

Implementation and Analysis of Sensor Security Protocols in a Home Health Care System

Kalvinder Singh

IBM, Australia Development Lab, and Griffith University
Gold Coast, Australia
Email: kalsingh@au.ibm.com

Vallipuram Muthukumarasamy

School of Information and Communication Technology
Griffith University
Gold Coast, Australia
Email: v.muthu@griffith.edu.au

Abstract—A Wireless Sensor Network can be used in a home health care system to monitor the elderly or patients with chronic diseases. The quality of security of the home health care system is an important requirement. If the security services or servers malfunction then the system itself may become compromised. We show that with multi-server protocols, even if one or more servers become unavailable or untrustworthy, it may still be possible for the sensor nodes to establish a good session key. We compare and contrast different multi-server protocols. We also show which protocols are more suited for our system. The protocols were implemented in TinyOS and run on mica2 motes. The time elapsed, complexity of the code, and memory requirements are analysed in detail. We show that a symmetric key implementation has advantages over an asymmetric implementation.

Keywords—intelligent sensors; security

I. INTRODUCTION

The aging population and the increase of chronic diseases have placed an immense financial burden on health services. A home health care system [1], with both body and external sensors can be used to help reduce the costs. Sensors can be used to remotely monitor elderly patients suffering from chronic diseases and allow them to have relatively independent lives. Sensor networks are inherently complex. For instance, blood pressure increasing due to exercise is normal. However, increase in blood pressure while at rest could mean a serious medical condition. Sensors may not just measure physiological values, but also body motions, which can lead to a number of different sensors needing to communicate with each other. As the number of heterogeneous sensors increases, so will the complexity of interactions between the sensors.

The quality of security is a very important criteria of the home health care system. Security services should be available to facilitate the smooth operation of the system even if a component malfunctions or is compromised. As the numbers of components in a home health care system increases so will the likelihood of failure. The system should be designed so that the failure of a component does not jeopardize the security of the system.

A home health care system [2] can have communication between sensors, a mobile phone and a health controller.

The health controller is a specialized device that is situated at home and communicates the necessary information to the hospital. Depending on the health risks and privacy concerns of the patient all of the information may not be transmitted to a hospital. For instance, cameras (home sensors) may only start recording if the body sensors detect that there may be a medical emergency, such as the patient lying horizontal in the kitchen. Surveillance software, such as S3 [3] can be used to detect if the patient is cleaning the kitchen, or getting something from the ground, or there is actually an emergency. If the software does detect an emergency, the hospital staff is notified; they examine the information, and decide on the best course of action. The mobile phone gives feedback to the patient about the condition of their body, as well as the status of the sensors. The mobile phone can notify the patient of any detected emergency, allowing the patient to report back a false alarm if one has occurred. The mobile phone can be replaced with a PDA or any other hand-held communication device.

Security threats in the system include: impersonation of user and service; modification of code and data; disclosure; denial of service; repudiation. Health information collected from sensors needs to be secured and in some countries (for example the USA) security is mandated [4]. Ensuring the quality of security services in a home health care system is difficult, mainly because of the number of various components, and each one with different characteristics. The home health care system's security services should be able to handle the failure of a single or multiple components. Singh et al. proposed three multi-server protocols [5] that were designed to handle failures in the authentication security service.

Singh et al. developed the protocols to use the differences in computing power, as well as differences in communication costs, to make the protocols more efficient. They address the use of small keys in their protocols. 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 was supplied.

We will show that the proposed protocols can also be used in the home health care system, and thus increase the quality of security. We will also compare other existing multi-server protocols with the proposed protocols. An implementation of each of the protocols is run in our micaz sensor environment and timing figures are compared.

II. PROBLEMS AND LIMITATIONS

The home health care system is a multi-tiered system, consisting of cheap sensor nodes and more expensive cameras. Sensor nodes may be motion detectors, and if they are triggered, they will notify the camera to start recording. Cameras need to be connected to a high bandwidth and throughput network. For instance, Axis has wireless cameras that can connect to a 802.11g access point [5]. The Omnicast system can handle up to 50000 cameras, where the bottleneck is in the network [5]. Cameras are also designed to handle inputs and supply outputs to external sensors. SensEye [6] is an example of a multi-tier camera, where smaller cheaper cameras, with low bandwidth requirements, are used to notify more expensive cameras with higher bandwidth requirements of an event. Different networks are used in a home health care system, and each network has different requirements.

One of the networks in the home health care system is the Wireless Sensor Network (WSN). 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.

Some sensor nodes are resource constrained, such as the Mica mote [7]. The Mica motes contain a 4 MHz processor with 512 KB flash memory and 4 KB of data memory. 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 Star-gate platform, the GNOME platform, the Medusa MK-2 platform, and the MANTIS platform [5]. 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.

A. 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.

Several researchers have addressed security of the controller nodes and/or the base station. SIA [8] 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 [9]. Another way to protect results is to use a witness node mechanism [10].

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 [11]. 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 challenges. 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. Another solution is to use multi-server protocols.

III. MULTI-SERVER PROTOCOLS

Boyd and Mathuria have produced a survey of the current key establishment protocols using multiple servers [12]. In their survey, two protocols were listed: Gong's multiple server protocol [13], and the Chen-Gollmann-Mitchell protocol [14]. However, this survey did not take into account the unique problems of a sensor environment. Singh et al.

proposed three multiple server protocols [5] for the 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. Gong Multi-Server Protocol

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 [15]. 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 protocol sends a total of $2n + 3$ messages, where n is the number of authentication servers. There is a total of five different message types, and two of those message types have a size of $O(n^2)$.

B. Chen et al. Multi-Server Protocol

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 protocol sends a total of $2n + 4$ messages, where n is the number of authentication servers. There is a total of six different message types, and two of those message types have a size of $O(n^2)$. 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.

C. Singh et al. PP3 Multi-Server Protocol

Singh et al. created multiple server protocols specifically for a WSN environment, where there was a multiple network tiers. Unlike the other multiple server protocols, their protocols have limitations on number of components that can calculate the final session key. The final session key can only be calculated by the two sensors that require it. The three multi-server protocols are Singh et al. PP3, Singh et al. PP4, and Singh et al. PP5 [5].

The multi-server protocol, Singh et al. PP3, 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 .

Unlike the Chen et al. and Gong et al. protocols, the Singh et al. PP3 generates two keys. The session key K_{AB} and the long lasting key K_S . The key K_S was designed so that future session keys can be created without the need to go through the entire multi-server protocol again.

D. Singh et al. PP4 Multi-Server Protocol

The protocol assumes 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 also has an Ethernet connection it can use as a high speed backbone network to communicate with other GNOME machines. In Singh et al. PP4, 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.

E. Singh et al. PP5 Multi-Server Protocol

The Singh et al. PP5 protocol is designed to strengthen the security over the Singh et al. PP4 protocol. The Singh et al. PP4 protocol under certain conditions suffers from a replay attack on the key K_S . The key K_S 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. There were environments where this was not a concern, where the keys between the sensors and the authentication servers were updated regularly, which removes any replay attack that was possible. However, in our health-care system, it is not guaranteed that the keys between sensors and authentication servers will be updated more frequently than the update of the key K_S . The Singh et al. PP5 protocol removes the replay attack, but does add an extra message. The advantage of the Singh et al. PP5 protocol over the Singh et al. PP4 protocol is that if K_S is ever compromised, then the protocol can be run again safely.

IV. IMPLEMENTATION

Singh et al. [5] compared the multi-server protocols using simulators to measure the number of packets and number of bits sent. However, the simulation was not able to compare both the packets and the bits sent at the same time. One comparison only examined the number of packets. The comparison assumed that each packet was of equal cost, a packet with one byte of data is the same as a packet with 28 bytes of data. The assumption was not meant to be accurate, but rather a close approximation. Another comparison was the total number of bits sent. Once again, this was not meant to be accurate, since two packets sending one byte each is not equivalent to one packet sending two bytes each.

Instead of using simulators, we implemented and compared the different security protocols on a Crossbow mica2 MPR2600 mote [7]. The protocols were developed using the implementation found in [1]. We had one sensor attached to a workstation while the other sensors were placed stand-alone. After the running of the protocols, the elapsed time was then sent via the serial connection, to a PC running a Linux[®] distribution where we have a Java[®] application reading the TinyOS packet from the serial port, and report that data to the user.

Sensors normally use small keys, encrypting large amounts of data is time and energy consuming. The small keys, such as K_{AB} in the protocols, should be refreshed frequently. The refresh should happen before the average time it will take an adversary to calculate the key using brute force. However, larger keys can exist in the sensor network. The large keys, such as the key K_S can be used to generate the small keys. If the key K_S was small, such as 64 bits, then an adversary is able to use brute force to calculate K_S in a short time, which defeats the purpose of having a long term key.

In our experiments we originally implemented the protocols using 64 bit keys. However, our home health care system requires keys that may exist for weeks or months in the system. So we ran the same experiments using a 128 bit keys.

Figure 1 shows the total time taken to for a single protocol run. Each of the protocols do not use the benefits that a multi-network infrastructure can provide. Every message is either sent to a sensor or is received by a sensor. For completeness, we also included the times for the protocols when they were using 64 bit keys. We did not include the Gong 128 bit key protocol run, since it took over four seconds to complete eight server scenario.

We do see that there is a notable difference between using an 128 bit key over a 64 bit key. Hence, this protocol should be run rarely in a sensor environment. Also, the Singh et al. protocols generated two keys, whereas both the Gong and Chen et al. protocols only generated one key. In our implementation, we did not include the cost of generating

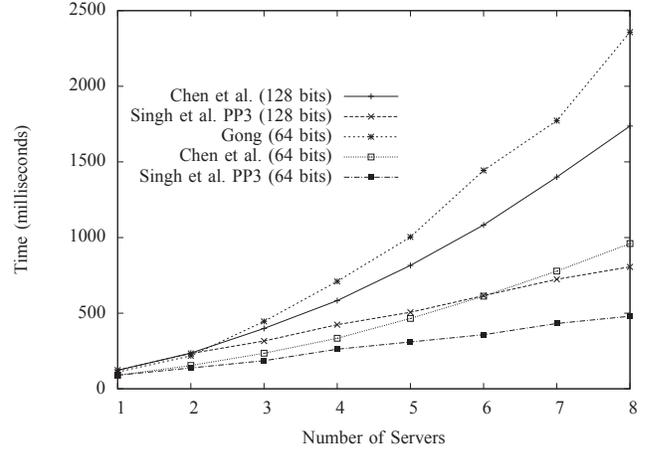


Figure 1. Total Time Taken by different Protocols

the session key. The initial figures from implementing the Gong and Chen et al. protocols showed that the protocols were heavier-weight than the Singh et al. protocols.

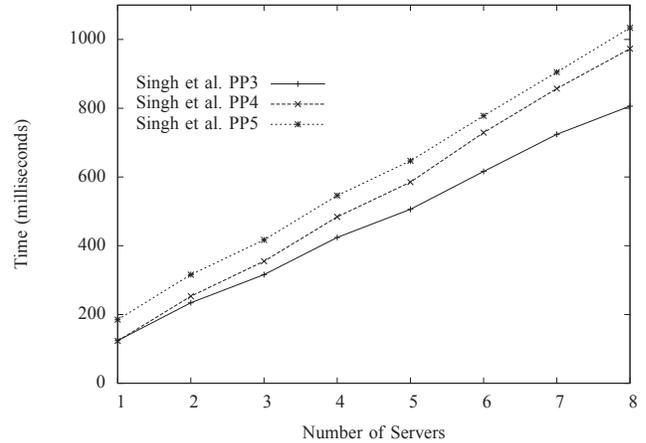


Figure 2. Total time taken on single network

Figure 2 compares the times of the Singh et al. protocols if there is only a single network and no multi-tiered network infrastructure. The Singh et al. PP4 and Singh et al. PP5 protocols has higher performance costs than the Singh et al. PP3 protocol. The Singh et al. PP5 protocol, which is the most secure, has the highest performance cost.

Figure 3 compares the times of the Singh et al. protocols on multi-tiered network infrastructure. In our setup we removed the message time results for any message that was using a faster network. We compared performance of the different protocols only on the slower sensor network. The Singh et al. PP4 and Singh et al. PP5 protocols has lower performance costs than the Singh et al. PP3 protocol. As the number of servers increases so does the benefit of

using protocols that utilize faster networks. There still is a considerable benefit of using Singh et al. PP4 and Singh et al. PP5 when the number of servers is small, such as two or three servers.

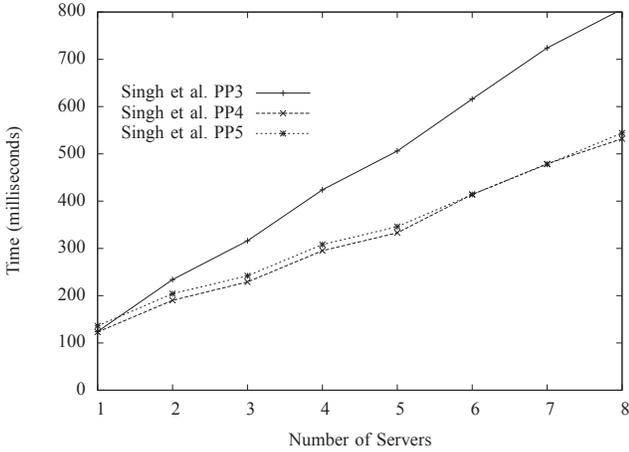


Figure 3. Total time taken using different networks

The Singh et al. PP5 protocol, which is the most secure, has similar performance costs as Singh et al. PP4. For the extra security and little performance costs, the Singh et al. PP5 protocol seems to be the best choice.

A. Performance costs of cryptography

We found that the performance costs of the cryptography algorithms was insignificant compared to the costs of communication. Where the communication can be measured in the milliseconds the computation costs had to be measured in the microseconds range.

Before comparing the different cryptographic primitives, and the benefits that one implementation has over another, we created skeleton code based on TinyOS 2.x. The skeleton code initializes the sensor node, and after the sensor is initialized we obtained the initial time in milliseconds. We then run a cryptographic primitive in a loop for 2000 iterations, before obtaining a new time. We subtracted the new time from the initial time to obtain the elapsed time in milliseconds to run our cryptographic primitive for 2000 attempts.

The key establishment protocols uses exclusive-or (xor) to encrypt the new session key. We compare this method with other methods of encrypting the new session key for body sensor networks. We have implemented RC5, SKIPJACK, MD5-HMAC and ECC cryptographic primitives on the mica2 MPR2600 motes using TinyOS 2.x.

Table I supplies the ratio of time taken for each of the algorithms compared to exclusive-or algorithm. When we ran the algorithm on the mica2 mote, over the 2000 iterations it took approximately one millisecond. In the ATEMU simulator it took approximately 7000 instructions.

Table I
RATIO COMPARED TO EXCLUSIVE-OR

Algorithm	Mica2	ATEMU
RC5	453	456
SKIPJACK	739	741
HMAC-MD5	18400	18500
ECC	4820000	4920000

We found little difference between the simulation results and the amount of time an operation takes when put on the mica2 mote. The most notable difference in results was for the ECC algorithm where there was a two percent greater difference, and was not as accurate.

B. Size

The size of the application both with number of lines of code and the size in bytes is important when choosing an algorithm. Table II list the extra number of lines of code and the extra size in bytes that are added to an application that previously had no security.

Table II
TIME MEASUREMENTS FOR DIFFERENT ALGORITHMS

Algorithm	Lines of Code	Size (bytes)
RC5	426	828
SKIPJACK	617	1798
HMAC-MD5	427	12714
ECC	5038	9988

The *Lines of Code* indicates the complexity for the coder to implement the algorithm. The *Size (bytes)* indicates the size in bytes of the application.

The RC5 application took considerable more effort than the exclusive-or (xor) application. We found an RC5 implementation for TinyOS 1.x in the TinySEC library [1], however, it has yet to be ported to TinyOS 2.x. Most of our effort was spent porting the code to the new platform.

The SKIPJACK application had similar problems as the RC5 application. Where there was an implementation for TinyOS 1.x in the TinySEC library but there was not one for TinyOS 2.x. Once again, most of our effort was spent porting the code to the platform.

For HMAC-MD5 application we could not find any previous implementations of HMAC-MD5 in any version of TinyOS. In this case we obtained code from RFC1321 and RFC2104 and ported the code to first the nesc language and then to the TinyOS application. This was considerably more effort then either RC5 or SKIPJACK implementations.

The ECC application also had similar problems to the RC5 and SKIPJACK implementations. We ported an ECC library [1] developed for TinyOS 1.x to TinyOS 2.x. The ECC application used a 160 bit points.

The HMAC-MD5 application is the largest, however the application was a straight port from the RFCs, where the

code was not intended for sensors. The ECC application is much larger than a RC5 or SKIPJACK application.

We also examine the memory requirements of the application, as shown in Table III. The combination of `.bss` and `.data` segments use SRAM, and the combination of `.text` and `.data` segments use ROM. 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.

Table III
MEMORY OVERHEAD IN BYTES ON MICA2 PLATFORM

Memory	RC5	ECC
ROM	6942	15792
RAM	367	1283
<code>.data</code>	14	10
<code>.bss</code>	353	1273
<code>.text</code>	6928	15782

V. CONCLUSIONS AND FUTURE WORK

We have explained why the quality of security services in our home health care system is an important requirement. If the security services of the system is down then the system itself may become compromised. We showed that with multi-server protocols, even if one or more servers become unavailable or untrustworthy, it may still be possible for the sensor nodes to establish a good session key. We examined existing multi-server key establishment protocols in our home health care system. Implementation of the protocols involved either porting libraries or creating new libraries in TinyOS 2.x. The time elapsed, complexity of the code, and memory requirements are analysed in detail on mica2 sensors. We found that a symmetric key implementation has advantages over an asymmetric implementation. Future work will include recording the energy usage of each of the protocols described in this paper.

ACKNOWLEDGMENTS

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. Other company, product, or service names may be trademarks or service marks of others.

REFERENCES

- [1] K. Singh and V. Muthukkumarasamy, "Verification of key establishment protocols for a home health care system," in *Proceedings of the Fourth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Sydney, Australia, December 2008.
- [2] —, "Authenticated key establishment protocols for a home health care system," in *Proceedings of the Third International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Melbourne, Australia, December 2007.
- [3] A. Hampapur, L. Brown, J. Connell, N. Haas, M. Lu, H. Merkl, S. Pankanti, A. Senior, C.-F. Shu, and Y. Tian, "S3-r1: the ibm smart surveillance system-release 1," in *ETP '04: Proceedings of the 2004 ACM SIGMM workshop on Effective telepresence*. New York, NY, USA: ACM Press, 2004, pp. 59–62.
- [4] USA, "Summary of hipaa health insurance probability and accountability act," US Department of Health and Human Service, May 2003.
- [5] K. Singh and V. Muthukkumarasamy, "Performance analysis of proposed key establishment protocols in multi-tiered sensor networks," *Journal of Networks*, vol. 3, no. 6, 2008.
- [6] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu, "Senseye: a multi-tier camera sensor network," in *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*. New York, NY, USA: ACM Press, 2005, pp. 229–238.
- [7] Crossbow, "Crossbow," <http://www.xbow.com/>, 2006.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] C. Boyd and A. Mathuria, *Protocols for Authentication and Key Establishment*, U. Maurer and R. Rivest, Eds. Springer Berlin / Heidelberg, 2003.
- [13] 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.
- [14] 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.
- [15] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.