

AN RC4 BASED LIGHT WEIGHT SECURE PROTOCOL FOR SENSOR NETWORKS

Chang N. Zhang and Qian Yu
Department of Computer Science, University of Regina
3737 Wascana Parkway, Regina, SK S4S 0A2 Canada
{zhang, yu209}@cs.uregina.ca

Abstract

As sensor networks edge closer toward wide-spread deployment, security issues become a central concern. So far, the main research focus has been on making sensor networks feasible and useful, and less emphasis was placed on security.

In this paper we present a new light weight secure protocol based on RC4 stream cipher. In addition, we construct an RC4 based one-way hash function, which can be used for key generation and data authentication. The proposed protocol is an idea for resource-constrained environments and wireless communication, and can be widely used in the applications of one-to-one secure communication as well as broadcasting and multicasting where the communication nodes have limited processing and storage capabilities.

Key Words

RC4 Stream Cipher, Sensor Network, Secure Protocol, Secure Multicast, Hash Function.

1. Introductions

Security in sensor networks is not easy. Compared with conventional desktop computers, severe challenges exist – sensors have limited processing, storage, bandwidth and energy. So light-weight is an important characteristic for the design of the secure protocol of sensor networks.

Stream ciphers and block ciphers are two classes of encryption algorithms in symmetrical cryptography. Stream ciphers encrypt individual characters of a plaintext message one at a time, using a simple time-dependent encryption transformation. Block ciphers simultaneously encrypt groups of characters of a plaintext message using a fixed encryption transformation [6]. Stream ciphers and block ciphers have their respective characteristics, but the fastest stream cipher is faster than the fastest block cipher [11] and has less complex hardware circuitry [6].

RC4 is probably the most widely used stream cipher in

the world due to its simplicity and efficiency. RC4 algorithm is a variable key-size stream cipher scheme based on a secret internal state of 256 bytes and two pointers. The data is encrypted by XORing data with the cipher stream generated by RC4 from an RC4 key. RC4 includes two parts: a Key-Scheduling Algorithm (KSA) which turns a random key into an initial permutation S , and a Pseudo-Random Generation Algorithm (PRGA) which uses this permutation to generate a pseudo-random output sequence to be the cipher stream.

A number of papers have been published to analyze methods of attacking RC4. But none of these approaches is practical against RC4 with a reasonable key length, such as 128 bits [8]. However, a serious problem was reported by S. Fluhrer et al [13]. It demonstrated that the WEP protocol is vulnerable in a number of areas which make its continued use as a security mechanism for wireless untenable. In essence, the problem is not with RC4 itself but the way in which keys are generated for use as input to RC4. This particular problem does not appear to be applicable to other applications using RC4 and can be remedied in WEP by changing the way in which keys are generated. This problem points out the difficulty in designing a secure system that involves both cryptographic functions and protocols that make use of them [8].

State Based Key Hop (SBKH) is a newly proposed one-to-one secure protocol [3] in which two communicating nodes share the common knowledge of the RC4 state. It has good potential to be commercialized for wider applications, and particularly is suitable for battery operated devices in wireless networks. SBKH is a good encryption scheme by using RC4 with offset to avoid weak key issue with RC4 and SBKH continually updates and reuses the RC4 state for a whole communication process and does not reinitialize RC4 states. But SBKH has its drawbacks because SBKH protocol suffers from fake acknowledgement attacks and does not support multicast and broadcast due to its strong resynchronization nature.

Hence, with the requirement of secure communication increases, especially support wireless networks and

applications for sending data from a source to a group of recipients with low-end devices such as wireless sensors, we need a simple, robust, and lightweight secure protocol. The proposed protocol is such a one. The proposed RC4 based light weight secure protocol works lightly with less implementation complexity and costs. In addition, the proposed protocol can be widely used in most applications. It supports one-to-one secure communication as well as broadcast and multicast.

The paper is organized as follows. Section 2 presents the proposed RC4 based light weight secure protocol. Section 3 briefly describes the secure multicast applications of the proposed protocol. Section 4 provides security and performance analysis of the proposed protocol. Section 5 concludes this paper.

2. Proposed RC4 Based Secure Protocol

2.1 Reversible RC4 States

The proposed light weight secure protocol is based on the reversible nature of the RC4 state which means that if $(S^*, i^*, j^*) = PRGA^k(S, i, j)$ then it has $(S, i, j) = IPRGA^k(S^*, i^*, j^*)$ where $PRGA^k$ denotes applying PRGA by k rounds (the same for $IPRGA^k$), S is the RC4 state of 256 bytes, i and j are two 8-bit pointers, and $IPRGA$ is the reverse algorithm of PRGA. The algorithms of PRGA and IPRGA are below (i, j, S and PRGA are the parameter of RC4, the detailed definitions of them are in [8]).

```

PRGA(S)
Initialization:
i ← 0
j ← 0
Generation loop:
i ← (i + 1) mod 256
j ← (j + S[i]) mod 256
S[i] ↔ S[j]
Output z ← S[(S[i] + S[j]) mod 256]

IPRGA(S, i, j)
Generation loop:
S[i] ↔ S[j]
j ← (j - S[i] + 256) mod 256
i ← (i - 1 + 256) mod 256
Output z ← S[(S[i] + S[j]) mod 256]

```

This nature means that any previous RC4 state can be recovered from current RC4 state by running several times of IPRGA, each of which involves a simple addition and swap operation. The reversible nature of RC4 is being used in the proposed protocol. The detailed usage of it is in Section 2.3.

2.2 Terminologies and Notations

Offset: Offset is a value of how many rounds of applying RC4-PRGA. It indicates how far down the cipher stream after running RC4-KSA to start encrypting or decrypting messages. It happens only when a key rollover takes place. The purpose is to discard offset number of encryption octets from the start of a stream to avoid the weakness of the initial RC4 states [6].

Fixed-length packet: In the proposed protocol, the length of the data packets is fixed. The input plaintext of length L is divided into one or several fixed-length data packet(s). We use F to denote the number of byte of the fixed-length packet, and n to denote the number of data packets of the input plaintext.

Sequence Counter (SC): The proposed protocol uses the sequence counter to sort order for the fixed-length packets. So the sequence counter indicates the order of the fixed-length packets.

2.3 One-to-one Secure Protocol

The proposed RC4 based light weight one-to-one secure protocol can be described as follows.

Before the transmission, the sender and receiver share the RC4 base key, the offset value, the number of packets n , and the fixed-length value F through sender-to-receiver authenticated channel. Firstly, both sides (the sender and receiver) process the RC4-KSA and get the initialization state of S . Secondly, after applying offset rounds of RC4-PRGA, both sides apply RC4-PRGA for key stream generation as shown in Figure 1. Both sender and receiver reset their SC to zero.

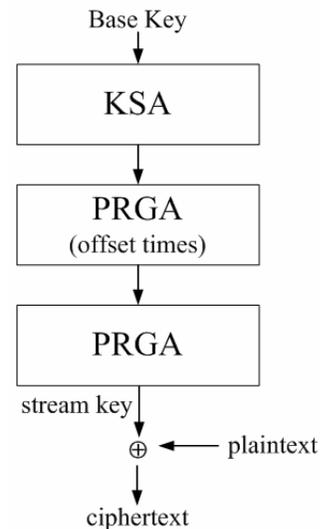


Figure 1: Flowchart of the improved RC4 algorithm

The sender divides the plaintext into one or several fixed-length data packet(s). If there are not enough data in the last fixed-length data packet, fill in random numbers in the rest of space.

The sender begins to encrypt the fixed-length data packets. First, the sender increases the SC by one (SC=1) and encrypts the first packet plaintext by applying RC4-PRGA F rounds. In each round, a stream key (byte) is generated to XOR with the next data (byte) of the packet. The encrypted packet is then sent out. Apply the same steps to the rest of the data packets stream.

When receiver gets a packet and compares the SC received with its own SC, it is easy to calculate out the suitable key stream to decrypt this packet by the reversible nature of RC4 state, and then XOR them with the encrypted data packet to restore the corresponding fixed-length data packet. The receiver can use this way to decrypt all received packets, and combine them into the entire input plaintext.

2.4 Retransmissions and Authentication

In wireless environment, some of transmitted packets may delay or drop. For the proposed protocol, the case that the packets receive in arbitrary order is allowed. For example, the sequential order of packets is 1, 2, 3, 4 when there are four fixed-length packets, but the second packet delayed in transmission, the actual order for the receiver is 1, 3, 4, 2. The proposed protocol still can handle this situation from the sequence counter. Moreover, if the receiver did not receive all packets longer than a fixed time, the receiver will send a request to the sender and let the sender retransmit the missing packets. How to choose the fixed time depends on the actual application.

In the two-party communication case, the authentication is achieved through a purely symmetric mechanism: The sender and the receiver share a secret key to compute a message authentication code (MAC) of all communication data. When a message with a correct MAC arrives, the receiver knows that it must have been sent by the sender [2]. Another approach is that the sender and the receiver share a base key. By using the RC4-based one-way hash function presented in Section 3.1, it can generate different keys. Different keys can be used to compute the different MAC, and the different MAC can be assigned for different communicated data. When a message with the corresponding MAC arrives, the receiver knows that it was sent by the sender.

3. Application to Multicast

Multicast is an inter-network service that provides efficient delivery of data from a source to a group of recipients. It reduces the communication costs for applications that send

the same data to multiple recipients. Instead of sending data via multiple unicasts, multicast minimizes the bandwidth consumption, sender and router processing, and delivery delay. In the wireless networks with low-end devices, due to the limitation of the computation, storage, power and bandwidth, it is important to reduce communication overhead and power consumption.

Group key management solution, secure data transmission, and data authentication are the major components of a secure multicast protocol. In this section, we present how to deploy the proposed secure protocol for secure data transmission in one-to-many multicast.

3.1 An RC4 Based One-Way Hash Function

Because of the reversible nature, RC4 can not be used directly to construct the one-way hash function. By a small modification, we can construct a one-way hash function by using RC4. The approach of the one-way hash function is shown by figure 3 as below.

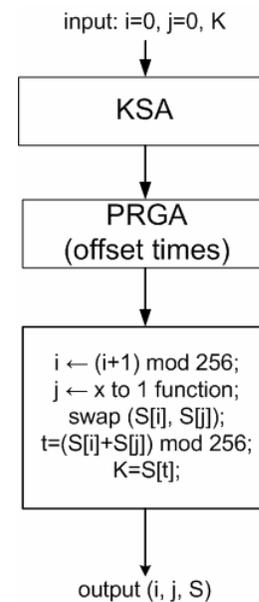


Figure 3: The proposed one-way hash function

The x to 1 function ($x \geq 2$) in figure 3 is a function with several input values but only one output value. The aim of using the x to 1 function is to break the reversible nature of RC4 and can be used to construct the one-way hash function. The x to 1 function can be realized by a large number of methods. For the different application and its security level, the user may choose the different method. The requirements and suggestions for this function are below:

- Many values can be chosen to be the input values. For example, $S[i]$, $S[j]$, $S[(i+j) \bmod 256]$, $S[(i-j) \bmod 256]$, $S[(2i+j) \bmod 256]$, $S[(i+2j) \bmod 256]$ and so on.

- The output value is an integer from 0 through 255. Generally, it can be realized through mod 256.
- Because the one-way hash function is used in the proposed light weight secure protocol, so the operations of this function need to be as simple as possible.

The proposed one-way hash function consists of three parts. The first two parts are RC4-KSA and RC4-PRGA, the last part from RC4-PRGA with a modification. The modification is in the generation process of j in the third part of the proposed one-way hash function. The RC4-PRGA gets the value of j from $j = (j+S[i]) \bmod 256$, and the third part of the proposed one-way hash function gets the value from the x to 1 function. Comparing these two methods, the scope of the result values and the probability of generating different result values are the same. The generation process is different. So the constructing approach of the proposed one-way hash function is safe.

3.2 Group Key Management and Data Transmission

An efficient and scalable group key management scheme was proposed for secure one-to-many multicast [4]. We adopt this group key management scheme combining the proposed secure protocol to provide secure multicast.

The proposed multicast scheme uses sub-grouping of multicast members to address scalability and adopts a 3-level secure multicast tree. The sender is the root of the tree, the intermediate level is composed of subgroup controllers (SCs), and multicast group members are leaves. Subgroup 0 consists of the sender and all SCs, while subgroup i ($i > 0$) consists of SC_i and some group members. Figure 4 depicts an example of the secure multicast tree.

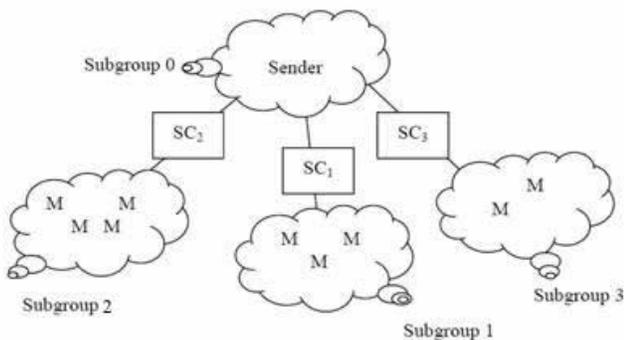


Figure 4: An example of the secure one-to-many multicast tree

The proposed multicast scheme involves the following three sets of entities: the sender, which is responsible for encrypting and distributing multicast payload data to the multicast groups and managing SCs' join and leave; a set of

registered members; and a set of SCs, which are hired by the sender to handle the workload of managing subgroups. In the proposed multicast scheme, SCs are members of the multicast group so that are believable.

We can use proposed RC4 based light weight secure protocol for data transmission. The sender is responsible (including generates a random RC4 base key, the offset value, the sequence counter and the fix-length value F , and sends them to other members in that group, the four values are the shared message in that group) for subgroup 0 and SC_i is responsible for subgroup i . The sender encrypts multicast data by using proposed secure protocol with shared message of group 0 and then distributes encrypted data to all SCs. When SC_i receives them, SC_i decrypts them and encrypt again by using proposed secure protocol with the shared message of group i , and then distributes encrypted data to all members of group i .

Each group/subgroup shares a different group/subgroup base key, so the base key is only known to the member belonging to that group/subgroup. Each new key K^m is generated by applying a one-way hash function F to the previous key K^{m-1} where m is the value of the key counter. This means that $K^m = F^m(K^0)$, where K^0 is the current key sent in group/subgroup and $F^m(x) = F(F^{m-1}(x))$. We present an RC4 based one-way hash function in Section 3.1 which can be used for group/subgroup key generation.

The adopted key management scheme [4] can efficiently handle massive member join and move requests. For the reason of limited space, we won't describe them here. The detailed approaches of approach such as member join, member leave, member move are in Zhang et al [4].

3.3 Data Authentication

The power of multicast is that one packet can reach millions of receivers. However, an attacker who sends one malicious packet can also potentially reach millions of receivers. Therefore, receivers need the multicast source authentication to ensure that a given packet originates from the correct source [12]. As for data authentication, uTESLA [2]'s advantages of loss tolerance and high efficiency make it an ideal solution to be adopted to handle unreliable multicast services and high dynamic membership updates. Therefore, we adopt uTESLA for data authentication. Below is a brief description of the uTESLA protocol.

uTESLA requires that the sender and receivers be loosely time synchronized [5]. To send an authenticated packet, the sender computes a MAC on the packet with a key that is secret at that point in time. When a receiver gets a packet, it can verify that the corresponding MAC key was not yet disclosed by the sender (based on its loosely synchronized clock, and the time schedule at which keys are disclosed). Since the receiver is assured that the MAC key is known

only by the sender, the receiver is assured that no adversary could have altered the packet in transit and stores the packet in a buffer. At the time of key disclosure, the sender notifies all receivers of the verification key. When a receiver gets the disclosed key, it can verify the correctness of the key. If the key is correct, the receiver can then use it to authenticate the packet stored in its buffer.

The RC4 based one-way hash function presents in Section 3.1 can be used in uTESLA for the data authentication. The sender first chooses a random number as the last key K_n in a one-way key chain, and generates the remaining values $(K_0, K_1, \dots, K_{n-1})$ by successively applying the proposed RC4 based hash function $F : K_j = F(K_{j+1})$. Here we assume n is large enough that the key chain is sufficiently long for the duration of the multicast. Because F is a one-way function, anybody can compute forward, e.g. compute K_0, \dots, K_j from a given K_{j+1} . On the other hand, nobody can compute backward, e.g., compute K_{j+1} for given only K_0, \dots, K_j . In the one-way key chain, keys are self-authenticating. The receiver can easily and efficiently authenticate subsequent keys of the one-way key chain using one authenticated key.

4. Security and Performance Analysis

The proposed secure protocol is based on RC4, but with several modifications. The modifications include using offset and reversible RC4 states of RC4, and changing an operation by using RC4 to construct the one-way hash function.

Reversible state is a nature of RC4. Any previous RC4 state can be recovered from the current RC4 state after simple addition and swap operations. We just use this nature and did not make any changes on RC4, so the proposed protocol does not reduce the security level of RC4.

Offset have been used in the proposed protocol to indicate how far down the cipher stream after running RC4-KSA to start encrypting and decrypting messages. The purpose is to discard offset number of encryption octets from the start of a stream to avoid the weakness of the initial RC4 states [6]. So it is not the essential change of RC4, and can increase efficiency level of RC4.

The proposed one-way hash function consists by three parts. The first two parts are RC4-KSA and RC4-PRGA, the last part has a modification from RC4-PRGA. The second part of the proposed one-way hash function runs n rounds of RC4-PRGA, and it is safer than runs of one round of RC4-PRGA. The modification is in the generation process of j in the third part of the proposed one-way hash function. The RC4-PRGA gets the value of j from $j = (j+S[i]) \bmod 256$, and the third part of the proposed one-way hash function gets the value from the x to 1 function. Comparing

with the using of the two ways, the scope of the result values and the probability of generating different result values are the same, and just the generation process is different. So the constructing approach of the proposed one-way hash function is not less secure than RC4.

The proposed protocol is simpler than most existing secure protocols. This protocol does not require key initialization frequently. This translates directly to some simplification and likely power saving. All operations in the proposed protocol are very simple. They are just swap, XOR and simple arithmetic operations. Because of the simple implementation complexity, the proposed protocol is less expensive in terms of power and CPU resources.

5. Conclusion

This paper focuses on the designs of a light-weight secure protocol which is suitable for wireless sensor networks that have limited processing, storage, bandwidth, and energy, and can be used to multicast. Below are the important features of the proposed secure protocol.

- By using offset, the proposed protocol makes effective use of RC4's strengths, and minimizes or eliminates most of its weaknesses.
- By using the reversible nature of RC4, the proposed protocol avoids the strong synchronization requirement as SBKH does.
- The one-way hash function is adopted in the proposed protocol to be used to generate new keys to handle massive member join and move requests efficiently, which eliminates secret key transmission and encryption/decryption computation. The one-way hash function is also used for the purpose of efficient authentication without the help of any public-key cryptosystem, which is especially suitable for resource-constraint equipments.
- It only uses RC4 for data transmission, key generation and data authentication in the proposed secure protocol, so the hardware is easy to design and produce, and operates with low power, processing and time costs. In addition, considering most existing hardware supporting for RC4, the proposed protocol should be compatible with existing hardware.

• The proposed secure protocol can be used in most applications, not only one-to-one secure transmission, but also broadcasting and multicasting.

From what has been discussed above, we believe that the proposed RC4 based light weight secure protocol is an ideal secure solution for resource limited environment, not only for one-to-one transmission, but also for multicast and broadcast.

References

- [1] A. M. Eskicioglu, "Multimedia Security in Group communications: Recent Progress in Key Management, Authentication, and Watermarking", *Multimedia Systems*, September 2003, pp (9) 239-248.
- [2] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security Protocols for Sensor Networks", *Wireless Networks*, 2002, pp 521-534
- [3] S. Michell, K. Srinivasan, "State Based Key Hop Protocol: A Lightweight Security Protocol for Wireless Networks", *1st ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, Venice, Italy. October 4, 2004, pp 112-118.
- [4] C.N. Zhang, and Z. Li, "An Efficient Group Key Management Scheme for Secure Multicast with Multimedia Applications", *1st European PKI Workshop*, Samos island, Greece. June 2004, pp. 364-378.
- [5] D. Mills, "Simple network time protocol (snTP) version 4 for ipv4, ipv6 and OSI", *Network Working Group*, RFC 2030, October 1996.
- [6] I. Mantin, "Analysis of the Stream Cipher RC4". Master thesis, *The Weizmann Institute of Science*, 2001.
- [7] S. Deering. "Host extensions for IP multicasting", IETF RFC 1112, August 1989.
- [8] W. Stallings, "Cryptography and Network Security, Principles and Practice, Third Edition", 2003.
- [9] A. Stubblefield, J. Ioannidis, and A.D. Rubin, "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP", AT&T Labs Technical Report TD-4ZCPZZ, Aug. 2001.
- [10] W. A. Arbaugh, N. Shankar, J. Wang, and K. Zhang, "Your 802.11 network has no clothes", *IEEE Wireless Communications Magazine*, vol. 9, pp. 44 - 51, December 2002.
- [11] R. Venugopalan, P. Ganesan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichertiu, "Encryption overhead in embedded systems and sensor network nodes: Modeling and analysis", *International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, 2003, pp 188-197.
- [12] A. Perrig, R. Canetti, D. Song, and J. D. Tygar. "Tesla: Multicast source authentication transform introduction", *IETF Internet Draft*, draft-msec-tesla-intro-01.txt, October 2002.
- [13] S. Fluhrer, I. Mantin, and A. Shamir, "Weakness in the Key Scheduling Algorithm of RC4", Proceedings, *Workshop in Selected Areas of Cryptography*, 2001.
- [14] M. Moyer, J. Rao, and P. Rohatgi, "A survey of security issues in multicast communications", *IEEE Network*, November/December 1999, pp 12-23.
- [15] P. McDaniel, H. Harney, P. Dinsmore, and A. Prakash, "Multicast security policy", *IETF Internet Draft*, draft-irtf-smug-mcast-policy-01.txt, November 2000.