Matrix GF2 Algebra
On A5/1 Explained I-bis

By Juan Chamero, juan.chamero@intaq.org, as of April 2006

LFSR: Mathematical Description

Be the primitive polynomial Pi of each register (i) given by the following expression:

Pi(z) = SUM(h)(a(i, h)*z^h, for h=0, 1, 2, ..., r(i) or under the form
Pi(z) = SUM(h)(a(i, r(i) – h)*z^h, for h=0, 1, 2, ..., r(i)

Being r(i) the size/range of register i and SUM the summa extended to its range through h. As an example we have for register R1 of A5/1 the following “row” vector A(1, .) of components a(1, h), with h from 1 to r(i):

A(1, .) = (1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

Giving place to its corresponding primitive polynomial F1(z):

F1(z) = 1 + z + z^2 + z^5 + z^19

Note: Where the z^19 coefficient (and generally the coefficient corresponding to z^ri) is supposed to “fall” immediately outside the register.

At its turn a “state” Si(t) is defined either by a vector or a series of r(i) bits changing their values along (t) which stands for “time”, mementos , time steps, iterations, of the A5/1 algorithm. Let’s see first how a Shift Register works regularly generating 2^r series for a three bits register (r = 3) that for its “primitive polynomial” [1 + x + x^3] generates from “seed” (001) the following series of states (horizontally) and output (vertically) bits:

S(0): x 001
S(1): 0 010
S(2): 0 101
S(3): 1 011
S(4): 0 111
S(5): 1 110
S(6): 1 100
S(7): 1 001
0

Where the output: **[x 0 0 1 0 1 1 1 0 ….]** if continued generates a cyclic stream of (2^3 – 1) non repeated pseudo random n-bit “states” . (x) stands for an initially undetermined bit.

We may imagine these transitions like performed by a Single State Transition Matrix A as follows:

A*S(0) = S(1)
A*S(1) = S(2)
……
A*S(2^n – 2) = S(2^n – 1)
A*S(2^n – 1) = S(0)
S(1)
S(2)
……

In general we have

A^2*S(0) = S(2)
……
A^k*S(0) = S(k)

And in general to “jump” from state j to k

$$A^{(k-j)}S(j) = S(k)$$

That defines a kind of “distance” (k – j) between two given states S(k) and S(j). For our n=3 example we have

A

0 1 0
0 0 1
1 1 0

Powers of A

A^1				A^2				A^3				A^4		
0	1	0		0	0	1		1	1	0		0	1	1
0	0	1		1	1	0		0	1	1		1	1	1
1	1	0		0	0	1		1	1	1		1	0	1
A^5				A^6				A^7				A^8		
1	1	1		1	0	1		1	0	0		0	1	0
1	0	1		1	0	0		0	1	0		0	0	1
1	0	0		0	1	0		0	0	1		1	1	0

This series has 7 = (2^3 – 1) three bits terms and its corresponding output string both cycling regularly throughout these 7 values always following the same sequence: [0010111 0010111 0010100 .....].

Note: You may appreciate that the time series bits considered in packages of three determines states accordingly. This is the milestone fact for the **“Triplets”** idea to navigate trough states at high speed. .

We have to be careful then when we talk about states that deal with bits as seen along a register for a given common time (t) and outputs bits that always refers to different times going either forwards or backward. Let’s see how this “space to time” equivalence works.

States and Output bits definition and computation

The output sequences

$$xi = \text{Series } (xi(h)) \text{ from } h=1 \text{ to infinite}$$

for each register are infinite series of output bits that have cycles of length (2^r(i) – 1).
State Si(0) at time 0 could be then defined by the first r(i) bits of the output string:

$$Si(0) = \text{Series } (xi(t)) \text{ from } t=0 \text{ to } t=r(i) - 1$$

For our example we have: S(0) = (x(0) x(1) x(2)) = (0 0 1) our seed. Let’s see how we may compute mathematically the following outputs as a linear recursive process:

x(0)	x(1)	x(2)	x(3)	x(4)	x(5)	x(6)	x(7)	x(8)	x(9)	x(10	x(11)
0	0	1	0	1	1	1	0	0	1	0	1

00101110.....

x(3) = a11*x(2) + a12*x(1) + a13*x(0),
with A being a vector defined as: (0, 1, 1)

x(3) = 0*x(2) + 1*x(1) + 1*x(0) = 0
x(4) = 0*x(3) + 1*x(2) + 1*x(1) = 1
x(5) = 0*x(4) + 1*x(3) + 1*x(2) = 1
x(6) = 0*x(5) + 1*x(4) + 1*x(3) = 1
x(7) = 0*x(6) + 1*x(5) + 1*x(4) = 0
x(8) = 0*x(7) + 1*x(6) + 1*x(5) = 0
x(9) = 0*x(8) + 1*x(7) + 1*x(6) = 1
x(10) = 0*x(9) + 1*x(8) + 1*x(7) = 0
x(11) = 0*x(10) + 1*x(9) + 1*x(8) = 1
.....

$$x(i, t) = SUM(h)(a(i, h)*x(i, t – h)), \text{ for } h= 1 \text{ to } r(i), \text{ and for } t>= r(i)$$

States at any time (t) are then defined as:

$$Si(t) = Series\ (s(i, m)(t)) \text{ from } m=1 \text{ to } m=r(i), \text{ for times } >=0$$

Clock System

The clock state could be imagined also like a machine of states. At each time step three Tap Clock bits, one for each register, located approximately in their middles (positions 8, 10, 10 for R1, R2, and R3 respectively counting rightmost bit as 0) determines the “Clock State” vector that has 8 possible states and 4 possible outcomes according to the following table:

State	Outcome
000	[1 2 3]
001	[1 2]
010	[1 3]
011	[2 3]
100	[2 3]
101	[1 3]
110	[1 2]
111	[1 2 3]

The outcome result determines which registers GO. We may easily see that “in the average” from 8 trials two times three registers GO and six times move only two. It’s a GO and STOP clock system that statistically activates each registers equally, ¾ of time steps. The function g that computes the outcomes determines for each state its “Majority bit”, the dominating bit either 0 or 1, being the value of this dominancy 2 or 3: In [000] the 0 is dominant being 3 the value of this dominance; In [101] 1 is dominant being 2 the value of this dominance. Registers with Clock Tap bits that match the dominant bit GO.

Main stages of the A5/1

Based on the didactic version of Marc Briceno, thinking about its use for experimental purposes we may distinguish 8 control points as depicted in the figure below. Important aspects to take into consideration are Memory-process tradeoff, statistic analysis, and overall efficiency of some cryptanalysis tasks related to this type of algorithm. A5/1 could be considered as a discrete “Black Box” that internally generates sequences of “states” as pseudo random numbers of 64 bits. This “states machine” generates always the same sequence like a large ring of states. The sequence is “deterministic”: once known one “initial state” this BB will always run cyclically through 2^64 – 1 different states. Let’s imagine a small decimal BB version capable to generate the following decimal sequence:

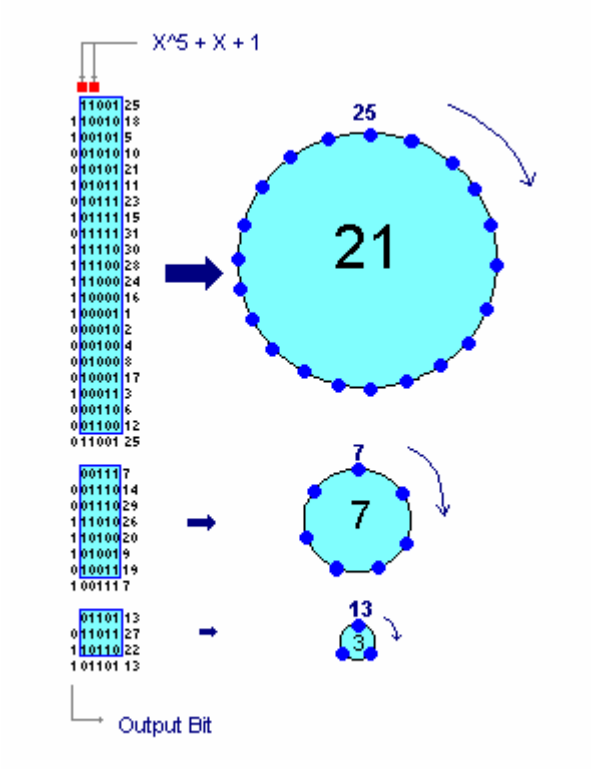
1-10-7-2-9-4-8-3-6-1-10-7-2…….

If the generation process starts at 9 instead of 1, the sequence will be 9-4-8-3-6-1-10-7-2-9-4-8-3-6…….Instead, even though deterministic. In A5/1 the sequence length is 2^64-1, and if nurtured with an initial state equal to [0000……..001] the algorithm will go through all those states “randomly”.

These type of BB's built with LFSR's may generate outcome bits streams that reproduce the whole sequence of states if considered taken in packages of n bits at a time. As in A5/1 case n=64, ideally considered as behaving "linear" (as we will see it only behaves as linear along the first 86 time steps) we may imagine a stream vector of 2^64-1 bits as the straight representation of a discrete circle of 2^64 – 1 points. At any position of this vector the n-1 positions that follow the pointed bit, reproduce the state corresponding to this position. If the position is getting to the end of this vector, the bits segment is completed by "wrapping" circularly around the end and continuing from its beginning.

Outcome bits are defined by XORing the most significant bits of each register once clocked. In fact shift and feedback bit computation for each register should be performed simultaneously. If registers are initialized to 1 we may activate all three separately either working "linear" (with the Majority Function not activated) or "non linear" with the Majority Function activated. The A5/1 algorithm will work as a pseudo random sequential machine in despite of having three LFSR's registers instead of one. Effectively, in (2^19 – 1) shifts (at any time register shifts are generally less than time steps because the activation system) Register R1 will wrap along its sequence meanwhile registers R2 is still far from its wrapping condition (2^22 - 2^19)= 3,670,016 shifts) and R3 farther (2^23 – 2^19) = 7,864,320 shifts. Ideally process will continue going through 2^64-1 = 18446744073709551615 states!.

Warning: In order to generate full sequences (2^n – 1) polynomials that rule LFSR's must be "primitive". If not sequences will be partial like in the figure below where we depict for a five bits LFSR sequence rings when polynomials are not primitive.



"Prima facie" it seems that using such a machine to only generate two 114 binary cipher masks would be like to kill mosquitoes with a machine gun. Another fact is that along the first time steps, if registers initialization is apparently so naïve, and straightforward as setting them to 1 ([00000….0001] it would take many time steps to render outcomes with a good quality of "randomness". That's true, most pseudo random generation "machines" take a time to enter into regions where 0's and 1's are evenly distributed.

In order to get a reasonable good level of randomness and at the same time to make sure that each message is properly protected and generating different states along a conversation and along sessions for different persons, A5/1 will start its 228 bits generation from very different states, ideally evenly distributed along the whole spectrum of 2^64-1 states. To accomplish this purpose the A5/1 algorithm runs along stages 0, 1, 2 and 3 as in the figure below:

E0: A Private 64 bits Key Kc is provided to be injected in parallel to registers R1, R2 and R3 previously cleared to zero. Next a 22 bits Fn Public Key, defined by the content of the counter that broadcast frame messages, pieces of 114 bits each, is injected the same way.

E1, E2: 64 Kc bits, followed by 22 Fn bits are then injected bit-a-bit via XOR synchronized with the clocking system that controls R1, R2 and R3. At the end of Kc injection it's defined the machine state S(64) and at the end of Fn injection the state S(86). At this time step the state is strictly related to a specific "Session Key" Kc, and to a particular frame of it. The A5/1 is positioned as projected in a different "start point" within the space of machine states. However a more intense random shuffling is needed.

E3: A5/1 runs now within its non linear region, with its registers activated by the Majority Function along 100 time steps. This stage presupposes a rather good level of randomness to start the outcome bits generation process. Along this stage outcome bits are ignored. Along Stages 1 and 2 we may go either forwards-backwards by matrix algebra or by using the non linear part of A5/1 algorithm. In stages 3 and 4 we may go either forwards-backwards by matrix algebra probabilistically, by "guessing" shifts performed by each register. Another way to go backwards would be by "climbing" up a tree that has as its root a state within the non linear region. From state S(186), at the end of stage 3, we may go upwards till S(86) through a tree whose root is this state. The process is of Markovian nature, critical type. It means that probability to find ancestors is 1 in the average. Effectively, the root may have from none to 4 possible ancestors because the intrinsic nature of the Majority Function with the following probabilities:

P(0) = 12/32
P(1) = 13/32
P(2) = 3/32
P(3) = 3/32
P(4) = 1/32

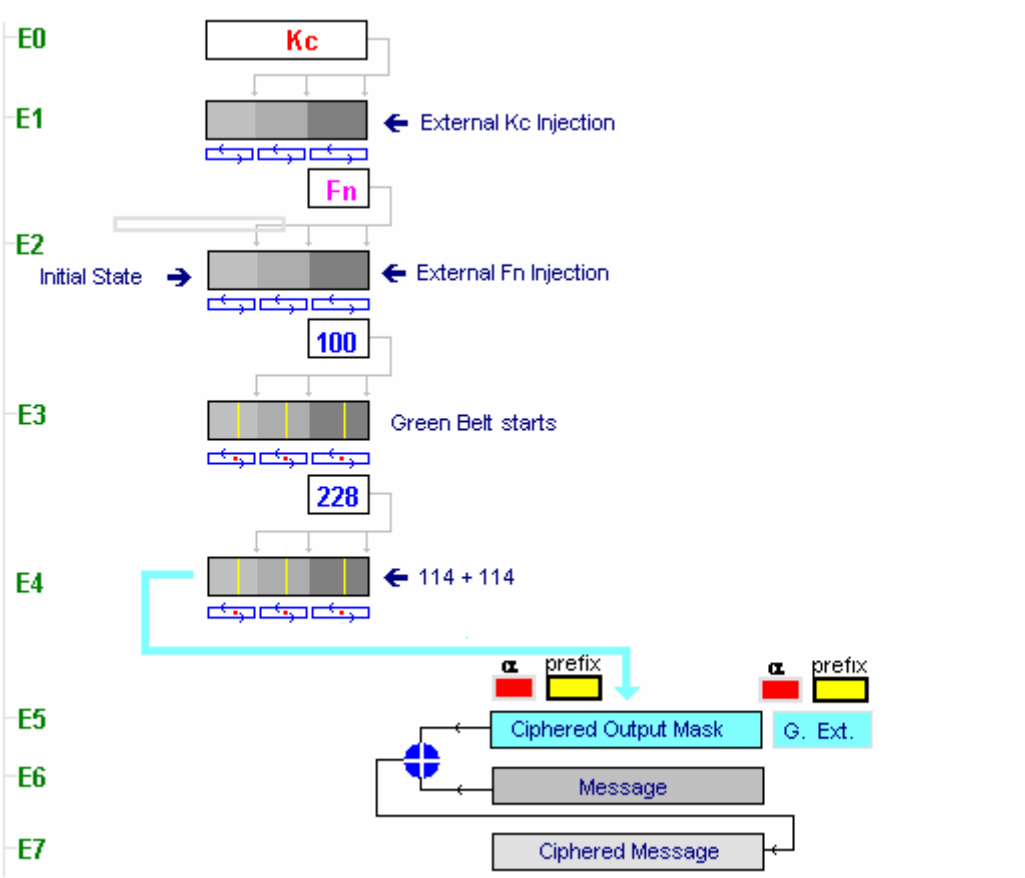
Note: See details of backwards process in our crypt_06.pdf document.

Because this characteristic not any state content (for instance enforced to be any state S(j)) along from time steps 186 to 414 will have ancestors. Another fact is that we may obtain more than one state when going upwards. For cryptanalysis work it will be necessary to go backwards from any state within the nonlinear region, namely: from 414 till 187.

E4: The A5/1 delivering 228 bits "Ciphering Masks", split in two blocks of 114 bits each. Yellow bars in stages 3 and 4 denote Tap Clock bits. These streams are determined by Kc and Fn. If A5/1 is fed by the pair [Kc Fn] the output will be always the same, a function of it. Of course we ignore Kc because Fn is associated to each frame. However if we "guess" at least one internal state, from 414 to 86, we will be able to unveil Kc. That's the challenge!.

E5, E6 and E7 briefly describes the ciphering process performed outside A5/1. If we were able to know cipher masks we may analyze their content looking for specific "strange" patterns like a given "chain" of bits for instance [1000000000000000], depicted as "alpha" in the figure below. The central idea is that this pattern is generated by some internal states S(j) that is a given 64 bits content at "distance" j measured in time steps. If we have pre-computed a set of "special" states capable to generate this "alpha prefix" at a given position within the A5/1 outcome we may infer the presence of it virtually "hidden" within its non linear region. However a problem rests: how to identify the "colliding" special state. As the outcome generation is at large deterministic, the bits that follows the "alpha prefix" along the stream are related to it (to the state), behaving like its "short name" (suffix in the figure) because the amount of bits is significantly less than 64.

If we pre compute a rather “dense” set of 2^h states with h<64, we may think short names of h bits. If the “alpha prefix” has 16 bits and h=35, each pre computed special state could be imagined like a set of 2^h pairs [Special State, Short Name], of size 64+35=99 bits (or simply 64 bits if we have a storage architecture with 2^h cubicles).



Some hints to care of

The real work of A5/1 algorithm is not publicly known yet. There are many “uncertainties”, in certain extent degrees of freedom that keep its secrecy, like for instance how the 228 bit generation proceeds (in one or two steps), how the LFSR simultaneity of clocking and feedback work, 100 steps of mixing, more or less, and many other. The first 100 output bits, $(y(t))_{t=1}^{100}$, are discarded, the following 114 bits are used as the key stream for one direction of communication in the full-duplex mode, then the next 100 bits are again discarded, and the following 114 bits are used as the key stream for the reverse direction of communication. The encrypted messages are thus very short and the resynchronization is frequent.

The A5 stream cipher is allegedly used to encrypt the links between individual cellular mobile telephone users and the base station in the GSM system. Therefore, if two users want to communicate to each other via their base station(s), the same messages get encrypted twice which makes the known plaintext cryptanalytic attack possible, provided a cooperative insider user can be established. Note also that links between base stations are not encrypted. For any user, a 64-bit secret session key is generated by another algorithm from the secret master key specific to the user and a public random 128-bit key transmitted in the clear from the base station to the user. So, a possible reconstruction of one or more session keys for a user opens a door for a cryptanalytic attack on the master key of that user.