Nonlinear Analysis: Hybrid Systems 30 (2018) 92–105

Contents lists available at ScienceDirect

Nonlinear Analysis: Hybrid Systems

journal homepage: www.elsevier.com/locate/nahs

Flatness and structural analysis as a constructive framework for private communication

B. Dravie [a,b], P. Guillot [c], G. Millérioux [a,b,*]

[a] Université de Lorraine, CRAN, UMR 7039, France
[b] CNRS, CRAN, UMR 7039, France
[c] Université Paris 8, LAGA, France

ARTICLE INFO

Article history:
Received 21 December 2016
Accepted 23 April 2018

Keywords:
Flatness
LPV and switched linear systems
Structural analysis
Ciphers

ABSTRACT

This paper deals with the use of control theoretical concepts, essentially flatness, along with structural analysis, in the context of private communication. A new and systematic methodology to design a cryptographic architecture belonging to the special class of ciphers called Self Synchronizing Stream Ciphers (SSSC) is proposed. Till now, the constructions of SSSC were based on automata with finite input memory involving shifts or triangular functions (T-functions) as state transition functions. Besides, only ad-hoc approaches were available in the literature. The contribution of this paper is to propose a general framework to design SSSC involving dynamical systems taking the form of switched linear systems over finite fields with arbitrary state transition functions. An example of cipher with complete specifications is given. Its implementation on a small scale breeding of an ICS-SCADA equipment is described.

1. Introduction

Synchronization has been an important topic in automatic control for years. Roughly speaking, by synchronization, it is meant correlated (according to given criteria) behaviors of at least two or more interconnected entities in virtual or physical networks. Throughout the past centuries, scientists have attempted to explain the emergence of order through the concept of synchronization. C. Huygens in 1665 can be considered as a pioneer. A lot of examples of synchronized phenomena (see [1]) are borrowed from nature, biology, neuroscience, physiology and more recently social networks.

Synchronization can be a very efficient way of tackling engineering issues as well. For example, there is a growing interest in cooperative control problems. Such problems involve several autonomous entities (also called agents) which try to collectively reach a global objective by a suitable connectivity or by adequate couplings. The related applications are mobile robots, unmanned and autonomous vehicles, satellites, air traffic control. Synchronization is also central in communication: video broadcasting, Phase Lock Loop-based equipment. When synchronization must occur in a peer-to-peer communication setup without any external control, it is called self-synchronization. It turns out that self-synchronization is central in cryptography, more specifically, in symmetric cryptography involving the so-called Self-Synchronizing Stream Ciphers (SSSC) (see [2]). Such ciphers are based on generators which can take the form of dynamical systems operating on finite fields and must deliver sequences of high complexity. Those sequences are used to scramble the information to be safely transmitted. For proper decryption, those sequences must be self-synchronized.

* Corresponding author at: Université de Lorraine, CNRS, CRAN UMR CNRS 7039, France.
E-mail addresses: brandon.dravie@univ-lorraine.fr (B. Dravie), philippe.guillot@univ-paris8.fr (P. Guillot), gilles.millerioux@univ-lorraine.fr (G. Millérioux).

https://doi.org/10.1016/j.nahs.2018.04.007
1751-570X/

Self-Synchronizing Stream Ciphers were patented in 1946. The self-synchronization property has many advantages and is especially relevant to group communications. Since 1960, specific SSSC have been designed and are still used to provide bulk encryption (for hertzian line, RNIS link, ...) in military applications or governmental radio mobile networks. In the early 90s, studies have been performed in [3,4] to propose secure designs of SSSC. These works have been followed by effective constructions [4–6], but till now, all of these SSSC have been broken, which motivates the search of new constructions of SSSC. It is this issue that is investigated in this paper.

More precisely, we aim at addressing the design of SSSC in terms of control theoretical concepts. The ciphers being viewed as dynamical systems, the main result is that the concepts of flatness together with structural analysis and graph-theory allow for a convenient and systematic way to construct classes of SSSC which are more general than the ones proposed so far. Indeed, it will be shown that the resulting ciphers can incorporate dynamical systems with state transition functions more general than the so-called T-functions which are known to suffer from weakness [7]. Hence, we can expect that this generalization yields more secure SSSC. The special class of flat Linear Parameter Varying (LPV) systems is motivated. Let us stress that since the dynamics operate on finite fields, all the variables, including the varying parameters, take value in a finite set of elements. Hence, the LPV system can be considered as a switched linear system with a number of modes depending on the cardinality of this set.

The paper is organized as follows. Section 2 is devoted to the problem statement. The architecture of Self-Synchronizing Stream Ciphers is presented in terms of dynamical systems. In Section 3, the special class of LPV dynamical systems is presented, the definition of difference flatness is introduced and is particularized to this class. An algebraic characterization in terms of state space matrix representation is provided. The connection between flatness and SSSC is made. It is shown that an automaton with a finite input memory, as required for an SSSC, can be systematically obtained from the left inverse of a flat system. In Section 4, a construction of a family of SSSC involving LPV automata is provided. It is based on structural analysis and graphs. A simple example of construction is given to make a clear understanding of the method. Then, the design of a realistic example and its implementation on a small scale breeding of an ICS-SCADA equipment are described.

2. Self-Synchronizing Stream Ciphers and dynamical systems

2.1. Generalities on stream ciphers

For a stream cipher, it must be given an alphabet A, that is, a finite set of basic elements named symbols. Hereafter, the index k will stand for the discrete-time. On the transmitter part, the plaintext (also called information or message) $m \in \mathcal{M}$ ($\mathcal{M}$ is the message space) is a string of plaintext symbols $m_k \in A$. Each plaintext symbol is encrypted, by means of an encryption (or ciphering) function e, according to

$$c_{k+b} = e(z_{k+b}, m_k), \tag{1}$$

where $z_k \in A$ is a so-called keystream (or running key) symbol delivered by a keystream generator. The function e is invertible for any prescribed z_k. The resulting quantity $c_k \in A$ is the ciphertext symbol. The integer $b \geq 0$ stands for a potential delay between the plaintext m_k and the corresponding ciphertext c_{k+b}. This is explained by computational or implementation reasons, for instance pipelining (see [8] for example). Consequently, for stream ciphers, the way how to encrypt each plaintext symbol changes on each iteration. The resulting ciphertext $c \in \mathcal{C}$ ($\mathcal{C}$ is called the ciphertext space), that is the string of symbols c_k, is conveyed to the receiver through a public channel.

At the receiver side, the ciphertext c_k is decrypted according to a decryption function d which depends on a running key $\widehat{z}_k \in A$ delivered, similarly to the cipher part, by a keystream generator. The decryption function d obeys the following rule. For any two keystream symbols $\widehat{z}_{k+b}, z_{k+b} \in A$, it holds that

$$\widehat{m}_{k+b} := d(c_{k+b}, \widehat{z}_{k+b}) = m_k \text{ whenever } \widehat{z}_{k+b} = z_{k+b}. \tag{2}$$

Eq. (2) means that the running keys z_k and $\widehat{z}_k$ must be synchronized for a proper decryption. The generators delivering the keystreams are parametrized by a secret key denoted in the sequel by $\theta \in \mathcal{K}$ ($\mathcal{K}$ is the secret key space). The distinct classes of stream ciphers (synchronous or self-synchronizing) differ each other by the way on how the keystreams are generated and synchronized. Next, we detail the special class of stream ciphers called Self-Synchronizing Stream Ciphers.

2.2. Keystream generators for Self-Synchronizing Stream Ciphers

A well-admitted approach to generate keystreams has been first suggested in [3]. It is based on the use of so-called state automata with finite input memory as described below. This is typically the case in the cipher Moustique [9]. At the ciphering side, the automaton delivering the keystream takes the following form:

$$\begin{cases} q_{k+1} = g_\theta(q_k, c_{k+b}), \\ z_{k+b} = h_\theta(q_k) \end{cases} \tag{3}$$

where $q_k \in A$ is the internal state, g is the next-state transition function parametrized by $\theta \in \mathcal{K}$. As previously stressed, the delay b is due to the fact that the output (also called filtering) function h is pipelined with an architecture involving b stages.