A Minimal Protocol for Authenticated Key Distribution in Wireless Sensor Networks

Kalvinder Singh [1,2], Vallipuram Muthukkumarasamy [2]
[1] *Australia Development Laboratory, IBM*
kalsingh@au.ibm.com
[2] *School of Information and Communication Technology, Griffith University*
Gold Coast, Australia, v.muthu@griffith.edu.au

Abstract

Wireless sensor networks provide solutions to a range of monitoring problems. However, they introduce a new set of problems mainly due to small memories, weak processors, limited energy and small packet size. Thus only a very few conventional protocols can readily be used in sensor networks. This paper proposes efficient protocols to distribute keys in wireless sensor networks. This is achieved without the necessity of using traditional encryption. The proposed solution replicates the authentication server such that a group of malicious and colluding servers cannot compromise security or disrupt service. We show that the proposed multiple server authentication protocols will only have $O(n)$ complexity, where n is the number of authentication servers. The protocols use information from the sensor nodes and the servers to generate a new key, and do not solely rely on the sensor nodes to generate good random numbers. The scheme works well even when the base stations are untrusted. The proposed protocols guarantee that the new key is fresh and that the communicating nodes use the same key.

1. Introduction

When using symmetric key cryptography, if two entities sharing no previous secret want to communicate securely with each other, they generally do so with the assistance of a third party. In wireless sensor networks, the two entities are typically resource–constrained sensor nodes, and the third party is a resource–heavy base station. Typically, the base station provides an authentication service that distributes a secure session key to the sensor nodes. The base station is sometimes referred to as a trusted third party, since every client has to trust it by sharing a secret with it. Many applications using wireless sensor networks do not require encryption, but do require authenticated messages. However, key establishment protocols require encryption to safely transport a new session key between the nodes.

Another problem is that the security of a typical key distribution protocol depends on the assumption that the authentication server is trustworthy [1]. When wireless sensor networks are deployed in battlefields or developed to monitor homeland security, they have a likelihood of becoming the target of adversaries. In an open environment, an individual server may not be completely and permanently trustworthy. Several existing protocols [2], [3] handle the shortcomings of untrusted servers. We show the existing protocols have $O(n^2)$ complexity, and are therefore not suitable to a resource constrained environment.

In this paper we propose a number of protocols to address these problems. Our proposed multiple server authentication protocols will have $O(n)$ complexity in time and space, and $O(n)$ messages, where n is the number of authentication servers. The proposed schemes do not rely on the sensor nodes to generate cryptographically good pseudo–random numbers. The protocols use information from the sensor nodes and the servers to generate a new key, and do not require traditional encryption to transport the new session key. We will show that the sensor nodes can prove the new key is fresh. We will also demonstrate how key confirmation ensures the nodes are guaranteed to be using the same key.

2. Related Work

A. Sensor Networks

We refer to a sensor network as a heterogeneous system combining small, smart, cheap, and sensing devices with general–purpose computing elements. Sensor network applications [4] include tracking bushfires, monitoring wildlife, conducting military surveillance, and monitoring public exposure to contaminants.

The sensor nodes are resource constrained. A typical sensor node is the Mica mote [5]. It contains a 4 MHz processor with 512 KB flash memory and 4 KB of data memory. It also has a separate 512 KB flash memory unit accessed through a low–speed serial peripheral interface. The RF communication data transfer rate is approximately 40 kbps. The transmission range is approximately 100 metres in open space. Communication is the most energy intensive operation, and many protocols are designed to reduce as much communication overhead as possible.

Mica sensor nodes run TinyOS [6], an event–driven operating system specifically designed for wireless sensor environments. The memory footprint for TinyOS is small, a minimum installation (the core components) uses 400 bytes of data and instruction memory. TinyOS is developed in nesc and supports other hardware platforms. TinyOS network packet wraps the

payload during sensor node communication. To save space, it does not transmit the source address of the sender.

Network security in sensor environments differ in many ways from other distributed systems. Sensor nodes have little computational power, thus even efficient cryptographic ciphers must be used with care. Security protocols should use a minimal amount of RAM. Communication is extremely expensive; any increase in message size caused by security mechanisms comes at significant cost. Energy is the most important resource, each additional instruction or bit transmitted means the sensor node is a little bit closer to death. Nearly every aspect of sensor networks is designed with extreme power conservation in mind.

Perrig et al. proposed a cryptography library using symmetric keys [7]. Much of the research on authentication and key establishment protocols has used a symmetric key cryptography library, since until recently it was believed asymmetric key algorithms were too heavy weight. Perrig et al. [7] has proposed an authentication and key establishment protocols using symmetric keys. Recent work has shown asymmetric keys can also be used [8]. Singh et al. [9] has proposed an efficient key establishment protocol using elliptic curves. However, little work has been done to examine the performance of these protocols in different sensor environments. Vogt provided a survey of current authentication mechanisms in wireless sensor networks [10]. It should be noted that none of the solutions described above handle authentication when the base station is compromised.

B. Security for the Base Station

A simple approach is to replicate the authentication services of the server so that any one of several servers can perform authentication. However, this approach reduces the level of security; if one server is compromised, security for every replicated server is compromised. SIA [11] addresses the issue of compromised nodes by using statistical techniques and interactive proofs, ensuring the aggregated result reported by the base station is a good approximation to the true value, even if a small number of sensor nodes and the aggregation node may have been compromised. However, the communication overhead between sensor nodes and the base station is high. Other works have shown that some of the statistical methods used are not resilient to a group of malicious sensor nodes, and the end user should be aware of which statistical methods are easily cheated [12]. Another way to protect results is to use a witness node mechanism [13].

A different approach is to protect the base station location. Routing mechanisms to protect the location and disguise the identity of the base station have been proposed [14].

To make a key distribution protocol work in an environment where sensor nodes do not trust an individual base station, an authentication scheme, which can be used with limited resources and can reduce the requirement for trusting servers, must be found.

3. NOTATION AND ASSUMPTIONS

This paper will use the following notation to describe security protocols and cryptographic operations:

TABLE 1: NOTATIONS

Notation	Description
A, B	The two nodes wishing to share a session key.
S	A trusted server.
S_i	ith server in a set of n servers $\{S_1, \ldots, S_n\}$.
N_A, N_B	Nonces generated by nodes A and B respectively.
$\{M\}_K$	Encryption of message M with key K to provide confidentiality and integrity.
$[[M]]_K$	Encryption of message M with key K to provide confidentiality.
$[M]_K$	One–way transformation of message M with key K to provide integrity.
K_{AB}	The long–term key initially shared by A and B.
K_{AS}, K_{BS}	Long–term keys initially shared by A and S, and B and S respectively.
K_{AS_i}, K_{BS_i}	Long–term keys initially shared by A and S_i, and B and S_i, respectively for each $i \in 1, \ldots, n$.
X, Y	The concatenation of data strings X and Y.
$A \rightarrow B : m$	A sends a message m to B.
$A \rightarrow S_i : m$	A sends a message m to each server $\{S_i\}$.
$S_i \rightarrow A : m$	Each server $\{S_i\}$ sends a message m to A.

In protocols using a multiplicity of servers, we assume A and B do not trust any individual server.

4. SINGLE SERVER PROTOCOL

We investigated the available single server protocols in the hope of extending a protocol into a suitable key establishment protocol for wireless sensor networks. Boyd and Mathuria have surveyed the previous work on protocols for authentication and key establishment [15]. However, the way these protocols work and interact in the sensor environment has yet to be investigated. Of particular interest in this paper are the protocols using shared key cryptography.

Boyd and Mathuria have listed a total of twenty–two server–based key establishment protocols [15]. In this study we have reduced the number of protocols to a more manageable level. Protocols requiring sensor nodes that store old messages to prevent replay attacks have been removed. If an optimized version of the protocol exists, it is looked at in preference to the older protocol. The requirement of a flexible protocol ruled out protocols relying on timestamps, since not all sensor environments are guaranteed to have secure time synchronization. Another requirement is to minimize the amount of communication. We removed any protocols requiring five or more messages, leaving us with protocols sending less than or equal to four messages.

From the remaining filtered protocols, the protocol with the most properties was the Boyd four–pass protocol. The Boyd four–pass protocol [1] was chosen over the other server–based protocols due to the key confirmation property. However, if key confirmation is not considered important, then another protocol such as Bellare–Rogaway [16] can be used.

The Boyd four–pass protocol provides key authentication, key freshness and key confirmation in four messages. The clients A and B exclusively share a secret (K_{AS} and K_{BS}

respectively) with the trusted authentication server S. By executing the *Protocol 1*, as shown below, A and B intend to establish a session key K_{AB}. A by–product of this protocol, is the creation of K_S, a good random key created by the server, and used in the creation of K_{AB}. An interesting property of the key established between A and B is the use of information from S, A, and B to create the session key.

Protocol 1 Boyd key agreement protocol

$M1$	$A \rightarrow S:$	A, B, N_A
$M2$	$S \rightarrow B:$	$\{A, B, K_S\}_{K_{AS}}, \{A, B, K_S\}_{K_{BS}}, N_A$
$M3$	$B \rightarrow A:$	$\{A, B, K_S\}_{K_{AS}}, [N_A]_{K_{AB}}, N_B$
$M4$	$A \rightarrow B:$	$[N_B]_{K_{AB}}$

The nonces N_A and N_B can either be random numbers or a counter. The last two messages supply the key confirmation functionality for A and B.

An attractive feature of this protocol is that K_{AB} is generated using information from A, B and S, as shown in Equation (1). Although the nonces are used to guarantee the key is fresh, they need not be random. However, K_S should be a good random number.

$$K_{AB} = [N_A, N_B]_{K_S} \tag{1}$$

The requirement for $M2$ and $M3$ to have the locations of A and B be encrypted is not essential. The *Modified Protocol 1*, as shown below, is a slightly different version without the encryption of the locations, although, the locations are used to create the MAC (Message Authentication Code) values. Another modification is that message $M2$ has been split into two distinct messages, one that goes to A, and another that goes to B. Both messages can be sent at the same time. This minimizes the message size sent by the sensor node B.

Modified Protocol 1 Modified Body key agreement protocol

$M1$	$A \rightarrow S:$	A, B, N_A
$M2$	$S \rightarrow B:$	$[[K_S]]_{K_{BS}}, [A, B, K_S]_{K_{BS}}, N_A, A$
$M2'$	$S \rightarrow A:$	$[[K_S]]_{K_{AS}}, [A, B, K_S]_{K_{AS}}$
$M3$	$B \rightarrow A:$	$[N_A]_{K_{AB}}, N_B$
$M4$	$A \rightarrow B:$	$[N_B]_{K_{AB}}$

The security of the new protocol relies upon the unforgeability property of the message authentication code. The security is slightly weakened because the location names are no longer verified from decrypting the message and MACs. When B and A receive their messages, the location names for B and A can only be verified from the message authentication codes. However, conventional security protocols err on the side of caution [17]. Most algorithms producing MACs are good enough, because the probability that the location names are not A and B is extremely low. The benefit to performance is considered to be worthwhile, at the cost of a minimal decrease in security.

This protocol will only need to be run once between A and B. The sensor nodes can cache K_S and instead of contacting the server again, they can then use a different protocol to establish a new key. Upon further investigation, we discovered that we can use the Bellare–Rogaway MAP1 protocol [18], a provably secure entity authentication protocol, to produce a new session key.

Protocol 2 Bellare–Rogaway MAP1 protocol

$M1$	$A \rightarrow B:$	A, N_A
$M2$	$B \rightarrow A:$	$N_B, [B, A, N_A, N_B]_{K_{AB}}$
$M3$	$A \rightarrow B:$	$[A, N_B]_{K_{AB}}$

The MAP1 protocol, as shown in *Protocol 2*, provides mutual authentication for A and B. If K_{AB} is calculated using Equation (1), then this protocol becomes a key establishment protocol. The new K_{AB} key is guaranteed to be fresh, since N_A and N_B are used to create the new key. Key confirmation for both sensor nodes is another feature of this protocol. By creating a new K_{AB} we have transformed an entity authentication protocol into a key establishment protocol.

An interesting property of *Protocol 2* is that encryption is not used to establish the key. There are situations where wireless sensor environments do not need to support encryption, and may only need integrity checking. However, most key establishment protocols, where there is a trusted server involved, require some form of encryption.

A. Removing the need for encryption

Janson and Tsudik developed an authenticated key distribution that did not require traditional encryption when establishing a new session key [19]. We extend their technique to remove the need for encryption in our proposed protocol.

In our *Proposed Protocol 1*, the following constructs are used: $AUTH_A = [A, B, K_S]_{K_{AS}}$ and $MASK_A = [[AUTH_A]]_{K_{AS}}$. There are similar constructs for sensor node B. The $MASK$ can either be created from an encryption algorithm or from a MAC. In cases like CBC–MAC, where the algorithm uses an underlying encryption algorithm, it may be more efficient to use encryption. In other cases where there is no encryption algorithm available (for instance there is only HMAC–MD5) then the $MASK$ can be created by using the MAC. The value for K_S is calculated using Equation (1).

Proposed Protocol 1 No Encryption Key Agreement Protocol

$M1$	$A \rightarrow S:$	A, B, N_A
$M2$	$S \rightarrow B:$	$A, N_A, AUTH_B, MASK_B \oplus K_S$
$M2'$	$S \rightarrow A:$	$AUTH_A, MASK_A \oplus K_S$
$M3$	$B \rightarrow A:$	$[N_A]_{K_{AB}}, N_B$
$M4$	$A \rightarrow B:$	$[N_B]_{K_{AB}}$

B. Analysis

There are only two messages that contain the $MASK$; the message $M2$ – server S sending to node B, and the message $M2'$ – server S sending to node A. Neither A nor B ever send out the $MASK$. However, an adversary can try to obtain the

$MASK$ value by interrogating S. If an adversary pretends to be A, it does not matter what value gets passed to S, because S should produce a new K_S, and therefore a new $MASK$ and a new $MASK \oplus K_S$.

As shown in Equation (2), if two exclusive–ors produce the same value, and the keys are different, then the $MASK$ will have to be different.

$$MASK \oplus K_S = MASK' \oplus K'_S \tag{2}$$

If the $MASK$ is the same, as shown in Equation (3), then no extra information about K_S can be obtained, as long as a strong MAC is used. It is assumed that the adversary does not know the long term key K_{AS}.

$$[A, B, K_S]_{K_{AS}} = [A, B, K'_S]_{K_{AS}} \tag{3}$$

Since B does not initiate the protocol, it has no input into the creation of $MASK$. So an adversary who pretended to be B has got no more knowledge or input then if they pretended to be A.

The integrity of the key is also ensured since key modification requires simultaneous modification of $AUTH$ as well as $MASK \oplus K_S$.

The analysis given above makes the assumption that the key K_S has not been compromised. Since K_S is never used as a session key, or used to encrypt any plaintext, the keys K_{AS} and K_{BS} should be compromised before K_S. The sensor nodes should regularly refresh their keys with the base station. This of course will fail if either A or B becomes compromised. However, key establishment protocols between A and B cannot detect if either one of the sensor nodes is compromised. Communication between the nodes will need to be analysed to detect if any false data has been sent [11],[13].

If the server becomes compromised, the key K_S may also become compromised. A simple solution to this is to use multiple servers to create the key, so that even if one or more servers become compromised it will not affect the security of K_S.

5. Limitations in using Multiple Servers

Boyd and Mathuria have produced a survey of the current key establishment protocols using multiple servers [15]. In their survey, two protocols were listed: Gong's multiple server protocol [2], and the Chen–Gollmann–Mitchell protocol [3]. However, this survey did not take into account the unique problems of a sensor environment.

The main goals of using multiple servers in a sensor network are:

- Even if one or more servers become unavailable, it may still be possible for the sensor nodes to establish a session key.
- Even if one or more servers are untrustworthy, the sensor nodes may still be able to establish a good key.

A feature of Gong's protocol is that the sensor nodes choose the keying material while the n servers act as key translation centres. This allows keying material from one sensor node to be made available to the other. To ensure that the correct key can be recovered (even if some servers become unavailable), the sensor nodes split up their secrets using a threshold scheme such as Shamir's scheme [20]. To prevent compromised servers from disrupting the protocol, the sensors form a cross–checksum for all the shares. The cross–checksums are a combination of a one–way hash of the shares.

The other multiple server protocol is the Chen *et al.* multi-server protocol. A feature of this protocol is that the servers, rather than the sensor nodes, choose the keying material. Both nodes employ a cross–checksum to decide which servers have given valid inputs.

The major problem with this protocol is the size of the messages. The messages sizes are of $O(n^2)$, which is not desirable in a sensor network. Another problem with the protocols is that the size of the output of the one–way function will have to be reasonably large (otherwise a malicious server can quickly calculate the possible values of the shares). So for small values of n, the message sizes themselves will still be very large for a sensor network. The cross–checksums in Chen's protocol also have to be encrypted, which wasn't in Gong's protocol.

Another aspect of the multiple server protocols is the creation of the new session key. The sensor nodes retrieve the new key by using a secret sharing mechanism such as the one defined in [20]. The time complexity to compute n shares with t trusted sources is $O(nt)$. The time complexity to recover the key is $O(t \log_2 t)$. Having such a scheme or similar scheme in a sensor node will consume more resources in an already resource–constrained environment.

The existing multiple server protocols are therefore not suited for a sensor network environment. An ideal solution with the desired characteristics requires new multiple server protocols.

6. Proposed Multiple Server Protocol

The multiple server authentication protocol we have developed is based on the concept of *Protocol 1*. This will maintain security characteristics similar to those of the centralized authentication protocol, as shown in *Protocol 1*. Due to the severe resource constraints that exist in sensor nodes, a multiple authentication server protocol should have low computational complexity.

In our attempt to create an efficient multiple server protocol, we specified n servers. The sensor node A sends the first message, A, B, N_A, to each of the servers. Each server sends their message to both sensor nodes A and B. Sensor node B sends N_B, the keying data, and the cross–checksums created by B. It is important to note at this stage that K_S is unknown, so unlike the original protocol, B is not able to send $[N_A]_{K_{AB}}$. When sensor node A receives the next message, it will calculate its own cross checksums, and compare them against the cross–checksums created by B. At this stage, the keys K_S and K_{AB} are created. Sensor node A sends its cross–checksums to B, so B can create K_{AB}. The final message completes the key confirmation between A and B.

The *Proposed Protocol 2* provides key authentication, key freshness and key confirmation, using multiple authentication servers. In our *Proposed Protocol 2*, the following constructs are used: $AUTH_{Ai} = [A, B, K_i]_{K_{AS_i}}$ and $MASK_{Ai} = [[AUTH_{Ai}]]_{K_{AS_i}}$. There is similar corresponding constructs for node B. The value for K_S is calculated using Equation (1).

Proposed Protocol 2 A Preliminary Multiple Server Protocol

$M1$	$A \rightarrow S_i:$	A, B, N_A
$M2$	$S_i \rightarrow B:$	$N_A, A, S_i, AUTH_{Bi}, MASK_{Bi} \oplus K_i$
$M2'$	$S_i \rightarrow A:$	$S_i, AUTH_{Ai}, MASK_{Ai} \oplus K_i$
$M3$	$B \rightarrow A:$	$cc_B(1), \ldots, cc_B(n), N_B$
$M4$	$A \rightarrow B:$	$cc_A(1), \ldots, cc_A(n), [N_B]_{K_{AB}}$
$M5$	$B \rightarrow A:$	$[N_A]_{K_{AB}}$

Both of the sensor nodes and the servers contribute to the key value. The values N_A and N_B are generated by A and B respectively as input to the MAC function, that determines the session key. The key used with the MAC function is generated by the servers. Both A and B compute the session key as $K_{AB} = [N_A, N_B]_{K_S}$. The nodes should have a minimum number of servers returning valid results before confirming that the key is valid. Node B will calculate $cc_B(i)$ for $\forall i \in 1, \ldots, n$.

$$cc_B(i) = \begin{cases} [K_i]_{K_i}, & \text{if valid,} \\ EM, & \text{otherwise} \end{cases} \quad (4)$$

where EM is an error message; an example will be the value zero. There is a remote chance that a valid case may be zero. If the valid value is zero, the server should be considered a compromised server (even though it is not a malicious server). Node A will calculate $cc_A(i)$, and compare it with $cc_B(i)$. If they are the same, then the server S_i is valid. Below is a way to calculate the cross checksum for $cc_A(i)$.

$$cc_A(i) = \begin{cases} cc_B(i) = [K_i]_{K_i}, & \text{if valid,} \\ EM, & \text{otherwise} \end{cases} \quad (5)$$

After the comparison of all the cross checksums, a set of valid keys $V_1, \ldots, V_m$ should remain. The creation of K_S is defined as follows. The symbol $\oplus$ denotes exclusive–or.

$$K_S = V_1 \oplus \ldots \oplus V_m \quad (6)$$

Where V_i is the valid key from S_i, and m is the total number of valid servers $t \leq m \leq n$, where t is the minimal number of trusted servers. However, unlike the existing multiple server protocols, the trusted servers will not be able to calculate K_S. The calculated $cc_A(i)$ values are returned to B, where B performs similar checks as A does and calculates K_S.

7. Analysis

Our *Proposed Protocol 2* has a number of advantages, one of which is that the nodes do not need good random number generators to create the nonces. The nodes could even safely use a counter for their nonce values. Another advantage is that if a server or a number of servers are unavailable, the authentication service itself still exists through the other servers. The servers and the sensor nodes have different keys; even if one or more servers become compromised, the authentication service or the security of the system is not compromised.

The proposed protocol only encrypts random information. If the encryption cipher uses an IV value (such as the one currently used in TinyOS [6]) then we can use a constant IV value. However, the constant IV value chosen for our protocol must only be used to encrypt the random data and should never be used to encrypt other information. Also, a wide variation of different ciphers can safely be used.

Some Message Authentication Codes (MACs) have vulnerabilities when the message sizes are variable. All of our message sizes are of constant value, allowing us to safely use a wider range of MACs than previously available. The size of the MACs can be lower than that of conventional protocols. The integrity checking is performed by the sensor nodes. If x is the size of the MAC in bits, then an adversary has 1 in 2^x chance in blindly forging a valid MAC for a particular message. The adversary should be able to succeed in 2^{x-1} tries. Due to the low bandwidth of sensor nodes, a 4 byte MAC, requiring 2^{31} packets, will take years to complete. If an adversary did attempt this attack, the sensor node would be non–functional within that period. In addition, an adversary will need to forge $2t$ MACs; t MACs to A and t MACs to B, and stop traffic from the other base stations before they can determine the value of K_{AB}.

In order to study the performance impacts of each of the multiple server protocols, we first define the following symbols. The size of location indicator is a_0, the nonce size is a_1, the key size is a_2, the hash size is a_3, and the number of servers is n. The following equations are used to define how many bytes are sent for each message: $M1_i = 2a_0 + a_1$, $M2_i = 2a_0 + a_1 + a_2 + a_3$, $M2'_i = a_0 + a_2 + a_3$, $M3 = a_1 + na_2 + 2na_3$, $M4 = (n+1)a_3$, and $M5 = a_3$. However, there are n messages of type $M1$, $M2$ and $M2'$ sent.

8. Comparison

The computational complexity of the existing multiple server protocols is greater due to the fact that the key is constructed using a key sharing mechanism. This has two major drawbacks in a sensor network environment. The first is the amount of extra code (and therefore memory overhead) needed when creating the new key. The second is the additional computation (and therefore extra energy) required when creating the new key. If we were to use a threshold scheme such as Shamir's scheme [20] to recover the key, the computational complexity will be $O(t \log_2 t)$. Whereas, our proposed protocols use a simple exclusive–or function (as describe in Equation (6)) to recover the key, having a computational complexity of only $O(n)$. Not only does this save on computational cost, but also has the added benefit of requiring less code than a full blown key sharing threshold scheme. Also, the existing multiple server protocols have message sizes of $O(n^2)$, whereas our proposed protocols are of $O(n)$.

The disadvantage of not using a full blown key share threshold scheme, is that t servers can only calculate the new session key between A and B if they were the only servers involved in the protocol. However, for performance reasons we have not placed the same restriction on our proposed multiple server protocols.

A memory comparison for our application was performed in a TinyOS environment. We compared CBC–MAC using SkipJack and HMAC–MD5 on our application, as shown in Table 2. The ROM memory is greater for HMAC–MD5 than it is for CBC–MAC. The amount of RAM used is less for HMAC–MD5.

TABLE 2: MEMORY OVERHEAD IN BYTES ON MICA2 PLATFORM

Memory	CBC–MAC	HMAC–MD5
ROM	22410	32532
RAM	800	775
.data	28	92
.bss	772	683
.text	20382	32440

The .bss and .data segments use SRAM, while the .text segment uses ROM. The values above indicate that HMAC–MD5 uses less RAM and more ROM than CBC–MAC. The .text contains the machine instructions for the application. The .bss contains uninitialized global or static variables, and the .data section contains the initialized static variables. The .text section of HMAC–MD5 can be further decreased if the common transformation macros are re–factored into functions.

Our proposed protocols rely on the fact that the key K_S does not get compromised. We have shown that this is unlikely in the single server case, and very unlikely in the multiple server case. Other protocols, which do not have the concept of K_S do not have this problem.

9. CONCLUSIONS

Key distribution protocols, without the assumption of trusting an individual authentication server, are needed where clients have no reason to trust individual servers.

We have proposed a multiple server protocol for authentication in a wireless sensor environment. Most of our methodology can be used in other networks with low–end devices. With the removal of the requirement for the servers to know the keys between each of the sensor nodes, we have proposed a multiple server protocol where the message sizes are of $O(n)$. We have also shown how an entity authentication protocol can be transformed into a key establishment protocol.

The proposed protocols also have the added benefit of not solely relying on the sensor nodes to generate cryptographically sound pseudo–random numbers, but still using information from each of the sensors to generate the new key. We have shown our protocols are flexible enough for them to be used with most cryptographic primitives and in many environments.

A security and performance analysis was carried out to demonstrate the strength and limitations of our proposed protocols. The previous section shows that our proposed protocol has a smaller message size than the existing multiple server protocols. A memory comparison between HMAC and CBC–MAC shows different memory characteristics between the two cryptographic algorithms.

REFERENCES

[1] C. Boyd, "A class of flexible and efficient key management protocols," in *CSFW '96: Proceedings of the Ninth IEEE Computer Security Foundations Workshop*. Washington, DC, USA: IEEE Computer Society, 1996, p. 2.

[2] L. Gong, "Increasing availability and security of an authentication service," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, pp. 657–662, June 1993.

[3] L. Chen, D. Gollmann, and C. J. Mitchell, "Key distribution without individual trusted authentication servers," in *8th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1995, pp. 30–36.

[4] N. Bulusu, "Introduction to wireless sensor networks," in *Wireless Sensor Networks: A Systems Perspective*, N. Bulusu and S. Jha, Eds. Artech House, 2005.

[5] Crossbow, "Crossbow," http://www.xbow.com/, 2006.

[6] TinyOS, "http://www.tinyos.net/," 2006.

[7] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security protocols for sensor networks," in *Seventh Annual International Conference on Mobile Computing and Networks (MobiCOM 2001)*, Rome, Italy, July 2001.

[8] D. J. Malan, M. Welsh, and M. D. Smith, "A public–key infrastructure for key distribution in tinyos based on elliptic curve cryptography," in *Proc. 1st IEEE Communications Society Conference on Sensor and Ah Hoc Communications and Networks (SECON '04)*. Santa Clara, CA, USA: IEEE Computer Society Press, October 2004, pp. 71–80.

[9] K. Singh, K. Bhatt, and V. Muthukkumarasamy, "Protecting small keys in authentication protocols for wireless sensor networks," in *Proceedings of the Australian Telecommunication Networks and Applications Conference*, Melbourne, Australia, December 2006, to be published.

[10] H. Vogt, "Exploring message authentication in sensor networks." in *ESAS*, Heidelberg, Germany, August 2004, pp. 19–30.

[11] B. Przydatek, D. Song, and A. Perrig, "Sia: secure information aggregation in sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2003, pp. 255–265.

[12] D. Wagner, "Resilient aggregation in sensor networks," in *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*. New York, NY, USA: ACM Press, 2004, pp. 78–87.

[13] S. Zhu, S. Setia, and S. Jajodia, "Leap: efficient security mechanisms for large-scale distributed sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2003, pp. 308–309.

[14] J. Deng, R. Han, and S. Mishra, "Intrusion tolerance and anti-traffic analysis strategies in wireless sensor networks," in *Dependable Systems and Networks, 2004 International Conference on*. IEEE, June 2004, pp. 637–646.

[15] C. Boyd and A. Mathuria, *Protocols for Authentication and Key Establishment*, U. Maurer and R. Rivest, Eds. Springer Berlin / Heidelberg, 2003.

[16] M. Bellare and P. Rogaway, "Provably secure session key distribution: the three party case," in *STOC '95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*. New York, NY, USA: ACM Press, 1995, pp. 57–66.

[17] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: a link layer security architecture for wireless sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2004, pp. 162–175.

[18] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *CRYPTO '93: Proceedings of the 13th annual international cryptology conference on Advances in cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1994, pp. 232–249.

[19] P. Janson and G. Tsudik, "Secure and minimal protocols for authenticated key distribution," *Computer Communications*, vol. 18, no. 9, pp. 645–653, September 1995.

[20] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.