Data Authentication for Sensor Networks

Philip Basford
School of Electronics and Computer Science, University of Southampton, SO17 1BJ
pjb304@zepler.net

Abstract. Recent technological developments have enabled an increasing number of sensor networks to be deployed. Despite this rapid increase in the number of deployed sensor networks there has been relatively little research into how to prevent third parties injecting false data into the system. This paper summarises the existing solutions available however none can provide protection all the way from the node gathering the data to the party analysing it. All the solutions focus on the protection of the inter node communication. This paper suggests a solution which solves the problem by providing an end to end solution using TinySec and Elliptic Curve Cryptography.

1. Introduction

Over the last few years advances in both wireless technologies and hardware miniaturisation [1] have allowed smaller and cheaper sensors, processors and communication interfaces to be developed [2]. These components have enabled the deployment of an increasing number of sensor networks. Sensor networks consist of a group of low-cost low-power devices spread over a geographical area [1], these sensors are then able to monitor the environment in which they are placed. This data is then collated by the Base Station (BS) a node which has external connectivity. The BS then forwards the collated data to the organisation that deployed the sensor network for further analysis. Sensor networks have a wide variety of uses including monitoring environmental issues [3-7], or military surveillance [7, 8], other deployments of sensor networks are described in [9]. Communication in sensor networks can use either wired or wireless protocols [10]. Wireless communication is usually more flexible and generally easier to deploy, however with this added convenience come additional security risks [11, 12].

Sensors networks can be deployed into a wide variety of situations, with each situation having different security requirements. The lowest level of security does not apply any security measures to the data transmitted, meaning that anyone can eavesdrop on the data exchange. When the attacker has observed enough data they will be in a position in which they can inject false data into the network as shown in Figure 1.

The next level of security up from this involves making sure that the data comes from a valid source - an official node in the sensor network - and has not been changed whilst in transit to the destination. An example where this might be needed is when the data will be used for scientific analysis, as the researchers need to be able to ensure that their data is reliable. In this situation it does not matter if a third party gains access to the data, so a signed hash of the data would suffice. If the

privacy of the data is important then a higher level of security will be needed. In this highest level all data transmitted within the network will need to be encrypted. As the level of security increases the computational processing requirements also increase.

When developing software for sensor networks additional challenges are faced because the sensors have a strictly limited power supply [13, 14]. This scarce resource has to be used carefully in order to maximise the useful life of the sensor, as typically it is difficult to recharge the sensors batteries [1]. Transmitting data over radio links is very power intensive [15], with one study [16] determining that transmitting a single bit of data consumes approximately the same amount of power as executing 800 – 1000 processor instructions. The exact ratios will obviously vary depending on the hardware used but these figures demonstrate that data transmission is expensive, therefore the amount of data transmitted should be minimised. Other methods of minimising the power requirements include limiting the speed of the processor and the size of the memory. These restrictions mean that performing complex calculations will be time consuming or not even possible. Despite these limitations in the amount of bandwidth [17], processing power and memory available "power is the scarcest resource" [16, 18]. This means that the "level of security versus the consumption of energy […] constitute a major design trade-off" [19].

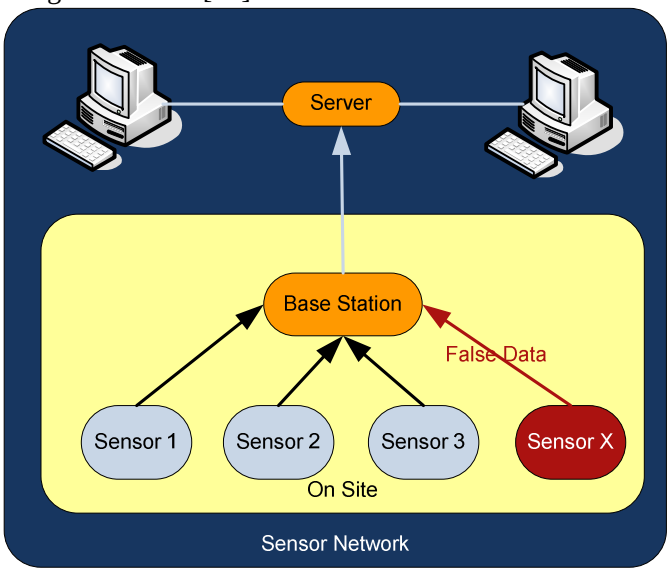

Figure 1 An example of data injection

The BS will typically be much more powerful than the individual sensor nodes [16, 20], which means that more computationally intensive processing can be performed on the BS if needed. As all communication between the sensor network and the data analysts passes through the BS, if the BS is compromised the entire sensor network becomes at best untrustworthy and at worst useless [12]. This means that the BS is a primary target [21, 22] for attack, because of this measures should be taken to make it tamper proof. It is therefore assumed that the BS will not be compromised [23, 24]. However due to cost constraints making the individual sensor nodes tamper-proof is not feasible [16, 20, 25].

Given the above discussion of power consumption and the security of individual nodes it is important to consider where the authentication of the data will be carried out. If all the sensor nodes in the network are just a single hop away from the BS then it makes sense for the authentication to be

performed at the BS. If the sensor nodes are more than one hop away from the BS then data will have to pass though other nodes, expending their resources. This leads to the conclusion that it would make sense for the authentication to be carried out en-route to the BS so the unauthorised data can be dropped, because whilst this adds computational complexity to the intermediate nodes this is cheaper than retransmitting unauthorised data.

If the authentication is carried out en-route to the BS then the infrastructure for carrying out such authentication will need to be in place Setting up this infrastructure will require the transmission and storage of data, it therefore need to be ascertained whether the cost of setting up the authentication framework is greater than the cost of transmitting unauthenticated packets, and whether this cost is acceptable. Another consideration is that in order for the nodes on the path to the BS to be able to authenticate the data the individual nodes will need to have keys in common with the node that transmitted the data. This can lead to increased security risks if the sharing of keys is managed incorrectly. As it is prudent to assume that at least one node will be compromised [26] therefore steps should be taken to make sure that the compromise of a single node will not affect the security of the entire sensor network.

For the purposes of this paper it will be assumed that the individual nodes are behaving as they should, rather than behaving selfishly and refusing to pass on data because they the energy cost is too high. For a further discussion of node behaviour see [27].

2. Existing Solutions

There have already been several studies in to securing sensor networks. Typically the studies focus on the confidentiality of the data rather than the integrity and authentication of the data. Although encrypted data is likely to have come from a valid source there is no guarantee that it has not been altered whilst in transit [28, 29].

Some of the existing schemes make use of Timed Efficient Stream Loss-tolerant Authentication (TELSA) [30], which enables the properties provided by asymmetric algorithms to be achieved using symmetric algorithms. The TELSA scheme has low computational overheads, low per packet communication overheads, and can tolerate some packet loss. It is however limited to unidirectional communications, so is suited to broadcast messages only. In order for TELSA to work properly all nodes must maintain loose time synchronisation, because if the clocks have too wide a variation then the receiver will not be able to verify the data. The asymmetric properties are achieved by transmitting the data in timeslots with each slot using a different encryption key. When the packets are sent only the sender knows the key, which is then transmitted a couple of timeslots after the messages using that key. When the messages are received they are buffered and stored until the key is received. As the key is transmitted after the encrypted message is received the receiver can verify that the message was not altered in transit and it was from the correct sender.

The following discussion focuses on the existing solutions to security in sensor networks beginning with the oldest. It is worth noting that sensor networks are a form of ad-hoc networking, and whilst the complexity of some of the security schemes for ad-hoc networks (examples of which include [31-33]) prevents their direct application to sensor networks, some of the ideas discussed may be applicable.

The simplest algorithm to use for securing communication between nodes in a sensor network would be for every node in the network to share a pre-determined secret key, which is then used to

encrypt the message using any suitable algorithm. As all nodes share the same key the message could be authenticated by every node on the route to the BS. Whilst this solution is simple it is not particularly effective because if a single node is compromised then the entire network is compromised. As well as the security concerns raised by having a globally shared key there are also problems of scalability [34].

One possible solution to the problem of all nodes in the network sharing the same key is to provide a mechanism by which the key may be periodically changed. One method by which this can be achieved is described in [35]. The approach taken is to split the network into clusters in order to reduce the complexity of key management. Once the network has been split, a node in each cluster is elected as the cluster head (CH). These CH then form a back-bone network for the system. From this back-bone of CH a key manager is elected. Once the process of electing a key manager is complete the manager can generate a key and distribute it to all the other nodes in the network via the CH, this key is then used for all communication. Whilst this system has exactly the same weaknesses as the globally shared key system the fact the key can be changed regularly and reduces the risk of it being compromised by packet sniffing, as the volume of traffic transmitted with each key will be lower. This method also allows different levels of security to be supported, as if a particular network has higher security requirements then the key could be updated more regularly than for a network with less rigorous security requirements.

The Security Protocols for Sensor Networks (SPINS) [12] protocol consists of two sub parts: Sensor Network Encryption Protocol (SNEP) and µTELSA (a minimal implementation of the TELSA protocol). The system makes the assumption that the BS is trusted and only communication between the BS and the nodes (in either direction) needs to be encrypted. In order to make sure that keys are not reused for the encryption of data the nodes share a master key and use it for the generation of encryption keys. SNEP relies on each communicating node having a shared counter, which enables weak freshness to be ensured and protects against replay attacks. It is however possible for these counters to become inconsistent, so there is a protocol to resynchronise the counters, and if necessary the counter can also be sent with each message. The encryption algorithm chosen for use in the SPINS project is RC5 which was chosen because of its “small code size and high efficiency” [12]. The RC5 algorithm is used in counter mode, meaning the cipher text is exactly the same length as the plain text which means there is no transmission overhead in sending the encrypted data when compared to sending the plain text. Using the counter mode also has the advantage that if the same message is sent multiple times the encrypted texts will be different.

The Localized Encryption and Authentication Protocol (LEAP) [23] is another encryption scheme which can offer different levels of security for different types of message, which can be achieved because each node has to store 4 different types of key. Every node has an individual key which is shared only with the BS. Each node will also have a copy of the group key, which is shared by all the nodes on the system. This key is usually used for the BS to send out broadcasts to the nodes. The third key that a node has is the cluster key: a key shared by the node and its neighbours which means that local broadcast messages can be secured. The final type of key that a node will have are the pair-wise shared keys. Each node will have a pair-wise shared key for each of its neighbours. By using these different keys the messages sent can be protected to different extents. LEAP also implements µTELSA to enable broadcast messages from the BS to be secured. In order to authenticate a message the sending node signs it with its cluster key and transmits it. When this message is received by another node it is verified using the relevant cluster key, the data is then authenticated with its own cluster key and then forwarded on to the next node.

TinySec [36] is an encryption scheme which has 2 modes of operation. It can either be used in authenticated encryption (TinySec-AE) mode, in which the payload is encrypted and then a Message Authentication Code (MAC) calculated over the encrypted data and the packet header. The second mode provides only authentication (TinySec-Auth), which is achieved by calculating a MAC over the plain text of the message. This means that the system is more flexible than some of the other alternatives because when confidentiality is not required the encryption of the packet can be disabled. TinySec uses an 8 byte Initialisation Vector (IV), which is calculated using the source and destination address, the message type and length and a counter (which starts at 0). Given the method by which the IV is generated and the fact it is so short there are likely to be messages which generate the same IV, TinySec cannot use stream ciphers [36]. TinySec uses CBC-MAC to authenticate that the message has not been altered, but rather than using a 8 or 16 byte MAC TinySec uses just 4 bytes. Given the context into which TinySec will be deployed even this length MAC is enough, because the only way of knowing if a MAC is valid is to try it. Given the low speed of communications links in a sensor network the time taken to try 2^{31} combinations (half the key space) is impractical. TinySec, whilst providing a means for the encryption and authentication of data, does not solve the key distribution problem. The easiest solution is to have a global key, although a better solution is provided by TinyPK [34].

All the previous systems use symmetric key encryption, however TinyPK [34] is based on RSA and makes use of asymmetric keys. As is common with other implementations of asymmetric algorithms the encryption of the messages is performed using a symmetric algorithm, in this case TinySec [36]. TinyPK uses a CA whose public key has to be preloaded onto the nodes in the networks, meaning that the nodes will require some pre-configuration before they can be deployed in the field this can pose saleability problems. However TinyPK does not use certificates as there is no real-time access to the CA, this poses problems when it comes to revoking keys.

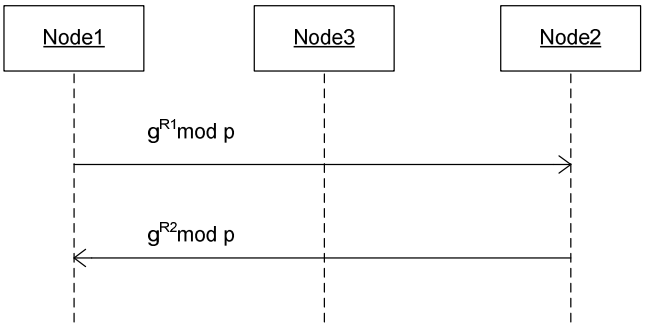

Figure 2 Data Exchange for Diffie-Hellman as used in TinyPK

When TinyPK was implemented it was discovered that the RSA calculations were far too slow [34] so instead TinyPK has been implemented using Diffie-Hellman, as it is sufficiently fast. The messages sent between the nodes to establish the keys are shown in Figure 2, once this is complete the key can be calculated according Formula 1. With nodes 1 and 2 having enough information to perform the calculation, but node 3 despite having seen all the communication does not.

$$key = \left(g^{R1} \bmod P\right)^{R2} \bmod P = \left(g^{R2} \bmod P\right)^{R1} = g^{R1*R2} \bmod P. \quad \textbf{(1)}$$

Another scheme for ensuring that compromised nodes cannot insert incorrect data is to compare the data from one node to the data from the surrounding nodes. If the data gathered is within a certain threshold, or a certain number of nodes agree on the value then it can be assumed that the data from all the nodes is correct. If it is not within the threshold then it is discarded [37]. This approach is suitable for data such as temperature or water level in a river where the difference between data gathered by adjacent nodes is likely to be small. However when the data being gathered is more localised, such as the orientation of a sensor in a glacier [3], this validation scheme is likely to reject correct data because the data set from one sensor can legitimately be different to those from neighbouring sensors.

The Peer Intermediaries for Key Establishment (PIKE) [21] protocol relies on each node having a set of keys of $O\left(\sqrt{n}\right)$ where n is the number of nodes in the network. Each key is shared with only one other node. This means that if A needs to communicate with B then there will exist a node C which shares a secret key pair with A and a secret key pair with B. This means that the message can be sent from A to C, at which point it will be decrypted and encrypted with the key for B and sent on. PIKE relies on each node being addressable so that the intermediaries know which key-pair to use to decrypt the message they have received. There will typically be multiple nodes which share a key with both A and B and are therefore candidates for being the intermediary, and the potential intermediaries will be compared on the cost of sending a message using them, and the node with the lowest cost used.

3. Encryption Algorithms

The methods described above make use of a wide variety of different encryption algorithms. When comparing the algorithms in use the main factors to consider are the computational costs, memory requirements and the way to distribute the key. Another factor which can affect the choice of an algorithm is how well it is known, since the better known the algorithm is the more scrutiny it will have been subjected to therefore the higher confidence that the algorithm is not flawed.

The majority of these solutions use some form of symmetric encryption because asymmetric cryptography is several orders of magnitude more computationally expensive than symmetric key encryption [38], which makes it slower [39] and unsuitable for most sensor network applications [20, 40-43]. Most implementations of asymmetric algorithms switch to using symmetric algorithms once a key has been established [29]. Each method has advantages and disadvantages. The computation required for asymmetric algorithms is high, but the key redistribution algorithms for symmetric keys are vulnerable to attack [44]. Whereas asymmetric algorithms are a relatively recent invention, symmetric algorithms were originally used centuries ago [38] and there are therefore many more symmetric algorithms available then there are asymmetric algorithms. Some of the algorithms which could be used will now be examined in more depth.

The most popular asymmetric algorithm is Rivest-Shamir Adleman (RSA) [45] which is based on the fact that factoring large numbers is computationally difficult. In order for the algorithm to be secure the numbers used in the algorithm must be fairly large – in the order of 100s of digits. This means that many sensor nodes will not have the memory required to store the numbers required for the algorithm to work in its original form.

Another asymmetric algorithm which could be considered is Elliptic Curve Cryptography (ECC) [46], which is based on elliptic integration, in which the data to be encrypted is a point on the curve which is then mapped to a different point in a manner defined by the algorithm. The main advantage of ECC over RSA is that the same level of security can be achieved using significantly shorter keys [38, 47] this means that the memory requirements of the algorithm are significantly less, which in turn means the algorithm is more suited to being implemented on devices with limited power. The other encryption algorithms that could be used are all symmetric.

The Rijndael encryption algorithm [48] (also known as AES) has been designed for both 32 bit architectures and smart cards, which means that implementations for 8 bit architectures are available [38] which could be important for implementations on sensor networks. It also supports the use of different key lengths meaning that different levels of security and processing requirements can be supported. Rijndael works by repeatedly manipulating the data, the number of rounds of manipulation is determined by the key length, from 10 rounds with a 128 bit key, up to 14 rounds with a 256bit key.

The Rijndael algorithm is the current NIST standard and was preceded by the Data Encryption Standard (DES) [49]. DES is a block cipher which uses a Feistel System [50], containing various functions which expand and contract that data (in a lossless manner) this data is then combined with the key and further manipulated This mixing and manipulation of the data continues for 16 rounds.

The Tiny Encryption Algorithm (TEA) [51] is a symmetric algorithm which has designed to have a very small footprint. Rather than using a very complicated algorithm the designers of TEA decided to use a large number of iterations instead. It uses a 128 bit key to protect against brute force methods. It can be simply implemented in both hardware and software, with the hardware implementation being the same order of complexity as DES [51], and one software implementation being 3 times faster than a good software implementation of DES [51]. However TEA has not been submitted to the NIST or been used as widely as DES or AES which means that it has not been submitted to the same level of scrutiny.

4. Attacks against Sensor networks

There are two classes of attacks that can be carried out against wireless networks, passive attacks which just involve the attacker listening and analysing the data gathered, and active attacks in which the attacker listens and broadcasts on the wireless link. Physically altering the nodes can also be considered an active attack. Whilst passive attacks just involve listening to and analysing the data, active attacks can be more diverse in the mechanisms used. Although typically active attacks will be preceded by passive attacks in which the attacker will build up information about the network to be attacked.

Possibly the simplest active attack is to jam the wireless link so that no traffic can be sent over the link, however this attack has to be dealt with at the physical layer using methods such as DSSS [52] and FHSS [53]. Another simple attack against wireless sensor networks is to spoof a node and insert false data into the network. If the data is drastically different from what is expected and what is being reported by other nodes in the network then the false data could be detected using simple statistical analysis. However if the data being injected is similar to the genuine data or multiple nodes have been compromised then detection will be much trickier.

It is important to remember that whilst in the majority of situations the above attacks will be deliberately executed, some of the attacks such as radio jamming or physically damaging the nodes can be performed accidently. This is a very brief description of the attacks and countermeasures that can be employed against and by sensor networks, as the area is large enough to support an entire paper.

5. A Data Authentication Scheme for Sensor Networks

The various schemes discussed in Section 2 provide ways to encrypt the data sent within the sensor network, however none of the solutions extend the system to allow the validity of the BS to be checked by the users analysing the data collected. This means that in theory an attacker could replace the BS with their own version and have complete control over the data sent to the users without the user having any idea of the compromise. If the communication between the analysts and BS is via an IP network the same attack could be achieved by manipulating routing tables.

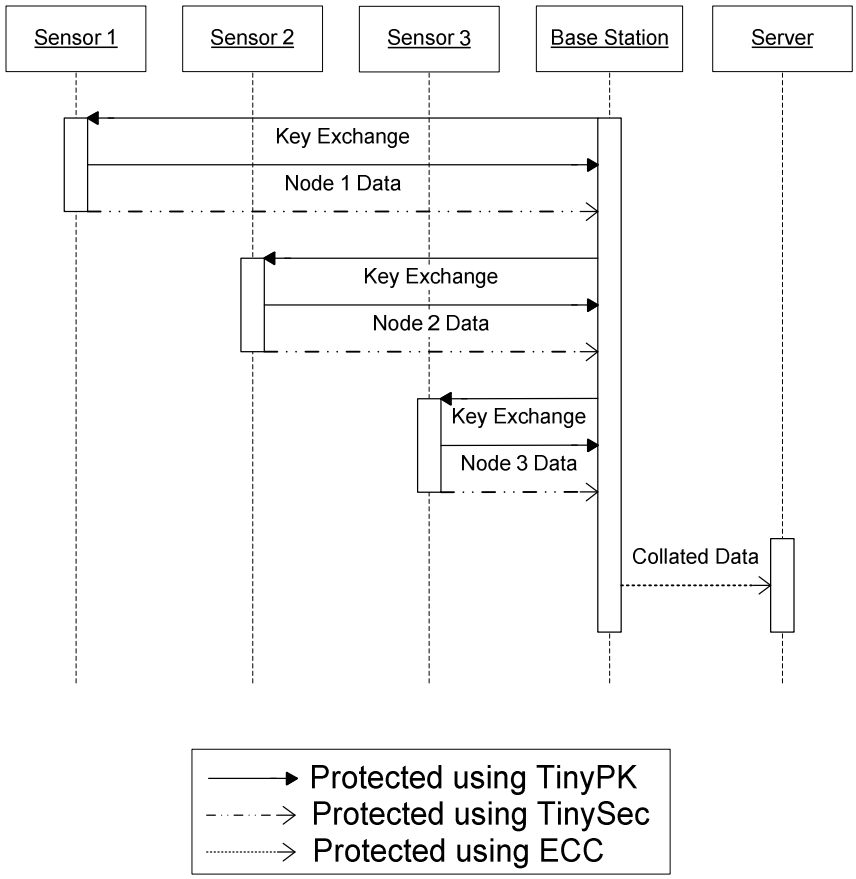

Figure 3 Diagram showing flow of data from sensor nodes to storage server

To overcome this there needs to be an authentication method between the BS and the user remotely accessing the data. This system does not have to adhere to the tight power restrictions placed on the individual sensor nodes. This authentication between BS and the client could be ensured by using ECC to sign the messages from the BS. The advantages of using a public key algorithm have been discussed in Section 3. The choice of ECC over RSA has been made because ECC is less computationally expensive and requires less data to be transmitted and stored [54], which is important because although the BS will typically be less constrained than the individual nodes it is likely to have some restrictions on the energy and computational power available. For details on how to use ECC for digital signatures (ECC-DSA) see [47].

Having chosen to use ECC between the BS and the server on which the data is collated, a decision needs to be made about which method to use to encrypt the data between the sensor nodes and the BS. The most suitable of the solutions previously discussed is TinySec as it allows for both encryption and authentication, however TinySec does not address the problem of key distribution. To overcome this TinySec can be combined with TinyPK, and it is in this configuration that it is recommended.

These choices of algorithms enable the sequence of events shown in Figure 3 to be performed whenever the BS wants to request data from the nodes (only 3 nodes are shown for simplicity, it could be a greater number). First the BS will initiate the key generation with the node using Diffie-Hellman as shown in Figure 2. When the key exchange is complete the actual data can be sent using TinySec to provide authentication. Once the data is on the BS it can be stored until a scheduled upload, or uploaded immediately. Either way before the data is transmitted it is signed using ECC. The personnel analysing the data can then use it knowing it is from a trusted source. This can be verified by comparing the signature with the public key for the BS which sent the data.

6. Conclusions

This paper has examined the area of data authentication for sensor networks by comparing the existing solutions and algorithms which could be used. These comparisons lead to the recommendation that in order to deploy a sensor network in which the origins of the data received can be verified a combination of TinySec, TinyPK and ECC should be used. This combination is recommended because TinySec offers two modes of operation: TinySec-Auth and TinySec-AE, and is therefore more flexible than the other solutions examined. TinySec however does not address the problem of distributing keys to the individual nodes. This problem is solved by using TinyPK to enable the communicating nodes to create a secret shared key. The combination of TinySec and TinyPK allows the data to be authenticated when it is sent to the BS. In order to sign the data between the BS and the server on which the data is stored it is recommended that ECC is used as it allows the data to be signed and can achieve the same levels of security as RSA with shorter key lengths meaning it is less processor intensive.

As the need for security in sensor networks varies depending on the application into which it is deployed the solution proposed is flexible in that it can be disabled, provide authentication services for the data or with only minor modifications (pre-loading the storage servers public key onto the BS) can provide end-to-end encryption of the data as well. The system has been explained with direct communication between the nodes and the BS however there is no reason why the scheme could not be extended to support multiple hop routing, although the power consumption of this system would need to be carefully considered.

The area of securing data in sensor networks is a vast research area which so far has had relatively little attention. Further research into this area should continue to devise different and more efficient implementation of encryption algorithms, as well as focusing on other possible solutions to the key distribution problem. Further work should also include the implementation and evaluation of the authentication scheme proposed in Section 5.

References

1. Tubaishat, M., Madria, S.: Sensor networks: an overview. Potentials, IEEE **22** 20-23 (2003)
2. Chong, C.-Y., Kumar, S.P.: Sensor networks: evolution, opportunities, and challenges. In: Kumar, S.P. (ed.): Proceedings of the IEEE, Vol. 91 1247-1256 (2003)
3. Martinez, K., Ong, R., Hart, J.: Glacsweb: a sensor network for hostile environments. The First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, Santa Clara, USA(2004)
4. Padhy, P., Martinez, K., Riddoch, A., Ong, H.L.R., Hart, J.K.: Glacial Environment Monitoring using Sensor Networks. Real-World Wireless Sensor Networks, Stockholm, Sweden(2005)
5. Pottie, G.J.: Wireless sensor networks. Information Theory Workshop, 1998 139-140 (1998)
6. Djenouri, D., Khelladi, L., Badache, A.N.: A survey of security issues in mobile ad hoc and sensor networks. Communications Surveys & Tutorials, IEEE **7** 2-28 (2005)
7. Hao, Y., Fan, Y., Yuan, Y., Songwu, L., William, A.: Toward resilient security in wireless sensor networks. Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing. ACM, Urbana-Champaign, IL, USA(2005)
8. Jing, D., Richard, H., Shivakant, M.: Defending against path-based DoS attacks in wireless sensor networks. Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks. ACM, Alexandria, VA, USA(2005)
9. Xu, N.: A survey of sensor network applications. University of Southern California(2002)
10. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System architecture directions for networked sensors. SIGPLAN Not. **35** 93-104 (2000)
11. Borisov, N., Goldberg, I., Wagner, D.: Intercepting mobile communications: the insecurity of 802.11. Proceedings of the 7th annual international conference on Mobile computing and networking. ACM, Rome, Italy(2001)
12. Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.E.: SPINS: security protocols for sensor networks. Wireless Networks **8** 521-534 (2002)
13. Akyildiz, I.F., Weilian, S., Sankarasubramaniam, Y., Cayirci, E.A.C.E.: A survey on sensor networks. Communications Magazine, IEEE **40** 102-114 (2002)
14. Przydatek, B., Song, D., Perrig, A.: SIA: secure information aggregation in sensor networks. Proceedings of the 1st international conference on Embedded networked sensor systems. ACM, Los Angeles, California, USA(2003)
15. Slijepcevic, S., Potkonjak, M., Tsiatsis, V., Zimbeck, S.A.Z.S., Srivastava, M.B.A.S.M.B.: On communication security in wireless ad-hoc sensor networks. In: Potkonjak, M. (ed.): Enabling Technologies: Infrastructure for Collaborative Enterprises, 2002. WET ICE 2002. Proceedings. Eleventh IEEE International Workshops on 139-144 (2002)
16. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: attacks and countermeasures. In: Wagner, D. (ed.): Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on 113-127 (2003)
17. Chen, M., Cui, W., Wen, V., Woo, A.: Security and deployment issues in a sensor network. Berkeley(2000)

18. Hu, F., Sharma, N.K.: Security considerations in ad hoc sensor networks. Ad Hoc Networks **3** 69-89 (2005)
19. Jolly, G., Kuscu, M.C., Kokate, P., Younis, M.: A low-energy key management protocol for wireless sensor networks. In: Kuscu, M.C. (ed.): Computers and Communication, 2003. (ISCC 2003). Proceedings. Eighth IEEE International Symposium on 335-340 vol.331 (2003)
20. Shi, E., Perrig, A.: Designing secure sensor networks. Wireless Communications, IEEE [see also IEEE Personal Communications] **11** 38-43 (2004)
21. Haowen, C., Perrig, A.: PIKE: peer intermediaries for key establishment in sensor networks. In: Perrig, A. (ed.): INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, Vol. 1 524-535 vol. 521 (2005)
22. Haowen, C., Perrig, A., Song, D.: Random key predistribution schemes for sensor networks. In: Perrig, A. (ed.): Security and Privacy, 2003. Proceedings. 2003 Symposium on 197-213 (2003)
23. Zhu, S., Setia, S., Jajodia, S.: LEAP: efficient security mechanisms for large-scale distributed sensor networks. Proceedings of the 10th ACM conference on Computer and communications security. ACM, Washington D.C., USA(2003)
24. Ye, F., Luo, H., Lu, S., Zhang, L.: Statistical en-route filtering of injected false data in sensor networks. Selected Areas in Communications, IEEE Journal on **23** 839-850 (2005)
25. Haowen, C., Perrig, A.: Security and privacy in sensor networks. Computer **36** 103-105 (2003)
26. Carman, D., Kruss, P., Matt, B.: Constraints and approaches for distributed sensor network security. NAI Labs(2000)
27. Shneidman, J., Parkers, D.C.: Rationality and self-interest in peer to peer networks. Lecture notes in computer science **2735** 139-148 (2003)
28. Bellovin, S.M.: Problem areas for the IP security protocols. Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography - Volume 6. USENIX Association, San Jose, California(1996)
29. Krawczyk, H.: The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology. Springer-Verlag(2001)
30. Perrig, A., Tygar, J.D., Song, D., Canetti, R.: Efficient Authentication and Signing of Multicast Streams over Lossy Channels. IEEE Symposium on Security and Privacy (2000)
31. Luo, H., Kong, J., Zerfos, P., Lu, S., Zhang, L.: URSA: ubiquitous and robust access control for mobile ad hoc networks. IEEE/ACM Trans. Netw. **12** 1049-1063 (2004)
32. Sencun, Z., Shouhuai, X., Setia, S., Jajodia, S.A.J.S.: LHAP: a lightweight hop-by-hop authentication protocol for ad-hoc networks. In: Shouhuai, X. (ed.): Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on 749-755 (2003)
33. Zhou, L., Haas, Z.J.: Securing ad hoc networks. Network, IEEE **13** 24-30 (1999)
34. Watro, R., Kong, D., Cuti, S.-f., Gardiner, C., Lynn, C., Kruus, P.: TinyPK: securing sensor networks with public key technology. Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks. ACM, Washington DC, USA(2004)
35. Basagni, S., Herrin, K., Bruschi, D., Rosti, E.: Secure pebblenets. Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking \& computing. ACM, Long Beach, CA, USA(2001)
36. Karlof, C., Sastry, N., Wagner, D.: TinySec: a link layer security architecture for wireless sensor networks. Proceedings of the 2nd international conference on Embedded networked sensor systems. ACM, Baltimore, MD, USA(2004)
37. Zhu, S., Setia, S., Jajodia, S., Ning, P.: An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In: Setia, S. (ed.): Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on 259-271 (2004)

38. Trappe, W., Washington, L.C.: Introduction to Cryptography with Coding Theory (2nd Edition). Prentice-Hall (2005)
39. Schneier, B.: Applied cryptography : protocols, algorithms and source code in C. Wiley, New York (1996)
40. Cam, H., Ozdemir, S., Muthuavinashiappan, D., Nair, P.A.: Energy efficient security protocol for wireless sensor networks. In: Ozdemir, S. (ed.): Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th, Vol. 5 2981-2984 Vol.2985 (2003)
41. Ganesan, P., Venugopalan, R., Peddabachagari, P., Dean, A., Mueller, F., Sichitiu, M.: Analyzing and modeling encryption overhead for sensor network nodes. Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications. ACM, San Diego, CA, USA(2003)
42. Bohge, M., Trappe, W.: An authentication framework for hierarchical ad hoc sensor networks. Proceedings of the 2nd ACM workshop on Wireless security. ACM, San Diego, CA, USA(2003)
43. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. Proceedings of the 9th ACM conference on Computer and communications security. ACM, Washington, DC, USA(2002)
44. Huang, Q., Cukier, J., Kobayashi, H., Liu, B., Zhang, J.: Fast authenticated key establishment protocols for self-organizing sensor networks. Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications. ACM, San Diego, CA, USA(2003)
45. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21** 120-126 (1978)
46. Koblitz, N.: Elliptic Curve Cryptosystems. Mathematics of Computation **48** 203-209 (1987)
47. Caelli, W.J., Dawson, E.P., Rea, S.A.: PKI, Elliptic Curve Cryptography, and Digital Signatures. Computers and Security **18** 47-66 (1999)
48. Daemon, J., Rijmen, V.: AES Proposal: The Rijndael Block Cipher. Proton World Int. Katholieke University(1999)
49. Howard, R.: Data encryption standard. Inf. Age **9** 204-210 (1987)
50. Feistel, H.: Cipher System using a variant key matrix. IBM (US)(1981)
51. Wheeler, D.J., Needham, R.M.: (TEA), A Tiny Encryption Algorithm. Lecture notes in computer science **1108** (1995)
52. Heidari-Bateni, G., McGillem, C.D.: A chaotic direct-sequence spread-spectrum communication system. Communications, IEEE Transactions on **42** 1524-1527 (1994)
53. Burchall-Cooper, W., Nelson, K.P., Jones, D.A., Avery, J.W.: Frequency hopping spread spectrum data communications system. In: USPTO (ed.), U.S.(1993)
54. Wander, A.S., Gura, N., Eberle, H., Gupta, V.A.G.V., Shantz, S.C.A.S.S.C.: Energy analysis of public-key cryptography for wireless sensor networks. In: Gura, N. (ed.): Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on 324-328 (2005)