CRYPTOLOGIA July 1991 Volume XV Number 3

TREE FUNCTIONS AND CIPHER SYSTEMS

Ross J. Anderson

ADDRESS: 4 Old Hall Lane, Worsley, Manchester M28 4GG, UNITED KINGDOM.

ABSTRACT: A number of encryption systems work by combining each plaintext bit with a hash function of the last n ciphertext bits. Such systems are self-synchronising in that they recover from ciphertext errors with an error extension of n.

We show firstly that if the hash function is a tree function, then the system is vulnerable to a chosen ciphertext attack and, under certain circumstances, to a chosen plaintext attack; secondly, that all hash functions are equivalent to some tree function; thirdly, that whether or not this gives a computable attack on a given algorithm depends on the connectivity of a graph associated with the hash function; and, fourthly, the implications for DES, for RSA key selection, and for algorithm design in general.

KEYWORDS: Cryptography, cryptanalysis, self-synchronising algorithms, graphs, DES, RSA

1. SELF-SYNCHRONISING CRYPTOSYSTEMS

Encryption equipment which is designed to operate at the link level often uses algorithms which recover automatically from noise or line faults with controlled error extension. Typically, a bit error in the ciphertext will result in from 48 to 256 bits of plaintext being in error with probability 0.5 for each bit.

This type of design has two advantages:

1. The equipment does not have to interrupt the session to resynchronise. This keeps the device relatively simple.

2. Data integrity on the line can be assured by the use of a standard communication protocol such as SDLC. If a bit or burst error occurs in the ciphertext, the plaintext CRC will be wrong (with probability $1-2^{-15}$) and so the frame will be rejected and re-sent. This means that no Message Authentication Codes (MACs) are required, and again keeps the design simple.

If we want to minimise the capital cost and simplify the maintenance of a secure network, a standard self-synchronising cryptobox is the obvious choice.

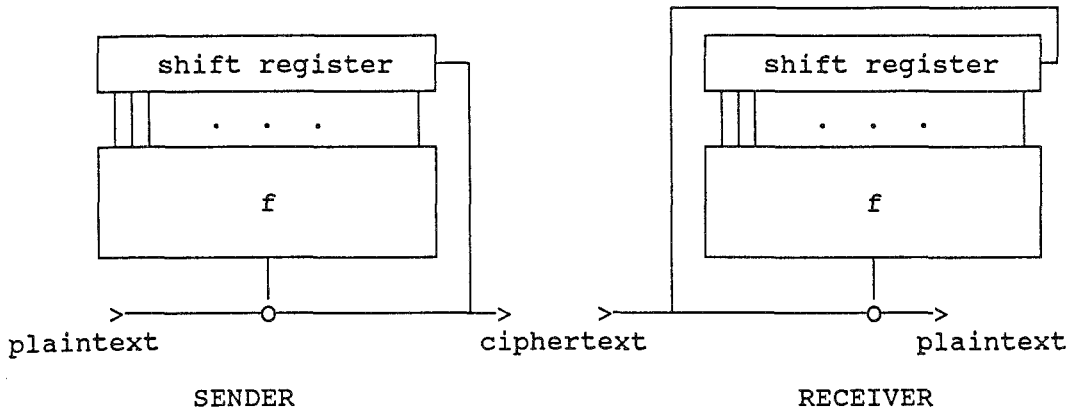

Figure 1. Typical cipher feedback system.

However, there is very little published material analysing the possible attacks on such systems.

The standard self-synchronising design is a ciphertext feedback (CFB) system. This is described in, for example, Meyer and Matyas [2]. It is shown in Figure 1.

The output from the hash function f is xor'ed with the plaintext, bit by bit. In Meyer and Matyas' case, f is the DES algorithm; its input is the last 64 bits of ciphertext, and only one output bit is used. Provided both DES chips are loaded with the same key, the system will self-synchronise with a bit error extension of 64.

2. A TYPICAL SELF-SYNCHRONISING SYSTEM

Rather than starting with DES system, we will first analyse a system proposed as a data encryption standard for use in South Africa by G. J. Kuehn of the University of Pretoria [1]. For the sake of readers who do not have access to the COMSIG proceedings, we will describe the algorithm briefly.

The system is as in Figure 1, except that it uses a proprietary hash function f, which has 125 inputs and one output. Its bit error extension is therefore 125.

The function f is built up from a basic logic cell which has five inputs and one output. The reason given for this choice is that five inputs are needed in order to get second-order correlation immunity. Using arithmetic modulo 2, i.e. where addition stands for exclusive or and multiplication for logical and, the logic cell implements:

$$f(x_1, ..., x_5) = x_1 + x_2 + (x_1 + x_3)(x_2 + x_4 + x_5) + (x_1 + x_4)(x_2 + x_3)x_5$$

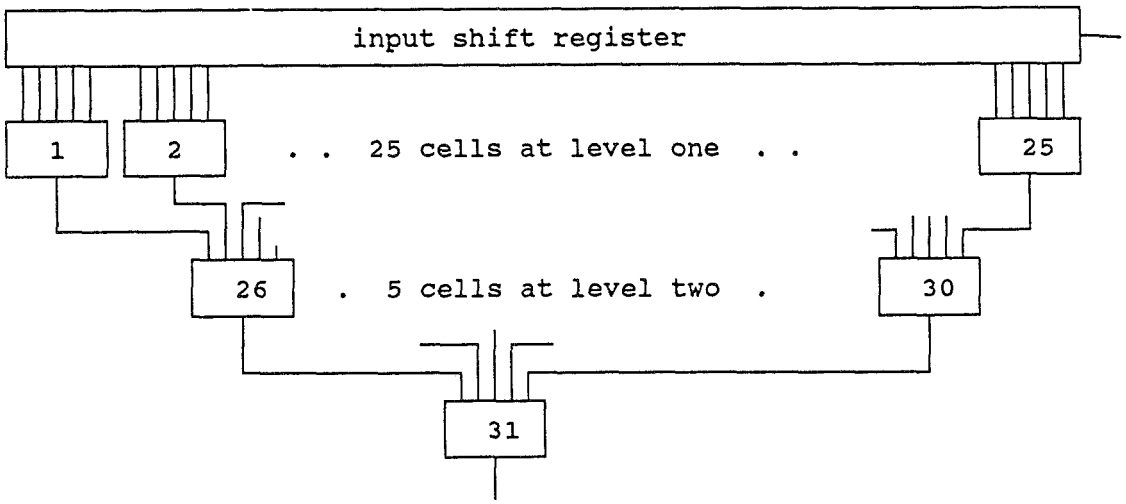

Figure 2. Kuehn's System.

The cells are combined in a three-level tree so that 125 input bits are hashed to 25 bits after one round, 5 bits after two, and one bit after three as shown in Figure 2.

The algorithm specification calls for 155 key bits. These are introduced by being exclusive-or'ed with the input bits to each of the cells.

3. AN ANALYSIS OF KUEHN'S ALGORITHM

There are circumstances in which chosen ciphertext attacks on a cryptographic system may be both feasible and undesirable. For example, if a bank is providing encryption equipment to dial-up customers, attackers will be able to subscribe too. They might be able to enter arbitrary ciphertext inputs and trap the decrypted plaintext which results. Yet this same key is being used by the bank to encrypt outgoing financial transactions, and if it can be derived from the observed decrypts, it could be used to forge a message from the bank.

We will now show a chosen ciphertext attack on Kuehn's algorithm.

Model f by replacing each cell and its associated key bits with a truth table. Filling in the truth tables will solve the algorithm, since the new function f' will give the same output as f for the same input. As each truth table will have 32 entries, f' has 992 key bits instead of 155, but has a lot of key redundancy.

An attacker who can enter ciphertext of his choice into the system can test f with arbitrary inputs. He can now build f' as follows:

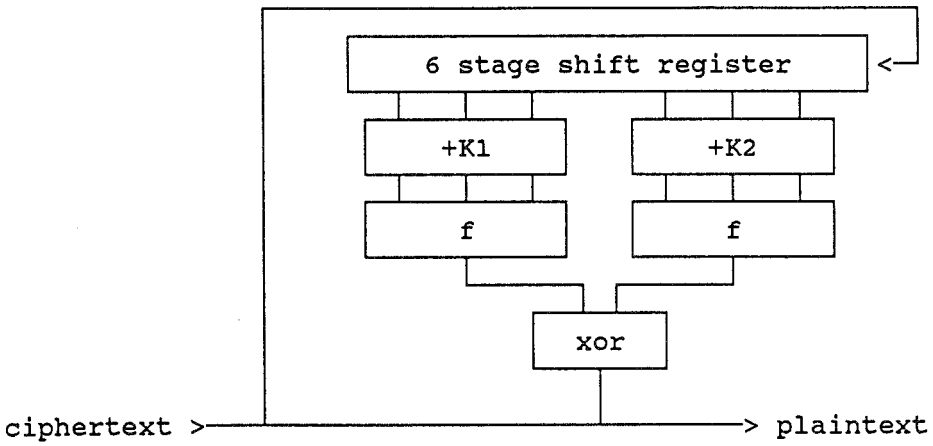

Figure 3. Simplified cipher feedback system.

1. Test two inputs which vary only in the first five bits. Does the output bit change? If so, go to 3. If not, go to 2.
2. Vary the first five bits through the other thirty combinations which have not already been tried. Now construct a truth table for cell 1 with a 1 where the output bit is 1, and a 0 where it is 0. Go to 4.
3. If the output bit doesn't depend on the output of the current cell, then randomly change one of the other bits in the shift register load, and go back to 1.
4. Repeat the process for cells 2 through 25, and finally for 26 through 31.

In order to illustrate this method of attack, we consider a simplified system with six inputs and three functional units:two three-input boxes whose outputs feed into a two-input box. The three-input boxes implement the function

$$f(x_1, x_2, x_3) = x_1 + x_2 + x_3 + x_1x_3 + x_1x_2x_3$$

while the two-input box implements exclusive or, i.e.,

$$\text{xor}(x_1, x_2) = x_1 + x_2$$

(see Figure 3.)

In Figure 3 the key bits are introduced in two blocks $K1$ and $K2$ of three bits each between the outputs from the shift register and the inputs to the two functions f. If these key bits are $K1 = (0,1,0)$ and $K2 = (1,0,1)$, then our attack will proceed as follows:

Ciphertext input	Algorithm output
000000	1
001000	0
010000	0
011000	1
100000	0
101000	1
110000	1
111000	1

We therefore replace $K1$ and f by a truth table $T1 = 10010111$; similarly $K2$ and f can be replaced by $K2 = 11101010$. The interested reader can check by hand that this gives the correct result.

Note that it may also be possible to mount a known plaintext attack on cipher feedback systems. If the unit starts up with a fixed initialisation vector, we can start it up with a number of short plaintexts which differ in their last few bits and map the top level truth tables this way.

In the special case of Kuehn's algorithm, we can guess the key bits of cell 31 (in an average of 16 tries); we can also, without loss of generality, set the key bits of cells 26 - 30 to zero (because if, for example, the first key bit in cell 27 in Figure 2 is in fact a one, this just inverts the truth table for cell 6).

It follows that a chosen plaintext attack will be easy unless the system always chooses a random initialisation vector; and even in this case, an algebraic attack may be just feasible (it would involve finding at most 800 variables given an arbitrary number of equations of degree up to nine, which can be simplified in various ways).

4. GENERALISING THE CHOSEN CIPHERTEXT ATTACK

In general, if we have any function $f : GF(2^n) \rightarrow GF(2)$ which is computable on a deterministic Turing machine, we can implement it as a string of calculations $T(i)$, $i = 1 \ldots K$, each of which is defined by a truth table, plus a list of connections $C(j)$, each of which will connect the output bit of one $T(i)$ to an input bit of another, or to the output bit of f.

This is because each step in the computation has a finite number n of input and memory bits, and can therefore be evaluated by a truth table of 2^n entries. If a calculation has more than one output bit - say m outputs - we just use m truth tables.

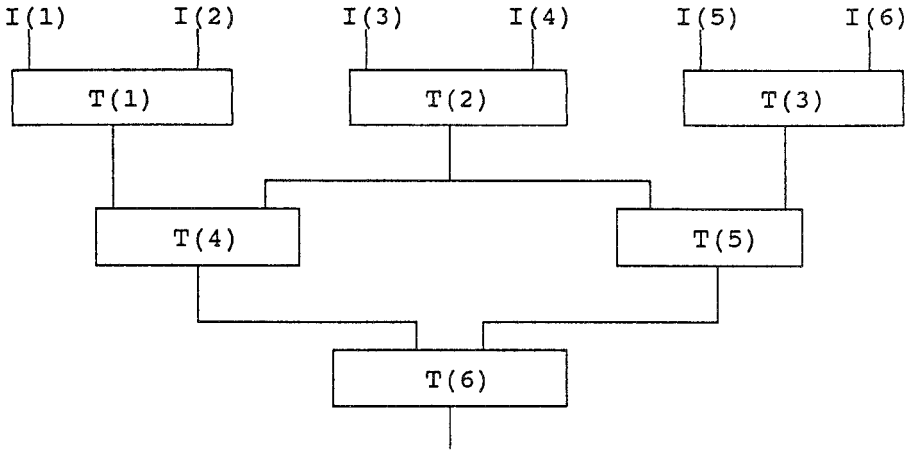

Figure 4. Function with Hamiltonian cycle.

Now draw a graph with a vertex for each input, each output and each truth table, and an edge for each connection. If this graph is a tree, we will call f a tree function. It is clear that our reconstruction attack will work against any tree function.

In order to construct a function f which is not vulnerable, there must be confusion about the values of at least two of the $T(i)$. This implies that the function's graph must contain a Hamiltonian cycle.

However, this is not as simple as it looks. Take for example the function shown in Figure 4.

Here we have a Hamiltonian cycle $T(2) - T(5) - T(6) - T(4)$. It may seem that this prevents us reconstructing the values of $T(4)$, $T(5)$, and $T(6)$. However we can eliminate the cycle by *unfolding* the graph at $T(2)$, and thus covering it with a graph which is a tree.

The graph in Figure 4 can be covered as shown in Figure 5.

Here we have duplicated $T(2)$ and its inputs; we create a table $T(7)$ equal to $T(2)$, and let $I(7) = I(3)$ and $I(8) = I(4)$.

Some effectively computable functions contain recursion of unbounded depth. All other such functions can be transformed into functions with no recursion, i.e., in which the $C(j)$ form a partial ordering, and these can be covered by a tree function, as in Figure 5 above. Each time such an unfolding is done, the number of inputs *above* it must be added to the total number of inputs.

Most of the many-to-one hash functions used in cryptography have no or finite recursion, and are thus equivalent to an expansion E followed by a tree function T, where the expansion maps each input to one or more of the tree function's

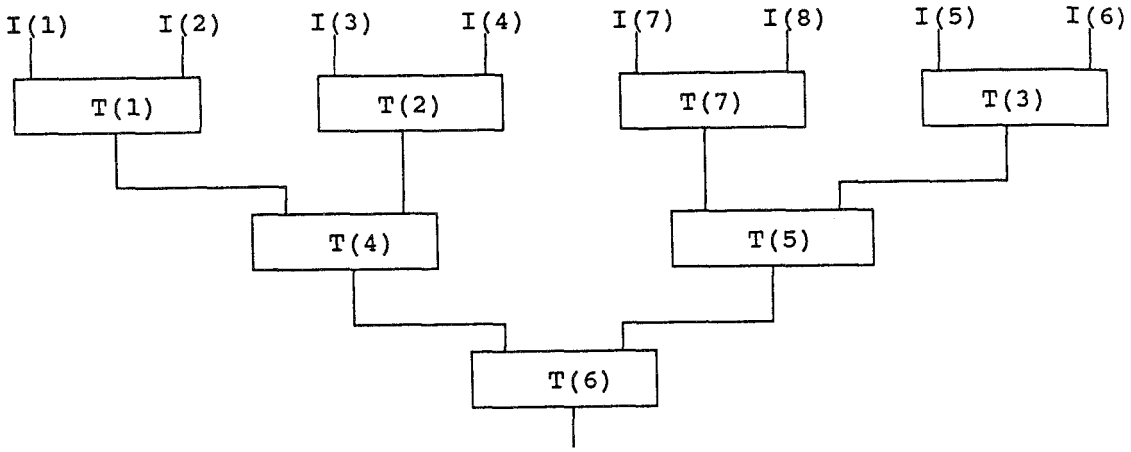

Figure 5. Covering tree of function in Figure 4.

inputs.

Given any ciphertext feedback system, we can try to attack it by constructing an equivalent function $\{E, T\}$. There will be a large number of these. In particular, each table except the lowest one can be replaced by its complement, with corresponding bits being toggled in tables lower down in the tree.

In order to solve an algorithm, it will be sufficient to map all the tables of an equivalent function by testing the inputs. This will be trivial if all the truth tables are accessible to testing.

Thus, for a reconstruction attack to be impossible, the algorithm must expand in such a way that some tables are not ever testable. In other words, for some i, any attempt to manipulate the inputs of f to try to fix elements of $T(i)$ will cause a change of state in some other $T(j)$ which will confuse the effect of $T(i)$ on the output of f.

5. IMPLICATIONS FOR ALGORITHM DESIGN

For an algorithm to be strong, one would intuitively want its covering tree to be intractably large, or a large number of the tables in any of its equivalent functions to be inaccessible, or (preferably) both.

Now consider the DES algorithm. At each stage of the autoclave, the expansion of the right half of the block from 32 to 48 bits creates 16 cycles in the graph. It follows that the number of inputs to the tree function which covers ciphertext feedback DES is at least $16^{16}64$, or 2^{70}, and a naïve $\{E, T\}$ attack on DES system would mean inverting a matrix of rank 2^{48}, which is probably not

feasible; an exhaustive keysearch attack would be cheaper.

It also seems likely that, in the case of DES, the majority of the tables in T would not be accessible. After all, we only have 64 independent inputs available for testing. Another way of looking at this is that even if we did have the computational resources to invert the matrix of rank 2^{48}, we would *expect* it to have a zero determinant. It would be nice to get a proof of this!

The RSA algorithm is also interesting. Its graph can contain even more cycles than DES, but the number of them is variable, as the algorithm generates a cycle every time we carry a bit. We should therefore be careful of instances of RSA where the computational overhead is significantly reduced for some reason or other.

Finally, although we have indicated a desirable condition for crypto algorithms, it is by no means a sufficient condition. We have not proved that either DES or RSA is strong.

6. CONCLUSIONS

When a cryptographic algorithm is expressed as a string of truth tables and connections, its graph should contain a number of cycles. Otherwise it may be decomposed and attacked table by table.

There should be enough cycles to ensure that the covering tree is very large, or that enough of its nodes are inaccessible, or both.

This requirement enables us to understand why the DES algorithm contains a 32-to-48 bit expansion in each round of the autoclave.

Finally, we are led to speculate that if x is a sparse binary number (i.e. it contains few ones and many zeros) then it may be relatively less safe as a public exponent for use with RSA.

REFERENCES

1. Kuehn, G. J. 1988. Algorithms for Self-Synchronising Ciphers. *Proceedings COMSIG 88*. IEEE TH0219-6/88. Pretoria.

2. Meyer, C. H. and S. M. Matyas. 1982. *Cryptography: A New Dimension in Computer Data Security*. New York: John Wiley & Sons.

BIOGRAPHICAL SKETCH

Ross Anderson qualified as a computer engineer through the IEE in 1976, obtained a BA from Trinity College, Cambridge, in Mathematics and Natural Sci-

ence in 1978 and an MA (also from Cambridge) in 1982. He is a Fellow of the Cambridge Philosophical Society and has worked on avionics, typographical systems, medical electronics, and communications security systems.

He currently consults to international banks on various aspects of security, including Tempest, software design, and, of course, cryptography.