SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2016; **9**:3314–3323
Published online 7 July 2016 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1536

RESEARCH ARTICLE

Energy efficient secure communication architecture for wireless sensor network

Satyajit Mondal[1]*, Sraban Kumar Mohanty[1] and Sukumar Nandi[2]

[1] Department of Computer Science and Engineering, Indian Institute of Information Technology Design and Manufacturing, Jabalpur, India
[2] Department of Computer Science and Engineering, Indian Institute of Technology, Guwahati, India

ABSTRACT

The wireless sensor network (WSN) is more vulnerable than the wired network because it is relatively easy for an adversary to eavesdrop, insert, alter, and intercept the message communicated in the network. Sensor nodes have very limited power and memory. While designing the secure communication architectures, the main goal is to design energy and memory efficient protocols. Most of the well-known communication architectures like TinySec, LLSP, and MiniSec compromise with the security issues because of energy and memory constraints of sensor nodes. In WSN, communication consumes more energy than computation. By minimizing communication overhead, the energy consumption can be reduced. We propose a new "energy efficient secure communication architecture" called EESCA for WSN, to make the communication secure. Along with authentication, confidentiality and integrity, the proposed architecture provides strong replay protection, and data freshness as permutation and substitution are used to generate massage authentication code. EESCA is robust against heavy packet loss because reset or resynchronization of counter is not required. The proposed scheme is energy efficient as the security packet overhead is reduced by 1 byte over the best-known secure communication architectures. The efficacy of EESCA is shown through theoretical and experimental analysis. Copyright © 2016 John Wiley & Sons, Ltd.

KEYWORDS
sensor network security; energy efficiency; strong replay protection

***Correspondence**
Satyajit Mondal, Department of Computer Science and Engineering, Indian Institute of Information and Technology Design and Manufacturing, Jabalpur, India.
E-mail: satyajit.mondal@iiitdmj.ac.in

1. INTRODUCTION

Wireless sensor network (WSN) is gaining popularity day by day because of its low cost, ease of deployment, and wider applicability. The WSN is used in situations where terrain, climate, and other environmental constraints hinder the setting up of conventional network [1,2].

The major issues that affect the performance and applicability of the WSN are scalability, memory and power limitation, quality of service, security, and many more.

Wireless sensor networks are deployed to collect sensitive information. It is more vulnerable than wired network because of lack of physical boundary [3]. It is relatively easy for the adversary to insert, capture, eavesdrop, or tamper a mote, so security is an important aspect of WSN. Another important security attack in the context of energy constraint of sensor nodes is replay attack. In replay attack, an attacker intercepts a genuine packet and later replays the packet, which has serious implications in WSN as a malicious node can employ this attack to drain the energy of the routing nodes. This attack can be detected by checking the counter or sequence number of the current packet with the last received packet.

SPINS [4], ZigBee [5], TinySec [6], energy efficient link layer architecture for WSP (LLSP) [7] and MiniSec [8] are the popular secure architecture for WSN designed over TinyOS. Due to low energy resource and low memory of wireless sensor nodes, the existing secure communication architecture compromises with security requirements and energy consumption.

To provide security in secure communication architectures, extra fields called security overhead are added in the packet header like source address (SRC), counter (CTR), and message authentication code (MAC) . Also, some header fields like Group and cyclic redundancy check (CRC) are removed. All these well-known secure

communication architectures reduce energy consumption by reducing the size of security overhead. Security overhead of SPINS is 8 bytes, TinySec is 5 bytes, and MiniSec and LLSP are 3 bytes over TinyOS. However, counter resynchronization adds extra communication costs in SPINS. TinySec does not provide replay protection. If the packet drop between two received packets is more than eight then counter needs to be reset, which adds extra communication cost in MiniSec-U. MiniSec-B uses boom filter to reduce memory requirement, but it increases the computational cost. MAC is generated for each packet loss in LLSP, which adds extra computational cost.

In this paper, we propose a link layer communication architecture called EESCA for WSN. The proposed architecture provides strong security using symmetric key crypto system. Especially, EESCA provides strong replay protection because we use permutation and substitution to generate MAC. EESCA is robust against heavy packet loss because reset or resynchronization of counter as done in SPINS or MiniSec-U is not required. Also, MAC is computed only once at sender and receiver side. The security overhead is reduced by 1 byte over LLSP and MiniSec and 3 bytes over TinySec making EESCA energy efficient. The efficacy in terms of energy efficiency of our proposed scheme is shown by both theoretical and experimental analysis by emulating a sensor node using AVRORA simulator [9].

The rest of the paper is organized as follows. Section 2 discusses on background and related works. Proposed communication architecture is explained in Section 3, and Section 4 evaluates the security and performance of the architecture. Concluding remarks and future work are given in Section 5.

2. BACKGROUND AND RELATED WORK

2.1. Security requirements for wireless sensor network environment

Most of the WSN applications like military and medical deal with sensitive information, so an attacker poses many security threats like eavesdropping, node capture, message alteration, and replay [10,11]. To secure the communication over WSN against different types of attack, it is very much essential for WSN to have high security properties like authentication, confidentiality, integrity, replay protection, and data freshness. These security requirements are briefly discussed in the following.

2.1.1. Confidentiality.

Confidentiality protects sensitive data of any sensor node from discloser to any unauthorized node. To provide confidentiality, sender and receiver share a common secret key. Sender encrypts data and generates a cipher text. Receiver decrypts the cipher text using the same encryption key. The data encryption standard (DES), advanced encryption standard (AES), and Skipjack are the popular symmetric key encryption algorithm used in WSN to provide confidentiality. Only encryption may not provide high security, the same encryption algorithm and key is used to encrypt the same data, then same cipher text is generated every time. Solution of this problem is semantic security [4], which generates different cipher text even though data and key are same. This is achieved using block cipher modes of operation.

2.1.2. Authentication.

By authentication, receiver can confirm that packet has come from the claimed sender. Authentication is important in the context of WSN as adversary can inject fraudulent messages to the network, which would degrade the performance. If the data are used for decision-making process then the node must verify the authenticity of the source. To provide authentication, sender and receiver share a common secret key and generate massage authentication code (MAC) using it. After receiving the packet, the receiver compares with the received MAC. If it matches, then the packet has come from a claimed sender. The block ciphers like AES, DES, and RC5 are used in cipher block chaining (CBC) mode to create MAC(CBC-MAC).

2.1.3. Integrity.

Integrity confirms that received packet is not altered during the transmission by any adversary. Data integrity is achieved using hash function, MAC, or CRC.

2.1.4. Data freshness and replay protection.

Data freshness ensures that accepted packets are fresh, and they are not replayed. To provide replay protection, any older genuine packet must be rejected. Bloom filters or counter table is a common technique used in WSN to provide replay protection.

2.2. Cipher block chaining mode of operation

Cipher block chaining mode encrypts the plain text block by block, and cipher text will be change depending on random initial vector(IV), if same plain text is repeated [12]. In this mode, first block of the plain text (P_1) is XOR with IV and then encrypted to generate the first block of the cipher text (C_i) as shown in equation (1).

$$C_1 = E(K, (IV \oplus P_1)) \tag{1}$$

Then in each j^{th} iteration, cipher text block C_{j-1} is used in the place of IV as the feedback for the next block C_j. The process is repeated as follows:

$$C_j = E\left(K, \left(C_{j-1} \oplus P_j\right)\right) \tag{2}$$

Now, all the cipher text blocks are sent as payload. To decipher the ciphertext, first decrypt the last block of the

cipher text and then XOR with the previous cipher text block as shown in equation (3), and the process is repeated.

$$P_j = C_{j-1} \oplus D(K, C_j) \tag{3}$$

The decryption of the first block is performed as shown in equation (4).

$$P_1 = IV \oplus D(K, C_1) \tag{4}$$

The CBC mode of operation is illustrated in Figure 1

2.3. Secure communication architectures for wireless sensor network

In TinyOS communication, each packet is divided into three parts: packet header, payload, and packet footer. The size of the header is of 5 bytes, which comprises of 2-bytes destination address (DST), 1-byte active message(AM), 1-byte length (LEN), and 1-byte group ID (Group) field. Payload contains data that varies from 0 to 29 bytes, and packet footer contains 2 bytes CRC field [13]. The packet format of TinyOS is shown in Figure 2. SPINS [4], TinySec [6], energy efficient link layer architecture for WSN (LLSP) [7], and MiniSec [8] are the popular secure architecture for WSN designed over TinyOS. Due to low energy resource and low memory of wireless sensor nodes, the existing secure communication architecture compromises with security requirements and energy consumption. Here, we briefly describe some of the popular communication architectures.

2.3.1. SPINS.

Security protocols for sensor network (SPINS) have two secure building blocks namely SNEP and μ-TESLA. Confidentiality, two party authentication, and data freshness are provided by SNEP, and μ-TESLA provides authenticated broadcast. Here, CBC-MAC is used to provide authentication, and RC5 is used in counter mode to provide confidentiality. MAC is generated on counter and encrypted data. Counter is incremented by one for each packet. This provides replay protection as for each MAC, a new counter is used. In WSN, packet may obtain lost, which would lead to counter inconsistency at both ends. So counter resynchronization is required. The counter resynchronization is achieved as follows:

In counter resynchronization, sender sends its counter (CTR_A) to receiver. Receiver generates MAC on CTR_A and its won counter (CTR_B) and sends CTR_B and MAC. Sender verifies it by generating MAC on CTR_A and CTR_B using same symmetric key and resends the MAC to the receiver. The resynchronization is completed if the received MAC is same as the sent MAC.

SPINS has 8 bytes security overhead over TinyOS packet format where 8 bytes MAC is sent along with encrypted data. 2 bytes CRC is removed to reduce packet overhead. In addition to this counter resynchronization for each packet loss also requires extra communication because of which SPINS is energy inefficient [8]. Moreover, although SPINS provides a secure communication infrastructure for WSNs, but to the best of our knowledge, detailed description of frame format of SPINS is neither fully specified nor implemented [6,7].

2.3.2. TinySec.

TinySec is a link layer architecture for WSN, which reduces energy consumption over SPINS. TinySec has two modes of operations Authentication only and authentication–encryption (AE). Authentication only provides authentication and integrity whereas AE provides confidentiality along with authentication. Here, CBC-MAC is used for authentication and integrity, and skipjack-CBC is used for confidentiality. Packet format of TinySec is shown in Figure 3. For semantic security, TinySec uses 16-bit counter and 8-byte IV. IV is generated by concatenating 2-bytes DST, 1-byte AM, 1-byte LEN, 2 bytes source address (SRC) and 2 bytes counter (CTR) as shown in equation (5). TinySec sends IV in packet header and 4 bytes MAC in the place of CRC in packet footer. The group id and CRC are not added in the packet.

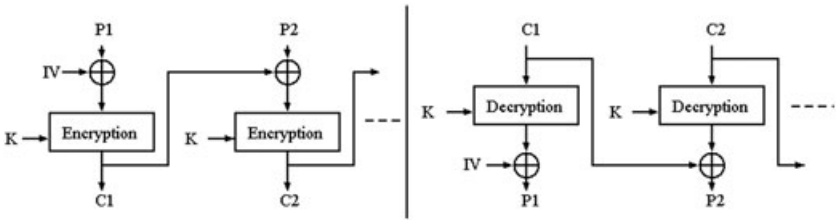

Figure 1. Cipher block chaining mode of operation.

DST (2)	AM (1)	LEN (1)	Group (1)	Data (0-29)	CRC (2)

Figure 2. Packet format of TinyOS (size of fields are in bytes).

Throughout the paper, the numeric value in the subscript of a packet field (e.g., DST_2) shall denote the number of bytes.

$$IV_{TinySec} = DST_2 \| AM_1 \| LEN_1 \| SRC_2 \| CTR_2 \quad (5)$$

The frames of both Authentication and AE are illustrated in Figure 3. Clearly, packet overhead of TinySec-AE is 5 bytes over TinyOS. Also, TinySec does not provide replay protection. It uses only 16 bits counter to provide semantic security; hence, strong security is compromised.

2.3.3. LLSP.

LLSP is also a secure link layer architecture for WSN developed over TinyOS. LLSP overcomes the security issues of TinySec by incorporating replay protection. Here, AES-CBC is used for confidentiality and CBC-MAC for authentication and integrity. MAC is generated on IV and encrypted data. IV is generated as follows:

$$IV_{LLSP} = DST_2 \| AM_1 \| LEN_1 \| SRC_2 \| CTR_4 \quad (6)$$

This provides replay protection. Here, the receiver remembers the counter value of last received packet. After receiving a packet, receiver increments the stored counter and generates MAC. If the received MAC matches with the computed MAC then packet is accepted. LLSP generates MAC on IV and encrypted data. Here, CTR is not sent. That is why the security overhead is reduced by 2 byes over TinySec. LLSP packet format is shown in the Figure 4.

If p is the packet loss in the WSN between two received packets for a receiver, then LLSP computes MAC p times more to find the correct CTR, which was used to generate the received MAC. Here, the security overhead is 3 bytes, which is 2 bytes less than TinySec.

2.3.4. MiniSec.

MiniSec is a network layer secure architecture that has two operating modes: MiniSec-U for unicast and MiniSec-B for broadcast. MiniSec provides confidentiality by encrypting data using Skipjack in output feedback mode. Also, MAC is generated on header and data using the same procedure at the same time, that is, output feedback-skipjack generates the cipher text and MAC as $E(K, (nonce, (data \parallel header)))$. Here, MAC is the last 4 bytes of the last output block. In MiniSec, nonce is a non-repeating 64-bits counter, which is used to provide semantic security. Packet header consists of DST, AM, LEN, and SRC. Nonce is not sent with the packet. MiniSec-U sends last 3 bits of nonce in LEN field. MiniSec-B send last 8 bits of nonce in LEN and SRC fields. Packet format of MiniSec-B is shown in Figure 5. Base station maintains a counter table, which stores the counter of the last received packet to decrypt the message and to verify the MAC.

MiniSec-U provides weak replay protection. The receiving node checks the freshness of the received packet by comparing the last 3 bits of the counter of the new packet with the last received packet. If last 3 bits of the counter is greater, then packet is fresh. To provide strong

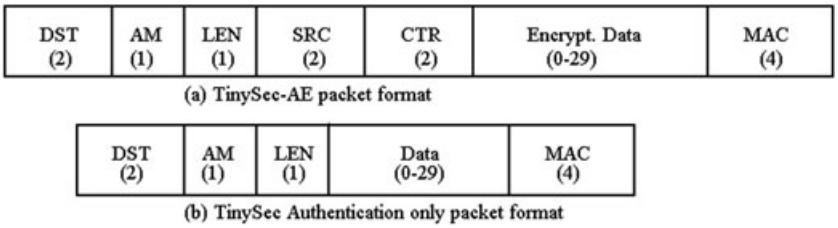

Figure 3. Packet format of TinySec (size of fields are in bytes): (a) TinySec-AE packet format and (b) TinySec authentication only packet format.

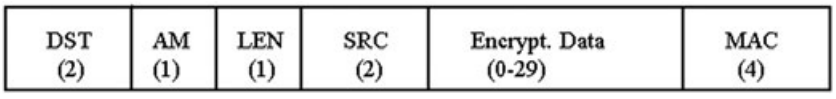

Figure 4. Packet format of LLSP (size of fields are in bytes).

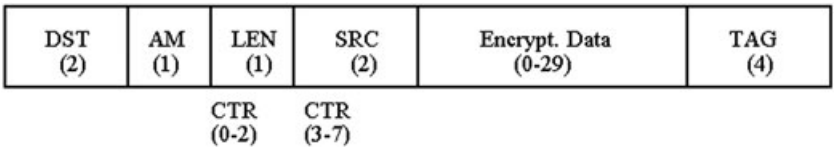

Figure 5. Packet format of MiniSec-B (size of fields are in bytes).

replay protection, MiniSec-B uses Bloom filter, which also reduces memory usage.

The level of security is compromised because one key is used to generate both MAC and cipher text. MiniSec does not check authentication in routing nodes. Again, if routing nodes have the key to verify MAC to remove unauthorized packets from WSN, then using node capture attack, an attacker may obtain cryptographic information. In this way, the confidentiality is compromised. If authentication is not verified in the routing nodes, then routing nodes will loose their energy by sending unauthorized packets.

Other secure architectures also have been proposed in the literature, like ZigBee, FlexiSec, and C-Sec [5,14,15]. But to the best of our knowledge, they are either lacking in providing strong security or they consume high energy during communication.

3. PROPOSED WORK

We propose a secure link layer communication architecture called EESCA. The proposed architecture provides strong security using non-repeatable IV and symmetric key crypto system.

3.1. Selection of cryptographic primitives

3.1.1. Symmetric key encryption.

The security issues like authentication, confidentiality, and integrity can be provided either using public-key cryptography or using symmetric-key cryptography. However, public-key cryptosystems require more memory and computation time than symmetric-key cryptosystems. So symmetric-key cryptography is preferred over public-key cryptography in WSN.

In WSN, DES is not considered to be secured because of its 64 bit key. To provide high security, triple DES is another option, but it is too slow. AES provides high security, but it also takes more computational time because the key length is at least 128 bits. The alternatives are Skipjack and RC5. But according to NIST (National Institute of Standards and Technology), recommendation Skipjack is not secured beyond 2010 [16] and [6] has shown that RC5 takes a lesser time than AES. TinySec shows that RC5 can be used to faster the computation using inline assembly code. So to provide high security and to faster the computation, we use RC5 with 80-bits key. Also to generate 32-bits MAC, RC5-32/16/10 is used [17].

Two different algorithms can be used for data encryption and to generate MAC. But that will increase the size of the required memory. To minimize the memory requirement, same encryption algorithm (RC5) is used for both. But to provide high security, two different keys, one to encrypt data and another to generate MAC are used. EESCA uses CBC mode to generate cipher text and MAC. Data is encrypted and MAC is generated in 64-bits blocks.

For authentication, sender generates MAC over packet header and payload. IV is always initialized as zero. Receiver verifies the MAC by generating the MAC over header and payload using the same key. The integrity and authentication is guaranteed if the generated MAC matched with the received MAC.

3.1.2. Keying mechanism.

Sharing and distribution of keys needed for data encryption and MAC generation are another important parts of any secure protocol. In [4,18], different types of keying mechanisms were discussed, which are global key, group key, individual key, pairwise shared key, etc. Pairwise-sheared key can provide strong security, but it is not suitable for WSN because of heavy energy consumption, and too many keys are needed. Individual key is the best option for strong confidentiality for a small network, but the problem is that for large WSN it requires more number of keys.

So EESCA uses group key(k1) to provide data confidentiality. A different key(K2) is used to generate MAC. The MAC is verified in each routing node. If unauthorized packet is detected, then the packet is not forwarded further and is dropped by the routing node. So key k2 is distributed in such a way that a node and the routing nodes share the same key. If routing path is changed after certain time interval, key should be updated accordingly. So, distribution of K2 depends on routing protocol. For our experiment, global key is used to check the replay packets.

3.2. Assumptions

To design EESCA, we make the following realistic assumptions:

- The symmetric keys have already been distributed in the network using appropriate symmetric key distribution mechanisms.
- Maximum number of nodes in the network is not more than 2048. As shown in [8,19], usually in real life scenarios, more than 2048 number of nodes in a WSN are not desired.
- Maximum packet loss in the WSN between any two received packets from the same sender is p where p varies from network to network.
- Initially, counter is initialized as zero for each sensor node.

3.3. Operation of energy efficient secure communication architecture

We divide the functionalities of EESCA into two parts based on the operations performed at the sender and at the receiver end (the receiver may be a routing node). To provide both confidentiality and authentication message encryption and MAC is generated using RC5 in CBC mode. We explain the sender and the receiver operations in the following.

DOI: 10.1002/sec

3.3.1. Sender operation.

Maximum TinyOS payload is 29 bytes. So to represent the payload length, we need maximum 5 bits. We use 11 bits for source address. We add a new 2-bytes field in the packet called LENSRC (length and source address) by appending last 5 bits of length field with 11 bits of source address (SRC) field. Data encryption in CBC mode needs non-repeating IV to provide semantic security. In EESCA, we encrypt the data in 64 bit blocks, so size of IV is 64 bits. Also, counter size is 32 bits, which is secured enough in the context of WSN, which is described in Section 4.1.2. IV is initialized by appending 2 bytes of destination address (DST), 2 bytes LENSRC, and 4 bytes counter as shown in equation (7).

$$IV_8 = DST_2 \| LENSRC_2 \| CTR_4 \tag{7}$$

Each packet contains IV, so the receiver needs the IV to decrypt the message. To reduce packet overhead, EESCA does not send counter (CTR) in packet header. Instead, we add a new field as shown in equation (10) called MwC (MAC with COUNTER), and MAC is replaced by MwC. So the packet header consists of DST, AM, and LENSRC as shown in equation (8).

$$\text{packet header} = DST_2 \| AM_1 \| LENSRC_2 \tag{8}$$

To provide strong replay protection, we incorporate permutation and substitution in the MwC creation phase. Assume that CTR, MAC, and MwC are denoted as $(c_{31}c_{30}\cdots c_1c_0)$, $(m_{31}m_{30}\cdots m_1m_0)$, and $(M_{31}M_{30}\cdots M_1M_0)$ where c_i, m_i, and M_i for $i = 0, \ldots, 31$ represent the i^{th} bit position of CTR, MAC, and MwC, respectively. Also, let X is the integer value of the last 5 bits of the counter $(c_4c_3c_2c_1c_0)$. Then perform a right circular shift on MAC by X positions or a left circular shift on MAC by 32-X positions as shown in equation (9).

$$MAC_X = (MAC \ >> \ X) \mid (MAC << (32 - X)) \tag{9}$$

Finally, MwC is computed by taking XOR of MAC and Counter as given in equation (10).

$$MwC_4 = (MAC_X)_4 \oplus CTR_4 \tag{10}$$

Energy efficient secure communication architecture encrypts data to provide confidentiality. The addition of the new filed MwC is to provide authentication and integrity. To provide replay protection, the counter is incremented by 1 for each packet. The packet creation procedure at the sender node is described in Algorithm 1.

3.3.2. Receiver operation.

The *MAC* is generated from the received encrypted data and packet header using same process used by the sender. Then take the right circular shift by X positions of the MAC. Generate MwC_r by taking bitwise XOR of MAC_X and CTR. If the received MwC matches with the computed MwC_r then the packet is accepted. Otherwise, repeat the process for p times. If the packet is not accepted, then reject the packet. In this way, authentication and data integrity are verified. The detailed description is given in Algorithm 2.

3.4. Packet format

Packet overhead is the main contributing factor for energy efficiency. EESCA is developed over TinyOS. TinyOS packet header fields are destination address (DST-2 bytes), AM-1 byte, length (LEN-1 byte), group ID (group-1 byte), and CRC-2 bytes. EESCA packet (as illustrated in

Algorithm 1
Packet creation algorithm at the sender side:

Input: Data, keys for RC5 (K_1) and (K_2), counter(CTR) of previous packet and packet header.

Output: Packet to be sent.

CTR = CTR + 1
// Increment the previous CTR value
IV = DST || LENSRC || CTR
Encrypted Data = E(K_1, (IV, Data))
// Data is encrypted using RC5 in CBC mode.
MAC = E(K_2, (IV_1, (packet header || Encrypted Data)))
// Again 'Encrypted Data' along with the packet header is encrypted using RC5 where $IV_1 = \mathbf{0}$
// in CBC mode to generate 4 bytes MAC.
$MAC_X = (MAC \ >> \ X) \mid (MAC << (32 - X))$
// Right circular shift on MAC by X positions or left circular shift on MAC by 32-X positions
//where X is integer value of last 5-bits of the CTR
$MwC = MAC_X \oplus CTR$
packet = Header || Encrypted Data || MwC

Algorithm 2
MwC check algorithm at the receiver side

Input: Received packet, maximum packet loss parameter (p) in the network, Counter (CTR) at receiver side, key (K_2 for RC5) .

Output: Received packet is accepted or rejected at receiver end.

MAC = E(K_2, (IV_1, (packet header || Encrypted Data)))
// where $E = RC5$, mode = CBC and $IV_1 = \mathbf{0}$
CTR_1= CTR at receiver side
FLAG=0
for $i = 0$ **to** $(p - 1)$ **do**
$CTR_1 = CTR_1 + 1$
$MAC_X = (MAC >> X) \mid (MAC << (32 - X))$
// Right circular shift on MAC by X positions or left circular shift on MAC by 32-X positions
//where X is integer value of last 5-bits of the CTR
$MwC_r = MAC_X \oplus CTR_1$
if(MwC == MwC_r)
then Update CTR as $CTR = CTR_1$
"Accept the packet"
FLAG=1, exit ()
endif
endfor
if (FLAG==0) **then** "Reject the packet"

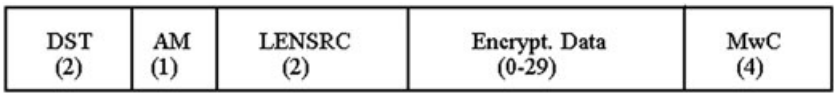

Figure 6. Packet format of energy efficient secure communication architecture (size of field s are in bytes).

Figure 6) contains length and source address (LENSRC-2 bytes) and MAC with CTR (MwC-4 bytes) fields. LEN, Group, and CRC fields are not used because of LENSRC and MwC fields. LENSRC contains LEN and SRC fields together where Group can be detected from SRC. MwC checks integrity; hence, CRC is no more required. As a result, EESCA contributes only 2 bytes increase in overhead over TinyOS packet whereas TinySec has 5 bytes, LLSP, and MiniSec has 3 bytes packet overhead over TinyOS packet.

4. SECURITY AND PERFORMANCE EVALUATION OF ENERGY EFFICIENT SECURE COMMUNICATION ARCHITECTURE

4.1. Security analysis

The main focus of EESCA is to provide authentication, integrity, confidentiality, replay protection, and data freshness in WSN. We also show that the replay protection does not depend on packet loss.

4.1.1. Authentication and integrity.

EESCA provides authentication and integrity using RC5 in CBC mode. Four bytes of the last block of the CBC mode is the MAC. MwC is the XOR of MAC with CTR. So security depends on length of MwC. In EESCA, length of MwC is 4 bytes that means an adversary has to try 2^{32} times in the worst case to correctly generate an MwC for a fabricated message. It is secured enough considering the lifetime of the sensor network for any real-life
application.

4.1.2. Confidentiality.

Disclosure of information is possible if IV is repeated affecting semantic security. In EESCA, we use 4-bytes counter. So counter is repeated after encryption of 2^{32} packets. But same counter values of different users may lead to same plaintext and ciphertext pairs. To eliminate this kind of problem, IV is the concatenation of DST, LENSRC, and CTR. So the IV will be repeated if a source encrypts 2^{32} packets. This makes the network computationally secure because if a sender sends a packet in 1 s, cipher text repetition can happened after 60 years.

DOI: 10.1002/sec

4.1.3. Replay protection and data freshness.

Energy efficient secure communication architecture uses counter to provide replay protection and data freshness. Counter of the older or replayed packets must be less than or equal to the counter of the last received packet. Hence, if the counter of the last received packet is one less than the new packet, then the packet is fresh and is not a replay one. This happens in an ideal network. But WSN is usually used in uncontrolled areas. The possibility of packet loss is high because of noise or collision of packets, etc. [20]. So we cannot be strict about counter of the new received packet.

Assume that p is a network-dependent parameter that denotes the maximum consecutive packet loss in the WSN. If the counter (CTR) of the last received packet is t, and the CTR of the new received packet is less than or equal to $t + p$, then the packet is accepted. So EESCA does not need any extra operation like counter resynchronization [4] or counter reset [8] for packet loss. Hence, EESCA is not dependant on packet loss.

Now, let us discuss the role of permutation and substitution in the MwC generation phase.

4.1.4. Permutation and substitution.

One kind of replay attack is possible if the permutation is not done. Assume that the CTR_R is the CTR used to generate MwC of the replay packet, that is, $MwC_R = MAC_R \oplus CTR_R$. Let CTR_C denotes the CTR of the last received packet at the receiver end. In this replay attack, an adversary can intercept a packet and can XOR i, where $i \in \{1, 2 \ldots, p\}$ with the MwC to generate a new MwC and can replay the packet directly afterwards. In some cases, $CTR_R = CTR_C + i$ where $i \in \{1, 2 \ldots, p\}$ and the packet will be accepted as a genuine packet.

Table I. Memory (RAM) requirements by difference schemes.

Schemes	Memory in bytes
TinySec	707
LLSP	749
MiniSec	788
EESCA-skipjack	710
EESCA-RC5	876

Now, let us see how the permutation protects from this replay attack. MAC is always rotated by x positions, where x is the integer value of the last five positions of CTR_C, that is, m_i bit is shifted to $m_{(i+x) \bmod 32}$-bit position. Assume that an adversary intercepts a packet and XOR j, where $j \in \{1, 2 \ldots, p\}$ with MwC to generate a new MwC_R, that is, m_i bit is shifted to $m_{(i+x+j) \bmod 32}$-bit position. This MwC_R will be accepted if it matches with the MwC, which is computed from CTR_C. This is possible if $m_{(i+x) \bmod 32} = m_{(i+x+j) \bmod 32}$ for all $i, j \in \{1, 2 \ldots, p\}$, that is, $(i + x) \bmod 32 = (i + x + j) \bmod 32$. This is possible for all $j \in \{0, 32, 64 \ldots\}$. In real life WSN applications, maximum packet loss between two received packets having same LENSRC is not more than 31 where all dropped packets have same SRC. So MwC is checked for maximum $p = 31$ in Algorithm 2.

4.2. Performance analysis

We have implemented EESCA for Mica2 motes having Atmel processor [21] and CC1000 radio chip where transmission current I is 21.8~mA, voltage V is 3~V, and data rate is 19.2 kbps [7,22]. We use TinyOS-1.x [13] operating system having integrated TinySec library. TinySec library provides a new link-layer protocol stack, which is used to implement our proposed scheme using nesC [23] programming language. For simulation and performance analysis purpose, we use AVRORA [9] simulator.

AVRORA is a simulation and analysis tool that can emulate the behavior of motes like, Mica2 and Micaz. It is used to analyze the energy consumption of various components of the Mica2 mote. For our experimention, we use TestTinySec and TosBase application for Mica2 sensor nodes and base station, respectively.

The performance metrics of WSN are transmission time, energy consumption, memory used, computational time, throughput, etc. For memory uses point of view, EESCA needs 876 bytes RAM and extra 8 MB flash over TinySec where as if skipjack is used, it takes only extra 3-bytes RAM over TinySec (shown in Table I) . Our scheme is as par with others as far as memory consumption is concerned.

Packet overhead increases both energy consumption and transmission time. So we calculate the packet over-

Table II. Comparison of transmission time with respect to packet length over TinyOS for Mica2 mote with data rate of 19.2 kbps.

Schemes	Fixed payload (B)	Packet overhead (B)	Total size(B)	Transmission time (mS)	Increased in transmission time over TinyOS for Mica2 mote (%)
TinyOS	8	22	30	12.500	—
TinySec	8	27	35	14.583	16.67
LLSP	8	25	33	13.750	10
MiniSec	8	25	33	13.750	10
EESCA	8	24	32	13.333	6.6

Table III. Comparison of energy consumption due to computation and communication overhead.

Schemes	Theoretical (mJ)	Increased over TinyOS (%)	Simulated (mJ)	Increased over TinyOS (%)
TinyOS	0.408	—	0.445	—
TinySec	0.544	33.33	0.549	23.37
LLSP	0.491	20.34	0.522	17.3
MiniSec	0.491	20.34	0.515	15.73
EESCA	0.463	13.48	0.504	13.25

head over the TinyOS. The transmission time and energy consumption of the competing schemes are calculated considering a packet having fixed payload and variable security overheads.

Table II illustrates the packet transmission time. It is increased by 6.6% for proposed scheme whereas it is increased by 16.7% for TinySec and 10% for MiniSec and LLSP over the TinyOS. Here the packet overhead includes packet header and media access control information (start symbol and synchronization field) [7].

The simulated energy consumption is calculated considering the energy consumption because of the computation and the communication. The computation cost of TinyOS is assumed as zero; hence, the energy consumption is also zero for computation. To calculate energy consumption theoretically, we calculate the power needed to transmit packet overhead and payload. Power is calculated as [7]:

$$P = I*V \tag{11}$$

where P is power, I is transmission current, and V is voltage. Energy depends on transmission time (T) and power, that is, E = P * T.

Increase in both theoretical and simulated energy consumption over TinyOS packet is shown in Table III. Clearly, both the simulated and theoretical energy consumption are increased by (13.25% and 13.48%) for the proposed scheme whereas they are increased by (23.37% and 33.33%), (17.3% and 20.34%), and (15.73% and 20.34%) for TinySec, LLSP, and MiniSec over TinyOS, respectively. We observe that there is a little difference in energy consumption for computational requirement of the different schemes.

5. CONCLUSION AND FUTURE WORK

We proposed a secure and energy-efficient link-layer communication architecture namely EESCA. Along with authentication, integrity, and confidentiality, it provides strong replay protection and data freshness, and it does not depend on heavy packet loss. To the best of our knowledge, EESCA reduces the communication overhead by at least 1 byte in each packet over the existing schemes. The authentication and integrity at routing node are also checked to reduce the energy consumption of routing nodes. EESCA is also not computationally demanding as the computational cost is of same order of magnitude as compared with the existing schemes. Both the factors contribute to the energy efficiency of EESCA.

Counter table takes 8 KB memory to store counter of last received packets. It has been shown that using hierarchical routing protocols, total network memory can be reduced [24–27]. One of our future goals is to design a suitable hierarchical routing protocol to adopt EESCA and to reduce network memory requirement.

REFERENCES

1. Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E. A survey on sensor networks. *IEEE Communications Magazine* 2002; **40**(8): 102–114.
2. Rawat P, Singh KD, Chaouchi H, Bonnin JM. Wireless sensor networks: a survey on recent developments and potential synergies. *The Journal of Supercomputing* 2014; **68**: 1–48.
3. Ren J, Li T, Aslam D. A power efficient link-layer security protocol (LLSP) for wireless sensor networks. *Military Communications Conference (MILCOM '05)*, IEEE, Atlantic City, NJ, 2005; 1002–1007.
4. Perrig A, Szewczyk R, Tygar JD, Wen V, Culler DE. SPINS: security protocols for sensor networks. *Wireless Network, Springer* 2002; **8**(5): 521–534.
5. Alliance Z. ZigBee specification. *Technical Report 053474r17*, ZigBee Standards Organization, Davis, California, 2008.
6. Karlof C, Sastry N, Wagner D. TinySec: a link layer security architecture for wireless sensor networks. *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, ACM, Baltimore, MD, USA, 2004; 162–175.
7. Lighfoot LE, Ren J, Li T. An energy efficient link-layer security protocol for wireless sensor networks. *Proceedings of the IEEE Conference on Electro/Information Technology (EIT '07)*, Chicago, IL, 2007; 233–238.
8. Luk M, Mezzour G, Perrig A, Gligor V. MiniSec: a secure sensor network communication architecture. *Proceedings of the 6th International Conference on*

Information Processing in Sensor Networks (ISPN '07), ACM, Cambridge, Massachusetts, USA, 2007; 479–488.
9. Titzer BL, Lee DK, Palsberg J. Avrora: scalable sensor network simulation with precise timing. *4th International Symposium on Information Processing in Sensor Networks (ISPN '05)*, IEEE, Los Angeles, California, 2005; 477–482.
10. Martins D, Guyennet H. Wireless sensor network attacks and security mechanisms: a short survey. *13th International Conference on Network-Based Information Systems*, IEEE, Takayama, 2010; 313–320.
11. Chen X, Makki K, Yen K, Niki P. Sensor network security: a survey. *IEEE Communications Surveys and Tutorials* 2009; **11**(2): 52–73.
12. Stallings W. *Cryptography and Network Security Principals and Practices*, 4th edn. Prentice Hall: Upper Saddle River, New Jersey, 2005.
13. Levis P, Madden S, Polastre J, Szewczyk R, Woo A, Gay D, Hill J, Welsh M, Brewer E, Culler D. *TinyOS documentation*. Available from: http://tinyos.stanford.edu/tinyos-wiki/index.php/Main_Page.
14. Jinwala D, Patel D, Dasgupta KS. FlexiSec: a configurable link layer security architecture for wireless sensor networks. *Computing Research Repository, JIAS* 2012; **abs/1203.4697**: 582–603.
15. Mohd A, Aslam N, Robertson W, Phillips W. C-sec: energy efficient link layer encryption protocol for wireless sensor networks. *Proceedings of the 9th Consumer Communications and Networking Conference (CCNC '12)*, IEEE, Las Vegas, Nevada USA, 2012; 219–223.
16. Barker E, Roginsky A. Transitions: recommendation for transitioning the use of cryptographic algorithms and key lengths. *Technical Report 800-131A*, National Institute of Standards and Technology, 2011.
17. Rivest RL. The RC5 encryption algorithm. *Fast Software Encryption (FSE)*, Springer, Leuven, Belgium, 1994; 86–96.
18. Zhu S, Setia S, Jajodia S. LEAP+: efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Networks (TOSN)* 2006; **2**(4): 500–528.
19. Levis P, Lee N, Welsh M, Culler D. TOSSIM: accurate and scalable simulation of entire TinyOS applications. *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SENSYS '03)*, ACM, Los Angeles, California, USA, 2003; 126–137.
20. Yu H, Wu D, Mohapatra P. Experimental anatomy of packet losses in wireless mesh networks. *6th Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '09)*, IEEE, Rome, 2009; 1–9.
21. Hill R Jason abd Szewczyk, Woo A, Hollar S, Culler D, Pister K. System architecture directions for networked sensors. *SIGPLAN Notices, ACM* 2000; **34**(5): 93–104.
22. Anastasi G, Conti M, Falchi A, Gregori E, Passarella A. Performance measurements of motes sensor networks. *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '04)*, ACM, Venice, Italy, 2004; 174–181.
23. Brewer E, Gay D, Levis P, von Behren P, Welsh M, Culler D. The nesC language: a holistic approach to networked embedded systems. *SIGPLAN Notices, ACM* 2003; **38**(5): 1–11.
24. Heinzelman WR, Chandrakasan A, Balakrishnan H. Energy-efficient communication protocol for wireless microsensor networks. *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS '00)*, IEEE Computer Society, Hawaii, US, 2000; 1–10.
25. Ye M, Li C, Chen G, Wu J. EECS: an energy efficient clustering scheme in wireless sensor networks. *24th International Performance, Computing, and Communications Conference (IPCCC '05)*, IEEE, Phoenix, Arizona, 2005; 535–540.
26. Raymond DR, Marchany RC, Midkiff SF. Scalable, cluster-based anti-replay protection for wireless sensor networks. *Information Assurance and Security Workshop (IAW '07)*, IEEE, West Point, NY, 2007; 127–134.
27. Jadidoleslamy H. TMS-HCW: a trust management system in hierarchical clustered wireless sensor networks. *Security and Communication Networks* 2015; **8**: 4110–4122.