A Hardware Design Model for Cryptographic Algorithms

Joan Daemen, René Govaerts and Joos Vandewalle

Katholieke Universiteit Leuven, Laboratorium ESAT
Kardinaal Mercierlaan 94, B-3001 Heverlee, Belgium

Abstract. A hardware implementation model is proposed that can be used in the design of stream ciphers, block ciphers and cryptographic hash functions. The cryptographic finite state machine (CFSM) model is no mathematical tool, but a set of criteria that have to be met by a real hardware finite state machine that will be used in the implementation of a cryptographic algorithm. Diffusion is studied by means of the diffusion graph and dependence matrix. For the study of confusion differential cryptanalysis is used.
In the paper the design of a high-speed cryptographic coprocessor is presented called Subterranean. This coprocessor can be used for both cryptographic pseudorandom sequence generation and cryptographic hashing. It can be implemented in a straightforward way as (part of) a chip. The small gate-delay allows high clockfrequencies, and even a moderate estimation of 20 MHz leads to a (stream-)encryption speed of 0.3 Gbit/s and hashing speed of 0.6 Gbit/sec.

Keywords: Hardware Cryptography, Stream Ciphers, Block Ciphers, Cryptographic Hash Functions, Pseudorandom Sequence Generators.

1 Introduction

In this paper the design of high speed hardware oriented cryptographic algorithms is addressed. The idea is that the algorithms make use of a finite state machine that will actually be built in hardware as a chip or part of a chip. This finite state machine must realize high diffusion and confusion. Because high speed applications are aimed at, the updating time of the finite state machine is considered the main limiting factor. Other obvious restrictions are circuit complexity, number of pins and on-chip memory.

After the finite state machine model is given, it is shown how a block cipher, a hash function and a stream cipher can be defined in terms of it. For each case the required properties of the finite automaton are given and motivated.

In Sect. 4 the *diffusion graph* and its *dependence matrix* are presented as useful tools to study the diffusion realized by the updating function. Section 5 addresses the more complicated issue of confusion. For strong confusion the diffusion must be high, but this is not enough. Absence of (residual) algebraic structure and resistance against differential cryptanalysis are presented as the two basic requirements for strong confusion.

Y. Deswarte et al. (eds.), *Computer Security - ESORICS 92*

The presented approach is conceived for the design of hardware oriented cryptographic algorithms. The tools that are applied encourage the use of symmetry and uniformity, leading to compact descriptions. We present a cryptographic coprocessor that can be used as a high-speed cryptographic pseudorandom sequence generator (CPRG) and cryptographic hash function (CHF). The approach can however also be used in the analysis of existing algorithms consisting of a number of equal rounds.

This paper gives an outline of a number of different ideas, methods and their relations. Because of the space limitations no proofs are given and definitions are stated in an informal way.

2 The Cryptographic Finite State Machine Model

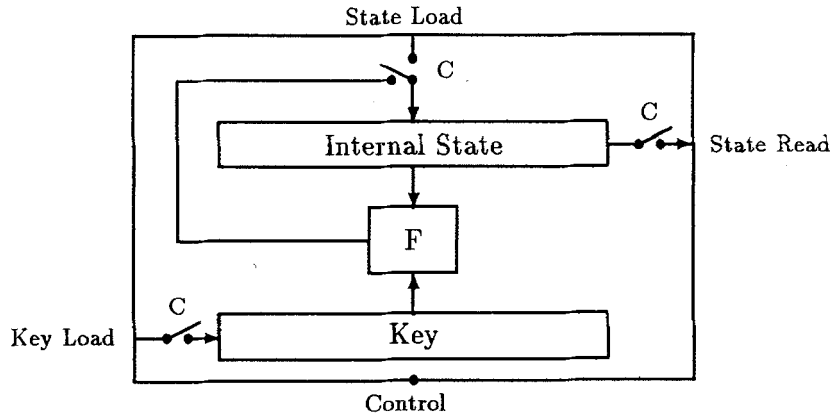

Fig. 1. scheme of the cryptographic finite state machine model

As can be seen in Fig. 1, the cryptographic finite state machine model(CFSM) consists of 4 basic components: the stateregister, the keyregister, the updating logic and the control- and load(CL) logic. The *internal state*, denoted by A, is an array of n statebits: $a_0 a_1 \ldots a_{n-1}$. The *key*, denoted by K, is an array of m keybits: $k_0 k_1 \ldots k_{m-1}$. The internal state is said to be updated, if it is assigned a new value according to the *updating function*: $A^{t+1} = \mathrm{F}(A^t, K^t)$. This is called an *updating operation, iteration* or *round* of the CFSM. A^{t+1} is the *successor state* of A^t. The function F can be seen as the juxtaposition of n *component* updating functions $a_i^{t+1} = \mathrm{f}_i(A^t, K^t)$ realized by the updating logic. In practice the output of every f_i always depends on only a subset of all bits of A and of K. These sets are called the *input set* and *key input set* of f_i. Whether an updating operation takes place or the internal registers are loaded is communicated by means of the CL logic.

The properties that are required from the CFSM depend on the type of algorithm for which it is used. In the following section it will be shown how three different cryptographic primitives can be implemented using a CFSM: stream ciphers, block ciphers and cryptographic hash functions.

3 Building Algorithms with the CFSM Model

3.1 Stream Ciphers

There is a distinction between synchronous and self-synchronizing stream ciphers. Because of the specific requirements of an SSSC, application of the proposed CFSM model would lead to a very inefficient design. The design of SSSC is treated in [7].

Synchronous stream ciphers are based on a cryptographically strong pseudorandom bitstream generator (PRG). The CFSM model can easily be used to build a PRG. This can be seen in Fig. 2. The system is initialized by loading the Initial State into the stateregister and the Key into the keyregister. Each clockcycle k bits that are at specified positions of the stateregister are presented at the output as pseudorandom bits.

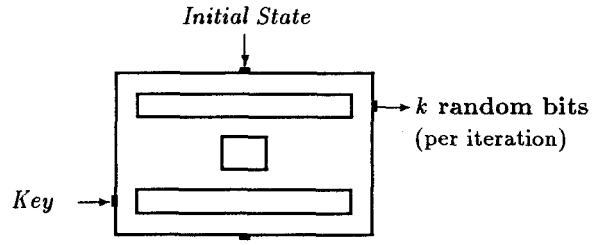

Fig. 2. CFSM based pseudorandom bitstream generator

Because this PRG is intended for cryptographic use, (even partial) recovery of the key or internal state must be computationally infeasible. More precisely, suppose a cryptanalist knows an indefinite number of output bits generated after a certain time $t = \tau$. Statebits that are unknown to the cryptanalist are referred to as private statebits. For this cryptanalist it must be computationally infeasible to perform one of the following tasks.

- Knowing some or all bits of the internal state at a certain time t_i with $t_i < \tau - \rho$, gain any information about keybits or private statebits at $t \geq \tau$. (ρ is a small integer depending on the actual PRG, typical $\rho \approx \log_2 n$)
- Knowing some or all keybits, gain any information about private statebits.

In this context computationally infeasible must be interpreted as 'expectedly slower than exhaustively trying all unknown key-statebit combinations'.

¿From these two properties it follows that the security can be solely based upon the secrecy of the key, or solely upon the secrecy of the initial state. The former property allows for re-initialization in the clear between two users who share a secret key.

The way these cryptographic properties are realized differs from the traditional approach using linear feedback shift registers(LFSR). In the CFSM approach, cryptanalysis must be prevented by the diffusion and confusion caused by the updating operation itself.

Because practical PRGs always have a finite number of states, an initial state must always lead to a cycle. This cycle must have a large length to avoid repetitions in the output sequence. PRGs based on LFSR have the advantage that almost all internal states lie on one cycle, hence choosing the number of statebits large enough solves this problem. For PRGs based on the CFSM model the cycle length depends on the initial state and the key. It is hard to predict the cycle length in individual cases. However, if the mapping realized by $A' = \mathrm{F}(A, K)$ with a constant key K can be considered a random permutation, the cycle length has a flat probability distribution [2]. With this distribution, the expected cycle length is 2^{n-1} and the probability to choose an initial state with a cycle shorter than α is $\alpha/2^n$(n is the number of statebits). For non-invertible updating functions, the expected cycle length is always shorter. Hence the updating function should be constructed that it is invertible with respect to the internal state for each key. In the following this property will be called 'state-invertibility'. The high diffusion and confusion inherent to a 'good' updating function justify the randomness assumption.

Intuitively the influence of the key on the updating function should be as great as possible. This is realized if the updating of a state A with two different keys K_1 and K_2 never gives rise to the same state. More formally $\forall A$: $\mathrm{F}(A, K_1) = \mathrm{F}(A, K_2) \Rightarrow K_1 = K_2$. This is equivalent to the statement that the function $A' = \mathrm{F}(A, K)$ with K considered to be the input and A a constant is an injection. This implies $m \leq n$. In the following this property will be called 'key-injectivity'.

3.2 Block Ciphers

Figure 3 shows how an iterated block cipher can be built around a CFSM. The system is initialized by loading the Plaintext into the stateregister. Subsequently the CFSM is iterated a specified number of times s. During the iterations the keyregister is updated according to a key schedule, realized by an external (to the CFSM) module that contains the Key. It is however also conceivable to have a block cipher with no key schedule. In this case the Key is loaded directly into the keyregister. After the s iterations, the internal state is output as the Ciphertext.

This block cipher must be resistant to cryptanalysis. Suppose a cryptanalist is able to obtain from a black box the ciphertexts corresponding to an indefinite number of (adaptively) chosen plaintext blocks and vice versa. These ciphertext

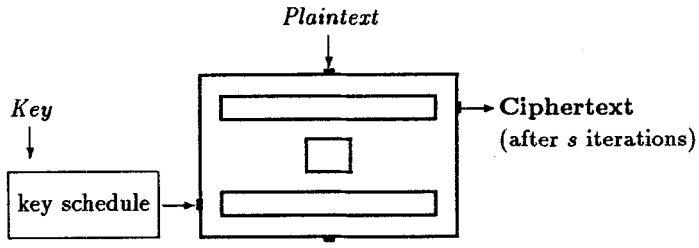

Fig. 3. CFSM based block cipher

and plaintext blocks are referred to as 'known'. We say the block cipher is resistant to cryptanalysis if it is computationally infeasible for this cryptanalist to obtain any non-trivial information about the key or plaintexts (ciphertexts) corresponding to unknown ciphertext(plaintext) blocks. By trivial information is meant exclusion of 'known' plaintext blocks as candidates for decryptions of 'unknown' ciphertext blocks and vice versa.

For decryption of the ciphertext back to the plaintext to be possible, the updating function must obviously be state-invertible. Moreover, to guarantee the diffusion and confusion of the round-keys, key-injectivity is a desirable property.

For decryption to be practical, the function $A = \mathrm{F}^{-1}(A', K)$ must be easily implementable using a CFSM with complexity comparable to that of the CFSM realizing the forward updating function. This restriction was not present with stream ciphers. Preferentially both F and F^{-1} can be realized by the same CFSM.

3.3 Hash Functions

The construction of a cryptographic hash function in terms of a CFSM is depicted in Fig. 4. The system is initialized by loading the specified Initial Value into the stateregister. Subsequently the CFSM is iterated while the keyregister is loaded with bits coming from an external module that takes care of bit selection and message padding. The number of iterations depends on the length of the message and the bit selection scheme. After the last iteration (part of) the internal state is output as the Hash Result.

If this hash function is to be used for cryptographic purposes it must be collision free. This means that finding two different messages M_1 and M_2 that have the same hash result would require a computational effort of the order of $2^{n/2}$ applications of the hash function.

The bit selection and key loading mechanism is an essential part of the hash function. Each messagebit should appear several times in the keyregister during the hashing process. In this way the content of the keyregister during a specific iteration can not be chosen without affecting it during other iterations. State-invertibility of F is a desirable property because it assures that there are no intermediate hash results that have the same successor state with the same

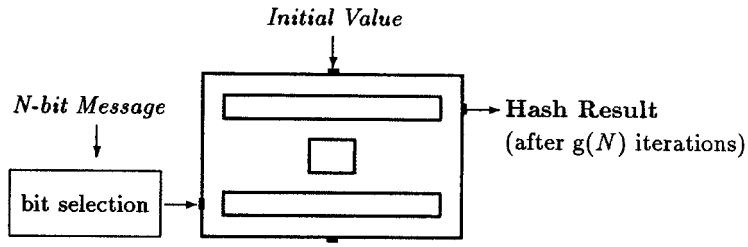

Fig. 4. CFSM based cryptographic hash function

key. Key-injectivity is desirable because it guarantees the dependence of the intermediate hash result on the selected messagebits.

3.4 The CFSM Model Revisited

Because state-invertibility and key-injectivity of the updating function are advantageous in all three cases, we incorporate these properties in our CFSM model.

In case the CFSM is used as a stream cipher or a block cipher without key schedule, the content of the keyregister is fixed during the iterations. If a block cipher with key schedule or a hash function are implemented, new bits are loaded into the keyregister in between the iterations. These will be called respectively *fixed key* and *variable key* applications.

4 Diffusion

Diffusion is the term introduced by C. Shannon [1] to denote the quantitative aspect of information propagation. In this section the diffusion in the internal state caused by CFSM iteration is addressed. A useful tool in this study is the diffusion graph and equivalently the dependence matrix.

4.1 Definitions

A diffusion graph is a weighed directed graph that is associated with a given updating function *and* key. Each statebit a_i is represented by a node (vertex) i. There is a directed edge from i to j if a_i belongs to the input set of $\mathrm{f}_j(A, K)$. The weight $0 < w_{ij} \leq 1$ of this edge is given by the probability that complementing only bit a_i before updating will cause bit a_j to complement after updating. The dependence matrix is the $n \times n$-matrix with element w_{ij} in row i and column j and 0 if there is no edge from i to j. If a state A is interpreted as an n-bit binary number and bitwise EXOR is denoted by $\oplus$ we have

$$w_{ij} = 2^{-n} \sum_{A} \mathrm{f}_j(A, K) \oplus \mathrm{f}_j(A \oplus 2^i, K) \ .$$

¿From the definition it is clear that the diffusion graph depends on the particular value of the key. This complicates the analysis of multiple round diffusion in a variable key application. Moreover, our goal is the design of an updating function that realizes very high diffusion after successive iterations for *all* possible keys. Therefore in the following only updating functions are considered where the diffusion graph is independent of the key. This can be realized by imposing certain restrictions on the key-injection.

4.2 Single Iteration Properties

The diffusion graph can be used to get an idea of local and global diffusion. The sum of the weights of the edges that start from node i can be considered the diffusion *from* bit a_i caused by updating. It is the expected Hamming distance between the two successor states of two states that differ only in a_i. The sum of the weights of the edges that arrive in a node i can be considered the diffusion *to* bit a_i caused by updating. It is a measure for the diffusion caused by the binary function $f_i(A, K)$. The sum of the weights of all edges in the graph, divided by the number of nodes, is a measure for the average diffusion per bit. This number is called the *diffusion factor* $\mathcal{D}_F$ of the updating function. We would like this factor to be maximum under the given constraints.

The number of edges arriving in a node i is equal to the cardinality of the input set of $f_i(A, K)$. Hence the diffusion is limited by the number of arguments of the f_i. Equivalently, the diffusion is limited by the number of different input sets a_i belongs to. The contribution of an edge to the total diffusion is proportional to its weight. If $w_{ij} = 1$ bit a'_j depends on a_i in a linear way, thereby realizing unconditional propagation. Given the topology of the diffusion graph, the diffusion can be maximized by choosing all weights equal to 1. However, this implies a completely linear updating function and strong confusion demands the presence of nonlinearity. Hence there must always be a substantial number of $w_{ij} < 1$. The diffusion graph with only the edges drawn with weight 1 reveals the part of the diffusion that is certified. This so called *linear subgraph* provides a skeleton for the information propagation.

Because the CFSM has to be used in high speed applications, the updating speed is important. While this speed is limited by the component updating function that has the largest gate delay, every f_i contributes to the diffusion. Hence once the updating speed has been fixed, it will be advantageous for the diffusion to design the f_i as complex as possible within the given timing and area constraints. Obviously this will cause all f_i to have comparable complexity.

4.3 Multiple Iteration Properties

In this subsection the diffusion caused by several consecutive iterations is treated. This can readily be investigated with the diffusion graph. A statebit a_j at time $t = k$ depends on statebit a_i at time $t = 0$ if there is a path of length k from node i to node j. There can be multiple paths of length k from i to j. This k-round dependence can be depicted in a k-round diffusion graph. Calculation

of the weights of the edges of this graph requires in general reconstruction of the iterated function. This is only feasible for small k and relatively simple functions. Moreover, the simultaneous occurrence of nonlinearity and multiple paths introduces key dependence into the graph.

Our goal is to maximize the k-round diffusion for $k = 2, 3, \ldots$. For $k = 2$, bit a_i at $t = 2$ written in terms of statebits at $t = 0$ gives

$$a_i'' = \mathrm{f}_i(a_j', a_k', \ldots, a_p', K') = \mathrm{f}_i(\mathrm{f}_j(A, K), \mathrm{f}_k(A, K), \ldots, \mathrm{f}_p(A, K), K') \ .$$

If the input sets of $\mathrm{f}_j, \mathrm{f}_k, \ldots, \mathrm{f}_p$ don't overlap, a_i'' depends on the maximum number of statebits possible. This is equivalent to saying that there are no multiple length-2 paths from any node to node i. If this holds for all nodes, the updating function is said to be *(2)-matched*. In this case the weights of the 2-round diffusion graph can be calculated easily by multiplication. Moreover, if the component updating functions are mutually similar, it can easily be shown that the 2-round diffusion is maximized. This can easily be extended to 3 (or more) rounds by considering paths of length 3. Consider a CFSM with mutually similar f_i and diffusion factor $\mathcal{D}_\mathrm{F} = \alpha$. If it is matched its 2-round diffusion factor will be α^2, its 3-round α^3 ...Hence for a matched updating function the k-round diffusion factor grows exponential in k for small enough k. For larger k the diffusion factor stabilizes around $n/2$.

In all algorithms, the diffusion and confusion of the keybits into the internal state is essential for the cryptographic security. After the first iteration the dependence of the internal state on the key is assured by the key-injectivity. During the following iterations the keybits are diffused over the state by the updating function. In fixed key applications the same keybits enter the updating function again every new iteration. In variable key applications some new bits are introduced in the keyregister every iteration. Using the diffusion graph, weaknesses in the key schedule and message bit selection can be found easily.

5 Confusion

Even more important than high diffusion is strong confusion. The term confusion was introduced by C. Shannon [1] to denote a qualitative aspect of information propagation. Strong confusion corresponds to involved and complicated dependencies, weak confusion to simple dependencies. The cryptographic strength of all cryptographic algorithms is ultimately based on the presence of strong confusion.

The problem with studying confusion is that the apparent complexity of a function depends on the point of view. For instance, linear modular arithmetic looks very complicated if studied at the bit level. Hence a function that apparently realizes complex dependencies may have a simple form if looked at from another angle. Every discrete function can be expressed at the bit level. Higher levels of description usually require certain algebraic properties. We believe the probability that there are 'simpler' ways of describing an iterated function can be minimized by designing the function at the lowest level possible: the bit-level.

This means that in practice the building blocks of a CFSM updating function can be limited to logical gates, *small* look-up tables and bit-permutations.

Two necessary properties for the realization of strong confusion are high diffusion and nonlinearity. It is not immediately clear how this nonlinearity should be realized. However, if the updating function is designed at the bit-level and has no higher algebraic structure, there exists an excellent set of tools to investigate the confusion. By this set we mean differential cryptanalysis [3]. The most important criterion for good diffusion is the absence of high probability multiple (updating function) round characteristics.

If the CFSM is used as a PRG, a number of statebits are presented at the output after every iteration. The positions of these statebits should be carefully chosen to minimize the correlation between the output bits. If there are no high probability multiple round characteristics, partial knowledge of the state at a certain time is of no use after some iterations.

In all three CFSM applications, the confusion of the keybits is essential. To prevent recovery or manipulation of part of the key in variable key applications, every keybit (or messagebit) must reside in the keyregister for several iterations. The confusion of the keybits is then guaranteed by the property of key-injectivity and the strong confusion caused by the updating function.

6 Subterranean: A High Speed Cryptographic Coprocessor

A cryptographic coprocessor design is presented that can be used as a pseudo-random bit generator *and* a hash function, respectively called Substream and Subhash. Subhash is related to the function Cellhash that was presented at Asiacrypt '91 [6]. With respect to Cellhash the updating function is modified to optimize the resistance against differential cryptanalysis without augmenting the circuit complexity. In Cellhash the hash result was obtained by reading the internal state after a number of iterations. Because the possibility of reading the internal state would compromise the security of Substream, this has been avoided in Subhash.

Substream and Subhash are powerful primitives in the realization of computer security. A CPRG can be used for confidentiality of stored or transmitted data by stream encryption [4]. A CHF is an indispensable component of practical data integrity, authentication and digital signature schemes [4, 5]. Moreover, the security of many cryptographic protocols depends on a CHF and unpredictable random bits that can be produced by a CPRG [4]. In the providing of security services, all bulk operations on large variable-length files, namely encryption and hashing, can be performed by the proposed coprocessor.

6.1 The Design Approach

In this section we give the design approach taken to realize the proposed CFSM. The basic idea behind it is *simplicity*. By introducing uniformity and symme-

try in the updating function, the analysis of diffusion and confusion is greatly simplified. Moreover, the resulting hardware description is short and elegant.

The updating function is defined as the composition of a number of subsequent transformations that treat the statebits in the most uniform and symmetric way possible. Each transformation takes care of a required property of the updating function. There are essentially four different transformations:

- **Nonlinearity:** a 'local' nonlinear operation. The uniformity demands that every statebit enters at least one nonlinear function.
- **Diffusion:** a 'local' linear operation. The linearity of this transformation gives a certified diffusion.
- **Dispersion:** a bit permutation that moves statebits that depend on overlapping sets of input bits away from each other to accomplish matching.
- **Key Injection:** m statebits are each EXORed with a different keybit. This guarantees key-injectivity.

The first three transformations have to be invertible to guarantee state-invertibility. In variable key applications the key load mechanism is an important part of the algorithm and has to be properly defined.

6.2 Specification of the Internal Functions

The operation of the Coprocessor is given by the calculation of *next states* A^{t+1} and K^{t+1} and output Z^t from A^t, K^t and the input B^t. For both registers there are options, indicated by means of the control logic. For the internal state there are 3 options : *reset* (to the all-0 state), *hold* and *update*. For the key there are 2 options : *hold* and *load*. Every iteration a 16-bit value Z is presented at the output. We will now specify the updating, loading and output functions in detail.

The updating function $A^{t+1} = \mathrm{F_s}(A^t, K^t)$ can be considered as a 5-step transformation of the internal state A. In the following, all indices should be taken modulo 257, $\vee$ means OR and $\oplus$ means EXOR.

$$
\begin{array}{lll}
\text{Step 1 :} & a_i = a_i \oplus (a_{i+1} \vee \bar{a}_{i+2}), & 0 \le i < 257 \\
\text{Step 2 :} & a_0 = \bar{a}_0 & \\
\text{Step 3 :} & a_i = a_i \oplus a_{i+3} \oplus a_{i+8}, & 0 \le i < 257 \\
\text{Step 4 :} & a_i = a_i \oplus k_{i-1}, & 1 \le i < 257 \\
\text{Step 5 :} & a_i = a_{12*i}, & 0 \le i < 257
\end{array}
$$

Figure 5 clarifies how the five steps of $\mathrm{F_s}$ contribute to the calculation of one statebit. Step 1 is a nonlinear cellular automaton (CA) operation where each bitvalue a_i is updated according to the bitvalues in its neighborhood (in this step and step 3 periodic boundary conditions apply). This particular CA operation is invertible if the length of A is odd. Step 2 consists merely of complementing 1 bit to eliminate circular symmetry in case all statebits are 0. Step 3 is a linear CA operation. This step is invertible if the length of A is no multiple of 7 or 31. In step 4 the actual keybits are injected in A. Step 5 is a bit permutation where

bits are placed away from their previous neighbors. The length of A is 257 (a prime) to make step 1 and 3 invertible and to avoid circular symmetric patterns in A.

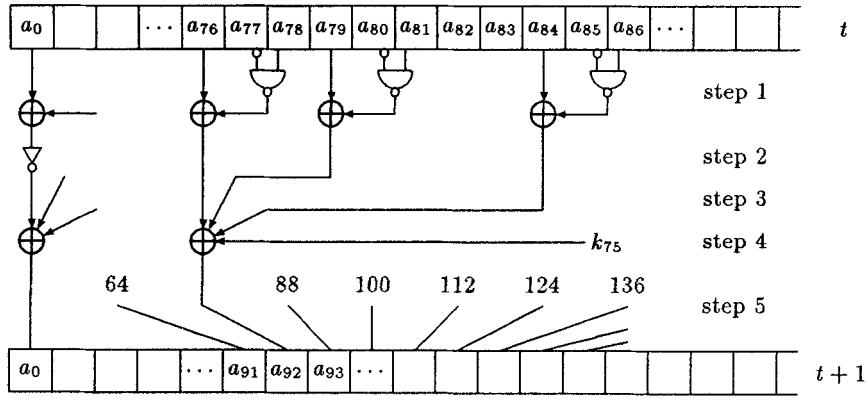

Fig. 5. schematic overview of the calculation of one output bit using the F_s function. It clearly shows that each output bit depends on 9 input bits and that nearby output bits depend on non-overlapping sets of input bits.

The updating function is invertible with respect to the state. For a fixed key, every state has exactly one predecessor. However, this inverse function is very hard to implement both in hardware and software.

In the key load option 32 bits are loaded into the keyregister in parallel. If a 32-bit word $B = b_0 b_1 \ldots b_{31}$ is loaded at time t we have

$$k_i^{t+1} = b_i \quad \text{for } 0 \leq i < 32 \qquad \text{and} \qquad k_i^{t+1} = k_{i-32}^t \quad \text{for } 32 \leq i < 256$$

The 16 output bits $z_0 z_1 \ldots z_{15}$ at time t are taken from the internal state A^t. The indices of the used statebits are given by

$$(11, 24, 37, 48, 60, 73, 84, 98, 117, 130, 143, 154, 168, 200, 235, 249)$$

These positions have been chosen such that:

- No bit of Z^t depends on bits of Z^{t-1} through F_s.
- Output bits Z^t depend on non-overlapping sets of bits of A^{t-1}.
- No bit of A^t depends on more than one bit of Z^{t-1} through F_s.

6.3 Diffusion

In this subsection the diffusion resulting from applying the updating function to the internal state is studied.

A bit of A^t depends on 9 bits of its predecessor state A^{t-1}. More precisely the bit a_i^t depends on a_{12*i+k}^{t-1} with $k \in \mathcal{R} = \{p \mid 0 \leq p < 6 \text{ or } 8 \leq p < 11\}$. This can be checked by explicitly combining all five steps of F_s and is illustrated in Fig. 5 for bit a_{92}. By recursively applying this the diffusion of multiple iterations can be studied: a_i^t depends on $a_{12*(12*i+k)+\ell}^{t-2}$ with $k, \ell \in \mathcal{R}$. Hence every bit of A^t depends on 81 bits of A^{t-2}. After three iterations the dependence is complete, i.e. a statebit at time t depends on all bits of A^{t-3}. Alternatively a bit of A^t affects 9 bits of A^{t+1}, 81 bits of A^{t+2} and all bits of A^{t+3}. The updating function has $D_F = 6$ and is 2-matched.

6.4 Confusion

In differential cryptanalysis the propagation of *differences* in the input to intermediate values is studied [3]. Suppose we have two different internal states A and A^* and their difference is defined by the bitwise EXOR : $A' = A \oplus A^*$.

An *n-round characteristic* is given by an initial EXOR A'^0, intermediate EXORs A'^i and terminal EXOR A'^n. The probability of this characteristic is the probability that two internal states A^0 and A^{*0} with $A'^0 = A^0 \oplus A^{*0}$ give rise to the specified succession of EXORs. The importance of a characteristic in differential cryptanalysis is proportional to its probability.

For the analysis we split the function F_s in the nonlinear step 1 and the linear steps 2-5. The result after performing step 1 on $A^{(*)}$ is called $B^{(*)}$. The result of performing steps 2-5 on $B^{(*)}$ is called $C^{(*)}$. It is assumed that the key is equal in the two cases. We will first study the propagation of differences during the first step.

If the local rule of step 1 is rewritten with + denoting EXOR and concatenation denoting AND, a bit of $B^{(*)}$ is obtained by $b_i^{(*)} = 1 + a_i^{(*)} + a_{i+2}^{(*)} + a_{i+1}^{(*)} a_{i+2}^{(*)}$. For the bits of B' we have

$$b'_i = a'_{i+2} a_{i+1} + a'_{i+1} a_{i+2} + (a'_i + (a'_{i+1} + 1) a'_{i+2}) \qquad (1)$$

For a given input difference the a'_i are fixed and the a_i are variables. From (1) it can be seen that b'_i depends *in a linear way* on bits of A. For a given A' the values of B' can be expressed in terms of A-bits by a vector equation $B' = MA + K$. Here M and K are respectively a matrix and a vector fixed by the value of A'. If r is the rank of this matrix (of A'), the number of possible B'-configurations is 2^r, each appearing with probability 2^{-r}. The rank M is equal to the number of linearly independent equations in bits of A. Using (1) it can easily be shown that r is equal to the number of 1-bits plus the number of 001-patterns in A'. Hence the probabilities of the output differences are only large for input differences with small Hamming weight.

The difference C' obtained after performing steps 2-5 to B and B^* can be calculated using only B'. This is due to the linearity of step 2-5. Hence for a given B' there is only one C', obviously with probability 1. The contribution of steps 2-5 to the confusion becomes clear only when characteristics over multiple iterations (rounds) are considered.

In practice, the bits of the intermediate states that are relevant for a certain characteristic can be considered as independent. In this case the probability P of an n-round characteristic is given by

$$\log_2 \mathrm{P} = - \sum_{0 \leq i < n} r_i \tag{2}$$

with r_i the rank associated with intermediate difference A'^i. Hence the most probable n-round characteristic will be the one with a minimum number of 1 and 001 patterns in the intermediate EXORs A'^0 to A'^{n-1}. Using (2) and the uniform (and circular symmetric) behaviour of the updating function with respect to EXOR propagation, upper limits for these probabilities are easily found. Moreover, for small n the range of essentially different candidate high-probability characteristics can be restricted enough to allow exhaustive search over the remaining possibilities in reasonable time.

Table 1 lists the probabilities and initial EXORs $A^{0\prime}$ of 1- to 5-round characteristics with maximum probability for the Cellhash round function and for Subterranean. We believe that the sharp decrease of the probabilities for growing n is an indication that calculations involving bits from internal states separated by several iterations become extremely complicated even if a small (say 8) number of iterations are considered. This sharp decrease is caused by the certified diffusion mainly due to (the linear) steps 3 and 5 of $\mathrm{F_s}$.

n	Cellhash	Subterranean
1	2^{-2}	2^{-2}
2	2^{-8}	2^{-8}
3	2^{-20}	2^{-25}
4	2^{-46}	2^{-68}
5	2^{-96}	2^{-154}

Table 1. the highest probabilities of characteristics for a small number of rounds

6.5 Substream

In Substream mode the Cryptographic Coprocessor is initialized by fixing the *Initial State* and the *Key*. This takes 16 input words (of 32 bits) and 16 clock-cycles. This can be expressed in a *sequence diagram*:

Clockcycle	Internal State	Key Load	Output
$t = 0$	0*(reset)*	I_t*(load)*	
$t = 1\ldots,7$	-*(hold)*	I_t	
$t = 8$	F_s*(Update)*	K_{t-8}	
$t = 9\ldots,15$	-	K_{t-8}	

After initialization 16 random bits $r_0 r_1 \ldots r_{15}$ are presented at the output per iteration and the key is not changed:

Clockcycle	Internal State	Key Load	Output
$t \geq 16$	F_s	-*(hold)*	$R^t = Z^t$

It is claimed that Substream meets the cryptographic criteria stated in Sect. 3.1 with $\rho = 8$. The expected cycle length is 2^{256} and the probability to choose an initial state with a cycle shorter than α is $\alpha/2^{257}$.

We would like to stress that no absolute proof of security can be given for any practical cryptographic algorithm. However, in the following paragraph we will give the line of reasoning that has lead to the cryptographic claims.

If the key is known, cryptanalysis boils down to total reconstruction of the internal state at a certain time (say A^{t_0}). *Partial* reconstruction of the internal state at a certain time is of little value because of the high diffusion and confusion caused by updating. In a known plaintext attack the cryptanalist is provided with 16 bits per iteration. Hence the statebits that are used in the calculation of A^{t_0} must originate from at least 17 different iterations. If both the key and the state are unknown to the cryptanalist, statebits from at least 33 different iterations are needed. The claimed cryptographic security of Substream is based on the infeasibility of calculations involving statebits separated by multiple iterations.

6.6 Subhash

The system is initialized by resetting the internal state and making sure that the keyregister contains only 0-bits. The (padded) message is loaded into the keyregister 32 bits at a time while the finite state machine is iterated. After loading all messagewords 24 more iterations are performed. During these iterations all-0 words are loaded into the keyregister. The Hash Result is given by the words Z output during the last 16 iterations.

Suppose we want to calculate the hash result H_S of a b-bit message using Subhash. Here b may be any integer. Before hashing, the message has to be padded so that its length is a multiple of 32.

Padding of the message

The message is extended with a number p of 0-bits so that its length in bits is a multiple of 32 and $0 \leq p < 32$. Subsequently the message is extended with a 32-bit word representing the value $2^{32} - 1 - p$, most significant bit first. The resulting message can be written as $M_0 M_1 \ldots M_{N-1}$, i.e. the concatenation of N (32-bit) words M_i.

The hashing process

Clockcycle	Internal State	Key Load	Output
t = −7..., −1	−	*(load)* 0	
t = 0	0*(reset)*	M_t	
t = 1..., N−1	F_s	M_t	
t = N..., N+7	F_s	0	
t = N+8..., N+23	F_s	0	$H_{t-(N+8)} = Z^t$

The **Hash Result** is defined by $H_0H_1 \ldots H_{15}$.

The claimed cryptographic security of Subhash is based on the fact that every messagebit is injected into the state 8 times during the hashing process, realizing very strong confusion.

6.7 Practical Considerations

The clock frequency is limited by the gate-delay of the updating function. This is equal to the sum of the gate-delays of one NAND, and three EXORs. An estimation of 20 MHz leads to a (stream-)encryption speed of 0.3 Gbit/s and hashing speed of 0.6 Gbit/sec.

The interface of the finite state machine to the outside world is formed by a 32-bit bus and the control connections. The key and internal state cannot be read from the outside.

In Substream mode, a new key is loaded only rarely. It must be possible to use the same key for a long time. For some applications an on-chip key memory (EPROM, EEPROM, ...) that can store a few (1 to 16) keys would be desirable.

At the moment we are investigating the implementation of the proposed cryptographic coprocessor in collaboration with IMEC (Interuniversitair Micro Electronica Centrum).

7 Conclusions

The design of conventional cryptographic algorithms is studied from an engineering point of view. A cryptographic finite state machine model is introduced that can be used at the core of stream ciphers, block ciphers and hash functions. The design of such a finite state machine is considered as an optimization of the updating function with respect to diffusion and confusion within certain circuit complexity and speed constraints.

A cryptographic finite state machine is presented that can be used for hashing *and* stream encryption, both in the Gbit/sec. range.

References

[1] C.E. Shannon, *Communication theory of secrecy systems,* Bell Syst. Tech. Journal, Vol. 28, pp. 656–715, 1949.

[2] B. Harris, Probability Distributions Related to Random Mappings, *Annals of Mathematical Statistics*, **31** (1959), 1045–1062.

[3] E. Biham and A. Shamir, Differential Cryptanalysis of DES-like Cryptosystems, *Journal of Cryptology* (1991) **4** : 3–72.

[4] D. E. Denning, *Cryptography and Data Security*, Addison-Wesley, Reading, MA, 1982.

[5] I. Damgård, Collision-Free Hash Functions and Public-Key Signature Schemes. In *Advances in Cryptology–Eurocrypt '87*, pp.. 203–217. Lecture notes in Computer Science, vol. 304, Springer-Verlag, Berlin 1988.

[6] J. Daemen, R. Govaerts and J. Vandewalle, A Framework for the Design of One-Way Hash Functions Including Cryptanalysis of Damgård's One-Way Function Based on a Cellular Automaton, *Abstracts Asiacrypt '91.*

[7] J. Daemen, R. Govaerts and J. Vandewalle, On the Design of Self-Synchronizing Stream Ciphers, *Proceedings ISITA '92*, Singapore, Nov. 16–20 1992.