International Symposium on Nonlinear Theory and its Applications
Xi'an, PRC, October 7–11, 2002

Efficient Chaos-based Secret Key Generation Method For Secure Communications

Sergio Callegari[1], Riccardo Rovatti[1] and Gianluca Setti[2]

[(1)]DEIS, University of Bologna
Viale Risorgimento 2
40136 Bologna, Italy
Email: scallegari@deis.unibo.it, rrovatti@deis.unibo.it

[(2)]DI, University of Ferrara
Via Saragat 1
44100 Ferrara, Italy
Email: gsetti@ing.unife.it

Abstract

The choice of appropriate random sources is a critical point in the implementation of security schemes. Chaotic dynamical systems are a promising alternative to pseudo-random generators thanks to the impossibility of replicating their behaviors beyond limited amounts of time. However, for a wider acceptance, their implementations need to be further streamlined and improved. Here we propose an architecture which exploits the parallel operation of many units to relax timing issues, which avoids common robustness problems, and which alleviates the costs of analog design by re-using expertise originally accumulated in the design of A/D converters. Key design choices are commented and justified through the illustration of preliminary simulation results.

1. Introduction

Recent times have seen a growing economic/social impact associated with the secure transfer of information. Secure communications is built on cryptographic and authentication techniques whose ability to foil pattern analysis is critically dependent on secret bit-strings (keys) often produced by suitably manipulating the output of random sources. The present scenario has brought an increasing distrust of conventional digital pseudo-random generators, which have *repeatable behaviors* that a skillful attacker might exploit [1]. This background favors the spreading of chaos based random generators which are intrinsically non-repeatable [2].

However, in spite of a mature theoretical framework [3], chaos based methods are still not widely deployed and other physical phenomena are preferred for the secure generation of random data [4]. The normal reaction time to a new technology is here being amplified by practical difficulties in the implementation of chaotic sources, which require accurate *analog* hardware [5].

In the following we propose an implementation approach which collects many recent advances and proposes novel considerations in order to relieve some issues related to throughput, robustness and design costs. Key points are: *replication* to balance the accuracy-throughput tradeoff typical of analog realizations; *interleaving* to reduce the consequences of implementation errors; the adoption of an *intrinsically robust chaotic map*; and *efficient algorithms* to extract statistically independent bitstreams. Furthermore, the suggested arrangement permits to design analogue modules that closely mimic the stages of recent pipeline analog-to-digital converters (ADC) based on $1 + {}^1\!/_2$ bit cells [6]. The latter is a major advantage, as it permits a vast re-use of design competences developed in a field which has recently witnessed great advances pushed by large production numbers.

The digital processing that is necessary to achieve the statistical independence of the output bits and operates by discarding selected amounts of "data" in order to eliminate redundancy. By means of an analysis based on Markov chains [7], we illustrate that it can be carried out with a whole family of increasingly more complex techniques, the most complex of which corresponds to the data accumulation performed in $1 + {}^1\!/_2$ pipeline ADCs. Hence, also the digital part of pipeline converters can be potentially re-used. Evaluating whether this is convenient or not requires extensive simulation. Preliminary results suggest that more light-weight techniques can probably be favored.

2. Background: Markov chaotic sources

We focus on PWA Markov chaotic sources, i.e. systems based on the iteration of maps $M : [-1, 1] \to [-1, 1]$ where M is such that a partition $\{X_i\}$, $i = 0, \ldots p-1$ of $[-1, 1]$ exists, satisfying the properties that: *i)* M is affine on each X_i; *2)* either $M(X_i) \cap M(X_j) = \emptyset$ or $X_i \subseteq M(X_j)$ for any i, j. The system evolution is described as $x_{n+1} = M(x_n)$ where x_0 is the initial condition. It is known that under these assumptions the dynamics of $\{x_k\}$ embeds a Markov chain [3] and that the evolution can be studied through a *kneading matrix* defined component-wise as

$$\mathcal{K}_{ij} = \frac{\mu(X_i \cap M^{-1}(X_j))}{\mu(X_i)} \tag{1}$$

where μ is the usual Borel measure. Each partition set X_i is associated to a Markov state $\mathsf{x} = i$, so that $x_n \in X_i \Leftrightarrow \mathsf{x}_n = i$ (i.e. x is a quantization of x), and each entry $\mathcal{K}_{ij}$ is the conditioned probability $\Pr(\mathsf{x}_{n+1} = j \,|\, \mathsf{x}_n = i)$. It is well known that, for non-pathologic initial conditions, some maps result in sequences $\{\mathsf{x}_n\}$ made of independent samples. For instance, this is the case with the Bernoulli shift (BS), $x_{n+1} = 2x_n \bmod 2 - 1$, whose kneading matrix is characterized by $\mathcal{K}_{ij} = {}^1\!/_2 \ \forall i, j \in \{0, 1\}$. When the symbols $\{\mathsf{x}_n\}$ are not independent, independence can be recovered by processing them in *groups* and *discarding selected amounts of data*. Since constant data rate processing is desirable, one wants to process groups containing a fixed quantity m of symbols and to return q bits for each group. Obviously ${}^q\!/_m$ is constrained to be less than the number k of bits necessary to codify each symbol. Another constraint from chaos theory limits ${}^q\!/_m$ to be less than the *average entropy* of the chaotic map which, in turn, is bounded

All authors are also with ARCES, University of Bologna, Italy.

by the $\log_2$ of the modulus of the slope of M in its steepest point [8]. Note that in some cases independence can be recovered only asymptotically or for very large m. Obviously, for a system to be practical, independence should be recovered (at least to a good degree of approximation) for small m and large q/m.

3. The proposed solution

3.1. Map choice

The BS, which is often proposed as the most obvious map for random number generation, cannot be used *as is* in practical circuits, due to robustness issues related to the attraction basin of the invariant set $[-1, 1]$. Since the basin is $[-1, 1]$ itself, there is no margin to re-inject the state into the attractor if, for some reason, it gets out of it. As a result, the system is prone to diverging from its nominal behavior [9, 5, and references therein].

To overcome this issue, designers slightly lower the the map slope (e.g. as in $M(x) = (2 - \epsilon)x \bmod 2 - 1$, with ϵ small and positive) [9, 5]. With this the invariant set remains the same, but its basin of attraction is enlarged into $[-(1 - \epsilon)^{-1}, (1 - \epsilon)^{-1}]$, assuring robustness. Furthermore, if ϵ is rational, the system conserves a Markov nature[1]. The choice of ϵ influences the way in which the kneading matrix must be built and the number of Markov states one needs to introduce. In any case, since the map slope is < 2, x_n cannot be statistically uncorrelated and even processing the Markov states in groups one remains compelled to have $q/m < 1$. Too keep the processing simple a common strategy consists in generating a binary variable y_n at each iteration, which assumes the value 1 for all the Markov states where $x_n > 0$, and the value 0 otherwise. Then an output bit z_n is generated every m iterations as in $z_n = \bigoplus_{i=0}^{m-1} y_{i+m\cdot n}$, where $\oplus$ indicates logical ex-or. Thus the processing reduces to a nonlinear filtering and decimation which has been named *bit counting* [11]. The problems with this approach is that it produces rigorously independent bits only for very large (or asymptotically large) m, and that it makes q/m much smaller than 1 resulting very inefficient. In other terms, simplicity is paid by wasting most of the information delivered by the chaotic system.

To radically improve efficiency, we suggest using the recently proposed map $M(x) = (2x + 1) \bmod 2 - 1$ [12], which is plotted in figure 1 together with its Markov partition, kneading matrix and embedded Markov chain. The invariant set is still $[-1, 1]$ while, extrapolating the map, one can see that the basin of attraction is $[-2, 2]$ so that the system is robust.

Clearly the sequence of Markov states x_n cannot be independent, since the alphabet comprises 4 symbols that need 2 bits to be represented, while we cannot deliver more than 1 bit of information per iteration due to the slope (2) of the map. However, in this case one can build a rigorously independent binary sequence from x_n in a most efficient way. In fact, it is sufficient to aggregate the Markov states into the two macro-states shown in dotted lines. Its straightforward to prove that this results in a new Markov chain, which describes a process whose symbols are independent from each other. Operatively, one can achieve a result identical to the above mentioned aggregation by quantizing the analogue state

[1]While for an arbitrary ϵ one can still think of a finite kneading matrix to *approximately* describe the system behavior [10].

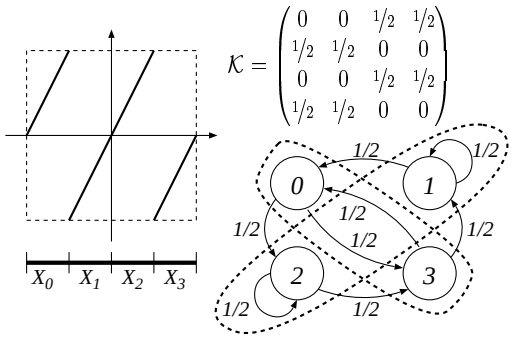

Figure 1: The proposed map.

variable x_n into a signed two bit word λ_n, corresponding to -1 for $x_n \in X_0$, to 0 for $x_n \in X_1 \cup X_2$ and to +1 in X_3. Then, one can simply look at the lsb of λ_n to know in which macro-state x_n falls at each timestep. Note that the fact that the states get aggregated shows that some data need to be discarded to build an independent bitstream.

Alternatively, one can consider the higher edge chain [7] obtained by following the chaotic trajectory over a few consecutive time steps [13]. For instance, if one follows the trajectories over two time-steps, the chain in figure 2 is obtained. Particularly, a

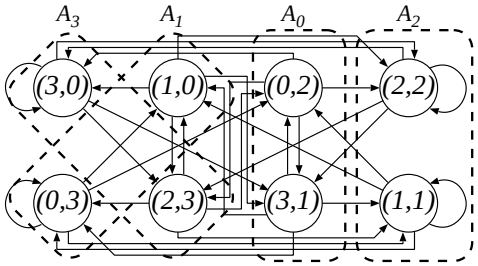

Figure 2: Higher edge Markov chain for the system in fig. 1, where the states are aggregated on two consecutive time steps. The transition probability is $1/4$ for every arrow.

new state $\bar{\mathsf{x}}_n = (a, b)$ is introduced to describe the situation where $(\mathsf{x}_{2n} = a) \wedge (\mathsf{x}_{2n+1} = b)$ and a departing arrow pointing to $\bar{\mathsf{x}}_{n+1} = (c, d)$ is drawn to report $\Pr[(\mathsf{x}_{2n+2} = c) \wedge (\mathsf{x}_{2n+3} = d)] \neq 0$. As it can be seen, the chain comprises states taken from an alphabet of 8 symbols and each symbol corresponds to 2 iterations of M. Hence, again, the sequence $\{\bar{\mathsf{x}}_n\}$ cannot be made of independent symbols. However, if one aggregates the states in macro-states as shown by the dashed lines in fig 2, a new Markov chain can be obtained, describing a process made of independent symbols. Although this might appear not completely straightforward, accept that operatively it can be done by quantizing the analogue state variable x_n into a signed two bit word λ_n as before. Then the λ values are added up in couples as in $\bar{\lambda}_n = 2\lambda_{2n} + \lambda_{2n+1}$. The msb of the words $\bar{\lambda}_n$ is eventually discarded and, with this, one gets an independent sequence of binary words.

The approach can be generalized to higher edge chains of arbi-

trary order m, which precisely corresponds to processing the the states x_n in groups of m states. It can be seen that operatively one needs to follow the obvious generalization of the above procedure: for each x_n a two bit signed word λ_n is produced; then the words are added up in m-uples, as in

$$\bar{\lambda}_n = \sum_{i=0}^{m-1} 2^{m-i-1} \lambda_{m \cdot n+i} \tag{2}$$

eventually, by stripping the msb of each $\bar{\lambda}_n$, one gets an independent sequence of binary words.

Note that, as long as the function M is computed *with no errors*, all the techniques are equivalent in that they all deliver rigorously independent bits with the same efficiency (1 bit for every map iteration, i.e. the maximum possible rate). As it will be seen, things change when small errors are introduced in the computation of M.

3.2. Architecture and implementation

Rather than seeking the direct implementation of $x_{n+1} = M(x_n)$, we suggest building a parallel system where a whole state vector comprising l scalars is updated at each iteration. As an example, consider $\vec{x}_{n+1} = M(\vec{x}_n)$, where $\vec{x}_n = (x_n^0, \ldots, x_n^{l-1})$. Replicating the hardware is costly but useful: in no physical system it can be possible to implement M exactly, analogue realizations bring unavoidable errors. Errors can compromise the operation of the algorithms described above, so they should be kept as small as possible. In analogue discrete-time systems error are typically related to the cycle time, i.e. the more rapidly we try to compute $M(x_n)$ the larger the errors we get. Exploiting replication, we can keep the computation time of $M(x_n)$ large at will without reducing the output rate.

Instead of implementing the model $\vec{x}_{n+1} = M(\vec{x}_n)$, we propose realizing in hardware its *interleaved version*:

$$\begin{cases} x_{n+1}^0 = & M(x_n^{l-1}) \\ x_{n+1}^1 = & M(x_n^0) \\ \ldots & \\ x_{n+1}^{l-1} = & M(x_n^{l-2}) \end{cases} \tag{3}$$

The equivalence between the two is self evident and relies on x_n^i in the non-interleaved model being the same as $x_n^{(n+i) \bmod l}$ in the interleaved one. The interleaved system can be realized as shown in the block diagram in figure 3. A notable advantage is that in a

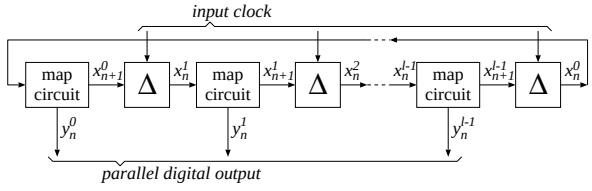

Figure 3: Interleaved pipelined architecture.

non-interleaved architecture an error in the implementation of the map circuit j can result in a sub-sequence $\{x_n^j\}$ with bad statistical features. In the interleaved scheme, each of the l chaotic systems uses all the l map circuits, so that deviations from the nominal behavior can average together, suggesting the possibility of better overall behaviors. However, apart from this, the major advantage is undoubtedly the possibility to follow closely the architecture of pipeline ADCs, which is exactly the same as that in figure 3, when the output-to-input feedback loop is ignored. Even more interestingly, if one uses the chaotic map in figure 1 it becomes possible to completely re-cycle the analog pipeline of the recent converters based on $1 + 1/2$ bit cells [6]completing the last cell and introducing the output-to-input link. For instance, figure 4 shows an ADC cell taken from [6], which precisely realizes the map in figure 1. Note that being based on a switched capacitor architecture, the cell

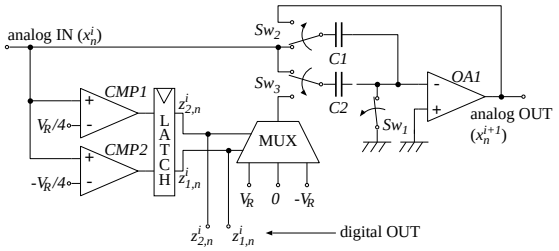

Figure 4: Pipeline stage realizing the map in figure 1 and proposed in [6] for ADC synthesis.

realizes both a map circuit block and its associated delay element, as shown in figure 3. Note also that the addition of an output-to-input link does not compromise the possibility of exploiting all the fine solutions proposed in [6] or in other ADC implementations to enhance accuracy, reduce the voltage supply or achieve other advantages.

In pipeline ADCs with $1 + 1/2$ bit cells the output of each cell corresponds to the output of two comparators, as shown in figure 4. In this example, the variable $z_{1,n}^i$ is active if the state variable x_n^i belongs to the partition intervals X_1, X_2 or X_3, while $z_{2,n}^i$ is active only in X_3. Hence, it is straightforward to apply the digital algorithms described in section 3.1, which require the two-bit signed λ values to be simply be computed at each cell. The lsb of λ_n^i, with $i = 1, \ldots, l$, is $z_{2,n}^i \oplus z_{1,n}^i$, while the msb is $\overline{z_{2,n}^i}$.

For the first, simplest algorithm it is sufficient to observe the aggregation of states X_0,–X_3 and X_1–X_2 and for this the lsb of λ_n^i suffices. Hence, at each cell one has only to take the ex-or of $z_{1,n}^i$ and $z_{2,n}^i$: with this, for a pipeline comprising l cells, l truly random bits are directly generated at each iteration, one per cell.

To apply the more complex algorithms, one needs to accumulate the λ values as shown by formula (2). However, the latter needs to be slightly modified, remembering that the architecture is interleaved: while the state variables are updated *in time*, they are also shifted *in space* from cell to cell. In other terms, the "aggregations in time" described in section 3.1 have to become "aggregations in space and time" on the interleaved architecture. For instance, for $m = 2$ one has to divide the architecture in couples of sibling cells. Then, the sum (2) is computed for each couple. As an example, for the couple composed by the first two cells one has $\bar{\lambda}_n^{0,1} = 2 \cdot \lambda_{n-1}^0 + \lambda_n^1$. Note that the past value λ_{n-1}^1 needs to be stored in a register to perform this sum. Note also that $\lambda_n^{1,2}$ is a 3 bit word whose msb needs to be discarded to obtain 2 independent bits.

The highest level of aggregation that one can comfortably apply is the one where the aggregation level m coincides with the pipeline length l. In this case, at each iteration one computes the $l+1$ bit word

$$\bar{\lambda}_n = \sum_{i=0}^{l-1} 2^{l-i-1} \cdot \lambda_{n-l+i+1}^{i} \tag{4}$$

Then the msb is discarded, as usual, to obtain l independent bits. Interestingly, equation 4 represents the computation that is performed onto the outputs of the individual cells of $1 + {}^1/_2$ bit per cell pipeline ADCs to obtain the conversion result. Hence, also the digital part of pipeline ADCs can be re-used for chaos based random number generation.

4. Simulation results

Since the computation of equation (4) is by far more complex than taking the simple ex-or of each individual cell (the simplest algorithm), one might wonder whether there are advantages in its application apart from re-using the whole of an ADC design rather than only a part of it. As we have already said, if the cell implementation is ideal, the various approaches are equivalent. However, real systems are never perfect, so that the way in which the implementation errors influence the various algorithms is probably the correct metric to judge them. Errors can be modeled as *perturbations* such as *slope errors*, *breakpoint misplacements* and *offsets*, applied to the various pipeline stages. Here the major problem is that as soon as one starts considering even a few perturbations, the system behavior becomes unmanageable from a formal point of view and one must resort to simulation.

Intuitively, one might be tempted to think that operating a *global* mixing of all the available information as in equation (4) could be preferable to directly outputting the data generated *locally* by each cell. To check whether this is true, we have run several simulations on a model 8-stage architecture, where we could independently perturb each parameter of each individual cell, and we could plug different digital algorithms at the pipeline outputs. We have run randomicity tests by generating $8 \cdot 10^6$ bits at each trial and by passing them through a Lempel-Ziv (LZ77) coding unit and a Burrows-Wheeler block sorting (BWS) compressor. Compression tests are coarse, but can be useful in a preliminary investigation since the relevant algorithms are highly optimized for space and speed. First we have checked the incompressibility of sequences generated by ideal systems; then we have applied *faults* and measured the consequent compression ratios. Table1 shows a condensed version of our results.

Simulations suggest that, contrarily to intuition, *local* operation (i.e. without aggregations in space and time) is fine if not better than the more complex algorithms, which is obviously very convenient from an implementative point of view. In a near future, we intend to complement these preliminary results with more thorough tests and to look for formal explanations for the phenomenon.

Test	Algorithm: No aggreg.	Algorithm: 2 bit aggreg.	Algorithm: 8 bit aggreg.	
A	$-182 \cdot 10^{-6}$	$-182 \cdot 10^{-6}$	$-182 \cdot 10^{-6}$	LZ77
	$-5.09 \cdot 10^{-3}$	$-4.69 \cdot 10^{-3}$	$-5.05 \cdot 10^{-3}$	BWS
B	$-182 \cdot 10^{-6}$	$-80 \cdot 10^{-6}$	$1.34 \cdot 10^{-3}$	LZ77
	$-4.67 \cdot 10^{-3}$	$-5.21 \cdot 10^{-3}$	$-4.67 \cdot 10^{-3}$	BWS
C	$1.42 \cdot 10^{-3}$	$42.6 \cdot 10^{-3}$	$40.4 \cdot 10^{-3}$	LZ77
	$50.1 \cdot 10^{-3}$	$43.5 \cdot 10^{-3}$	$40.1 \cdot 10^{-3}$	BWS
D	$-182 \cdot 10^{-6}$	$6.87 \cdot 10^{-3}$	$57.3 \cdot 10^{-3}$	LZ77
	$-3.05 \cdot 10^{-3}$	$-1.01 \cdot 10^{-3}$	$42.1 \cdot 10^{-3}$	BWS

Table 1: Compression ratios achieved on the sequences obtained by simulation (the lower, the better). Negative values mean that the sequence is incompressible and result from the addition of the coder tables to the compressed data. Tests: A) no errors on the units; B) +50% slope error on one unit; C) -25% slope error on one unit; D) 12.5% breakpoint misplacement on one unit.

References

[1] D. E. Eastlake, S. D. Crocker, and I. S. Jeffrey, "RFC 1750: Randomness recommendations for security", in *Request for Comments*, Internet Engineering Task Force, 1994.

[2] G. M. Bernstein and M. A. Lieberman, "Secure random number generation using chaotic circuit", *IEEE Transactions on Circuits and Systems*, vol. 37, pp. 1157–1164, Jan. 1990.

[3] T. Kohda and A. Tsundeda, *Information Sources Using Chaotic Dynamics*, ch. 4. In [14], 2000.

[4] B. Jun and P. Kocher, "The Intel® random number generator", tech. rep., Criptography Research Inc for Intel Corporation, Apr. 1999. Available on *www.intel.com/design/security/rng/rngppr.htm*.

[5] M. Delgado-Restituto and A. Rodriguez-Vazquez, "Integrated chaos generators", *IEEE Proceedings*, vol. 90, pp. 747–767, May 2002.

[6] A. M. Abo and P. R. Gray, "A 1.5-V, 10-bit, 14.3-MS/s CMOS pipeline analog-to-digital converter", *IEEE Journal of Solid State Circuits*, vol. 34, pp. 599–606, May 1999.

[7] D. A. Lind and B. Marcus, *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, 1995.

[8] E. Ott, *Chaos in dynamical systems*. Cambridge University Press, 1993.

[9] S. Callegari, R. Rovatti, and G. Setti, *Robustness of Chaos in Analog Implementations*, ch. 12, pp. 397–442. In [14], 2000.

[10] G. Setti, G. Mazzini, R. Rovatti, and S. Callegari, "Statistical modeling of discrete time chaotic processes: Basic finite dimensional tools and applications", *IEEE Proceedings*, vol. 90, pp. 662–690, May 2002.

[11] T. Stojanovski and L. Kocarev, "Chaos-based random number generators — part II: Practical realization", *IEEE Transactions on Circuits and Systems, Part I*, vol. 48, no. 3, pp. 382–385, 2001.

[12] M. Delgado Restituto and A. Rodríguez-Vásquez, "Piecewise affine markov maps for chaos generation in chaotic communication", in *Proceedings of ECCTD'99*, vol. 2, (Stresa), pp. 453–458, Aug. 1999.

[13] S. Callegari, R. Rovatti, and G. Setti, "ADC-based design of chaotic truly random sources", in *Proceedings of NDES'02*, (Izmir), pp. 5/9–5/12, June 2002.

[14] M. P. Kennedy, R. Rovatti, and G. Setti, eds., *Chaotic Electronics in Telecommunications*. Boca Raton: CRC International Press, 2000.