Design and Simulation of Enhancing RC4 Stream Cipher for Wi-Fi Security using Verilog HDL

Rajendar Racherla¹, S Nagakishore Bhavanam²

¹ Guru Nanak Institute of Technology, E.C.E Dept., Hyderabad, India

² Guru Nanak Institute of Technology, Asst .Prof, E.C.E Dept., Hyderabad, India

ABSTRACT

IEEE 802.11 is a wireless LAN technology is based on a cellular architecture where the system is sub divided into cells, Each cell is called Basic Service Set (BSS) is controlled by a base station is called Accesses Point (AP). Access points are connected through Distribution System (DS), typically Ethernet or wireless itself. The important part of the wireless MAC layer design is the security (WEP/RC4 algorithm) implementation. Security protocol is the part of the data link layer.

The main features of cryptography are to work out with problems, which are associated with secrecy, authentication and integrity. In order to handle all the cryptographic problems many kinds of cryptographic algorithms have been invented. The complexity of these problems made several categories of cryptographic algorithms. A much known is the RC4 stream cipher.

The RC4 encryption process is divided into three stages: session key negotiation stage, seed key generation stage and data processing stage.

Firstly, RC4 uses the original session key 16 byte random number KS0 generated by A, and then it's encrypted by Elgamal. When B receives the encrypted information, decrypts with its own private key, B will get the shared key, and then return A an OK message.

Secondly, A uses the RC4 algorithm and KS0 to get a pseudo-random output stream.

Thirdly, A uses a certain combination of Hash functions to get the seed key, and then we finish the seed key generation stage.

Finally, we get a new pseudo-random data stream with the seed key generated by RC4 algorithm, discard the first 48 bytes of the data stream, and then encrypt the P. A. Send the encrypted result cipher text C(RC4(key, P)) to B. B decrypts the received data in the same way and gets the message data Plain Text.

Keywords: Encryption, Decryption, RC4, WEP, Verilog HDL, Xilinx, Modelsim.

I. INTRODUCTION

Wireless Local Area Network (WLAN) is the network that utilizes radio frequency technology instead of traditional coaxial. WLAN is widely used in many conditions, especially when it's difficult to install traditional network. As the openness and sharing of wireless channel nature, the security of wireless data stream becomes particularly prominent [1].

IEEE802.11 standard for WLAN defines two types of authentication open system authentication and shared key authentication, and uses RC4 stream encryption algorithm of the Wired Equivalent Protection (WEP) protocol to enhance its security. However, the facts show that the WEP protocol has not met the desired level of safety. On the contrary, WEP itself also has fatal security flaws, tampering with the data for a variety of active attacks and passive eavesdropping on the data provided to facilitate aggression.

WEP uses the Initial Vector (IV) to avoid duplication of key stream. Beginning in 2001, several serious weaknesses were reported and they demonstrate that WEP protocol is vulnerable in a number of areas. In essence, the problem is not in RC4 itself but in the way to generate the key and in how to use the key for RC4 encryption. Many hackers and computer security experts have discovered the WEP design flaws, which indicate that IEEE802.11 standards can only provide limited support to confidentiality. WEP provides a 40-bit key, which may be sufficient to keep away a common hacker but incapable to ward off a professional hacker. Either a 40-bit key or a 128-bit key can be easily cracked within two or three hours. RC4 is probably the most widely used stream cipher nowadays due to its simplicity and high efficiency. This paper focuses on the research to enhance RC4 algorithm. The rest of the paper is organized as follows. RC4 algorithm is introduced in Section 2. In Section 3, we present the RC4 encryption and decryption. The wifi encryption and Decryption is presented in Section 4. In Section 5, we provide simulation results. Section 6 provides the Synthesis report. Section 7 concludes this paper.

II. RC4 ALGORITHM

The principle of RC4 algorithm consists of two components: key-scheduling algorithm (KSA) and pseudo random number generation algorithm (PRGA). The key function of KSA is to complete initialization of RC4 Key, while the key function of PRGA is to produce pseudo-random number. The pseudo code for RC4 algorithm (KSA and PRGA) is shown below.

KSA

begin for i=0 to 255 Si=i; Ki=K[i mod n]; end for k=0; for i=0 to 255 $j=(j+Si+Ki) \mod 256$ swap(Si,Sj) end for end PRGA begin i=0; j=0; while(true) i=(i+1) mod 256 $j=(j+Si) \mod 256;$ swap(Si,Sj); $t=(Si+Sj) \mod 256$ K=St; end loop end

Stream ciphers and block ciphers are two classes of encryption algorithms. Stream ciphers encrypt a one-bit plaintext at a time, using a time-dependent encryption transformation. Block ciphers encrypt groups of plaintext characters using a fixed encryption transformation. Stream ciphers and block ciphers have their respective characteristics, but stream ciphers are almost always faster and use far less code than block ciphers do.

RC4 is a variable key-size stream cipher based on a 256byte secret internal state and two one-byte indexes. The data is encrypted by XORing data with the key stream which is generated by RC4 from a base key. For a given base key, KSA generates an initial permutation state denoted by S0.

PRGA is a repeated loop procedure and each loop generates a one-byte pseudo-random output as the stream key. At each loop, a one-byte stream key is generated and it is XORed with one-byte of the plaintext, in the meantime a new 256-byte permutation state S as well as two one-byte indices I and j are updated, which defined by (Sk+1, ik+1, jk+1) = PRGA(Sk, ik, jk) where ik+1 and jk+1 are the indices and Sk+1 is the state updated from ik, jk, and Sk by applying one loop of PRGA.

III. RC4 ENCRYPTION AND DECRYPTION

The encryption process of WEP is shown in Figure 1, WEP uses 40-bit or 104-bit encryption key connected with 24-bit IV to generate 64-bit or 128-bit seed key, and then send the seed key to a random generator PRNG, encrypt the plaintext with pseudo-random sequence [2]. System uses CRC32 (32-bit cyclic checksum) for

Vol. 1, Issue 3, pp.653-659

integrity verifying to ensure that the message will not be modified during transmission that sends IV, plaintext and integrity check value (ICV) to the other [3].

The decryption process of WEP is shown in Figure 1. The decryption key sequence is generated in the same way that generates encryption key, XORed with cipher text to get the plaintext. Compare ICV with integrity check value ICV ' calculated by CRC32, if the encryption key is the same as decryption key, and ICV '= ICV, then the receiver gets the original plaintext data. Many encryption algorithms are widely available in wired networks. They can be categorized into a symmetric key encryption.

In symmetric key encryption and secret key encryption, only one key is used to encrypt and decrypt data and the key should be distributed before transmission between entities. It is also very efficient since the key size can be small, while the functions used for encryption are hardware operations, and the encryption time can be very short. However, in large communication networks, key distribution can be a significant problem.

Asymmetric key encryption or public key encryption is used to solve the keydistribution problem. This uses two keys, one for encryption and another for decryption, and there is no need for distributing them prior to transmission. Public key encryption is based on mathematical functions, computationally intensive and not very efficient for small wireless devices. Generally, most encryptions used in wireless devices are based on symmetric key encryption, such as RC4.

RC4 is a stream cipher designed by Ron Rivest in 1987 and it is widely used in many applications today and in wireless networks such as IEEE 802.11 WEP and CDPD. With a unique key, a stream of pseudo-random numbers is generated, and then the encryption of data XORs the pseudo-random numbers from the stream with the data. RC4 is known to be fast and efficient, for it can be written using only a few lines of codes and requires only 256 bytes of random access memory (RAM). Hence, it is one of the best encryption schemes during the past decade.

RC4 is standardized to provide security services in WLAN using the WEP protocol. However, Fluhrer and many researchers have discovered several vulnerabilities in the RC4 algorithm. The weaknesses in RC4 and loopholes in the WEP protocol have resulted in a new standard for security in WLAN (IEEE 802.11i) proposing a new protocol based on the advanced encryption standard (AES).

AES is a block cipher designed by Joan Daemen and Vincent Rijmen that has a variable key length of 128, 192, or 256 bits to encrypt data blocks of 128, 192, or 256 bits long. Both block and key length are extensible to multiples of 32 bits. AES encryption is fast and flexible, and it can be implemented on various platforms especially in small devices and smart cards. Also, AES

Vol. 1, Issue 3, pp.653-659



Fig.1. WEP Encryption and Decryption

Improvement:

Modern cryptographic technique is divided into two types, symmetric encryption system and public key encryption system. Symmetric encryption system communicating parts need a safe way to ensure key sharing; public key encryption system communicating parts have their own pair of keys. In general, data processing efficiency of public key encryption system is not as high as symmetric encryption system, but the key is easier to manage. Therefore, we use public key encryption system for both parts to consult and then consult the key, use symmetric cryptography for data encryption and decryption. This maximizes the advantage of two types of cryptography.

Elgamal, as a typical public key encryption system, is widely used, and we use Elgamal for key agreement to resolve the RC4 key management issues. Elgamal encryption is an asymmetric key encryption algorithm for public-key cryptography which is based on the Diffie-Hellman key agreement. Elgamal encryption consists of three components: the key generator, the encryption algorithm, and the decryption algorithm. Elgamal encryption can be defined over any cyclic group G. Its security depends upon the difficulty of a certain problem in G related to computing discrete logarithms.

Elgamal encryption is probabilistic, meaning that a single plaintext can be encrypted to many possible cipher texts,with the consequence that a general Elgamal encryption produces a 2:1 expansion in size from plain text to cipher text. Encryption under Elgamal requires two exponentiations; however, these exponentiations are independent of the message and can be computed ahead of time if need be. Decryption only requires one exponentiation. The RC4 algorithm encryption improved data processing is shown in Figure 2.



Fig 2. Data Processing of Improved RC4

The RC4 encryption process is divided into three stages: session key negotiation stage, seed key generation stage and data processing stage. First of all, RC4 uses the original session key 16 byte random number KS0 generated by A, and then it's encrypted by Elgamal. When B receives the encrypted information, decrypts with its own private key, B will get the shared key, and then return A an OK message. Secondly, A uses the RC4 algorithm and KS0 to get a pseudo-random output stream, after moving the first 48 bytes away from the output stream, select the interception between 49 bytes and 52 bytes as IV, select interception between 53 bytes and 80 bytes as the user shared key KC, and select the interception between 81 bytes and 96 bytes as the next key KS1. Thirdly, A uses a certain combination of Hash functions to get the seed key, and then we finish the seed key generation stage.

Finally, we get a new pseudo-random data stream with the seed key generated by RC4 algorithm, discard the first 48 bytes of the data stream, and then encrypt the P. A. Send the encrypted result cipher text C(RC4(key, P)) to B. B decrypts the received data in the same way and gets the message data P. In the entire process, after obtaining the key, B uses the same algorithm to generate seed key in order to obtain the cipher text C for decrypting later data. During each connection, the seed key used in the following communication is generated by using the initialize vector IV in step 2, plus 1, and then calculated by Hash function with KC. At the same time there is a limited time T0 to use the KC. During each communication, if the time that using KC is over T0, we use the KS1 in Step 2 of the last connection as the new key, and then repeat the Step 2 to regenerate a new IV and KC. Now, we discuss the improved algorithm's security. We no longer need to transfer plaintext and IV to the other by using the RC4 algorithm to generate the IV and KC, so the attackers can not simply use the weak IV keys to cryptanalysis and attack, at the same time it's more useful to avoid the reuse of the keys, avoid getting the known plaintext from some of the other clear plaintext. The IV initialization is no longer binding with the network card, but with part of random RC4 stream, so

has been rigorously tested for security loopholes for a few years before it was standardized by NIST.

the IV administration is also safe. As the key generated by both IV || RC and RC || IV can be analyzed by IV linear changes. In the improved method, we use the Hash Function to deal the IV and KC, so if the changes produced by the linear changed, IV will not be linear that avoids analyzing the key by the linear changes of IV. Assault for different lengths IV has different analysis lengths. If the IV length is 4 byte, the probability that each IV can be used for the first byte correlation analysis is only $4.32 \times 10-5$, and the number of weak IV that needs to analyze a byte KC in the key is 1.33×106 . In order to improve the RC4 security, we use a 256-bit key.

In the analysis of 8 byte RC4 pseudo-random streams, we get the result that the first output bit has 36% probability to equal with the approximate; the second bit has 35.9% probability, and so on. The 48th bit has 0.4% probability. Therefore, in order to ensure the difficulty of cryptanalysis, in the improved RC4 method, we don't use the first 48 bit pseudo-random stream to avoid the attack by using the bias of the first few bits in output stream. In the 11Mbps network, the transmission of 1500 byte data packets will come up with the situation that different packets use the same IV in about 5 hours:

11(Mbps)/ (1500Byte/packet*8bit/Byte) = 917 packet/s 224 = 1677216

1677216/917 = 5.1 h

Using the improved RC4 in the 11 Mb/s networks, the time that different packets use the same time IV in the transmission of 1500 Byte data packets is about 54 days. 232 = 4294967296

4294967296/917 = 1301 h = 54 d

IV. WIFI ENCRYPTION AND DECRYPTION ARCHITECTURE

It consists of Encryption and Decryption architectures and its description

4.1 WiFi Encryption Architecture



Vol. 1, Issue 3, pp.653-659



Fig.5 RC4 Internal Architecture

Description:

It gives the detailed architecture for WiFi Encryption. Architecture is divided into two parts. They are Key Generation and Plain Text data and controller. Plain Text Data is stored in FIFO. Its Data Length is 8. Key Data is stored in RAM. Its Data Length is 8. DataLenCnt is used to count the number of bytes. After xoring plain text and key data we will get cipher text.

4.2 WiFi Decryption Architecture



Fig.6 Decryption Architecture

Vol. 1, Issue 3, pp.653-659

Description:

It gives the detailed architecture for Wi-Fi Decryption. Architecture is divided into two parts. They are Key Generation and Cipher Text and controller.

Cipher Text is stored in FIFO. Its Data Length is 8. Key Data is stored in RAM. Its Data Length is 8. DataLenCnt is used to count the number of bytes. After xoring cipher text and key data we will get plain text.

V. SIMULATION RESULTS

Top Module: Block Diagram:



Fig.7 Top Module

Description:

It takes the Key Data and Plain text as inputs, it produces cipher text as output. Whenever EncryptionEna is high it starts working.

Waveforms:

Name	Va	S 1 2840 1 2860 1 2880 1 2900 1 2920 1 2940 1 2960 1 2980 1 3000 1 3020 1 3
R= Clk	0	mmmmmmm
R= Rst	1	
⊞R= FIFOIn	43	X XC7X43X45X05X XC7X43X45X05X XC7X43X45X05X XC7X43X45X05X XC
R= WrEna	0	
R= EncryptionEna	1	
⊞R= KeyData	48	8)(9)(8)(8)(8)(8)(8)(8)(8)(8)(8)(8)(8)(8)(8)
⊞R= MaxData	FF	
🏾 EncryptedData	0	
🗢 SBit	0	
⊞R= PS	6)2(4)6)7(8)2(4)6)7(8)2(4)6)7(8)2(4)6)7(8)2(4)
⊞R= NS.	7	<u>)+)@7)@2+)@7@2+@7@2+)@</u>
🗄 🗢 ERAMOut	00	
🕀 🕫 FIFOOut	00	
⊞ 🖶 XorOut	00	
R= NewData	0	

R= Clk	0	mmmmmmmmm
R= Rst	1	
€ R= FIFOIn	43	3)(8)(3)(10)(3)(8)(8)(3)(3)(8)(3)(10)(3)(8)(3)(10)(3)(8)(3)(3)(3)(3)(3)(3)(3)(3)(3)(3)(3)(3)(3)
R= WrEna	0	
R= EncryptionEna	1	
⊞R= KeyData	48	34(8)(8)(8)(3)(4)(8)(8)(3)(4)(8)(8)(3)(4)(8)(8)(3)(4)(8)(8)(4)(8)(8)(4)(8)(8)(4)(8)(8)(4)(4)(8)(8)(4)(4)(4)(4)(4)(4)(4)(4)(4)(4)(4)(4)(4)
⊞R= MaxData	FF	
EncryptedData	0	
● SBit	0	
€R≈ PS	6	
⊞R= NS	7	£\$
🗄 🕈 ERAMOut	00	
🗄 🔹 FIFOOut	00	
🗄 🖷 XorOut	00	
R= NewData	0	

R= Clk	1	Immm	MMM	www.
R= Rst	1			
■ R= FIFOIn	07	5/26/2B/07/C3/35/26/2B/07/C3	3)(35)(26)(2B)(07)(C3)(35)(26 (2B)07)C3 (35)26 (2B)07)C3 (35
R= WrEna	0			
R= EncryptionEna	1			
⊞R= KeyData	08	3 48 (C3 08 09 18 48 (C3 08 09	8)(8)(8)(8)(8)(8)(8)	48 (C8 (08 (09 (18 (48 (C8 (08 (09 (18
⊞R= MaxData	FF			
EncryptedData	1			
● SBit	1			
€R= PS	С		<u>Хв</u> Хс	XB XC
∎R= NS	C	XB)C XB	Xc	<u>)(B)(C)(B</u>
🗄 🕶 ERAMOut	00		X5E X00	X46 X00
🛨 🔹 FIFOOut	26	X07	Хсз	X35
🗄 🕈 XorOut	00	(2B)(00)(9D)(00	X73 X00
R= NewData	0			

VI. SYNTHESIS REPORT

It consists about Synthesis Reports, Map Reports, RTL Schematic Diagrams and Floor Plan Design

Synthesis Report for Encryption:

RTL Top Level Output File Nam	ne : TopModule.ngr
Top Level Output File Name	: TopModule
Output Format	: NGC
Optimization Goal	: Speed

Selected Device: v600efg676-8

Number of Slices: 6057 out of 6912 87% Number of Slice Flip Flops: 6574 out of 13824 47% Number of 4 input LUTs:11015 out of 13824 79% Number of IOs: 30 Number of bonded IOBs:30 out of 444 6% Number of GCLKs: 1 out of 4 25%

Synthesis Report for Decryption:

RTL Top Level Output File Name: TopModule.ngrTop Level Output File Name: TopModule

Output Format	: NGC
Optimization Goal	: Speed
Selected Device	: v600efg676-8

Number of Slices: 6030 out of 6912 87% Number of Slice Flip Flops: 6560 out of 13824 47% Number of 4 input LUTs:10967 out of 13824 79% Number of IOs: 37 Number of bonded IOBs: 29 out of 444 6% Number of GCLKs: 1 out of 4 25%

Internal Diagram of Encryption Top Module



Internal Diagram of Decryption Top Module



Floor plan Design for Encryption:

🐼 Xilinx Floorplanner - TopModule										E	6	
Pile Edit View Hererchy Pattern Ticopian Window Help												
DF = 8 + 8 = 8 9 9 9 9 4 1 4 8 8 2 4 9 X 9 9 X 9 2												
TopModule Design Hiera												
TopHodule "TopHodule" [3010	ΠT	TopModule Placement for xcv600e-8-fg676										
CarryChain_1 "CarryChain" CarryChain_3 "CarryChain" CarryChain_3 "CarryChain" CarryChain_1 "CarryChain"	2	0	Y	G		0						
M2/Nadd_Adder10ut_addt M2/Nadd_Adder10ut_addt M2/Nadd_Adder10ut_addt M2/Nadd_Adder10ut_addt M2/Nadd_Adder10ut_addt	Ň	(F)	×	(h)		(h)					()	
M2/Nadd_Adder10ut_addt M2/Nadd_Adder10ut_addt M2/Nadd_Adder10ut_addt		G		G	Y	G	×.	0 V			4.5.1	
TopModule Design Nets TopModule Design Nets Co.,BLFGP/BUFG Co.,BLFGP/BUFG N7542 N7542		(H)		(h)	×	(H)	×		F X			
N7541 N7550 N7539 N7539 N7535 N7535 N7535	7	0		0		6					()	
N7533 N7533 N7531 N7530 N7530 N7529	×]			(h)		(r)		FX		\mathbb{P}	<u> </u>	
M19/P5_10_23 N7528 M6/AddMusDuk7>_5	1.	~	÷,	R25C36		~	Ċ	_∧ ⊓	r r	R25C38	•	
For Help, press F1	_	_	~	_	- 01 -		- L m		R0C24	N	и	
JEWIR Z Xilinx - DE - H V Ro	.1 - Pairi		G Synthesis		뗖	sonistatic	I 🗐	HoorMan_Enc	XIInx Roorplan	(()))) \$ (0)	2:2391	

Vol. 1, Issue 3, pp.653-659

Floor Plan Design for Decryption

🛿 XIIInx Floorplanner - TopModule 📃 🖻 🗵												- 🖻 🛛	
File Edit View Herarchy Pattern Roorplan Window Help													
D # 8 8 # 8 = 4 9 9 9 8 4 7 1 4 6 8 4 9 X 9 9 X 9 8 4													
TopModule Design Hiera													
TopModule "TopModule" [2910		TopModu	ile Pla	cement for xcv600e-	3-fg676						1		
CarryChain_4 "CarryChain"	I P	Ø		0000	00	DE	ЪŨ	D	Ð	Ø			
CanyChain_3 "CanyChain" CanyChain_1 "CanyChain"	þ	D		DØDØ	DØ		DØ	Ð	D	D			
	þ	Þ			00	Þ	00	$\boxdot \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$	Ð		Ì		
	þ	Ð		DØDØ	DØ	D	DØ	\triangleright	Ð		Ð	1200-es	
	þ	Þ			Ð	Ð	ÐÐ	Ð	Į.	Ð	Ì	Ð	
	þ	Ð		DODO	Ð	Ð	\mathbb{D}	Ð	E	\square		D	
11 ×	þ	Ð			ÐØ	\square	Ð	Ð	Ę.		Ð	У	
Ck. BUFGP/BUFG Ck. BUFGP/BUFG Ck. BUFGP/BUFG Encompared bary Tb.	Þ	Ð		DODO	DØ	\square	Ð	\triangleright	Ē			DE	
EncryptedData(2) EncryptedData(2) EncryptedData(3) EncryptedData(4)	93	00	0		Ð	Ð	ÐØ	\triangleright	Ð	Ð		У	
EncryptedDatact5 EncryptedDatact5 EncryptedDatact5 N7454 N7454	L P i	D	0	DØDØ	Ð	D	DØ	DØ	D	Ð		DE	
	1.	1											
Tor Help, press #1 825027-51.09Y 968												NUM	
SEMR Z Minx - ISE - HI 😵			-	dinx Timing A 🕅 X		📴 Xiin				6.	120	11145 AM	

VII. CONCLUSIONS

WLAN has some security weakness due to RC4 weakness, linear weakness and IV weakness. The improved RC4 can raise the security level of RC4, so does the WLAN, and it can be used as temporary method as it's easy to update. The new block encryption algorithm, such as RC5, will be used as the security solution for its high encryption level in future. Enhancing RC4 algorithm consists of Elgamal, KSA,PRGA and Hash Function.

Architecture for Enhancing RC4 is designed. Individual modules are designed and integrated. Simulation results for individual and integrated modules are verifed. Synthesis results are obtained.

REFERENCES

[1] Yao Yao Jiang Chong , Wang Xingwei "Enhancing RC4 algorithm for WLAN WEP Protocol" 978-1-4244-5182-1/10/ $26.00 \otimes 2010$ IEEE

[2] Jun-Dian Lee and Chih-Peng Fan*, Efficient Low-Latency RC4 Architecture Designs for IEEE 802.11i WEP/TKIP, Pages:4, Proceedings of 2007 International Symposium on Intelligent Signal Processing and Communication Systems Nov.28-Dec.1, 2007 Xiamen, China, 1-4244-1447-4/07/\$25.00 ©2007 IEEE.

[3] P. Kitsos, G. Kostopoulos, N. Sklavos, and O. Koufopavlou, HARDWARE IMPLEMENTATION OF THE RC4 STREAM CIPHER, pp:1363-1366, 0-7803-8294-3/04/\$20.00 02004 IEEE.

[4] Yukiyasu Tsunoo, Teruo Saito, Hiroyasu Kubo, and Tomoyasu Suzaki, A Distinguishing Attack on a Fast Software-Implemented RC4-Like Stream Cipher pp.3250-3255, 2007

[5] Sammy H. M. Kwok and Edmund Y. Lam, Effective Uses of FPGAs for Brute-Force Attack on RC4 Ciphers. pp.1096-1100,2008

Vol. 1, Issue 3, pp.653-659

[6] A. G. Chefranov, Eastern Mediterranean University, T. A. Mazurova Taganrog State University of Radio Engineering, Pseudo-Random Number Generator RC4 Period Improvement, 1-4244-0361-8/06/\$20.00 ©2006 IEEE.

[7] K.H. Tsoi, K.H. Lee and P.H.W. Leong, A Massively Parallel RC4 Key Search Engine, Pages:9, Proceedings of the 10 th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02), 1082-3409/02 \$17.00 © 2002 IEEE.

[8] Jovan Dj. Goli'c, Linear Models for a Time-Variant Permutation Generator, pp.2374-2382,1999

[9] Zong Wang, Tughrul Arslan, and Ahmet Erdogan, Implementation of Hardware Encryption Engine for Wireless Communication on A Reconfigurable Instruction Cell Architecture, pages:5, 4th IEEE International Symposium on Electronic Design, Test & Applications

[10] Nathaniel Couture, Kenneth B. Kent, The Effectiveness of Brute Force Attacks on RC4. Pages:4, Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04) 0-7695-2096-0/04 \$20.00 © 2004 IEEE.