Reconfigurable Feedback Shift Register Based Stream Cipher for Wireless Sensor Networks

Guang Zeng, Xiaodai Dong, *Senior Member, IEEE*, and Jens Bornemann, *Fellow, IEEE*

***Abstract*—Secure wireless communications among sensor nodes is critical to the deployment of wireless sensor networks. However, resource limited sensor nodes cannot afford complex cryptographic algorithms. In this letter, we propose a low complexity and energy efficient reconfigurable feedback shift register (RFSR) stream cipher. The RFSR adds one new dimension, reconfigurable cipher structure, to the existing stream ciphers. The proposed RFSR is then implemented on a field programmable gate array platform. Simulation results show that much lower power consumption, delay and transmission overhead are achieved compared to the existing microprocessor based cipher implementations.**

***Index Terms*—Stream cipher, wireless sensor network, energy efficient cipher, feedback shift register.**

I. Introduction

AFTER years of research and development, wireless sensor networks (WSNs) are being deployed for various industrial and consumer applications. Sensor nodes in WSNs are low-cost, low-power, small size devices capable of monitoring, data collection and transmission. However, the fact that anyone with proper receiving tools has access to the signal in the air, makes security a main issue of WSNs. Modern WSNs are bi-directional, also enabling sensor nodes to control other logically connected devices. The use of control functions requires higher security mechanisms to prevent attacks.

However, sensor nodes are low-cost, computational- and energy-limited devices which cannot afford resource consuming cryptography algorithms. Therefore, proper security schemes befitting the requirements of WSN should guarantee both sufficient level of security and low resource consumption. Conventional public-key cryptography seems feasible but the computational overhead is too large for resource-limited sensor nodes [1]. Private-key cryptography, also known as symmetric cryptography, is suitable for environment constrained applications such as sensor nodes. Private-key encryption uses either stream ciphers or block ciphers. Compared with block ciphers, stream ciphers are often simpler but sufficiently secure. In a stream cipher, plaintext and keystream are bitwise combined using exclusive-or (xor) operation to generate ciphertext. The keystream is a pseudorandom bit stream generated serially using shift registers in a stream cipher. Ciphertext is transmitted over the air between two communicating nodes. The decryption process on a receiving node resembles the encryption process by bitwise xor of the ciphertext with the keystream to restore the plaintext.

Manuscript received April 23, 2013. The associate editor coordinating the review of this letter and approving it for publication was X. Fu.

The authors are with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, British Columbia, V8W 3P6, Canada (e-mail: {gzeng, xdong}@ece.uvic.ca, j.bornemann@ieee.org).

Digital Object Identifier 10.1109/WCL.2013.13.130292

Software implementation of cryptography algorithms is usually carried out by the embedded processor in a sensor node. However, the computational resource limited embedded processor is also responsible for other operations, such as sensor control, communication protocol execution and sensor data processing. While simple security algorithms may have weaknesses for certain security attacks, complex security algorithms will definitely take up much of a processor's resources and negatively impact other real-time tasks' running. Hardware encryption implementation frees a processor from heavy duty security function processing and becomes a natural choice for commercial uses such as A5/1 cipher of GSM, E0 cipher of Bluetooth, etc. Hardware oriented stream cipher design has relatively low power consumption, constant and predictable delay and high throughput rate, which makes it a good choice for sensor nodes.

A feedback shift register (FSR) based stream cipher uses feedback update functions to generate new internal states from the current internal states. The feedback update functions are fixed in stream ciphers. Traditional stream ciphers can increase their resistance against attacks by increasing the key and initial vector (IV) sizes. However, if the feedback update functions are designed to be dynamic, attacks will become harder to accomplish because both the cipher structure (the feedback update functions) and the secret key are unknown.

In this letter, we propose a light-weight hardware-oriented cryptography algorithm, i.e., the reconfigurable feedback shift register (RFSR) based stream cipher, and implement it on a reconfigurable device to test its performance. In our design, the feedback shift register based cipher is structure reconfigurable. This scheme guarantees high message confidentiality for WSNs. Comparing with the existing microprocessor based platforms, the proposed scheme achieves over 130 times less average energy consumption, and over 25 times less delay.

II. System Model

A. Network Model

A wireless sensor network is composed of resource limited sensor nodes and powerful base stations (BSs). The WSN may contain one or several BSs depending on its network topology. A sensor node communicates with a base station in one hop or multiple hops through other nodes or BSs. Encryption is performed in both application layer and data link layer. The application layer encryption guarantees that only the destination BS and the original sender have access to the sensor data. The data link layer encryption requires unique pairwise ciphers established between neighbouring nodes to protect relayed messages. Each sensor node is equipped with both a microprocessor and a hardware component. The

hardware component can be reconfigurable devices such as field-programmable gate array (FPGA) or application-specific integrated circuit (ASIC). In this letter, we simulate FPGA implementations to demonstrate the performance of the proposed cipher. The hardware component is used only for encryption and decryption.

B. Security Model

We assume that sensor nodes have no tamper-resistant mechanism. Once sensor nodes are captured and compromised by an adversary, all stored data such as cipher structures and keys will be exposed and can be utilized by reprogramming captured nodes. An adversary can also launch passive attacks which attempt to break the cipher by eavesdropping on communications between legitimate nodes. Denial of service (DoS) attacks can be mounted to disrupt regular communications between nodes, or to drain up nodes' energy. DoS attacks in the data link layer and the application layer are considered in the letter.

III. The Reconfigurable Feedback Shift Register Cipher

The proposed algorithm is based on Grain cipher [2], and hence we first briefly review Grain cipher and its application in WSN in Section III-A. Afterwards, the RFSR cipher is described in detail in Section III-B. Finally, the initialization process of the RFSR cipher is presented in Section III-C.

A. Grain Cipher

Grain is a family of stream ciphers selected for the final eSTREAM portfolio for Profile 2 by the eSTREAM project. It is known for its hardware-oriented, elegant and simple design. The essential feature of the algorithm is one output function and two sets of feedback shift registers - one with linear feedback update function and the other one non-linear. The feedback functions are fixed. Keys and IVs are used as the initial values of linear feedback shift register (LFSR) and non-linear feedback shift register (NFSR), respectively. The original design uses 80-bit key and 64-bit IV. Grain128 has 128-bit key and 96-bit IV. Previous research shows that radio transmission consumes much more power than computation [3][4]. Longer keys and IVs lead to larger communication overheads for key establishment and update. Since the IV is transmitted in each packet, reducing the size of the IV will significantly decrease total power consumption. Besides, Grain cipher can be conveniently modified to multiply its throughput rate by using additional parallel feedback update functions and output functions.

B. RFSR Cipher

Similar to the Grain cipher, the proposed reconfigurable feedback shift register based cipher, depicted in Fig. 1, consists of three main building blocks, namely a LFSR with linear feedback update function f, a NFSR with non-linear feedback update function g, and an output function h. In our design, we use a 32-bit LFSR and a 64-bit NFSR while 128-bit LFSR and NFSR are used in Grain128. Other choices of sizes can be

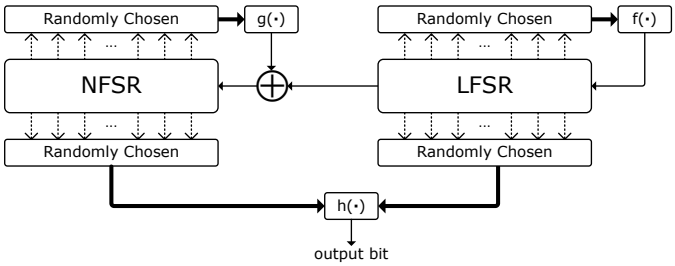

Fig. 1. RFSR cipher structure.

carefully designed as well. The states of the LFSR are denoted as $y_1, y_2, ..., y_{32}$. Similarly, the states of the NFSR are denoted as $z_1, z_2, ..., z_{64}$. The reconfigurable feedback update function of the LFSR f is defined as

$$f : y_0 = y_{a_1} + y_{a_2} + y_{a_3} + y_{a_4} + y_{a_5} + y_{32},$$

where a_1 to a_5 are carefully chosen so that f is a primitive polynomial of degree 32. The primitive polynomial guarantees that the internal states of the LFSR can reach the maximum period of $2^n - 1$, where n is the size of the LFSR. Since the primitive polynomial has been studied extensively, taps of the LFSR feedback update function in our design are randomly chosen from a structure pool of 5039 existing 32-bit primitive polynomials [5].

The feedback update function of the NFSR is defined as

$$g : z_0 = y_{32} + z_{64} + z_{b_1} + z_{b_2} + z_{b_3} + z_{b_4} + z_{b_5} \cdot z_{b_6} + z_{b_7} \cdot z_{b_8} + z_{b_9} \cdot z_{b_{10}} + z_{b_{11}} \cdot z_{b_{12}} + z_{b_{13}} \cdot z_{b_{14}} \cdot z_{b_{15}} + z_{b_{16}} \cdot z_{b_{17}} \cdot z_{b_{18}} \cdot z_{b_{19}},$$

where z_{b_1} to $z_{b_{19}}$ are randomly but not repeatedly chosen from the states of the NFSR, z_1 to z_{63}. According to boolean algebra, repeating values in b_1 to b_{19} will reduce monomial numbers and then compromise the designed security level.

The output function h gets its input from the states of both LFSR and NFSR. It is defined as

$$h : output\_bit = y_{c_1} + z_{d_1} + z_{d_2} + z_{d_3} + y_{c_2} \cdot y_{c_3} + y_{c_4} \cdot z_{d_4} + z_{d_5} \cdot z_{d_6} + y_{c_5} \cdot z_{d_7} \cdot z_{d_8},$$

where y_{c_1} to y_{c_5} and z_{d_1} to z_{d_8} are randomly but not repeatedly chosen from y_1 to y_{32} and z_1 to z_{64}, respectively.

For the RFSR cipher, the feedback functions f,g and h are all reconfigurable while these functions in Grain128 are fixed. In RFSR, f is composed of 4 or 6 dynamic taps while that in Grain128 has 6 fixed taps; g is composed of 6 degree-one, 4 degree-two, 1 degree-three and 1 degree-four monomials while that of Grain128 is composed of 6 degree-one, 7 degree-two monomials; h is composed of 4 degree-one, 3 degree-two and 1 degree-three monomials while that of Grain128 is composed of 8 degree-one, 4 degree-two and 1 degree-three monomials.

As indicated above, in addition to the key update of traditional stream ciphers, the proposed RFSR cipher can also update its structure. The structure update consists of three basic elements: f update, g update, and h update. A system can carry out partial or total structure update which means one or two, or all of the three basic elements are updated. Any change in the cipher structure will definitely change the output keystream and result in a brand new cipher. Since the keystream is a pseudorandom bit stream generated by the stream cipher and known to both communicating parties, it

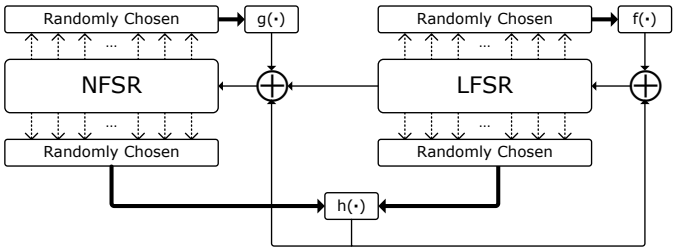

Fig. 2. RFSR initialization.

can be applied to generate a new cipher structure. Proper algorithms can be easily designed to update cipher structures by use of the previously generated keystream. Moreover, a security mechanism in upper layers can be designed to control the structure update, such as partial or total update, update frequency, etc.

C. Key and IV Initialization

In addition to the structure update, keys and IVs are updated on a regular basis. The cipher initialization process is shown in Fig. 2. It is first clocked 96 (=32+64) times without producing the keystream to make sure that all the initial states of the cipher have influence on the cipher states after initialization. Although there exists a related key-IV pair weakness in the initialization as Grain cipher [6], it can hardly lead to any effective key recovery attacks.

IV. Security Analysis

A. Cipher Security

Since the proposed RFSR cipher is designed based on the Grain cipher, the cryptographic analysis on Grain can also be applied to RFSR ciphers. By now, no key recovery attacks more effective than brute force attack are known against Grain128, indicating the level of security of Grain. Several minor differences between RFSR and Grain are analyzed below.

In RFSR, the simplified feedback and output functions and the smaller 96 internal states of 32-bit LFSR and 64-bit NFSR in the contrast to the 256 internal states of 128-bit LFSR and NFSR in Grain128, seem to make RFSR cipher more vulnerable to attacks. However, the changeable cipher structure makes the RFSR cipher much more difficult to succumb to attacks. Assume an adversary may have access to the entire 5039 LFSR structure pool by compromising a large number of sensor nodes. The basic NFSR feedback boolean function and basic output boolean function may also be available. But the random taps used in the NFSR feedback function or in the output function are still unknown, which are 19 taps of NFSR states in the NFSR feedback function, 8 taps of NFSR states and 5 taps of LFSR states in the output function. The total possible structure will be $5039 * \binom{63}{19} * \binom{32}{5} * \binom{64}{8} \approx 1.33 * 2^{114}$. Therefore, it is hard to launch attacks on RFSR ciphers.

B. Attack Analysis

The large number of possible cipher structures makes eavesdropping hard to compromise system security. Note that different pairwise RFSR ciphers are established and used in the data link layer between neighbouring sensor nodes, and another RFSR cipher is used in the application layer between the BS and the source sensor node. Even when several nodes are captured by adversaries, only the ciphers owned by these nodes are exposed but they cannot be utilized to break uncompromised nodes' ciphers. DoS attacks and forgery from an outsider are defended with the use of a message authentication code (MAC). A MAC is added to each message's payload and helps the receiver verify the authenticity and integrity of the received messages. A message is considered valid only if the received MAC is correct. The remedies for the DoS attacks coming from the captured nodes have been proposed in the literature, such as switching to low duty cycle and conserving power, locating attack area and re-routing traffic, and adopting prioritized transmission [7].

V. Implementation, Simulation and Performance

Altera Cyclone II EP2C8T144C6 FPGA was chosen as the target implementation device. The simulation software platforms are Altera Quartus II V12.1 and Mentor Graphics ModelSim SE 10.1a. We simulate the power consumption using the Altera PowerPlay power analysis tool [8]. PowerPlay uses actual design placement and routing and logic configuration which is claimed to be accurate (to within $\pm 10\%$) for the actual device power consumption [9]. Existing experiments [10] also show that the result of PowerPlay power estimation on Cyclone II series is reasonable.

The total FPGA power consumption comprises static power and dynamic power. Static power is the power consumed by a device due to leakage currents when in quiescent state. Dynamic power is the additional power consumed through device operation caused by signals toggling and capacitive loads charging and discharging. Therefore, with increasing operating clock frequency, the dynamic power increases accordingly but the static power remains the same.

Firstly we execute the gate-level timing simulation, which takes all the routing resources and the exact logic array resource usage into account to obtain an accurate power estimation. Then PowerPlay is run to measure the average power consumption of each operation. We obtain the power consumption directly from the PowerPlay tool report and calculate the energy-per-bit performance.

A. Cipher Implementation

The proposed implementation achieves several cipher functionalities with only one structure implementation. Each cipher's specific information, such as key, IV, feedback taps, and output function, are stored in random access memory (RAM). Since one sensor node needs several RFSR ciphers for data link layer pairwise encryption and application layer encryption, the proposed implementation builds upon a basic cipher infrastructure, and the system automatically loads specific cipher information from RAM when required.

Similar to the Grain's structure, the throughput of the proposed RFSR cipher can be easily multiplied by implementing feedback functions and output functions several times. Average power consumptions are compared with different

TABLE I
COMPARISONS WITH MICROPROCESSOR PLATFORMS

Platform	Algorithm	Clock(MHz)	Delay(us)	Energy(nJ/bit)
FPGA	RFSR	50	2.08	0.32
ATmega103	RC4	4	3262	105.12
StrongARM	RC5	206	53	41.41

TABLE II
CIPHER COMPARISONS ON FPGA

Algorithm	Logic Elements	Delay (us)	Energy (nJ/bit)
RFSR	5207	2.08	0.56
RC4	12917	6.40	11.18
RC5	6172	18.56	7.75

throughput rates at 1 and 8 bits per clock cycle, and different clock rates at 10 and 50 MHz. We find from simulation that the average energy consumption of 8 bits per clock implementation is almost 6 times less than 1 bit per clock. As expected, with different clock rates, the static power is almost the same but the dynamic power is proportional to the operating frequency.

B. Comparison with Microprocessor Platforms

Previous research [11] studied the performances of several ciphers and hash functions on microprocessor platforms. We choose ATmega103 and StrongARM microprocessor platforms which respectively represent low-end and high-end processors. Since a microprocessor processes one instruction per clock, the fastest encryption scheme for a particular platform is also the most energy efficient scheme. The most energy efficient algorithms for ATmega103 and StrongARM platforms are RC4 and RC5, respectively. Existing flaws [12][13] make RC4 and RC5 susceptible to attacks, while brute force attack remains one of the most effective attack against Grain128, indicating the higher security of Grain128. The plaintext to be encrypted is 512 bits long, and initialization is executed before encryption. Per bit energy consumption is calculated by averaging both initialization and encryption energy consumption over the total 512 bits.

The results and comparisons are shown in Table I. The average energy consumptions of ATmega103 and StrongARM are 329 and 130 times more than that of the proposed RFSR scheme, respectively. Even though StrongARM is running 4 times faster, the delay is still 25 times larger than that of RFSR FPGA implementation while the delay of ATmega103 is 1568 times larger with 12.5 times slower clock than those of the proposed.

Comparing with FPGA, ASIC implementation runs faster and is more energy efficient but much more expensive to prototype. Existing research [14] compares FPGA and ASIC design in circuit speed and power consumption and shows that ASIC designs are 87 and 14 times less than FPGA design, in static and dynamic power consumption respectively. The proposed scheme therefore uses even less power with ASIC implementation. Even though the comparisons in Table I are based on different platforms and different encryption algorithms, it is clear that the hardware-oriented RFSR scheme is better suited for use in sensor nodes due to low energy consumption and small delay.

We also implement RC4, RC5 and the proposed RFSR cipher on the same FPGA platform for fair comparison. The tests are run on Altera Cyclone II EP2C15AF256A7 at 50 MHz clock rate. As shown in Table II, the hardware-oriented RFSR cipher entirely outperforms RC4 and RC5.

C. Comparison with Grain128

The proposed design is compared with Grain128 on the energy consumption of keystream generation with 10 MHz clock and 8 bit/clock throughput rate. The results are comparable: 0.253 nJ / bit for Grain128 and 0.544 nJ / bit for the RFSR scheme. However, considering the transmission overheads caused by the IV size, the energy consumption of the proposed RFSR scheme is 10.3% lower than Grain128 for a packet size of 512 bits. Besides, to break RFSR by brute force, it requires about $1.33 * 2^{34}$ times more complexity than for Grain128.

VI. CONCLUSION

In this letter, we have proposed a low complexity reconfigurable feedback shift register (RFSR) based stream cipher and shown that it is more secure than the widely used Grain, RC4 and RC5 algorithms. Implemented on an FPGA platform, the proposed scheme consumes over 130 times less average energy, and renders over 25 times less delay than existing microprocessor platforms.

REFERENCES

[1] D. Malan, M. Welsh, and M. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," in *Proc. 2004 IEEE Conf. on Sensor and Ad Hoc Communications and Networks*, pp. 71–80.
[2] M. Hell, T. Johansson, and W. Meier, "Grain: a stream cipher for constrained environments," in *Int'l J. Wireless and Mobile Computing*, pp. 86–93. Jan 2007.
[3] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *Proc. 2005 IEEE Intl. Conf. Pervasive Computing and Comm.*, pp. 324–328.
[4] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *SIGARCH Comput. Archit. News*, pp. 93–104, Nov. 2000.
[5] New Wave Instruments, "Tables of M-Sequence Feedback Taps," http://www.newwaveinstruments.com/resources/articles/m_sequence_linear_feedback_shift_register_lfsr.htm.
[6] Ö. Küçük, "Slide resynchronization attack on the initialization of grain 1.0," in eStream ECRYPT Stream Cipher Project Report, vol. 44, 2006.
[7] A. Wood and J. Stankovic, "Denial of service in sensor networks," *Computer*, pp. 54–62, Oct. 2002.
[8] Altera Corporation, *Quartus II Handbook Version 12.1*, Nov. 2012.
[9] ——, *FPGA Power Management and Modeling Techniques*, Dec. 2010.
[10] D. Meintanis and I. Papaefstathiou, "Power consumption estimations vs measurements for FPGA-based security cores," in *Proc. 2008 Intl. Conf. on Reconfigurable Computing and FPGAs*, pp. 433–437.
[11] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichitiu, "Analyzing and modeling encryption overhead for sensor network nodes," in *Proc. 2003 ACM Intl. Conf. on Wireless Sensor Networks and Applications*, pp. 151–159.
[12] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of rc4," in *Sel. Areas in Cryptography*, pp. 1–24, 2001.
[13] A. Biryukov and E. Kushilevitz, "Improved cryptanalysis of rc5," in *Proc. 1998 Advances in Cryptology-EUROCRYPT*, pp. 85–99.
[14] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, pp. 203–215, Feb. 2007.