Construction of dead-beat switched automata: application to cryptography

Hamid Boukerrou, Gilles Millérioux, Marine Minier, Mirko Fiacchini

Construction of dead-beat switched automata: application to cryptography*

Hamid Boukerrou[1] and Gilles Millérioux[1] and Marine Minier[2] and Mirko Fiacchini[3]

***Abstract*— The work is concerned with dead-beat stability of autonomous discrete-time switched linear systems, having in mind a potential application to cryptography. As far as control theory is concerned, we propose an algorithm to construct a switched system whose shorter dead-beat stabilizing sequence has a prescribed length. We discuss the pecularities when the dynamical systems under consideration are defined over finite fields. Then, it is shown how the algorithm can be used to addressed the design of self-synchronizing stream ciphers involving switched automata.**

I. INTRODUCTION

Switching systems are dynamical systems for which the state dynamics vary between different operating modes. The switch is orchestrated according to a switching sequence [1]. They are relevant models in many fields. As typical examples, we can quote networked control systems [2], [3], congestion control for computer networks [4], viral mitigation [5], abstractions of complex hybrid systems [6], and many other ones (see e.g. [7], [8], [9] and references therein). Stability as well as stabilization of switching systems has been the purpose of many works ([10], [11], [12], [9], [13], [14]). Among many issues related to stability of this class of systems, characterizing dead-beat stabilizability for discrete-time switched linear systems still remains an open problem. However, to go further, the paper [15] established a necessary and sufficient condition and proposed an algorithm for constructing dead-beat stabilizable switched systems. In the present paper, we bring the connection between such a control-theoretic issue and the design of ciphers called self-Synchronizing Stream Ciphers (SSSC for brevity) used in the realm of cryptography.

It is interesting to note that for years, cryptography (see [16] for an introduction) has been used to secure data and systems. The aim is to directly protect data conveyed through public channels. Since 2015, cryptography has entered the scene of Cyber-Physical Systems (CPS) and control theory in an unprecedented fashion. Indeed, several attempts to incorporate cryptography into networked control systems have appeared to enhance cybersecurity. It is the concept of encrypted control as reported for example in [17]. More generally, the recent literature reveals that automatic control can be relevant to tackle security of Cyber Physical Systems (CPS). As examples, the paper [18] (and references therein) investigates the concept of *Covert channel*, originated in 1973 by Butler Lampson. The paper [19] is devoted to a bibliographical review and addresses control-oriented perspectives for CPS security. Bringing together the issues of security and automatic control is one of the objective of the present paper.

More precisely, this work is concerned with symmetric cryptography, in particular stream ciphers. They are based on automata that are dynamical systems operating on finite fields. Roughly speaking, they aim at delivering sequences of symbols, named keystreams, of high complexity from a statistical point of view. Then, those sequences are used to scramble information. In this context, the motivations of the present paper are the following. Having in mind a trade-off between high statistical complexity of keystream sequences and low architectural complexity for the sake of implementation, it turns out that switched linear systems are potential good candidates as keystream generators. As a general principle of stream ciphers, the keystream generated by the automata at the cipher and decipher sides must be synchronized to ensure proper decryption. To achieve the synchronization, the dynamical systems, at both sides, should get specific properties. In this paper, it will be shown that finite-time stabilizability will be central. The pecularities due to finite fields will be highlighted.

The outline of the paper is the following. Section II recalls the general principle of SSSCs. The role of dynamical systems in this context is emphasized. Then, after motivating the use of the special class of switched linear systems as keystream generators of the ciphers, a connection between the necessary property of finite input memory for the generators and dead-beat stabilizability is brought out. Section III is devoted to the methodology to construct a statistical SSSC mainly based on the construction of dead-beat stabilizing sequences. The pecularities due to the fact that the automata operate on a finite field are discussed. Finally, a complete code written in SageMath is provided along with an illustrative example describing the design of a statistical SSSC and its execution.

Notation: For a vector z_t indexed by the time $t \in \mathbb{N}$, z_t^i denotes its i-th component. Given $n \in \mathbb{N}$, define $\mathbb{N}_n = \{j \in \mathbb{N} : 1 \leq j \leq n\}$. The set of q switching modes is $\mathcal{I} = \mathbb{N}_q$ and the related matrices forms a finite collection $\mathcal{A} \subseteq \mathbb{R}^{n \times n}$,

*This work was partly supported by the french PIA project Lorraine Université d'Excellence, reference ANR-15-IDEX-04-LUE.

[1]Hamid Boukerrou and Gilles Millérioux are with Université de Lorraine, CNRS, CRAN, F-54000 Nancy, France {hamid.boukerrou,gilles.millerioux}@univ-lorraine.fr

[2]Marine Minier is with Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France {marine.minier}@univ-lorraine.fr

[3]Mirko Fiacchini is with Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, 38000 Grenoble, France mirko.fiacchini@gipsa-lab.grenoble-inp.fr>

whose i-th element is denoted with A_i, i.e. $\mathcal{A} = \{A_i\}_{i\in\mathcal{I}}$, with $A_i \in \mathbb{R}^{n\times n}$ for all $i \in \mathcal{I}$. The mode at time t of a switched system is denoted $\sigma(t)$. All the possible sequences σ of modes of length N is $\mathcal{I}^N = \prod_{j=1}^{N} \mathcal{I}$, with $\mathcal{I}^N = \emptyset$ if $N = 0$. The length of a sequence σ is denoted by $|\sigma|$ and $|\sigma| = N$ if $\sigma \in \mathcal{I}^N$. Given σ, δ sequences of modes, (σ, δ) is their concatenation. Given $\sigma \in \mathcal{I}^N$, define $\mathbb{A}_\sigma = \prod_{j=1}^{N} A_{\sigma(j)} = A_{\sigma(N)} \cdots A_{\sigma(1)}$ and $\prod_{j=n_0}^{n_1} A_{\sigma(j)} = I_n$ if $n_0 > n_1$ where I_n stands for the identity matrix.

II. Keystream Generators for Self-Synchronizing Stream Ciphers

The principle of using a state automaton to design a keystream generator has been first suggested in [20]. Hereafter, the automaton operates on a finite field $\mathbb{F}$. Thus, necessary basics on finite field should be recalled. It is the purpose of the following subsection.

A. Basics on finite fields

A field is a 3-uplet ($\mathbb{F}$,+,·) where $\mathbb{F}$ is a set, + an operation usually called *addition* and · an operation usually called *multiplication*. The following properties apply

1) For all a and b in $\mathbb{F}$ both $a + b$ and $a \cdot b$ are $\mathbb{F}$,
2) For all a, b and c in $\mathbb{F}$, associativity holds: $a+(b+c) = (a + b) + c$ and $a \cdot (b \cdot c) = (a \cdot b) \cdot c$,
3) For all a and b in $\mathbb{F}$, commutativity holds: $a+b = b+a$ and $a \cdot b = b \cdot a$,
4) There exists an element of $\mathbb{F}$, called the additive identity element and denoted by 0 such that for all a in $\mathbb{F}$, $a + 0 = 0 + a = a$. Likewise, there is an element called the multiplicative identity element and denoted by 1, such that for all a in $\mathbb{F}$, $a \cdot 1 = a$. The identity elements 0 and 1 have to be different,
5) For every a in $\mathbb{F}$, there exists an element $-a$ in $\mathbb{F}$ such that $a + (-a) = 0$. Similarly, for any a in $\mathbb{F}$ other than 0, there exists an element a^{-1} in $\mathbb{F}$ such that $a \cdot a^{-1} = 1$,
6) For all a, b and c in $\mathbb{F}$ distributivity of the multiplication over the addition holds: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.

The number of elements of a finite field is called its order. A finite field of order Z exists if and only if Z is a prime power Q^m (where Q is a prime number and m is a positive integer). Thus, the field $\mathbb{F}_{Q^m} = GF(Z)$ (where GF stands for Galois Field) may be explicitly constructed in the following way. One first chooses an irreducible polynomial P in the ring of polynomials defined by the variable X, $GF(Q)[X]$ of degree m (such an irreducible polynomial always exists). Then, the elements of $GF(Z)$ are the polynomials over $GF(Q)$ whose degree is strictly less than m. The addition and the subtraction are those of polynomials over $GF(Q)$. The product of two elements is the remainder of the Euclidean division by P of the product in $GF(Q)[X]$. The multiplicative inverse of a non-zero element may be computed with the extended Euclidean algorithm.

B. General principle of SSSC

At the ciphering side, the automaton delivering the keystream takes the form:

$$x_{t+1} = g_K(x_t, c_t) \tag{1}$$

where $t \in \mathbb{N}$ is the discrete-time, $x_t \in \mathbb{F}^n$ is the internal state, g_K is the next-state transition function parameterized by the secret key $K \in \mathbb{F}^L$, being L the length of the secret key. The symbol $c_t \in \mathbb{F}^m$ is the cryptogram. It is calculated by

$$c_t = e(z_t, m_{t-r}), \tag{2}$$

where $m_t \in \mathbb{F}^m$ is the plaintext symbol and $z_t \in \mathbb{F}^m$ is called the keystream (or running key) symbol that is computed by

$$h : x_t \in \mathbb{F}^n \mapsto z_t = h(x_t) \tag{3}$$

The function e is the encrypting function and is invertible for any z_t. The integer $r \geq 0$ stands for a possible delay between the plaintext m_t and the corresponding keystream z_t and ciphertext c_t symbols. It is sometimes introduced for computational or implementation reasons.

By iterating (1) a finite number of times, if there exists a function ℓ_K and two finite integers ℓ and ℓ' (with $\ell, \ell' \in \mathbb{Z}$ and $|\ell'| \geq |\ell|$, $|\cdot|$ stands for the absolute value) such that

$$x_t = \ell_K(c_{t-\ell}, \ldots, c_{t-\ell'}). \tag{4}$$

then,

$$z_t = h(\ell_K(c_{t-\ell}, \ldots, c_{t-\ell'})). \tag{5}$$

Otherwise stated, if after a finite number of iterations, the current state of the automaton (1) will only depend on shifted cryptograms, such an automaton is called a finite input memory automaton according to [20]. The word input is used since the cryptogram c_t is considered as the input of the automaton (1).
Actually, the fact that the keystream symbol can be written in the general form

$$z_t = \alpha_K(c_{t-\ell}, \ldots, c_{t-\ell'}), \tag{6}$$

with α_K a function involving a finite number of shifted ciphertexts from time $t - \ell$ to $t - \ell'$ ($\ell, \ell' \in \mathbb{Z}$), is the central feature of the SSSC. Equation (6) is called the canonical equation. The integer $M = |\ell'| - |\ell| + 1$ is called the delay of memorization.
At the deciphering side, the automaton takes the form

$$\widehat{x}_{t+1} = g_K(\widehat{x}_t, c_t), \tag{7}$$

where $\widehat{x}_t \in \mathbb{F}^n$ is the internal state and $\widehat{z}_t \in \mathbb{F}^m$ verifies

$$\widehat{z}_t = h(\widehat{x}_t) \tag{8}$$

which stands as the keystream symbol similarly to z_t at the ciphering side.

The decryption function d obeys the following rule. For any two keystream symbols $\widehat{z}_t, z_t \in \mathbb{F}^m$, it holds that

$$\widehat{m}_t := d(c_t, \widehat{z}_t) = m_{t-r} \text{ whenever } \widehat{z}_t = z_t. \tag{9}$$

The automaton has the same dynamics than the cipher, thus it is a finite input memory one. It means that iterating Equation (7) a finite number of times also yields

$$\hat{x}_t = \ell_K(c_{t-\ell}, \ldots, c_{t-\ell'}),$$

and thus,

$$\hat{z}_t = \alpha_K(c_{t-\ell}, \ldots, c_{t-\ell'}).$$

Hence, it is clear that after a transient time of maximal length equal to M, it holds that, for $t \geq M$,

$$\widehat{x}_t = x_t \text{ and } \widehat{z}_t = z_t. \tag{10}$$

In other words, the keystream generators synchronize automatically after at most M iterations. Hence, the decryption is automatically and properly achieved after at most M iterations too. No specific synchronizing protocol between the cipher and the decipher is needed. This explains the terminology Self-Synchronizing Stream Ciphers and it is one of its practical interest.

Two classes of SSSC can be defined according to the delay of synchronization:

- *Deterministic*: the delay of synchronization is bounded by the constant M a priori fixed.
- *Statistical*: the bound of the delay of synchronization is not constant but is a random variable with respect to the sequence of ciphertexts or the initial state vector.

The general architecture of a SSSC is depicted in Figure 1. Statistical SSSC have been introduced for the first time

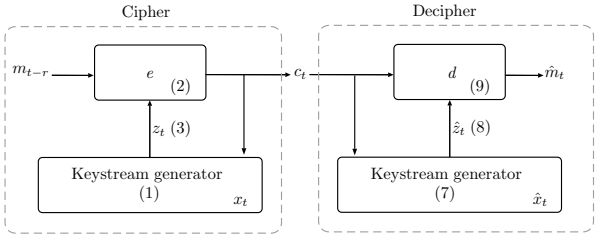

Fig. 1. Self-Synchronizing Stream Cipher block diagram.

in [21]. Essentially, this concept calls for several modes orchestrated by a switching rule. It is motivated by the fact that each mode benefits from a specific outcome in terms of security but on the other hand they may suffer from weakness in terms of computational complexity for example. The hybrid architectures allows to getting a trade-off. The present paper focuses on this class of SSSC.

For SSSC, it has been stressed that the state vector x_t of the automaton (1) must be expressed as a function of a finite number of its shifted outputs as described by Equation (4). In next section, the connection between the design of SSSC, that are dynamical systems having this specific property, and the construction of dead-beat stabilizable switched systems is brought out.

C. The connection between the properties of finite input memory automata and dead-beat stabilizability

Switched linear systems correspond to the Maiorana Mc-Farland construction which has proved to produce functions that have many interesting cryptographic properties like high non linearities, high correlation immunity and good propagation characteristics (see [22] when considering Boolean functions defined over the two-element field). Hence, as a special case of the state transition function g_K of (1), it is interesting to consider the nonlinear automaton in the form

$$x_{t+1} = P_{\sigma(t)}x_t + R_{\sigma(t)}c_t \tag{11}$$

where $x_t \in \mathbb{F}^n$ is the state at time $t \in \mathbb{N}$ and σ is a switching rule which depends in a nonlinear way on a finite sequence of past ciphertexts $c_t, \ldots, c_{t-s}$ with $s \in \mathbb{N}$. Hence, at each time t, the switching function selects the mode by performing $\sigma(t) = \varphi(c_t, c_{t-1}, \cdots, c_{t-s})$ in $\mathcal{I}$. The n-dimensional square matrix $P_{\sigma(t)}$ belongs to a finite set of q matrices with entries in $\mathbb{F}$ and $R_{\sigma(t)}$ is a $(n \times m)$-dimensional vector with entries in $\mathbb{F}$.

As a clue to tackle the design of statistical SSSC, it is worthwhile realizing that the property of finite input memory with a statistical delay of memorization expressed by Equation (4) is obtained for (11) whenever there exists at least a switching sequence of length N so that $\prod_{i=1}^{N} P_{\sigma(t+i-1)} = 0$. It turns out that it is related to the property of dead-beat stabilizability. Indeed, let us consider the so-called auxiliary system of (11) defined by

$$x_{t+1} = A_{\sigma(t)}x_t \tag{12}$$

where $x_t \in \mathbb{F}^n$ is the state at time $t \in \mathbb{N}$ and $A_{\sigma(t)}$ is the n-dimensional square matrix that verifies $A_{\sigma(t)} = P_{\sigma(t)}$ for all $t \in \mathbb{N}$. The following definition is recalled from [15].

Definition 1: The system (12) is dead-beat stabilizable if and only if

$$\exists N \in \mathbb{N}, \quad \exists \gamma \in \mathcal{I}^N \quad \text{s.t.} \quad \mathbb{A}_\gamma = 0. \tag{13}$$

All in all, the automaton defined by Equation (11) has a finite input memory whenever the auxiliary system (12) is dead-beat stabilizable. Since the switching rule σ depends on past ciphertexts and that the ciphertexts are assumed to be random and uniformly distributed (a common feature of the cipher), the time before a stabilizing switching sequence appears is also random. As a result, the upper bound M of the memorization delay is a random variable. The resulting SSSC belongs to the class of statistical, what is precisely our objective.

III. Methodology to build a SSSC

This section aims at giving a methodology along with a code for SSSC design perspectives. To this end, first, we recall a Necessary and Sufficient Condition for dead-beat stabilizability and the algorithm that allows to construct a

dead-beat stabilizable system. Then, we highlight the peculiarities due to the consideration of finite fields. The notion of orthogonality deserves a special treatment. Finally, we propose a code written in SageMath along with an illustrative example.

A. Dead-beat stabilizability and algorithm

Let us consider the systems described by Equation (12). For this system, let us introduce the following notation

$$\begin{aligned} \mathcal{I}_s &= \{i \in \mathcal{I} : \ \det A_i = 0\}, \\ \mathcal{I}_{ns} &= \{i \in \mathcal{I} : \ \det A_i \neq 0\}, \end{aligned} \tag{14}$$

the sets of modes of singular and nonsingular matrices A_i in $\mathcal{A}$ and by q_s and q_{ns} the number of elements of $\mathcal{I}_s$ and $\mathcal{I}_{ns}$, respectively. Clearly, one has $q_s + q_{ns} = q$ where it is recalled that q is the number of modes of the system (12).

The aim of this section is to illustrate how to build the set $\mathcal{A}$ of matrices such that condition (13) is satisfied for a given, desired $N \in \mathbb{N}$ but it is not for any $N' < N$. This means that the set $\mathcal{A}$ of matrices would allow to generate sequences γ with length greater than or equal to N such that $\mathbb{A}_\gamma = 0$, but not shorter ones, and then there exists only one sequence such that $\mathbb{A}_\gamma = 0$ among the q^N of length N at most. This property is useful to design a SSSC whose upper bound M of the memorization delay is a design parameter determined by q and N.

The main underlying idea of the proposed method relies on results from [23], [15], in which it is proved that γ such that $\mathbb{A}_\gamma = 0$ exists, and hence (12) is dead-beat stabilizable, if and only if there is a set of $\tilde{m}$ switching sequences σ^p of finite length r^p, with $p \in \mathbb{N}_{\tilde{m}}$ and $1 \leq \tilde{m} \leq n$, whose last element is related to a singular matrix, and such that the intersection of the kernel of $\mathbb{A}_{\sigma^p}$ and the image of the product $\prod_{k=1}^{p-1} \mathbb{A}_{\sigma^k}$ of matrices has a dimension strictly greater than zero. In particular if $\tilde{m} = n$ and denoting with $X_{p-1} \in \mathbb{R}^{n \times n-p+1}$ a basis matrix of $\prod_{k=1}^{p-1} \mathbb{A}_{\sigma^k}$, then the necessary and sufficient condition recalled above is equivalent to

$$\dim \ker \left(\mathbb{A}_{\sigma^p} X_{p-1}\right) = 1 \tag{15}$$

for all $p \in \mathbb{N}_n$, which implies that

$$\prod_{k=1}^{n} \mathbb{A}_{\sigma^k} = 0 \tag{16}$$

and then (13) holds with $\gamma = (\sigma^1, \ldots, \sigma^n)$ and $|\gamma| = N$. It can be also proved that $|\sigma^1| = 1$, see [23], [15].

The proposed approach consists in generating a set $\mathcal{A}$ of matrices for which sequences σ^p exist such that (15) holds, and then also (16). Moreover, by choosing the length $r^p \in \mathbb{N}$ of the subsequences, i.e. such that $|\sigma^p| = r^p$, for $p \geq 2$, the desired length N of the sequence γ can be fixed since $|\gamma| = 1 + \sum_{p=2}^{n} r^p$ (let us recall that $|\sigma^1| = 1$). The algorithm presented in [15] is recalled and commented hereafter, more detailed explanations can be found in the cited paper.

Algorithm 1 Build a dead-beat stabilizable system.

Input: q_s and q_{ns} cardinalities of $\mathcal{I}$ and $\mathcal{I}_s$, subsequences lengths r^p.

1: $\mathcal{I}_s \leftarrow \emptyset$ ▷ Initialization
2: $\mathcal{I}_{ns} \leftarrow \emptyset$
3: $\mathcal{A} \leftarrow \emptyset$
4: **for** $s \in \mathbb{N}_{q_s}$ **do** ▷ Generate $\mathcal{I}_s$
5: generate A_s singular ▷ Item (i)
6: insert s in $\mathcal{I}_s$
7: insert A_s in $\mathcal{A}$
8: **end for**
9: **for** $j \in \mathbb{N}_{q_{ns}-n+1}$ **do** ▷ Generate a part of $\mathcal{I}_{ns}$
10: generate A_{q_s+j} nonsingular ▷ Item (ii)
11: insert $q_s + j$ in $\mathcal{I}_{ns}$
12: insert A_{q_s+j} in $\mathcal{A}$
13: **end for**
14: random selection of $s_1 \in \mathcal{I}_s$ ▷ First step
15: $\sigma^1 \leftarrow s_1$
16: **for** $p \in \{i \in \mathbb{N} : 2 \leq i \leq n\}$ **do** ▷ p-th step
17: random selection of α_p and β_p ▷ Item (a)
18: random selection of $s_p \in \mathcal{I}_s$ ▷ Item (b)
19: compute B_p ▷ Item (c)
20: $A_{i_p} \leftarrow B_p$ ▷ Item (d)
21: insert i_p in $\mathcal{I}_{ns}$
22: insert A_{i_p} in $\mathcal{A}$
23: $\sigma^p \leftarrow (\beta_p, i_p, \alpha_p, s_p)$
24: **end for**

Output: $\mathcal{A}$ ▷ Matrix set of the stabilizable system

The initialization (lines 4-13) consists in randomly computing the q_s singular matrices and $q_{ns} - n + 1$ nonsingular ones. In particular the nonsingular matrices are given by $A_{ns} = T_s$ and the singular ones by $A_s = T_s^{-1} \Lambda_s T_s$ with:

(i) Λ_s a diagonal matrix whose diagonal has 0 as first entry and the other ones are randomly generated but non-null.
(ii) T_s a randomly generated invertible matrix.

The first iteration of Algorithm 1 (line 15) consists in choosing the only matrix composing σ^1 among the singular ones. Then, the algorithm builds a sequence of subsequences σ^p (lines 16-24) such that (15) holds. To this end:

(a) compute the random sequences α_p and β_p of modes related to nonsingular matrices and such that $|\alpha_p| + |\beta_p| = r^p - 2$ (line 17). Note that $\mathbb{A}_{\alpha_p}$ and $\mathbb{A}_{\beta_p}$ are nonsingular;
(b) select $s_p \in \mathcal{I}_s$, then A_{s_p} singular (line 18);
(c) define (line 19)

$$\begin{aligned} B_p &= \mathbb{A}_{\alpha_p}^{-1} T_{s_p}^{-1} R_p C_p^{-1} \\ C_p &= \begin{bmatrix} \mathbb{A}_{\beta_p} X_{p-1} & (\mathbb{A}_{\beta_p} X_{p-1})^\perp \end{bmatrix} \end{aligned} \tag{17}$$

with

$$R_p = \left[\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \bar{R}_p & \\ 0 & & & \end{array}\right]$$

where $\bar{R}_p \in \mathbb{R}_p^{n-1 \times n-1}$ is an arbitrary nonsingular matrix and where $V^\perp$ is a basis of the subspace orthogonal

to the one spanned by the columns of V, implying the nonsingularity of C_p;

(d) define the new mode i_p such that $A_{i_p} = B_p$, define $\sigma^p = (\beta_p, i_p, \alpha_p, s_p)$ and include A_{i_p} in the matrix set $\mathcal{A}$ (lines 20-23). It is proved in [23] that $\dim \ker(A_{s_p} \mathbb{A}_{\alpha_p} B_p \mathbb{A}_{\beta_p} X_{p-1}) = 1$ and then (15) is satisfied with $\sigma^p = (\beta_p, i_p, \alpha_p, s_p)$.

B. The pecularities of finite fields

Generally speaking, if we have a vector space E over a field $\mathbb{F}$ and a bilinear form $B : E \times E \to \mathbb{F}$, it is said that v, w in E are orthogonal if $B(v, w) = 0$. For the special case $E = \mathbb{F}^n$, letting $v = (a_1, ..., a_n)$ and $w = (b_1, ..., b_n)$, the bilinear form is defined by $a_1 b_1 + ... + a_n b_n$. Orthogonality is central when examining the construction of matrix C_p (see Equation (17)). Indeed, given both matrices $\mathbb{A}_{\beta_p}$ and X_{p-1}, we must build a basis of the subspace orthogonal to $\mathbb{A}_{\beta_p} X_{p-1}$. And yet, regarding orthogonality, some pecularities due to the fact that we do not consider the field of real numbers, should be highlighted. For example, it is possible for an element of the vector space E over a field $\mathbb{F}$ to be "orthogonal to itself". It means that the bilinear form may vanish *i.e.* $B(v, v) = 0$ for $v \neq 0$. As an example, let us consider the field $\mathbb{F}_2$ as specified in Section II. The field $\mathbb{F}_2$ is the basic field with 2 elements $\{0, 1\}$ with the two laws: the addition which is the exclusive or XOR (*i.e.* the addition modulo 2) and the multiplication which is the logical and $\wedge$. It turns out that the vector $(1, 0, 1)^T$ is orthogonal to itself. Another pecularity should be highlighted and encompass the previous one. Let $E^{'\perp}$ be the subspace defined by

$$E^{'\perp} = \{v \in E | B(v, w) = 0 \text{ for all } w \in E'\}$$

If $E' \subset E$ is a k-dimensional subspace, then $E^{'\perp}$ is of dimension $n-k$. However, unlike in the field of real numbers, $(E, E^{'\perp})$ is not necessary a basis because it may happen that $E \cap E^{'\perp} \neq \emptyset$. In other words, elements of $E^{'\perp}$ may be linearly dependent from the basis of E while they are orthogonal to the elements of E. The fact that a vector can be orthogonal to itself enters such a situation. In our context, the consequence is that Algorithm 1 may fail when computing B_p at line 19. Indeed, C_p may not be invertible. As as example, let us consider again the field $\mathbb{F}_2$ and the matrices

$$\mathbb{A}_{\beta_p} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, X_{p-1} = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}^T$$

Then, we have that

$$\mathbb{A}_{\beta_p} X_{p-1} = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}^T$$

The vectors which are orthogonal to $(1, 0, 1)^T$ are all the vectors $(b_1, b_2, b_3)^T$ that fulfill $1b_1 + 0b_2 + 1b_3 = 0$. It yields the conditions $b_1 = b_3$ and b_2 arbitrary. Hence, the set of vectors orthogonal to $(1, 0, 1)^T$ is the space generated by $(1, 0, 1)^T$ and $(0, 1, 0)^T$. Consequently,

$$C_p = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

which is not invertible.

As another example, let us consider the matrices defined on the field $\mathbb{F}_4$

$$\mathbb{A}_{\beta_p} = \begin{pmatrix} 1 & 1 & \alpha+1 & 1 \\ 1 & \alpha & \alpha & 1 \\ \alpha+1 & 0 & \alpha & \alpha+1 \\ 0 & 1 & 1 & \alpha+1 \end{pmatrix},$$

$$X_{p-1} = \begin{pmatrix} 1 \\ \alpha \\ 0 \\ \alpha \end{pmatrix}$$

Then, we have that

$$\mathbb{A}_{\beta_p} X_{p-1} = \begin{pmatrix} 1 \\ 0 \\ \alpha \\ \alpha+1 \end{pmatrix}$$

The set of vectors orthogonal to $\mathbb{A}_{\beta_p} X_{p-1}$ is the space generated by:

$$(\mathbb{A}_{\beta_p} X_{p-1})^{\perp} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \alpha & 0 & \alpha+1 \end{pmatrix}$$

and

$$C_p = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \alpha & 0 & 0 & 1 \\ \alpha+1 & \alpha & 0 & \alpha+1 \end{pmatrix}$$

which is not invertible. Indeed, $v_1 = \alpha \cdot v_4 + v_2$ where v_i are the column vectors of C_p $(= 1, \ldots, 4)$.

It turns out that we can derive a necessary condition for Algorithm 1 to perform successfully at line 19. This condition is expressed as an inequality between the size of the field over which the SSSC operates and the dimension n of the system. It is based on a result borrowed from coding theory [24].

In coding theory, the purpose of an error-correcting code is to increase the ability to detect and correct transmission errors while not adding more overhead than necessary. A linear code takes a sequence of k symbols (the dimension of the code) and encodes it as a sequence of N symbols (the length of the code). These symbols come from an alphabet of size ℓ. Data are processed through a generator matrix G. Matrix G is a matrix whose rows form a basis for the linear code. It is a $k \times N$ matrix. Usually, error correcting codes are defined over finite fields such as $\mathbb{F}_{Q^m}$ as defined in Section II. The ability of a code to detect and correct errors is measured by its minimal distance d between code words. For the linear codes, the optimal minimal distance is reached by the codes called MDS (Maximum Distance Separable, see [24] for more details). For MDS codes, each extraction of the G matrix of square matrices is of full rank. This property is central for our purpose. Indeed, MDS codes could only be constructed on fields with a sufficient number of elements to ensure the full rank property [24]. This last remark guarantees an existence bound stated as a corollary given below.

Corollary 1: The finite field $\mathbb{F}_{Q^m}$ over which the SSSC operates, considering that the matrix C_p is of size $k \times (N-k)$ with $N = 2k$, must fulfill $N - k \leq Q^m + 1$.

Indeed, the standard form for a generator matrix G is $(I_k||H)$ where H is a $k \times (N-k)$ matrix ($||$ stands for the concatenation and it is recalled that I_k denotes the identity matrix of size k). Back to our context, consider that the matrix $(I_k||C_p)$ is a virtual generator matrix G. Since the size of G is $k \times N$ and the size of C_p is $n \times n$ (being n the dimension of the system), that amounts to setting $N = 2k$ and $k = n$. Hence, the inequality of Corollary 1 turns into

$$n \leq Q^m + 1 \tag{18}$$

It means that the size of the finite field $\mathbb{F}_{Q^m}$ must be chosen according to the dimension n of the system.

The SageMath code of Algorithm 1 is given in Appendix. The inputs are the dimension n of the system, the number of singular and nonsingular matrices q_s and q_{ns} at line 40, α_p, β_p at lines 76-77. The outputs are the sequence γ at line 132 and the corresponding matrices at line 61.

C. Proof Of Concept Example

We illustrate how to build an SSSC cipher and we show how it operates. All along this section, we will use the particular field $\mathbb{F}_4$ seen as the extension of degree 2 of the finite field $\mathbb{F}_2$. From this basic field, we construct $\mathbb{F}_4$ as a degree two extension, thus $\mathbb{F}_4$ is defined on $\mathbb{F}_2$ with the following polynomial of degree 2 with its coefficients over $\mathbb{F}_2$: $X^2 + X + 1$. Thus, we have $\mathbb{F}_4 = \mathbb{F}_2[X]/(X^2 + X + 1)$. An other way to describe $\mathbb{F}_4$ is to consider the root α of the polynomial $X^2 + X + 1$. Then, the elements of $\mathbb{F}_4$ are written with respect to α. More precisely, in this case, the elements of $\mathbb{F}_4$ are: (0,1, α, $\alpha + 1$). All those elements have a degree stricly smaller than 2 as the definition polynomial is of degree 2.

First, we must build a switching linear automaton like (11) that acts as the cipher part of an SSSC. The dimension of the automaton is $n = 4$. With such a setting, the inequality (18) is verified since $n = 4$, $Q = 2$ and $m = 2$ and so $n = 4$ is less than $2^2 + 1 = 5$. Thus, invertible matrices C_p exists and Algorithm 1 should not fail at line 19.

Next, we build a dead-beat stabilizable auxiliary system (12) and so, a dead-beat stabilizable sequences of matrices A_i $(i = 1, \ldots, q)$. Then, we set $P_i = A_i$ $(i = 1, \ldots, q)$ for (11). For simplicity, The matrices $R_{\sigma(t)}$ in (11) are all equal and set to the identity matrix.
The switching rule σ is defined by a mere one-to-one correspondance between the cryptogram $c_t \in \mathbb{F}_4$ (four possible elements) and the mode $\sigma(t) \in \{1, 2, 3, 4\}$ $(q = 4)$.
Let us consider an instanciation of the automaton (11) that reads

$$\begin{cases} x_{t+1} = P_{\sigma(t)} x_t + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} c_t \\ z_t = x_t^1 + x_t^2 \end{cases} \tag{19}$$

The encryption function (2) is defined as

$$c_t = z_t + m_t \tag{20}$$

The equations of the decipher (7) are exactly the same than (19) except the fact that the state vector x_t is replaced by the vector $\hat{x}_t$. The equations of the deciphering function (9) are also the same than (20). Indeed, the addition and the substraction on the field $\mathbb{F}_4$ coincide one another.

Let $\alpha_p = 1$, $\beta_p = 1$, $r^p = 4$ for all $p > 1$ and $r^1 = 1$ by construction. The number of singular and nonsingular matrices are respectively $q_s = 2$ and $q_{ns} = 2$. The corresponding matrices are respectively (A_0, A_1) and (A_2, A_3). As expected since inequality (18) is verified, Algorithm 1 performs successfully and gives

$$A_0 = \begin{pmatrix} \alpha & \alpha \\ 1 & 1 \end{pmatrix}, A_1 = \begin{pmatrix} 0 & \alpha + 1 \\ 0 & \alpha + 1 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} \alpha + 1 & \alpha \\ \alpha & 1 \end{pmatrix}, A_3 = \begin{pmatrix} \alpha + 1 & \alpha \\ 0 & 1 \end{pmatrix}$$

The sequence of modes γ is $\gamma = (\sigma^1, \sigma^2) = ((1), (2, 3, 2, 1))$.
A realization of a synchronization time plot is depicted on Figure 2, Figure 3 and Figure 4.

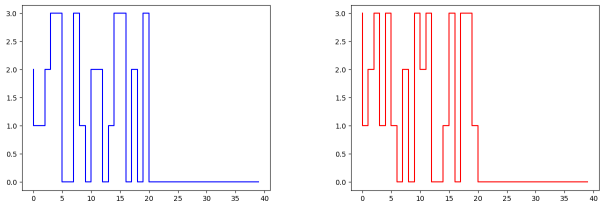

Fig. 2. Time evolution of $x_t^1 - \hat{x}_t^1$ (left) and $x_t^2 - \hat{x}_t^2$ (right)

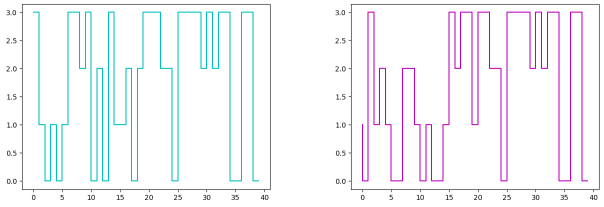

Fig. 3. Time evolution of m_t^1 (left) and $\hat{m}_t^1$ (right)

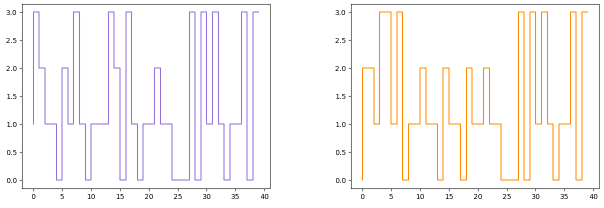

Fig. 4. Time evolution of m_t^2 (left) and $\hat{m}_t^2$ (right)

From arbitrary initial conditions x_0 and $\hat{x}_0$, a random sequence of plaintext symbols $m_t \in \mathbb{F}_4^2$ is generated. At $t = 15$, the dead-beat stabilizing sequence γ appears, causing the self-synchronization to succeed and so a proper decryption, after $M = 20$.

For random messages and random initial conditions, 1000 runs have been performed. The ratio between the number of successful resynchronizations after a time t and the total number of runs has been reported in Figure 5. As expected, the plot tends towards 1 as the time t before synchronization tends towards infinity. The stairs are explained by the fact that the ratio between the number of successful resynchronizations after a time t and the total number of runs has been not calculated for every t and is kept constant between two time windows.

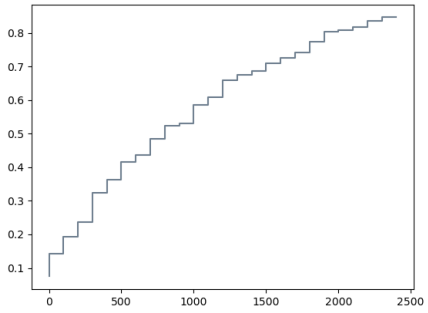

Fig. 5. Experimental probability of successful resynchronizations after a time t.

IV. Conclusion

It has been shown how the concept of dead-beat stabilizability can yield an approach for the design of self-synchronizing stream ciphers. This Proof-Of-Concept clearly deserves further investigation to give a complete cipher. As most of the remaining issues enter the area of pure cryptography, they are not addressed here but the main lines are mentioned in this conclusion. For a real-world application, automata with higher dimension and larger sequences should be used. The design approach remains unchanged and still work. However, the more the length of the dead-beat stabilizing sequences, the longer the time before synchronization. Hence, the setting of the parameters should obey a trade-off. Besides, as usual in cryptography, security analysis must be performed. The way how to incorporate the secret key is also an important issue.

Appendix

```
T_inverse = []
#Generate the singular matrice
def random_singular_matrix(field, size):

    test = True
    while(test == True):
         A = random_matrix(field, size)
         v_p = []
         vp = A.eigenspaces_left(format='galois')
         for i in range(len(vp)):
             v_p.append(vp[i][0])
         if A.rank() == size:
             test = 0 in v_p


    B = matrix(field, 1, 1, [0])
    A = block_matrix([ [B, 0], [0, A] ], subdivide = False)


    T = random_matrix(field , size + 1 , size + 1)

    while T.rank() != size + 1:
        T = random_matrix(field , size + 1 , size + 1)

    A = T^(-1) * A * T

    T_inverse.append(T^(-1))
    return A

#Generate the nonsingular matrice
def random_nonsingular_matrix(field, size):
     test = True
     while(test == True):
         A = random_matrix(field, size)
         test = A.is_singular()

     return A


def stabilizable_system(field, size, q_s, q_ns):

    I_s  = []
    I_ns = []
    A_s  = []
    A_ns = []

    sigma_p = []

    #Generate I_s
    for s in range(q_s):
        A = random_singular_matrix(field, size - 1)
        A_s.append(A)
        I_s.append(s)

    #Generate a part of I_ns
    for j in range(q_ns - size + 1):
        A = random_nonsingular_matrix(field, size)
        A_ns.append(A)
        I_ns.append(j + q_s)

    A = A_s + A_ns

    #The first subsequence, of one element, corresponds to a singular matrix
    s0 = choice(I_s)

    sigma_p = []
    sigma_p.append([s0])

    stable = [A[s0]]

    x_im = [A[s0]]

    for k in range(1, size):

        #Compute the random sequences a_p and b_p
```

```
        ap = [choice(list(I_ns)) for i in range(1)]
        bp = [choice(list(I_ns)) for i in range(1)]

        #select s_p in I_s
        sp = choice(I_s)

        #Compute A_ap
        A_ap = ones_matrix(field, size)
        A_ap = A[ap[-1]]

        for i in reversed(ap[:-1]):
            A_ap = A_ap * A[i]

        #Compute A_bp
        A_bp = ones_matrix(field, size)
        A_bp = A[bp[-1]]

        for i in reversed(bp[:-1]):
            A_bp = A_bp * A[i]

        #Compute R_p
        R = random_nonsingular_matrix(field, size - 1)
        B = matrix(field, 1, 1, [1])
        R_p = block_matrix([ [B, 0], [0, R] ], subdivide = False)

        #Basis matrix
        if (k == 1):
            A_p = A[s0]
            s = A_p.column_space()
            X = s.basis_matrix()

        if (k >= 2):
            A_p = A[sigma_p[-1][-1]]
            for i in reversed((sigma_p[-1][:-1]) ):
                A_p =  A_p * A[i]

            x_im.append(A_p)
            for i in reversed(x_im[:-1]):
                A_p =  A_p * i

            v = A_p.image()
            s = A_p.column_space()
            X = s.basis_matrix()


        #Compute C_p
        Cp = matrix(field, 0, size)
        Cp = block_matrix([[A_bp * X.transpose(), ((A_bp * X.transpose()).kernel()).matrix().transpose()]], subdivide = False)

        #Compute B_p
        B_p = A_ap^(-1) * T_inverse[sp] * R_p * Cp^(-1)
        A.append(B_p)

        I_ns.append(1 + I_ns[-1])

        #Compute sigma_p
        sigma_p.append(bp + [I_ns[-1]] + ap + [sp])
```

REFERENCES

[1] D. Liberzon. *Switching in Systems and Control*. Systems and Control: Foundations and Applica- tions. Birkhäuser, Boston, MA, 2003.

[2] Rajeev Alur, Alessandro D'Innocenzo, Karl Henrik Johansson, George J Pappas, and Gera Weiss. Compositional modeling and analysis of multi-hop control networks. *IEEE Transactions on Automatic control*, 56(10):2345–2357, 2011.

[3] Raphael M Jungers, Alessandro D'Innocenzo, and Maria Domenica Di Benedetto. Feedback stabilization of dynamical systems with switched delays. In *Proc. of the 51st IEEE Conference on Decision and Control*, pages 1325–1330, 2012.

[4] Robert Shorten, Fabian Wirth, and Douglas Leith. A positive systems model of TCP-like congestion control: asymptotic results. *IEEE/ACM Transactions on Networking*, 14(3):616–629, 2006.

[5] Esteban A Hernandez-Vargas, Richard H Middleton, and Patrizio Colaneri. Optimal and MPC switching strategies for mitigating viral mutation and escape. In *Proc. of the 18th IFAC World Congress Milano (Italy) August*, 2011.

[6] Daniel Liberzon and A Stephen Morse. Basic problems in stability and design of switched systems. *IEEE Control Systems Magazine*, 19(5):59–70, 1999.

[7] Raphaël Jungers. The joint spectral radius. *Lecture Notes in Control and Information Sciences*, 385, 2009.

[8] H. Lin and P. J. Antsaklis. Stability and stabilizability of switched linear systems: a survey of recent results. *IEEE Transaction on Automatic Control*, 54(2):308–322, 2009.

[9] Robert Shorten, Fabian Wirth, Oliver Mason, Kai Wulff, and Christopher King. Stability criteria for switched and hybrid systems. *SIAM review*, 49(4):545–592, 2007.

[10] Ji-Woong Lee and Geir E Dullerud. Uniform stabilization of discrete-time switched and markovian jump linear systems. *Automatica, 42(2), 205-218*, 2006.

[11] Ray Essick, Ji-Woong Lee, and Geir E Dullerud. Control of linear switched systems with receding horizon modal information. *IEEE Transactions on Automatic Control*, 59(9):2340–2352, 2014.

[12] Hai Lin and Panos J Antsaklis. Stability and stabilizability of switched linear systems: a survey of recent results. *IEEE Transactions on Automatic control*, 54(2):308–322, 2009.

[13] Atreyee Kundu and Debasish Chatterjee. Stabilizing switching signals for switched systems. *IEEE Transactions on Automatic Control,*, 60(3):882–888, 2015.

[14] Jamal Daafouz and Jacques Bernussou. Parameter dependent lyapunov functions for discrete time systems with time varying parametric uncertainties. *Systems & Control Letters*, 43(5):355–359, 2001.

[15] M. Fiacchini and G. Millérioux. Dead-beat stabilizability of discrete-time switched linear systems: algorithms and applications. *IEEE Trans. on Automatic Control*, 64:3839 – 3845, 2019.

[16] A. J. Menezes, P. C. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, October 1996.

[17] Moritz Fauser and Ping Zhang. Resilient homomorphic encryption scheme for cyber-physical systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 5634–5639, 2021.

[18] A. Abdelwahab, W. Lucia, and A. Youssef. Covert channels in cyber-physical systems. *IEEE Control Systems Letters*, 5(4):1273–1278, 2021.

[19] Helem S. Sánchez, Damiano Rotondo, Teresa Escobet, Vicenç Puig, and Joseba Quevedo. Bibliographical review on cyber attacks from a control oriented perspective. *Annual Reviews in Control*, 48:103–128, 2019.

[20] U. M. Maurer. New approaches to the design of self-synchronizing stream cipher. *Advance in Cryptography, In Proc. Eurocrypt '91, Lecture Notes in Computer Science*, pages 458–471, 1991.

[21] Oliver Jung and Christoph Ruland. Encryption with statistical self-synchronization in synchronous broadband networks. In Çetin K. Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems*, pages 340–352, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[22] C. Carlet. *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, chapter Vectorial Boolean Functions for Cryptography. Cambridge Press, 2010.

[23] M. Fiacchini and G. Millérioux. Dead-beat stabilizability of autonomous switched linear discrete-time systems. *IFAC-PapersOnLine*, 50(1):4576–4581, 2017.

[24] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error correcting codes*, volume 16. Elsevier, 1977.