SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks. (2009)
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/sec.140

A lightweight secure data transmission protocol for resource constrained devices

Qian Yu*, † and Chang N. Zhang

Department of Computer Science, University of Regina, Regina, SK S4S 0A2, Canada

Summary

In this paper we present a lightweight but robust security protocol based on the forward and backward property of RC4 stream cipher. The proposed protocol offers data confidentiality, data authentication, data integrity, and data freshness with less operation and low overhead. Also, it allows data packets to be received in any arbitrary order and achieves semantic security. Furthermore, it eliminates the requirement of frequent key renewal to overcome a special limitation of the stream ciphers. Due to the competitive efficiency and lightweight, the proposed protocol is considerable to provide secure data transmission among resource constrained devices, where the communication nodes have limited power resources and computational capabilities. This protocol not only applies to one-to-one communications, but also works for the broadcasting and multicasting. Copyright © 2009 John Wiley & Sons, Ltd.

KEY WORDS: RC4; security protocol; backward property of RC4 states; resource constrained devices

1. Introduction

Nowadays, the resource constrained devices, such as wireless sensor networks (WSNs) and radio frequency identification devices (RFIDs), have been applied to a large number of areas. Security is a critical factor for any application due to the impact on privacy, trust, and control [1]. It is also very important for many applications powered by resource constrained devices, such as battlefield control, emergency response, building monitoring, eHealth and eHome service, and nature resource management [2].

Since resource constrained devices are highly constrained in terms of resources, how to implement security to those devices is one of the major challenges [3]. Generally speaking, in order to implement security, extra resources including processor and memory usage, overhead generation and transmission, power consumption, and process time are required. However, resource constrained devices do not have enough resource and ability to process complex computing, to take heavy memory usage, and to transmit big data overhead. How to implement security affordably and efficiently without sacrificing the strength of their security properties is a big challenge. In this paper, we present a security protocol for resource constrained devices, which uses lesser resource but implement necessary security protection. To promote simplicity, we use WSNs as an example to present, but this protocol is also applicable to other resource constrained applications.

A WSN is a wireless network consisting of spatially distributed autonomous devices, sensors, to cooperatively monitor physical or environmental conditions

*Correspondence to: Qian Yu, Department of Computer Science, University of Regina, Regina, SK S4S 0A2, Canada.
†E-mail: yu209@cs.uregina.ca

such as temperature, sound, vibration, pressure, motion, or pollutants, at different locations [4–6]. It often consists of one or more control centers (base stations) and one or more routing forests established by sensor nodes. A base station is typically a gateway to another network, a powerful data processing and storage center. Typically, sensor nodes establish a routing forest with a base station at the root. A sensor node is typically equipped with a radio transceiver, a 4-bit or 8-bit low-end microprocessor, small sized memories, and a battery.

The resource constrained nature of WSNs poses great challenges to apply security. Size and cost constraints on sensor nodes result in insufficient power supplies, low bandwidth, small-size memories, and limited computing ability. It is impractical to apply conventional security mechanisms which are designed for powerful digital systems to WSNs. An affordable and efficient security approach is desired to reduce the computational time, to minimize the overhead size, and to limit the consumption of the power and memory.

SPINS [7] is a secure data transmission scheme for WSNs which contains two parts: SNEP provides data confidentiality, two-party data authentication, and data freshness, and μTESLA provides broadcast authentication. SPINS provides data confidentiality by RC5 block cipher and it avoids using asymmetric cryptography, which is normally very heavy, to achieve broadcast authentication.

TinySec [8] is a generic security package that can be integrated into WSNs applications. TinySec achieves data confidentiality through block cipher and achieves data authentication and data integrity through the message authentication code (MAC). The default block cipher for TinySec is Skipjack.

Both SPINS and TinySec achieve data confidentiality through block ciphers. However, as indicated by References [8,9] stream ciphers are almost always faster and use far less code than block ciphers. For this reason, we consider an efficient stream cipher should be considered to design a secure protocol for resource constrained devices in terms of reducing overhead, process time, and consumption of power and memory. There are two issues that limit the application of stream ciphers in data transmission, they are real-time decryption issue and frequent base key renewal issue. The proposed protocol solves those issues successfully (refer Section 2 for the detail). Also, we compared the process times among the proposed protocol, WEP, RC5 based protocol, and Skipjack based protocol in Section 4. From the comparison, we can see that the proposed protocol improves the process time significantly.

We also notice that there is a research called State Based Key Hop protocol (SBKH) [11], which attempts to design a security solution for WSNs through stream cipher. However, SBKH requires strong synchronization and maintains two to four RC4 states for each node. The strong synchronization makes SBKH unable to be applied to secure broadcast, which is an important service for WSNs. In addition, the resynchronization is a relatively high-cost process so it takes more resource usage. Furthermore, it requires more memory and operation to maintain more RC4 states for each note. The proposed protocol solves those disadvantages as it avoids the strong synchronization and it only requires each note maintains one RC4 state.

The rest of the paper is organized as follows. The introduction of the RC4 stream cipher and its forward and backward property are covered in Section 2. Section 3 presents the proposed security protocol. In Section 4, we provide security and performance analysis. Section 5 concludes this paper.

2. RC4 and its Forward and Backward Property

Stream cipher is an important class of encryption algorithms and they encrypt each digit of plaintext one at a time, using a simple time-dependent encryption transformation. In practice, the digit is typically single bit or byte. A special requirement or limitation of stream ciphers is that a basic key cannot be reused for a new message. This limitation may result in more overhead and difficulties such as key generation and distribution. The proposed protocol eliminates the requirement of frequent key renewal to overcome this weakness and keeps the compact and efficiency of the stream cipher.

RC4 is the most widely used stream cipher nowadays due to its simplicity and high efficiency [9]. RC4 is a variable key-size stream cipher based on a 256-byte internal state and two one-byte indexes. RC4 consists of two parts that are key-scheduling algorithm (KSA) and pseudo-random generation algorithm (PRGA). For a given base key, KSA generates an initial 256 bytes permutation state denoted by S_0. PRGA is a repeated loop procedure and each loop generates a one-byte pseudo-random output as the stream key. That is, at each loop a one-byte stream key is generated and is XORed with one-byte of the plaintext, in the meantime a new 256-byte permutation state S as well as two one-byte indexes i and j are updated. We call (S, i, j) an RC4 state.

DOI: 10.1002/sec

For the security strength of RC4, a number of papers have been published to analyze the possibility of attack to RC4, but none is practical with a reasonable key length, such as 128 bits [9]. So far, the practical RC4 attacks (e.g., References [11–13]) remain with WEP attacks, which aim to a key derivation problem in WEP standard [14]. WEP encryption is based on the RC4 stream cipher so each packet has a different WEP key. The key derivation function used by WEP was flawed. In essence, this issue is in the generation of secure keys and not in RC4 itself. We also notice that some research (e.g., References [15,16]) report attacks on the RC4 stream cipher recently. They indicate that new attacks work even if the first 256 byte of the output remains unused. According to our studies, the new attacks require weak initialization state or require that the sum of the first two bytes of the encrypted message has a prescribed value. In the proposed protocol, the initialization state is not used and it is also not applied to have the sum of the first two bytes of the encrypted message has a prescribed value.

We present an important RC4 feature on which the proposed protocol is based on. We call this feature as the forward and backward property of RC4 states. This feature shows that any RC4 state can be forward to a new RC4 state by PRGA and backward to a previous RC4 state by IPRGA (PRGA and IPRGA are depicted in Algorithm 1). We have proved this feature in Theorem 1 and use this feature in the proposed protocol to avoid the frequent key generation.

PRGA(*S*)
Generation loop:
$i \leftarrow (i+1) \bmod 256$
$j \leftarrow (j + S[i]) \bmod 256$
$S[i] \leftrightarrow S[j]$
Output $z \leftarrow S[(S[i] + S[j]) \bmod 256]$

IPRGA(*S*, *i*, *j*)
Generation loop:
$S[i] \leftrightarrow S[j]$
$j \leftarrow (j - S[i] + 256) \bmod 256$
$i \leftarrow (i-1) \bmod 256$

Algorithm 1: *The operations of PRGA and IPRGA.*

Theorem 1: *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 (same for* $IPRGA^k$*) and IPRGA is the reverse algorithm of PRGA but do not generate a stream key. S is an RC4 state and i and j are the RC4 indexes.*

Proof. *The proof is conducted by induction on integer r.*

Base Case: *It is true when* $r = 0$*:* $IPRGA^0$ $(S_n, i_n, j_n) = (S_n, i_n, j_n)$.

Inductive Step: Assume that for any $r \le k$ we have IPRGA^k $(S_n, i_n, j_n) = (S_{n-k}, i_{n-k}, j_{n-k})$.

Consider case of $r = k + 1$:

IPRGA^{k+1} $(S_n, i_n, j_n) = \text{IPRGA}$ $(\text{IPRGA}^k$ $(S_n, i_n, j_n)) = \text{IPRGA}$ $(S_{n-k}, i_{n-k}, j_{n-k})$.

Let IPRGA $(S_{n-k}, i_{n-k}, j_{n-k}) = (S^*, i^*, j^*)$.

According to PRGA and our notation, we have

$\text{PRGA}(S_{n-k-1}, i_{n-k-1}, j_{n-k-1}) = (S_{n-k}, i_{n-k}, j_{n-k})$ where $i_{n-k} = i_{n-k-1} + 1$ and $j_{n-k} = j_{n-k-1} + S_{n-k-1}[i_{n-k-1} + 1]$

$S_{n-k}[m] = S_{n-k-1}[m]$ where $m \neq i_{n-k-1}$ and $m \neq j_{n-k-1} + S_{n-k-1}[i_{n-k-1} + 1]$

$S_{n-k}[i_{n-k-1} + 1] = S_{n-k-1}[j_{n-k-1} + S_{n-k-1}[i_{n-k-1} + 1]]$

$S_{n-k}[j_{n-k-1} + S_{n-k-1}[i_{n-k-1} + 1]] = S_{n-\underline{k}-1}[i_{n-k-1} + 1]$.

According to IPRGA $(S_{n-k}, i_{n-k}, j_{n-k}) = (S^*, i^*, j^*)$,

we have $S^*[m] = S_{n-k}[m] = S_{n-k-1}[m]$ where $m \neq i_{n-k-1}$ and $m \neq j_{n-k-1} + S_{n-k-1}[i_{n-k-1} + 1]$

and $S^*[i_{n-k-1} + 1] = S_{n-k}[j_{n-k-1} + S_{n-k-1}[i_{n-k-1} + 1]] = S_{n-k-1}[i_{n-k-1} + 1]$

$S^*[j_{n-k-1} + S_{n-k-1}[i_{n-k-1} + 1]] = S_{n-k}[i_{n-k-1} + 1] = S_{n-k-1}[j_{n-k-1} + S_{n-k-1}[i_{n-k-1} + 1]$

Thus $S^* = S_{n-k-1}$, $j^* = j_{n-k} - S^*[i_{n-k}] = j_{n-k-1} + S_{n-k-1}[i_{n-k-1} + 1] - S_{n-k-1}[i_{n-k-1} + 1] = j_{n-k-1}$,

$$i^* = i_{n-k} - 1 = i_{n-k-1}$$

Therefore, we have $(S^*, i^*, j^*) = (S_{n-k-1}, i_{n-k-1}, j_{n-k-1})$.

That is, for any $n > r > 0$, we have: IPRGA^r $(S_n, i_n, j_n) = (S_{n-r}, i_{n-r}, j_{n-r})$. ■

The forward and backward property of RC4 states is important in real-time decryption with networks. When using the stream cipher, if the decryption algorithm does not support the reversible feature, the receiver must receive all packets from the sender before decrypting the message. By using this, the receiver can decrypt each packet received in real time. If the packets arrive in a different order, we can move the RC4 state forward or backward to decrypt a certain packet.

DOI: 10.1002/sec

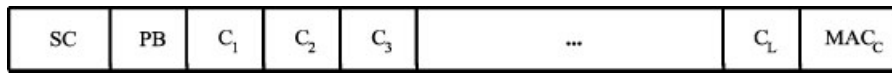

Fig. 1. Encrypted fixed-length data packet.

3. An RC4-based Lightweight Security Protocol

We present the proposed protocol in this section. For clarity, we declare the terminologies and notations first. After the presentation of the proposed protocol, we show its support to broadcast and multicast.

The proposed protocol requires the fixed data length for every data packet. Through checking the Sequence Counter number of an incoming fixed-length data packet and calculating from its corresponding RC4 state, the receiver can obtain the correct RC4 state to decrypt this packet based on the forward and backward property. A fixed-length data packet contains a sequence counter value, a position bit, a data segment, and a MAC checksum. Figure 1 indicates an encrypted fixed-length data packet. The undeclared terminologies in this paragraph are as below.

RC4 State: RC4 state includes a permutation of 256 one-byte elements *S* and two one-byte indexes *i* and *j*. We use (*S, i, j*) to denote an RC4 state.

Corresponding RC4 State (*CRS*): A certain RC4 state generates a certain stream key on RC4, therefore the same RC4 state needs to be used for both encryption and decryption for a given message. The proposed protocol requires each node which keeps an RC4 state in its memory and we call it CRS. CRS is updated after the encryption or decryption of a fixed-length data packet.

Offset (*O*): Offset is an integer to indicate the number of rounds of PRGA takes place after obtaining the initial permutation state S_0 from KSA, but before encrypting or decrypting message. It carries out only after the base key changes. The purpose is to discard the offset number of bytes at the beginning to avoid statistical bias in the distribution of the initial bytes of RC4 streams [11] to strengthen RC4.

Sequence Counter (*SC*): SC is a number which initially is zero and increases by one for each interval. The length of the sequence counter varies depending on different requirement and application. The sender and each receiver have their own SC. For the sender, SC is increased by one for each new fixed-length data packet. For each receiver, SC determines the order of the received fixed-length data packet.

Position Bit (*PB*): The position bit is one-bit segment for a fixed-length data packet to indicate whether it is the last fixed-length data packet.

In the following, we present the proposed protocol. During the handshaking phase (e.g., initial transmission or the base key change), the sender and receivers share an RC4 base key, a MAC key and an offset value O. After acquiring the information during the handshaking phase, both sides apply KSA to generate an initial permutation of the RC4 state S_0, and then apply offset rounds of PRGA to S_0 to generate a new RC4 state as CRS for key stream generation. At the same time, both sides set their SC to zero ($SC = 0$).

Note that, different from WEP and WPA 1.0 which reinitialize the RC4 state for every packet and generate the cipher stream from the initialized RC4 state, the proposed protocol does not reinitialize RC4 states frequently. Instead, the proposed protocol maintains the same RC4 base key for a duration known to each communicating node. This requires the initialization of the RC4 state (KSA) to be carried out only when the base key changes. Also, it is unnecessary to update the offset value O often.

Figure 2 depicts the operations executed by the sender. After the three steps operations, the encrypted fixed-length data packets are produced from an input plaintext message. The details of each step are presented below:

(1) The sender divides the input plaintext message into contiguous fixed-length data segments and assigns SC and PB to each of them. If there are not enough data in the data segment of the last data packet, pad end tag followed by 0 or random numbers (TR). Adding random numbers requires more resource, but could enhance the security.
(2) The sender calculates the MAC checksum by inputting SC, PB, unencrypted data segment, and MAC key, and then fills the MAC checksum into the MAC segment. Note that, 0 is used to pad for MAC generation purpose if there are not enough data in the last data packet.
(3) The sender produces the encrypted fixed-length data packets through encrypting data segment and MAC checksum. After the encryption, the sender updates its SC and CRS.

Figure 3 depicts the operations executed by the receiver. After the three steps, the input plaintext message is recovered from the corresponding fixed-length

DOI: 10.1002/sec

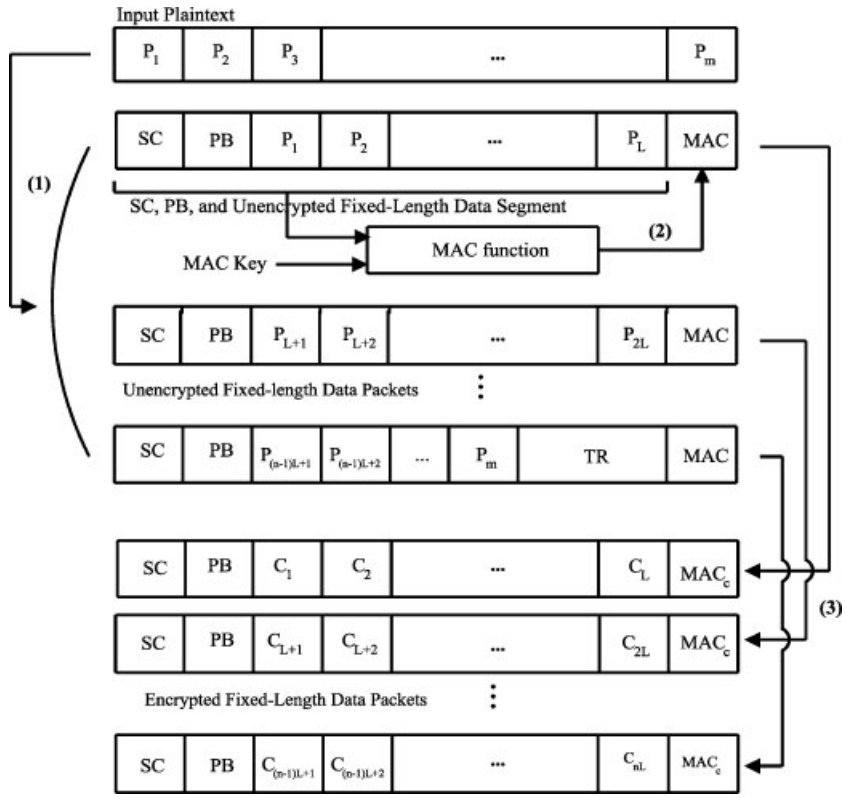

Fig. 2. The operations executed by the sender.

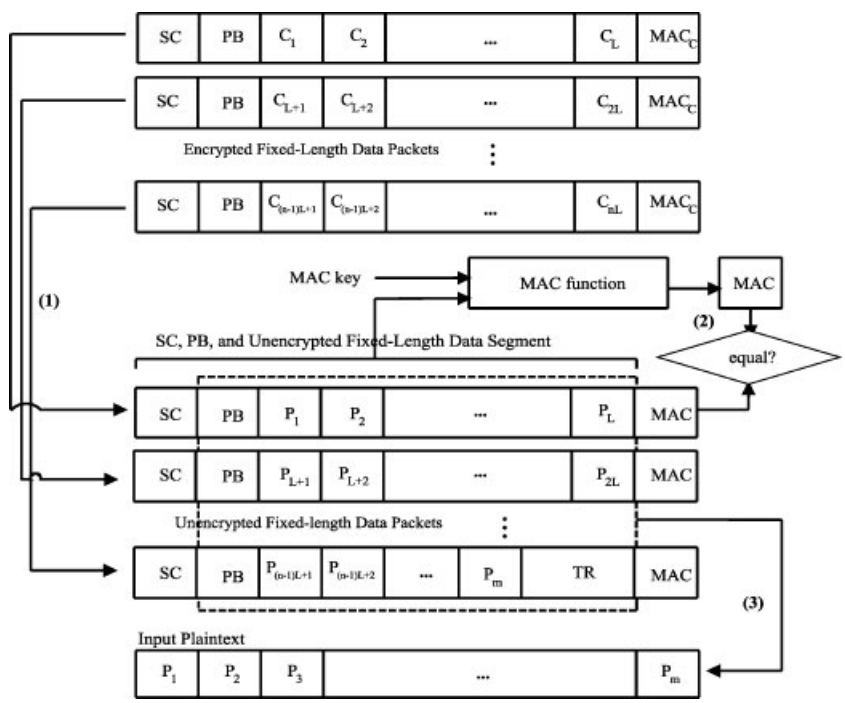

Fig. 3. The operations executed by the receiver.

DOI: 10.1002/sec

data packets. The details of each step are presented below:

(1) By checking the SC value of the incoming fixed-length data packet, the receiver can find the right CRS based on the forward and backward property. The receiver then calculates the key stream to decrypt the encrypted data segment and MAC checksum.
(2) The receiver verifies data packets by re-computing the MAC checksum.
(3) After all of the fixed-length data packets which associate with an input plaintext message have been decrypted and their MAC checksum are verified, the receiver restores them to an input plaintext message. After that, the receiver updates its SC and CRS.

Broadcast is an important network service to provide efficient delivery of data from a source to a group of recipients and it is very useful for data delivery amongst a large group of participants. For WSNs, broadcast is one of the key applications which can be operated for software updates, network queries, command dissemination, etc.

The main difference of the approach between one-to-one communications and broadcast is that the delayed or lost packets need to be addressed in the broadcast approach. The proposed protocol allows delayed and lost packets. The main idea is that the receiver can check the order of the incoming packet from SC and then retrieve CRS by the forward and backward property to decrypt the delayed or lost packet.

Figure 4 is an example of WSN. We use it to depict how the proposed protocol works for the delayed packets to support secure broadcast. B is a base station and S_1, S_2, S_3, and S_4 are four sensor nodes. B sends four data packets P_1, P_2, P_3, and P_4 to S_1, S_2, S_3, and S_4 and the right order of the packets is $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$. For some reason, packet P_2 is delayed to S_4 and P_3 is delayed to S_2. The process is described as follows:

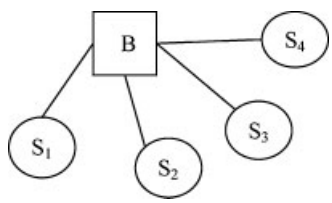

Fig. 4. An example of WSNs.

(1) B sends P_1, P_2, P_3, and P_4 to S_1, S_2, S_3, and S_4. The sending order is $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$.
(2) S_1 and S_3 receive four packets in right order, and then they decrypt the packets regularly.
(3) S_2 receives packets in the order of $P_1 \rightarrow P_2 \rightarrow P_4 \rightarrow P_3$. S_2 decrypts P_1 and P_2 first. Since P_4 is received prior to P_3, S_2 calculates the CRS to decrypt P_4 first and then calculates the CRS to decrypt P_3.
(4) S_4 receives packets in the order of $P_1 \rightarrow P_3 \rightarrow P_4 \rightarrow P_2$. S_4 decrypts P_1 first and then calculates the CRS to decrypt P_3. After that, S_4 decrypts P_4 in regular process and then calculates the CRS to decrypt P_2.

In the case of packet missing, a potential missing packet list (PMPL) is used to keep lost packets information for each receiver. We still use Figure 4 as an example to indicate how the proposed protocol works for the lost packets to support secure broadcast in WSNs. B sends four data packets P_1, P_2, P_3, and P_4 to S_1, S_2, S_3, and S_4 and the right order of the packets is $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$. For some reason, packet P_2 is lost to S_4 and P_3 is lost to S_2.

(1) B sends P_1, P_2, P_3, and P_4 to S_1, S_2, S_3, and S_4. The sending order is $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$.
(2) S_1 and S_3 receive four packets in right order, and then they decrypt the packets regularly.
(3) S_2 receives packets in the order of $P_1 \rightarrow P_2 \rightarrow P_4$ and P_3 is lost. S_2 decrypts P_1 and P_2 first. Since P_4 is received prior to P_3, S_2 adds P_3 to its PMPL and then calculates the CRS to decrypt P_4.
(4) S_4 receives packets in the order of $P_1 \rightarrow P_3 \rightarrow P_4$ and P_2 is lost. S_4 decrypts P_1 first. Since P_3 is received prior to P_2, S_4 adds P_2 to its PMPL and then calculates the CRS to decrypt P_3 and then P_4.
(5) After P_3 stays in S_2's PMPL and P_2 stays in S_4's PMPL longer than a period of time, S_2 and S_4 send requests to B for resending P_3 and P_2, respectively.
(6) B receives the requests and holds them for a period of time, and then processes all requests to retrieve the lost packet(s) by CRS and then resend them with resending indication.
(7) S_2 receives the corresponding lost packet P_3 and then calculate the CRS to decrypt P_3. Also, S_4 receives the corresponding lost packet P_2 and then calculate the CRS to decrypt P_2.

In general, in order to reduce the computation overheads to calculate right CRS for missing packets on the sender side, SC numbers of the missing packets are

DOI: 10.1002/sec

expected to be stored by the sender and then resent it in the stored order. The above examples show that the proposed protocol allows packet lost or delayed. Therefore, the proposed protocol supports secure broadcast in WSNs. Moreover, the proposed protocol can also be applied for secure multicast when associates with proper key management scheme.

4. Analysis and Implementation

As a security protocol, the first analysis we present is the security performance analysis. The proposed protocol is based on an improved RC4 algorithm. This security analysis can be made in the view of the security analysis of RC4 as well as the improvement. We have provided a detailed security analysis for RC4 algorithm in Section 2 and indicated that RC4 itself is secure. Following we present the analysis on the improvement, which includes offset and the backward property.

The forward and backward property of RC4 states is an intrinsic property of RC4 algorithm. We only use it and have no change to the original RC4 algorithm. For this reason it obviously would not reduce the security strength. Offset is used to discard the first few bytes of the key stream, and makes data encryption or decryption starting at a later position. The purpose of applying offset is to avoid the weaknesses of the initial RC4 states [8]. The inclusion of offset is not a fundamental change to RC4 and actually improves the security performance.

Based on the analysis above, we believe that the proposed protocol applies RC4 in a novel way which makes effective use of the strengths of RC4 and reduces its weaknesses. We simulated the protocol and different attacks, and proved that it achieves semantic security and defends against most popular attack effectively, such as replay attacks and modified packet attacks. Semantic security is achieved since different key stream are used to encrypt different messages with same content at different time. Since sequence number and MAC checksum are used, the proposed protocol can defend replay attacks and modified packet attacks. Furthermore, the proposed protocol achieves main security properties which a basic security scheme requires:

- Data confidentiality is achieved through encrypting data by RC4 with the improvement.
- Data authentication is achieved through checking MAC.
- Data integrity is achieved through data authentication, which is stronger than just checking data integrity.
- Data freshness is achieved through the monotonically increasing value of the SC for each data packet.

With respect to the performance, the proposed protocol runs efficiently benefiting from RC4 and its forward and backward property. As we indicate in Section 2, RC4 is remarkably simple and fast and the overhead is low. All operations in the proposed protocol are swap, XOR, and simple addition and subtraction, and the unit is 8-bit. Furthermore, the proposed protocol uses RC4 more efficiently. By using the forward and backward property, the proposed protocol does not require frequent key initialization. This translates directly to simplification and power saving, as well as improved performance over the ordinary RC4 based schemes. For example, if the length of the data is 256 bytes, since KSA is not required for every new packet, the proposed encryption/decryption process requires half operations and the half time compared to the ordinary RC4 based scheme (e.g., WEP).

Tables I and II indicate the comparisons of the execution times among AES-based scheme, RC5-based scheme, WEP (ordinary RC4-based scheme), skipjack-based scheme, and the proposed protocol. The comparisons are based on an 8-bit CPU. In Table I, t_1 is the time required for a logic operation or a simple arithmetic operation, t_2 is the time required for one time memory access. For simplicity, we assume $t = t_2 = 2t_1$ in Table II. The comparisons indicate that the proposed protocol runs much faster than others.

Table I. The comparisons of the execution times on an 8-bit CPU (t_1 is the required time for a logic operation or a simple arithmetic operation, and t_2 is the required time for one time memory access).

Data length	64 bytes	128 bytes	256 bytes	512 bytes	1024 bytes
AES-based	$5824t_1 + 6272t_2$	$11\,648t_1 + 12\,544t_2$	$23\,296t_1 + 25\,088t_2$	$46\,592t_1 + 50\,176t_2$	$93\,184t_1 + 100\,352t_2$
WEP	$1024t_1 + 2816t_2$	$1280t_1 + 3328t_2$	$1792t_1 + 4352t_2$	$2816t_1 + 6400t_2$	$4864t_1 + 10\,496t_2$
RC5-based	$14\,784t_1 + 832t_2$	$29\,568t_1 + 1664t_2$	$59\,136t_1 + 3328t_2$	$118\,272t_1 + 6656t_2$	$236\,544t_1 + 13\,312t_2$
Skipjack-based	$1280t_1 + 1664t_2$	$2560t_1 + 3328t_2$	$5120t_1 + 6656t_2$	$10\,240t_1 + 13\,312t_2$	$20\,480t_1 + 26\,624t_2$
Proposed protocol	$256t_1 + 512t_2$	$512t_1 + 1024t_2$	$1024t_1 + 2048t_2$	$2048t_1 + 4096t_2$	$4096t_1 + 8192t_2$

DOI: 10.1002/sec

Table II. The comparisons of the execution times on an 8-bit CPU (assume $t = t_2 = 2t_1$).

Data length	64 bytes (t)	128 bytes (t)	256 bytes (t)	512 bytes (t)	1024 bytes (t)
AES-based	9184	18 368	36 736	73 472	146 944
WEP	3328	3968	5248	7808	12 928
RC5-based	8224	16 448	32 896	65 792	131 584
Skipjack-based	2304	4608	9216	18 432	36 864
Proposed protocol	640	1280	2560	5120	10 240

The proposed protocol has been simulated and the result shows it works as expected. One sender and four receivers are included in this simulation. We list the conclusion of this simulation below.

(1) Compared with the ordinary RC4-based scheme, the proposed protocol is almost twice faster than the ordinary.
(2) The proposed protocol has the ability to against a number of possible attacks including acknowledge attack, replay attack, and modified packet attack.
(3) The proposed protocol works as expected to handle the circumstances of delayed or lost packets. The secure broadcast works correctly and smoothly.

Since the proposed protocol is state-based, each node must maintain an exact copy of the RC4 state so that the next set of encryption bytes can be matched. If a node gets out of synchronization by even a single byte, the decryption is lost. The challenge, therefore, is to verify that each node keep the same place in the encryption and decryption. Through the simulation, we verified that each node works well and each part of the communications keeps the same place in encryption and decryption.

5. Conclusion and Future Work

This paper focuses on the design of a lightweight protocol for providing a secure data transmission on resource constrained devices. At the beginning of this paper, we have proved that the RC4 state is reversible. As far as we know, the proposed protocol is the first to use this feature in secure data transmission.

The proposed protocol achieves data confidentiality, data authentication, data integrity, and data freshness with low overhead and simple operation. It also achieves semantic security and has the ability to defend a number of popular attacks such as replay attacks and modified packet attacks. By using offset, the proposed protocol makes effective use of the strengths of RC4, and reduces its weaknesses. Benefiting from the forward and backward property of RC4, the proposed protocol eliminates the requirement of frequent key renewal which is a special limitation of the stream ciphers.

In consideration of the resource constrained nature of the resource constrained devices, the proposed protocol is competitive with well-known security schemes due to the efficiency. Our simulation shows that the proposed protocol works as expected to deal with the circumstances of delayed or lost packet and the secure broadcast works correctly and smoothly.

Based on the above discussion, we believe that the proposed protocol is an ideal solution to provide secure data transmission for resource constrained devices. This protocol not only applies to one-to-one communications, but also works for the broadcasting and multicasting.

Further research work will focus on the topic of key management and distribution and sender verification. Key management in wireless networks is a more complicated problem because of the mobility of group members. How to assign and distribute the base key to all members efficiently but not to disclose it to others is a practical topic. Also, how to verify a sender and how to confirm that data are sent by the sender is also the related topic.

References

1. Kaps JP, Gaubatz G, Sunar B. Cryptography on a Speck of Dust. *IEEE Computer Magazine* 2007; **40**(2): 38–44.
2. Saraogi and Mayank. Security in Wireless Sensor Networks, Spring, 2005, www.cs.utk.edu/ saraogi/594paper.pdf
3. Roman R, Zhou J, Lopez J. On the Security of Wireless Sensor Networks. In *Proceedings of 2005 ICCSA Workshop on Internet Communications Security*, LNCS 3482, Singapur, May 2005, pp. 681–690.
4. Perrig A, Stankovic J, Wagner D. Security in wireless sensor networks. *Communications of the ACM* 2004; **47**(6): 53–57.
5. Römer K, Mattern F. The Design Space of Wireless Sensor Networks. *IEEE Wireless Communications* 2004; **11**(6): 54–61.
6. Haenselmann T. *GFDL Wireless Sensor Network Textbook* (retrieved on February 10, 2008), www.informatik.uni-mannheim.de/ haensel/sn_book

DOI: 10.1002/sec

7. Perrig A, Szewczyk R, Wen V, Culler D, Tygar JD. SPINS: Security Protocols for Sensor Networks. *Wireless Networks* 2002; **8**(5): 521–534.
8. Karlof C, Sastry N, Wagner D. TinySec: a link layer security architecture for wireless sensor networks. ACM SenSys, 2004.
9. Mantin I. Analysis of the Stream Cipher RC4. *Master's thesis*, The Weizmann Institute of Science, Nov. 2001.
10. Stallings W. Cryptography and Network Security: Principles and Practice, (4th edn), Prentice Hall: Upper Saddle River, NJ, 2006.
11. Mitchell S, Srinivasan K. State Based Key Hop Protocol: A Lightweight Security Protocol for Wireless Networks. In *Proceedings of the 1st ACM international workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, 2004; 112–118.
12. Stubblefield A, loannidis J, Rubin AD. Using the Fluhrer, Mantin, and Shamir Attack to Break WEP. *AT&T Labs Technical Report*, 2001.
13. Fluhrer S, Mantin I, Shamir A. Weakness in the Key Scheduling Algorithm of RC4. In *Proceedings of Workshop in Selected Areas of Cryptography*, 2001.
14. WEP Fix using RC4 Fast Packet Keying. *RSA Security Technical Notes*, 2002; www.rsasecurity.com/rsalabs/technotes/wep-fix.html
15. Klein A. Attacks on the RC4 stream cipher. *Design, Codes, Cryptography* 2008; **48**: 269–286.
16. Vaudenay S, Vuagnoux M. Passive-only key recovery attacks on RC4. *Lecture Notes in Computer Science* 2007; **4876**: 344–359.

DOI: 10.1002/sec