Deterministic Random Number Generator Algorithm for Cryptosystem Keys

Adi A. Maaita and Hamza A. A. Al_Sewadi

Abstract—One of the crucial parameters of digital cryptographic systems is the selection of the keys used and their distribution. The randomness of the keys has a strong impact on the system’s security strength being difficult to be predicted, guessed, reproduced, or discovered by a cryptanalyst. Therefore, adequate key randomness generation is still sought for the benefit of stronger cryptosystems. This paper suggests an algorithm designed to generate and test pseudo random number sequences intended for cryptographic applications. This algorithm is based on mathematically manipulating a publicly agreed upon information between sender and receiver over a public channel. This information is used as a seed for performing some mathematical functions in order to generate a sequence of pseudorandom numbers that will be used for encryption/decryption purposes. This manipulation involves permutations and substitutions that fulfill Shannon’s principle of “confusion and diffusion”. ASCII code characters were utilized in the generation process instead of using bit strings initially, which adds more flexibility in testing different seed values. Finally, the obtained results would indicate sound difficulty of guessing keys by attackers.

Keywords—Cryptosystems, Information Security agreement, Key distribution, Random numbers.

I. INTRODUCTION

INDEPENDENT unpredictably and uniformly distributed numbers that cannot be reliably reproduced are referred to as random numbers [1]. They play a major part in practical implementation and strength of most cryptographic systems. They may be used as keys for symmetric crypto-systems, public key parameters, session keys, etc. [2]. Failure of obtaining strong keys definitely will end up with data security compromise. Therefore, strong random number generators that exhibit high statistical quality and can withstand cryptanalysis efforts are keenly sought. Such strong random number generators constitute an important building block in the design and testing of high quality crypto-systems [3].

Generally random numbers can be truly random TRN, pseudo-random PRN or quasi-random QRN. Truly random numbers are unpredictable. Their generation stems from random physical or natural phenomena, such as radioactive decay, amplified noise generated by a resistor or a semiconductor diode, fed to a comparator or Schmitt trigger and then the output is sampled to get a series of bits which are statistically independent or random [4].

Pseudo-random numbers generators, also known as deterministic random bit generators, are computer programs that generate a sequence of numbers whose properties approximate the properties of sequences of random numbers [5], [6]. These sequences are not truly random as they are completely determined by a relatively small set of initial values called seeds; however, they are important in practice for their speed and reproducibility in number generation.

Quasi-random numbers are sequences in arbitrary dimensions which progressively cover a d-dimensional space with a set of points that are uniformly distributed. They are also known as low-discrepancy sequences [7]. The quasi-random sequence generators use an interface that is similar to the interface for random number generators, except that seeding is not required as each generator produces a single sequence.

Recently, a new type of generators which are called Lagged Fibonacci pseudo-random number generators [8], [9] have become increasingly popular generators for serial as well as scalable parallel machines. They are proved to be easy to implement, cheap to compute and they are performing reasonably well on standard statistical tests especially when the lag is sufficiently high.

After the brief definitions in section 1, related works will be summarized in section 2. Section 3 defines the important randomness tests that will be executed to examine the randomness of the generated keys. Section 4 explains the proposed Pseudo-random generator scheme; section 5 includes the implementation of the proposed scheme and the results of the randomness tests. Finally section 6 concludes the paper.

II. RELATED WORK

Random number generators may be classified into Integer generators, sequence generators, integer set generators, Gaussian generators, decimal fraction generators or row random byte generators depending if they generate integers, integer sequence, set of random integers, integers that fits normal distribution or numbers in the 0 and 1 range with configurable decimal places, respectively. Each of the mentioned types is useful for many cryptographic purposes [10]. Splitable pseudorandom number generators (PRNGs) were very useful for structuring purely functional programs that deal with randomness, because they allow different parts of the program to independently generate random values, thus...