exists the possibility of generating new keys.

The hash function implemented to compute the signature is the one developed at SEPT [6]. This function has no additionally redundancy bits:

1. Divide M into m blocks B_i less than N
2. Perform
$$H_0 = 0,$$
$$H_i = (H_{i-1} + B_i)^2 + B_i \bmod N$$
3. H_m is the result.

The key generation is done by the Monte Carlo method using only the strong pseudoprime test. The two primes p and q were computed in the known way: $p-1$ has a large factor p', $p'-1$ has a large factor p'', and the same for q, respectively. But in addition to that caution, q is calculated such that

$$\gcd(N-1, E) = \gcd(p \times q - 1, E) = E = 2^{2^4} + 1.$$

This property prevents an illegal installation of a key $N_{Attacker}$ which is not certificated by the secret key $D_{Authority}$ but by the manipulated "secret" key

$$D'_{Attacker} \equiv E^{-1} \bmod (N_{Authority} - 1).$$

This attack is possible if $N_{Authority}$ is a pseudoprime to the base $H(N_{Attacker})$ and if gcd $(N_{Authority} - 1, E) = 1$. The possibility of the last condition is very high if the two primes are computed in the known way. The presented way of key generation prevents this attack.

For users, the following features of the RSA Cryptography Processor are of intersest:

- operation as a co-processor,
- three DMA channels for three independent data streams of 64 $Kbit/sec$,
- key generation takes an average time of 2 sec,
- 8 bit wide data bus,
- parallel operation of the CU and the I/O Unit,
- about 150 000 transistors in 1.5μ-CMOS technology, and
- chip size of approximately $5mm \times 4.8mm$.

So far, a prototype consisting of about 5000 5μ-NMOS transistors has been realized. This chip contains all major parts of the RSA Cryptography Processor in a small version. Furthermore, based on the experience with the 5μ-NMOS process, the performance rates of the 1.5μ-CMOS process were obtained by theoretical considerations and at full length simulations. The development of a 1.5μ-CMOS design of the RSA Cryptography Processor will soon be completed and therefore, first prototypes will be available for testing in fall 1987.

References

[1] Conway,L., Mead,C., Introduction to VLSI Systems, Addison-Wesley, 1980.

[2] Hellmann,M.E., Die Mathematik neuer Verschlüsselungssysteme, Spektrum der Wissenschaft 10, 1979, 92-101.

[3] Meyer,C.H., Matyas,S.M., Cryptography - A new Dimension in Computer Data Security, Cryptography Competence Center, IBM, Kingston, New York, Wiley-Interscience Publication, 1982.

[4] Pomerance,C., Recent Developments in Primality Testing, The Mathematical Intelligencer 3, 1981, 97-104.

[5] Rivest,R.L., RSA Chips (Past/Present/Future), in "Advances in Cryptology", Proc. Eurocrypt 84, Paris, France, 1984, 160-164.

[6] Pailles,J.C., Girault,M., The security processor C.R.I.P.T., Information Security: The Challenge, Preprints of the Fourth IFIP Security on Information Systems Security, Monte-Carlo, December 2-4, 1986.

[7] Sedlak,H., Konzept und Entwurf eines Public-Key-Code Kryptographie-Prozessors, Institut für Theoretische und Praktische Informatik, Technische Universität Braunschweig, 1985.

IV

PUBLIC KEY TOPICS

Extension of Brickell's Algorithm for Breaking High Density Knapsacks

F. Jorissen, J. Vandewalle, R. Govaerts
Katholieke Universiteit Leuven,
Department of Electrical Engineering, ESAT Laboratory
K. Mercierlaan 94, B-3030 Heverlee, Belgium

Extended Abstract

A knapsack (or subset-sum) problem that is useful for cryptographic purposes, consists of a set of n positive integers $a = \{a_1, a_2, \ldots, a_n\}$, called the knapsack a, and a sum s. The density d of a knapsack is defined to be $n/\log_2(a_i)_{max}$. The knapsack problem then consists of finding the set, if any, of binary numbers $x = \{x_1, x_2, \ldots, x_n\}$, such that $\Sigma x_i . a_i = s$.

In {2}, E.F.Brickell presented an algorithm for breaking knapsacks of low density. It was expected {2} that an improved version of it would solve most knapsacks of density less than .54. For knapsacks of increasing density, the probability of having so-called "small coefficient identities" (SCI's) in the knapsack also increases. The presence of these SCI's appears to be the reason for the failure of Brickell's algorithm for knapsack problems of high density.

In this paper, a simple technique is proposed for circumventing this problem effectively, so that the algorithm becomes capable of solving knapsacks of higher densities.

Tests have shown that the effectiveness of the algorithm can thus be increased to solve knapsacks of densities even > .9 with high probability and reasonable supplementary computer power.
Since the number of available knapsacks decreases very quickly with its density (cfr. fig. p.7), only a relatively small set of high density knapsacks remains safe for cryptographic purposes. Finding safe knapsacks may therefore become very hard.
Moreover, all the advantages of Brickell's algorithm are preserved. The most important of these is that the algorithm is applicable to any knapsack public-key cipher based on the knapsack problem mentioned above, whether the cipher already exists or is to be invented.

We shall now give a very brief overview of Brickell's algorithm and the improvements that have been accomplished. More details can be found in {2},{1}. Brickell made the basic observation that a knapsack problem $\Sigma x_i.a_i=s$ is in fact a linear equation in n binary unknown x_i and with non-zero sum s. He concluded that if n linear equations ($\Sigma x_i.y_{ji} = s_j(k)$) could be found such that:
- they are not linearly dependent of each other
- they all conform to the solution x of the knapsack problem
- that finding such set of equations is computationally feasible

that then each knapsack (problem) for which this is possible may be regarded as solved.

In the first part of his algorithm, a technique is described to construct a nxn matrix Y from which the rows y_j correspond to the coefficients of the n equations in the n unknown x_i. The first row y_1 of Y consists of the elements $b_i = a_i.W \bmod M$. The other rows of Y are derived by applying modular mappings with the small sum property on previous rows of Y.{2},{1} These modular mappings are calculated with the L.L.L.-algorithm {3}.

In the second part of his algorithm it is proven that $\Sigma x_i.b_i \in \{s', s'+M,\ldots ,s'+(n-1)M\}$, with $s' = s.W \bmod M$. Thus n integers $s'(k) = s'+k.M$ $(0 \le k \le n-1)$ are derived, exactly one of which corresponds to the correct binary solution x. Because all rows of Y, except for the first one, have been calculated by applying modular mappings with the small sum property on previous rows, we can now calculate {2},{1} from the set of possible sums of the first row s'(k), a new set of possible sums $s_j(k)$ for every other row j, with:
$s_j(k) = \Sigma x_i.y_{ji}$, assuming that $\Sigma x_i.b_i = s'(k)$.
(for exactly one constant value of k, $s_j(k)$ will be the sum of the new knapsack problem $\Sigma x_i.y_{ji} = s_j(k)$ with the same x as the original knapsack problem, and this for every row y_j of Y.)
For each of the values k we can thus compose a nx1 matrix S(k) and we can calculate the possible solutions x as nx1 matrices X from the matrix problems Y.X = S(k).
If indeed s is the sum of a subset of the a_i's then one of the vectors S(k) will give a correct <u>binary</u> solution X for $Y^{-1}.S(k)$.

If the first part of the algorithm can be completed then it is always straightforward to execute the second part and to solve the knapsack problem.

In experiments run by Brickell, his algorithm appeared to be always successful except when there were identities satisfied by the a_i's of the form:
$\Sigma \alpha_i.a_i = 0$, with $\Sigma |\alpha_i| \le n$.
Such an identity is called a "small coefficient identity" (SCI).

It appears that some "dangerous" types of these SCI's are easily "inherited" from one vector to another through the use of modular mappings, and in particular by modular mappings with the small sum property.{1} Brickell also demonstrated that the expected number of SCI's of a given knapsack increases with its density.

Through experiments it has come clear to us that the basic algorithm of Brickell fails for knapsacks of high densities because the given knapsack contains one or more SCI's. Since the first row y_1=b of Y, consisting of the numbers b_i, is derived from the numbers a_i through a modular mapping, some of the SCI's present in the knapsack a are inherited by the first row of Y. Since all the other rows of Y are analogously derived through modular mappings with the small sum property of the previous rows, it may happen that one or more of the dangerous SCI's of the initial knapsack are present in all the possible rows of Y. As a consequence of this, it becomes impossible to find a matrix Y with linearly independent rows and the algorithm fails.

In order to be able to still use Brickell's algorithm in these cases, we propose to construct the first row b of Y such that its expected number of SCI's is smaller than unity. Hereto we proved that the density of b is so small for practical dimensions n that it would normally not contain any SCI's if it was constructed randomly. It is also proven that, by construction, b can exclusively inherit SCI's present in a, for practical dimensions {1}. It may therefore be concluded that the SCI's of b are mainly introduced by the non-random character of b.
We therefore propose to construct b as $b_i = a_i.W \mod M + k_i.M$.
The numbers k_i can initially be regarded as relatively small random numbers. It can be proven that as a consequence of this $\Sigma x_i.b_i \in \{s', s'+M, \ldots, s'+(n-1+\Sigma k_i)\}$.
So it remains possible for this choice of b to determine $\Sigma x_i.b_i$, but $n+\Sigma k_i$ vectors S(k) have to be tested instead of n. It is therefore advantageous to keep Σk_i small, e.g. by restricting all the k_i's to $k_i \in [1,K]$, in which K is a small positive integer value. It has been mathematically and experimentally verified that this new technique has several advantages:
- b can still exclusively contain SCI's of the knapsack a.
- Therefore,b can only contain SCI's if at least one of the SCI's of a is still present in b, the probability of which decreases with increasing K.
- The average supplementary computer power needed, however, can be roughly estimated as the processing of (K-1).n/2 extra vectors S(k), and the implications of larger values of the b_i's, mainly on the execution time of the L.L.L.-algorithm.

These results have been verified experimentally for knapsacks of low dimensions n, in two ways.

1. Knapsack problems with knapsacks of low dimensions and high density have been tested.

For example the knapsack problems with knapsack:

- a = { 2 , 5 , 6 , 10 , 20 }, with - dimension n = 5
  - high density d = 1.16

could not be solved with the original algorithm of Brickell due to the inheritance of the SCI α = { 0 , 0 , 0 , 2 , -1 } from a to b and from there to all possible rows of the matrix Y.
This knapsack was however solved with the better choice of b.

2. Computer tests have been run on the effectiveness of both the original and the new version of Brickell's algorithm.

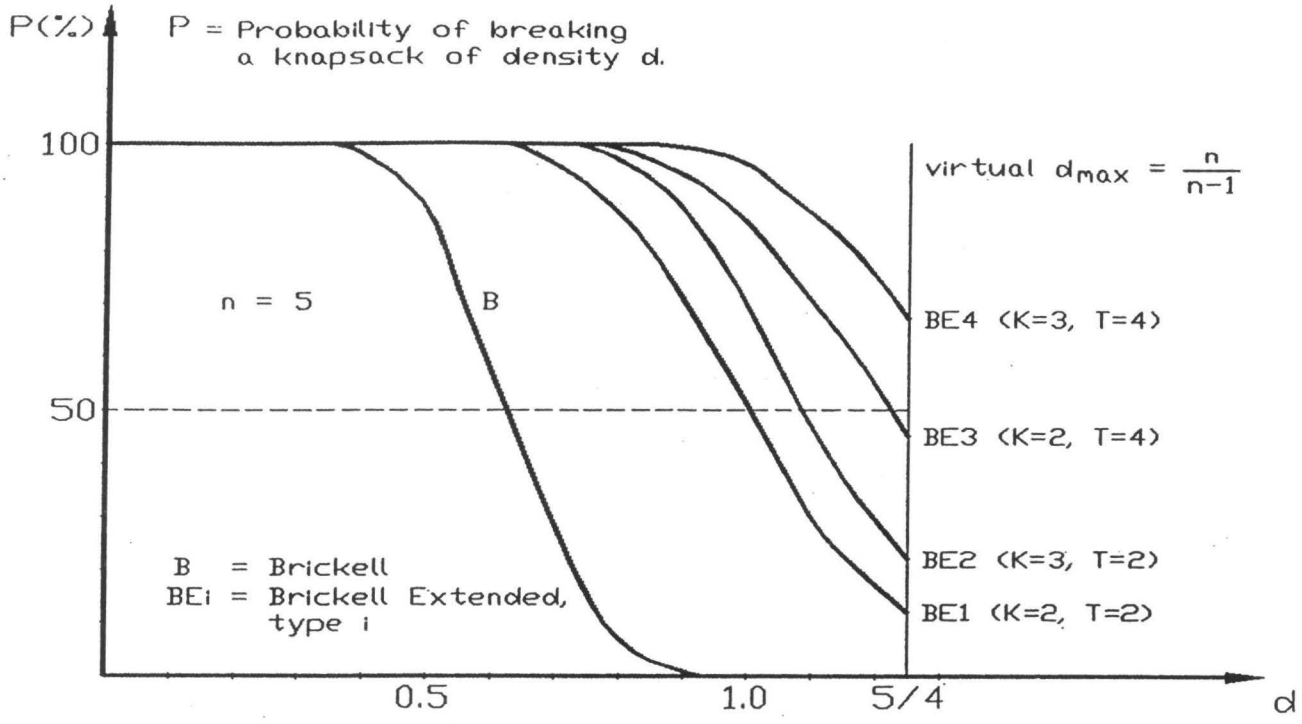
The purpose of these Monte-Carlo tests was to gain experimental results on the probability that b contains SCI's, for both algorithms. As parameters were chosen the dimension n and the density d, plus, for the new algorithm, the value of K and the number of trials T (i.e. the number T of vectors k that may be chosen for a given knapsack to try to find a SCI-free vector b for it). While considering the figures (p. 5), we must bear the following in mind:

- All figures represent dimension 5. Calculations were also made for dimension 8. For higher dimensions, the figures appear to remain flat till higher densities but then fall steeper.
- The figures give rather pessimistic results since:
  - Pessimistic approximations had to be made to calculate them.
  - It has been assumed that finding a SCI-free vector b is a necessary condition for solving a knapsack. To our experience however, a lot of knapsacks with b contaminated by SCI's could be solved,since only for dangerous SCI's the probability is high that they are inherited by all possible rows of Y.
  - In case only one SCI is transported through all possible rows of Y, a special countermeasure can be taken. {2}
  - The set of knapsacks that will remain unsolved will contain a relatively large fraction of "useless" knapsacks, since for increasing density, the probability also increases that knapsacks are not one-to-one.

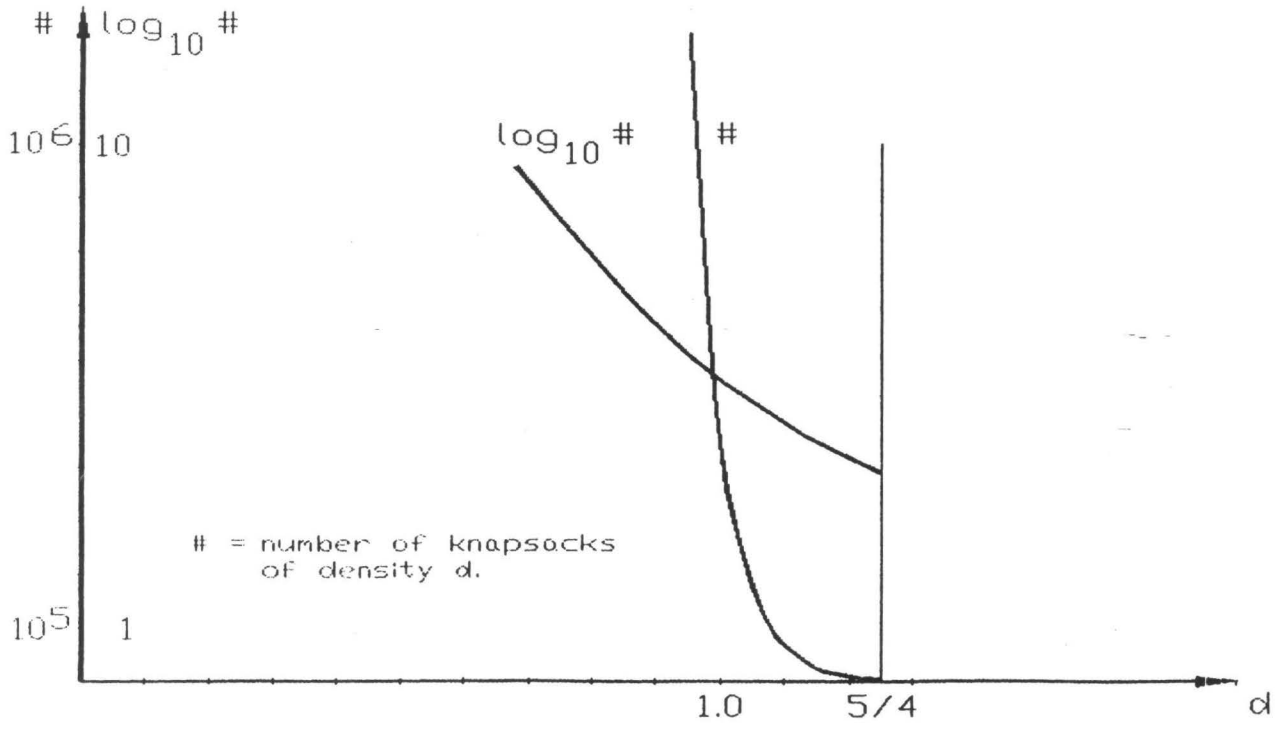
Conclusion

In our work, Brickell's algorithm {2} has been extended to the use of better vectors b. {1} This makes the algorithm capable of solving knapsacks of high densities even larger than .9, with reasonable supplementary computer power. We expect that this algorithm will reduce the set of safe knapsacks to such an extent that knapsack public-key ciphers may become impractical.

Figures



Test results of the extended algorithm of Brickell: Probability of breaking a knapsack, with both algorithms, and number of available knapsacks as functions of its density.



Acknowledgements

We are grateful to Y. Desmedt for many suggestions during the thesis work. We also thank S. Claeys for his help with the test programs. Last but not least, we wish to thank the company CRYPTECH for its support and stimulating interaction.

References

{1} F. Jorissen, "De cryptanalyse van knapzak publieke sleutel geheimschriftvormende algoritmes", M.Sc. Thesis, Katholieke Universiteit Leuven, Belgium, May 1985.

{2} E.F. Brickell, "Solving Low Density Knapsacks", Sandia National Laboratories, Albuquerque, USA.

{3} A.K. Lenstra, H.W. Lenstra Jr., and L. Lovasz, "Factoring Polynomials with Rational Coefficients", Mathematische Annalen, Vol.261, no.4, pp.515-534, 1982.

On Privacy Homomorphisms
(Extended Abstract)

Ernest F. Brickell
Yacov Yacobi

Bell Communications Research
435 South Street
Morristown, New Jersey 07960

Abstract

An additive privacy homomorphism is an encryption function in which the decryption of a sum (or possibly some other operation) of ciphers is the sum of the corresponding message. Rivest, Adleman, and Dertouzos have proposed four different additive privacy homomorphisms. In this paper, we show that two of them are insecure under a ciphertext only attack and the other two can be broken by a known plaintext attack. We propose an additive privacy homomorphism which we hope will either be secure or at least will be a challenge to break.

1. Introduction

A privacy homomorphism is an encryption function which allows the encrypted data to be operated on without knowledge of the decryption function. Privacy homomorphisms were introduced by Rivest, Adleman, and Dertouzos [RAD].

[RAD] mentioned that privacy homomorphisms could be used to establish data base systems in which an encrypted total of a list of items could be computed using only the encrypted values of the list of items. Privacy homomorphisms could also be used to establish a secure conference telephone call. In a typical conference call, a central facility adds together the signals of the speakers. If the signals were encrypted using an additive privacy homomorphism, then the central facility could "add" the signals together without decrypting them.

We must now give a more formal definition. A privacy homomorphism is a family of functions $(e_k, d_k, \alpha, \gamma)$ such that $d_k(\gamma(e_k(m_1), ..., e_k(m_r)) = \alpha(m_1, ..., m_r)$ for each k in key space $\mathbf{K}$ and any $m_1, ..., m_r$ in message space $\mathbf{M}$. The definition given in [RAD] is more general. It would include, for instance, an encryption function e_k in which the question "Is $m_1 > m_2$?" could be answered using only $e_k(m_1)$ and $e_k(m_2)$. In considering the security of these systems, we will assume that the cryptanalyst knows the functions e_k, d_k, α, γ but does not know the key k.

One privacy homomorphism mentioned in [RAD] is based on the multiplicative property of the RSA encryption function [RSA]. Let $n = pq$ where p and q are large primes. Let $\alpha(m_1, ..., m_r) = m_1 \cdots m_r \bmod n$ and $\gamma = \alpha$. Define e and d in the usual manner for RSA. This privacy homomorphism is as secure as RSA.

There are four more privacy homomorphisms mentioned in [RAD]. There are weaknesses in each of these which will be discussed in section 2.

We still say that $(e_k, d_k, \alpha, \gamma)$ is an R-additive privacy homomorphism if $\mathbf{M}$ is the set of positive integers less than or equal to some bound M, $\alpha(m_1, ..., m_r) = m_1 + \cdots + m_r$, and $d_k(\gamma(e_k(m_1), ..., e_k(m_r)) = \alpha(m_1, ..., m_r)$ only for $r \leq R$.

For the remainder of this paper we will consider the problem of finding R-additive privacy homomorphisms. It would be interesting to know whether such a function exists that is provably secure in the sense that cryptanalysis can be shown equivalent to some well studied problem. It is also desirable to limit the data expansion.

In section 3, we will present a function that is an R-additive privacy homomorphism. In section 4, we will show that if we weaken this system in various ways, then it can be cryptanalyzed.

2. Cryptanalysis of [RAD] privacy homomorphisms.

We present cryptanalysis of four privacy homomorphism systems which appear in [RAD] (examples 1,5,3,4 in this order).

System 1

Let g be a generator modulo a prime p, where $p - 1 = \prod_{i=1}^{k} p_i^{d_i}$, and for all i, $p_i \leq B$, for some small B. Let q be a large prime and let $n = p \cdot q$. Encryption of message M is done as follows: $C \equiv g^M \bmod n$. Decryption is $M \equiv \log_g C \bmod p$. The structure of p enables computation of the discrete logarithm by the method of Pohlig and Hellman [PH] in time $O(B^{1/2})$. This is a privacy homomorphism with α being addition mod $p - 1$ and γ multiplication mod n.

Cryptanalysis

Let $d \geq \max\{d_i | 1 \leq i \leq k\}$.

Lemma 1. $p \mid gcd(a^{B!^d}-1,\ n)$.

Proof. $a^{p-1} \equiv 1 \bmod p$, hence $a^{\prod_{i=1}^{k} p_i^{d_i}} \equiv 1 \bmod p$. Since $\prod_{i=1}^{k} p_i^{d_i} \mid B!^d$; $a^{B!^d} \equiv a^{\prod_{i=1}^{k} p_i^{d_i}} \equiv 1 \bmod p$. □

There appears to be potential problems with this attack. The cryptanalyst does not know the values of B or d. Also, $gcd(a^{B!^d}-1,\ n)$ could be n instead of p. However, in many cases, the cryptanalyst will make a good choice for B and d and the gcd will be p. For those cases when the gcd is 1, the cryptanalyst will just choose a larger B and d, and if the gcd is n, he will choose a smaller B and d. This is very similar to the Pollard $p-1$ factoring algorithm [P].

The expected running time for this cryptanalysis is $O(dB\log B\log n)$.

System 2

This is an additive privacy homomorphism. Let k be chosen so that any intermediate result is $<2^k$, and let $a_0, a_1, \ldots, a_{k-1}$ be randomly chosen positive integers. Let $x_0, x_1, \ldots, x_{k-1}$ be the binary representation of integer x, and let $f(z) = \sum_{i=0}^{k-1} x_i \cdot z^i$. Encryption of x is the k-tuple $y = (f_x(a_0), \ldots, f_x(a_{k-1}))$. Operations are done componentwise. Decryption is preformed by solving a set of k linear equations in k variables.

Cryptanalysis

Suppose j is the largest index such that $x_j = 1$, i.e., $x_{j+1}, \ldots, x_{k-1} = 0$. Let z be any positive integer. Then $z^j \le f_x(z) \le \sum_{i=0}^{j} z^i < (z+1)^j$. Thus $\lfloor f_x(z)^{1/j} \rfloor = z$. This leads to an obvious ciphertext only attack. Given a cipher $y = (y_0, \ldots, y_{k-1})$, guess a value for j, the largest index such that $x_j = 1$. Compute $\lfloor y_0^{1/j} \rfloor = b_0$. Write y_0 in base b_0 notation. If all the coefficients are 0 and 1, then probably $b_0 = a_0$ and x is easily found. If not try a different choice for j. The values of $y_1, \ldots, y_{k-1}$ can be used as an additional check.

System 3

Let p and q be large primes. Encryption of message M is an ordered pair $(M \bmod p, M \bmod q)$. The bridge can perform addition, subtraction or multiplication componentwise, without knowing the moduli p and q. Decryption is done using the Chinese remainder theorem.

Cryptanalysis

We assume known plaintext attack. Let $C_i \equiv M_i \bmod p$, $i = 1, 2, 3, \ldots, r$. $C_1 \cdot C_2 \equiv C_2 M_1 \equiv C_1 \cdot M_2 \bmod p$. Therefore $p \mid C_1 \cdot M_2 - C_2 \cdot M_1$. Taking many pairs, p most probably equals the *gcd* of their corresponding differences $C_i \cdot M_j - C_j \cdot M_i$. Cryptanalysis for q is likewise.

System 4

In this system encryption is just writing the message in a secret radix system. Referring to the least significant digit, this reduces to the cryptanalysis of the previous system.

3. An R-additive privacy homomorphism.

In this section we present an R-additive privacy homomorphism for which we have not found an attack.

Let the message space $\mathbf{M}$ be the integers between 0 and M. Let $N = R\,M$. Let p be a prime larger than N. (The first prime larger than N will be satisfactory.) Let A be an ℓ by ℓ matrix chosen randomly such that A is nonsingular $\bmod p$. Let q be an integer $> Rp$. Let z be an integer randomly chosen in the interval $[Rpq^{\ell-1}, 2Rpq^{\ell-1}]$, and let y be chosen relatively prime to z.

A block of ℓ messages $m_1, \ldots, m_\ell$ will be encrypted by taking the vector $\overline{m} = (m_1, \ldots, m_\ell)$ and forming $\overline{s} = A\,\overline{m} \bmod p$. Let $\overline{s} = (s_1, \ldots, s_\ell)$. Compute $t = \sum_{i=0}^{\ell-1} s_i\, q^i$. Finally set $c = ty \bmod z$.

There is one step in this process that we must elaborate on. Let f be a pseudo random function from $\mathbf{M}^\ell \rightarrow \{0, 1\}$. When forming $\overline{s} = A\,\overline{m} \bmod p$, if $s_i \not\equiv 0 \bmod p$, then let s_i be the smallest nonnegative residue mod p. However if $s_i \equiv 0 \bmod p$, we want to randomly set s_i to be either 0 or p. We do this using the function f, letting $s_i = p$ if $f(m_1, \ldots, m_\ell) = 1$ and $s_i = 0$ otherwise. This is to avoid a chosen plaintext attack on this system. We will discuss this further in the next section.

The decryption process is obvious. Given a cipher c form $t = cy^{-1} \bmod z$. Write t base q to obtain $\overline{s} = (s_1, \ldots, s_\ell)$. Then $A^{-1}\overline{s} \bmod p$ gives the desired message $\overline{m}$.

Showing that this is an R-additive privacy homomorphism is straightforward as well. Suppose $\overline{m}_1, \ldots, \overline{m}_r$ are r messages for $r \leq R$. Let $\overline{s}_j = (s_{j,1}, \ldots, s_{j,\ell})$, t_j, and c_j be the values computed in the encryption of $\overline{m}_j$. We need to show that when $c = c_1 + \cdots + c_r$ is decrypted the result is $\overline{m} = (m_{1,1} + \cdots + m_{r,1}, \ldots, m_{1,\ell} + \cdots + m_{r,\ell})$.

Clearly $t = cy^{-1} \equiv t_1 + \cdots + t_r \bmod z$. But $\sum_{j=1}^{r} t_j = \sum_{j=1}^{r} \sum_{i=0}^{\ell-1} s_{j,i}\, q^i = \sum_{i=0}^{\ell-1} \left(\sum_{j=1}^{r} s_{j,i} \right) q^i$. Also $\sum_{j=1}^{r} s_{j,i} < q$, for $0 \leq i \leq \ell - 1$. So $\sum_{j=1}^{r} t_j < z$ and writing t base q gives the values $\sum_{j=1}^{r} s_{j,i}$, for $1 \leq i \leq \ell$. Let $\overline{s} = \left(\sum_{j=1}^{r} s_{j,1}, \ldots, \sum_{j=1}^{r} s_{j,\ell} \right)$. Then $\overline{s}\,A^{-1} \equiv \overline{m}_1 + \cdots + \overline{m}_r \bmod p$. But since $m_{1,i} + \cdots + m_{r,i} < p$, for $0 \leq i \leq \ell - 1$, the result follows.

4. Possible cryptanalysis of this privacy homomorphism.

Since this encryption system uses modular multiplication, it is immediately suspect to the attacks that broke the knapsack cryptosystems [B,S]. These attacks, based on the Lovasz algorithm [LLL], do seem to give some additional information about the secret key for $R > 2$. For $\ell = 1$ and 2, it is clear that this information can be used to break the system. For larger ℓ, say $\ell \geq 30$, it is not known whether this information can be used. The details of this attack and the chosen plaintext attack referred to in section 3 will be discussed in the full paper.

5. Open problems.

We have introduced an R-additive privacy homomorphism for which we can make no convincing arguments about its security. Is this scheme secure? It would also be interesting to find other privacy homomorphisms.

References

[B] E. F. Brickell, "Breaking Iterated Knapsacks", *Advances in Cryptology, Proc. Crypto 84,* Santa Barbara, August 19-22, 1984, Lecture Notes in Computer Science, vol. 196, Springer-Verlag, Berlin, 1985, pp. 342-358.

[LLL] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovasz, "Factoring Polynomials with Rational Coefficients", *Mathematische Annalen 261*, pp. 515-534, 1982.

[PH] Stephen C. Pohlig and Martin E. Hellman, "An Improved algorithm for computing Logarithms over $GF(p)$ and its cryptographic significance", IEEE Trans. on Inf. Th. Vol. IT-24, No. 1 Jan. 1978. pp. 106-110.

[P] J. M. Pollard, "Theorems on factorization and primality testing", Proc. Cambridge Philos. Soc. vol. 76 (1974), pp. 521-528.

[RAD] Ronald L. Rivest, Len Adleman and Michael L. Dertouzos, "On data banks and privacy homomorphisms", in Foundation of Secure Computations, Academic Press 1978.

[RSA] R. L. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Commun. ACM,* vol. 21, pp. 294-299, April 1978.

[S] A. Shamir, "A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem", IEEE Trans. Inform. Theory, vol. IT-30, No. 5, September 1984, pp. 699-704.

AN IMPROVED PROTOCOL FOR DEMONSTRATING POSSESSION OF A DISCRETE LOGARITHM AND SOME GENERALIZATIONS

(Extended Abstract)

David Chaum

Jeroen van de Graaf

Centre for Mathematics and Computer Science
Kruislaan 413; 1098 SJ Amsterdam; The Netherlands.

Abstract:

An improved protocol is presented that allows A to convince B that she knows a solution to the Discrete Log Problem—i.e. that she knows an x such that $\alpha^x \equiv \beta \pmod N$ holds—without revealing anything about x to B. Protocols are given both for N prime and for N composite.

We also give protocols for extensions of the Discrete Log problem:

- Showing possession of multiple discrete logarithms to the same base at the same time, i.e. knowing $x_1, \ldots, x_K$ such that $\alpha^{x_1} \equiv \beta_1, \ldots, \alpha^{x_K} \equiv \beta_K$.
- Showing possession of several discrete logarithms to different bases at the same time, i.e. knowing $x_1, \ldots, x_K$ such that the product $\alpha_1^{x_1}\alpha_2^{x_2} \cdots \alpha_K^{x_K} \equiv \beta$.
- Showing possession of a discrete logarithm that is the simultaneous solution of two (or more) different instances, i.e. knowing x such that $\alpha_1^x \equiv \beta_1$ *and* $\alpha_2^x \equiv \beta_2$.

We can prove that the sequential versions of these protocols are zero knowledge. provided that A if she knows the order of α, or a multiple thereof. By changing the protocols slightly, the parallel versions of these protocols are reducible to solving Discrete Log.

The protocols presented in this paper can be applied in any group in which the group operation can be computed by both parties. An interesting new candidate group might be the set of points of an elliptic curve over a Galois field.

1. Introduction

Consider the following problem:

- Alice knows a solution to the **Discrete Log (DL)** problem: for a particular α, β and N, she knows the exponent x such

that $\alpha^x \equiv \beta \pmod N$ holds.

- Alice wants to convince Bob that she knows x.
- Alice is not willing to reveal the value of x.
- Bob accepts an exponentially small chance that Alice is cheating, i.e. that she pretends to know an x but doesn't. More precisely, the chance that Alice succeeds in cheating without being detected by Bob, is 2^{-T}, where T is proportional to the time and space required.

This paper presents a protocol which solves this problem, both for the case N a prime, and for the case $N = P_1 P_2$, where P_1 and P_2 are prime and are of roughly the same size. Notice that there is no probabilistic polynomial time algorithm known for finding x given α, β and N (for N composite we have to assume that factoring N is hard). When Alice is restricted to polynomial computational power this protocol is still of interest, since given α and N she can choose $x \in \{1, \ldots, N-1\}$ at random and then compute β simply by exponentiation (or a third party could supply Alice with x and β).

In [CEGP86] protocols were presented that solve the same problem. Compared to those protocols, the protocol presented here is perhaps easier to understand, to use, and to generalize. The existence of a protocol with the same functionality is implied by results of [GMW86], [BrCr86] and [Ch86]. However, these protocols are not very useful in practice. In [Ch87] efficient protocols that solve this problem are needed; this was the major motivation for our research.

We also present protocols for proving possession of a solution to some generalisations of the Discrete Log problem:

1 **Multiple Discrete Log (MDL):**

A shows to B that, given α and $\beta_1, \ldots, \beta_K$, she knows $x_1, \ldots, x_K$ such that $\alpha^{x_1} \equiv \beta_1, \ldots, \alpha^{x_K} \equiv \beta_K$. This protocol is more efficient than applying the basic DL-protocol for the pairs $(x_1, \beta_1), \ldots, (x_K, \beta_K)$ retaining the same probability of catching a cheating A. When a third party creates the x_i's at random and supplies Alice with the x_i's and β_i's, this protocol also offers the possibility to use DL as the basis for an authentication scheme in a way similar to Fiat & Shamir [FiSh86], whose scheme is based on the hardness of factoring.

2 **Relaxed Discrete Log (RDL):**

A shows to B that, given $\alpha_1, \ldots, \alpha_K$ and β, she knows $x_1, \ldots, x_K$ such that $\alpha_1^{x_1} \alpha_2^{x_2} \cdots \alpha_K^{x_K} \equiv \beta$.

3 **Simultaneous Discrete Log (SDL):**

A shows to B that, given $\alpha_1, \ldots, \alpha_K$ and $\beta_1, \ldots, \beta_K$, she knows x such that $\alpha_1^x \equiv \beta_1$ and $\alpha_2^x \equiv \beta_2$ and $\cdots$ and $\alpha_K^x = \beta_K$. Note that the α_i's can be in different groups.

Above the Discrete Log problem is stated in the group Z_N^*, the multiplicative group modulo N, with N prime or composite. However, in any group we can state a Discrete Log problem: let G be a group, $\alpha \in G$, $\langle \alpha \rangle$ the group generated by α, and $\beta \in \langle \alpha \rangle$; then find x such that $\alpha^x = \beta$ in G. The protocols presented in this paper are feasible in any group G, provided that A and B can apply the group operation in polynomial time and that A knows (a multiple of) the order of α in G. (For the RDL-protocol we also have to assume that G is Abelian.) With a slight modification the protocols are still feasible when A does not know a multiple of the order of α in G, but then the protocols leak information about x.

Of course, these protocols make sense only if no efficient algorithm for computing the Discrete Log in G exists. Apart from the case $G = Z_P^*$, with P prime, and the case $G = Z_N^*$, with N composite, we can also take $G = E(P)$, the set of points of an of elliptic curves over $GF(P)$, imposed with the usual group structure. (Discrete Log in $E(P)$ may not be computable in random polynomial time [Mi85][Ka86].)

For describing the protocols, we use the same protocol notation throughout the paper. The meaning of this notation is straightforward; only the next few things might need explanation:

— T is the **security parameter**, agreed upon before the protocol starts. Increasing T reduces A's chance of successfully cheating exponentially, but increases the amount of communication and computation only linearly.

— In the zeroth step of the protocol, A and B agree on α, β and N.

— Expressions shown on the left or right are known to that party only, and are secret from the other party.

— Expressions between square brackets indicate that the receiver of the message only gets to know the result of the computation. If not indicated otherwise, these expressions are to be taken (mod N).

— $e \in_R S$ means that an element e is chosen at random from the set S, where all elements of S have an equal probability of being chosen, independent of all previous events.

— In some steps of the protocol a party checks if a particular equality holds; this is denoted as: check $a \stackrel{?}{=} b$. If the check fails, cheating is detected and the protocol halts.

2. The basic protocol: Discrete Log

Instance: N, $\alpha \in Z_N^*$, $\beta \in \langle\alpha\rangle$

Solution: x such that $\alpha^x \equiv \beta \pmod N$

Note that knowing a discrete logarithm x is indeed a secret because of the following two assumptions:

Assumption 1: If N is a prime number, then there is no random-polynomial time algorithm for finding the solution to all instances of the Discrete Log problem.

Assumption 2: If N is composite and hard to factor, then there is no random-polynomial time algorithm for finding the solution to all instances of the Discrete Log problem.

The first assumption is well-known; the second follows from the fact that factoring integers is hard: if such an algorithm for finding x would exist, it could easily be used to find multiples of $\phi(N)$, and this is random-polynomial time reducible to factoring [De84]. Though the cases N prime versus N composite differ, we develop the protocols simultaneously and point out the differences. If N is composite, we assume that A knows its factorisation.

Protocol 1: Discrete Log: $\alpha^x \equiv \beta \pmod N$

	A		***B***
Step 0:		α, β, N	
	$<$	———	$>$
Repeat T times:			
	$r \in_R \{1, \ldots, \phi(N)\}$		
Step 1:		$[\alpha^r]$	
		——— $>$	
			$b \in_R \{0, 1\}$

Step 2: b

$\leftarrow$

Step 3: $[(r + bx) \bmod \phi(N)]$

$\rightarrow$

check $\alpha^{[(r+bx)]} \stackrel{?}{=} [\alpha^r]\beta^b$

Theorem 1.

(a) A can cheat in protocol 1 with probability at most 2^{-T} if she does not know x, and

(b) there exists a polynomial-time prover simulator for A.

Proof:

(a) Correctness: If A does not know x, then she is not able answer both the case $b = 0$ and $b = 1$. Hence she will get caught with probability at least ½ in each round. Thus A will get caught cheating with probability at least $1 - 2^{-T}$.

(b) Security: (in the final paper) □

3. Generalisation 1: Multiple Discrete Log

Instance: $N, \alpha \in Z_N^*, \beta_1, \ldots, \beta_K \in \langle\alpha\rangle$

Solution: $x_1, \ldots, x_K$ such that $\alpha^{x_1} \equiv \beta_1 \pmod N$, $\alpha^{x_2} \equiv \beta_2 \pmod N$, $\ldots$, $\alpha^{x_K} \equiv \beta_K \pmod N$

Protocol 2: Multiple Discrete Log: $\alpha^{x_1} \equiv \beta_1 \pmod N$, $\alpha^{x_2} \equiv \beta_2 \pmod N$, $\ldots$, $\alpha^{x_K} \equiv \beta_K \pmod N$

A ***B***

Step 0: $\alpha, \beta_1, \ldots, \beta_K, N$

$\leftrightarrow$

Repeat T times:

$r \in_R \{1, \ldots, \phi(N)\}$

Step 1: $[\alpha^r]$

$\rightarrow$

$b_i \in_R \{0, 1\}$,

Step 2: $b_1, \ldots, b_K$

$\leftarrow$

Step 3: $[(r + b_1x_1 + \cdots + b_Kx_K) \bmod \phi(N)]$

$\rightarrow$

check $\alpha^{[r+b_1x_1+\cdots+b_Kx_K]} \stackrel{?}{=} [\alpha^r]\beta_1^{b_1} \cdots \beta_K^{b_K}$

Lemma: Suppose that A does not know the discrete logarithm (with respect to α) of $\gamma\beta_1^{b_1}, \ldots, \beta_K^{b_K}$ for at least one vector

$\vec{b} = (b_1, \ldots, b_K) \in \{0, 1\}^K$. Then she doesn't know the discrete log for at least half the vectors $\vec{b} \in \{0, 1\}^K$.

Proof: We proceed by induction on K. For $K = 1$ the lemma is trivial. Suppose the lemma is true for $K = L - 1$, where $L = 2>$ (induction hypothesis). We shall prove the lemma for $K = L$.

Suppose that A knows the discrete logarithms of all the products $\gamma\beta_1^{b_1}, \ldots, \beta_{L-1}^{b_{L-1}}$ with $b_1, \ldots, b_{L-1} \in \{0, 1\}$. Then she does not know the discrete logarithm of β_L. Hence she cannot form the discrete logarithms of the products $\gamma\beta_1^{b_1}, \ldots, \beta_{L-1}^{b_{L-1}}\beta_L^{b_L}$ with $b_L = 1$. This implies our lemma. If we suppose instead that A knows the discrete logarithms of all the products $\gamma\beta_1^{b_1}, \ldots, \beta_{L-1}^{b_{L-1}}\beta_L$ then the lemma can be proved in the same way.

Now suppose that A does not know the discrete logarithm of at least one of the products $\gamma\beta_1^{b_1}, \ldots, \beta_{L-1}^{b_{L-1}}$ with $b_1, \ldots, b_{L-1} \in \{0, 1\}$ and also not the discrete logarithm of at least one of the products $\gamma\beta_1^{b_1}, \ldots, \beta_{L-1}^{b_{L-1}}\beta_L$ with $b_1, \ldots, b_{L-1} \in \{0, 1\}$. Then by the induction hypothesis she does not know the discrete logarithm of at least half of the products $\gamma\beta_1^{b_1}, \ldots, \beta_{L-1}^{b_{L-1}}$ with $b_1, \ldots, b_{L-1} \in \{0, 1\}$, and also, by the induction hypothesis with $\gamma\beta_L$ instead of γ, she does not know the discrete logarithm of at least half the products $\gamma\beta_1^{b_1}, \ldots, \beta_{L-1}^{b_{L-1}}\beta_{L-1}$ with $b_1, \ldots, b_L \in \{0, 1\}$. This completes the proof of our lemma. □

Theorem 2.

(a) A can cheat in protocol 2 with probability at most 2^{-T} if she does not know every x_i, and

(b) there exists a polynomial-time prover simulator for A.

Proof sketch

(a) Correctness: Assume Alice knows all x_i except one: x_1 (possibly after renumbering). Now Alice passes the protocol only if she guesses b_1 correctly in each round of the protocol. This happens with probability 2^{-T}.

If we assume that A doesn't know x_1 and x_2, then she still might know $[x_1 + x_2]$ if she did the following: 1) choose y at random and compute $\gamma = \alpha^y$. 2) choose β_1 at random. 3) compute $\beta_2 = \gamma\beta_1^{-1}$. Then A succeeds in cheating if $b_1 = b_2$ in each round, which happens also with probability 2^{-T}.

In general, if A doesn't know at least one x_i then for each K she passes the protocol with probability ½ in each round because of the lemma.

(b) Security: (in the final paper) □

If we assume that not A, but another party (B, or some mutually trusted party) generates the x_i at random and supplies the x_i's and β_i's to B, protocol 2 can be used as an interactive identification scheme as introduced by Fiat and Shamir [FiSh86]. In that scheme possession of several square roots modulo a composite (instead of several discrete logarithms) has to be shown. When compared, our scheme used with a prime has the advantage that there is no trapdoor information to be kept secret. However, this trapdoor enables putting relevant information in the value for which possession of the square root has to be shown. Moreover, the scheme of Fiat and Shamir is far more efficient, because it needs only squaring where we need a full exponentiation of a *log* N-bit number.

4. Generalisation 2: Relaxed Discrete Log

Instance: $N, \alpha_1, \alpha_2, \ldots, \alpha_K \in Z_N^*, \beta \in Z_N^*$

Solution: $x_1, \ldots, x_K$ such that $\alpha_1^{x_1}\alpha_2^{x_2} \cdots \alpha_K^{x_K} \equiv \beta$ Finding the solution of a RDL problem is hard:

Theorem 3. Under assumptions 1 and 2, there exists no random-polynomial time algorithm for finding the solution of all instances of the Relaxed Discrete Log problem, neither for N prime, nor N composite.

Protocol 3: Relaxed Discrete Log: $\alpha_1^{x_1}\alpha_2^{x_2} \cdots \alpha_K^{x_K} \equiv \beta \pmod N$

	A		***B***
Step 0:		$\alpha_1, \alpha_2, \ldots, \alpha_K, \beta, N$	
		$\longleftrightarrow$	
Repeat T times:			
	$r_1, \ldots, r_K \in_R \{1, \ldots, \phi(N)\}$		
Step 1:		$[\alpha_1^{r_1}], \ldots, [\alpha_K^{r_K}]$	
		$\longrightarrow$	
			$b \in_R \{0, 1\}$
Step 2:		b	
		$\longleftarrow$	
Step 3:		$[(r_1 + bx_1) \bmod \phi(N)], \ldots, [(r_K + bx_K) \bmod \phi(N)]$	
		$\longrightarrow$	
			check $\alpha_1^{[r_1+bx_1]}\alpha_2^{[r_2+bx_2]} \cdots \alpha_K^{[r_K+bx_K]} \equiv$ $\equiv [\alpha_1^{r_1}], \ldots, [\alpha_K^{r_K}]\beta^b$

Theorem 4.

(a) A can cheat in protocol 3 with probability at most 2^{-T} if she does not know $\vec{x} := (x_1, \ldots, x_K)$, and

(b) there exists a polynomial-time prover simulator for A.

5. Generalisation 3: Simultaneous Discrete Log

Instance: $N, \alpha_1, \alpha_2, \beta_1, \beta_2 \in Z_N^*$

Solution: x such that $\alpha_1^x \equiv \beta_1$ and $\alpha_2^x \equiv \beta_2$

(Protocol and theorem in the final version)

Acknowledgements

Jan-Hendrik Evertse made several valuable suggestions for this paper. Also discussions with René Peralta and Bert den Boer contributed to the results.. We would like to thank them for their remarks and suggestions.

References

[BrCr86] G. Brassard, and C. Crépeau, "Zero-Knowledge Simulation of Boolean Circuits," *Presented at Crypto 86*, (August 1986).

[Ch86] D. Chaum, "Demonstrating that a Public Predicate can be Satisfied Without Revealing Any Information About How," *Presented at Crypto 86*, (August 1986).

[Ch87] D. Chaum, "Blinding for unanticipated signatures," *Submitted to Eurocrypt 87*, (January 1987).

[CEGP86] D. Chaum, J.-H. Evertse, J. van de Graaf, and R. Peralta, "Demonstrating possession of a discrete logarithm without revealing it," *To appear in the Proceedings of Crypto 86*, (August 1986).

[De84] J.M. DeLaurentis, "A further weakness in the common modulus protocol for the RSA cryptoalgorithm," *Cryptologia 8 (3)*, July 1984, 253-259.

[FiSh86] A. Fiat, and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," *Presented at Crypto 86*, (August 1986).

[GMW86] O. Goldreich, S. Micali, and A. Wigderson, "How to Prove all NP-statements in Zero-Knowledge, and a Methodology of Cryptographic Protocol Design," *Presented at Crypto 86*, (August 1986).

[Ka86] B. Kaliski, jr., "A Pseudo-Random Bit Generator Based on Elliptic Logarithms," *Presented at Crypto 86*, (August 1986).

[Mi85] V. Miller, "Elliptic curves and cryptography," *Proceedings of CRYPTO 85*, (1985).

A Public Key Analog Cryptosystem

George I. Davida
Gilbert G. Walter

University of Wisconsin-Milwaukee
Milwaukee, WI 53201

The research reported in this paper was supported in part by NSF grant DCR-8504620.

ABSTRACT

In this abstract we present a public key cryptosystem based on error correcting codes [1, 7, 15]. The new public key system is obtained by extending the public key cryptosystem of McEliece [6, 12].

In this scheme a message M, consisting of a column vector of @k@ elements from a finite field is first scrambled by multiplying it by a non singular matrix @Q@ to get

$$M' = Q\ M$$

This scrambled message has parity check variables added to it, by multiplying it by a generator matrix G and then has all the variables reordered by multiplication by a permutation matrix P. Noise is then added to obtain the encrypted message

$$C = P\ G\ Q\ M + Z \tag{1}$$

The product of the three matrices $G' = P\ G\ Q$ is made public, but the factors are not.

The analogs of these digital operations are integral transforms, while the column vectors are functions which we take to be square integrable on (0,1) or (0,2) [2-5, 11, 13, 14, 18]. The message M will now be denoted by $x = x(t)$, and $x' = Q\ x$ will mean

$$x'(t) = \int_0^1 q(t,s)\, x(s)\, ds \tag{2}$$

where $q(t,s)$ is the kernel of the transformation. P will be a similar operator except that it must be an orthogonal operator to avoid changing the magnitude of the noise. This noise will consist of a realization of a random process $z(t)$ on (0,2). The operator G will have a special form to be discussed below.

In the McEliece digital scheme, the matrix $G = \begin{bmatrix} I \\ -A \end{bmatrix}$ where the submatrix @A@ introduces the new variables [12]. Decryption is accomplished by multiplying both sides of

$$P^T\ C = G\ Q\ M + P^T\ Z \tag{3}$$

by a matrix $H = [A\ I]$. This annihilates the term $G\ Q\ M$ and leaves the syndrome $S = H\ P^T\ C$ on the left. Coding Theory is used to estimate P^{TZ} which in turn is used with (3) to estimate $G\ Q\ M$. An estimate for M then is obtained by a projection operator followed by Q^{-1}.

In the analog scheme G is based on a kernel $k(t,s)$, $t,s \in (0,1)$ and is given by

$$(Gx)(t) = \int_0^1 \{ \delta(t-s) - k(t-1,s) \} x(s)\, ds, \quad 0<t<2$$

and H is given by

$$(Hy)(t) = \int_0^1 k(t,s)\, y(s)\, ds + \int_1^2 \delta(t+1-s)\, y(s)\, ds, \quad 0<t<1.$$

Clearly both are continuos operators and $H\ G = 0$. The analog encryption consists of first operating with Q, then G, then P on x and then adding noise, i.e. the ciphertext is

$$y = P\ G\ Q\ x + Z \tag{4}$$

where $y = y(t)$ is in $L^2(0,2)$. The kernel $q(t,s)$ of the operator Q and $k(t,s)$ may be taken to be Green's functions of appropriate differential operators. The kernel $p(t,s)$ of P may be taken to be

$$p(t,s) = \sum_n \phi_{i(n)}(t)\, \phi_n(s)$$

where $\{\phi_n\}$ is a complete orthonormal system in $L^2(0,2)$, and $i(n)$ is a permutation of the integers.

Decryption begins by operating with H on

$$P^{-1} y = G\ Q\ x + P^{-1} Z$$

to obtain the syndrome $S = HP^{-1}y$. The value of $P^{-1}Z$ of minimum error which satisfies

$$H\ Z' = H\ P^{-1}Z = S$$

is then found. This can be done by using the generalized inverse

$$\hat{Z}' = H^*\ (H\ H^*)^{-1} S$$

where H^* is the adjoint operator. This then is used to estimate $P^{-1}Z$ and $P^{-1}y - \hat{Z}'$ restricted to (0,1) is used to estimate Qx which is then inverted to obtain the estimate of the message.

The final estimate cannot be made completely noise free as in the digital case but the noise can be reduced by this method. Also the three operators P, G and Q can be combined in a single integral transform which can then serve as a public key.

References

[1] Berlekamp, E., *Algebraic Coding Theory*, McGraw Hill, New York (1968).

[2] Blum, J., Susarla, V., and Walter, G., "Estimation of the Prior Distribution Using Differential Operators," *Colloquia Mathematica Societatis*, (1980).

[3] Coddington, E.A. and Levinson, N., *Theory of Ordinary Differential Equations*, McGraw Hill, New York (1955).

[4] Davida, G., Gilbertson, C., and Walter, G., "Analog Cryptosystems," *Eurocrypt85*, (1985).

[5] Davida, G. and Walter, G., "A Class of Analog Cryptosystems," *SECURICOM 87*, (1987).

[6] Denning, D., *Cryptography and Computer Security*, Addison Wesley, Reading, MA (1982).

[7] Diffie, W. and Hellman, M., "New Directions in Cryptography," *IEEE Transactions on Information Theory* IT-22(6)(1976).

[8] Hellman, M., "An Extension of the Shannon Theory Approach to Cryptography," *IEEE Transactions on Informtion Theory* IT-23(1977).

[9] Hellman, M., "A Cryptanalytic Time-Memory Tradeoff," *IEEE Transactions on Information Theory* IT-26(4)(1980).

[10] Kak, S. and Jayant, N., "On Speech Encryption Using Waveform Scrambling," *Bell System Technical Journal*, (1977).

[11] Kak, S., "Overview of Analog Signal Encryption," *IEEE Proceedings*, (1983).

[12] McEliece, R., "A Public Key Cryptosystem Based on Algebraic Coding Theory," DSN Progress Rep. 42-44, Jet Propulsion Lab.,Cal Inst. of Tech, Pasadena, CA (Jan., Feb. 1978).

[13] Nashed, M., "Operator-Theoretic and Computational Approaches to Ill-Posed Problems with Apaplications to Antenna Theory," *IEEE Transactions on Antennas and Propogation*, (March 1981).

[14] O'Bryan, T. and Walter, G., "Mean Square Estimation of the Prior Distribution," *Sankya, The Indian Journal of Statistics*, (1979).

[15] Rivest, R., Shamir, A., and Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM* Vol 21(2) pp. 120-126 (Feb. 1978).

[16] Shannon, C., "Communcation Theory of Secrecy Systems," *Bell Systems Technical Journal* 28(1949).

[17] Stakgold, I., *Green's Functions and Boundary Value Problems*, Wiley, New York (1975).

[18] Tikhonov, A. and Arsenin, V., *Solutions of Ill-Posed Problems*, Winston-Wiley, New York (1977).

[19] Wyner, A., "An Analog Scrambling Scheme Which Does Not Expand Bandwidth: Part II -Continuos Time," *IEEE Transactions on Information Theory* IT-25(6)(1979).

[20] Wyner, A., "An Analog Scrambling Scheme Which Does NOt Expand Bandwidth: Part I -Discrete Time," *IEEE Transactions on Information Theory* IT-25(3)(1979).

V

AUTHENTICATION AND SECURE TRANSACTIONS

Message Authentication with Arbitration of Transmitter/Receiver Disputes*

Gustavus J. Simmons
Sandia National Laboratories
Albuquerque, NM 87185

Extended Abstract

In the most general model of message authentication, there are four essential participants: a transmitter who observes an information source†, such as a coin toss, and wishes to communicate these observations to a remotely located receiver over a publicly exposed, noiseless, communications channel; a receiver who wishes to not only learn the state of the source (as observed by the transmitter) but also to assure himself that the communications (messages) he accepts actually were sent by the transmitter and that no alterations have been made to them subsequent to the transmitter having sent them, and two other parties, the opponent and the arbiter. The opponent wishes to deceive the receiver into accepting a message that will misinform him as to the state of the source. We assume, in accordance with Kerckhoffs' criteria in cryptography, that the opponent is fully knowledgeable of the authentication system and that in addition he is able to both eavesdrop on legitimate communications in the channel and to introduce fraudulent communications of his own choice. Given this, he can achieve his objective in either of two ways:

1) he can impersonate the transmitter and send a fraudulent message when in fact no message was sent by the transmitter,

or

---

† Ideally we would call the states of the source "messages" as is the practice in communications theory. However, if we did this we would be forced to introduce terminology to designate the collection of sequences that are actually transmitted through the channel, perhaps "authenticating codewords," paralleling "error detecting and correcting codewords" from communications theory. Unfortunately, the natural contraction "codeword" already has an accepted meaning in communications theory so that we would either have to coin a new term to designate the specific sequence of symbols transmitted to convey and authenticate a message -- none of which seem very natural -- or else use the cumbersome term "authenticating codeword." The term "authenticator," which is usually used to denote an authenticating suffix appended to the information that is to be authenticated, has too restricted a connotation for the general case. We have opted instead to use the term "message" to designate the sequence of symbols actually transmitted and to tolerate the rather artificial device that the information conveyed by a message is the state of a hypothetical source.

* This work performed at Sandia National Laboratories supported by the U. S. Dept. of Energy under contract no. DE-AC04-76DP00789.

2) he can wait to intercept a legitimate message from the transmitter and substitute in its stead some other message of his own devising.

In either case, the opponent wins if the receiver accepts the fraudulent message as being a legitimate and unmodified communication from the transmitter.

In the simplest model of authentication, the transmitter and receiver are assumed to be mutually trusting and trustworthy and to act with the joint purpose of detecting attempted deceptions by the opponent. Authentication codes have been devised by Brickell, Simmons, Stinson and others that not only achieve this end, but also make perfect use of the information content of the transmitted message in the process. Unfortunately, up till now this has required that the transmitter and receiver be privy to precisely the same secret (from the opponent) information about the specific authentication protocol being used and hence, that each be able to do anything the other can do, i.e., to be able to impersonate each other. What this means, of course, is that if the assumption of mutual trustworthiness doesn't hold, that either will be able to defraud the other in a way that cannot be verified -- or demonstrated -- to a third party. It is here that the fourth party, the arbiter, comes in. The arbiter is provided with secret (known only to him and the transmitter) information as to which messages the transmitter is supposed to use in the communications protocol and may also include information that the arbiter shares in secret with the receiver as to which messages the receiver will accept. His sole function is to certify on demand whether a particular message presented to him is one that the transmitter could have used under the established protocol. He cannot say that the transmitter did send it -- only that he could have under the established protocol. In this setting, the transmitter can cheat if

3) he can cause the receiver to accept and act on a message that he (the transmitter) can later disavow. To be successful, he must choose a message that not only will the receiver accept, but which the arbiter will not certify, because it is not one that would have been used by the transmitter under the established authentication protocol.

If the transmitter succeeds in disavowing the message, the receiver will be, according to the terms of the protocol, held (unjustly) liable.

The receiver can cheat if he can successfully attribute a message of his own devising to the transmitter, i.e., a message not sent by the transmitter, but one which the arbiter will certify as being one that could have been sent by the transmitter under the established authentication protocol. There are two strategies for cheating available to the receiver, paralleling the two strategies available to the opponent:

4) He can claim to have received a message (which he fabricated) from the transmitter, when in fact no message was sent by the transmitter,

or

5) he can wait until he receives a legitimate message from the transmitter and then claim to have received some other message conveying different information than that communicated by the transmitter's message: replacing an order to buy with one to sell, for example.

In either case, if the arbiter later certifies the fraudulent message as being one that the transmitter might have sent under the established authentication protocol, the transmitter will, according to the terms of the protocol, be held (unjustly) liable.

The presence of an arbiter has no effect on the outcome of the opponent's attempted deception. If the opponent is successful in deceiving the receiver, the transmitter will, of course, appeal to the arbiter when he is later held liable (by the receiver) for a message that he (the transmitter) didn't send. The arbiter in this case, depending on whether the opponent chose a message that was not only acceptable to the receiver but which also might have been sent by the transmitter under the established authentication protocol, will assign the liability for the receiver's actions to the transmitter or otherwise to the receiver. The assignment of liability by the arbiter is unjust in either case since neither the transmitter nor the receiver cheated or failed to properly use the authentication protocol, however the arbiter can only verify whether a message is or is not consistent with the protocol in use, and not what its source might be. He can, therefore, never ascribe the liability to the opponent, even though there is always some nonzero probability that this is the source of the message. In keeping with the earlier definition of perfect authentication (without arbitration) systems by Simmons, an authentication system with arbitration will also be defined to be perfect if the probability of success for each of the five types of possible deception is simply the probability of randomly choosing a message that will result in the particular deception occurring.

The smallest example of a perfect authentication code, capable of detecting attempted deceptions by an opponent, but providing no protection against deception by either the transmitter or the receiver, i.e., without arbitration, is the following: The source is a fair coin toss, by the transmitter, whose outcome denoted H or T we take to be the state of the source. There are four encoding rules, e_i, which encode states of the source into one of four possible messages, m_j, according to the system

(1)

	m_1	m_2	m_3	m_4
e_1	H		T	
e_2	H			T
e_3		H	T	
e_4		H		T

The transmitter and receiver choose (in secret from the opponent) an encoding rule with the uniform probability distribution on the e_i (their optimal authentication strategy) in advance of the communication that they wish to authenticate. The opponent knows (1) and their strategy for choosing an e_i. If he chooses to impersonate the transmitter and send an unauthentic message when no message has yet been sent by the transmitter, it should be obvious that irrespective of which message, m_j, he chooses, the probability that it will correspond to an encoding of a source state under encoding rule e_i, and hence that it will be accepted by the receiver as an authentic message, is 1/2. Similarly, if the opponent waits to observe a legitimate communication by the transmitter, his uncertainty about the encoding rule being used will drop from one out of four equally likely possibilities to one out of two. However, his probability of choosing an acceptable (to the receiver) substitute message will still be 1/2. For example, if the opponent observes m_1 he knows that the transmitter and receiver are using either encoding rule e_1 or e_2. In the first case m_3 would be an acceptable message to the receiver while m_4 would be rejected as unauthentic, while in the second case, exactly the opposite would be true. Hence, the opponent's probability of deceiving the receiver is 1/2 irrespective of whether he impersonates the transmitter or substitutes (modifies) legitimate messages.

Clearly since the transmitter and receiver must both know the chosen encoding rule -- the transmitter so that he can encode the source state into a message the receiver will accept and the receiver so that he can decode and authenticate the message, either can do anything the other can. In particular the receiver can claim to have received a message when none was sent and the transmitter will be unable to prove to a third party that he didn't send it, or the transmitter can disavow a message that he did send and the receiver will be unable to prove that he received the message through the communications channel.

The essence of this paper is illustrated in an extension of this simple one-bit source example that in addition to one bit of protection against each of the two possible outsider (the opponent) deceptions, also provides one bit of protection against each of the three forms of insider (transmitter or receiver) cheating described earlier. The receiver first chooses one of the 16 encoding rules defined by the Cartesian product

$$\begin{pmatrix} H & H & - & - \\ H & - & H & - \\ - & H & - & H \\ - & - & H & H \end{pmatrix} \times \begin{pmatrix} T & T & - & - \\ T & - & T & - \\ - & T & - & T \\ - & - & T & T \end{pmatrix}$$

with a uniform probability distribution. For example, the first row of the product would be

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
e_1	H	H	-	-	T	T	-	-
⋮								

which says that a head outcome to the transmitter's coin toss could be communicated by the transmitter using either message m_1 or m_2. Similarly, messages m_5 or m_6 would communicate source state "tails", while messages m_3, m_4, m_7 and m_8 would be rejected by the receiver as unauthentic. The important point to note is that in each of the encoding rules there are exactly two acceptable (to the receiver) messages available for each state of the source. The receiver communicates his choice of an encoding rule to the arbiter in secret (from the transmitter and the opponent(s)). The arbiter next chooses one of the four vectors defined by the Cartesian product

$$\begin{pmatrix} 1 & - & - & 1 \\ - & 1 & 1 & - \end{pmatrix} \times \begin{pmatrix} 1 & - & - & 1 \\ - & 1 & 1 & - \end{pmatrix} \quad ,$$

again, with a uniform probability distribution, and forms the Schur product* of the chosen vector with the encoding rule selected by the receiver. The net result is, for the example of e_1 having been the encoding rule chosen by the receiver, that one of the four possible final encoding rules

$$\begin{matrix} H & - & - & - & T & - & - & - \\ H & - & - & - & - & T & - & - \\ - & H & - & - & T & - & - & - \\ - & H & - & - & - & T & - & - \end{matrix}$$

will be chosen (with a uniform probability distribution) as a result of the concatenated choices of the receiver and arbiter. The arbiter communicates, in secret (from the receiver and the opponent(s)), the resulting final encoding rule to the transmitter. This rule is then the established protocol that the transmitter is supposed to use to encode the observed state of the source into the message that is to be transmitted to the receiver. Assume, for example, that the arbiter chose the vector

---

* Given vectors $A = (a_i)$ and $B = (b_i)$ the Schur product is the vector $C = (a_i b_i)$.

- 1 1 - 1 - - 1

so that the resulting final encoding rule is

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
-	H	-	-	T	-	-	-

.

In this example, source state "heads" is to be communicated by the transmitter sending message m_2 while "tails" is to be communicated by sending message m_5.

Using this authentication scheme we now show that the immunity provided to each of the five types of cheating described earlier is to hold the cheater to a probability of 1/2, i.e., one bit of protection, as claimed irrespective of which type of cheating is considered. The easiest of the deceptions to analyze is the case of the outsider (opponent) who only knows the "system," i.e., he knows what the procedures are but does not know the receiver's or arbiter's choices. It should be clear that if he attempts to impersonate the transmitter and send a message when none has been sent, his probability of choosing one of the four (out of eight) messages that the receiver has agreed to accept (in his choice of an encoding rule) is 1/2 since in each case there are four equally likely messages that will be accepted as authentic and four that will be rejected as unauthentic. On the other hand, if he waits to observe a message, say m_1, his uncertainty about the encoding rule chosen by the receiver drops from one out of sixteen equally likely candidates to one out of four, however these four leave him with four equally likely possibilities for the message that the transmitter is to use to communicate the other state of the source, and much more importantly, with four equally likely pairings of messages that the receiver would accept as communicating the other state of the source, with each message occurring in precisely two of the pairs. The net result is that the opponent's probability of success in substituting a message that the receiver will accept as communicating the other state of the source is still 1/2.

Consider next, the next simplest case to analyze, the transmitter disavowing a message that he actually sent. In order to succeed, the transmitter must choose a message that the receiver will accept but that is not used in the established protocol forwarded by the arbiter. In other words he must choose a message that was used in the encoding rule that the receiver chose, but not used in the final encoding rule generated by the arbiter's choice. Continuing with the example used above, the transmitter knows from the final encoding rule that was given to him by the arbiter:

- H - - T - - -

that the arbiter must have chosen vector

- 1 1 - 1 - - 1

and hence can infer that the receiver must have chosen one of the four encoding rules.

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
H	H	-	-	T	T	-	-
H	H	-	-	T	-	T	-
-	H	-	H	T	T	-	-
-	H	-	H	T	-	T	-

Since messages m_3 and m_8 do not appear in any of these rules, the transmitter can be certain that they would be rejected by the receiver as unauthentic, and hence he will not send either of these. Each of the remaining four messages, m_1, m_4, m_6, and m_7, appear in two of the equally likely choices of an encoding rule, hence he cannot do better than choose one out of these four messages with equi-probability. Irrespective of which of the four he chooses, the probability that it will be accepted by the receiver is 1/2. If it is accepted, the transmitter can disavow having sent it, since he knows that the arbiter will not certify it as a message that would have been used under the established protocol.

Finally, we consider the two types of cheating available to the receiver. Of the four messages that he has agreed (with the arbiter) that he will accept as authentic, since they are used in his choice of an encoding rule, two will be certified as being messages that could have been used under the established protocol and two will not be certified. The receiver will succeed in fraudulently attributing a message to the transmitter if he is able to choose one of the pair that the arbiter will certify and will fail otherwise. It should be clear that his probability of success is 1/2 since the arbiter's selection procedure chooses among the acceptable (to the receiver) messages with a uniform probability distribution. If he waits until he receives a message from the transmitter, say m_2, he can reduce his uncertainty about the vector that the arbiter choose from one of four equally likely cases to one of two:

$$- \ 1 \ 1 \ - \ 1 \ - \ - \ 1$$

or

$$- \ 1 \ 1 \ - \ - \ 1 \ 1 \ -$$

in the example. The result however is that either message m_5 or m_6 is equally likely to be the one that will be certified, and his probability of successfully substituting a message conveying a different state of the source than was communicated in the message sent by the transmitter, i.e., of substituting one which will both communicate a different state of the source and will subsequently be certified by the arbiter as a message the transmitter could have sent under the established authentication protocol is 1/2.

This small example illustrates all of the essential features of authentication codes that permit arbitration. Three bits of information must be communicated to identify one of eight equally likely messages. According to the protocol, this communication provides one bit of information about the source state, one bit of protection against deception by outsiders and one bit of protection against cheating by insiders. Although channel bounds have not yet been proven for authentication codes that permit arbitration (tight bounds have been proven for authentication codes that do not permit arbitration) it seems reasonable (as mentioned above) to describe the code illustrated here, and generalized in the main paper, as perfect. As has been pointed out in earlier papers on authentication codes without arbitration, these "perfect" codes are also perfect in the natural sense that all of the information transmitted is used either to communicate the state of the source or else to confound one of the cheating parties.

Perfect and essentially perfect authentication systems

Albrecht Beutelspacher
Siemens AG
ZT ZTI SYS 4
D-8000 München 83

In any authentication system [2] with k keys available, the bad guy has at least a chance of $1/\sqrt{k}$ to deceive the system (Theorem of Gilbert, MacWilliams and Sloane [1]). A system is called **perfect** if the bad guy's chance is exactly $1/\sqrt{k}$, it is called **essentially perfect** if his chance is $O(1/\sqrt{k})$.

Using combinatorial properties of finite projective planes, Gilbert, MacWilliams and Sloane [1] have constructed perfect authentication systems with $k+1$ messages and k^2 keys, whenever k is a power of a prime number. These systems have the disadvantage that the number of messages is much smaller than the number of keys.

Using more sophisticated geometric structures, we shall construct essentially perfect, and even perfect, authentication systems, in which the number of messages exceeds the number of keys. The main ingredients of our constructions are projective spaces, partial spreads and geometries having many parallel classes.

Three system should be mentioned here in particular.

1. The projective space $\mathbf{P} = PG(3,q)$ of dimension 3 and order q. $\mathbf{P}$ has a set $\mathbf{F}$ of q^2-q mutually skew lines; there are exactly $(q+1)^2$ points not covered by $\mathbf{F}$. For example, one could take an "elliptic congruence" in which the lines of one hyperbolic quadric are missing.

Take as *messages* the lines of **F**, as *keys* the points not covered by **F** and as *authenticators* the planes through a line of **F**. It turns out that this system is perfect with $(q+1)^2$ keys and q^2-q messages.

2. Take again **P** = PG(3,q) embedded as a hyperplane in **P*** = PG(4,q). Define the *messages* to be (all) the lines of **P**, the *keys* to be the points of **P***–**P** and the *authenticators* to be the planes of **P***–**P**.

It turns out that this system is in fact essentially perfect with q^4 keys and $(q^2+1)(q^2+q+1)$ $(> q^4)$ messages. One can generalise this construction to arbitrary dimensions. This yields essentially perfect authentication systems, in which the factor m/k is unbounded above (m being the number of messages).

3. Let **P** = PG(2a–1,q), $a \geq 2$, and denote by **U** a subspace of dimension 2a–3 of **P**. then there is a set **S** of mutually skew lines of **P** which covers exactly the points of **P**–**U**. Let us define an authentication system A as follows:

Messages are the (a–1)-dimensional subspaces of **U**,

keys are the elements of **S**,

authenticators are the (a + 1)-dimensional subspaces which intersect **U** in a subspace of dimension a–1.

It turns out that this system is perfect. It has the remarkable property that the number of keys is $q^{2(a-1)}$, while the number of messages is $q^{(a-2)^2/2}$.

Reference

1. N.G. Gilbert, F.J. MacWilliams, N.J.A. Sloane; Codes Which Detect Deception. Bell Syst. Tech. J. **53** (1974), 405-424.

2. D.W. Davies and W.L. Price, Security for Computer Networks. John Wiley & Sons, 1984.

MESSAGE AUTHENTICATION AND DYNAMIC PASSWORDS

by Professor H.J. Beker and Mr. G.M. Cole, Racal-Guardata Limited.

The security of transactions flowing across a communications network is of ever increasing importance. In many such circumstances it is important not only to protect the messages from passive interception but also, and often of greater importance, to be able to detect any active attack against messages. An active attack may take the form of an interceptor tampering with the message: altering it, adding information, removing information and so on. While it is almost impossible to prevent an active attack there are many mechanisms to ensure, with a high probability, that such an attack may be detected and hence rendered harmless. The techniques to allow detection and thus audit take many forms of which the most common are normally cryptographically based and depend upon the generation, before transmission of the message, of a check-sum which is then appended to the message. The theory underlying this approach works on the basis that if a would-be fraudster changes any part of the message in any way then the check-sum will no longer be correct and thus the recipient of such message can compute, for himself, the expected check-sum, compare it with that received in the message and if they disagree will know the message has been altered. If on the other hand the expected and received check-sums agree then he knows with a high probability that the message has not been altered. This probability is dependent upon the amount of information within the check-sum (i.e. the longer it is) the lower the probability of an undetected alteration.

Many such systems exist. Some of these depend only upon an algorithmic check-sum, often called a test-key or authentication parameter. In this case the security level is often relatively low since someone attacking the system with knowledge of this algorithm may be aware of ways in which he can alter the message without affecting the check-sum computation. A trivial example of this is as follows: suppose the check-sum on a numeric message is computed solely as the modulo-10 sum of all digits in the message. An attack upon the system which simply involves altering the order of the digits in the message would not be detected by the check- sum.

A normally more secure technique involves the use of a cryptographic check-sum, often termed a message authentication code. In this case the check-sum is dependent not only upon the cryptographic algorithm but also a cryptographic key. An example of this, in common usage, is the system described within the American National Standards Institute (ANSI) standards X9.9 and X9.19. Within these standards the cryptographic algorithm is the Data Encryption Algorithm as described in FIPS 46 and ANSI X3.92. The cryptographic key is a 56-bit DEA key. The check-sum or message authentication code (MAC) is a 32-bit value appended to the message. It is currently generally accepted that provided the cryptographic key is kept secret then any alteration to the message will be detected by the recipient with a probability of 0.9999999998.

Within some communications systems protection of messages in the above manner is considered adequate. However there do also exist many systems within which it is important not only to detect any alterations to the message, and thus be able to provide alarms and an audit system of these, but also to identify the person or group of persons from which such a message originated. This is in some

sense equivalent to requiring a signature on the message. We shall now go on to describe how, by use of another commonly used technique of dynamic passwords, such messages can be signed and thus far greater protection afforded. We begin by describing the technique of dynamic passwords as it is commonly used for access control. We shall then go on to show how the technique can be combined with a check-sum to provide a 'signature'. As we shall see the combination of the two techniques, in the manner described, will provide far greater levels of security.

Computer access control systems often depend upon a static user password. These systems are notorious for their insecurity. Recently, dynamic password systems have become more popular. There are many variations on this particular theme. By way of example we describe one such method.

Having entered his user identity, the user is presented, by the system, with a challenge. The user must then provide the correct response to this challenge in order to be granted access. The theory behind this system is that since the system has control of the challenge and the response will be unique, for that user, to that challenge, the system is running, essentially, a one-time password system. Any unauthorised person will not know how to respond to the challenge in the correct way and will thus be denied access to the system. Similarly anyone recording the challenge and
response will be unable to directly use this information since ideally that challenge will never be used for that user again.

There are many techniques possible to allow the user to produce the correct response. These vary from biometric techniques to user tokens. A typical method involves a user token similar to a small calculator which can be correctly activated by the user via entry of a Personal Identification Number (PIN). Once the device has been correctly activated entry of the challenge will result in the correct response being generated by the token. This may be achieved, for instance, via a one-way function of the challenge and a cryptographic key unique to that user and embedded in his token. Thus, loss of the token does not enable an unauthorised user to enter the system since he requires the PIN to correctly activate the system. Indeed, a would-be hacker requires both the PIN and token as well as the user ID or the algorithm and cryptographic key corresponding to a user ID in order to enter the system. In the case of a token being used in this way it may well take the form of a 'smart card'. Biometric means may also be used.

We shall now give an example of how a check-sum can be combined with a dynamic password system in order to provide message and user authentication within a system.

For example we shall consider a user, issued with a dynamic password token, using a terminal which can provide a cryptographic check-sum or message authentication code (MAC). We shall also assume that the recipient of the message is in possession of the appropriate cryptographic keys to check both the MAC and 'response'.

Once the user has compiled his message the terminal will generate an appropriate MAC, or some derivative of it which is presented to the user as his 'challenge'. Once he has produced the correct response to that challenge and appended this response or a function of it to the message the message has been 'signed' by the user. On receiving the message the recipient can not only check the MAC but

may also via the user ID check the response to that 'MAC challenge' thus also authenticating the originator of the message.

Such a system may have considerable benefits within a scenario within which a corporation or institution is allowing users to enter messages into its computer network. Typically this might be a corporate banking network where the institution is a bank and is accepting payments, transfers etc. from its customers. This system may be set up as follows:

The institution issues to the user the cryptographic MAC facility in a tamper resistant form. This may constitute the entire terminal or a part of it. The cryptographic key upon which MAC security depends is contained within the tamper resistant enclosure. The corresponding cryptographic key may be held by the institution itself encrypted under a master key which again is contained in a highly tamper resistant enclosure. Similarly, the user is also issued with his dynamic password token itself containing a cryptographic key in a tamper resistant manner while again held by the institution encrypted under a master key. Assuming the tamper resistant enclosure containing the master key can also carry out the appropriate cryptographic functions then the institution can only be compromised while the devices are being set up or through a breach of the tamper resistant module containing the master key.

At the user level, the system can only be compromised via an attack upon both the user's cryptographic facility and his token. Bearing in mind that should he lose his token it will normally be in his interests to report this as soon as possible the system provides a high level of security.

Since this procedure is centred around the concept of using the MAC (or check-sum) as the `challenge' to the user let us see what extra security benefits are thus achieved.

1. Since the response now depends on the MAC it depends upon those sensitive parts of the message which the MAC was itself protecting and thus is a message dependent response. It is in this way that it provides a similar facility to a signature. In particular this response cannot be removed from this message and appended to another since it will no longer be appropriate and will therefore be detected by the recipient.

2. Even if some unauthorised person were able to discover the cryptographic key associated with the MAC, by breaking into the user's terminal or otherwise, this would not be sufficient to penetrate the system since any alteration of the MAC in turn would mean the response on the message would now be inappropriate and would therefore be detected by the recipient.

3. An implication of the above remark is that theoretically the institution could give all its users the same cryptographic key for the cryptographic MAC facility and still be assured a high level of security through the response confirmation.

4. Clearly if the user's identity was incorporated into the message and the MAC calculation, then only the holder of that corresponding dynamic password token (and corresponding PIN, if used) could 'sign' the message.

We therefore see that the system now has two interrelated security mechanisms: the MAC and the response. As we stated above an attack by an unauthorised user would need to be directed either at the institution's highly tamper resistant facility or at both a user's cryptographic facility and that user's token. We believe such an attack to be extremely difficult.

Abstract

IC cards in High Security Applications

Ingrid Schaumüller-Bichl
VOEST-ALPINE AG
A-4031 Linz/Austria

IC cards - plastic cards with imbedded CPU and memory chip - have been gaining growing interest during the last years.

Their wide range of applications comprises access control, data protection, electronic money, personal data files and operational functions.

Each application sets security requirements to the card, yet the scope of these requirements can be very different for each application.

The paper describes an IC card concept - based on special cryptographic functions - that provides a very high level of security against a broad range of potential attacks, thus being well suited for high security applications.

Two main ideas form the basis of this new concept:

1) Block structure:
   The data memory of the card is structured into blocks of arbitrary lengths. Each block is assigned to a specific application and protected by a specific PIN. Security levels of different blocks can be different.

2) On-card encryption:
   A block cipher algorithm is implemented on the card, serving for three main functions:

i) Encipherment of data stored on card:
All (confidential) data on the card is stored in enciphered form, the key is a combination of PIN and parameters unique to each card

ii) Communication encryption
Serves for cryptographic protection of communication between card and card reader, if required (depending on application)

iii) Black box cipher system:
Encryption of external data under a key stored in the card. This proves especially valuable for efficient key management functions.

This card concept provides
- a very high level of security
  both for data stored on the card - even employing an electron microscope does not reveal any confidential data - and the data communicated to the card reader.
  It also prevents unauthorized copying or simulation of the card efficiently.

- high flexibility:
  the fact that each block can be protected by a seperate PIN allows to realize different applications in one card without prior communication between card issuers ("Multifunctional cards")

- reduced PIN management problems:
  as the PIN is block dependent, user selectable and choosable at the time of block creation, generation, storage and transmission of PINs (if necessary at all) are much easier than in other concepts.

Possible applications of this IC card concept cover all known applications for IC cards.

Yet it is specially suited for high security applications in the area of data protection and access control.

There it is made full use of the two main functions of IC cards:

- storage of confidential data
  like cryptographic keys or passwords
- processing of special security functions
  like encryption under a master key

Typical fields of applications are key management for communication encryption and file encryption, access control, user identification, authentication and software protection.

VI

HASH FUNCTIONS AND SIGNATURES

COLLISION FREE HASH FUNCTIONS AND PUBLIC KEY SIGNATURE SCHEMES.

Ivan Bjerre Damgård[1]

(Extended Abstract)

1. Introduction.

The use of hash functions with digital signature schemes has been suggested many times before [1],[4]. The motivation for using a hash fuction can be to prevent undetected changes in a signed message, or to protect against certain types of chosen message attacks[1]. Since the output of a hash function is usually shorter than the input, the use of a hash function can also improve performance of a signature scheme.

Several attempts have been made to construct hash functions using e.g. DES or RSA as building blocks. However, none of these suggestions have been proved to be secure, and several of the proposals using DES have been proved to be insecure([5] and [6]). Other variations of hash functions have also been proposed[8]. But the use of these functions, like pseudo-random functions[9], require that sender and receiver share a secret key. They are therefore suitable for authentication purposes, but do not fit into the scenario of a public key signature scheme. We would like to have publicly known hash functions, that are easy for all users to compute.

2. Constructing collision free hash functions.

Let Σ be a finite alfabet and let M be the set of all finite words over Σ. A hash function is a map $h: M \to A$, where A is some finite set.

The most basic demand to a good hash function is that it should be computationally infeasible for an adversary to find collisions, ie. different messages hashing to the same value. A hash fuction h is said to be collision free if it is easy to compute h on any input, but computationally infeasible on inputs h, M and A to find any $x,y \in M$ such that $h(x) = h(y)$. (in the full paper, we choose a specific computational model to describe what "computationally infeasible" means, and we put the definition of collision free functions into a complexity theoretic setting by considering families of hash functions indexed by a security parameter. The same is true for the claw free sets of permutations discussed below).

If h is collision free, then h is easily seen to be one-way, while the converse is not necessarily true.

---

[1]This work was supported by DNSRC. The author is with Mathematical Institute, Aarhus University, Denmark.

In [2], the notion of claw free pairs of trapdoor permutations is introduced. We shall use a generalisation of this idea, without the trapdoor property:

A set of permutations $\{f_0,..,f_{r-1}\}$ is called claw free, if the following is satisfied:

All permutations in the set have the same domain.

For any x in the domain, $f_i(x)$ is easy to compute.

It is computationally infeasible to create a claw, i.e. to find x,y such that for some $i \neq j$, $f_i(x) = f_j(y)$.

Assuming the existance of clawfree sets of permutations, we can construct collision free hash functions. First some notation:

Let m be a finite word over Σ. We the let $[m]$ denote a prefix free encoding of m over Σ. m_i will denote the i'th letter in $[m]$. If $\{f_0,f_1,..,f_{r-1}\}$ is a set of permutations, and $I \in$ domain(f_0), we define

$$f_{[m]}(I) := f_{m_1}(f_{m_2}(...f_{m_s}(I)...)),$$

where $[m] = m_1\, m_2 ... m_s$, and the elements of Σ are denoted by numbers $0,...,r-1$. A similar construction is used in [2] with $r = 2$. Note that it is possible to choose an encoding such that the length of $[m]$ is a linear function of the length of m(see [3]).

Theorem 2.4

The function h constructed below is a collision free hash function.

Let $S= \{f_0,..,f_{r-1}\}$ be a set of claw-free permutations. We let Σ be the alfabet of cardinality r given by $\Sigma = \{0,1,..,r-1\}$. We then define h by:

$h(m) := f_{[m]}(I)$ for all $m \in M$.

proof.(sketch)

Assume for contradiction, that h is not collision free. This means that we can find m and m' in M such that: $m \neq m'$ and $f_{[m]}(I) = f_{[m']}(I)$. But the prefix free property of the encoding now implies that we also have a claw for S.

If we let T denote the time needed to compute the value of one of the permutations used in the construction, it is clear, that the time needed to compute h on a message of length L is $O(L \cdot T)$

Note, that with a claw free set with 2^s elements, we can hash a binary message by processing s bits at a time.

We now give a construction of clawfree sets of permutations under the assumption that factoring is hard. Suppose we want a set with r elements.

Let $n = p_1 \cdot p_2 \cdot ... \cdot p_t$, where all the p's are k bit prime numbers equivalent to 3 mod 4, and where t is the smallest integer such that $2^{t-1} \geq r$.

For each a prime to n, define

$$J(a) = \left(\left[\frac{a}{p_1}\right], \left[\frac{a}{p_2}\right], \ldots, \left[\frac{a}{p_t}\right] \right).$$

QR(n) will be the set of quadratic residues mod n. Choose a set of elements in Z/nZ $A = \{a_0, \ldots, a_{r-1}\}$ such that $J(a_i) \neq \pm J(a_j)$ for $i \neq j$. This is clearly possible by choice of t. A is called an injective set of numbers whenever it satisfies this condition. We can now define $\{f_0^{(n)}, f_1^{(n)}, .., f_{r-1}^{(n)}\}$ to be the set of permutations of QR(n) given by

$$f_i^{(n)}(x) = (a_i x)^2 \bmod n, \quad \text{for } x \in QR(n) \text{ and } i = 0, .., r-1.$$

It is easy to see that by choice of the a_i's, finding a claw is as hard as finding a non trivial factor of n given the injective set. A series of technical lemmas then proves that knowlegde of an injective set does not help in factoring n, even if the set size is allowed to grow polynomially with the security parameter: even without knowing the factors, an injective set can always be guessed with a constant probability greater than 0, which does not depend on the number of factors.

With this construction of clawfree sets, the value of a hash function can be computed in about the same time as is needed to apply RSA to the whole message. It will often be possible to improve performance of our hash functions by using large sets, rather than fx. pairs. One should be aware, though, that a larger set requires more prime factors and hence a longer modulus - note that a modulus with many shorter prime factors will not be secure because of factoring algorithms like Lentras elliptic curve method[7], whoose running time depends mostly on the size of the smallest prime factor. The gain in speed we can get therefore also depends on how modular multiplication is implemented.

3. Combining a signature scheme and a hash function.

We consider a general model of a public key signature scheme. We choose any scheme which fits in this model and call it "the original scheme". We then combine with a hash function: to sign a message, compute the value of the hash function on the message and sign this value using the original scheme. This is called "the combined scheme".

Following [2], we call a signature scheme "existentially forgeable" if, under some attack, an enemy can forge the signature of at least one message. The message is not necessarily chosen by the enemy.

An "adaptive chosen message attack" is an attack, where the enemy can choose messages to be signed during the process of forging a signature. Hence the choice may depend on the results of earlier computations.

It is of course essential to be able to compare the security of the two schemes. A first result in this direction is:

Theorem 3.1.

Suppose that the original scheme is not existentially forgeable under an adaptive chosen message attack, and that the hash function used is collision free. Then the combined scheme is also not existentially forgeable under an adaptive chosen message attack.

We also discuss how a collision free hash function may be able to improve the security of a weaker signature scheme. It follows that if we use the construction from Section 2, we can implement e.g. RSA combined with a hash function about as efficiently as RSA in its basic form, without extra hardware, but with much better security.

4. Speeding up the Goldwasser-Micali-Rivest scheme.

In [2], the Goldwasser-Micali-Rivest (GMR) signature scheme is introduced, and is proven not to be existentially forgeable, even under an adaptive chosen message attack. Thus by Theorem 3.1 above, we can obtain a signature scheme with the same level of security by combining with a collision free hash function. It is proved that in the factoring based implementation of the GMR-scheme, this will speed up the signing process by a factor of roughly the length of the moduli used, if messages are long compared to the length of a signature. Other methods have been proposed for speeding up the signing process, both by Goldreich[12] and in the full version of the GMR-paper[13]. Both these methods give roughly the same speedup as ours, but are fundamentally different: Goldreich's method is based entirely on properties of the factoring based construction of claw free permutations, while our method is potentially useful in any construction. The method in the GMR-paper does not(like ours) allow the use of non trapdoor permutations, since this would invalidate the proof of security for the GMR-scheme.

References.

[1] Denning: Digital Signatures with RSA and Other Public-Key Cryptosystems. Comm. of ACM, vol.27 nr.4, 1984, p.388-392.

[2] Goldwasser/Micali/Rivest: A "Paradoxical" Solution to the Signature Problem. Proc. of the 25'th ann Symp on Found. of Computer Science, Singer Island, Florida 1984, p.441-448.

[3] Berstel/Perrin: Theory of codes, Academic Press, 1985.

[4] Davis/Price: The Application of Digital Signatures based on Public Key Crypto Systems. Proc. of fifth Int. Comp. Comm. Conference 1980, p.525-530.

[5] Winternitz: Producing a One-Way Hashfunction from DES, proc. of Crypto 83, Plenum Press, 1984, p.203-208.

[6] Coppersmith: Another Birthday Attack, Proc. of Crypto 85, Springer 1986.

[7] H.M. Stephens: Lenstras Factoring Method based on Elliptic Curves, Proc. of Crypto 85, Springer 1986.

[8] Wegman/Carter: New Hashfunctions, J.Computer and Syst. Sci., v.22, 1981.

[9] Goldreich/Goldwasser/Micali: How to Construct Random Functions,MIT/LCS/TM-244, 1983.

[10] Blum/Micali: How to Generate Cryptographically Strong Sequences of Pseudorandom Bits, SIAM J. Computing, v.13,1984.

[11] Kochanski: Developing an RSA-Chip, Proc. of Crypto 85, Springer 1986.

[12] Goldreich: Two remarks concerning the GMR-signature scheme. Presented at Crypto 86.

[13] Goldwasser, Micali and Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen Message Attack. To appear.

HASH-FUNCTIONS USING MODULO-N OPERATIONS

(extended abstract)

Marc Girault
Service d'Etudes communes des Postes et Télécommunications
Péricentre V - 53, Avenue de la Côte de Nacre
14040 Caen Cedex, France.

0. INTRODUCTION

Today, there is a need for one-way hash-functions, essentially for use in digital signatures [1]. Along with many experts, we define that H is a one-way hash-function if it maps messages of arbitrary length to some small fixed length, and if, for any M and value H(M), it is computationally infeasible to find another message M' such that H(M') = H(M).

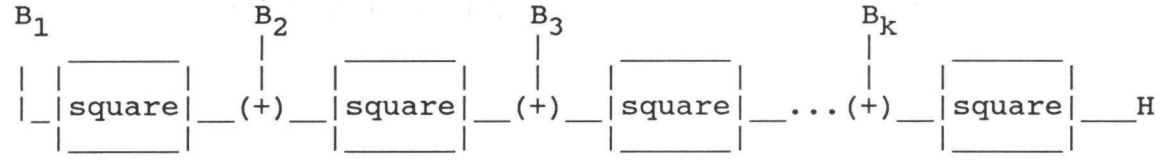
Until now, much attention has been paid to hash-functions based on a conventional encipherment algorithm, generally DES [2,3,4,5,6,7]. Nevertheless, it is also useful to design hash-functions using modulo-n operations, particularly if S is the secret function of a public-key system (RSA or a successor). In 1984 D.W. DAVIES and W.L. PRICE proposed such a hash-function [8,9]. The main objective of this paper is to show the weakness of their scheme (referred to below as the DP scheme) and of some variations of it. This weakness was pointed out (but without details) for the first time by A. JUNG [10].

I. DESCRIPTION OF THE DP SCHEME :

The basic idea, introduced by R.R. JUENEMAN but used by him in a somewhat different way [11], consists in choosing a (public) integer n, in dividing the message into k blocks B_i smaller than n and in performing :

$$
(E)\quad
\begin{cases}
H_1 = B_1^{\,2} & \bmod n \\
H_2 = (H_1 \ (+)\ B_2)^2 & \bmod n \\
\quad\vdots & \\
H_{i+1} = (H_i \ (+)\ B_{i+1})^2 & \bmod n \\
\quad\vdots & \\
H_k = (H_{k-1} \ (+)\ B_k)^2 & \bmod n
\end{cases}
$$

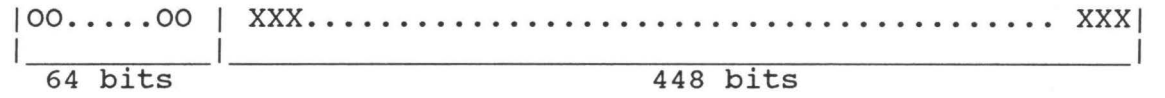
The final result is $H=H_k$ ((+) stands for : exclusive or)



Of course, if the B_i are not required to have some redundancy, the security of this scheme is jeopardized : the enemy can choose all the blocks as he likes, then "correct" them by properly choosing the last one.

So, the D.P.scheme is :

- choose a public 512 bits-long modulus n
- divide the message M into 448 bits-long blocks $B_1, B_2, \ldots B_k$; that is equivalent to imposing 64 bits of redundancy at the head of each B_i , these bits being equal to 0:



- apply the equations (E)

Unfortunately, this scheme is not very secure, as shown below.

II. THE WEAKNESS OF THE DP SCHEME

Let us call "valid" a 512 bit block with the 64 m. s. b. (most significant bits) equal to zero. A successful attack consists in replacing the valid blocks $(B_1, \ldots, B_k)$ by other valid blocks $(B'_1, \ldots, B'_q)$ leading to the same hash result.

* Replacement of (B_1, B_2) by (B'_1, B'_2)

Suppose that the enemy would like to replace B_1 by B'_1 . He should correct the corresponding variation of H_1 by replacing B_2 by B'_2 :

$$\begin{cases} H_1 = B_1^2 \\ H_2 = (H_1 \ (+) \ B_2)^2 \end{cases} \quad \text{becomes} \quad \begin{cases} H'_1 = B'^2_1 \\ H'_2 = (H'_1 \ (+) \ B'_2)^2 \end{cases}$$

(all these equations modulo n)

The blocks B'_1 and B'_2 must be chosen such that :

(i) B'_1 is valid

(ii) $B'_2 = B_2$ (+) H_1 (+) H'_1
(in order to have $H'_2 = H_2$, hence $H'_k = H_k$)

(iii) B'_2 is valid.

The difficulty is clearly that, if the fake (but valid) block B'_1 is chosen without precautions, and B'_2 as in (ii), then (iii) will not be satisfied, with a probability equal a priori to $1-2^{-64}$. In fact, (iii) will be satisfied if and only if H_1 and H'_1 have the same m.s.b.; so, the conditions (i), (ii) and (iii) can be replaced by :

(i) B'_1 is valid and is such that H_1 (+) H'_1 have the same m.s.b.

(ii) $B'_2 = B_2$ (+) H_1 (+) H'_1

We now show that the enemy can find a number of blocks B'_1, satisfying the new condition (i),so that one of these blocks may be advantageous for him. The technique uses the Extended Euclidean Algorithm (EEA) in a way similar to that used (indepedentely) by W.De JONGE and D. CHAUM [12].

Let d be such that $B'_1 = B_1 + d \pmod n$ (+ is addition modulo n). So :
$H'_1 = (B_1 + d)^2 = B_1^2 + 2dB_1 + d^2 = H_1 + (2dB_1 + d^2) \bmod n$

Roughly speaking, "B'_1 valid" means "d small" and "H_1 and H'_1 have the same m.s.b." means " $(2dB_1 + d^2) \bmod n$ small". Here, "x small" means about $|x| < 2^{448}$ (or a little less).

The second condition is satisfied in particular if :

| d is "very small" (say $< 2^{224}$)
| $2dB_1 \pmod n$ is "small".

The problem is now reduced to finding a very small d such that $2dB_1 \pmod n$ is small. The EEA finds, for a given n and a given x with $0<x<n$, some rational integers a_j and b_j such that : $a_j n + b_j x = r_j$ (====> $b_j x = r_j \pmod n$) where r_j is the j-th partial remainder of the Euclidean Algorithm : r_j is a decreasing function of j but b_j is an increasing one.

As we are going to apply this algorithm to $x = 2B_1 \pmod n$ (hence b_j will play the role of d), the hope is that we find some j such that "b_j very small" and "r_j small" are simultaneously true. In fact, a result mentioned in [12] transforms this hope into a quasi-certainty.

In practice, the EEA is so fast that we need not try and "guess" the interval (j_0, j_1) in which both conditions will be satisfied. Our algorithm, applied to some n and B_1 picked up at random, allowed us to find 121 pairs (B'_1, B'_2). In fact, we can find a substantially larger number of pairs (895) by slightly modifying the Euclidean Algorithm in the interval (j_0, j_1). Many other trials confirmed this first result.

*** Replacement of (B_i, B_{i+1}) by (B'_i, B'_{i+1})**

The same technique is also effective if the enemy wants to replace any pair (B_i, B_{i+1}) by (B'_i, B'_{i+1}).

III. VARIATIONS OF THE DP SCHEME

Several variations can be considered:

- increase the redundancy length : the attack remains effective with 128 bits (or even 160) of redundancy but fails with 176 bits or more;

- put the redundancy bits elsewhere :

  * on the right : the attack (slightly adapted) remains effective (but less than previously) up to 96 bits (or even 128) of redundancy;

  * in the middle : the attack seems to be no more effective ;

  * dispersed in the block : idem.

- change the redundancy bits (e.g. alternatively "0000" and "1111") : the attacks remains effective.

By combining these types of variations, A. JUNG designed the TTT/OSIS hash-function, currently proposed for standardization in CCITT [13] and ISO. It will be discussed in the full paper, as well as some other proposals (in particular from SEPT and D. CHAUM).

****

IV. REFERENCES

[1] - D. PINKAS : "The need for a Standardized Compression Algorithm for Digital Signatures", Eurocrypt 86.

[2] - D.E. DENNING : "Digital Signatures with RSA and Other Public-Key Cryptosystems", CACM, Vol. 27, N°4, April 84 pp. 388-392.

[3] - M. RABIN : "Digital Signatures", Foundations of Secure Computation, Academic Press, New York, 1978.

[4] - D.W. DAVIES : "Applying the RSA Signature to Electronic Mail", Computer, 1983.

[5] - R.S. WINTERNITZ : "Producing a One-Way Hash Function from DES", Crypto 83.

[6] - S.G. AKL : "On the Security of Compressed Encodings", Crypto 83.

[7] - D. COPPERSMITH : "Another Birthday Attack", Crypto 85.

[8] - D.W. DAVIES, W.L. PRICE : "Digital Signatures, an update", Proc. International Conference on Computer Communications, Sydney, October 1984, pp. 845-849.

[9] - "Modes of Operations and Hash-functions", ISO/TC97/SC20/WG2/N31, July 1985 .

2[10]- OSIS European Working Group WG1 : "OSIS Security Aspects", Final report, October 1985.

[11]- R.R. JUENEMAN, S.M. MATYAS, C.H. MEYER : "Message Authentication with Manipulation Detection Codes", Proc.Security & Privacy, 1983, pp 33-54, IEEE Catalog N° 83CH1882-0.

[12]- W. De JONGE, D. CHAUM : "Attacks on some RSA Signatures", Crypto 85.

[13]-"The Directory-Authentication Framework", CCITT/Com7/Q35, Draft Recommendation Xds7 (version 4), Geneva, October 1986.

Blinding for Unanticipated Signatures

David Chaum

Centre for Mathematics and Computer Science
Kruislaan 413 1098 SJ Amsterdam

Summary

Blind signature systems published until now require computation at least proportional to the number of types of signatures that can be used, and require that the number of such types be fixed in advance. Several applications of blind signatures require a large number of signature types, and some require a number of types that cannot in general be fixed initially. Here, a totally new blind signature technique is introduced that allows an unlimited number of signature types with only a (modest) constant amount of computation.

Background

In an RSA public key signature system [Rivest, Shamir & Adleman 78], a party that will be called the ***signer*** chooses two appropriate large primes p and q, and makes their product n (= $p \cdot q$) public. The signer also makes public l "public exponents" $e_1, ..., e_l$. Additionally the signer computes corresponding "secret exponents" $d_1, ..., d_l$ satisfying $d_i \equiv e_i^{-1} \pmod{(p-1)\cdot(q-1)}$, where $1 \le i \le l$. The signer then forms the ith signature on a number m as $m'_i \equiv m^{d_i} \pmod{n}$. Anyone can use the public n and e_i to verify the ith signature on m by checking that $m \equiv (m'_i)^{e_i} \pmod{n}$ holds.

Blind signatures prevent the signer performing a commercial service such as validating electronic bank notes, issuing signatures whos types encode credentials, notarizing or time stamping electronic documents, etc., from determining the exact content of each message signed—even if the signer has infinite computing power. In blind signature systems, a party wishing a signature on some message will be called the ***provider***. First the provider ***blinds*** the message before submitting it to the signer for a signature; when the signed but still blinded message is returned by the signer, the provider is able to ***unblind*** it and thereby recover the original (no blinded) message bearing the signature.

The previously published blind signature system [Chaum 83] worked as follows: the blinding of a message m with a suitably chosen random r produces $t \equiv m \cdot r^{e_i} \pmod{n}$; the sign-

ing of t yields $t' \equiv m^{d_i} \cdot r \pmod{n}$; and the provider unblinds t' by forming $m'_i \equiv t' \cdot r^{-1} \pmod{n}$, yielding $m'_i \equiv m^{d_i} \pmod{n}$.

Notice that it is necessary for the provider to ***anticipate*** the particular d_i to be used by the signer. It is possible, though computationally expensive, for the provider to anticipate a few d_i by forming $t \equiv m \cdot r^{e_1 e_2} \pmod{n}$ for example, and being able to unblind in case of signature with d_1 or d_2 by forming $m'_1 \equiv (m \cdot r^{e_1 e_2})^{d_1} \cdot r^{-e_2} \pmod{n}$ or $m'_2 \equiv (m \cdot r^{e_1 e_2})^{d_2} \cdot r^{-e_1} \pmod{n}$, depending on whether d_1 or d_2 was used to sign, respectively. But such an approach becomes prohibitively computation intensive as the number of alternatives increases: in general it requires the provider to perform more than one multiplication for each alternative anticipated, since each e_i should have a unique prime factor—otherwise some signatures can be made from others. Such effort required to anticipate all possible signatures may not be practical, and is also undesirable because the maximal extent of a system has to be fixed initially and effort required for this maximal extent has to be carried out from the beginning. Of course such an approach becomes impossible in practice when the number of alternatives is large or when the alternatives are not known in advance of blinding.

Even the simple payment system mentioned in [Chaum 85] has advantage in the bank's customers each supplying a large number of blinded items when they open an account, without the customer knowing in advance the particular choice of signature, which the bank will used to encode the denomination and possibly other data, such as expiration date, when it ultimately issues the notes. Credential systems encode each different kind of credential a person has as a digital signature of the corresponding type formed on a blinded copy of the persons digital "name" [Chaum 85; Chaum & Evertse 86]. There may be a great many different kinds of basic and subsidiary detail credentials, and the future requirement for such credentials may be difficult to anticipate at any particular instant. Thus there appears to be a substantial need in applications for blind signatures that are unblindable even after an unanticipated signature type has been applied.

Blind unanticipated signature protocols

The new blind signature protocols presented here are based on the use of one or more units (of the ring of residue classes modulo N) called ***generators***. Depending on how these generators are verified, as will be described in the next subsection, the blind unanticipated signature protocol will have one of three different forms.

In the simplest form of the protocol, the following congruences would hold:

$$t \equiv [m] \cdot g^k \pmod{n}$$

$$t' \equiv [m \cdot g^{k}]^{d_i} (\text{mod } n)$$

$$m' \equiv [(m \cdot g^{k})^{d_i}] \cdot g^{-d_i k} (\text{mod } n)$$

where, as before, m is the message to be blinded, t is the blinded form of the message, t' the signature computed for t, m' the unblinded signed m, g the generator, n the publicly known modulus, e_i and d_i public and private signature exponents respectively, and k the blinding key secret of the suplier chosen, say, uniformly from $\{1, \cdots, n^2\}$.

The square brackets show the input to the transformation whose output is shown on the left-hand-side, and thus they define the function of each of the three transformations in the order shown: blinding, signing, and unblinding. It is assumed that g^{d_i} is made public by the signer, which can be done without compromising security and need be done only once for each signature type i that is used. In the second form of the protocol, the following congruences hold:

$$t \equiv [m] \cdot g^{k_1} \cdot k_2 \ (\text{mod } n)$$

$$t' \equiv [m \cdot g^{k_1} \cdot k_2]^{d_i} (\text{mod } n)$$

$$m' \equiv [(m \cdot g^{k_1} \cdot k_2)^{d_i}] \cdot g^{-d_i k_1} \cdot k_2 (\text{mod } n),$$

where blinding key k_1 is as above and blinding key k_2 is chosen uniformly from $\{1, n-1\}$.

In the most general case, the following congruences would hold:

$$t \equiv [m] \cdot g_1^{k_1} \cdot g_2^{k_2} \cdot \ldots \cdot g_r^{k_r} (\text{mod } n)$$

$$t' \equiv [m \cdot g_1^{k_1} \cdot g_2^{k_2} \cdot \ldots \cdot g_r^{k_r}]^{d_i} (\text{mod } n)$$

$$m' \equiv [(m \cdot g_1^{k_1} \cdot g_2^{k_2} \ldots g_r^{k_r})^{d_i}] \cdot g_1^{-d_i k_1} \cdot \ldots \cdot g_r^{-d_i k_r} (\text{mod } n),$$

where each k_i is a secret blinding key chosen independently as k was above.

Testing a generator knowing the factorization

Knowing the factorization of $p-1$ and $q-1$ allows efficient verification that a particular proposed set of generators do in fact generate the whole group of units modulo n. (Notice, however, that these factorizations trivially allow one to compute p and q themselves.) Suppose, for example, that GCD$(p-1, q-1)=2$, which is quickly checked by comparing the factors of p and q, and that g and -1 are to be verified as generating the whole group. Then it is sufficient to verify

that the order of g is maximal modulo one factor of n, say p, and that it generates exactly half the elements modulo the other, q. This is readily accomplished: First raise g to $p_l^{-1} \cdot p_1 \cdot p_2 \cdot \ldots \cdot p_k$ and ensure that no l gives the result 1, where the p_i, $1 \le i \le k$, are the prime divisors of p and $1 \le l \le k$. Then repeat the same procedure for q, but exclude from the product the q_l equal to 2. The factors of n need be a secret of the signing party, however, since possessing them is sufficient to easily form signatures.

Hiding the factorization with physical security

This approach allows anyone to submit to the signer apparatus that conducts the kind of test just described. The signer supplies the secret parameters to the submitted apparatus and allows the logical result ("verifies" or "does not verify") to be communicated to the supplier of the apparatus, without allowing the secret parameters to be leaked by the apparatus to its supplier. One way to allow the signer to be sure that nothing is leaked to the supplier and to still provide confidence to the supplier is as follows: the supplier creates a number c' at random, applies a publically known and agreed on one-way function f, which is preferably one-to-one, to c', yielding $c = f(c')$; installs c' and n in the apparatus; and gives c to the signer along with the apparatus. Then the signer isolates the apparatus from the supplier and provides the apparatus the secret parameters (p_i and q_i); applies f to the output of the apparatus; and if this equals the original c supplied, returns this c' to the supplier. The apparatus is constructed to be tamper resistant enough to make it sufficiently difficult for the signer to obtain c' from it in the expected time interval, unless the generator tester yields a "verifies" output.

Linear cut and choose on modulus generator pairs

In another approach, a prospective signer makes public moduli N_i, $1 \le i \le k$ and corresponding generators G_i for each. Others determine the single modulus n among these that will be used, whereupon the signer must make public the factorizations of the $p-1$ and the $q-1$, for all the moduli except the one selected. Then anyone may use the verification techniques described above relying on possession of the secret to verify all the other moduli and corresponding generators, thereby obtaining the probability $1-k^{-1}$ that the selected modulus would also verify. One way to select the i would be a public event. Another way to determine i, requires each of some set of persons to form a b'_j at random and a corresponding $b_j = f(b'_j)$ using a preferably bijective public one-way function f. Then each person of the set makes public b_j. When all the b_j are public, the b'_j are revealed, checked, and added modulo the number of moduli, yielding the index i.

Generators not manipulable by the signer

Yet another way to allow those other than the signer to have confidence in the suitability of generators is for the generators to be determined in a way that cannot easily be manipulated by at least the signer. Generators could be chosen by a random process, such as those already described for selecting a particular modulus. Use of a single such randomly chosen and untested generator might not provide a high enough probability of providing adequate "unlinkability" as defined later; use of a plurality of randomly chosen generators improves the degree of hiding and unlinkability. For example, use of 20 or so generators provides a probability of roughly one-millionth that not all blinding factors in the reduced residue system modulo n can be generated. When additional things are known about the structure of n, such as that it has exactly two prime factors that are congruent to 3 modulo 4 [Peralta 86 & v.d. Gaaf 87], that $(p-1)(q-1)$ has no small odd divisors, or that GCD$(p-1, q-1)=2$, the probabilities improve greatly for the same number of generators. A further variation chooses subsets of a set of generators in a key dependent way.

Ensuring that messages blinded are powers of g

The previously described approach demonstrates that no suitability testing for generators is required. A variation under this approach still does not require verification of the generators: protection against linking is provided by almost any generator(s). The improvement derives from testing whether the thing to be signed, m, is a member of the group generated by the generator(s). When the m is actually issued by the signer, then the signer can form it as $m \equiv g^x$, where x is chosen at random by the signer. Then the signer can participate in a protocol that convinces the provider that some x satisfying the congruence is known to the signer. Such protocols for demonstrating possession of a discrete log are presented in [Chaum, Evertse, v.d. Graaf & Peralta 86].

Open questions

A much cleaner, more general, and perhaps even more efficient approach might be found. Notice, for example, that if $\frac{p-1}{2}$ and $\frac{q-1}{2}$ are primes, then any g with Jacobi symbol -1 (which can easily be checked only knowing n) together with -1 generates the whole group (apart from a few exceptional cases that immediately reveal the factorization of n). Thus, a protocol allowing anyone interacting with the signer to check that n is of such a form, without revealing its factorization, would allow easy verification of generators of the whole group. Any such "open" protocol for establishing a pair of generators (preferably one of which is -1) would be very desireable.

Unlinkability

Clearly all blinding factors cannot be equally likely with the techniques presented, since otherwise the blinder would have to choose the exponents uniformly from a distribution that is an exact multiple of $\phi(n)$. But it is easy to see, and will be formalized more completely in the final paper, that choosing the ks uniformly from $\{1, \cdots, nsubp2B)$ gives a maximum difference in the probability of any two that is less than n^{-1}.

Acknowledgements

I am indebted to J.-H. Evertse for substantial collaboration in related areas during the time that the basic results were developed. Subsequently, J.-H. Evertse, J. v.d. Graaf, and R. Peralta worked with the author to find improved techniques.

References

(1) Chaum, D., ''Blind signatures for untraceable payments'' *Advances in Cryptology: Proceedings of Crypto 82,* D. Chaum, R.L. Rivest and A.T. Sherman Eds., Plenum, NY, 1983, pp. 199-203.

(2) Chaum, D., ''Security without identification: transaction systems to make big brother obsolete,'' *Communications of the ACM,* 28, 10 (October 1985), pp. 1030-1044.

(3) Chaum, D. & Evertse, J.-H., ''A secure and privacy-protecting protocol for transmitting personal information between organizations,'' to appear in *Proceedings of Crypto 86.*

(4) Chaum, D., Evertse, J.-H., van de Graaf, J., & Peralta, R., ''Demonstrating possession of a discrete logarithm without revealing it,'' to appear in *Proceedings of Crypto 86.*

(5) v.d. Graaf, J., Private communication, 1987.

(6) Peralta, R., ''Three results in number theory and cryptography'' Ph.D. Thesis, University of California, Berkeley, 1986.

(7) Rivest, R., Shamir, A. and Adleman, L. ''A method for obtaining digital signatures and public-key cryptosystems'' *Communications of the ACM,* 21, 2, (February 1978), pp. 120-126.

VII

SYMMETRIC CIPHERS: THEORY

KEY-MINIMAL, PERFECT, LINEAR AND BILINEAR CIPHERS

James L. Massey
Ueli Maurer
Muzhong Wang

Institute for Signal and Information Processing
Swiss Federal Institute of Technology
8092 Zurich, Switzerland

Extended Abstract

The purposes of this paper are: (1) To give an appropriate general definition of a linear cipher, of a bilinear cipher, and of perfect secrecy for such ciphers; (2) to demonstrate for every blocklength the existence of perfect bilinear block ciphers that require the minimum possible amount of secret key; (3) to give some isolated examples of perfect linear stream ciphers with similarly minimum key that disprove an earlier conjecture of Massey and Rueppel; and (4) to suggest practical applications for such key-minimal perfect linear ciphers.

Only binary ciphers will be considered, i.e., the plaintext $\underline{x}$, the ciphertext $\underline{y}$ and the secret key $\underline{z}$ are all binary sequences whose components will be considered elements of finite field GF(2). In a block cipher, $\underline{x}$, $\underline{y}$ and $\underline{z}$ are finite sequences. The key is one-time, i.e., it is used to encipher a single ciphertext. The ciphering transformation, which must be invertible for every fixed choice of $\underline{z}$, will be written $\underline{y} = f(\underline{x},\underline{z})$.

The cipher will be called *linear* if the enciphering function is linear in $\underline{x}$ [with respect to the scalar field GF(2)] for every fixed choice of the key $\underline{z}$, i.e., if

$$f(\underline{x}_1 + \underline{x}_2,\underline{z}) = f(\underline{x}_1,\underline{z}) + f(\underline{x}_2,\underline{z})$$

for all $\underline{x}_1$, $\underline{x}_2$ and $\underline{z}$. The cipher will be called *bilinear* if the enciphering function is also linear in $\underline{z}$ for every fixed choice of the plaintext $\underline{x}$.

Shannon's notion of perfect secrecy (against a ciphertext-only attack) for a cipher system is that $\underline{x}$ and $\underline{y}$ be statistically independent. In a linear cipher, $\underline{x} = \underline{0}$ gives $\underline{y} = \underline{0}$ so such a cipher cannot be perfect unless $Pr(\underline{x} = \underline{0}) = 0$. Thus, we shall hereafter enforce the plaintext restriction $\underline{x} \neq \underline{0}$. Invertibility in a bilinear cipher also demands the key restriction $\underline{z} \neq 0$. Moreover, we shall say a linear cipher is perfect if, for a completely random key (chosen, of course, independently of $\underline{x}$), $\underline{x}$ and $\underline{y}$ are statistically independent for every choice of a probability distribution for $\underline{x}$ such that $Pr(\underline{x} = \underline{0}) = 0$. It is easy to see that Shannon's lower bound on the required secret key still holds, viz, a perfect linear cipher requires at least one bit of secret key per plaintext bit. A linear cipher meeting this lower bound with equality will be called key-minimal.

Consider a non-expanding linear block cipher of blocklength n, i.e., $\underline{x} = (x_1,\ldots,x_n)$ and $\underline{y} = (y_1,\ldots,y_n)$. If the cipher is key-minimal, then $\underline{z}$ also has length n. It is easy to see that the cipher is perfect if and only if, for a completely random key $\underline{z}$, $\underline{y}$ is also completely random for every choice of $\underline{x} \neq 0$. The following n = 2 bilinear cipher

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} z_1 & z_2 \\ z_2 & z_1+z_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 \\ x_2 & x_1+x_2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

is indeed legitimate because the z-matrix is invertible (i.e., nonsingular) for every $\underline{z} \neq \underline{0}$, and is perfect because the x-matrix is nonsingular for every $\underline{x} \neq \underline{0}$.

Key-minimal perfect bilinear ciphers exist for every block length n. Their construction is quite trivial. Let $\alpha_1,\ldots,\alpha_n$ be a basis for $GF(2^n)$ with respect to GF(2) and let $X = x_1\alpha_1 + \ldots + x_n\alpha_n$. Similarly define Y and Z as the elements of $GF(2^n)$ represented by $\underline{y}$ and $\underline{z}$. Then $Y = XZ$ is the equation of a key-minimal perfect bilinear cipher. The above example was constructed

this way taking $\alpha_1 = 1$ and $\alpha_2 = \alpha$ where $\alpha^2 + \alpha + 1 = 0$ in $GF(2^2)$.

At EUROCRYPT '84, Massey and Rueppel defined a linear additive stream cipher of memory M by the equation

$$y_i = x_i + f_i(x_{i-1},..,x_{i-M},\underline{z}) \stackrel{\Delta}{=} x_i + z'_i \quad (i \geq 1)$$

where f_i is linear in $(x_{i-1},...,x_{i-M})$ for every fixed choice of $\underline{z}$. Note that invertibility is always ensured and also that such a linear additive cipher can never be bilinear since $\underline{z} = \underline{0}$ does not ensure $\underline{x} = \underline{0}$. The appropriate plaintext restriction is $(x_{i-1},...,x_{i-M}) \neq \underline{0}$ for all i. They showed how to construct perfect ciphers of this type with 2 (new) bits of secret key per bit of plaintext and conjectured that this was the minimum possible. The following two examples (both due to the second author), however, use only 1 bit of key per bit of plaintext. That they are perfect linear ciphers follows from the fact that, for a completely random key, the additive sequence $z'_1, z'_2,...$ is completely random for every fixed choice of $\underline{x}$ satisfying the plaintext restriction.

M = 2 Stream Cipher:

$$\begin{bmatrix} z'_{3j+1} \\ z'_{3j+2} \\ z'_{3j+3} \end{bmatrix} = \begin{bmatrix} x_{3j-1} & x_{3j} & x_{3j} \\ x_{3j} & x_{3j}+x_{3j+1} & 0 \\ x_{3j+2} & 0 & x_{3j+1} \end{bmatrix} \begin{bmatrix} z_{3j+1} \\ z_{3j+2} \\ z_{3j+3} \end{bmatrix}.$$

M = 3 Stream Cipher:

$$z'_i = x_{i-1}\, z_{i+2} + x_{i-3}\, z_{i+1} + (x_{i-2} + x_{i-3}) z_i.$$

This M = 3 key-minimal perfect linear additive stream cipher can be written more suggestively as

$$z'_i = Tr(X_i Z_i)$$

where X_i and Z_i are the elements of $GF(2^3)$ represented by $(x_{i-1}, x_{i-2}, x_{i-3})$ and (z_{i+2}, z_{i+1}, z_i), respectively, with respect to the basis $\alpha_1 = \alpha^2$, $\alpha_2 = \alpha^3$, $\alpha_3 = \alpha^4$ where $\alpha^3 + \alpha^2 + 1 = 0$ in $GF(2^3)$, and where Tr(.) is the trace operator from $GF(2^3)$ to GF(2). [No similar perfect "multiplying cipher" exists for M = 2.]

At present, we do not know whether key-minimal perfect linear stream ciphers exist for M > 3.

Massey and Rueppel at Eurocrypt '84 showed the connection between perfect linear additive stream ciphers and good ensembles of convolutional codes. In theirs and our ciphers, the key "randomizes" the plaintext but also the plaintext "randomizes" the key. Thus these enciphering functions can be used to combine two pseudo-random sequences to produce a "better" pseudo-random sequence, a fact that Massey and Rueppel also exploited. Our M = 2 and M = 3 combiners should be superior to theirs as our "randomization" is better, but larger M's are needed for real practical utility.

Our perfect bilinear block ciphers have an advantage over the traditional one-time pad (where $\underline{y} = \underline{x} + \underline{z}$) in that an enemy who can alter $\underline{y}$ cannot predict the change in the resulting plaintext for our ciphers. Thus, an authentication pattern can be incorporated in $\underline{x}$ and used to foil such a substitution attack. Our linear ciphers could also be used (without a one-time key) to form secret (but insecure) pre-signatures for a plaintext $\underline{x}$ that would then be protected by an appropriate one-way function to form a cryptographic signature for appending to the plaintext. But perhaps most interestingly, the fact that the key and plaintext mutually randomize one another suggests that these perfect bilinear ciphers might well be used at some stages in a product cipher (where the other stages would, of course, incorporate nonlinear ciphering functions.)

LINEAR STRUCTURES IN BLOCKCIPHERS

Jan-Hendrik Evertse

Centre for Mathematics and Computer Science
Kruislaan 413 1098 SJ Amsterdam The Netherlands

1. INTRODUCTION

A blockcipher is a cipher which maps plaintexts, consisting of a fixed number of bits (zeros or ones), onto ciphertexts with the same number of bits under control of a key which also consists of a fixed number of bits. A well-known blockcipher is the NBS Data Encryption Standard (DES) ([NBS 77]). In [CE 85], a special class of linear structures in blockciphers, named "linear factors", was introduced. In this extended abstract, we deal with a general class of linear structures, which includes among others the linear factors and complementation properties like in DES. We discuss the crypt-analytic importance of these structures and consider the linear structures in DES.

Consider a blockcipher, and let P, K and C be fixed sets of plaintext bits, key bits and ciphertext bits of this blockcipher, respectively. A simultaneous complementation of the bits in P of a given plaintext and the bits in K of a given key will cause one of the following two changes in the exclusive-or sum of the bits in C of the corresponding ciphertext: either this exclusive-or sum is unchanged or it is complemented. In §2 we shall argue that blockciphers may be vulnerable to known- or chosen plaintext attacks faster than exhaustive key search if they possess *linear structures*, i.e. sets P, K and C of plaintext bits, key bits and ciphertext bits respectively, with the property that a simultaneous complementation of the plaintext bits in P and key bits in K results for *each* plaintext and key in the *same* change of the exclusive-or sum of the bits in C of the corresponding ciphertext. For instance, linear structures can be extracted from:

- the complementation property of DES (cf. [Hel 76]): simultaneous complementation of all plaintext bits and key bits results always in the complementation of all ciphertext bits;
- independencies of ciphertext bits of plaintext bits or key bits: some ciphertext bits can be expressed as boolean functions which do not have all plaintext bits and key bits as arguments; it was pointed out in [Me 78] and [CE 85] that truncated versions of DES with less

This research was supported by the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

than five rounds have such independencies;

- linear factors: structures which will change into independencies of ciphertext bits of plaintext bits or key bits when the blockcipher is modified by applying certain linear transformations (with respect to exclusive-or) to the plaintext, key and ciphertext, respectively.

We are particularly interested in linear structures of *product ciphers*. These are blockciphers composed of "simple" blockciphers ("rounds"). In §3 we explain how linear structures of product ciphers can be constructed from linear structures in their rounds; linear structures constructed in this way are said to be *recursive* over the rounds. Recursive linear structures in product ciphers are generalisations of the sequences of linear factors introduced in [CE 85]. In many situations, the linear structures of the rounds, and consequently the recursive linear structures of the product cipher, can be found quite easily; however it is often a hard problem to decide whether a product cipher has a linear structure not recursive over its rounds which is of any use in crypt-analysis.

By computing the linear structures in the rounds of DES (which is not too difficult), one can show that blockciphers consisting of seven or more consecutive rounds of DES have no recursive linear structures other than the complementation property. The precise result has been stated in §4.

In §5 we mention some possible extensions.

2. CRYPT-ANALYTIC SIGNIFICANCE OF LINEAR STRUCTURES

Let $\mathbb{F}_2=\{0,1\}$ be the finite field of two elements. When using notions from linear algebra such as linear spaces, linear mappings, etc., it is assumed that the underlying field of scalars is $\mathbb{F}_2$. $\mathbb{F}_2^m$ denotes the vector space containing all strings of m bits; we denote addition on $\mathbb{F}_2^m$, i.e. bitwise exclusive-or, by $+$. Elements of $\mathbb{F}_2^m$ are denoted by $\mathbf{a}$, $\mathbf{b}$, etc.; $\mathbf{0}_m$ denotes the string of m zeros and $\mathbf{1}_m$ the string of m ones. Vectors in cartesian products $\mathbb{F}_2^{m_1}\times\cdots\times\mathbb{F}_2^{m_r}$ are often denoted as $(\mathbf{x}_1,\ldots,\mathbf{x}_r)$, where $\mathbf{x}_i\in\mathbb{F}_2^{m_i}$ for $i=1,\ldots,r$. $[\mathbf{x}]$ denotes the vector space generated by $\mathbf{x}$. If $V_\alpha(\alpha\in A)$ are subspaces of the same vector space, then $\bigoplus_{\alpha\in A} V_\alpha$ denotes the smallest vector space containing each V_α. For any linear mapping A with domain $\mathbb{F}_2^m$ we put $ker(A)=\{\mathbf{x}\in\mathbb{F}_2^m: A\mathbf{x}=\mathbf{0}\}$ and $im(A)=\{A\mathbf{x}: \mathbf{x}\in\mathbb{F}_2^m\}$.

A *blockcipher* is a mapping

$$F: \mathbb{F}_2^m\times\mathbb{F}_2^k\to\mathbb{F}_2^m$$

(where $\mathbb{F}_2^m$ and $\mathbb{F}_2^k$ are the *message space* and *key space*, respectively) such that for each $\mathbf{k}$ in $\mathbb{F}_2^k$, the mapping

$$F_{\mathbf{k}}:=F(.,\mathbf{k}): \mathbb{F}_2^m\to\mathbb{F}_2^m \tag{1}$$

is invertible.

Definition 1. A *linear structure* of a blockcipher $F: \mathbb{F}_2^m \times \mathbb{F}_2^k \to \mathbb{F}_2^m$ is a pair of linear mappings (A,B) with domains $\mathbb{F}_2^m \times \mathbb{F}_2^k$ and $\mathbb{F}_2^m$, respectively, such that for each pair $(\mathbf{p_0},\mathbf{k_0})$ in $ker(A)$ and each $\mathbf{p}$ in $\mathbb{F}_2^m$ and $\mathbf{k}$ in $\mathbb{F}_2^k$ we have

$$BF(\mathbf{p}+\mathbf{p_0},\mathbf{k}+\mathbf{k_0})+BF(\mathbf{p},\mathbf{k}) = BF(\mathbf{p_0},\mathbf{k_0})+BF(\mathbf{0}_m,\mathbf{0}_k)\,. \tag{2}$$

Example 1. Complementation properties. Let (A,B) be a linear structure of the blockcipher $F: \mathbb{F}_2^m \times \mathbb{F}_2^k \to \mathbb{F}_2^m$ where B is the identity. Then there is a function ψ, defined on $ker(A)$ such that $F(\mathbf{p}+\mathbf{p_0},\mathbf{k}+\mathbf{k_0})+F(\mathbf{p},\mathbf{k})=\psi(\mathbf{p_0},\mathbf{k_0})$ for each $(\mathbf{p_0},\mathbf{k_0})$ in $ker(A)$, $\mathbf{p}$ in $\mathbb{F}_2^m$ and $\mathbf{k}$ in $\mathbb{F}_2^k$. (For instance, for DES: $\mathbb{F}_2^{64} \times \mathbb{F}_2^{56} \to \mathbb{F}_2^{64}$ one may take for A a linear mapping with $ker(A)=[(\mathbf{1}_{64},\mathbf{1}_{56})]$; then $\psi(\mathbf{1}_{64},\mathbf{1}_{56})=\mathbf{1}_{64}$). In [Hel 76], chap. III it was explained how to use the complementation property of DES in a chosen plaintext attack which is twice as fast as exhaustive key search. By a similar argument one can show that the blockcipher F is vulnerable to a chosen plaintext attack N times as fast as exhaustive key search if $ker(A)$ has cardinality $\geqslant N$.

Example 2. Linear factors. A linear factor of the blockcipher F is a triple of linear mappings (A_1,A_2,B) for which a mapping $\tilde{F}$ can be found with $BF(\mathbf{p},\mathbf{k})=\tilde{F}(A_1\mathbf{p},A_2\mathbf{k})$ for all plaintexts $\mathbf{p}$ and keys $\mathbf{k}$. In [CE 85], it was explained how such linear factors enable known-plaintext attacks faster than exhaustive key search.

In known or chosen plaintext attacks on blockciphers with linear structures one has to use the following fact (stated without proof):

Lemma 1. *Let $F: \mathbb{F}_2^m \times \mathbb{F}_2^k \to \mathbb{F}_2^m$ be a blockcipher and (A,B) a linear structure of F. Then there exist a linear mapping $C: \mathbb{F}_2^m \times \mathbb{F}_2^k \to im(B)$ and a (not necessarily linear) mapping $\tilde{F}: im(A) \to im(B)$, both easily computable from F, A and B, such that*

$$BF(\mathbf{p},\mathbf{k})=\tilde{F}(A(\mathbf{p},\mathbf{k}))+C(\mathbf{p},\mathbf{k}) \quad \textit{for all } \mathbf{p} \textit{ in } \mathbb{F}_2^m,\ \mathbf{k} \textit{ in } \mathbb{F}_2^k\,.$$

We now give an example of a known plaintext attack which uses linear structures. Let $F: \mathbb{F}_2^m \times \mathbb{F}_2^k \to \mathbb{F}_2^m$ be a blockcipher and let (A,B) be a linear structure of F. Let C and $\tilde{F}$ be the mappings satisfying the conditions of Lemma 1 and define the linear mappings A_1, A_2, C_1 and C_2 by $A(\mathbf{p},\mathbf{k})=A_1\mathbf{p}+A_2\mathbf{k}$, $C(\mathbf{p},\mathbf{k})=C_1\mathbf{p}+C_2\mathbf{k}$. Suppose that $0<n:=$dimension $ker(A_2)\leqslant k$. (Linear structures in which $ker(A_2)$ has dimension 0 but $ker(A)$ has dimension >0 can be used in a chosen plaintext attack, which we hope to describe in a later version of this paper). Suppose that a crypt-analist has a plaintext-ciphertext pair $(\mathbf{p},\mathbf{c})$, where $\mathbf{c}=F(\mathbf{p},\mathbf{k})$ for some secret key $\mathbf{k}$. In order to find $\mathbf{k}$, he proceeds as follows:

(i) he runs through all values $\tilde{\mathbf{k}}$ in $im(A_2)$ and checks for each $\tilde{\mathbf{k}}$, if the system of linear equations

$$\left.\begin{aligned} A_2\mathbf{k}&=\tilde{\mathbf{k}}\\ C_2\mathbf{k}&=B\mathbf{c}+\tilde{F}(A_1\mathbf{p}+\tilde{\mathbf{k}})+C_1\mathbf{p}\end{aligned}\right\} \quad \text{in } \mathbf{k}\in\mathbb{F}_2^k \tag{3}$$

is soluble (the costs of this are approximately those of a computation of F, if we suppose that F is much more "complicated" than a linear mapping); it follows at once from Lemma 1 that the unknown key $\mathbf{k}$ must satisfy (3);

(ii) for each $\tilde{\mathbf{k}}$ in $im(A_2)$ for which (3) is soluble, he checks if $F(\mathbf{p},\mathbf{k})=\mathbf{c}$ for each solution $\mathbf{k}$ of (3).

Supposing that the crypt-analist finds L values of $\tilde{\mathbf{k}}$ in (i), and that the null space of the linear mapping $\mathbf{k}\mapsto(A_2\mathbf{k},C_2\mathbf{k})$ has dimension $n_1\leqslant n$, he will find the key after about $2^{k-n}+L\times 2^{n_1}$ encryptions. In general, this is smaller than the number of encryptions needed in exhaustive key search, 2^k. By a heuristic argument like in [CE 85], chap. 2, one can argue that the expected time in which the crypt-analist finds the key can be made smaller if he has more plaintext-ciphertext pairs.

3. LINEAR STRUCTURES IN PRODUCT CIPHERS

Let $F_1,\ldots,F_R\colon \mathbb{F}_2^m\times\mathbb{F}_2^k\to\mathbb{F}_2^m$ be blockciphers. The *product* $F=F_R\cdots F_1$ of $F_1,\ldots,F_R$ is defined (cf. (1)) by

$$F_{\mathbf{k}}(\mathbf{p})=F_{R,\mathbf{k}}\cdots F_{1,\mathbf{k}}(\mathbf{p}) \tag{4}$$

(composition of mappings) for $\mathbf{p}\in\mathbb{F}_2^m$ and $\mathbf{k}\in\mathbb{F}_2^k$. $F_1,\ldots,F_R$ are called the rounds of F. We shall describe, in Lemma 2 below, how linear structures of F can be constructed from linear structures in $F_1,\ldots,F_R$.

For any blockcipher $F\colon \mathbb{F}_2^m\times\mathbb{F}_2^k\to\mathbb{F}_2^m$, and any subspace $\mathcal{V}$ of $\mathbb{F}_2^m\times\mathbb{F}_2^k$, we define the spaces

$$T(F,\mathcal{V})=\bigoplus_{\substack{(\mathbf{p}_0,\mathbf{k}_0)\in\mathcal{V}\\(\mathbf{p},\mathbf{k})\in\mathbb{F}_2^m\times\mathbb{F}_2^k}}[(F(\mathbf{p}+\mathbf{p}_0,\mathbf{k}+\mathbf{k}_0)+F(\mathbf{p},\mathbf{k}),\mathbf{k}_0)]\,,$$

$$U(F,\mathcal{V})=\bigoplus_{\substack{(\mathbf{p}_0,\mathbf{k}_0)\in\mathcal{V}\\(\mathbf{p},\mathbf{k})\in\mathbb{F}_2^m\times\mathbb{F}_2^k}}[F(\mathbf{p}+\mathbf{p}_0,\mathbf{k}+\mathbf{k}_0)+F(\mathbf{p},\mathbf{k})+F(\mathbf{p}_0,\mathbf{k}_0)+F(\mathbf{0}_m,\mathbf{0}_k)]\,.$$

Thus $T(F,\mathcal{V})\subseteq\mathbb{F}_2^m\times\mathbb{F}_2^k$, $U(F,\mathcal{V})\subseteq\mathbb{F}_2^m$, and, by (2),

$$(A,B)\text{ linear structure of }F\Leftrightarrow U(F,ker(A))\subseteq ker(B)\,. \tag{5}$$

Lemma 2. *Let $F_1,\ldots,F_R\colon \mathbb{F}_2^m\times\mathbb{F}_2^k\to\mathbb{F}_2^m$ be blockciphers and put $F=F_R\cdots F_1$. Suppose that $\mathcal{V}_0,\mathcal{V}_1,\ldots,\mathcal{V}_R$ are subspaces of $\mathbb{F}_2^m\times\mathbb{F}_2^k$, and $\mathcal{W}_0,\mathcal{W}_1,\ldots,\mathcal{W}_R$ are subspaces of $\mathbb{F}_2^m$, such that*

$$\left.\begin{aligned} &\mathcal{V}_i \supseteq T(F_i,\mathcal{V}_{i-1}),\\ &\mathcal{W}_0 = [\mathbf{0}_m],\\ &\mathcal{W}_i\times[\mathbf{0}_k] \supseteq T(F_i,\mathcal{W}_{i-1}\times[\mathbf{0}_k])\oplus U(F_i,\mathcal{V}_{i-1})\times[\mathbf{0}_k] \text{ for } i=1,\ldots,R. \end{aligned}\right\} \quad (6)$$

Then $U(F,\mathcal{V}_0)\subseteq\mathcal{W}_R$.

We shall not give the elementary proof of Lemma 2 in this abstract.

(5) and Lemma 2 motivate the following:

Definition 2. Let $F_1,\ldots,F_R$: $\mathbb{F}_2^m\times\mathbb{F}_2^k\to\mathbb{F}_2^m$ be blockciphers and put $F=F_R\cdots F_1$. A linear structure (A,B) of F is called *recursive* over $F_1,\ldots,F_R$ if there are subspaces $\mathcal{V}_0,\ldots,\mathcal{V}_R$ of $\mathbb{F}_2^m\times\mathbb{F}_2^k$ and $\mathcal{W}_0,\ldots,\mathcal{W}_R$ of $\mathbb{F}_2^m$ for which (6) holds and for which $ker(A)=\mathcal{V}_0$ and $ker(B)=\mathcal{W}_R$.

It is an interesting open problem how to find out whether a product cipher has linear structures which are not recursive over its rounds.

4. LINEAR STRUCTURES IN DES

For convenience we modify DES a little bit: we do not use the tables IP and PC1 in the NBS-description (cf. [NBS 77]) and combine the tables E and P, in the way described in [Dav 83], chap. 3. Plaintexts and ciphertexts of DES are denoted by $(\mathbf{p},\mathbf{q})$, where $\mathbf{p},\mathbf{q}\in\mathbb{F}_2^{32}$.

DES is composed of the following mappings (cf. [NBS 77], [CE 85]):
P:$\mathbb{F}_2^{32}\to\mathbb{F}_2^{32}$: bit permutation;
E:$\mathbb{F}_2^{32}\to\mathbb{F}_2^{48}$: bit expansion;
S_l:$\mathbb{F}_2^6\to\mathbb{F}_2^4$ ($l=1,\ldots,8$): S-boxes;
S:$\mathbb{F}_2^{48}\to\mathbb{F}_2^{32}$: $S(\mathbf{x}_1,\ldots,\mathbf{x}_8)=(S_1\mathbf{x}_1,\ldots,S_8\mathbf{x}_8)$ for $\mathbf{x}_1,\ldots,\mathbf{x}_8\in\mathbb{F}_2^6$;
L_i:$\mathbb{F}_2^{56}\to\mathbb{F}_2^{48}$ ($i=1,\ldots,16$): key scheduling; all L_i are permuted choices of key bits, defined by PC2 and the shifting pattern.

Let F_i: $\mathbb{F}_2^{32}\times\mathbb{F}_2^{32}\times\mathbb{F}_2^{56}\to\mathbb{F}_2^{32}\times\mathbb{F}_2^{32}$ be the i-th round of DES, defined by

$$F_i(\mathbf{p},\mathbf{q},\mathbf{k})=(\mathbf{q},\mathbf{p}+S(EP\mathbf{q}+L_i\mathbf{k})) \quad \text{for } i=1,\ldots,16$$

and $DES_{RS}=F_S\cdots F_R$. The next theorem states that product ciphers, composed of seven or more consecutive rounds of DES, have no "non-trivial" recursive linear structures other than the complementation property.

THEOREM. *Let R, S be integers with* $1\leqslant R<S\leqslant 16$ *and* $S\geqslant R+6$, *and let* (A,B) *be a linear structure of* DES_{RS} *such that* $ker(A)$ *is not equal to* $[(\mathbf{0}_{64},\mathbf{0}_{56})]$ *or* $[(\mathbf{1}_{64},\mathbf{1}_{56})]$ *and* (A,B) *is recursive over* $F_R,\ldots,F_S$. *Then* $ker(B)=\mathbb{F}_2^{64}$.

Very rough sketch of proof. Express the spaces $T(F_i,\mathcal{V})$ and $U(F_i,\mathcal{V})$ in terms of linear structures of the S-boxes, for each subspace $\mathcal{V}$ of $\mathbb{F}_2^{64}\times\mathbb{F}_2^{56}$ and each i in $\{1,\ldots,16\}$, and compute the linear structures in the S-boxes. An S-box S_l is said to have a linear structure if there are

$\mathbf{a} \in \mathbb{F}_2^6$ and a linear mapping $D: \mathbb{F}_2^4 \to \mathbb{F}_2$ such that $DS_l(\mathbf{x}+\mathbf{a})+DS_l(\mathbf{x})$ assumes the same value (0 or 1) for each $\mathbf{x} \in \mathbb{F}_2^6$.

5. POSSIBLE EXTENSIONS

In [Hel 76], chap. 4, Hellman et. al. suggested the following way to break DES: change a few outputs of the S-boxes of DES such that the resulting blockcipher DES', with the modified S-boxes, is easy to break. Then DES and DES' give the same ciphertexts for a non-negligible fraction of pairs of plaintexts and keys. For these plaintexts and keys, the key in DES can be found by searching for the key in DES'.

As mentioned in §4, recursive linear structures in DES can be obtained from linear structures in the S-boxes, and the same is true for DES'. If each S-box in DES were chosen in such a way that for any set of inputbits and any set of outputbits of this box, a simultaneous complementation of the inputbits in the given set results in about 50% of the cases in a complementation of the exclusive-or sum of the outputbits in the given set, then each S-box would get a linear structure only after a change of about 16 of its outputs, in other words no S-box would have "near" linear structures.

The S-boxes in DES have not been chosen in this way; for instance S-box 4 has linear structures (cf. [Hel 76], chap. 5 and [CE 85], chap. 4, Lemma 4). Not all near linear structures in the S-boxes will be of use in crypt-analysis; it is still an open problem whether the S-boxes in DES have any useful near linear structures.

REFERENCES

[CE 85] Chaum, D. and Evertse, J.-H, *Cryptanalysis of DES with a reduced number of rounds; sequences of linear factors in block ciphers,* in Advances in Cryptology: Proc. Crypto '85, H.C. Williams, ed., Lecture Notes in Computer Science 218, Springer Verlag, Berlin etc. (1986), pp. 192-211.

[Dav 83] Davio, M., Desmedt, Y., Fosseprez, M., Govaerts, R., Hulsbosch, J., Neutjens, P., Piret, P., Quisquater, J.J., Vandewalle, J., Wouters, P., *Analytical characteristics of the DES,* in Advances in Cryptology: Proc. Crypto '83, D. Chaum, ed., Plenum, New York (1984), pp. 171-202. pp. 359-376.

[Hel 76] Hellman, M., Merkle, R., Schroeppel, R., Washington, L., Diffie, W., Pohlig, S., Schweitzer, P., *Results of an initial attempt to cryptanalyze the NBS Data Encryption Standard,* Information Systems Lab. report SEL 76-042, Stanford University (1976).

[Me 78] Meyer C.H. *Ciphertext / plaintext and ciphertext / key dependencies vs. number of rounds for the Data Encryption Standard,* AFIPS Conference Proceedings, 47, (June 1978), pp. 1119-1126.

[NBS 77] National Bureau of Standards, *Data Encryption Standard,* U.S. Department of Commerce, FIPS pub. 46 (January 1977).

Fast Data Encipherment Algorithm FEAL

Akihiro Shimizu Shoji Miyaguchi
NTT Electrical Communications Laboratories, Yokosuka-shi, 238-03 Japan.

BACKGROUND

In data communication and information processing systems, cryptography is the most effective way to secure communications and stored data. The most commonly used cryptographic method is DES [1]. However, it is generally implemented with hardware, and the cost is prohibitive for small scale systems such as personal computer communication. Accordingly, an encipherment algorithm that has safety equal to DES and is suitable for software as well as hardware implementation is needed. The FEAL (Fast Data Encipherment Algorithm) fills the need.

EVALUATION INDICES FOR ALGORITHM STRENGTH

In the FEAL design, evaluation indices [2] are adopted to evaluate objectively the data randomization ability of the algorithm. The indices express the approximation degree of ciphertext variation distribution for input plaintext or key variations to the binomial distribution $B(n,1/2)$, in which n is the ciphertext bit length. Two indices, M and M_σ are used.

M is the average approximation degree of the distribution of ciphertext variations according to the plaintext or key variations from one-bit to n-bit. M_σ is the variance of the approximation degree. When M approaches 100 and M_σ approaches zero, the algorithm does not leave clues which could be used to count backward to the input plaintext or key, in the ciphertext. M and M_σ are defined separately so that M_P and $M_{P\sigma}$ are for the plaintext variations and M_K and $M_{K\sigma}$ are for the key variations.

To get the indices, many plaintexts or keys have to be used. Nevertheless, the amount of data which can be treated is generally small compared to the population. Thus, it is important to get the theoretical index values according to the amount of data by means of statistical calculation. For example, the theoretical values for 4096 pieces of data, which are a combination of 16 plaintexts, 16 keys and 16 plaintext or key variations, are $M = 96.5$ and $M_\sigma = 2.6$ (Table 1). When the measured values of the indices are close to the theoretical values, the algorithm is considered effective.

DESIGN

FEAL consists of two processing parts (Fig. 1). One is the key schedule which generates the 192-bit intermediate key from the 64-bit secret key. It is designed to generate different intermediate keys for different secret keys. The other is the data randomizer, which generates 64-bit ciphertext from 64-bit plaintext under control of the intermediate key. The data randomizer uses combination of involutions [3]. One program can perform two func-

tions, enciphering and deciphering, except for the intermediate key entry order. Moreover, the setting of 64-bit intermediate keys by means of an exclusive OR operation at the entrance and exit make attack on the algorithm difficult.

FEAL's f function (Fig. 2) is especially designed to overcome the problems existing in f of DES. The construction of f is such that input bit variations influence all output data. Experiment confirmed that FEAL's f function randomization efficiency is three times that of DES.

The s function in the f function, a one byte data substitution, is as effective as DES's S-BOX. The s function is defined as:

$s(x, y, \delta) = ROL2((x+y+\delta) \bmod 256)$;

x, y: one byte data; δ : constant (0 or 1);

ROL2: 2-bit circular function to the left.

The f_K function (Fig. 3) used in the key schedule is same as the f function except for the entry position of parameter β .

STRENGTH AND PERFORMANCE

FEAL is safe from the all-key attack because it is controlled by a 64-bit key, more secure than the 56-bit DES key. There is no more effective method than the all-key attack to get the secret key from the plaintext and the ciphertext. The FEAL structure is considered strong because the entire secret key can not be got even if any part, under 64 bits, of the secret key or intermediate key are already known. Regarding ciphertext randomization, FEAL is considered safe because the randomization indices are closer to the theoretical values than those of DES.

Next, FEAL performance other than the randomization indices is compared with the DES performance in Table 1. When FEAL is implemented in assembly language on a i-8086 16-bit microprocessor with 8MHz clock, it is confirmed that the program size is under 400 bytes and the execution speed reaches 150 Kbit/sec.. Compared to DES, the program size is one sixth as large, and 75 times faster.

CONCLUSION

FEAL is a safe encipherment algorithm suitable for software implementation. It can be applied widely to small scale or other existing systems unable to use DES hardware because of cost. Moreover, FEAL is suitable for hardware implementation, too. Implemented as an LSI, it can be used as the cryptographic method in all data communication fields.

REFERENCES

[1] FIPS PUB 46, Data Encryption Standard (1977)

[2] S. Miyaguchi, M. Hirano : "Evaluation Criteria for Encipherment and Authentication Algorithms", Trans. of IECE of Japan, J69-A, No. 10, pp. 1252-1259 (Oct. 1986) (in Japanese)

[3] A. G. Kohnheim : "Cryptography : A Primer", A Wiley Interscience Publication, pp. 236-240 (Jan. 1986)

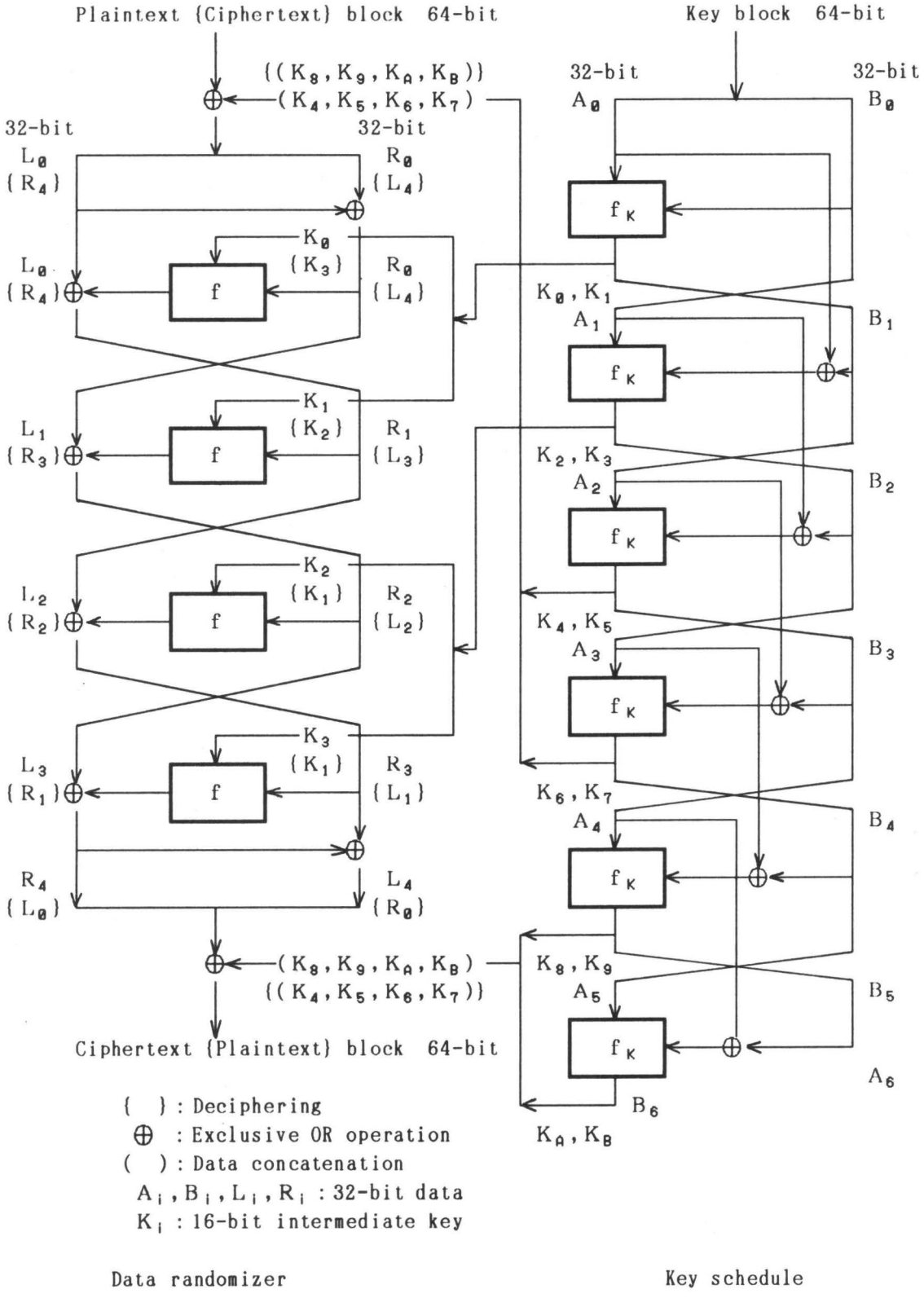
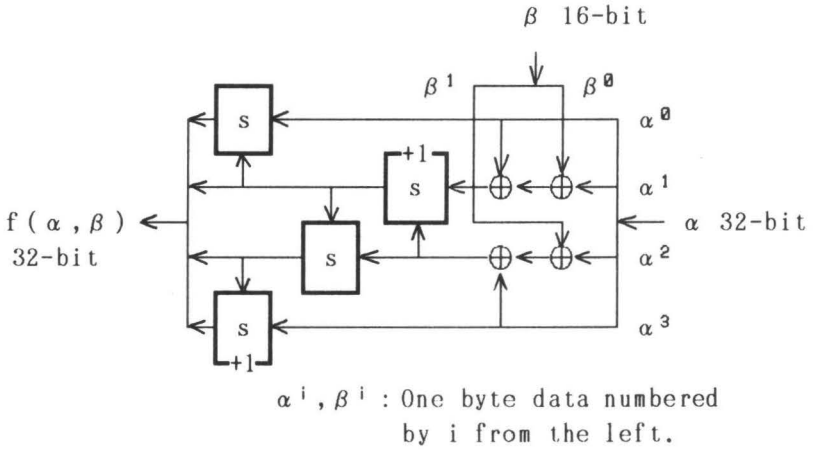


Fig. 1 FEAL Algorithm

Table 1 Performance comparison of FEAL and DES

Items of performance		FEAL	DES	Theoretical values
Randomization indices	M_K	97.1	93.4	96.5
	$M_K\ \sigma$	2.1	4.9	2.6
Data amount 16^3	M_P	96.8	95.5	96.5
	$M_P\ \sigma$	2.4	3.4	2.6
Program size (Byte)		390	2300	—
Execution speed (Kbps)		150	2	—

(μP i-8086, Clock 8MHz)



α^i, β^i : One byte data numbered by i from the left.

Fig. 2 Function f

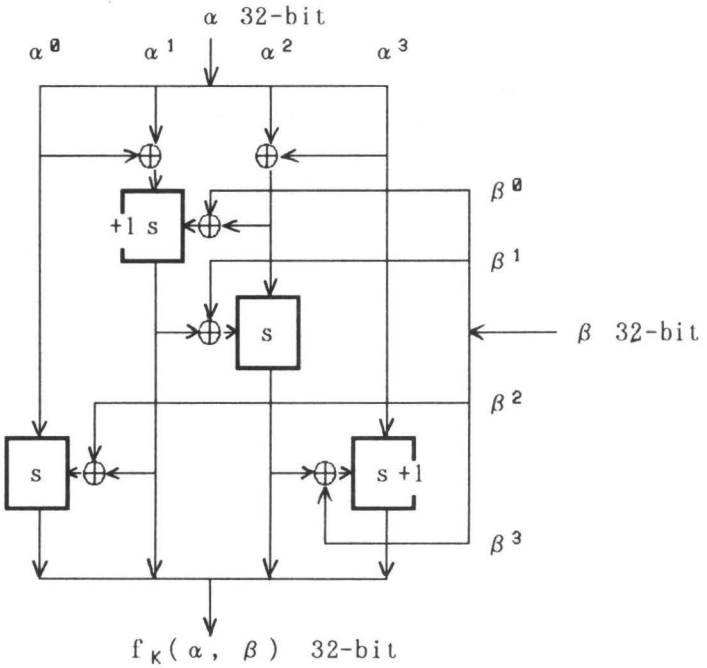


Fig. 3 Function f_K

VIII

SYMMETRIC CIPHERS: APPLICATION

MODES OF BLOCKCIPHER ALGORITHMS AND THEIR PROTECTION AGAINST ACTIVE EAVESDROPPING

Cees J.A. Jansen
Philips USFA B.V.
Eindhoven, The Netherlands

Dick E. Boekee
Delft University of Technology
Delft, The Netherlands

Abstract.

Blockcipher algorithms are used in a variety of modes for message encryption and/or message authentication. We present an overview of a number of modes and their protection against active eavesdropping. In particular we will discuss the consequences of addition, deletion and repetition of parts of the ciphertext in the different modes.
Also a possible application of a method introduced at Eurocrypt 86 will be shown in detail. This method is based on pseudo-randomly selecting one out of a number of invertible functions of two characters (blocks). In this application simple functions are used, which are easy to implement on a microprocessor.
The method, which may be regarded as a combination of the OFB and ECB modes as are known with the Data Encryption Standard, offers a good protection against the forms of active eavesdropping mentioned here and has no error extension over the block boundary.

Let a blockcipher be given by its encryption and decryption operators E_k and D_k, which act on m-bit blocks under a key k.

Then the following modes are possible:

ECB:	$C_n = E_k(M_n)$	$M_n = D_k(C_n)$
CBC:	$C_n = E_k(M_n + C_{n-1})$	$M_n = D_k(C_n) + C_{n-1}$
CFB:	$C_n = M_n + E_k(C_{n-1})$	$M_n = C_n + E(C_{n-1})$
OFB:	$C_n = M_n + R_n$ $R_n = E_k(R_{n-1})$	$M_n = C_n + R_n$
PBC:	$C_n = E_k(M_n) + M_{n-1}$	$M_n = D_k(C_n + M_{n-1})$
PFB:	$C_n = M_n + E_k(M_{n-1})$	$M_n = C_n + E_k(M_{n-1})$
CBCPD:	$C_n = E_k(M_n + M_{n-1} + C_{n-1})$	$M_n = D_k(C_n) + C_{n-1} + M_{n-1}$
OFBNLF:	$C_n = f_{R_n}(M_n)$ $R_n = E_k(R_{n-1})$	$M_n = f^{-1}_{R_n}(C_n)$

Here C_n and M_n denote the n^{th} ciphertext and plaintext blocks; R_n is the n^{th} block of pseudo-random bits and $f_{R_n}(M_n)$ is the $R_n{}^{th}$ invertible function acting on plaintext block M_n.
The first four modes are well known with the DES.
The other four modes denote Plaintext Block Chaining, Plaintext FeedBack, Cipher Block Chaining of Plaintext Difference and Output FeedBack with Non-Linear Function respectively.

From the equations one can easily see what happens if the n^{th} ciphertext block is deleted, repeated or added to some block S_n.
It turns out that active eavesdropping is not possible with the OFBNLF mode, but occasional errors in the ciphertext will not give rise to block-error extension in the plaintext.

As an application of a method introduced at eurocrypt 86 by the first author, consider the following situation. Eight bit characters are encrypted by first cyclically shifting them N times (0 - 7, so 3 bits needed to indicate the shift) and then adding a pseudo-random byte to it in one out of eight ways. The eight ways of addition could be the following:

- 8 bits mod 2
- 4x2 bits mod 4
- 2x3 bits mod 8 + 1x2 bits mod 4
- 2x4 bits mod 16
- 1x5 bits mod 32 + 1x3 bits mod 8
- 1x6 bits mod 64 + 1x2 bits mod 4
- 1x7 bits mod 128 + 1 bit mod 2
- 1x8 bits mod 256.

The 8+2x3 = 14 pseudo-random bits can be obtained for example from the DES in OFB mode. This also gives the possibility of extra key bits and encryption of up to four characters in parallel.

Security Considerations in the Design and Implementation of a new DES chip.*

I. Verbauwhede[1] F. Hoornaert[2] J. Vandewalle[3]
H. De Man[1,3] R. Govaerts[3]

10 Jan 1987

IMEC v.z.w.[1]	CRYPTECH n.v.[2]	ESAT, K.U.Leuven[3]
Kapeldreef 75	Lloyd George Av. 6	K. Mercierlaan 94
B-3030 Heverlee	B-1050 Brussel	B-3030 Heverlee
Belgium	Belgium	Belgium
Tel: 32-(0)16-281211	Tel: 32-(0)2-6425931	Tel: 32-(0)16-220931

1 Introduction.

This paper describes the impact of cryptographic requirements on the design of a new powerful DES chip implementation. On the one hand security and flexibility are required by cryptographers. On the other hand design, implementation and test restrict the feasibility. It is the aim of this contribution to show how both can be combined. The result is a *single* chip implementation of the DES algorithm [1] with a number of unique features.

2 Enhanced security : required by cryptographers.

Since there are questions around the safety of DES, the first objective is to implement *DES as well as DES like* algorithms. One could ask for other S-boxes or for an increase of the number of rounds. One could use another key scheduling scheme or a longer key by using more than one key for one DES calculation.

All MODES have to be realised *ON* chip. One has provided all modes as described in [2] in 8 bytes form (ECB, CBC, CFB, OFB) and in 1 byte form (CFB, OFB).

Key safety requires that, once entered, keys can not leave the chip anymore [3]. So one should provide at least 4 key registers for 2 master and 2 session keys. Key exoring of incoming keyparts is possible and parity is checked on chip.

Triple encryption is executed on a single command. And enough keyregisters must be provided.

More general, system requirements are that a cryptographic device should be *fast* and *easy to use*. The insertion of an encryption device may not slow down the overall performance of a system and it must be usable in a large number of environments. One should implement user friendly

*Research performed in collaboration with the company Cryptech n.v.

and powerful commands. However, a general reset, especially for the keys, must be available when someone tries to tamper.

3 Implementation and test requirements.

From the design point of view an optimal trade-off has to be made between algorithm, speed and chip area, while keeping the chip testable.

To enhance *speed*. first the *off chip communication* should be minimized. This means that all tasks are fulfilled on chip. Powerful commands are necessary so the number of commands for one execution is limited. Second *pipelining* raises speed. Hereby one calculates independent things in parallel as much as possible. E.g., the previous ciphertext is written out while the actual plaintext is being enciphered and the next one is read in. At first sight this seems to be in conflict with the feedback modes. But this is not the case, as explained further and extensively in [4].

Area is expensive and should be kept minimal. So the routing must be reduced and a good floorplan is essential. E.g., the 16 DES rounds are calculated in sequence on the same hardware part.

When designing a chip one will always end up with *testing*. The following three tests are necessary : 1) functional and timing tests during development, 2) validation tests, 3) maintenance tests during lifetime.

4 Chip Architecture.

4.1 Floorplan.

In order to optimize the trade-off between security and chip design requirements the floorplan (fig 1.) is divided in four independent parts :

1. The *DES part*. half of the area, contains hardware for one DES round.
2. The *Key part*. one quarter of the area, consists of four key registers and the key scheduling scheme, included the key permutation PC2.
3. The *Modes and Internal Transport part* ensures a fast calculation of the modes on chip. This is done while exchanging the newly entered data going from the IO part to the DES part and the enciphered data going the opposite way from the DES part to the IO part (fig. 2). At the encounter they can be exored. On figure 2 this is explained for CBC. This fast internal calculation in a very small amount of time is done between every two timesteps of the pipeline [4].
4. The *Input Output (IO) part* realises the off chip communication. The Modes and IO part take one quarter of the actual chip.

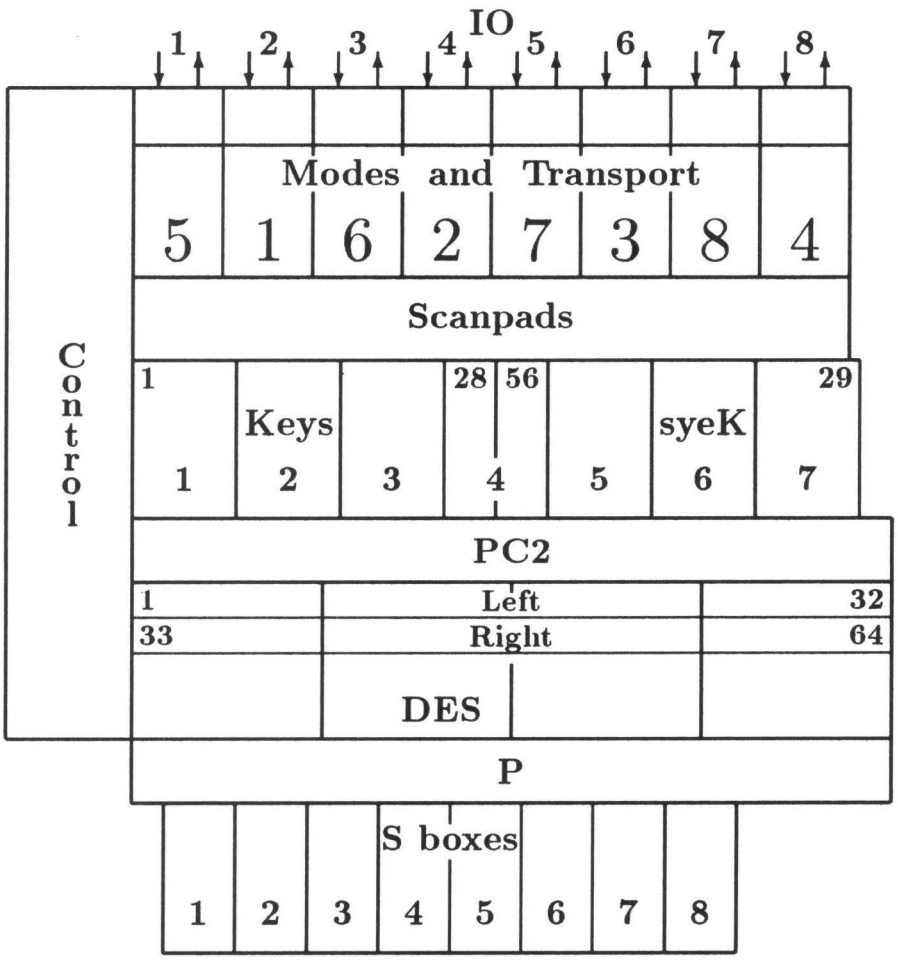


fig.1 Floorplan

4.2 Modular design : Divide and Conquer.

The four parts of the floorplan are designed independently, work independently and only communicate with each other through one common register which is strictly watched.

Each part has its own local controller. So if one decides to implement a DES like algorithm, one has just to plug in another local DES part controller.

This modularity is also reflected in the floorplan. And the routing between the blocks is minimal.

A modular design is *easily changeable*. A change has only local influence. E.g., one can easily cut the actual S-boxes and replace these by others.

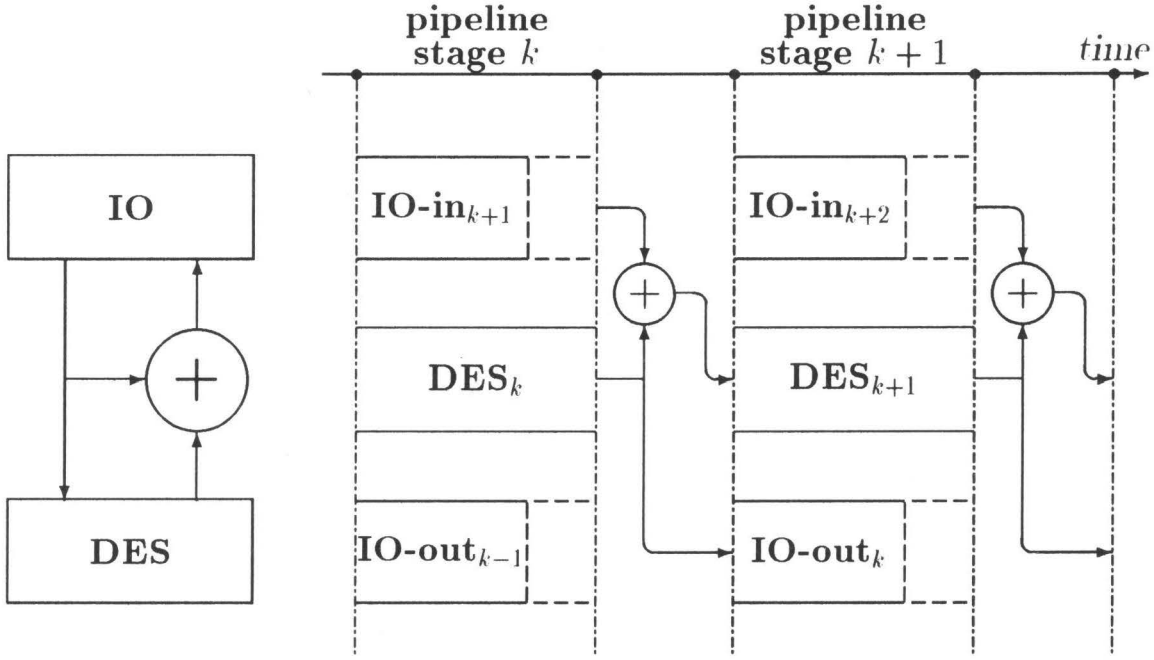


fig.2 Hardware flow of the data

Evolution in time : fast internal transport between every two consecutive pipelinestages.

4.3 Byte Oriented

The communication off chip is byte per byte in order to reduce the number of pins.

While going on_off chip *the permutations IP and IP-1* are realised. The elimination of these permutations is based on a technique of shiftregisters and an 8 bit permutation, as explained in [4]. Due to an optimal placement on the floorplan, even this 8 bit routing is avoided.

The same shiftregister technique is applied to realise key permutation PC1. The irregularity in PC1 is solved by mirroring the bitnumbers 29 through 56. And this mirroring is included in PC2, while this permutation has to be hardware routed anyway.

5 Testing.

Testing during development means *controllability and observability* [5]. When developping one must be able to reach every part of the chip (controllability) and must be able to watch the reaction of every part (observability). This is done with special *scan registers* which can isolate parts of the chip, activate and watch by scanning out the observed data. This is in conflict with security demands. One must be unable to scan out the intermediate ciphertext, initialisation vectors or keys. Therefore scanpads are provided only during development and are realised as independent blocks instead of combining them with the existing registers. Due to the modularity, one can simply cut out these pads for safe commercial implementations.

Validation tests check if the implemented algorithm really executes DES (or the chosen DES like algorithm) and if all modes are correctly designed. [6]

During lifetime, *maintenance tests* are necessary to check whether the chip still executes DES or not [7]. These tests are based on *signature*. DES has a random nature, so no extra hardware is necessary to let it act as a signature. The more random the DES nature is, the better the signature covers the faults.

6 Chip photograph

The design (fig. 3) is implemented in a 3 μ n-well CMOS process, contains 12K transistors and has an area of 25 mm^2. The main purpose was to test the functional working of the different parts, mainly the DES and the Modes part. The functionality of the chip is tested. One expects a 14 Mbit/sec datarate for a 10 Mhz clockfrequency on an 8 bytes mode.

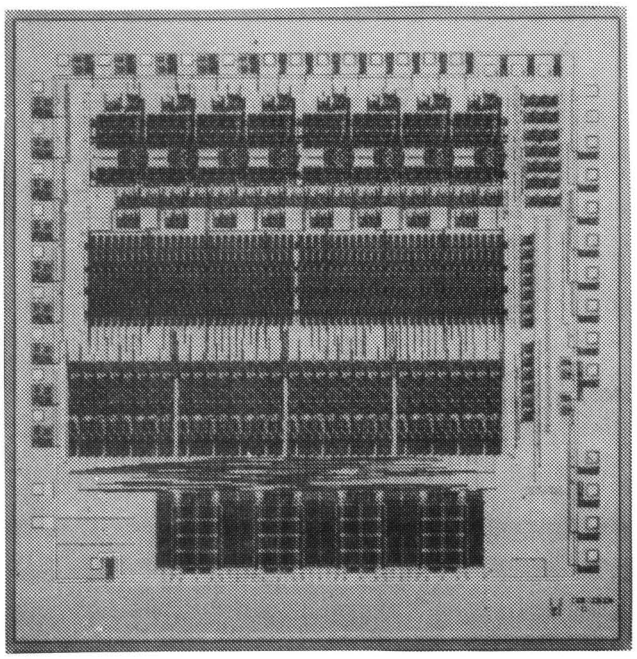


fig.3 : Chip photograph (mirrorred compared with fig.1)

7 Conclusions

A single chip has been designed and implemented which executes DES with a number of cryptographical advantages : all modes, safety, key management, speed, triple encryption, etc. Due to the modularity the main architecture and many building blocks can be reused in a flexible way for DES like algorithms. It is a compact design which can be used as a small module in larger digital VLSI circuits, but also as a fast stand alone device. In short, it has a number of unique features not found in other devices.

Acknowledgment

The authors wish to thank Dr. Y. Desmedt for the discussions on the security considerations.

References

[1] "Data Encryption Standard," FIPS, Federal Information Processing Standard, Pub no.46, National Bureau of Standards, January 1977

[2] "DES modes of operation," FIPS, Federal Information Processing Standard, Pub no.81, National Bureau of Standards, December 1980

[3] "Financial Institution Keymanagement." Draft American National Standard, Document N216, April 1984.

[4] M. Davio, Y. Desmedt, J. Goubert, F. Hoornaert, and J.-J. Quisquater, "Efficient hardware and software implementations of the DES," Advances in Cryptology Proc. Crypto 84, August 84.

[5] T.W. Williams and K.P. Parker, "Design for Testability – A Survey," IEEE Transactions on Computers, Vol. C-31 No. 1. January 1982.

[6] J. Gait, Computer Science and Technology, " Validating the Correctness of Hardware Implementations of the NBS Data Encryption Standard," Special Publication 500-20, U.S. Department of Commerce, National Bureau of Standards

[7] J. Gait, Computer Science and Technology, "Maintenance Testing for the Data Encryption Standard," Special Publication 500-61, U.S. Department of Commerce, National Bureau of Standards

High-performance Interface Architectures for Cryptographic Hardware

David P. Anderson
P. Venkat Rangan

Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720

1. Introduction

In general, secure communication in a distributed system that spans physically insecure networks and hosts must be implemented using cryptography. Software implementations of cryptographic algorithms such as DES are much slower than typical network bandwidths. However, fast hardware implementations of these algorithms are being developed [1,6] and are projected to have encryption speeds comparable to network bandwidths (i.e., 10-100 megabits per second).

Current efforts at increasing the performance of hardware encryption are directed largely at increasing the speed of encryption within the device itself [2]. Less attention is being paid to the efficiency of the interface between the cryptographic hardware and the rest of the computer system.

While implementing a secure network communication system [3,4] using commercially available components, we found that interface to the encryption device, rather than the encryption speed of the device, imposed the major limits on performance. Specifically, CPU speed was both the bandwidth bottleneck and the major source of delay, and CPU overhead was significant.

Sponsored by MICRO, IBM, Olivetti, MICOM-Interlan, Defense Advanced Research Projects Agency (DoD) Arpa Order No. 4871 Monitored by Naval Electronic Systems Command under Contract No. N00039-84-C-0089. Venkat Rangan is also supported by an IBM Fellowship.

In this paper we address the problem of designing an interface to encryption hardware that removes many of the performance limitations we encountered. With such an interface, the performance of secure network communication is determined by memory bandwidth, encryption speed, and network performance. We feel that these interface considerations should influence future hardware implementations of cryptographic algorithms.

2. Components of Network Performance

Three components are of primary interest in evaluating the performance of network communication:

- Message latency: the interval from the time a message is generated to the time it is received by its destination process.
- Throughput: the average data transfer rate that can be sustained between processes.
- Processor overhead: the fraction of processor time spent in network communication.

File server access (virtual memory paging and access to user files) is the dominant component of network communication in current distributed systems. Low latency is critical to the performance of network file access [5], and to applications involving real-time control and user interfaces. High bandwidth is required for many applications, such as those involving graphic and audio user interfaces. Processor overhead can have a significant effect on local processing speed, and also affects the latency and throughput components.

3. A Current Interface and Its Limitations

Our secure communication system was developed using widely available hardware: Sun-3 workstations with a built-in interface for the Zilog 8068 DES encryption processor. The operation of this hardware is as follows: the CPU first loads the DES chip with a key. Starting from the beginning of the data, the CPU loads an 8-byte block of data into the processor, waits for the encryption operation to finish, then removes the encrypted data from

the DES chip. The CPU repeats this process until there is no more data left in the input. Once the encryption is completed, the message can be transmitted over the network.

In the above sequence of operations, data transfers in and out of the DES chip are done one byte at a time. During the encryption of a long message, the CPU is devoted entirely to operating the cryptographic hardware, either copying data or polling the status of the cryptographic hardware for completion of an operation.

Our measurements show that during the encryption of long messages, 90% of the time is spent copying data to and from the chip, and the remaining time is spent polling the chip for completion. Maximum throughput of the encryption operation alone is 2.88 megabits per second, and maximum network throughput of secure messages (including encryption, transmission, and decryption) is 2.80 megabits per second. Maximum network throughput of unencrypted messages is 7.60 megabits per second.

The way in which the DES chip is interfaced have the following implications for the various performance components:

Message Throughput:

Because of software copying of data, the maximum encryption throughput of the system is limited by processor speed. Even if encryption time were zero, the encryption throughput would be only 3.18 megabits per second.

Message Latency:

Latency of encrypted packets is the sum of latency without encryption, the time for encryption, and the time for software data copying. In our system, the total latency for an 1024 byte message is 7.60 milliseconds, of which 5.60 milliseconds is due to encryption. Of this, 5.04 milliseconds is spent in software data copying and would be present even if encryption time were zero.

Processor Overhead:

Under a communication workload taken from traces of a real system (a heavily-used file server), the CPU overhead due to communication is 47.0% with encryption and 20.6% without. Of this, 42.0% is due to software copying and would be present even if encryption time were zero.

Therefore, the limitations on the performance of secure network communication in this system are imposed by the way in which the encryption hardware is interfaced, rather than by the speed of the encryption hardware.

4. Proposed Features of Encryption Hardware Interfaces

We propose the following hardware architectural features for the interface between any cryptographic hardware and rest of a computer system.

4.1. Interface to Main Memory

Software data copying should be avoided. This can be achieved either by integrating the cryptographic hardware with the network interface (discussed in the next section) or by providing the cryptographic hardware with *direct memory access* (DMA) capability. If the network interface hardware is fixed, only the latter alternative is possible. Whether or not DMA is used, the width of the data interface should be a word (32 or 64 bits) instead of a byte.

A DMA interface would work as follows: the CPU loads the interface with the start and end addresses of a data area in memory, and instructs it to begin an encryption, decryption, or cryptographic checksumming operation. During the operation, the interface fetches data during memory cycles "stolen" from the CPU. The CPU is free to do other work during encryption. The interface interrupts the CPU after completing the operation.

If the memory bandwidth is high enough to support the demands of both the CPU and the cryptographic hardware, the CPU and cryptographic hardware can operate at full speed in parallel. In this case the CPU overhead is essentially eliminated, and throughput is lim-

ited by memory bandwidth, encryption speed, and network bandwidth, rather than by CPU speed. If memory bandwidth is not this high, CPU operation is slowed by encryption DMA, but there will still be improvements in throughput, latency, and CPU overhead relative to software copying. The DMA technique is inherently limited by memory speed; if encryption speed is significantly greater, other approaches are required.

4.2. Pipelined Operation

The operation of the cryptographic hardware should be *pipelined* with that of the network interface, so that the encryption of a long message is overlapped with its transmission. In a non-pipelined system, the latencies for encryption and decryption are added directly to the total latency. In a pipelined system, the latency due to the combination of encryption and transmission is the maximum of the two latencies, rather than their sum. The same applies for reception and decryption.

This pipelining can be achieved in several ways. First, if the network interface and the cryptographic hardware are independent DMA devices, their operations in sending a particular packet can potentially be done in parallel. This solution is effective if the memory bandwidth is greater than that required by either operation alone, and is maximally effective when the memory bandwidth is at least the sum.

In this *DMA pipelining* technique, the devices must be synchronized so that (a) at the transmitting host the network interface does not transmit data yet to be encrypted, and (b) at the receiving host the cryptographic hardware decrypts data only after it has been received by the network interface. This synchronization is automatic if the appropriate device (the sender's encryption device, and the receiver's network interface) is faster and has higher DMA priority. If the second device in the pipeline is slower than the first, synchronization can be ensured by giving it a sufficient head start. A third alternative is to use a special-purpose *synchronizing DMA controller* that can perform multiple operations simultaneously, and in addition can delay the first operation in the pipeline to prevent it

from advancing through the data faster than the second.

The second pipelining approach is to combine cryptographic and network functions in a single hardware device, within which the two operations are pipelined and synchronized. The unit would require a complex control interface, since different regions within a network packet may need to be encrypted with different keys or not encrypted at all. This approach has the significant advantage that no extra memory bandwidth is used for encryption.

Both of the above designs can be extended to include other I/O devices. As was mentioned previously, the latency of disk I/O is significant in network file access. Ideally, this latency could be overlapped with, rather than added to, that of network transmission and encryption. This could be done by either 1) having a single interface unit control all three devices, or 2) interfacing the disk via a common DMA controller capable of synchronizing 3 independent operations (disk access, encryption, and network transmission). In the latter case, memory bandwidth is again a limiting factor on the effectiveness of the technique.

4.3. Cryptographic Checksumming

In situations where authentication rather than secrecy is needed, cryptographic checksumming (using chained encryption and retaining only the final encrypted block) may be used rather than complete encryption. This reduces memory traffic by a factor of two, since data needs to be copied into, but not out of, the encryption hardware. This reduction yields an improvement in throughput, latency and CPU overhead, particularly in the cases mentioned above in which memory bandwidth is a limiting factor.

To exploit this efficiency, the encryption hardware and its interface must support the checksumming operation. This is not the case with the Zilog DES chip, which requires that all encrypted data be read from the chip.

4.4. Large Key Bank

The cryptographic hardware should have a large number of write-only registers for key storage. Keys can be loaded by software as secure communication channels are established. Encryption operations identify their key by an index into the register bank. The bank should have as many entries as the largest number of secure channels commonly in use (perhaps 256 or so).

This scheme has the following advantages: 1) it saves time since there is no need to load a key before each cryptographic operation; 2) the write-only property and the fact that keys are not kept in main memory ensure that keys are not compromised if an intruder gains control of the kernel on the host computer.

5. References

[1] M. Davio et al.
Efficient Hardware and Software Implementations for the DES.
Proceedings of the CRYPTO 84, pages 144-147, 1984.

[2] F. Hoornaert et al.
Efficient Hardware Implementation of the DES.
Proceedings of the CRYPTO 84, pages 147-174, 1984.

[3] D.P. Anderson, D. Ferrari, P. Venkat Rangan and S. Tzou.
The DASH Project: Issues in the Design of Very Large Distributed Systems.
University of California, Berkeley/Computer Science Department Technical Report 87/338, January 1987 .

[4] D. P. Anderson and P. Venkat Rangan.
A Basis for Secure Communication in Large Distributed Systems.
IEEE Symposium on Security and Privacy, April 1987.

[5] D. R. Cheriton.
The V Kernel: A Software Base for Distributed Systems.
IEEE Software, pages 19-42, April 1984.

[6] M. Kochanski.
Developing an RSA Chip.
, Proceedings of the CRYPTO 1985.