PARALLEL PLATFORM FOR NEW SECURE STREAM CIPHERS BASED ON NP-HARD PROBLEMS

KHALED MOHAMMAD A. SUWAIS

UNIVERSITI SAINS MALAYSIA

2009

PARALLEL PLATFORM FOR NEW SECURE STREAM CIPHERS BASED ON NP-HARD PROBLEMS

by

KHALED MOHAMMAD A. SUWAIS

Thesis submitted in fulfillment of the requirements
for the degree of
Doctor of Philosophy

JUNE 2009

ACKNOWLEDGMENT

During my work on this thesis, there were many people who sincerely guided and supported me. I would like to take this opportunity to express my gratitude to all of them, for giving me the opportunity to complete this thesis.

I am deeply indebted to my supervisor, Associated Professor Azman Samsudin, for his guidance during my research at School of Computer Sciences, Universiti Sains Malaysia (USM). He was always accessible and enthusiastic in research. His enthusiasm motivated me, and made my research life became smooth and rewording.

I want to express my gratitude to School of Computer Sciences, Universiti Sains Malaysia and the general office staff, for supporting this study financially and morally.

I would like to take this opportunity to thank my beloved parents, Mohammad and Ilham. Their unlimited support and true love had maimed the geographical distances between us. They stand by me, raised me, supported me, taught me, and love me. To them I dedicate this thesis.

I would also like to thank my lovely brother Abdul Aziz and my beloved sisters Dina, Abeer and Haneen for their support and love during my study abroad.

I wish to thank the person who stand by me and shared my happiness and sadness. The person who supported me and provided me with a caring environment, and unforgettable moments in Malaysia. Thank you dear Beh Bee Cheng.

I would like to thank Khader Alrawajfih, Nagham Almadi and Mohammad Almadi, for being my second family in Malaysia. With them, I felt the support and shared happy and unforgettable moments during my study.

Last but not least, I would like to thank my colleagues, Ibrahim, Hisham, Ali, Fazila and Hamid, for their support and help. I also want to thank my dearest frinds Mohammad Wasef, Mohammad Bakkar, Wesam Hatamleh, Mohammad Alia, Muthanna Albuldawi, Bashar Alashal, Osama Abu-Dayyeh Sarah Shalini, Adnan Hunaif Ahmad Alzoubi and Mohammad Betar for their care and support.

Khaled M. Suwais

TABLE OF CONTENTS

CHAPTER TWO: STREAM CIPHERS AND PARALLELISM

CHAPTER THREE: NP-HARD BASED STREAM CIPHERS AND PARALLEL PLATFORM

CHAPTER FOUR: SECURITY AND PERFORMANCE ANALYSIS

APPENDIX B: ENCRYPTION RATES AND STATISTICAL RESULTS

APPENDIX C: GMP AND NTL LIBRARIES

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

NP-hard:	Non-deterministic Polynomial-hard
PRNG:	Pseudo-random Number Generator
NS:	Next-State
LFSR:	Linear Feed-back Shift Register
NLFSR:	Non-Linear Feed-back Shift Register
FCSR:	Feed Carry Shift Register
FPGA:	Field Programmable Gate Array
ANF:	Algebraic Normal Form
T-Function:	Triangular-Function
S-Box:	Substitution-Box
AES:	Advanced Encryption Standard
SPN:	Substitution-Permutation Network
SSL:	Secure Socket Layer
WEP:	Wired Equivalent Privacy
IV:	Initial Vector
SHA:	Secure Hash Algorithm
KSA:	Key-Scheduling Algorithm
PRGA:	Pseudo-random Generation Algorithm
NoC:	Network on Chip
DLP:	Discrete Logarithm Problem
ECDLP:	Elliptic Curve Discrete Logarithm Problem
ECC:	Elliptic Curve Cryptography
ECDSA:	Elliptic Curve Digital Signature Algorithm

ECMQV:	Elliptic Curve Menezes-Qu-Vanstone
SVP:	Shortest Vector Problem
CVP:	Closest Vector Problem
TSP:	Travelling Salesman Problem
VHDL:	VHSIC Hardware Description Language
PS-LFSR:	Parallelized-Structured Linear Feed-back Shift Register
ONB:	Optimal Normal Form
PB:	Polynomial Basis
IS:	Initialization Stage
KGS:	Keystream Generation Stage
ES:	Encryption Stage
NIST:	National Institute of Standard and Technology
GCD:	Greatest Common Divisor
LSB:	Least Significant Bit
MSB:	Most Significant Bit
GS:	Generator Selector
PEx:	Parameter Extractor
KG:	Keystream Generator
KC:	Keystream-size Controller
PEn:	Plaintext Encoder
MCD:	Machine-core Detector
TCC:	Thread Creation Controller
BsC:	Bit-sync Controller
NOB:	Number of Bits
NOC:	Number of Cores
DualC:	Dual-Core

QuadC:	Quad-Core
BDAY:	Birthday Spacing Test
OPERM:	Overlapping 5-Permutation Test
RANK:	Binary Rank Test for Matrices
BSTM:	Bitstream Test
DNA:	DNA Tests
OPSO:	OPSO Test
OQSO:	OQSO Test
CBYTE:	Count-The-1's Test on stream of bytes
CSBYTE:	Count-The-1's Test for specific bytes
PARKL:	Parking Lot Test
MDIST:	Minimum Distance Test
3DS:	3Dspheres Test
SQEZ:	Squeeze Test
OSUM:	Overlapping Sums Test
RUN:	Run Test
CRAP:	Crap Test
GMP:	GNU Multi-precision
NTL:	Number Theory Library

PELANTAR SELARI UNTUK SIFER ALIRAN SELAMAT YANG BARU BERASASKAN PERMASALAHAN *NP-HARD*

ABSTRAK

Tujuan kajian ini adalah untuk mengenal pasti unsur-unsur utama reka bentuk sifer aliran yang selamat dan pantas. Dalam bidang kriptografi, sifer aliran ialah algoritma kekunci simetri yang direka untuk menyulitkan dan menyahsulitkan data-data sulit. Penyulitan dan penyahsulitan sejumlah besar data-data ini memerlukan reka bentuk alternatif sifer aliran yang menjanjikan tahap keselamatan dan kepantasan yang lebih tinggi bagi penyulitan data.

Kedua-dua masalah *NP-hard* dan keselarian digunakan dalam kajian ini. Masalah-masalah *NP-hard* digunakan untuk memberi tahap keselamatan tinggi pada sifer aliran memandangkan ketiadaan algoritma untuk menyelesaikan masalah tersebut dalam sistem polinomial. Konsep keselarian diperkenalkan sebagai platform bagi sifer aliran berdasarkan masalah *NP-hard*. Platform selari ini direka bentuk bagi membolehkan sifer aliran ini berfungsi lebih pantas di atas pemproses berbilang teras (multicore processor).

Integrasi antara sifer aliran berdasarkan masalah *NP-hard* dengan platform selari merupakan faktor utama dalam penghasilan sifer aliran yang selamat dan pantas. Analisis keselamatan dan statistik menunjukkan bahawa sifer aliran berdasarkan masalah *NP-hard* kami adalah selamat daripada serangan-serangan *cryptanalysis* dan statistik. Analisis prestasi ke atas platform selari menunjukkan keputusan mengagumkan

bagi sifer aliran yang diuji, di mana platform selari tersebut dapat mempercepatkan kadar enkripsi lebih kurang 1.8 dan 3.75 kali, masing-masing pada pemproses dwi-teras dan empat-teras. Platform tersebut didapati cekap menggunakan pemproses berbilang teras di mana ia mampu meningkatkan prestasi sifer aliran yang diuji selaras dengan peningkatan bilangan teras dalam pemproses berbilang teras.

Kajian ini memperkenalkan reka bentuk praktikal (bagi sifer aliran dan platform selari) yang berskala menggunakan pemproses berbilang teras. Masa depan platform selari ini adalah cerah memandangkan bilangan teras dalam pemproses berbilang teras meningkat secara eksponen. Justeru, bilangan teras yang meningkat ini akan memacu prestasi sifer aliran berasaskan permasalahan *NP-hard* terpalam.

PARALLEL PLATFROM FOR NEW SECURE STREAM CIPHERS BASED ON NP-HARD PROBLEMS

ABSTRACT

The purpose of this study was to identify the key elements for secure and fast stream cipher's design. In cryptography, stream cipher is a symmetric key algorithm, which is designed to encrypt and decrypt stream of confidential data. Encrypting and decrypting massive amount of data, necessitates alternative design for stream cipher, which compromises high level of security and fast data encryption.

Both *NP*-hard problems and parallelism were utilized in this study. The *NP*-hard problems were used to provide stream ciphers with high level of security, since there is no algorithm exists to solve *NP*-hard problems in polynomial time. Parallelism was introduced as a platform for stream ciphers based on *NP*-hard problems. The parallel platform was designed to enable stream ciphers based on *NP*-hard problems to perform faster on multi-core processors.

The integration between the stream ciphers based on *NP*-hard problems and the parallel platform was the primary factor in producing secure and fast stream ciphers. The security and statistical analysis showed that our *NP*-hard problem-based stream ciphers are secure against cryptanalysis and statistical attacks. The performance analysis on the parallel platform revealed impressive results for the tested stream ciphers, where the parallel platform accelerated the encryption rate by approximately 1.8 and 3.75 times on Dual-core and Quad-core processors respectively. The platform was found efficient in

utilizing multi-core processors, in which it was able to speed up the performance of the tested stream ciphers relatively to the increasing number of cores, on multi-core processors.

This study has introduced a practical design (stream cipher and parallel platform), which was scalable in utilizing multi-core processors. The future of the parallel platform is promising, since the number of cores in multi-core processors is increasing exponentially. Thus, the increasing number of cores will effectively accelerate the performance of plugged-in *NP*-hard problem-based stream ciphers.

CHAPTER ONE
INTRODUCTION

1.0 Introduction to Cryptography

Cryptography is the fundamental component for any computer security application used to provide cryptographic services for secure communication over public and unsecured channels. Cryptography focuses on issues of securing messages so that only the relevant parties can read the message (Mollin, 2007). The main purpose of cryptography is to encode the data (plaintext) to unreadable form (ciphertext) and vice versa. Transforming a message to an incomprehensive form is accomplished by a process known as encryption. In contrast, transforming an encrypted message to its original form is accomplished by a process known as decryption.

The use of cryptography was important through the centuries, in which cryptographic applications were used for civilian usage (companies, individuals, etc), or even were used in military operations as in World War I (e.g. cipher wheels or marks on papers) and World War II (e.g. Purple machine and Enigma) (Kahn, 1967). Cryptography is generally designed to provide confidentiality, authentication, integrity and accessibility services (Menezes, et al., 1993; Mouratidis, et al., 2003). Confidentiality service is used to ensure that messages are accessible only to authorized recipients. Authentication is normally used to authenticate the identity of the connected parties. Preventing eavesdroppers from changing the content of the messages sent from source to destination is basically a service provided by the

integrity service. Lastly, accessibility is designed to only allow authorized parties to use the available information resources.

In modern times, cryptographic systems (cryptosystems) have been used extensively in our daily communications to provide us with high level of security. In practice, cryptography is applied in numerous applications such as: internet communication, wireless communication (mobile phones) and banking transactions. The development of the cryptographic tools and systems has played an important role in re-shaping the communication style in a significant manner.

Based on the cryptography taxonomy found in (Kahn, 1967; Feistel, 1970; Beutelspacher, 1996), cryptography and cryptanalysis are combined together to form the science of Cryptology. The sciences of Cryptology and its cryptographic primitives are categorized in Figure 1.1.

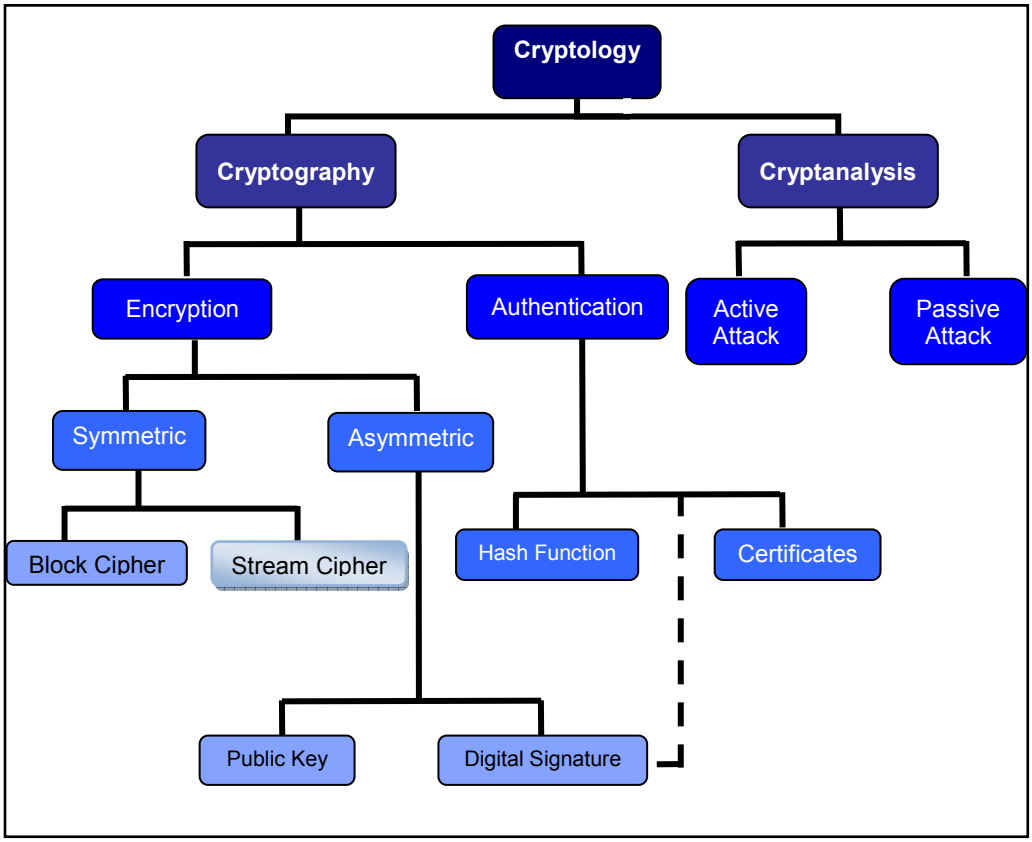

Figure 1.1: Taxonomy of Cryptology

Cryptography, as shown in Figure 1.1, deals with two categories of cryptosystems. The first category includes encryption algorithms which are classified into two further sub-categories: symmetric and asymmetric cryptosystems. The other category includes authentication algorithms which are classified into two main categories: hash functions and certificates authentication. In this research, we are focusing only on the stream cipher which falls under the encryption category of cryptography.

In principle, symmetric and asymmetric cryptosystems share the same design goal, in which both cryptosystems are responsible for encrypting and decrypting user's messages. The main difference between these cryptosystems is the use of additional key in asymmetric cryptosystems. Symmetric key cryptosystems use a single key to encrypt and decrypt a given message. On the contrary, asymmetric key cryptosystems (also known as public-key cryptosystems), use two different keys to encrypt and decrypt a message.

In stream ciphers, a sequence of random bits is generated and used as a keystream which will never be used again during the run of the cipher. In general, stream ciphers are faster than block ciphers, making them suitable to be used in a resource-constrained environment (Karlof, et al., 2004). The basic idea of stream ciphers is inherited from the theoretical encryption algorithm called a One-Time Pad, in which a plaintext is XORed with a key of the same length as the plaintext (Daswani, 2007). The general structure of stream ciphers is shown in Figure 1.2. Some of the classical ciphers are: Affine cipher, Vigenère Autokey Polyalphabetic cipher and Playfair cipher (Bishop, 2003; Stallings, 2006).

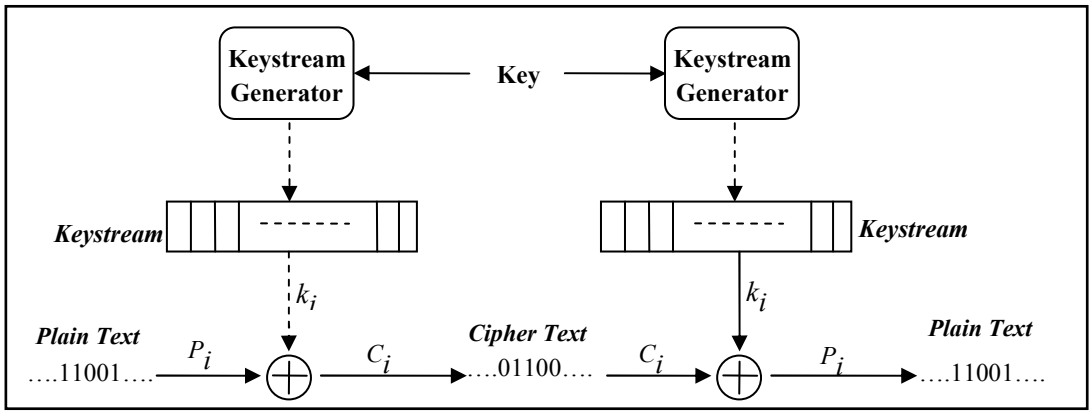

Figure 1.2: Stream cipher

The essential motivation of using stream ciphers, instead of block ciphers, is the ability of stream ciphers to perform faster in environment where computational power and memory capacity are limited (Karlof, et al., 2004). However, current technologies tend to overcome the limitation problems (such as performance and computational power) by presenting new multi-core processors which can improve the performance of the running systems.

1.1 NP-Hard Problems and Parallelism

NP-hard problems are complex mathematical problems with no algorithm to solve them in polynomial time is exist. NP-hard problems are well known in the field of cryptography since they proved to provide cryptosystems with high security. The use of NP-hard problems was efficient in different symmetric key cryptosystems, key exchange protocols, digital signature algorithms, and many others. However, the problem associated with NP-hard problems is the intensive number of calculations required during the run of the algorithm. Therefore, parallelism is introduced to fill

up the gap between the intensive calculations of NP-hard problems and their performance by utilizing the existence of multi-core processors.

High performance computing is increasingly in demand especially in several scientific and commercial applications. The need for secure and high performance communication has become an important ingredient in our daily life. Achieving higher performance is feasible by utilizing more computer resources, where several jobs are accomplished simultaneously and therefore completed in shorter time. Parallel processing is described as the concurrent use of multiple processing resources (e.g. multiple CPUs) to solve a computational problem, where a given problem is broken into smaller segments and solved concurrently using multiple processors. Technically, the achievement of higher performance depends on two essential factors: faster hardware devices and processing techniques (Briggs, 1986). Several researches were conducted to improve the hardware architecture and software techniques in order to achieve higher performance.

Under the processing techniques factor, multithreading technique was introduced in 1960s (SunSoft, 2002) to enhance the performance of applications by running them in parallel on the available resources. Multithreading technique aims to create a virtual multiprocessors environment and use this environment to run multiple tasks on single processor. From other perspective, the recent hardware revolution has played a significant role in improving systems performance through the multi-core technology. Multi-core is a technology where a single physical processor contains the logical core of more than one processor as shown in Figure 1.3. The main goal of this technology is to enable the multi-core processor to run

multiple tasks concurrently in order to achieve higher performance compared to single-core processors.

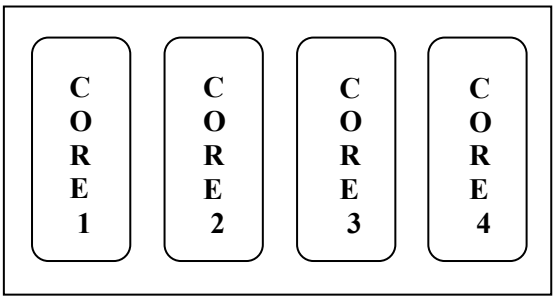

Figure 1.3: Quad-core processor

Applying parallel techniques in cryptography has become essential for higher throughput and improved performance, especially with the current available resources. Therefore, in this study we are utilizing the new multi-core technology with the multithreading techniques to speed up the encryption process in stream ciphers, in order to provide secured and better performance cryptosystems.

1.2 Research Problem

This study is conducted due to the importance of stream ciphers in securing information, which is considered as the most strategic resources. The state-of-art of stream ciphers is found lack of a comprehensive literature review that analyzes the constructional design of existing stream ciphers. Therefore, the exploration and analysis of existing stream ciphers resulted in addressing two problems, which are: **the security and the design properties**.

The massive numbers of existing stream ciphers are found vulnerable (security-breakable) to cryptanalysis attacks due to the weaknesses of their keystream generator. From the other perspective, the constructional design of these stream ciphers is found sequential. Therefore, these stream ciphers do not allow parallelism and not able to take full advantage of parallel computer architecture which is ready and available in markets.

As a summary, identifying the problems associated with existing stream ciphers is crucial to determine the efficient solutions. However, solving the security issue require applying complex and hard mathematical problems (i.e. NP-hard problems) at the keystream generator level to increase its resistance against cryptanalysis attacks. The use of NP-hard problems in stream ciphers is promising since there is no algorithm to solve these problems in polynomial time. On the other hand, re-constructing the design of stream ciphers is possible by identifying the independent components of their internal design. By doing that, stream ciphers will be able to utilize multi-core processors to enhance their performance, and then deliver the security community with highly secure and fast stream ciphers.

1.3 Research Scope

Stream ciphers are found lack of several security and statistical attributes, making them subjected to cryptanalysis attacks. The scope of this thesis is presented in introducing alternative stream ciphers based on the following *NP*-hard problems:

- Elliptic Curve Discrete Logarithm Problem (ECDLP).
- Discrete Logarithm Problem (DLP).

- Shortest Vector Problem in Lattice (SVP).

This research aims to use *NP*-hard problems in stream cipher cryptosystems, with the focus on the three *NP*-hard problems mentioned above. Security and performance analysis and assessment on the proposed stream ciphers and the parallel platform are also carried out to ensure their practicality.

The limitation of this research is associated with the choice of the highly studied *NP*-hard problems (ECDLP, DLP and SVP) for the proposed stream ciphers. In fact, there are several other *NP*-hard problems which are not well studied and therefore such problems need to be studied first before they can be used in cryptography.

1.4 Research Goal and Objectives

The main goal of the research is to overcome the security and design problems (as addressed in Section 1.2), by utilizing both *NP*-hard problems and parallelism on multi-core processors. The general objectives of this research are listed as follows:

- Propose alternative methods of generating sequences of keystream in stream ciphers based on:
    - Elliptic curve discrete logarithm problem represented by point multiplication over selected elliptic curve.
    - Discrete logarithm problem represented by polynomial representations using polynomial arithmetic.

- Shortest vector problem in lattice spaces represented by Low-Hamming weigh polynomials.

- Propose new parallel platform for stream ciphers based on *NP*-hard problems with proper design specifications.

- Perform intensive security and performance for the proposed stream ciphers and the parallel platform.

1.5 Research Contributions

This thesis provides a comprehensive review of stream ciphers which assist the development process of new stream ciphers. This thesis also present a secure and fast stream ciphers that has great impact on the field of information security. The main contributions of this research are summarized as follows:

1. Developing new highly secure stream ciphers based on NP-hard problems for cryptographic applications.

2. Presenting new research direction for parallel stream ciphers through the utilization of multi-core technologies.

3. Presenting new constructional design for stream ciphers capable to work on parallel environments.

4. Designing new parallel platform to accelerate the performance of secure stream ciphers.

1.6 Research Methodology

Stream ciphers are classified into two main categories: software and hardware based stream ciphers. These stream ciphers rely on several software and hardware techniques to achieve better security. In our research, we found that many of the subcategories, under the main two categories, suffer from some security vulnerabilities (such as cryptanalysis attacks and statistical biased), and sequential structure. Therefore, this study aims to use *NP*-hard problems (new subcategory under the software-based category) to achieve optimum level of security.

Achieving the research objectives is accomplished in two phases. The two phases are as follows:

- Stream Cipher Implementation Phase: the design of the proposed stream ciphers is divided into three stages: Initialization stage, Keystream Generation stage and Encryption stage. Dividing the structure of the stream ciphers into stages has enabled stream cipher to work in parallel, therefore transforming the sequential structure of existing stream cipher into new parallelizable structure. The employment of *NP*-hard problems will be targeted to the second stage of the design. Accordingly, generating infinite new keystream depends on infinite number of iterations during the run of the stream cipher. The iteration rests on a counter created for incremental operation on the initial value of the input key obtained from the initial stage. Finally, the obtained keystream is used to encrypt a stream of plaintext bits and generate a stream of enciphered bits. The general design of the proposed stream ciphers is shown in Figure 1.4.

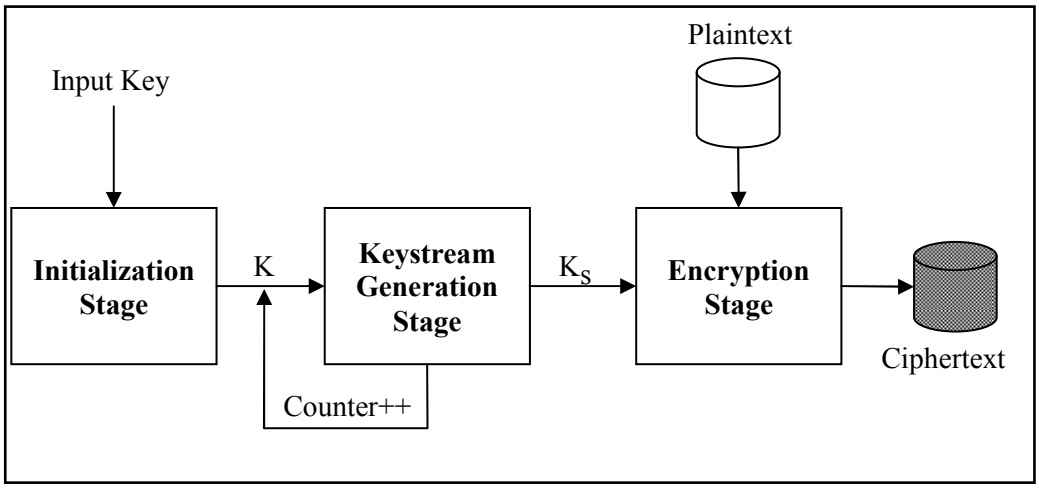

Figure 1.4: Proposed stream cipher design

- Platform Implementation Phase: the platform consists of several components and special designed detector and controllers. It is divided into five main components: Generator Selector, Parameter Extractor, Keystream Generator, Keystream-size Controller and Plaintext Encoder. These components are designed to work best with the stream ciphers obtained from the previous stage. The main goal of this platform is to enable stream ciphers generating multiple keystream concurrently and independently from each other.

Evaluating the two phases (stream ciphers implementation phase and platform implementation phase) is an important step in our research. The first phase is evaluated by verifying the reliability of our stream ciphers security against cryptanalysis attacks, and examining the statistical attributes of the generated keystream. On the other hand, the evaluation of the platform includes extensive performance tests on different multi-core processors. In addition, the platform is also verified statistically; by re-verifying the statistical properties of the generated keystream by the attached stream ciphers to the parallel platform. Figure 1.5 shows the workflow of the main components carried out in this research.

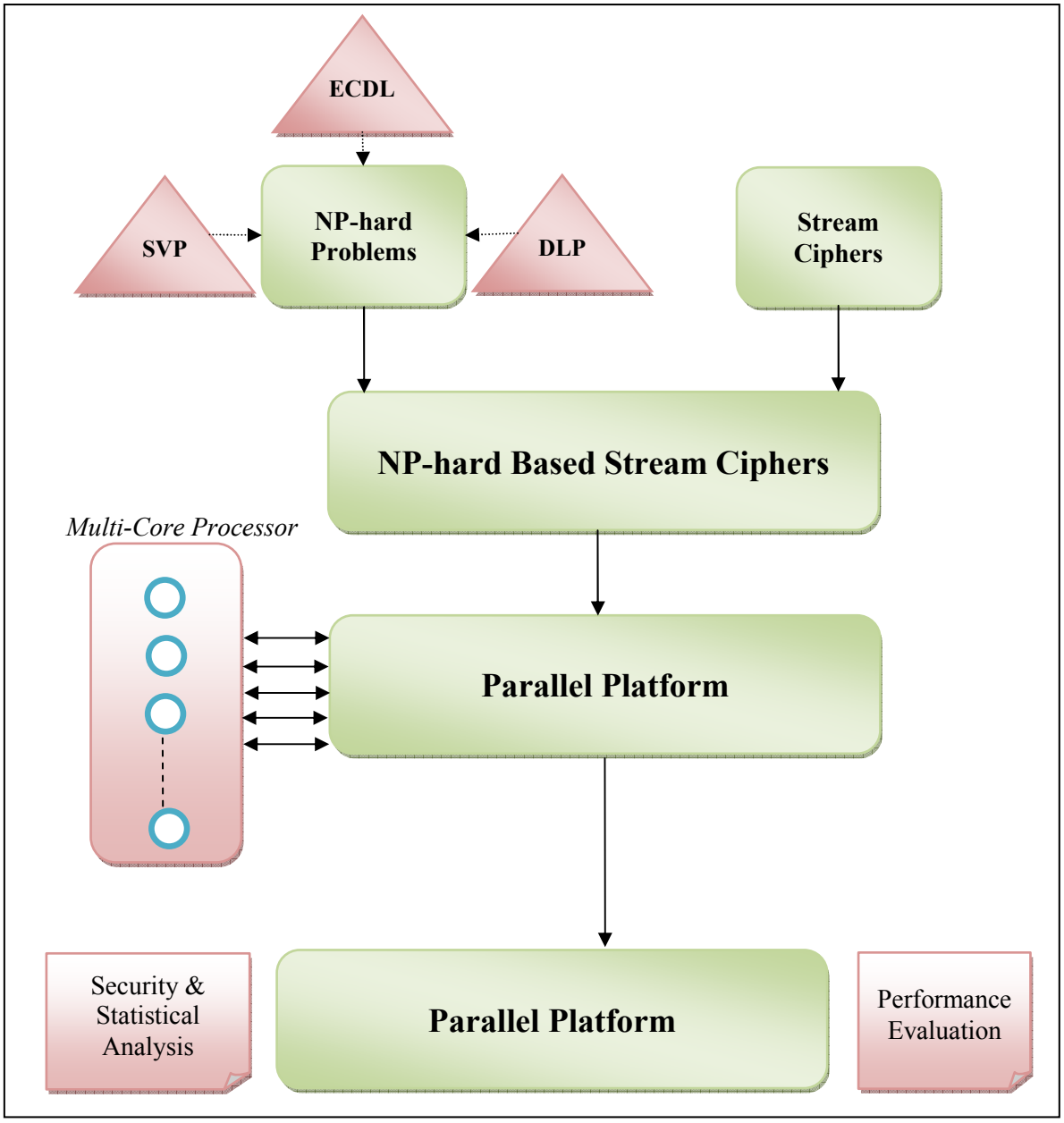

Figure 1.5: Research workflow

1.7 Thesis Organization

The work conducted in this thesis is presented in five chapters with appendices. The first chapter provides an introduction to the work by introducing a brief introduction to cryptography and its related concepts. In addition, a brief introduction to our work is also conducted here (Chapter 1). The reminder of the thesis is organized as follows.

Chapter 2: In this chapter, we discuss the important concepts of stream ciphers and their classifications. The use of the mathematical *NP*-hard problems and parallelism in cryptographic systems is also discussed.

Chapter 3: A complete description of the proposed methodology is included in this chapter. The chapter is divided into two sections. In Section 3.1, we discuss the stream ciphers design and specifications. The design and the implementation of the parallel platform components are described in Section 3.2.

Chapter 4: This chapter presents the security and performance analysis of both stream ciphers and parallel platform. The security analysis is conducted from two perspectives: cryptanalysis attacks and statistical analysis, to show the security level provided in the proposed ciphers. The performance evaluation aims to show the efficiency of the platform in terms of the ability of delivering stream ciphers with higher performance.

Chapter 5: Finally, we conclude our work and illustrate the possible future works for this study.

CHAPTER TWO

STREAM CIPHERS AND PARALLELISM

2.0 Introduction

Cryptographic systems are designed to provide digital transmitted data with security attributes over un-secured communication channels. Figure 2.1 sheds light on the main components of a conventional cryptosystem model to understand the environment that cryptographers are dealing with.

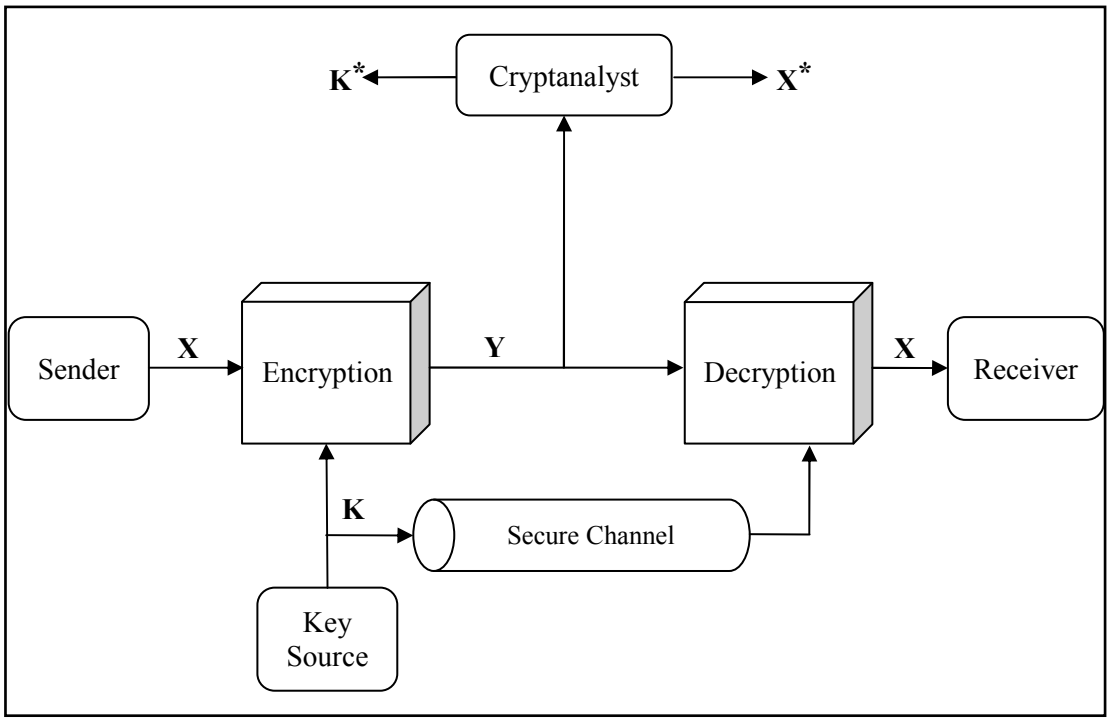

Figure 2.1: Model of conventional cryptosystem (Stalling, 2003)

This chapter will study the encryption component of the cryptosystem model as appears in Figure 2.1. The study will focus on one type of symmetric key algorithms called stream ciphers. The rest of this chapter will discuss the important

concepts of stream ciphers in terms of their design and mechanisms (Section 2.1). In Section 2.2, we survey the existing stream ciphers and classify them into categories based on the similarity between these stream ciphers. Stream ciphers are found somehow lack some of security attributes, making them subject to cryptanalysis attacks.

It is obvious that sending the message **X** from the sender to the receiver is not simple due to the obstacles that might threaten the security (e.g. integrity, authenticity, confidentiality, etc) of the message. However, cryptosystems rely on some cryptographic primitives (e.g. Encryption, Digital Signatures, Certificates, etc) to provide the message with the required level of security. Cryptanalyst will sometimes be able to attack the encrypted message **Y** to remove the disguise and recover the message **X** using several cryptanalysis attacks. These attacks are basically generating a message estimate $\mathbf{X}^*$ and a key estimate $\mathbf{K}^*$ to recover either message **X** or key **K**, or even both.

The definition of the mathematical concepts related to *NP*-hard problems, as well as survey the existing cryptosystems based on *NP*-hard problems are discussed in Section 2.3. The use of complex mathematical problems is considered costly in terms of computation. The use of such problems will require the keystream generator to perform intensive calculations to generate the sequence of keystreams. One way to overcome this limitation is by applying parallelism to improve the performance of stream ciphers based on *NP*-hard problems. Hence, parallel techniques and their usage in cryptographic applications are discussed in Section 2.4 and 2.5 respectively. Finally, section 2.6 will summarize the important results obtained from the above analyzed survey.

2.1 Stream Ciphers: Concept and Definition

Cryptographic systems are divided into two types of systems: Secret-key (Symmetric) or Public-key (Asymmetric) cryptosystems. In the later systems, the sender uses public information of the receiver to send a message to the receiver. The receiver then uses his private information to recover the original message. In secret-key cryptosystems, both the sender and receiver have previously set up secret information in which they use this information for encryption and decryption. Further division of symmetric cryptosystems is the block ciphers and stream ciphers. Rueppel states: *"*Block ciphers operate with a fixed transformation on large blocks of plaintext data; stream ciphers operate with a time-varying transformation on individual plaintext digit*"* as quoted in (Robshaw, 1995).

Definition (Encryption): Let $k_1, k_2, ..., k_n \in K$ be a set of keystream in the key space K, $m_1, m_2, ..., m_n \in M$ be a set of plaintext in the plaintext space M, and $c_1, c_2, ..., c_n \in C$ be a set of ciphertext in the ciphertext space C. The encrypted ciphertext is generated by:

$$E_{k_i}(m_i) = c_1, c_2, ..., c_n \in C \qquad \forall i : 1 \le i \le n \tag{2.1}$$

From the above definition, the encryption process of a stream cipher E_k is bijective for every k_i. The plaintext space and key space are typically represented in bit or byte representations. Most importantly, keeping the key k is essential for the security of stream ciphers.

The idea of stream ciphers was inspired from the famous cipher called the One-time Pad (also called the Vernam cipher) (Mollin, 2007; Delfs, 2002). This

cipher is based on XOR'ing ($\oplus$) the message bits and the key bits. The One-time pad is defined by Delfs (2000) as in Equation 2.2:

$$\boldsymbol{E} : \{0,1\} \times \{0,1\} \rightarrow \{0,1\},\ (\boldsymbol{m}, \boldsymbol{k}) \rightarrow \boldsymbol{m} \oplus \boldsymbol{k} \tag{2.2}$$

where plaintext, keystream and ciphertext bits are in the space $\{0, 1\}$. The encryption transformation is given by:

$$\boldsymbol{E}_{k_i}\left(\boldsymbol{m}_i\right) = \boldsymbol{m}_i \oplus \boldsymbol{k}_i = \boldsymbol{c}_i \in \boldsymbol{C} \tag{2.3}$$

and the decryption transformation is given by:

$$\boldsymbol{D}_{k_i}\left(\boldsymbol{c}_i\right) = \boldsymbol{c}_i \oplus \boldsymbol{k}_i = \boldsymbol{m}_i \in \boldsymbol{M} \tag{2.4}$$

In One-time pad cipher, a truly random key is generated and securely transmitted to the receiver. The length of the key is the same as the length of the transmitted message. On the other hand, most stream ciphers other than One-time pad depend on the use of pseudorandom number generator (PRNG) instead of generating truly random key. Pseudorandom number generator generates key sequence (known also as keystream) that approximate the random numbers properties (Viega, 2003).

Generally, stream cipher uses *n*-iterations to generate *n*-successive keystream based on the stream cipher internal state. The mechanisms used in updating the internal state provide us with the classification of stream ciphers: Synchronous and Asynchronous stream ciphers. The two types of the stream ciphers are discussed in Sections 2.1.1 and 2.1.2 respectively.

2.1.1 Synchronous Stream Ciphers

A stream cipher is said to be synchronous if the next internal state used for generating new keystream is defined independently and without the use of either the plaintext or the ciphertext generated from the previous iteration as shown in Figure 2.2.

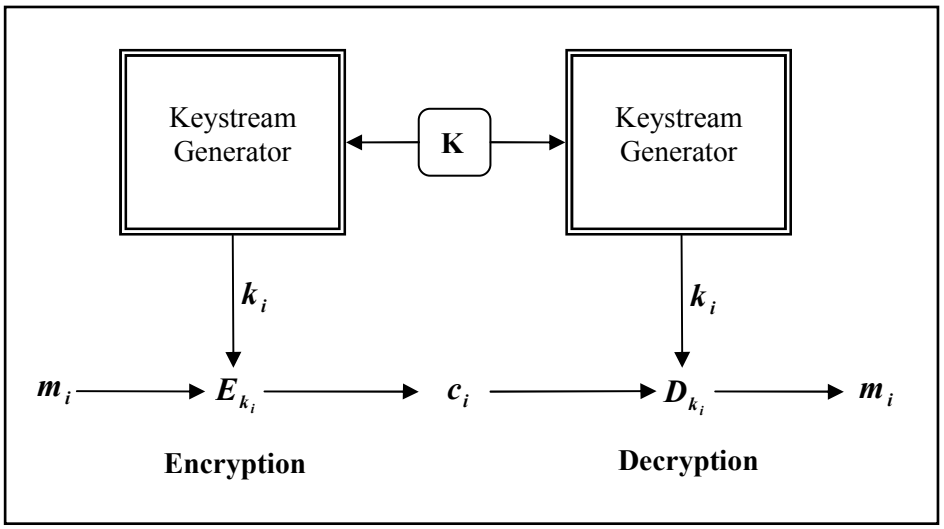

Figure 2.2: Synchronous stream cipher (Mollin, 2007)

The encryption process of a synchronous stream cipher is best described by the following equations:

$$\partial_{i+1} = \boldsymbol{NS}\left(\partial_i, \boldsymbol{K}\right), \tag{2.5}$$

$$\boldsymbol{k}_{s_i} = \boldsymbol{KG}\left(\partial_i, \boldsymbol{K}\right), \tag{2.6}$$

$$\boldsymbol{c}_i = \boldsymbol{EN}\left(\boldsymbol{k}_{s_i}, \boldsymbol{m}_i\right) \tag{2.7}$$

where ∂_0 is the initial state determined by the input key $\boldsymbol{K}$, $\boldsymbol{NS}$ is the next-state function, $\boldsymbol{KG}$ is the keystream generation function which generates $\boldsymbol{k}_s$, and $\boldsymbol{EN}$ is the

output function which generates the ciphered text c_i by combining both the plaintext m_i and the keystream k_{s_i}.

The properties of synchronous stream ciphers are summarized in the following points:

- Synchronization requirements: both of the sender and receiver must use the same secret key and operating at the same position within that secret key for proper decryption. This mechanism will ensure the synchronization between the connected parties during the transmission process. The loss of synchronization due to some modification on the ciphertext being transmitted will fail the decryption process. In such case, a re-synchronization process such as re-initialization or placing special marks at regular intervals in the ciphertext is required.
- Error propagation: In synchronous stream ciphers, errors in ciphertext symbols (digits or characters) during transmission do not propagate and affect the decryption of other ciphertext symbols.
- Active attack: Any modification (insertion, deletion) or replay of ciphertext symbols by an eavesdropper causes loss of synchronization, which allows the receiver (decrypter) to detect the modification.

2.1.2 Asynchronous Stream Ciphers (self-synchronous)

Unlike synchronous stream ciphers, self-synchronous stream ciphers generate keystreams as a function of the key k and a fixed number of previous ciphertext

symbols (Stallings, 2006). In other words, self-synchronous ciphers utilize plaintext in the process of generating a new keystream as shown in Figure 2.3.

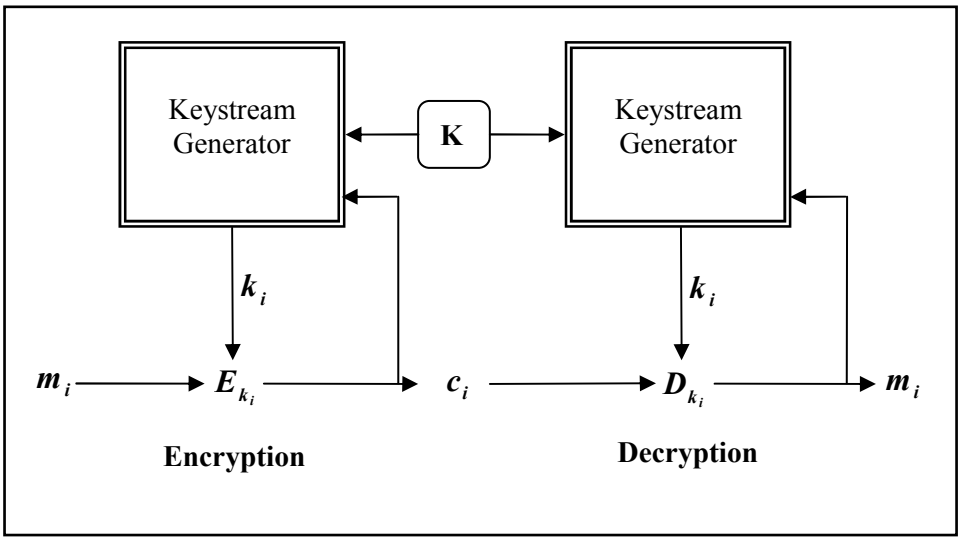

Figure 2.3: Asynchronous stream cipher

The encryption function of a self-synchronous stream cipher is best described by the following equations:

$$\partial_i = (c_{i-t}, c_{i-t+1}, ..., c_{i-1}), \tag{2.8}$$

$$k_{s_i} = KG(\partial_i, K), \tag{2.9}$$

$$c_i = EN(k_{s_i}, m_i) \tag{2.10}$$

where ∂_0 is the initial state, K is the input key, KG is the keystream generation function which produces k_s, and EN is the encryption function which combine both of the keystream and plaintext m_i to produce the ciphertext c_i.

In this type of ciphers and unlike synchronous ciphers, the selection of the key may raise problem since any statistical regularities in the plaintext will show up

in the key (Bishop, 2003). Self-synchronous stream ciphers have the following properties (Mollin, 2005):

- Synchronization requirements: the decryption process in self-synchronous stream ciphers basically depends on a fixed number *N* of preceding ciphertext digits. Hence if modifications have been made on ciphertext digits during transmission, self-synchronization will take place.
- Error propagation: self-synchronous ciphers are subjected to error propagation due to the dependency of the keystream generation's internal state on the previous ciphertext digits of length *N*. When modifications on ciphertext occur during transmission, decryption of up to *r* subsequent ciphertext symbols may be incorrect, after which correct decryption resumes.
- Active attacks: any modification of ciphertext digits by an eavesdropper causes multiple ciphertext symbols to be encrypted incorrectly. According to Stallings (2003), detecting the modification in self-synchronous stream ciphers is more difficult than in synchronous stream ciphers.

During the process of designing a stream cipher, the cryptographer has to satisfy the following requirements (Lee, et al., 2002):

- Error Propagation: minimum number of redundant bits during the encryption and decryption process.
- Redundant Information: minimum number of redundant bits during the information insertion.
- Cryptographic Security: the length of the secret key should be long enough to avoid exhaustive key search attack.

- Implementation Simplicity: the design of the stream cipher should be suitable for both hardware and software implementations.
- Performance Speed: the stream cipher should be performable at minimum speed of 1.544 Mbps.

2.2 Stream Ciphers: Categorization and Related Works

In contrast to block ciphers, stream ciphers have no standard model for their construction design, which leads cryptographers to construct various models of stream ciphers. The basic structures often found in stream ciphers may include LFSR-based cipher (Linear-Feedback Shift Register), NLFSR-based cipher (Non-Linear Feedback Shift Registers), Block cipher-based stream cipher, T-function-based ciphers and other structures. However, this study classifies stream ciphers into categories whereby each category includes stream ciphers that share specific properties. Figure 2.4 shows the classification of different stream cipher algorithms as appears in the literature. Note that the classification is based on how keystream generators operate in generating the keystream.

In our classification, stream ciphers are divided into two main categories: Hardware-based and Software-based stream ciphers. The classification aims to look at stream ciphers from the implementation perspectives in which we found that they fall into two categories (hardware-implementation and software-implementation stream ciphers). The third level on the hierarchy of the classification includes deeper categorization based on hardware or software related issues. The in-depth classification of hardware-based stream ciphers includes: FCSR/NLFSR-based and

clock control and LFSR based stream ciphers. On the other hand, the software-based stream cipher category includes: T-Function-based stream ciphers, Block cipher-based stream ciphers, *NP*-hard problem-based stream ciphers (our proposed direction of stream ciphers) and other alternatives that do not fit into the previous categories.

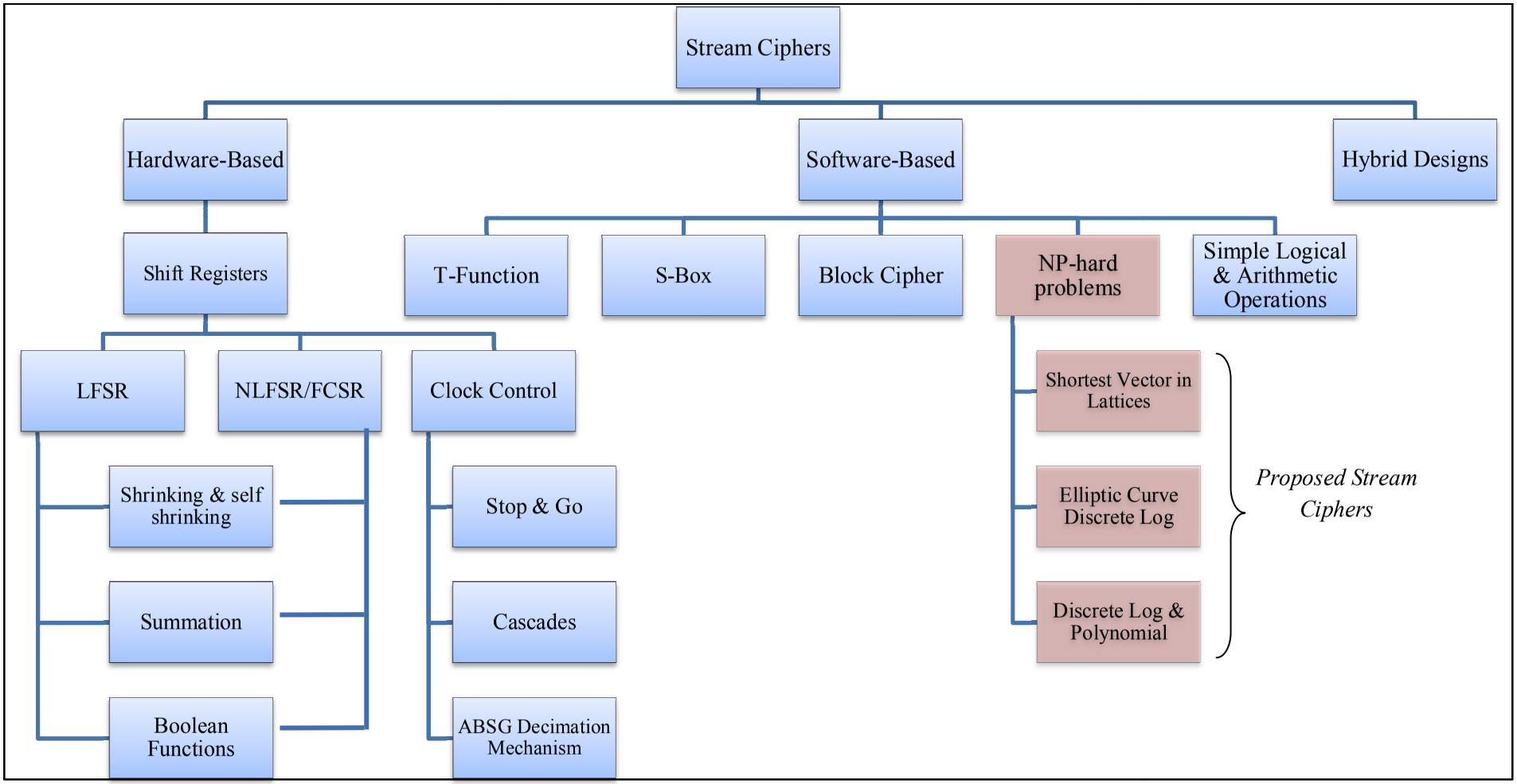

Figure 2.4: Stream ciphers classifications