Information Processing Letters 116 (2016) 192–196

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

An improvement of a cryptanalysis algorithm

Oualid Benamara [a], Fatiha Merazka [b,*], Kamel Betina [a]

[a] LATN Lab. Institute of Mathematics, University of Science and Technology Houari Boumediene (usthb), P.O. Box 32, El Alia, Algiers, Algeria
[b] LISIC Lab. Telecommunications Department, University of Science and Technology Houari Boumediene (usthb), P.O. Box 32, El Alia, Algiers, Algeria

ARTICLE INFO

Article history:
Received 20 September 2014
Received in revised form 14 July 2015
Accepted 6 August 2015
Available online 12 August 2015
Communicated by S.M. Yiu

Keywords:
Cryptography
Markov Chain Monte Carlo
Classical cryptosystems
Pseudo random number generators

ABSTRACT

In this paper we present simulations that show how some pseudo random number generators can improve the effectiveness of a statistical cryptanalysis algorithm. We deduce mainly that a better generator enhances the accuracy of the cryptanalysis algorithm.

1. Introduction

Cryptography refers to the science that concerns encrypting data so that, without a secret key, a third party other than the sender and the receiver cannot recover the secret data [10]. At the same time, the cryptanalysts try to break the cryptosystems in order to prove that there is a security flaw, and then proceed to the correction of the cryptographic system. According to the Kerckhoffs principal, the algorithm should be known by all the parties and even the adversary who wants to break the code; the security should rely on the secret key, rather than the algorithm used [2]. We refer to [8] for the standard definitions.

Classical cryptosystems operate at the byte level of the data whereas the modern ones at the bit level. We will study the substitution–transposition ciphers in this work but the cryptanalysis techniques may be applied to the modern cryptosystems due to the fact that the operation mode is the same, yet the parameters of the two kinds of cryptosystems are not the same, like the alphabet space and the fact that the latter relies on more sophisticated operations.

In this paper we deal with classical cryptosystems and try to improve a cryptanalysis algorithm by testing different pseudo random number generators (PRNG). A pseudo random number generator is a key parameter in the cryptanalysis process. The goal of this study is to evaluate how different PRNGs impact the performance of the MCMC algorithm in terms of the metrics that will be defined in the later sections.

This paper is organized as follows: the next sections present Markov Chain Monte Carlo (MCMC) algorithm and a survey of the related theory. The outputs of the application of some PRNGs to MCMC are given thereafter, wherein simulation results are presented. We analyze those results and give our interpretation and analysis in Section 4.

* Corresponding author.
E-mail addresses: benamara.oualid@gmail.com (O. Benamara), fmerazka@usthb.dz (F. Merazka), kamelbetina@gmail.com (K. Betina).

http://dx.doi.org/10.1016/j.ipl.2015.08.002
0020-0190/

2. Markov Chain Monte Carlo

In this section, we give the definition of Markov Chain Monte Carlo (MCMC). A Markov Chain is a stochastic process $X_1, X_2, \ldots$ (the X_i's are random variables) together with the following property [9]:

$$P(X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1) = P(X_{n+1} = x_{n+1} | X_n = x_n)$$

The above property means that the present state at $n+1$ depends only on the previous state n and not on the other ones $(n-1), (n-2), \ldots, 2, 1$. A Markov Chain is said to be on a finite or countable states space, if the random variables take only finite or countable values. A Markov Chain is said to be irreducible if it is possible to go from any state to another state in the state space within a non-zero probability. A state i in the state space is said to be aperiodic if the return to the state i occurs at irregular times. The chain is then aperiodic if the whole set of the states is aperiodic. Markov Chain Monte Carlo (with its variants) is a set of techniques or algorithms intended to generate a Markov Chain which converges to a given probability distribution [11].

The below proposition will justify the MCMC procedure.

Proposition 1. *If a Markov chain X_n on a finite or countable state space X is irreducible and aperiodic, with stationary distribution π, then for every subset $A \subseteq X$,*

$$\lim_{n\to\infty} P(X_n \in A) = \int_A \pi(x)dx$$

The above proposition means that the values of X_n are distributed according to π for sufficiently large n.

2.1. MCMC algorithm

In this section we rewrite the MCMC algorithm as described in [4].

Let π be a probability distribution. The MCMC algorithm operates as follows:

- Choose an initial state $X_0 \in X$, where X is all the possible states that the Markov Chain may take. In probability theory we call X *the universe*.
- For $n = 1, 2, 3, \ldots$.
- Propose a new state $Y_n \in X$ from some symmetric proposal density $q(X_{n-1}, \ldots, X_0)$.
- Let U_n *Uniform*$[0, 1]$, independently of $X_0, \ldots, X_{n-1}, Y_n$.
- If $U_n < (\pi(Y_n)/\pi(X_{n-1}))$, then "accept" the proposal by setting $X_n = Y_n$, otherwise "reject" the proposal by setting $X_n = X_{n-1}$.

At the end of this process, we obtain a Markov Chain $(X_n)_{n\in N}$ which converges to π.

2.2. An MCMC algorithm to break a substitution–transposition cryptosystem

The substitution ciphers operate by replacing each letter in the clear text by another letter according to the key, which defines the correspondence. The one-time pad (OTP) cipher, which belongs to this class, is one of the rare cryptosystems whose security is proved. However, it is impractical, due to the fact that the secret key is as long as the clear text. The transposition cipher applies a permutation to the clear text, but leaves the correspondence between the letters unchanged. Combining the above two functions results in the substitution–transposition cipher [8].

In order to apply the above MCMC algorithm to the deciphering process, we proceed as described in [4]. At the implementation step, we will use the best parameters found in the above study, such as the scale parameters and the number of iterations, resulting in a more accurate algorithm:

1. Choose an initial state (the states here are all the possible encryption keys), and a fixed scaling parameter $p > 0$.
2. Repeat the following steps for many iterations (e.g. 10 000 iterations).
   - Given the current state x, propose a new state y from some symmetric density $q(x, y)$.
   - Sample U_n *Uniform*$[0, 1]$, independently from all other variables.
   - If $u < (\pi(y)/\pi(x))^p$, then "accept" the proposal by replacing x with y, otherwise reject y by leaving x unchanged.

$$\pi(x) = \prod_{\beta_1,\beta_2} r(\beta_1, \beta_2)^{f_x(\beta_1,\beta_2)}$$

where r are the frequencies of letters of the reference text and the f_x are those of the decrypted text using the key x.

2.3. Testing methodology

We have tested the MCMC algorithm with different pseudo random number generators (PRNGs). The results in terms of accuracy and number of successful decryption are given in the next paragraph. The definition of the metrics are given therein. Also we will provide the full algorithm of the PRNG used. The source code of the MCMC is available at [14]. Once the files downloaded and installed, download the following two files [15] and the following file [16]. Once done, change the corresponding files in the MCMC directory. After that, switch between the PRNGs by uncommenting and commenting out the functions at RandUtil::newRand() found in the file RandUtil.cpp. Once done, build the program again and run it.

Each simulation consists of the following:

- Generate a secret key.
- Encrypt the text using the above key.
- Run the MCMC algorithm over the encrypted text in order to recover the secret key.
- Once done, compute the average correctness and evaluate if the cryptanalysis was successful or not (if the key has been recovered), according to the definitions given in the next section.

The above steps are repeated 100 times. At the end, we output the average correctness and the number of success-