A COMPARATIVE ANALYSIS OF DIFFERENT LFSR BASED CIPHERS AND PARALLEL COMPUTING PLATFORMS FOR DEVELOPMENT OF GENERIC CIPHER COMPATIBLE ON BOTH HARDWARE AND SOFTWARE PLATFORMS

Trishla Shah [*], Darshana Upadhyay [1], Priyanka Sharma [2]

[*]Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad 382481, India
Student and Corresponding author, email: 13mcei15@nirmauni.ac.in
[1]Assistant Professor, Institute of Technology, Nirma University, Ahmedabad 382481, India email:
darshana.upadhyay@nirmauni.ac.in
[2]Associate Professor, Institute of Technology, Nirma University, Ahmedabad 382481, India, email: priyanka.sharma@nirmauni.ac.in

ABSTRACT

Pseudo-Random numbers are at the core of any network security application. They find their application in the network security domain in key-generation, re-keying, authentication, smart-phone security etc. These random numbers are produced through PRNG (Pseudo Random Number Generator). Hence, if the PRNG produces predictable sets of random numbers, then the entire application would be prone to attacks. Therefore, development of a generic framework for generating strong sets of pseudo-random numbers is proposed. Hardware implementation for GSM stream cipher is already available under a particular segment of mobile communication. The project advanced into many dimensions like, vulnerability assessment, protocol design, implementation in both software and hardware and evaluation. The proposal aims to build an in-general framework and a unified model for enhanced security specifically for LFSR (Linear Feedback Shift Register) based stream ciphers. Hence, a thorough study on already existing LFSR based ciphers is done which aims to extract out the behaviour of different ciphers under different application domains. As pseudo-random numbers are used in both software (stream ciphers, protocol design) and hardware (wireless devices, smart phones) areas of security, the generic model proposed is aimed at using a co-simulation of both. For software development of the cipher, a parallel computing environment has been chosen because in today's computing trends, multi-core processors are superseding the sequential ones, hence the primary engine for processor performance growth is to increase parallelism rather than increasing the clock rate. The paper thus presents the CSPRNG (Cryptographically Secure Pseudo Random Number Generator) model based on hardware and software co-simulation, using a generic approach.

Key words: CSPRNG, GSM, Attacks, Keys, Generic, Co-simulation, LFSR

INTRODUCTION

In today's era, the use of networks and its applications are growing rapidly. Users often reveal critical information like account numbers, bank passwords, personal and financial details, important transaction details etc., over the Internet. Apart from its legitimate use, attacks like password theft, virus attacks, spoofing, message confidentiality threats, message integrity threats etc., have been found, causing potential loss of the users' private information. Hence it is important to build a secure system providing a perfect balance of confidentiality, integrity and availability of users' private data. These security parameters are provided by a mechanism of key generation (public and private keys), random password generation, one-time password (OTP) generation, strong authentication etc. Implementation of these mechanisms is done through generation of unpredictable sets of random numbers having high uncertainty, called pseudo-random numbers. Hence, pseudo-random numbers are at the core in providing security to network applications. These random numbers are produced through a Pseudo-Random Number Generator. Hence, if the PRNG (Pseudo-Random Number Generator) produces predictable sets of random numbers, then the entire application would be prone to attacks. Therefore, development of a generic framework for generating strong sets of pseudo-random numbers, using a co-simulation of hardware and software is proposed. The proposal aims to build an in-general framework and a unified model for enhanced security specifically for LFSR (Linear Feed-Back Shift Register). Here, the design of the model has been constrained specifically for enhanced security of LFSR based stream ciphers, owing to its good statistical properties, large period, well suited to low power or high speed requirements. For the software implementation, a parallel computing platform i.e. GPU programming is chosen, for increasing throughput. Therefore the entire model aims to develop a CSPRNG (Cryptographically Secure PRNG), using hardware and software co-simulation, for its use in various security applications. The research is thus constrained to network security domain.

Basic Concept

LFSR based stream ciphers are currently used in almost all network security applications (e.g., military cryptography, etc.) Recent research shows that these are prone to various threats like eavesdropping, snooping, masquerading and in the specific wireless network domain poor security mechanisms are explored.[1] Stream ciphers currently, are implemented on both hardware (A5/1, A5/2, KASUMA, E0, MICKEY, GRAIN, SNOW, FISH) and software (HC- 256, Rabbit, Salsa20, SOSEMANUK) [2] platforms. These ciphers have been detected to be prone to various network attacks like dynamic cube attack, basic correlation attack, refinement attack, guess-and-determine attack, linear approximation attack, algebraic attack, Berlekamp-Massey attack, fast time memory trade-off attack (which requires some pre computation) [3]. Hence, designing a strong LFSR based PRNG, resistant to above mentioned stream cipher attacks, is needed.

Challenges

Many stream ciphers have been designed for the generation of a strong set of pseudo random numbers but certain limitations are observed like: i) While designing hardware ciphers, the computational complexity over software performance decreases ii) Very few ciphers have been designed, working for network security applications in both hardware and software domains. iii) The software implementation is mostly done sequentially increasing time complexity overhead. iv) Ciphers compatible for generating good pseudo random series on a generic platform for diverse applications has not yet been designed.

LITERATURE SURVEY

The design features of this CSPRNG are done, considering its compatibility with both hardware and software. Hence, the entire literature survey is divided into analysis of Network Applications requiring PRNs, analysing hardware and software ciphers and analysing parallel computing platforms.

Analysis of Network Applications requiring pseudo-random numbers:

i) Application in generation of keys and in re-keying:

As per Tara Chand Singhal [4], key-distribution and re-keying are major problems in any research, and in wireless environment, these problems increase due to lesser sources of infrastructure, power and memory cost. The stream ciphers are used in secure communication in WEP (Wired Equivalent Privacy) and military applications. Hence, in all these applications, generation of pseudo random numbers is important for maintaining privacy and security. Hence, a strong cipher needs to be designed, which provide a highly random and attack resistant encrypted text and is to be designed at SSL (Secure Socket Layer). Re-keying is used, if in the same communication band (which may be long enough), different keys are used for security purpose.

ii) Mobile Devices for Mobile-Agent Communication:

With mobile agents, like mobile devices, major attacks occur during the process of communication and migration from one cell to another. Agent state, which is gained at previous executions, needs to be encrypted, so that an intruder cannot change or take advantage of it. This requires the generation of secret key, which is a strong pseudo random number. [5]

iii) Application on Smart Phones:

In any security applications, ubiquitous computing devices, not having necessary computing capacities are hard to operate. Smart cards have PRN (Pseudo-Random Numbers) for security. Usually PRN are produced by physical random number generator, but these are vulnerable to environmental changes. Hence, for securing against attacks, generation of the PRN is required. [6]

iv) Authentication to counter DOS (Denial of Service) Attack on 802.11:

WLANs (Wireless LAN) which are based on 802.11 standards, are vulnerable to DOS attacks due to unprotected authentication management and control frames. They can be filtered by pseudo random number generator authentication. Here a mechanism for authentication is provided for security. A strong and highly unpredictable random sequence is required. Hence, a PRNG based on software mechanism is required. [7]

Analysis of Hardware Cipher

For understanding the design specification of hardware, following hardware ciphers have been studied: i) GRAIN -128 [8], ii) GRAIN-128a [9], iii) SNOW 2.0 (both h/w and s/w) [10], iv) SNOW 2.0 modified [11], v) RFID (AES) [12].The table presents a detailed study of all hardware ciphers, useful in cipher designing.

Table 2 Analysis of hardware ciphers.

Parameters	RFID	GRAIN-128	GRAIN-128a
Purpose	Providing security using strong symmetric authentication, using low-power and low die-size.	Providing security in all hardware applications with low memory and low power, using lesser components	Enhanced from GRAIN-128, supporting improved authentication and hardware performance.
Security Issues to be overcome by the ciphers	Consumer tracking, tag forgery and the unauthorized access to the tag's memory content.	Correlation Attack, Chosen IV attack, Time Memory Trade-off attack	All Attacks observed by Grain-128
Input to Ciphers	Blocks(128 bits)	Bit-oriented	Bit-oriented
Reason for input	AES provides better security	Bit-oriented, as it is easy to implement in hardware	Bit-oriented, as it is easy to implement in hardware
Functions Applied to input text(bytes)	Functions Applied to input text(bytes)	LFSR, NFSR, filter function	LFSR, NFSR, pre-output function
Key-Size	128-bits	80 bits	128-bits
Reason for key-size selection	-	To prevent all attacks with computational complexity lower than 2^{80}	-
S-box or NFSR use(if yes), then reason for selection	The more S-boxes are used the less clock cycles are needed for encryption.	Generation of non-linearity	Both shift registers are regularly clocked so the cipher will output one bit every second clock. This regular clocking is an advantage, both in terms of performance and resistance to side-channel attacks,

			compared to using irregular clocking or Decimation
Reason for selection of algorithms on hardware implementation	**AES-**Main aim of using AES in RFID is using min hardware (constraint is size) and min power consumption. Hence, an 8-bit architecture instead of 32-bit, reduces number of S-boxes and reduce in power consumption.	**GRAIN-** Main aim is to avoid attacks with computational complexity not more than 2^80 and min hardware. Hence a memory of 160-bits is chosen and functions are chosen appropriately minimize hardware.	**GRAIN-128a-** Main aim is to provide in-built support for authentication and improve hardware performance against older version of GRAIN. The authentication depends on the security of pre-output stream (to provide more randomness).
Throughput	**Gate Equivalent-** 3595 **Clock Rate-** 992	**Gate Equivalent-** 2243 **Clock Rate-**256	**Gate Equivalent-** 2133 **Clock Rate-**160
Improvements from previous ciphers	Previous AES implementations never focused on AES module low die-size and low power-consumption requirements. This implementation focused only on low hardware complexity and low power consumption, providing authentication	The AES implementation on RFID used more number of gates, thus increasing hardware complexity. GRAIN 128 is specifically tailored for using low hardware complexity and security against attacks.	GRAIN 128 didn't have authentication so, GRAIN 128a provided authentication support, and high security by its highly random pre-output generator.

The hardware, hence to be used in cipher, is concentrated on its clock cycles, feasibility in applications, its orientation in bits or words etc.

Analysis of Software ciphers

The software ciphers implemented till date were designed specifically for sequential generation. Hence, these ciphers are studied to understand their sequential computation and replace it with parallel computing and to check the feasibility of these ciphers for parallel computation.

Table 3 Analysis of software cipher.

Cipher	Usage	Implemented Approach
SNOW 1.0	Development of a more secure and fast cipher	A.1) Outputs from two components LFSR and FSM is independent of each other yet it is sequential. A.2) A technique called hard-coding is used, to increase the speed of computation but memory used is high A.3) XORing outputs of LFSR and FSM is done sequentially.
SNOW 2.0	Improvements over previous version	1)Mathematical equations derived for SNOW 2.0 are as follows: (x) = x16 + x14+1x5 + 1 F232 [x],4 = 233 + 2452+ 48 + 239 MUL[c] = (c23,c245, c48, c239) MUL1[c] = (c16, c39, c6, c64). All above equations are solved sequentially using gcc or Microsoft C++ Compiler.
RC4 Cipher	Used for checking out effect of adversaries on embedded devices	The implementation of RC4 is on CPU with the verification process being sequential, leading to overheads.

The analysis of software ciphers depicts the need to use parallel rather than sequential computing in their implementation approach. The advantages of using CUDA, is done in latter part of paper.

EXPERIMENTAL BASIS

For software implementation, it is necessary to choose a robust platform equalizing the trade-off between time and speed which is satisfied by using a GPU rather than CPU. Hence, parallel computing is beneficial rather than sequential. While surveying on parallel platforms, two most prominent candidates are: i) Nvidia's GPU and ii) Intel's GPU. The API used for Nvidia is CUDA and for Intel is OpenCL. A thorough analysis of these frameworks has been done. The analysis branches up in following segments:

Analysis of Parallel Computing Platform

The software implementation is to be done on parallel computing platforms. Here parallel platform is chosen rather than normal sequential computing to increase efficiency and decrease time. A parallel computing environment has been chosen because in today's computer trends, multi-core processors are superseding the sequential ones, hence the primary engine for processor performance growth is to increase parallelism rather than increasing the clock rate. Hence, increased parallelism would increase the efficiency of random number generation. Many parallel programming platforms are available like CUDA (Compute Unified Device

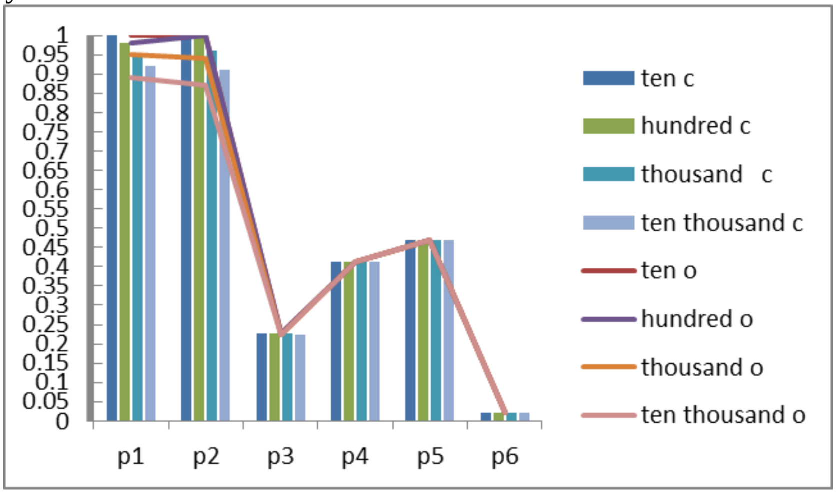
Architecture), OpenCL etc. are available. The survey analysis to find a better platform is done. The following section shows the performance metrics of CUDA over OpenCL in terms of throughput, timings, overheads etc.

Survey of default Pseudo Random Generating Libraries:

Both the platforms have in-built pseudo-random number generating libraries. The CUDA library for pseudo random number generation is CURAND and OpenCL, has PRNGCL for pseudo random number generation. The common basic algorithm which is used in both for pseudo random number generation is MTGP (Mersenne Twister). A thorough analysis of MTGP algorithm is done and is based on LFSR. This made the direction of survey much clear and precise.

Analysis of randomness of generated algorithms through NIST statistical toolkit:

The random numbers generated from the two platforms were tested on NIST statistical toolkit. Randomness was checked based on 14 parameters: [14]. The below table indicates a comparison between CUDA and OpenCL based on various tests. The tests were carried out for various bit streams including 10,100,1000,10,000. The X-axis indicates various tests and Y-axis indicates its comparison results. The results showed CUDA's MTGP to have a better randomness rather than OpenCL. Hence it was concluded that CUDAs pseudo random platform is more effective in PRNG generation rather than OpenCL. The brief analysis is shown in table 4:



P1	P2	P3	P4	P5	P6
App Entropy Test	Block Frequency Test	Cumulative Test Forward	Cumulative Test Reverse	FFT	Fq

Fig.1 NIST analysis of CUDA and OpenCL

Table 4 Effectiveness of CUDA for designing existing ciphers

Implemented approach using sequential computing	Suggested approach using CUDA	How is CUDA better

1) In many ciphers, Output from two components, LFSR and FSM is independent of each other, yet it is done sequential. 2) A technique called Hard-coding is used, to increase the speed of computation, but memory used is high. 3) XORing of outputs of LFSR and FSM, is done sequentially.	1) Using CUDA, generation of outputs from LFSR and FSM can be done in parallel. 2)Hardcoding LFSR is done sequentially; this can be done in parallel. 3) XORing of outputs of LFSR and FSM can be done in parallel.	Generation of parallel outputs would save time and increase efficiency.
Mathematical equations derived for SNOW 2.0 are as follows: (x) = x16 + x14 +1x5 + 1 F232 [x],4 = 233 + 2452 + 48 + 239 MUL[c] = (c23, c245, c48, c239) MUL1 [c] = (c16,c39, c6, c64). All above equations are solved sequentially using gcc or Microsoft C++ Compiler.	These equations can be solved in parallel like splitting entire equation as x16, x14, 233 etc , with one thread solving one term. All these can then be added in parallel.	Computational complexity of Matrix multiplication for these equations would decrease exponentially to the base 2.
The implementation of RC4 is on CPU with the verification process being sequential, leading to overheads	The same approach can be done in parallel, leading to low overheads of cycles.	With CUDA the entire algorithm can be optimized.

OBSERVATIONS

On the basis of above survey, following conclusions have been made for the proposed cipher. i) A hybrid of word oriented and bit oriented cipher is to be implemented for designing LFSR. This would best optimize the initial cycles as well as increase efficiency in software based ciphers. ii) A cipher is to be designed keeping in mind its basic utility i.e. security over communication with multiple messages using a single common key, and in telecommunication scenario for recovery from frame loss of sync messages. To design the above features, MODES can be designed in the cipher. iii) To increase efficiency, the component structure needs to work independently i.e. their o/p must be independent of each other and only the final output must be XORed. This can be best fitted in CUDA.

DISCUSSION

A primary objective of this paper is to design, implement and evaluate the cryptographically secure PRNG on parallel computing platforms. Towards the realization of this objective, the short term goals of this proposal are to:
i) Investigate vulnerabilities and security mechanisms in LFSR based stream ciphers. ii) Design wireless interface and techniques for stream ciphers vulnerability modelling and evaluate security requirements for each component network; iii) Design the proposed algorithm for PRNG using a hybrid of various methods (shrinking generator, nonlinear filter generator and alternating step generator) to break the predictability of LFSRs. iv) Comparative analysis of different parallel computing environment, namely OpenCL and CUDA. v) Analyse and design proposed algorithm using VHDL (very high speed integrated circuit hardware description language) on hardware platform FPGA-SPARTAN 6 and using CUDA on software parallel platform. vi) Identifying the hardware utilization using Spartan-6, FPGA, measurement of execution speed using parallel Computing software - CUDA, evaluate randomness of key stream using the NIST statistical test package.

TECHNICAL REQUIREMENTS AND FEASIBILITY

As the project is focused on both hardware and software implementation, it confines its technical requirements in both these domains. **Hardware Requirements:** VHDL- Very High Speed Integrated Circuit Hardware Description Language Analysis and designing of the proposed algorithm is done using VHDL language. FPGA-SPARTAN 6- The simulation of the proposed algorithm is to be done, using FPGA-Spartan 6 toolkit. **Software Requirements:** CUDA- Compute Unified Device Architecture is a parallel computing platform to parallelize the given algorithm, developed by NVIDIA. The GPU used is GeForce 480.

CONCLUSION AND FUTURE WORK

Through this paper, a precise review on different network applications, hardware and software ciphers, parallel computing platforms have been done. The study thus enforces the need to build a generic cipher which works efficiently both on hardware and software platforms. From the literature and the experimental basis, designing of the strong cipher is quite clear and easy. An n-bit LFSR cipher, customized for different application and different requirements of computation capacities is proposed. Figure 2 shows the block diagram of proposed cipher.

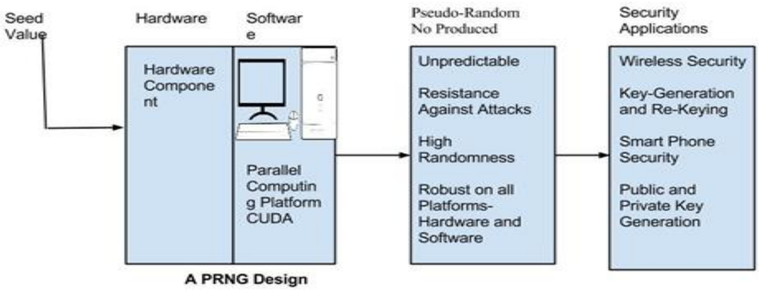


Fig.2 Block Diagram of PRNG Generation

ACKNOWLEDGMENT

Towards the progress of the project, I would firstly like my organization INSTITUTE OF TECHNOLOGY, NIRMA UNIVERSITY for granting of ample resources and a strong platform. I would like to thank all the associated professors and dignitaries.

REFERENCES

1. M. Garland, S. Le Grand, J. Nickolls, J. Anderson, J. Hardwick,S. Morton, E. Phillips, Y. Zhang, and V. Volkov, “Parallel Computing Experiences with CUDA,” IEEE Micro, vol. 28, pp. 13–27, July 2008.
2. Wasim A Al-Hamdani and Ivory J Griskell. A proposed curriculum of cryptography courses. In Proceedings of the 2nd annual conference on Information security curriculum development, pages 4-11. ACM, 2005.
3. Bart Preneel, Christof Paar, and Jan Pelzl. Understanding cryptography: a textbook for students and practitioners. Springer, 2009.
4. Tara Chand Singhal. Systems and methods for complex encryption keys, January 29 2013. US Patent 8,363,834.
5. U. Topaloglu, C. Bayrak, and K. Iqbal. A pseudo random number generator in mobile agent interactions. In Engineering of Intelligent Systems, 2006 IEEE International Conference on, pages 1-5, 2006.
6. Jian-Wei Fan, Chao-Wen Chan, and Ya-Fen Chang. A random increasing sequence hash chain and smart card-based remote user authentication scheme. In Information, Communications and Signal Processing (ICICS) 2013 9th International Conference on, pages 1-5. IEEE, 2013.
7. Mansoor Ahmed Khan and Aamir Hasan. Pseudo random number based authentication to counter denial of service attacks on 802.11. In Wireless and OpticalCommunications Networks, 2008. WOCN'08. 5th IFIP International Conference on, pages 1-5. IEEE, 2008.
8. Hell, Martin, Thomas Johansson, and Willi Meier. "Grain: a stream cipher for constrained environments." International Journal of Wireless and Mobile Computing 2.1 (2007): 86-93.
9. Martin_Agren, Martin Hell, Thomas Johansson, and Willi Meier. A new version of grain-128 with authentication. In Symmetric Key Encryption Workshop 2011
10. Patrik Ekdahl and Thomas Johansson. Snow-a new stream cipher. In Proceedings of First Open NESSIE Workshop, KU-Leuven, pages 167-168, 2000.
11. Patrik Ekdahl and Thomas Johansson. A new version of the stream cipher snow. In Selected Areas in Cryptography, pages 47-61. Springer, 2003.
12. Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong authentication for rfid systems using the aes algorithm. In Cryptographic Hardware and Embedded Systems-CHES 2004, pages 357-370. Springer, 2004.
13. Jianbin Fang, Ana Lucia Varbanescu, and Henk Sips. A comprehensive performance comparison of cuda and OpenCL. In Parallel Processing (ICPP), 2011 International Conference on, pages 216-225. IEEE, 2011.
14. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications.