13

Reliability of Authenticated Key Establishment Protocols in a Complex Sensor System

Kalvinder Singh[1] and Vallipuram Muthukkumarasamy[2]
[1]*IBM and Griffith University*
[2]*Griffith University*
Australia

1. Introduction

Wireless sensor and actuator networks are an important component in mechatronics. As the sensors permeate the environment they can monitor objects, space and the interaction of objects within a space. Sensors can monitor a wide range of diverse phenomena by collecting information such as vibrations, temperature, sound, and light. Different sensors have different associated costs. For example, a sensor simply detecting light will have different costs to a sensor recording sound. However, less costly sensors can be used to detect a phenomenon before alerting the more costly sensors to start their monitoring. As the number of heterogeneous sensors increases, so will the amount of interactions between the sensors. In an instrumented system sensors are becoming more interconnected, interactive, intelligent, and interdependent.
There are many different types of sensor environments, ranging from large areas covered by sensors, to many sensors in a small area.Different environments have a wide range of different characteristics. The different environments may include :

Transportation The management of traffic and the transportation of goods. This can include traffic lights, traffic controllers to manage traffic flow, or shock sensors, impact indicators for fragile goods.

Smart Buildings A distributed control system to efficiently manage building automation. For instance, smart buildings allows people to visualize and manage air–conditioning, lighting, or other appliances in the building.

Environmental Monitoring The monitoring of the characteristics and quality of the environment. This can include continuous monitoring a river system for pH level, dissolved oxygen, conductivity, turbidity and colour

Health and Well–Being The monitoring and delivery of health to a patient. This can include wearable wireless health sensors for remote bio-monitoring. Enabling patients to live independently.

Water The management of distribution of water to the populace. Water management includes the planning, developing, and distributing water resources.

Each environment can be considered to be a distributed control system with similar requirements. Figure 1 is a common framework used by each environment. The *real world* signifies the environment that needs to be managed. For instance, *real world* could be a building, a patient, water ways, toll way, or a rainforest. To manage each system, there needs to be a *data and measurement platform*. The platform is used to collect in situ continuous sensor data. The data is then sent to the *analysis platform*. The analysis of the data involves the assimilation, interpolation and explanation of the data. By the end of the analysis cycle the system should have high–quality trusted data. The high–quality trusted data is then sent to the *decision model*, where simulations are run and predictions are made. The outcomes or final decisions are sent to the actuators in the system. Actuators can include air-conditioning, lights or even a person.

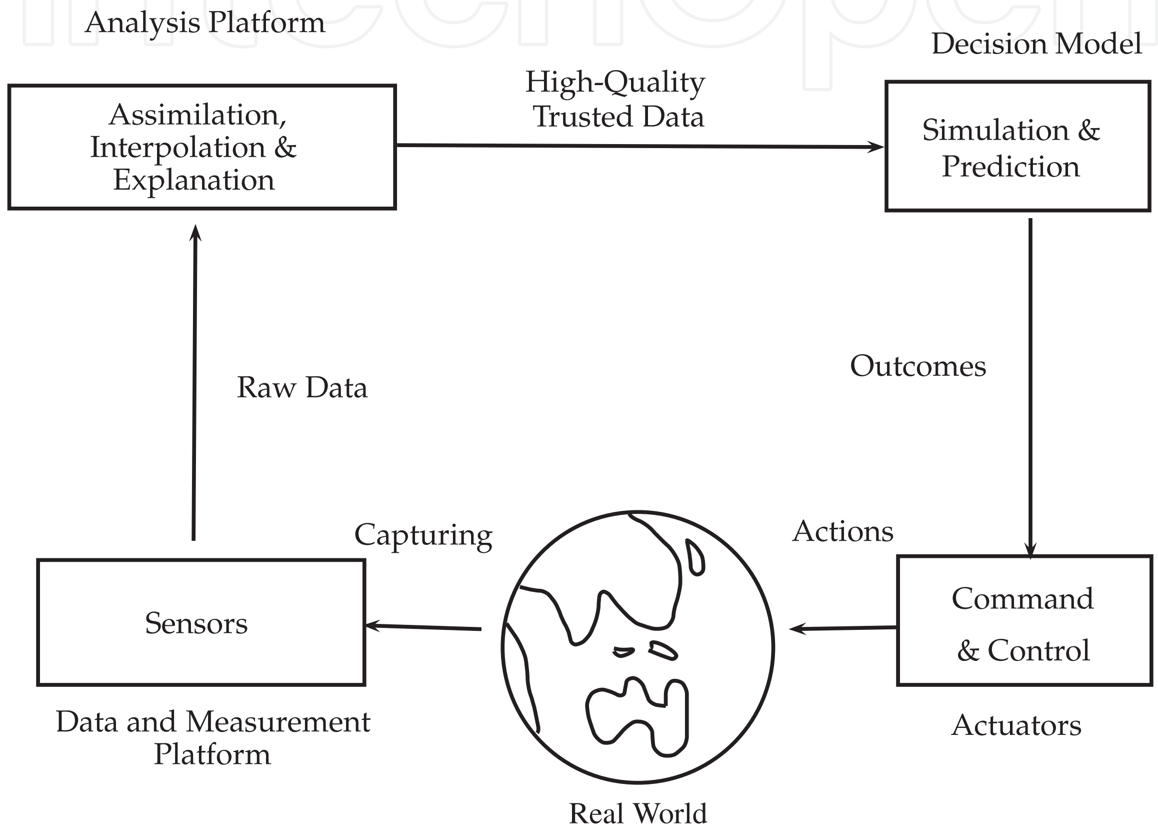

Fig. 1. Logic System Flow Diagram

The requirement for high–quality trusted data in the system, places the need for security, information assurance and reliability into the system. The future networks are increasingly becoming heterogeneous and highly dynamic. Cooperative networking paradigm envisages advanced wireless networks to cost effectively provide multimedia services and applications (incorporating various sensors) anywhere, anytime. Self-organising, opportunistic networking tries to exploit resources (bandwidth, memory, computing power, etc) available in various devices and nodes in the system. This approach of moving away from exclusively relying on infrastructure networks raises serious challenges to overall reliability of the system.
The complex nature of operation of sensor network applications involve use of different communication technologies, ranging from Cellular, WLAN, WiMax, WPN, WMN, WSN. Although much work has been done on the inter-operability between these networks, the authentication of nodes and the level of security available for services in such situations are still needs innovative solutions.

The system may not require encryption, but may require authentication. For instance, a sensor detecting the amount of light in a publicly available room supplies no or little information to an adversary. However, the information the light sensor sends to another device should be authenticated before any actions are taken.

One of the security vulnerabilities investigated in this chapter is the use of small keys. For performance reasons, sensors use small keys, however, small keys give an attacker a greater opportunity to compromise a sensor node. Thus, small keys will require frequent refreshing. Old keys can be used to generate new keys. But, if the old key is compromised then the new key can easily be compromised. A secure, efficient and scalable mechanism to freshen the keys between the sensors nodes is discussed.

When sensors 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 or an intermediate sensor node may not be completely and permanently trustworthy. Several existing protocols (Chen et al., 1995; Gong, 1993) found in traditional network protocols handle the shortcomings of untrusted servers. We, however, show that those existing protocols have an $O(n^2)$ complexity, and are therefore not suitable to a resource constrained environment.

In this paper, we describe five protocols to address the problems described above. These protocols use information from the sensor nodes and the servers to generate a new key. Some of the protocols do not rely on the sensor nodes to generate cryptographically good pseudo–random numbers. We will show the sensor nodes can prove that the new key is fresh, and will demonstrate how key confirmation ensures that the nodes are guaranteed to be using the same key.

Section 2 shows the notations used throughout this paper. Section 3 provides a background to sensor networks, and the unique security challenges. Section 4 describes the multiple server protocols found in traditional networks, and the problems with using those protocols in sensor networks. Section 5 describes three multiple server protocols for sensor networks. Section 5.3 provides a detailed analysis of the multiple server protocols. Section 6 compares the protocols in different sensor network environments. Finally, a conclusion is provided in Section 7.

2. Notation and assumptions

Table 1 gives a list of notations, which are used when describing authentication and key establishment protocols in this paper.

In protocols using a multiplicity of servers an assumption is made that A and B do not trust any individual server. TinyOS is used as the development environment, with the following restrictions on the size of the data structures. The key size is 64 bits, the nonce size is 1 byte, the packet size is 29 bytes, and finally the location size is 2 bytes (which allows 64K of nodes in a sensor network).

3. Background

We refer to a sensor network as a heterogeneous system combining small, smart, and cheap, sensing devices (sensors) with general–purpose computing elements. A sensor network consists of a potentially large number of sensors; there may also be a few control nodes, which may have more resources. The functions of the control nodes include: connecting the sensor network to an external network; aggregating results before passing them on; controlling the sensor nodes; providing services not available to a resource constrained environment.

Notation	Description
A and B	The two nodes who wish to share a new session key.
S	A trusted server.
S_i	A server in a set of servers $S_1, \ldots, S_n$, where n is the numbers of servers.
N_A	A nonce generated by A.
$\{M\}_K$	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'_{AB}	The value of the new session key.
K_{AS}, K_{BS}	Long-term keys initially shared by A and S, and by B and S for centralized authentication server.
K_{AS_i}, K_{BS_i}	Long-term keys initially shared by A and S_i, and by B and S_i, for each $i \in 1, \ldots, n$.
X, Y	The result of the concatenation of data strings X and Y.
$A \to B : m$	Denotes that A sends a message m to B.
$A \to S_i : m$	Denotes that A sends a message m to each server.
$S_i \to A : m$	Denotes that each server sends a message m to A.
$X \oplus Y$	Exclusive–or operation with X and Y.

Table 1. Notations

Sensor network applications (Bulusu, 2005) include tracking bushfires, monitoring wildlife, conducting military surveillance, and monitoring public exposure to contaminants.
Some sensor nodes are resource constrained, such as the Mica mote (Crossbow, 2006). The Mica motes contain a 4 MHz processor with 512 KB flash memory and 4 KB of data memory. A Mica mote also has a separate 512 KB flash memory unit accessed through a low-speed serial peripheral interface. The RF communication transfer rate is approximately 40 kbps. The maximum transmission range is approximately 100 meters in open space. Communication is the most expensive operation in sensor networks.
Other components in the sensor network may have more computation power and memory. Examples are the Stargate platform (Crossbow, 2006), the GNOME platform (GNOME, 2006), the Medusa MK–2 platform (Medusa, 2006), and the MANTIS platform (MANTIS, 2006). These platforms may use other higher–level operating systems such as the Linux operating system. The platforms themselves may have additional communication mechanisms. For instance, the GNOME platform also has an Ethernet connection.

3.1 Key establishment in wireless sensors

Security in sensor environments (where there are sensors with low resources) differs in many ways from that in other systems. Efficient cryptographic ciphers must still 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 a significant cost. Energy is the most important resource, as each additional instruction or bit transmitted means the sensor node is a little closer to becoming nonfunctional. Nearly every aspect of sensor networks is designed with extreme power conservation.
There are many aspects to WSN security (Deng et al., 2005); ranging from data fusion security, location aware security, to the lower level security primitives such as cryptography,

authentication and key establishment protocols. We will not cover all aspects of WSN security in this paper, instead only concentrate on some of the lower level primitives: authentication and key establishment protocols.

Several cryptography libraries using symmetric keys (Karlof et al., 2004; Perrig et al., 2001) have been proposed. Recent work has shown that even asymmetric keys may be used in WSNs (Malan et al., 2004; Watro et al., 2004). Singh et al. (Singh et al., 2006) has proposed an efficient key establishment protocol using elliptic curves. However, they still consume considerably more resources than the symmetric counterparts.

Key establishment protocols are used to exchange and set up shared secrets between sensor nodes. Asymmetric cryptography is unsuitable for most sensor architectures because of the higher computational overhead, and energy and memory consumption. When using symmetric keys, we can classify the key establishment protocols in WSNs into three main categories: Pair–wise schemes; Random key predistribution schemes; Key Distribution Center (KDC) schemes.

The simplest is the full pair–wise scheme (Chan et al., 2003), where each node in a network of total of n nodes shares a unique pairwise key with every other node in the network. The memory overhead for every sensor node is $(n-1)$ cryptographic keys. Other pair–wise schemes (Blundo et al., 1993; Leighton & Micali, 1994) also have $O(n)$ memory cost. In a pair–wise scheme, the sensor network is not compromised even if a fraction of the sensors are compromised.

Random key predistribution schemes are the second category (Chan et al., 2003). This is a major class of key establishment protocols for sensor networks. They rely on the fact that a random graph is connected with high probability if the average degree of its nodes is above a threshold. After the connected secure network is formed, the protected links can be further used for agreeing on new keys, called *path–keys*. One of the main problems with random key predistribution schemes is that if a certain number of sensor nodes become compromised, then the entire sensor network can be compromised.

The third category is the KDC scheme. If two entities sharing no previous secret want to communicate securely with each other, they can receive assistance from a third party. In WSNs the two entities are typically resource–constrained sensor nodes, and the third party is a resource–heavy base station. However, in a multi-tiered environment, such as a military battlefield, the third party may be a resource heavy camera. Typically, the base station provides an authentication service that distributes a secure session key to the sensor nodes. The level of security of a typical key distribution protocol depends on the assumption that the third party is trustworthy (Boyd, 1996).

KDC schemes use the least amount of memory compared with the other two categories, and has an extra advantage of providing authentication for the sensor nodes. Examples of KDCs in WSNs were first proposed in SPINS (Perrig et al., 2001). However, the SPINS protocol may not be suited for every WSN topology. For instance, it does not easily scale to a large WSN, since the non–uniform communication will focus the load onto the KDC. This may cause the battery life of the network nodes to diminish considerably. However, a KDC mechanism is suitable for a surveillance sensor environment.

Hybrid schemes can also by created by combining different key establishment categories. The PIKE scheme (Chan & Perrig, 2005) is such a scheme, it combines a pair–wise scheme with the KDC scheme, where one or more sensor nodes act as a trusted intermediary to facilitate the key establishment. The scheme was developed to limit the amount of memory used by the pair-wise and random key predistribution schemes, and also to limit the communication load

because of the KDC schemes. However, the difficulties in using a sensor node as the trusted third party are: the trusted intermediary can easily become compromised; key sizes in sensors nodes are not large; sensor networks may only accept authenticated messages, so may not have access to an encryption algorithm.
It should be noted that none of the above sensor key establishment schemes can handle authentication when the third party is compromised. Also, if the KDC is a sensor or controller node, then there is a higher likelihood that it can be compromised.

3.2 Security for the KDC

Several researchers have addressed security of the controller nodes and/or the base station. SIA (Przydatek et al., 2003) 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 (Wagner, 2004). Another way to protect results is to use a witness node mechanism (Zhu et al., 2003).
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 (Deng et al., 2004). Hop-by-hop re-encryption of each packet's header and data fields is designed to change the presentation of a packet so that it cannot be used to trace the direction toward or away from the base station. Uniform rate control is advised so that traffic volume nearer the base station is undifferentiated from traffic farther from the base station. Time decorrelation between packet arrivals and departures further increases the difficulty of tracing packets.
However, ensuring that the authentication services are not hindered by a compromised or broken controller node or base station presents different challengers. 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.

3.3 Limitations and concerns

When WSNs are deployed in battlefields or developed to monitor national security, they have a likelihood of becoming the target of adversaries. In an open environment, an individual server or an intermediate sensor node may not be completely and permanently trustworthy. 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, needs to be found.
For performance reasons, sensors use small keys. However, small keys give an attacker a greater opportunity to compromise a sensor node. Thus, small keys will require frequent refreshing. Old keys can be used to generate new keys. But, if the old key is compromised than the new key can easily be compromised. Many existing sensor protocols concentrate on the initial key distribution but do not have a secure mechanism available to update the keys. Keys in sensor networks are usually 64 bits in size, and they may become easily compromised. Sensors that are short–lived may not require that cryptographic keys be updated, however any sensors that exist for an extended amount of time will require updating of keys.

4. Traditional reliable key establishment protocols

Many key establishment protocols assume that server is never compromised. In a sensor environment, this may not be always true. A method alleviate the problem with a compromised server compromising the entire system is to use multiple server key establishment protocols. Boyd and Mathuria have produced a survey of key establishment protocols using multiple servers (Boyd & Mathuria, 2003) in traditional networks. In their survey, two multiple server protocols were listed: Gong's multiple server protocol (Gong, 1993), and the Chen–Gollmann–Mitchell protocol (Chen et al., 1995). However, this survey did not take into account the unique nature 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 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.

Protocol 1 Gong's simplified multi-server protocol

$M1$	$A \rightarrow B:$	$A, B, N_A, \{A, B, x_1, cc(x)\}_{K_{A1}}, \ldots, \{A, B, x_n, cc(x)\}_{K_{An}}$
$M2$	$B \rightarrow S_i:$	$A, B, N_A, N_B, \{A, B, x_i, cc(x)\}_{K_{Ai}}, \{B, A, y_i, cc(y)\}_{K_{Bi}}$
$M3$	$S_i \rightarrow B:$	$\{B, N_A, y_i, cc_i(y)\}_{K_{Ai}}, \{A, N_B, x_i, cc_i(x)\}_{K_{Bi}}$
$M4$	$B \rightarrow A:$	$\{B, N_A, y_1, cc_1(y)\}_{K_{A1}}, \ldots, \{B, N_A, y_n, cc_n(y)\}_{K_{An}}, \{N_A\}_{K_{AB}}, N_B$
$M5$	$A \rightarrow B:$	$\{N_B\}_{K_{AB}}$

A simplified version of Gong's original multiple server protocol is described in (Boyd & Mathuria, 2003). We describe this version as shown in *Protocol 1*. One of the main features of this protocol is that the nodes, A and B, choose the keying material while the n servers, $S_1, S_2, \ldots, S_n$, act as key translation centers that allow keying material from one node to be made available to the other. Initially A shares a long-term key K_{Ai} with each server S_i, and similarly B shares K_{Bi} with S_i. Node A has split the key x into $x_1, x_2, \ldots, x_n$ and node B has split the key y into $y_1, y_2, \ldots, y_n$. The session key is defined as $K_{AB} = h(x, y)$ where h is a one–way function. The protocol sends a total of $2n + 3$ messages.
To prevent compromised servers from disrupting the protocol, A and B form a cross-checksum for all the shares. The cross-checksum for x is shown in Equation (1).

$$cc(x) = (h(x_1), h(x_2), \ldots, h(x_n)) \tag{1}$$

The $cc_i(x)$ (should be equal to $cc(x)$) and $cc_i(y)$ (should be equal to $cc(y)$) are the cross–checksums returned by server S_i. The node will give a credit point to the servers if their cross–checksum values are the same as the values obtained from the majority of servers. When all the checks are complete, B retains the value x_j with the most credit points.
The major problem with this protocol is the size of the messages. The message sizes of $M1$ and $M4$ in the Gong multi–server protocol are of $O(n^2)$. Message $M5$ is $O(1)$, while $M2$ and $M3$ are of $O(n)$. A message size of $O(n^2)$ is not desirable in a sensor network. Another problem 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 for x and y). So for small values of n, the message sizes themselves will be very large for a sensor network.
The second multiple server protocol we consider is the Chen et al. multiple server protocol as shown in *Protocol 2*. One of the main features 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 protocol sends a total of $2n + 4$ messages.

Protocol 2 Chen-Gollmann-Mitchell multi-server protocol

$M1$	$A \rightarrow B:$	A, B, N_A
$M2$	$B \rightarrow S_i:$	A, B, N_A, N_B
$M3$	$S_i \rightarrow B:$	$\{B, N_A, K_i\}_{K_{Ai}}, \{A, N_B, K_i\}_{K_{Bi}}$
$M4$	$B \rightarrow A:$	$\{B, N_A, K_1,\}_{K_{A1}}, \ldots, \{B, N_A, K_n\}_{K_{An}}, cc_B(1), \ldots, cc_B(n)$
$M5$	$A \rightarrow B:$	$cc_A(1), \ldots, cc_A(n), \{B, N_B, N'_A\}_{K'_{AB}}$
$M6$	$B \rightarrow A:$	$\{A, N'_A, N_B\}_{K'_{AB}}$

The cross-checksum used in this protocol is different from the one used in the Gong multi–server protocol. The sensor node B calculates the $cc_B(i)$ as shown in Equation(2).

$$cc_B(i) = \{h(K_1), h(K_2), \ldots, h(K_n)\}_{K_i}, \ \forall i \in (1, \ldots, n) \tag{2}$$

To prevent A or B imposing the session key, the choice of $h()$ is limited; for example, it cannot be an exclusive–or–operation. If B doesn't receive any message from server S_j then $cc_B(j)$ is an error message, and $h(K_j)$ is replaced by an error message in the calculation of the other $cc_B(i)$ values. When A receives the checksums, A will first decrypt the values and compares the values with its own calculations of the cross-checksums. The valid K_i secrets are retained for the majority of i values and others are discarded. The session key K_{AB} is defined to be the hash of all the good K_i values concatenated, as shown in Equation (3).

$$K_{AB} = h(K_j, \ldots, K_m) \tag{3}$$

The messages $M4$ and $M5$ in the Chen et al. multi–server protocol have a computational complexity of $O(n^2)$. While messages $M2$ and $M3$ have $O(n)$, messages $M1$ and $M6$ have $O(1)$ computational complexity.

This protocol encounters similar problems as the Gong multi–server protocol with regard to the size of the messages. Once again, several messages are of $O(n^2)$ in size. With the cross–checksums containing the outputs of a one–way function where the inputs are key values, once again the size of the output will need to be large. The cross-checksums in this protocol are encrypted instead of only requiring a hash algorithm. However, the Chen et al. multi–server protocol is considerably more efficient with regard to the size of the messages.

Another aspect of the multiple server protocols is the creation of the new key K_{AB}. The nodes A and B retrieve the new key by using a secret sharing mechanism such as the one defined in (Shamir, 1979). Secret sharing is a mechanism allowing the owner of a secret to distribute *shares* amongst a group. Individual shares or a small number of shares are no help in recovering the secret. The n shares are distributed, such that any set of t (or more) shares is sufficient to obtain the secret. The most well-known threshold scheme uses polynomial interpolation.

When polynomial interpolation is used in cryptographic applications the field is typically $\mathbb{Z}_p$, the field of integers modulo p, for some prime p. To share a secret $s \in \mathbb{Z}_p$ in the (t, n) threshold scheme, the dealer generates a polynomial of degree $t - 1$:

$$f(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1} \tag{4}$$

The coefficients are randomly chosen in $\mathbb{Z}_p$ except for $a_0 = s$. The shares are values of $f(x)$ with $1 \leq x \leq n$. If t shares are known s can be recovered. For example, if $f(1), f(2), \ldots, f(t)$ are known then:

$$s = \sum_{i=1}^{t} f(i) \prod_{i<j\leq t} \frac{j}{j-i} \tag{5}$$

Given any t points on the polynomial (excluding the value of 0), the value for $f(0)$ can be obtained.
The time complexity to compute n shares is $O(nt)$. The time complexity to recover a_0 is $O(t \log_2 t)$ (Gong, 1993). Having such a scheme or similar scheme in a sensor node will consume significant amount of resources in an already resource constrained environment.
The existing multiple server protocols are therefore not suitable for a sensor network environment. An ideal solution with the desired characteristics requires innovative multiple server protocols, specifically designed for the sensor environment.

5. Multiple server protocols for sensor environments

Three multiple server protocols have been developed specifically for the sensor environment (Singh & Muthukkumarasamy, 2008). In this chapter we will label the three Multiple Server Protocols (MSP) as Singh et al. MSP1, Singh et al. MSP2, and Singh et al. MSP3. The multiple authentication server protocols Singh et al. have developed are based on the concept of the Boyd four–pass protocol (Boyd & Mathuria, 2003). These will maintain similar security characteristics as that of the centralized authentication protocol, as shown in the Boyd four–pass protocol. Because of the severe resource constraints which exist in sensor nodes, a multiple authentication server protocol should have low computational complexity in both time and space. A multiple server authentication protocol in a sensor environment should have the following characteristics:

- Small computation overhead,
- Minimal number of messages,
- Sensor nodes should not be relied upon to generate good randomness,
- The new key should be fresh,
- There should be key confirmation by both nodes.

The next several sections will go into detail about the three different sensor multiple server protocols.

5.1 Singh et al. Multiple Server Protocol 1

The Singh et al. MSP1 is an efficient multiple server protocol specifies n servers. The protocol has the following message flows. 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, as shown in *Protocol 3* (Singh & Muthukkumarasamy, 2006).
The Singh et al. MSP1 provides key authentication, key freshness and key confirmation, using multiple authentication servers. In the Singh et al. MSP1 protocol, the following constructs are used:
$AUTH_{Ai} = [A, B, K_i]_{K_{AS_i}}$
$MASK_{Ai} = [[AUTH_{Ai}]]_{K_{AS_i}}$
$AUTH_{Bi} = [A, B, K_i]_{K_{BS_i}}$
$MASK_{Bi} = [[AUTH_{Bi}]]_{K_{BS_i}}$

Protocol 3 Singh et al. MSP1

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

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 $cc_B(i)\ \forall i \in 1, \ldots, n$.

$$cc_B(i) = \begin{cases} [K_i]_{K_i} & \text{if valid,} \\ EM & \text{otherwise} \end{cases} \tag{6}$$

Where EM is an error message; an example will be the value zero. There is a remote chance a valid case may be zero. If the valid value is zero, the server needs to 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 the nodes compare the cross checksum for $cc_A(i)$ and $cc_B(i)$.

$$cc_A(i) = \begin{cases} cc_B(i) = [K_i]_{K_i} & \text{if valid,} \\ EM & \text{otherwise} \end{cases} \tag{7}$$

After the comparison of the entire cross checksums, a set of valid keys $V_1, \ldots, V_m$ should remain. The creation of K_S is defined as follows.

$$K_S = V_1 \oplus \ldots \oplus V_m \tag{8}$$

Where V_i is the i^{th} valid key given by a server, 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 and calculates K_S.

5.1.1 Analysis of singh et al. MSP1

The Singh et al. MSP1 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 protocol only encrypts random information. If the encryption cipher uses an IV value (such as RC5 and SKIPJACK currently used in TinyOS (TinyOS, 2007)) then we can use a constant IV value. However, the constant IV value chosen for Singh et al. MSP1 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 MACs have vulnerabilities when the message sizes are variable. All of the 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. Because of 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: $M1_i = 2a_0 + a_1$, $M2_i = 2a_0 + a_1 + a_2 + a_3$, $M2'_i = a_0 + a_2 + a_3$, $M3 = a_1 + na_3$, $M4 = (n+1)a_3$, and $M5 = a_3$. However, there are n messages of type $M1$, $M2$ and $M2'$.

5.2 Singh et al. Multiple Server Protocol 2

The Singh et al. MSP1 protocol was investigated and extended further to reduce the large number of messages sent through the WSN. Thereby the Singh et al. MSP2 was developed as shown in *Protocol 4*. The assumption is made that the servers can communicate through a different network, other than the low bandwidth WSN used by the sensor nodes. For instance, the GNOME platform (GNOME, 2006) also has an Ethernet connection it can use as a high speed backbone network to communicate with other GNOME machines. In the Singh et al. MSP2 protocol, the sensor node A only sends one message to a server, denoted as S_1. Server S_1 then gathers all the required information from the other servers through the server network (rather than the sensor network). Server S_1 concatenates the information and sends it to sensor node B. The list of servers may either be a static list, known by the sensor nodes, or it may be a list based on the trustworthiness of the servers.

5.2.1 Security analysis

If S_1 becomes malicious, there are a number attacks that the server can try. The simplest attack is a denial of service, where the server will not gather any extra information from other servers, or doesn't respond to the sensor node B. If sensor node A does not receive a response from B in a required amount of time, then it should try S_2.

Protocol 4 Singh et al. MSP2 protocol

$M1$	$A \rightarrow S_1 :$	A, B, N_A
$M2$	$S_1 \rightarrow S_i :$	A, B
$M3$	$S_i \rightarrow S_1 :$	$MASK_{Bi}, AUTH_{Bi} \oplus K_i, MASK_{Ai}, AUTH_{Ai} \oplus K_i$
$M4$	$S_1 \rightarrow B :$	$S_1, MASK_{B1}, AUTH_{B1} \oplus K_1, \ldots, S_n, MASK_{Bn}, AUTH_{Bn} \oplus K_n, N_A, A$
$M4'$	$S_1 \rightarrow A :$	$MASK_{A1}, AUTH_{A1} \oplus K_1, \ldots, MASK_{An}, AUTH_{An} \oplus K_n$
$M5$	$B \rightarrow A :$	$cc_B(1), \ldots, cc_B(n), N_B$
$M6$	$A \rightarrow B :$	$cc_A(1), \ldots, cc_A(n), [N_B]_{K_{AB}}$
$M7$	$B \rightarrow A :$	$[N_A]_{K_{AB}}$

Another possible attack S_1 can try is to forge the MACs to create legitimate messages from the other servers. As discussed earlier, it is unreasonable to assume a server can forge a message, let alone $2t$ messages.

If there are t malicious servers, then S_1 can contact those servers and not involve the trusted servers. However, the Singh et al. MSP1 is also vulnerable to t malicious servers.

If contacting only one server is still a concern, then a higher level reputation based framework (Ganeriwal & Srivastava, 2004) on top of existing authentication protocols to make sure the nodes collaborate with only trustworthy base stations. Another solution is that A can send the first message to p servers, where $p < n$, and each server is allocated servers from which to obtain information (however, this will require more code and logic within the sensor applications).

5.2.2 Cost/complexity analysis

The Singh et al. MSP2 decreases the number of packets sent by A. The following equations are used to define how many bytes are sent for each message, $M1 = 2a_0 + a_1$, $M2_i = 2a_0$, $M3_i = 2a_2 + 2a_3$, $M4 = a_0 + a_1 + na_2 + na_3$, $M4' = na_2 + na_3$, $M5 = a_1 + na_3$, $M6 = (n+1)a_3$, and $M7 = a_3$.

Although the Singh et al. MSP2 decreased the number of messages sent by A, it is not as reliable as Singh et al. MSP1. If S_1 is down, either A will need to detect this and try S_2, or S_1 itself will need to be a clustered system. The drawback of a true replicated clustered system is that it is more likely that the system can be compromised, since there are more machines available for an adversary to attack. Another problem if S_1 becomes malicious is that it may not return any information from the other servers. Once again, A can try other servers.

5.3 Singh et al. Multiple Server Protocol 3

The third multiple server protocol is the Singh et al. MSP3. The third protocol was designed for the case where if the key K_S becomes compromised, and the long–term keys K_{AS} and K_{BS} have not changed. The key K_S is important because it can be used in the future to create or renew a session key between A and B. The multiple server scenarios have strengthened the security between the nodes and the KDC. Compromised keys between a node and one (or more) servers, does not affect the security of the protocol. An adversary can replay previous (portion of) messages to force the sensor nodes to use the same K_S as before.

The Singh et al. MSP3 protocol was designed to remove this problem. The advantage of the Singh et al. MSP3 over Singh et al. MSP2 is that if K_S is ever compromised and the long term keys K_{AS} and K_{BS} have not changed, then the protocol can be run again safely. The calculation for $AUTH$ was also done so that a replay attack is not possible.

$AUTH_{Ai} = [A, B, N_A, N_B]_{K_{Ai}}$
$AUTH_{Bi} = [A, B, N_A, N_B]_{K_{Bi}}$

Protocol 5 Singh et al. MSP3

$M1$	$A \rightarrow B:$	A, N_A
$M2$	$B \rightarrow S_1:$	A, B, N_A, N_B
$M3$	$S_1 \rightarrow S_i:$	A, B, N_A, N_B
$M4$	$S_i \rightarrow S_1:$	$AUTH_{Bi}, MASK_{Bi} \oplus K_i, AUTH_{Ai}, MASK_{Ai} \oplus K_i$
$M5$	$S_1 \rightarrow B:$	$AUTH_{B1}, MASK_{B1} \oplus K_1, \ldots, AUTH_{Bn}, MASK_{Bn} \oplus K_n$
$M5'$	$S_1 \rightarrow A:$	$AUTH_{A1}, MASK_{A1} \oplus K_1, \ldots, AUTH_{An}, MASK_{An} \oplus K_n$
$M6$	$B \rightarrow A:$	$cc_B(1), \ldots, cc_B(n), N_B$
$M7$	$A \rightarrow B:$	$cc_A(1), \ldots, cc_A(n), [N_B]_{K_{AB}}$
$M8$	$B \rightarrow A:$	$[N_A]_{K_{AB}}$

The Singh et al. MSP3 has one more message than Singh et al. MSP2, however, Singh et al. MSP2 has a larger message. This is because of the need to send A and N_A in $M4$, whereas in Singh et al. MSP3 the message $M5$ does not need to send this extra information. The maximum size message in Singh et al. MSP2 can be decreased if another message $M1'$ is sent as shown in Equation (9).

$$A \rightarrow B : A, N_A \tag{9}$$

5.4 Performance analysis

When looking at authentication algorithms, a number of different aspects need to be taken into consideration. Apart from the security properties, such as key establishment, key freshness, and key confirmation, a number of performance aspects should also be looked at. In the past, symmetric key algorithm performance was categorized by the number of messages and the number of rounds. However, in sensor networks these are not true indicators of the performance of an algorithm. Other measures such as computational costs, number of bytes and packets sent and received, the amount of memory consumed are also important in a sensor environment.

The computational costs to the scheme arises because of the encryption, decryption and integrity checking of the keys generated by the servers. There is also the generation of the K_{AB} by both A and B. An inefficient aspect of the Boyd four–pass protocol is the need to decrypt the message before the integrity check is done. The protocols have a similar restriction if bits were changed in the $MASK$ then it will not be known until both $MASK$ and $AUTH$ were calculated. Bogus messages injected by a third party cannot be detected using any computationally efficient methods. The Bellare–Rogaway uses the encrypted messages when performing the integrity check, rather than the decrypted message. A very simple modification to the Boyd four–pass protocol removes this minor limitation, as shown in the *Protocol 6*.

Protocol 6 Boyd protocol integrity change

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

The size of the messages does not change, and there is the added benefit of the integrity check performed before the decryption of the message. This technique was also extended to be used

in Singh et al. protocols (however, encryption algorithms are then required to be implemented on the sensor nodes). However, it has been shown that the communication costs are almost an order of magnitude more than the computational costs in security systems (Perrig et al., 2001).
The communication costs will be heavily dependent on the topology of the network. Also, different protocols have different communication overheads for the sensor nodes, and the servers. The communication overheads of the different protocol is examined in detail in Section 6.

6. Comparison

We now compare Singh et al. protocols with the Chen et al. protocol and the Gong protocol. The computational complexity of the existing multiple server protocols is greater since 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 use a threshold scheme such as Shamir's scheme (Shamir, 1979) to recover the key, the computational complexity will be $O(t \log_2 t)$. Whereas Singh et al. protocols use a simple exclusive–or function (as describe in Equation (8)) to recover the key, which has 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. The advantage of using a full–blown key–share threshold scheme, is that t servers can calculate the new session key between A and B if they are the only servers involved in the protocol. However, for performance reasons we have not placed the same restriction on Singh et al. multiple server protocols.
Another comparison is the communication cost of Singh et al. protocols compared with the Chen et al. and Gong protocols. For simplicity, we will assume that the output of the one–way function used to calculate the cross–checksums in the existing multiple server protocols is the same size as the integrity function used in Singh et al. protocols. Although, as described earlier, for security reasons the one–way function in the existing multiple server protocols will need to be larger.
We will compare the total number of bytes sent by sensor node A for each of the existing multiple server protocols, and Singh et al. MSP1 and Singh et al. MSP2. However, similar calculations as the ones shown here can be used to calculate the impact on the base stations and sensor B.
The total number of bytes sent by sensor A in the Gong multi–server protocol is $n^2a_3 + 2na_0 + na_2 + 2a_0 + 2a_1 + a_3$, which has a complexity of $O(n^2)$. The number of bytes sent by sensor A in the Chen et al. multi–server protocol is $n^2a_3 + na_3 + 3a_0 + 3a_1 + a_3$. The above two cases have a complexity of $O(n^2)$. However, the number of bytes sent by sensor A in Singh et al. MSP1 and Singh et al. MSP2 are $2na_0 + na_1 + (n+1)a_3$ and $2a_0 + a_1 + (n+1)a_3$, respectively. Both of these messages have a complexity of $O(n)$.
Figure 2 compares the total number of bits sent out in the entire network, relative to the number of servers used. The Gong protocol is the most expensive, followed by the Chen protocol. Also, the protocols consider for this graph send all their messages over the sensor network, rather than sending some of the messages over a faster backbone network.
The existing multiple server protocols have message sizes of size $O(n^2)$, whereas Singh et al. protocols are $O(n)$. It should be noted that the existing multiple server protocols do have added functionality, where the trusted servers are able to calculate the key K_{AB}. However, for

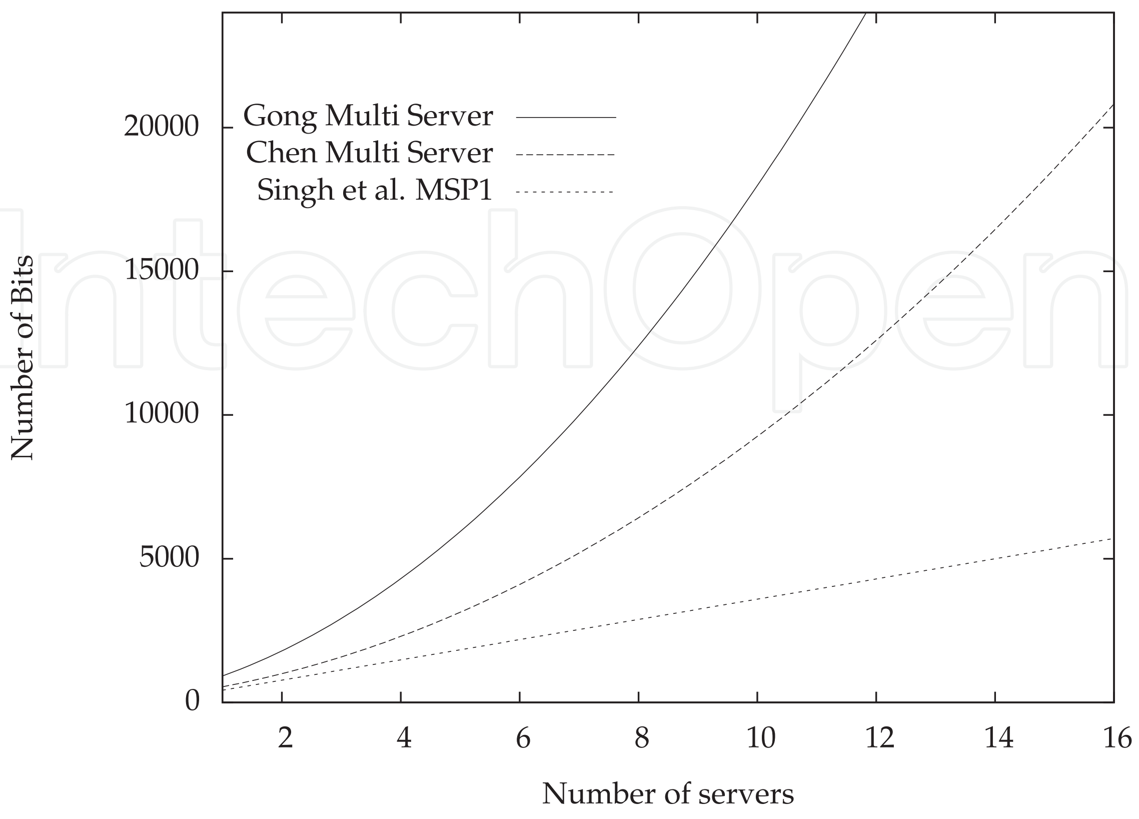

Fig. 2. Total Number of bits sent by different Protocols

performance reasons we have not placed the same restriction on Singh et al. multiple server protocols.

Table 2 describes the costs of the Singh et al. MSP1, where n is the number of servers. The table shows that in the protocol sensor node A transmits the most amount of messages. However, even though sensor node B only transmits two messages, the size of one of its messages is based on the number of servers. Hence, the number of bits sent by sensor node B is similar to the number of bits sent by sensor node A. The messages and size of messages transmitted by a server S_i is constant and independent on the number of servers in the protocol. The number of messages received and the number of bits received is very similar when comparing sensor node A and sensor node B. The message and number of bits received by a server S_i is insignificant compared to all the other messages.

Node	A	B	S_1	S_i
Msgs Tx	$n+1$	2	2	2
Msgs Rx	$n+2$	$n+1$	1	1
Bits Tx	$72n+32$	$32n+40$	248	248
Bits Rx	$144n+40$	$168n+32$	40	40

Table 2. Performance Figures for Singh et al. MSP1

Table 3 describes the costs of the Singh et al. MSP2, where n is the number of servers. The table shows that in the protocol the server S_1 transmits the most amount of messages. However, even though sensor nodes A and B only transmits several messages, some of their message sizes is based on the number of servers. Hence, the number of bits sent by sensor nodes A

and B is still linear. The messages and size of messages transmitted by a server S_i (except S_1) is constant and independent on the number of servers in the protocol. The number of messages received and the number of bits received is very similar when comparing sensor node A and sensor node B. The message and number of bits received by a server S_i (except S_1) is insignificant compared to all the other messages. The number of bits received by S_1 is linear in nature.

Node	A	B	S_1	S_i
Msgs Tx	2	2	$n+1$	1
Msgs Rx	3	2	n	1
Bits Tx	$32n+72$	$32n+40$	$232n-40$	192
Bits Rx	$128n+40$	$128n+56$	$192n-152$	40

Table 3. Performance Figures for Singh et al. MSP2

Table 4 describes the costs of the Singh et al. MSP3, where n is the number of servers. The protocol has many of the same features as Singh et al. MSP2. For instance, the server S_1 transmits the most amount of messages. The sensor nodes A and B only transmits several messages, some of their message sizes is based on the number of servers. Hence, the number of bits sent by sensor nodes A and B is still linear. However, the message sizes are slightly larger when compared against the previous protocol because of the extra security. The messages and size of messages transmitted by a server S_i (except S_1) is constant and independent on the number of servers in the protocol. The number of messages received and the number of bits received is very similar when comparing sensor node A and sensor node B. As in the previous protocol, the message and number of bits received by a server S_i (except S_1) is insignificant, however, it is slightly larger when compared to the previous protocol. As in the previous protocol, the number of bits received by S_1 is linear in nature.

Node	A	B	S_1	S_i
Msgs Tx	2	3	$n+1$	1
Msgs Rx	3	3	n	1
Bits Tx	$32n+72$	$32n+88$	$240n-48$	192
Bits Rx	$128n+40$	$128n+72$	$192n-144$	48

Table 4. Performance Figures for Singh et al. MSP3

The Singh et al. MSP1 puts a larger emphasis on communication costs on sensor node A. The other two protocols put greater emphasis on the server nodes. The extra message in Singh et al. MSP3 does not affect sensor node A, and for a large number of servers it is insignificant.
The number of cryptographic operations performed by each node in the three protocols shown in the table, is the same for each of the protocols. The number of $AUTH$ calculations is $2n+4$ for both A and B. The number of $MASK$ calculations is $2n$ for both A and B. For S_i the number of $AUTH$ and $MASK$ calculations is two. The only nodes that need to create good random numbers are each of the servers S_i and they only create one for each run through of the protocol. Both N_A and N_B can be a simple counter.
Another comparison is the number of packets sent by each of Singh et al. multiple server protocols, as shown in Figure 3. The comparison is done over two different network topologies. The servers either communicate over the sensor network or they have a separate network to communicate over. The Singh et al. MSP1 has the same cost over both network topologies, since every message interacts with a sensor node, and the servers do not need to

communicate with one another. Because of the larger number of messages sent by the Singh et al. MSP2 and Singh et al. MSP3, it is natural for the number of packets to be more than Singh et al. MSP1, when all the messages have to be sent on the same sensor network. As the number of servers increases, the difference (when on the same network) between Singh et al. MSP1 and the other two protocols does not dramatically increase. The other two protocols have the advantage of concatenating messages, which the original multiple server protocol does not have. If the servers can communicate over a different network, the number of packets sent by Singh et al. MSP2 and Singh et al. MSP3 stay very close to one another, and sometimes they are the same. This is because of the first protocol having larger message sizes, and the messages need to be segmented sooner than they do in the other protocol. The number of packets sent on the energy–constrained sensor network is significantly larger if using Singh et al. MSP1, when the servers are on different networks.

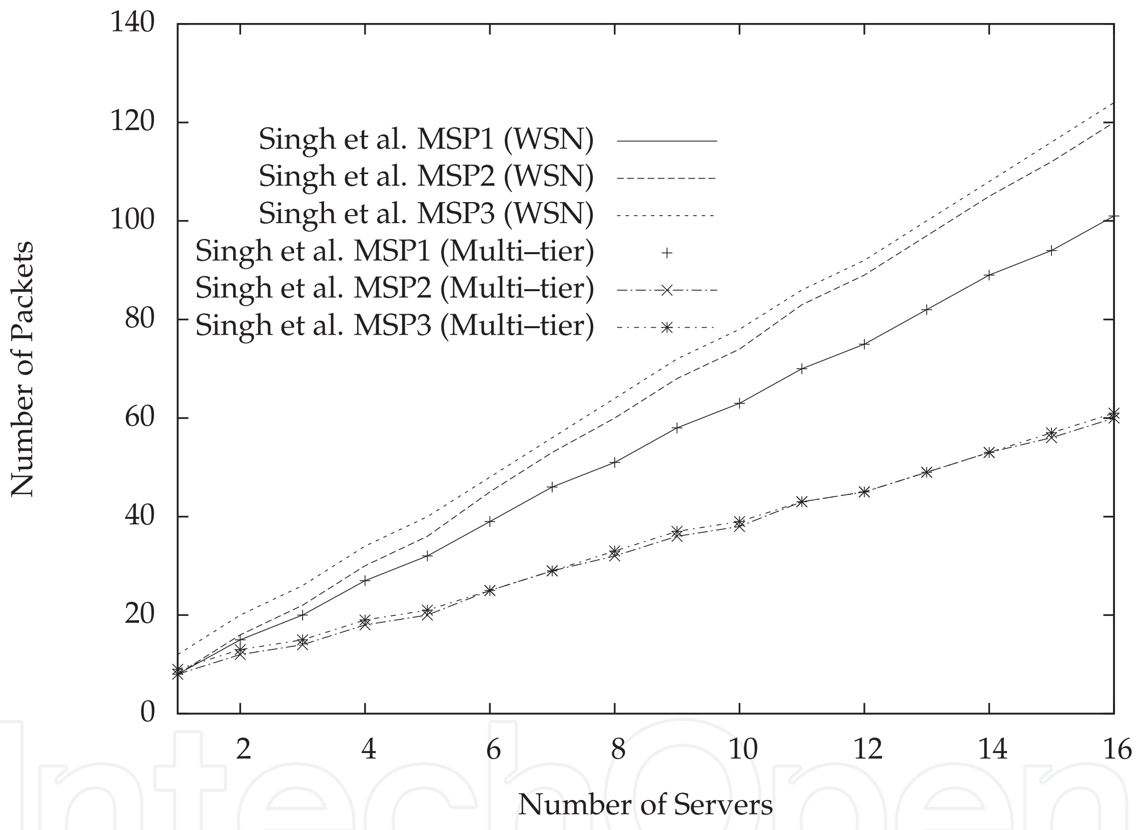

Fig. 3. Number of packets sent by different Protocols

TinyOS has a seven byte overhead when sending a single packet. When comparing the bits sent by each of the protocols, we included the packet overhead. We once again cover the case of two network topologies, as shown in Figure 4. The Singh et al. MSP1 has the same cost over both networks. When comparing the protocols, if the servers can communicate over a different network, there is virtually no difference between Singh et al. MSP2 and Singh et al. MSP3. However, the extra overhead of sending data to the server S_1, which then sends the same data down to the sensor nodes, causes both protocols to have significantly more overhead when run over the same network.

Table 5 compares the existing multiple server protocols with some of Singh et al. protocols. In the table we look at the protocol reliability, and compare the protocols when the servers use

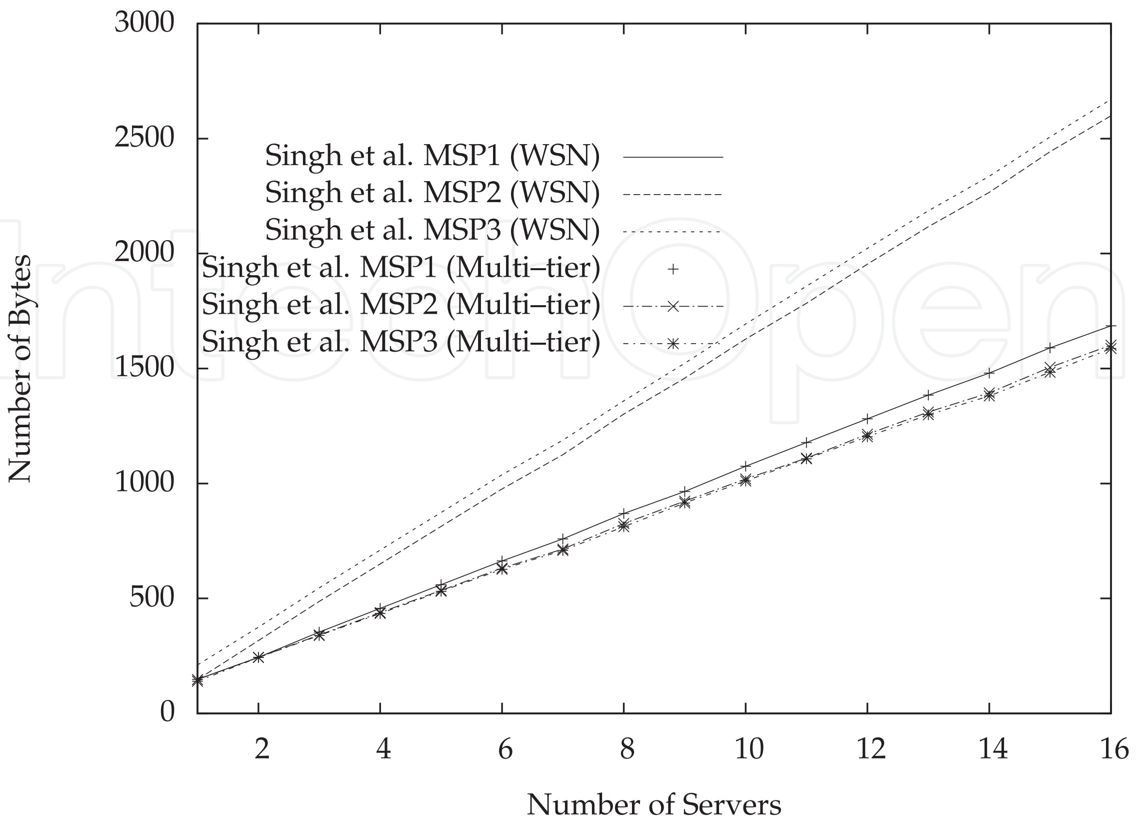

Fig. 4. Number of bytes sent by different Protocols

the WSN or a different network. We also evaluate the robustness of the protocols based on security attacks in any environment.

Protocol	Properties of the Protocol			
	Protocol Reliability	Efficiency of WSN	Efficiency of multi–tiered	Security
Gong	Excellent	Bad	Bad	Excellent
Chen et al.	Excellent	Bad	Bad	Excellent
Singh et al. MSP1	Excellent	Good	Average	Good
Singh et al. MSP2	Average	Average	Excellent	Good
Singh et al. MSP3	Average	Average	Excellent	Excellent

Table 5. Comparison of Multiple Server Protocols

After the detailed analysis of the protocols and the comparison with existing protocols, we conclude that Singh et al. MSP1 should be used in situations where the environment is concerned with reliability. If reliability isn't a major concern and if the servers can communicate over a different network then Singh et al. MSP2 should be used. If the environment is not multi–tiered then the most reliable protocol (Singh et al. MSP1) is also the most efficient. If concerned about the security of K_S then Singh et al. MSP3 should be used. In a complex environment the need for reliability and security puts extra costs on the protocols.

7. Conclusions

Key establishment protocols, without the assumption of trusting an individual authentication server, are needed in sensor environments where clients cannot trust individual servers. We examined the existing multiple server protocols developed for traditional networks, and investigated the problems using those protocols for WSNs. We then three multiple server protocols for sensor networks and a complex sensor system containing multi–tiered networks, and provided a detailed analysis and comparison of each the protocols.

The multiple server protocols designed for a sensor environment removed the requirement for the servers to know the keys between each of the sensor nodes, and thus helping to limit the message sizes to $O(n)$. The protocols 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 that our protocols are flexible enough for them to be used with almost any cryptographic primitive and in a range of environments.

8. References

Blundo, C., Santis, A. D., Herzberg, A., Kutten, S., Vaccaro, U. & Yung, M. (1993). Perfectly–secure key distribution for dynamic conferences, *Lecture Notes in Computer Science* 740: 471–486.
URL: *citeseer.ist.psu.edu/blundo95perfectlysecure.html*

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

Boyd, C. & Mathuria, A. (2003). *Protocols for Authentication and Key Establishment*, Springer Berlin / Heidelberg.

Bulusu, N. (2005). Introduction to wireless sensor networks, *in* N. Bulusu & S. Jha (eds), *Wireless Sensor Networks: A Systems Perspective*, Artech House.

Chan, H. & Perrig, A. (2005). PIKE: Peer intermediaries for key establishment in sensor networks, *Proceedings of IEEE Infocom*, IEEE Computer Society Press.

Chan, H., Perrig, A. & Song, D. (2003). Random key predistribution schemes for sensor networks, *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, IEEE Computer Society, Washington, DC, USA, p. 197.

Chen, L., Gollmann, D. & Mitchell, C. J. (1995). Key distribution without individual trusted authentication servers, *8th IEEE Computer Security Foundations Workshop*, IEEE Computer Society Press, pp. 30–36.

Crossbow (2006). Crossbow, URL: *http://www.xbow.com/*.

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

Deng, J., Han, R. & Mishra, S. (2005). Sensor–network security, privacy, and fault tolerance, *in* N. Bulusu & S. Jha (eds), *Wireless Sensor Networks: A Systems Perspective*, Artech House.

Ganeriwal, S. & Srivastava, M. B. (2004). Reputation-based framework for high integrity sensor networks, *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, ACM Press, New York, NY, USA, pp. 66–77.

GNOME (2006). A controller node, URL: *http://cmlab.rice.edu/projects/sensors*.

Gong, L. (1993). Increasing availability and security of an authentication service, *IEEE Journal on Selected Areas in Communications* 11(5): 657–662.

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

Leighton, F. T. & Micali, S. (1994). Secret-key agreement without public-key cryptography, *CRYPTO '93: Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, Springer-Verlag, London, UK, pp. 456–479.

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

MANTIS (2006). The mantis contoller node, URL: *http://mantis.cs.colorado.edu/*.

Medusa (2006). The medusa mk–2 controller node, URL: *http://nesl.ee.ucla.edu/*.

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

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

Shamir, A. (1979). How to share a secret, *Commun. ACM* 22(11): 612–613.

Singh, K., Bhatt, K. & Muthukkumarasamy, V. (2006). Protecting small keys in authentication protocols for wireless sensor networks, *Proceedings of the Australian Telecommunication Networks and Applications Conference*, Melbourne, Australia, pp. 31–35.

Singh, K. & Muthukkumarasamy, V. (2006). A minimal protocol for authenticated key distribution in wireless sensor networks, *ICISIP '06: Proceedings of the 4th International Conference on Intelligent Sensing and Information Processing*, IEEE Press, Bangalore, India, pp. 78–83.

Singh, K. & Muthukkumarasamy, V. (2008). Performance analysis of proposed key establishment protocols in multi–tiered sensor networks, *Journal of Networks* 3(6).

TinyOS (2007). An operating system for sensor motes, URL: *http://www.tinyos.net/*.

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

Watro, R., Kong, D., fen Cuti, S., Gardiner, C., Lynn, C. & Kruus, P. (2004). Tinypk: securing sensor networks with public key technology, *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, ACM Press, New York, NY, USA, pp. 59–64.

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