Quantum Bio-Informatics IV
eds. L. Accardi, W. Freudenberg and M. Ohya
© 2011 World Scientific Publishing Co. (pp. 1–15)

THE QP-DYN ALGORITHMS

LUIGI ACCARDI, MASSIMO REGOLI and MASANORI OHYA*

Centro Vito Volterra, Università di Roma Tor Vergata,
**Quantum Bio-informatic Center, Tokyo Universty of Science*

1. QP–DYN algorithms: general scheme

The common denomination QP–DYN refers to a family of symmetric cryptographic algorithms based on a single mathematical structure, but differing in a potentially infinite class of specific realizations. We will see in the following how this flexibility in the choice of the realization can be used to arbitrarily increase security.

The construction of these algorithms is inspired to a special class of deterministic dynamical systems (Anosov systems) whose chaotic properties are well known in the mathematical literature (see [5]).

The theoretical bases of QPK–DYN algorithms are in the theory of chaotic dynamical systems and are described in the paper [2] which develops the previous [1] in which the main idea of the method was applied to the construction of sequences of pseudo–random vectors..
This paper also explains why the transition from the pseudo–random generation method to the cryptographic algorithm is non trivial: the point is that almost all results on this class of dynamical systems (in particular the proof of the of the caoticity proprerties) are based on techniques of measure theory and therefore are valid up to sets of zero measure. On the other hand it can be proved that the set of rational numbers, which are the only ones computers can deal with, is in the excluded zero measure set.

Therefore the standard mathematical theory cannot guarantee the required chaotic properties. More refined mathematical arguments (dealing with the *ergodic properties of periodic orbits*, see discussion in [2]) can be of help, but this direction of the theory is much less developed than the standard one. For this reason the mathematical theory, even if providing a useful intuition of the direction to pursue, cannot guarantee a priori satisfactory statistical properties of the generated sequences and these have to be verified a posteriori using the standard batteries of statistical tests.

Moreover a straightforward transposition of the pseudo–random generation program produces an easily breakable cryptographic protocol (for more details see section (8)).
Therefore the simulation of the chaotic dynamic system must be integrated with tricks of specifically cryptographic nature.

After the above mentioned proposal in [1], other authors have developed the idea to use the hyperbolic automorphism of tori (Anosov systems) for the generation of pseudo–random sequences (e.g. [12], [9], [4], [13], [11], [6]), however we have not found evidence, in the literature, of cryptographic applications of such algorithms.

In the analogy with dynamical systems the secret key is the dynamical law and the potentially public part of the key (*seed*) is the initial state of the system. Given these data, the *secret shared key (SSK)* is easily constructed from the orbit of the system corresponding to the given initial state.
The general idea of the algorithm can thus be summarized as follows: A (Alice) and B (Bob) share the (secret) dynamical law of a dynamical system whose state space is public. In order to produce an SSK they publicly exchange an initial state of the system and each of them constructs the SSK by applying a pubicly known procedure to the orbit.

By increasing the complexity of the dynamical law one can increase the security beyond any limit, but also the constructive complexity of the system increases and this decreases its speed. The balance between these two competitive requirements characterizes the good cryptographic algorithms.

A posteriori one can recognize that, by appropriately choosing the state spce, any pseudo–random generator can be fitted into this scheme. Thus what makes the difference are the specific features of the state spce and of the dynamical law.

The goal of every algorithm of the QP–DYN family is the following: given in input a text T of binary length $l_T \in \mathbb{N} \cup \{\infty\}$, produce a key of length equal to l_T.
Such algorithms are used in two situations:
(i) the length of the text is known a priori: $l_T < \infty$
(ii) the length of the text is not known a priori: $l_T = \infty$
Case (i) is typical in storage applications, Case (ii) in streaming applications.

The QP–DYN protocols are separated in three independent sub–algorithms:

(I) Initialization algorithm
(II) Secret shared key (SSK) generation algorithm
(III) Codification and decodification algorithms (i.e. use of the SSK to exchange messages).

The role of the initialization algorithms is to quickly loose memory of the public initial state (*seed*). This is equivalent to introduce a modification of the dynamical law in the first steps of the algorithm and can be done in a number of ways. The codification and decodification algorithms can be chosen arbitrarily however, since in the present case the SSK has the same length as the text and has very good statistical properties, while the potentially public part of the key can be changed for every text at zero cost, as codification/decodification algorithm it is sufficient to use the XOR of the SSK with the clear text in *one time pad* modality which, because of Shannon's theorem, is the one of maximum security.

We will concentrate our attention on the core of the algorithm, i.e. the dynamical law which produces the SSK.

All the specific protocols described in the following have been realized in software programs that have been tested in a variety of applications.

The paper [8], based on part of of Markus Gabler's PhD thesis, discusses the results of a statistical analysis of the QP-DYN pseudo–random generators. This analysis has been repeated many times by several independent groups confirming the good statistical properties of these generators.

The report [10] has been realized by the research group directed by Prof. Giuseppe Italiano of the Department of Computer science, Systems and Production, of the University of Rome Tor Vergata and compares the performances of the QP-DYN algorithms on cellular telephones with several known suites of cryptographic algorithms realizing public key exchange (RSA, Diffie–Hellmann, Elliptic curves) and subsequent encoding/decoding (AES, RC5).

The result is that the QP-DYN suite produces longer SSK in shorter times.

The content of the present paper is the following:
– section (2) describes the dynamical systems underlying the QP–DYN algorithms
– section (3) introduces the notion of *key generating function* (KGF) and describes how the secret key shared (SSK) is constructed from the orbits of the dynamical systems discussed in section (2)
– section (4) explains why the dynamical systems described in section (2) are not adequate for cryptographic purposes and outlines the modifications

of the dynamical law introduced inm order to achieve this goal
– section (5.1) describes the orbit jump function
– section (6) describes the mechanism of machine truncation
– section (7) introduces the use of multiple dynamical systems and explains how the KGF is modified in this framework
– section (8) shows how the introduction of the *cut function* alone is sufficient to increase enormously the complexity of attacks even to the single matrix algorithm
– section (9) shows how the introduction of a second dynamical system changes qualitatively the situation with respect to possible attacks in the sense that the attacker now faces an indeterminate rather than a difficult problem.

Finally let us notice that the full control on the mathematical structure, in particular the heavy use of modular multiplications, has a price in terms of speed of the algorithm (about 80 machine cycles per byte): this is quite fast for most purposes, but not enough to rank the present algorithm among the fastest presently available stream ciphers.

A faster version (by a factor of about 8) of the QP-DYN algorithm (QP-DYN-S) has been implemented in software and submitted to all the tests of the evaluation program of the Lausanne SASC Conference (13-14 February 2008) (see [14]), available in the web page of the conference and consisting of 8 measures of speed and agility.

We compared the performances of QP-DYN-S with the 8 finalist algorithms in the software profile selected by the conference. The results of these tests proved that QP-DYN-S was among the most performing 4 finalist algorithms. No algorithm, among the 8 finalists (plus QP-DYN-S), turned out better than the other ones in all these 8 parameters. For example our QP-DYN-S was about twice slower than the fastest one (10,25 machine cycles per byte against 4,48) but better in agility (21,44 against 29,50) and definitively faster than some popular algorithms, such as Salsa 20.
A detailed description of the P-DYN-S algorithm will be discussed elsewhere.

2. Dynamical systems underlying the QP–DYN algorithms

The QPK–DYN cryptographic algorithms are realized using variations of the class of (discrete time) dynamical systems described in the present section.

A dynamical system of this class is determined by:
(i1) a natural integer $d \in \mathbb{N}$, called *dimension* of the algorithm
(i2) an invertible $d \times d$ matrix M with coefficients $M[i,j]$ in the natural integers (we will write simply $M \in M(d;\mathbb{N})$) representing *the law of motion* of the dynamical system
(i3) a natural integer $p \in \mathbb{N}$, called *module* (typically it is a large prime number)
Therefore a such dynamical system is determined by the triple:

$$\{d, M, p\} \tag{1}$$

As usual we often identify a natural integer with the string of 0's and 1's defined from its binary expansion. We will use integers of m bits, typically

$$m = 32$$

or m is a multiple of 32. Denoting $\mathbb{Z}_p$ the finite field with p elements, identified to the set of natural integers $\{0, 1, 2, \ldots, p-1\}$, the state space of the dynamical system is by definition the vector space $\mathbb{Z}_p^d$. The system is supposed to be *reversible*, i.e. denoting $det(M)$: the determinant of the matrix M:

$$det(M) \neq 0 \mod p$$

Definition 2.1. The *orbit* of the vector $v_0 \in \mathbb{Z}_p^d$ is by definition the set

$$\mathcal{O}(M, v_0) := \{v_0\} \cup \{\ v_i \in \mathbb{Z}_p^d \quad | \quad v_{i+1} = Mv_i \quad , \quad 1 \leqslant i \in \mathbb{N}\ \}$$

and v_0 is called the *initial vector* of the orbit.

Since $\mathbb{Z}_p^d$ contains exactly p^d different vectors any orbit $\mathcal{O}$ is a finite set and therefore for each vector v_0 there exists $T \in \mathbb{N}$ such that $v_T = v_0$.

The smallest T with this property is called the *period* of v_0.

Since the dynamical system is deterministic and reversible, any vector in $\mathcal{O}(M, v_0)$ has the same period as v_0 and an orbit can intersect itself only if it comes back to the initial vector v_0.

3. From sequences of vectors to sequences of bits

The orbits described in the previous section generate pseudo-random binary sequences of arbitrary length through the use of a sequence of *key generating functions (KGF)*

$$\kappa_n : (\mathbb{N}^d)^n \to \mathbb{N} \qquad ; \qquad n \in \mathbb{N}$$

As explained above we identify an integer with its binary expansion therefore each function κ_n can be thought to transform n d–dimensional vectors with integer components into a single binary string.

Each step of the algorithm produces a d–dimensional vector with integer components (more precisely in $\{0, 1, \ldots, p-1\}$). Each of these numbers is represented in the basis 2 with b binary digits (typically $b = 32$). Therefore every step of the algorithm produces a string of $d \cdot b$ bit. Consequently, after n steps of the algorithm a string of $n \cdot d \cdot b$ bit will be obtained.

The sequence of key generating functions (κ_n) uses these strings to produce iteratively the SSK as follows:
starting from the initial vector v_0 at step 0, after n steps the algorithm either has halted or has produced the vectors $v_0, v_1, \ldots, v_n$.

The $(n+1)$–th step is the following. The algorithm:
(i) compares $\kappa_n(v_1, \ldots, v_n)$ with l_T
(ii-a) stops if

$$\kappa_n(v_n, \ldots, v_{n-1}, \ldots, v_0) \geq l_T \tag{2}$$

(ii-b) otherwise computes the $(n+1)$–th vector

$$v_{n+1} = Mv_n \pmod{p} \tag{3}$$

(iii) computes the $(n+1)$–step key $\kappa_{n+1}(v_1, \ldots, v_{n+1})$
(iv) goes to the next step.

If the iteration is stopped at step N the pseudo–random sequence produced is $\kappa_N(v_0, \ldots, v_N)$.

3.1. *Recursive construction of the sequence* (κ_n)

A computationally efficient way to construct the sequence (κ_n) consists in computing recursively each κ_n by fixing a function:

$$\kappa : \mathbb{N}^d \times \mathbb{N} \to \mathbb{N}$$

and defining *the first step KGF* $\kappa_1 : \mathbb{N}^d \to \mathbb{N}$ by means of the prescription

$$\kappa_1 : x \in \mathbb{N}^d \to \kappa_1(x) := \kappa(x, 0) \in \mathbb{N}$$

The sequence (κ_n) is then defined inductively for $n \geq 2$ in the following way:

$$\kappa_{n+1} : (x, y) \in \mathbb{N}^d \times \mathbb{N} \to \kappa_{n+1}(x, y) := \kappa(x, \kappa_n(y)) \in \mathbb{N}$$

Definition 3.1. A function $\kappa : \mathbb{N}^d \times \mathbb{N} \to \mathbb{N}$ that satisfies the condition

$$\kappa(x, n) \geq \kappa(0, n) := n \quad ; \quad \forall x \in \mathbb{N}^d \quad \forall n \in \mathbb{N} \tag{4}$$

will be called a *binary d–dimensional* KGF.

Remark 3.1. There are many interesting classes of binary d–dimensional KGF which are computationally easy to handle.

The choice of such functions can be used:
(i) in order to create personalizations of the algorithm
(ii) in order to increase its robustness by keeping secret such choices
In the following section we will describe the choice made in the present implementation of the QP-DYN algorithm.

3.2. *KGF by left concatenation*

Definition 3.2. Given a function

$$\lambda : \mathbb{N} \to \mathbb{N} \equiv \bigcup_{m=1}^{\infty} \{0,1\}^m$$

the λ–left concatenation function

$$\kappa_\lambda : \mathbb{N}^d \times \mathbb{N} \to \mathbb{N}$$

is defined by

$$\kappa_\lambda((n_1, \ldots, n_d); n) := [\lambda(n_d), \ldots, \lambda(n_1), n] \qquad ; \quad n_1, \ldots, n_d, n \in \mathbb{N} \quad (5)$$

where the right hand side denotes the binary string obtained by left concatenation of the binary strings $\lambda(n_d), \ldots, \lambda(n_1), n$ in the given order.

Remark 3.2. The λ–left concatenation function κ_λ, defined by (5), depends on the choice of the function λ.

The two choices that we have currently implemented are:
(i) the random truncation:
$\lambda(n)$ removes all the bits of n from the left up to the first 1 included (if the first 1 is not removed, then each component of a vector produces a string of bits whose left extreme is always 1, thus decreasing the chaoticity of the procedure)
(ii) the deterministic truncation of order c:
$\lambda(n)$ removes the first τ bits of n from left, where τ is a pre–defined number (this choice is more convenient in hardware implementations).

Example 3.1. If the components of the vectors are b–bit numbers, then in case (ii) each component produces $b - \tau$ bits so that n vectors produce a string of $(b - \tau)dn$ bits.

4. Modifications of the dynamical law

The cryptographic robustness of the SSK, constructed in section (3) above, is based on the fact that the reconstruction of the dynamical law of a complex deterministic system from its orbits, is a very difficult problem even if these orbits are relatively simple.
For example the reconstruction of the gravitational law from the elliptic orbits of the planets in the solar system has requided nearly one century of hard work of the best mathematicians, physicists and astronomers.

If the reconstruction of the dynamical law of the system from its orbits is easy, then the cryptographic scheme outlined in the previous sections is weak under clear text attacks in the sense that, if an attacker E can obtain a pair of the form (*clear text* , *encrypted text*) then she can easily reconstruct the secret key.

The dynamical system described in section (2) has two main defects:
(i) it is not enough chaotic, i.e. does not pass some statistical tests
(ii) it is not enough complex, i.e. the matrix M can be easily reconstructed once one knows a segment of orbit including a number of vectors of order d (this attack is described in the first lines of section (8)).

The reason of this weakness is the linearity of the dynamical law described in section (2). One can remedy to these drawbacks by introducing additional nonlinearities which are strong enough to destroy any attempt to reconstruct, in an efficient way, the dynamical law from an arbitrary number of its orbits, but simple enough to implement, in order not to reduce the speed of the algorithm.

The additional operations, introduced to hide the initial algebraic structure of the dynamical law are the following:
(i) the cut (already described at the end of section (3.2))
(ii) the jump of orbit (see section (5))
(iii) the machine truncation (see section (6))
(iv) the XOR with another sequence produced by another dynamical system (see section (7))

Section (7.1) is dedicated to estimate how much part of the algebraic structure can be recovered after the single operation of cut and at which computational cost.
The estimate is done in the case of fixed cut. In the case of random cut, corresponding to the currently implemented software version of the algorithm the complexity grows.

5. Orbit jump function

We have already seen that, since all operations are taken modulo p, the space of the possible vectors contains a finite number of points, hence every orbit of every dynamical system is periodic and this can create problems even with simple statisical tests.
It is usually assumed that a desirable condition for good cryptographic sequences is to have good statistical properties, i.e. to be able to pass some strongly demanding batteries of statistical tests (see however the considerations in [16] which show that this dogma has to to be taken with great caution). In order to achieve such a good statistical behavior it is necessary to make so that the periods of such orbits are at least very long (which is a necessary but not sufficient condition for chaoticity).

To achieve this goal the original dynamical system is modified as follows. The algorithm confronts, at every step, the last vector produced, say v_n, with the initial vector of the orbit v_0 (since the system is reversible only v_0 has to be memorized). If the two coincide, v_n is replaced by $J(v_{n-1})$ where $J : \mathbb{N}^d \to \mathbb{N}^d$ is a function, called *jump function.*

This is equivalent to begin a new orbit from the vector $J(v_{n-1})$, that therefore becomes the new initial point. Thus the protocol described in section (3) becomes modified as follows:
(ii-c) after step (3) the algorithm verifies if

$$v_{n+1} \neq v_0 \tag{6}$$

(ii-d) if this happens, goes to step (iii)
(ii-e) if (6) does not hold, defines

$$v_{n+1} := J(v_n) \tag{7}$$

and then goes to step (iii).

It is clear from the above description that the role of the jump function J is to prevent, as long as possible, the occurrence of a periodic orbit, improving in this way the chaoticity of the generated sequences.

Clearly this empirical prescription is not sufficient to guarantee the absence of short periods, in fact it is not difficult to build examples of systems that, even in presence of a jump function, are locked for ever in two short orbits.

It is however an empirical fact that, with the introduction of a jump function, the occurrence of periodical orbits becomes very rare: in fact after several years of trials and millions of terabytes produced, such an occurrence has never shown up.

5.1. *A choice of the orbit jump function*

It is clear that the jump function is largely arbitrary and its inclusion into the secret parameters of the algorithm improves its security.

The algorithm actually implemented uses the following orbit jump function

$$J : v \in \mathbb{Z}_p^d \longrightarrow \begin{pmatrix} 2 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} v + \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{Z}_p^d \tag{8}$$

6. Machine truncation

The fact that usual machines deal with numbers with a pre–definite number of bits, say m, can be exploited to introduce an additional nonlinearity which increases the randomness of the system.

Let m be the number of binary digits (*precision*) available for the computation and let the modulus p be chosen so to satisfy

$$2^m \geq 2^p$$

Suppose that the entries of the secret key matrix M are **not** taken modulo p but are large enough (the more of them have m or $m-1$ bits the better) so that, when one constructs the orbit, the summation occurring in the matrix–vector product, may lead to vectors whose components exceed $2m$ bits. When this happens the machine truncates the result to m bits, before computing the modulo p, according to the scheme:

$$\left[\left(\sum_{k=1}^{d} M'[i,k] \cdot v[k] \right) \mod 2^{2m} \right] \mod p$$

7. Use of multiple dynamical systems

An additional nonlinearity to the dynamics described in section (2) can be introduced by replicating the original procedure.

Two possible choices to achieve this goal are:
(i) to introduce a new dynamical law M' leaving the environment, i.e. the field $\mathbb{Z}_p$, unaltered
(ii) to introduce a new environment $\mathbb{Z}_{p'}$, leaving the dynamical law M unaltered.

Both choices lead to different dynamical systems, i.e. to different orbits.

Since the choice (ii) has the computational advantage that, at each step of the iteration, one saves a matrix–vector multiplication, we have implemented this choice. In what follows we will illustrate this implementation.

7.1. *The 2 prime protocol*

Given a text T of length l_T we consider two dynamical systems

$$\{d, M, p', v_0, \kappa_n, J\} \qquad ; \qquad \{d, M, p'', v_0, \kappa_n, J\} \tag{9}$$

with:
(i1) the same dimension $d \in \mathbb{N}$
(i2) the same dynamical law $M \in M(d; \mathbb{N})$
(i3) the same initial vector $v_0 \in \mathbb{N}^d$
(i4) the same orbit jump function $J : \mathbb{N}^d \to \mathbb{N}^d$, given by (8)
(i5) the same key generating functions (KGF) $\kappa_n : \mathbb{N}^{dn} \to \mathbb{N}$ $(n \in \mathbb{N})$
(i6) different moduli $p', p'' \in \mathbb{N}$ (prime numbers).
One then executes the algorithm described in section (3) with the only difference that step (iii) (computation of the $(n+1)$–step key) is replaced by the following two steps:
((iii)-2syst-a) having produced, without any cut, the two $(n+1)$–step keys

$$\kappa_{n+1}(v_1, \ldots, v_{n+1}) \qquad ; \qquad \kappa_{n+1}(v_1', \ldots, v_{n+1}')$$

(remember that, in the present implementation, the KGF κ_n are the same) the algorithm computes

$$\kappa_{n+1}(v_1, \ldots, v_{n+1}) \oplus \kappa_{n+1}(v_1', \ldots, v_{n+1}') \tag{10}$$

where, for any two binary strings x, y, $x \oplus y$ denotes the string $x\ XOR\ y$ and if necessary the two strings are made of the same length by adding zeros on the left.
(v-2syst-b) then removes from the bits of (10) all the leading 0's and the first leading 1 (or, in the case of fixed cut, the first τ bits from left).

The result is the $(n+1)$–step key of the modified algorithm:

$$\overline{\kappa_{n+1}}(v_1, \ldots, v_{n+1}\ ;\ v_1', \ldots, v_{n+1}')$$

The stopping rule is the same as in section (3).

Remark 7.1.
It may happen that the string (10) has some leading bits equal to zero because, depending on the choice of the initial vector, the modulo operations

with p' and p'' may enter the game only after a certain number of orbit steps: in these steps the vectors produced by the two systems are identical.

Therefore an initial part of the resulting sequence should not be considered: the length of the omitted part is a parameter (which can even be public with no harm for the security of the algorithm) included in the initialization procedure.

8. Attacks to the 1–matrix algorithm

The robustness analysis that follows has been developed in the worst possible hypotheses for the defender. That is:
– it is only considered the case of a single dynamical system (thus excluding the most important security factor)
– the machine cut is excluded
– the jump function is excluded
– one supposes that the only secret key is the matrix M
while the following information are considered public:
– the prime number p (module),
– the dimension d,
– the initial data initial v_0,
– the KGF sequence (κ_n): left concatenation without permutations

Furthermore:
– the bit cut (see the end of section (3.2)) is considered fixed and public.
– the most favorable case for the attacker E is considered, i.e.: *the clear text attack*, in which the both original text T and the encrypted text are known to the attacker.

Clearly the degree of security grows if, as it is always possible, some of these informations are part of the secret key shared a priori.

The fact that, even under these extreme conditions, the breaking complexity of the algorithm can be very high helps to guess why up to now it has not been possible to find, even at theoretical level, attacks to the 2–matrix version of the algorithm.

Suppose that:
(i) E knows $d+1$ consecutive (column) vectors of the orbit starting from some $l \in \mathbb{N}$: $\{v_l, v_{l+1}, \cdots, v_{l+d}\}$
(ii) the first d among these vectors are linearly independent
and define the following (column) matrices:

$$V = (\, v_l, \cdots, v_{l+d-1}\,) \in M(d;\mathbb{N}) \qquad : \qquad V' = (\, v_{l+1}, \cdots, v_{l+d}\,) \in M(d;\mathbb{N})$$

then $MV = V'$ and this allows to obtain the secret key $M = V'^{-1}$ hence to break the algorithm.

However, since E only knows a binary string her problem is to recover from it the components of the vectors v_{l+i}. This means that E has to discover which bit is the first bit of the first component of v_l. Once she has this information, since she knows from the public structure of the algorithm that the bits are generated from the vectors by left concatenation without permutations and that the cut is constant and equal to τ, E can determine each component of each of the vectors $\{v_{l+1}, \cdots, v_{l+d}\}$ up to an ambiguity of τ bits per component.

This implies 2^τ possibilities for component and therefore $2^{d\tau}$ possibilities per vector.

Since E needs $d+1$ vectors, she has to choose among $2^{d(d+1)\tau}$ possibilities.

For example, if $d = 10$ (a dimension that an usual personal computer can manage without any difficulty), then $d(d+1) = 110$.

Supposing, in order to further facilitate E's task, that $\tau = 2$, we see that E has to choose among 2^{220} possibilities.

For each of these choices E must carry out one inversion and one multiplication of matrices of order 10 (each of these operations requires an order of 10^3 multiplications).

Finally notice that an increment of d or τ, e.g. 15 instead of 10 or 3 instead of 2, increases the construction complexity of the orbit by a factor that is at most quadratic in the increment, while the complexity of attack increases exponentially.

9. Attacks to the 2–matrix algorithm

The attacks described in section (8) cannot be applied to the 2–matrix algorithm because in this case E can only recover the sequence

$$\overline{\kappa_N}(v_1, \dots, v_N\ ;\ v'_1, \dots, v'_N) := \kappa_N(v_1, \dots, v_N) \oplus \kappa_N(v'_1, \dots, v'_N)$$

(where $\oplus$ denotes the *XOR* operation) and it is impossible to know if, in this sequence, a 1 has been obtained from the combination of a 0 in $\kappa_N(v_1, \dots, v_N)$ (SSK of the first dynamical system) and of a 1 in $\kappa_N(v'_1, \dots, v'_N)$ (SSK of the second dynamical system), or vice-versa. Similarly it is impossible to know if a 0 from two 0's or two 1's.

In other words, and this is one of the main ideas of the new algorithm,

E is not facing a *difficult problem*, but an *indeterminate one*, namely: *given a sum of two elements in a ring, reconstruct the value of the addends.*

Since, fixing arbitrarily one of the two elements, the knowledge of the sum determines the other one uniquely and since, given the information available to E, all the elements of the ring are equiprobable, it follows that the ambiguity is of the same order of the number of elements of the ring. In our case this means that, for every component of every vector, E has an ambiguity, of order p. For each vector the ambiguity will be therefore of order p^d and, for $d+1$ vectors, of order $p^{d(d+1)}$.

Finally the simultaneous use of the three different fields, i.e. $\mathbb{Z}_{p'}, \mathbb{Z}_{p''}, \mathbb{Z}_2$ (where the last one refers to the *XOR* operation), makes an algebraic attack, even at the statistical level, practically impossible.

References

1. Accardi L., F. de Tisi, A. Di Libero: Sistemi dinamici instabili e generazione di sequencei pseudo-casuali, In: Rassegna di metodi statistici e applicazioni, W. Racugno (ed.) Pitagora Editrice, Bologna (1981) 1-32
2. Abundo M., Accardi L., Auricchio A.: Hyperbolic automorphisms of tori and pseudo-random sequences, Calcolo 29 (1992) 213–240
3. Accardi L., Regoli M.: Some simple algorithms for forms generations, L. Accardi (ed.) Fractals in nature and in mathematics, Acta Encyclopaedica, Istituto dell'Enciclopedia Italiana (1993) 109–116
4. L. Afferbach and H. Grothe, J. Comput. Appl. Math. 23, 127 (1988)
5. Arnold V.I., Avez A.: Ergodic problems in classical mechanics, New York: Benjamin (1968)
6. L. Barash, L.N. Shchur, Periodic orbits of the ensemble of Sinai-Arnold cat maps and pseudorandom number generation Physical Review E 73, 036701 (2006) The American Physical Society (2006)
7. M.Cugiani: Metodi Numerico statistici (1980)
8. Markus Gäbler: Statistical Analysis of Random Number Generators October (2007); see also M. Gäbler's paper in these proceedings.
9. H. Grothe, Statistiche Hefte 28, 233 (1987)
10. Giuseppe F. Italiano, Vittorio Ottaviani ,Antonio Grillo, Alessandro Lentini: BENCHMARKING FOR THE QP CRYPTOGRAPHIC SUITE August (2009)
11. P. L'Ecuyer and P. Hellekalek, in Random and Quasi-Random Point Sets, No. 138 in Lectures Notes In Statistics Springer, New York (1998)
12. H. Niederreiter, Math. Japonica 31, 759 (1986)
13. H. Niederreiter, J. Comput. Appl. Math. 31, 139 (1990)
14. Mattew Robshaw, Olivier Billet (Eds.): New Stream Cipher Designs, The eSTREAM Finalists State-of-the-Art Survey, LNCS 4986 Springer (2008)
15. Regoli, M., pre-mRNA Introns as a Model for Cryptographic Algorithm: Theory and Experiments, proceedings: QUANTUM BIO-INFORMATICS

III From Quantum Information to Bio-Informatics Tokyo University of Science, Japan, 11-14 March 2009
16. Regoli, M., A redundant cryptographic symmetric algorithm that confounds statistical tests, Open Systems and Information Dynamics (2011) to appear