FINITE STATE MACHINE MODELLING OF CRYPTOGRAPHIC SYSTEMS IN LOOPS

Franz Pichler
Laboratory for Systems Theory and Information Engineering
Institute of Systems Science
Johannes Kepler University Linz
A-4040 Linz, Austria

Introduction

Finite State Machines and Sequential Switching Circuits are important system types for modelling cryptographic devices. This is especially true for the case of secret key systems for which automata-theoretical models can be very easily found, either for the whole system or for certain parts of it. Typical examples are stream-cipher systems with a Pseudo-Noise Machine PNM which can be modelled by an aggregate of coupled shift-registers. However, the concept of a Finite State Machine is also very useful for block-cipher operations, such as realized by the DES-Algorithm. An example is the sequential generation of the DES substitution boxes.
This paper is based upon the assumption that Finite State Machine Theory (FSM theory) can be effectively applied to cryptologial investigations, such as the design of ciphering devices or in cryptanalysis.
However, it is well known, that many problems of FSM theory are considered computationally hard. Some have even been proven to be NP complete. Furthermore the state sets of FSM-models of cryptographic devices are often inevitably very large. Therefore we have to take into account that the application of FSM-theory to cryptology is generally very limited. This is especially true whenever we have to use the usual algorithms described in textbooks and articles on FSM-theory. As a consequence, very little attention has been paid to FSM theory in cryptology.
To ameliorate this situation we suggest that the following studies be undertaken:

- the investigation of special types of finite state machines, which have a mathematical structure rich enough to allow a "speed up" of related algorithms,
- the construction of interactive data-bases for FSM methods to support the user in explorative programming.

Our laboratory is now conducting studies in both of these fields. This paper mainly describes our efforts to establish an interactive data-base for FSM methods, which we designate as CAST.FSM (Computer Aided System Theory in the case of Finite State Machines).
CAST.FSM is implemented by the object oriented programming environment LOOPS which is based on INTERLISP-D. It runs on workstations SIEMENS 5815 (equivalent to XEROX 1108). Today, CAST.FSM is still in an early stage of development. Currently it is mainly being used for educational purposes. We hope to perfect our system so that it can be successfully applied to chip design ("Design for Testability") and to cryptography.

Finite State Machines and Cryptography

Before we describe CAST.FSM we would like to explore the general situation for the application of finite state machines in cryptology. We will limit ourselves to secret key systems. This category of cryptographic systems is still the principal one which is being used in practical applications. Secret key systems can be divided into stream cipher systems and block cipher systems.

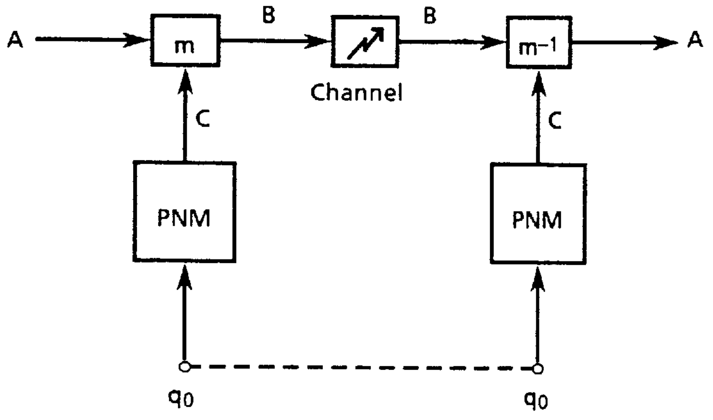


Figure 1: Stream-cipher system (E,D)

Figure 1 shows the block diagram of a stream cipher system (E,D). The encryption operation E consists of the mixing function m:AxC→B which transforms letter by letter from the plaintext into the ciphertext. This is controlled by a keying-sequence generated by the

pseudo-noise machine PNM. The decryption operation D is constructed by the inverse mixing operation m^{-1} together with a copy of PNM.
It is a well known fact that the pseudo noise machine PNM can very often be successfully modelled by FSM concepts. Therefore CAST.FSM can be applied to the design and analysis of stream-cipher systems.

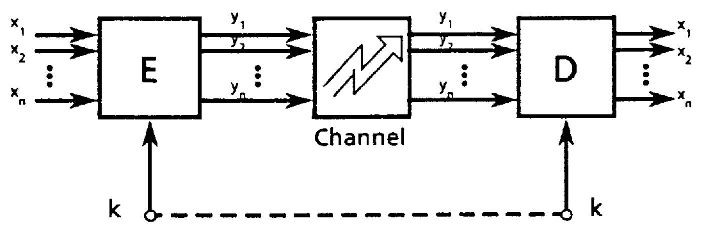


Figure 2: Block-cipher system (E,D)

In Figure 2 we show the diagram of a block-cipher system.
The encryption operation E receives the plaintext x in vector form $x=(x_1,x_2,\ldots,x_n)$ and computes under the control of the session key k the ciphertext $y=(y_1,y_2,\ldots,y_n)$.
A prominent example of a block-cipher system is the DES-algorithm in book-cipher mode.

The "inners" of the operations E and D is usually constructed by an iterative array as shown in Figure 3 (for the case of E).

Functions are the most frequently used mathematical models for E and D of a block-cipher system with a structure such as seen in Figure 3. However, the functions involved have to be, in a certain sense, very "complex" and the cryptographic design is, as a consequence, then also very difficult. The program system CAST.FSM has proven to be a useful tool for the analysis of such functions. To provide the reader with a concrete example of a block cipher operation with an internal iterative array structure we refer to the DES, in which m=16 (= number of "rounds") and the individual functions $E^{(i)}$ which are iterated have an internal structure such as shown in Figure 4.

We applied CAST.FSM successfully to the investigation of the symmetry of the S-boxes used to realize the function of Figure 4.

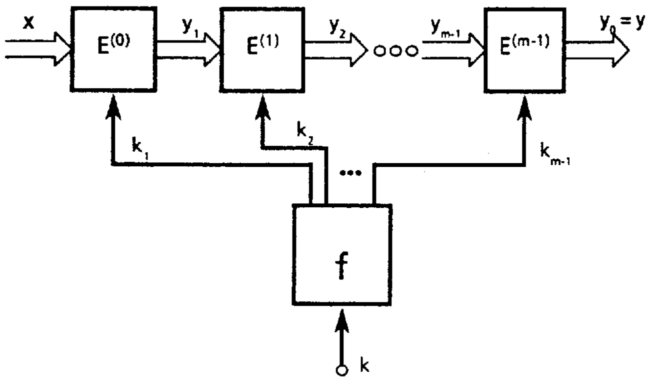


Figure 3: Iterative array structure of a block-cipher operation E

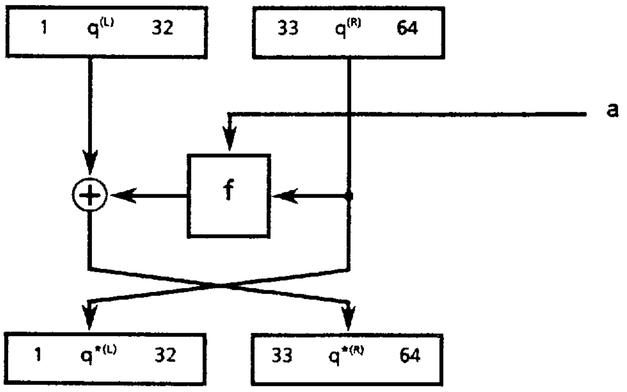


Figure 4: Internal structure of the functions $E^{(i)}$ of the DES

CAST-Computer Aided Systems Theory

Many important results of systems theory have not yet found any practical applications. One of the main reasons for this is that software for applying a systems method is often not available for the practitioner. If available, it frequently has a low degree of portability. Therefore, systems theorists have recently directed a lot of attention towards the improvement of this situation. They are striving to make it easier for the practitioner to gain access to operationally usable systems-theoretical knowledge.

One undertaking of this kind has been designated as CAST. An important difficulty which must be overcome is the establishment of a framework applicable for the definition of the shell of concrete CAST method bases.

We have developed such a framework in the form of the STIPS-machine STIPS.M (STIPS stands for Systems Theory Instrumented Problem Solving).
STIPS.M consists of a set of well-defined general systems-types, a set of systems-transformations which map systems types to each other and production rules which define the (type, transformation)→type transition. For CAST we need, in addition to the conceptional CAST shell STIPS.M, a programming environment which can be conveniently used for the development of CAST algorithms (= a well-selected sequence of systems-transformations) and for "ad hoc" programming tasks.

CAST.FSM an interactive data base in LOOPS for finite state machines

CAST.FSM is a finite state machine problem solver which is based on the CAST-shell STIPS.M. It is implemented in LOOPS/INTERLISP-D on SIEMENS 5815 workstations.
The currently available system classes can be selected from the Systems Theory Class Browser of CAST.FSM shown in Figure 5. This browser constitutes a tool on the first command level for solving FSM problems.

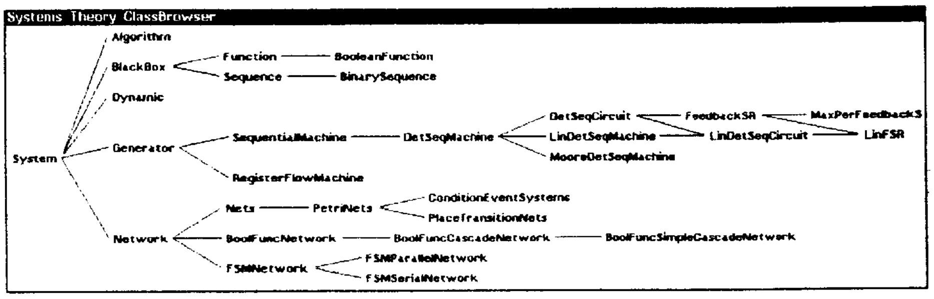


Figure 5: Systems Theory Class Browser

The development of a systems algorithms for solving FSM problems is supported by the realization tree browser. Figure 6 shows a typical example of a realization tree; it was developed during the computation of a parallel-decomposition of a finite state machine M.

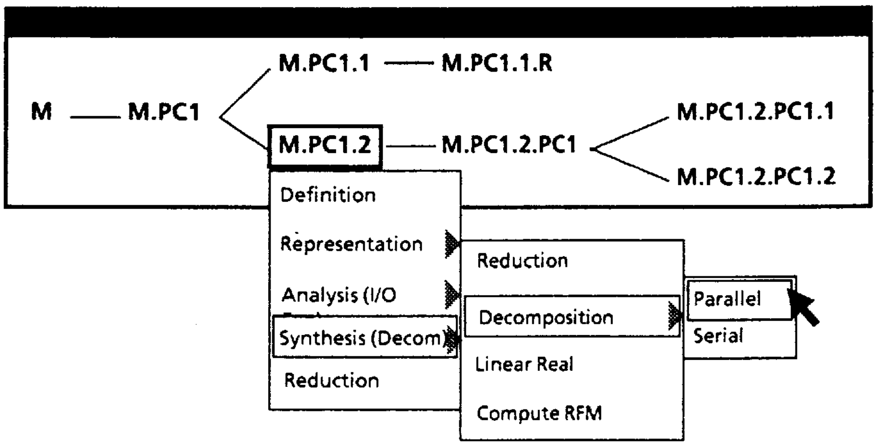


Figure 6: Realization tree

To elucidate the kinds of systems types and systems transformations which have been incorporated in CAST.FSM we provide the following examples:

systems types

- finite state machine FSM
  represented by table or state graph
- linear finite state machine LFSM
  represented by matrices (A,B,C) or by data-flow graph
- parallel network of FSM's
  represented by block diagram. The relative size of the individual machines is shown by the block-size.
- serial network of a FSM
- Petri-net
  represented by a bi-partite graph with token transitions
- register flow machine
  represented by tables or data flow graph

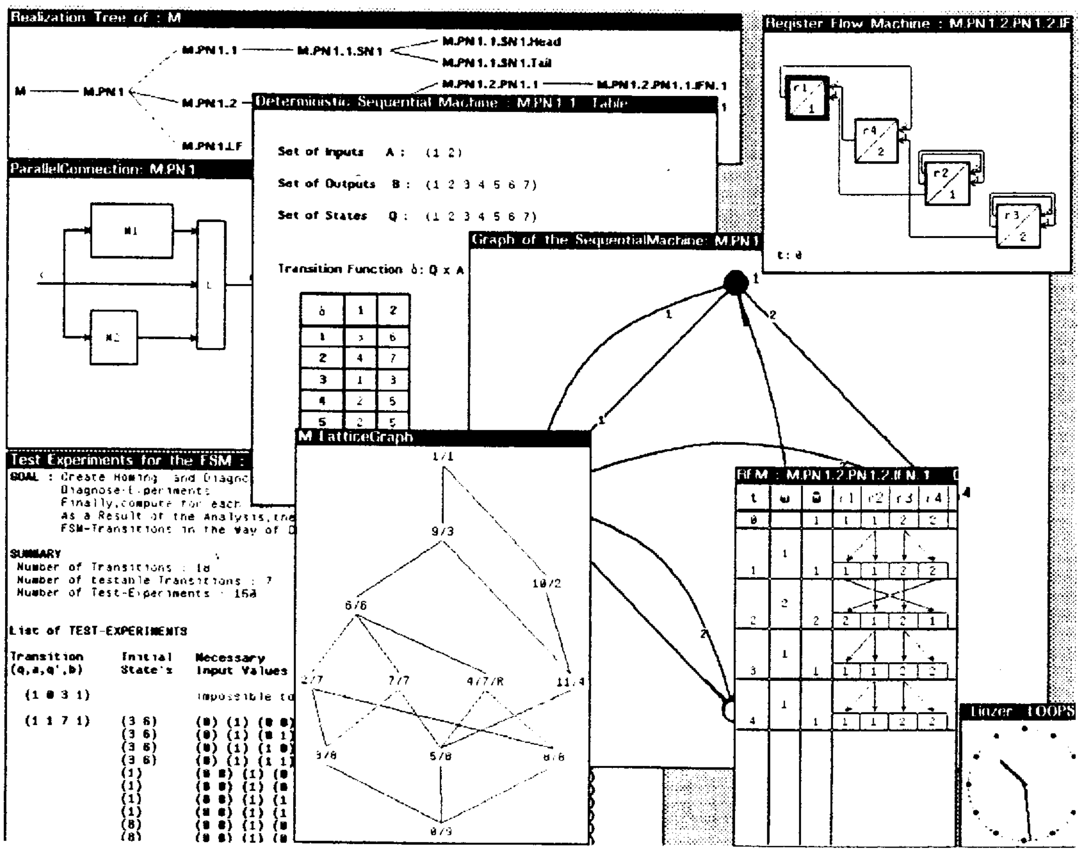
- lattice of a FSM
  represented by state-set partitions together with the related Hasse-diagram
- homing tree, diagnosing tree, synchronizing tree
  represented by tables

systems transformations

- FSM decomposition
  (method of Hartmanis-Stearns)
- linear realization of FSM's
  (method of B. Reusch)
- register flow machine realization of FSM's
  (method of Bainbridge)
- shiftregister realization of FSM's
  (method of Böhling)
- computation of the minimal length LFSR for a given binary sequence
  (method of Massey-Berlekamp)
- parallel-decomposition of LFSM's
  (by computation of the rational-canonical form of the matrix A)
- cascade-decomposition of switching functions
  (method of Ashenhurst)
- FSM state-identification
  (by homing-, diagnosis- and synchronizing experiments)
- FSM inversion

Figure 7 shows a typical display obtained when a certain finite state machine M is investigated with CAST.FSM.

CAST.FSM has already proven to be, even at its current stage of development, as a useful tool for dealing which cryptographic tasks. Besides the implementation of additional transformations we are also interested in studying the different possibilities for implementing CAST.FSM on other, more powerful workstations such as the SIEMENS WS 30 (Apollo DN 3000) enhanced by the PARWELL multiprocessor-system.


Realization Tree of : M
M
M.PN1
M.PN1.1
M.PN1.1.SN1
M.PN1.1.SN1.Head
M.PN1.1.SN1.Tail
M.PN1.2
M.PN1.2.PN1.1
M.PN1.2.PN1.1.FN.1
M.PN1.LF
ParallelConnection: M.PN1
M1
M2
Deterministic Sequential Machine : M.PN1.1 Table
Graph of the SequentialMachine: M.PN1
M LatticeGraph
Register Flow Machine : M.PN1.2.PN1.2.IF
RFM : M.PN1.2.PN1.2.IF N.1
Linzer LOOPS

```
Set of Inputs   A :  (1 2)
Set of Outputs  B :  (1 2 3 4 5 6 7)
Set of States   Q :  (1 2 3 4 5 6 7)

Transition Function δ: Q x A
```

δ	1	2
1	3	6
2	4	7
3	1	3
4	2	5
5	2	5

```
Test Experiments for the FSM :
GOAL : Create Homing  and Diagnc
       Diagnose-Experiments
       Finally,compute for each
       As a Result of the Analysis,the
       FSM-Transitions in the way of D

SUMMARY
 Number of Transitions : 18
 Number of testable Transitions : 7
 Number of Test-Experiments : 158

List of TEST-EXPERIMENTS

Transition      Initial    Necessary
(q,a,q',b)      State's    Input Values

  (1 0 3 1)                impossible to

  (1 1 7 1)     (3 6)      (0) (1) (0 0
                (3 6)      (0) (1) (0 1
                (3 6)      (0) (1) (1 0
                (3 6)      (0) (1) (1 1
                (1)        (0 0) (1) (0
                (1)        (0 0) (1) (0
                (1)        (0 0) (1) (1
                (1)        (0 0) (1) (1
                (8)        (0 0) (1) (0
                (8)        (0 0) (1) (0
```

Figure 7: Typical CAST.FSM display

Conclusions

The paper emphasizes the importance of finite state machine theory for cryptology. A fundamental requirement for the application of this theory is the availability of relevant computerized tools.
The software system CAST.FSM constitutes such a tool. The LOOPS language and the user-friendliness of the SIEMENS 5815 workstation enables one to concentrate on the problem solving task and frees him from cumbersome programming and data handling. Although most of the researchers in cryptography seem to ignore the "good old stuff" of finite state machine theory, the author believes that this theory, together with appropriate CAST implementations, could prove to be a valuable tool in crpytographical research.

References

Pichler Franz and Herbert Prähofer:
CAST.FSM
A program system to support Finite State Machine methods
Proceedings of EMCSR 88 (ed. R. Trappl)
North Holland, Amsterdam 1988 (in preparation)

Pichler Franz and Heinz Schwärtzel:
CAST: Computerunterstützte Systemtheorie
Springer-Verlag Berlin (in preparation)

Prähofer Herbert:
LOOPS-Implementierung von automatentheoretischen Methoden für "Design for Testability" Anwendungen
Master thesis, University Linz, September 1986

Pichler Franz:
Anwendung der Automatentheorie in der Kryptologie
in: E und I (Elektrotechnik und Informationstechnik),
105. Jahrgang, Heft 1, Jänner 1988, Springer-Verlag Wien
(in press)

SECTION II

HARDWARE TOPICS