An Efficient PKC-Based Security Architecture for Wireless Sensor Networks

Md. Mokammel Haque, *Student Member, IEEE,* Al-Sakib Khan Pathan, *Student Member, IEEE,*
Byung Goo Choi, and Choong Seon Hong, *Member, IEEE*

***Abstract*—In spite of previous widely held belief of the incompatibility of public key cryptography (PKC) schemes for wireless sensor networks (WSNs), some recent works have shown that, PKC based schemes could be implemented for such networks in some ways. The major challenge of employing a PKC scheme in wireless sensor network is posed by the limitations of resources of the tiny sensors. Considering this feature of the sensors, in this paper, we propose an efficient PKC based security architecture with relatively less resource requirements than those of the other previously proposed PKC schemes for WSN. Our security architecture comprises basically of two parts; a key handshaking scheme based on simple linear operations and the derivation of decryption key by a receiver node. Our architecture allows both base-station-to-node or node-to-base-station secure communications, and node-to-node secure communications. Analysis and simulation results show that, our proposed architecture ensures a good level of security for communications in the network and could effectively be implemented using the limited computation, memory and energy budgets of the current generation sensor nodes.**

***Index Terms*— Architecture, Pseudoinverse, Public Key, Sensor**

I. Introduction

SECURITY is often viewed as a standalone component of many systems' architectures. But, in case of wireless sensor networks (WSNs), security is more than an important issue that must get commensurate emphasis. The types of services expected from wireless sensor networks often make security as the most important concern for deploying and using such types of networks. In many cases, the tiny sensors in a sensor network are used to collect specific data from particular target areas and the collected data are often considered secret and not to be disclosed in public. Hence, efficient and secure mechanisms are needed to transmit acquired data secretly to the appropriate recipients.

Sometimes wireless sensor networks carry confidential information which if exposed to the adverse units, might cause debacle for the friendly units. Especially in military applications of WSN, employing apropos security mechanisms for data transmissions is very crucial as these data could be used for taking tactical military decisions. If an adversary can thwart the work of the network by perturbing the information produced, stopping production, or pilfering information, then the usefulness of sensor networks is drastically curtailed. Likewise, for example in a disaster management related application, accurate and unmodified data are needed to predict upcoming disaster(s) and to warn the concerned people in advance about the occurrence of event(s).

Ensuring complete and a good level of security for such types of networks however, is not a trivial task. As these sorts of networks use wireless communications, the threats and attacks against WSNs are more diverse and often large in scale. It is not possible to deal with all sorts of security threats with a single mechanism. Rather, a combination of different security schemes for a single network could be the solution. For example, an attack at the physical layer like, jamming [1] could not be handled by any key management scheme. Hence, several mechanisms at different layers could be employed at the same time to provide holistic security [2] for wireless sensor networks and side by side the level of security in the data transmission and communication phase could be increased using efficient key management schemes. Public key cryptography (PKC) could be the best choice for ensuring a satisfactory level of security for data transmissions within the network. However, the major challenge of employing a public key security scheme directly in wireless sensor network is the constrained energy, computation, and memory budgets of sensors participating in the network. Among several public key schemes, Elliptic Curve Cryptography (ECC) based algorithms have a proven and acceptable performance for low-powered sensor nodes [3], [4], [5]. Considering both the software and hardware configurations, elliptical curve based public key cryptography (PKC) has shown relatively better results on 8 bit mote platforms. However, the use of certificates in such a scheme consumes a huge amount of bandwidth and power.

Considering the special characteristics of wireless sensor networks, in this paper, we propose an efficient public key

Manuscript received April 12, 2007. This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (KRF-2006-521-D00394)

Md. Mokammel Haque, Al-Sakib Khan Pathan, and Byung Goo Choi are with the Networking Lab, Department of Computer Engineering, Kyung Hee University, South Korea (phone: +82 31 201-2493; fax: +82 31 204-9082; e-mail: {malinhaque, spathan, bgchoi}@networking.khu.ac.kr).

Dr. Choong Seon Hong is with the Department of Computer Engineering, Kyung Hee University, South Korea. (phone: +82 31 201-2532; fax: +82 31 204-9082; e-mail: cshong@khu.ac.kr).

based security architecture for WSN. In our scheme, we use pseudoinverse matrix for the first part while the second part is a simple method for transferring decryption key to the receiver node. Our analysis and simulation results show that our scheme demonstrates a considerable gain in the level of security and is suitable enough to be employed with the current generation sensor nodes.

The rest of the paper is organized as follows: Section 2 presents the literature review, Section 3 states the preliminaries and assumptions for our security architecture. Section 4 presents our proposed architecture and schemes, Section 5 deals with the analysis and simulation results, and finally, Section 6 concludes the paper stating the achievements from this work with future research directions.

II. Literature Review

Most of the works regarding public key cryptography in wireless sensor networks are conducted to fit the low-power characteristic of sensor nodes. Recently, several works like [4], [5], [6], [7] have addressed or successfully have implemented public key schemes with current generation sensors. Both from software and hardware perspectives, PKC schemes have shown reasonable performances. In this section, we discuss some of these exclusive research works and compare their contributions in this area.

[3] presents the first known implementation of elliptic curve cryptography over F_{2p} for sensor networks based on 8-bit, 7.3828 MHz MICA2 mote. The results show that, public key based scheme is viable for the modern-era sensors. In [4], the authors have conducted a comparative energy analysis upon RSA and ECC based public key algorithms for wireless sensor networks. They have used a simplified version of SSL for mutual authentication and key exchange. For their experiments, they have used Berkley/Crossbow motes platform, specifically the MICA2dots [8]. With the outcomes of their experiments, we see that, contrary to the widely held beliefs, authentication and key exchange protocols using optimized software implementations of public key cryptography are very viable on small wireless devices.

In [5], the authors have proposed C4W which is basically an identity based PKC infrastructure. They have shown that their identity based scheme consumes less energy as it is certificateless and thus it is efficient both in terms of computation and communication costs. The TinyPK system demonstrated in [9] shows that, a public-key based protocol is feasible even for an extremely lightweight sensor network. TinyPK is a software-based implementation of public key system tested on UC Berkeley MICA2 motes.

[6] has shown that special purpose ultra-low power hardware implementations of public key algorithms can be used on sensor nodes. The authors have shown that PKC tremendously simplifies the implementation of many typical security services and additionally reduces transmission power due to less protocol overhead. [6] also provides an in-depth comparison of three different PKC implementations (Rabin's scheme, NtruEncrypt and Elliptic curve) particularly targeted at wireless sensor networks.

A more recent work on hardware implementation of PKC is proposed for elliptic curve over binary extension fields in [7]. The authors have proposed a dedicated coprocessor for certain cryptographic operations. They have shown that a reasonable amount of power can be reserved in this case and thus improved performance could be achieved without degrading other performance parameters. Though the actual data path is 8 bits only, this specific purpose coprocessor can handle operands of even 163 bits.

Other than the above mentioned works, in [10], Du et al. have suggested sparing use of PKC due to its high power consuming characteristic and proposed the use of one-way hash function instead of certificate. Construction of Merkle tree forest from sensors' public keys and selection of height of the tree are the basics of their scheme. They have compared their scheme with other popular PK schemes for sensor networks and plotted the results which show significant performances.

A distributed and cooperative public key authentication is proposed in [11]. It is also a hash key based scheme. In this cooperative mechanism, each node stores a limited number of hashed keys for other nodes which help in the authentication procedure during public key operation. According to [11], this scheme is free from any cryptographic operations which is designed to be fit for the constrained resources of the sensors.

[12] has looked at several additive homomorphic public key encryption schemes and their applicability to WSNs when implemented on computationally limited sensor devices. The authors in this work, have provided recommendations for selecting the most suitable public key schemes based on the topologies and the scenarios of wireless sensor networks.

Reviewing all these works, we are motivated to propose a public key based architecture for secure wireless sensor networks. We have considered the constrained resources of the sensors and have proposed an approach that shows considerable performance with the modern-era sensor node platform. The following section describes our proposed PKC-based security scheme in detail.

III. Our Network Model and Preliminaries

A. Network Model

We assume that, the base station (BS) has enough processing power and energy to do the calculations for the sensors in the network. The base station is a trusted entity and cannot be compromised in any way. The BS also has sufficient storage capacity to support the network. The sensors deployed in the network have the computational, memory, communication and energy resources like the current generation of sensor nodes (e.g., MICA2 motes [8]). Once the sensors are deployed over the target area, they remain relatively static in their respective positions.

B. Pseudoinverse Matrix

The pseudoinverse matrix or generalised inverse matrix

[13], [14] has a very nice property that could be used for cryptographic operations. It is well known that, a nonsingular matrix over any field has a unique inverse. For a general matrix of dimension $k \times n$, there might exist more than one generalized inverse. This is denoted by, $M(k,n) = \{A$: A is a $k \times n$ matrix$\}$. Let, $A \in M(k,n)$. If there exists a matrix $B \in M(n,k)$ such that,

$$ABA = A \text{ and } BAB = B$$

then each of *A* and *B* is called a generalized inverse matrix (or pseudoinverse matrix) of the other. In this paper, we use the notation A_g to denote the generalized inverse matrix of *A*. We use pseudoinverse matrix for the key handshaking process in our security architecture.

It should be noted that, $(A_g)_g = A$ is not always true. The set of all possible pseudoinverse matrices of A is denoted by $\{A_g\}$, and $|\{A_g\}|$ is the cardinality of $\{A_g\}$. Then we have:

Lemma 1: Let A_g be a pseudoinverse matrix of A. Then,

$$rank(A_g) = rank(A)$$

Lemma 2: Let $A \in M(k,n)$ with $rank(A) = k$. If A can be written as $A = [A_1; 0]$, where A_1 is a $k \times k$ nonsingular matrix then,

$$\{A_g\} = \{\begin{bmatrix} A_1^{-1} \\ Z \end{bmatrix} : Z \in M(n-k,k) \text{ is an arbitrary matrix}\}$$

Proof: Let $B = \begin{bmatrix} X \\ Z \end{bmatrix} \in M(n,k)$. It is then easy to verify that both $ABA = A$ and $BAB = B$ hold if and only if $X = A_1^{-1}$.

IV. Our PKC-Based Security Architecture

In this section, we present our public key based security architecture with two separate but interrelated parts. The first part is the key handshaking process, in which each node in the network could derive a secret common key with the base station and the second part is used for confidential and authenticated data transmissions between two participating nodes in the network.

A. Key Handshaking between any Node and Base Station

Let n_i be a node in the network and S be the base station or sink. To derive a shared secret key between the node n_i and the base station, the following operations are performed:

1. Node n_i randomly generates a matrix X with dimension $m \times n$ and its psedoinverse matrix, X_g. These matrices are kept secret in the node.

2. n_i calculates $X_g X$ and sends it to the base station S.

3. In turn, S randomly generates another matrix Y with dimension $n \times k$, and finds out its pseudoinverse matrix Y_g. These matrices are also kept secret in the base station.

4. S calculates $X_g XY$ and $X_g XYY_g$. Then it sends the resultant matrices to n_i.

5. Upon receiving the products of matrices from S, n_i calculates, $XX_g XYY_g = XYY_g$ and sends it back to the base station.

6. Now, both the node n_i and the base station S can compute the common secret key. n_i gets it by calculating $X(X_g XY) = XY$ and the base station gets it by calculating $(XYY_g)Y = XY$. Both of these outcomes (XY) are the same matrix with dimension $m \times k$.

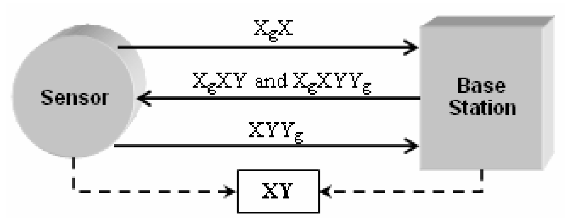

Fig. 1. Key handshaking process between a sensor and base station to derive a shared secret key.

Figure 1 shows the communications between a sensor and the base station in the key handshaking process. Basically, the key XY is locally computed by the node and the base station. Our mechanism ensures that, the individually calculated keys are same and this common key is used for encrypting the messages in the network. Thus, key handshaking process ensures a secure and efficient way of deriving distinct secret key (shared with the base station) for each node taking part in the wireless sensor network. The derived common key could be used for node to base station or base station to node secure communications.

B. Encryption and Decryption of Data for Node-to-Node Communications

The main module in secure node to node communications is a central key generator (CKG) which is located at the base station. The CKG helps any node in the network to decrypt the received encrypted messages from other nodes. If a node n_i wants to send message securely to another node n_j, it uses the key that it has derived using the key handshaking process. Say for example, the encrypted message sent from n_i to n_j is $E_{XY}(M)$. Here, *M* is the message sent from the sender to the receiver. E_{XY} means the message is encrypted with the key XY which is actually the shared secret key between the base station and the sender n_i. Upon receiving the encrypted message, n_j places its own identity and the identity of the sender to the CKG. In turn, CKG generates a decryption key and transmits it to n_j encrypting it with the secret shared key that it has with n_j. As the CKG in the base station has prior knowledge about the shared secret keys of both the nodes, it uses that knowledge to generate the decryption key. Now, n_j first decrypts the encrypted message (i.e., containing the corresponding decryption key) with its shared key, finds out the decryption key and uses that key to decrypt the message sent from node n_i.

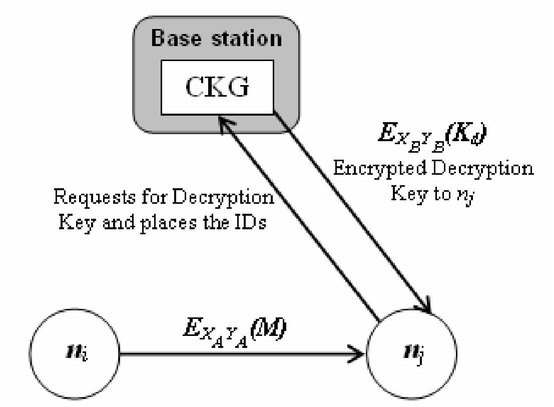
Base station
CKG
$E_{X_B Y_B}(K_d)$
Encrypted Decryption Key to n_j
Requests for Decryption Key and places the IDs
$E_{X_A Y_A}(M)$
n_i
n_j

Fig. 2. Encryption and decryption of message by two communicating nodes in the network.

Figure 2 shows the secure communication method between two nodes in the network. In the figure, $X_A Y_A$ is the shared secret key between n_i and base station, $X_B Y_B$ is the shared secret key between n_j and the base station, and K_d is the decryption key provided by the base station.

V. Performance Evaluation and Comparison

To analyze the performance of our security scheme, we considered the specifications of MICA2dot [8] mote platform. We analyzed the PKC-based architecture in terms of energy cost, memory cost, security and scalability. In this section, we mention the detailed analysis and simulation results of our approach.

In the key handshaking process, we have used linear matrix operations, more specifically matrix multiplication. The complexity of matrix multiplication is very low; hence it could be performed very quickly. In our shared secret key derivation scheme, node n_i sends the base station an $n \times n$ matrix which is of n^2 bits. In turn, the base station sends an $n \times k$ matrix and an $n \times n$ matrix. For this the total number of bits passed for the matrices is, $n^2 + nk = n(n+k)$ bits. Again, the node n_i sends the base station an $m \times n$ bits. So, total number of bits for the matrices transmitted for deriving the shared key in the whole key handshaking process is,

$$n^2 + n(n+k) + mn$$
$$= n(n+n+k+m)$$
$$= n(2n+k+m) \text{ bits}$$

All the calculations here are linear and can be performed very easily.

In the first part, for key handshaking we use the public channel for the message transmission. However, capturing the messages like $X_g X$, $X_g XY$, $X_g XYY_g$ and XYY_g could not be helpful to construct the locally computed secret shared key XY. It might seem that a prospective attack would be by gaining some information of matrix *Y* from the knowledge of $X_g X$ and $X_g XY$. Let us consider, $rank(X) = r$. Let us assume that,

$$X_g X = \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix}$$

Here, I_r is an identity matrix of order $r \times r$. Then, only the first *r* rows of Y can be determined from $X_g XY$. As Y is chosen randomly, there are $2^{(n-r)\cdot k}$ ways to choose the last $(n-r)$ rows of Y. The knowledge of $X_g XYY_g$ might be helpful in determining Y but according to lemma 2 it doesn't help much to the adversary. Without the knowledge of last $(n-r)$ rows of Y, even if X is completely known, the probability for determining the correct value of each element of XY would be $\frac{1}{2}$, considering that any row vector of X has a nonzero element in any of the last $(n-r)$ positions, which is likely when $(n-r)$ is considerably large and X is chosen at random. However, the possibility of getting X from the knowledge of $X_g XYY_g$ and XYY_g is even smaller as $rank(A_g ABB_g) \le rank(A_g A)$. So, it could be assumed that , based on the above analysis of the key handshaking phase, the probability of successful breaking of this scheme is, $2^{-(n-r)\cdot k}$. So, the security of the handshaking scheme is reasonably high for carefully chosen parameters. To ensure that, $2^{(n-r)\cdot k}$ is a large number, n must be considerably larger than r. And this can be guaranteed by making sure that, $m < n$.

A potential attack could arise in the key handshaking process between a node and the base station, if there exists any sort of identification problem of the participating entities during the communications. In our scheme however, this threat is completely eliminated because; (a) the base station is a trusted entity and could not be compromised in any way and (b) the ids of the communicating nodes are checked by the base station before further communications. In the second part of node-to-node communication, when the receiver node requests for the corresponding decryption key, the key is not sent as a plain message, rather it is encrypted with the shared secret key of the receiver. So, in no way, any other adversary can get the decryption keys for a particular sender-receiver pair.

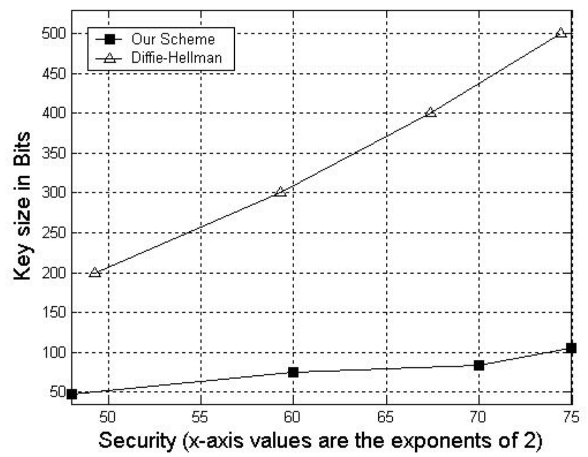

Fig. 3. Required sizes of the key to provide almost same level of security in our key handshaking scheme and Diffie-Hellman key exchange scheme. It is evident that, our handshaking scheme requires much less size key for each of the cases plotted in this figure.

Now, let us compare our shared key derivation scheme with Diffie-Hellman's scheme [15]. In Diffie-Hellman key agreement scheme, a primitive element α over a finite field $GF(p)$ is used. The first user randomly chooses an integer x from $GF(p)$ and sends $\alpha^x \bmod p$ to the second user. In turn, the second user chooses an integer y and sends $\alpha^y \bmod p$ to the first user. Both the users could derive a common secret key $\alpha^{xy} \bmod p$ which is $\log_2 p$ bits in length. And $2\log_2 p$ bits of data are transmitted via the public channel for this key exchange.

Diffie-Hellman (D-H) scheme's security is based on the hardness of discrete logarithm problem such that, given α^x and α^y over the finite field $GF(p)$, how to determine α^{xy}. If x could be found from α^x, which is actually a discrete logarithm problem, then $\alpha^{xy} = (\alpha^y)^x$ could be found. The average computational complexity of this discrete logarithm problem using the best method known so far [16] is $O(\exp((1.923+o(1))(\log_2 p)^{\frac{1}{3}}(\log_2 \log_2 p)^{\frac{2}{3}}))$ bit operations.

Based on the above analysis, we found that, to achieve a security level (complexity) of $2^{49.3}$ in D-H scheme, a key size of 200 bits is needed. On the other hand, to achieve almost the same level security (1/probability) of 2^{48}, 48 bit key is required in our scheme. Likewise, to get security of $2^{59.3}$, $2^{67.4}$ and $2^{74.4}$ in D-H scheme, 300, 400 and 500 bit keys are required respectively. On the other hand, to get security of 2^{60}, 2^{70} and 2^{75} in our approach, 75, 84 and 105 bit keys are required respectively. In this analysis, the sizes of p in bits for D-H scheme are 200, 300, 400 and 500 respectively. For our approach, (m,n,k) are (4,8,12), (5,9,15), (6,11,14) and (7,12,15) correspondingly. Figure 3 shows the level of security to be achieved with required size of the keys in our approach and in D-H scheme. It is clear from the figure that, in all of these cases, our approach needs keys with much less size than that of D-H scheme. In the figure, along the x-axis we have shown the values of the exponent of 2 to plot the security level for various key sizes. That is, in the figure, a value say, 48 along the x-axis indicates 2^{48}. This is done to fit the values in the graph for visual representation.

The amount of traffic carrying the key related information is also dependent on the size of key. Figure 4 plots the key sizes in bits versus the amount of traffic (considering only the key related information) needed to pass through the open public channel in our whole security architecture and Diffie-Hellman scheme. The data plotted in this figure are based on the fact that, same level (or almost same) of security is to be ensured for both of the schemes. From the figure it could be noticed that, in our approach, the key sizes and the amounts of key information traffic are relatively smaller than those of Diffie-Hellman scheme. For example, when the size of the key in bits is only 48 in our scheme, Diffie-Hellman scheme requires 200 bit key to provide the same level of security. So, the key information related traffic is much higher in their scheme than that of our scheme. Considering this case, the amount of traffic in our scheme is 304 bits while in case of Diffie-Hellman scheme, it is 400 bits. As the key size increases, the difference between the amount of key information traffic in these schemes also increases.

Now, as Diffie-Hellman scheme is based on discrete logarithm problem, the eavesdropper's computational capability is considered when the security is analyzed. However, our first part is not a discrete logarithm problem rather it is a probabilistic problem to break the scheme. Hence, the increase of computational capability of the eavesdropper or the man-in-the-middle attacker is not relevant with the possibility of breaking the scheme.

In the second part of our architecture, only two messages are transmitted over public channel. When the receiver node needs the decryption key to decrypt a message from a particular sender node, it requests the base station for the corresponding decryption key. In return, base station encrypts the decryption key with the shared secret key of the receiver node. As the shared secret key is not known to any other node in the network, the decryption key for that particular sender-receiver pair could not be exposed. Now, the problem arises if the shared secret key of a node in the network is somehow compromised. In such a case, the base station revokes the shared key and the key handshaking process is re-initiated for that particular node. If such a compromise happens, even in that case, only one node is affected in the network while all other nodes could continue to operate and transmit their messages.

As any node can get a corresponding decryption key from the base station for any sender node in the network, any pair of nodes in the network could communicate between themselves maintaining the high level of security. As mentioned earlier, for base station to node communications or node to base station communications, the shared secret key derived from the handshaking process is used which takes very little computation and message transmission.

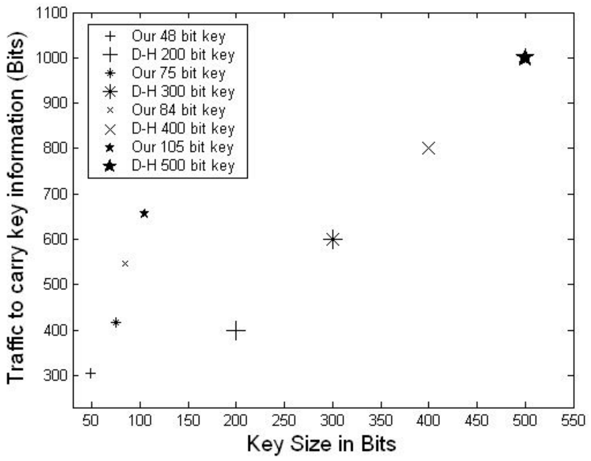

Fig. 4. Key size versus amount of traffic to carry the key information through the public channel in our approach (as a whole) and in Diffie-Hellman (D-H) scheme to ensure almost the same level of security.

In our simulation, we considered the specifications of Berkeley/Crossbow MICA2dot motes [8]. These motes are equipped with 8-bit ATmega128L microcontrollers with 4 MHz clock speed, 128 kB program memory and Chipcon CC1000 low-power wireless transceiver with 433-916 MHz frequency band. The major power consumers in this mote are the processor and the wireless transceiver. During the transmission and reception operations, the microcontroller is turned on alongside the wireless transceiver. According to our calculations, the cost of transmission of one byte is 59.2 μJ while the reception operation takes about half of the transmission cost (28.6 μJ). The power to transmit 1 bit is equivalent to roughly 2090 clock cycles of execution of the microcontroller. In our simulation, we considered a packet size of 41 bytes (payload of 32 bytes, header 9 bytes). With an 8 byte preamble (source and destination address, packet length, packet ID, CRC and a control byte) for each packet we found that, to transmit one packet $49\times 59.2 = 2.9008$ mJ ≈ 2.9 mJ energy is required. Accordingly, the energy cost for receiving the same packet is $49\times 28.6 = 1.4014$ mJ ≈ 1.4 mJ. Considering the same packet size for all the network operations, to set up a shared secret key with the base station each node needs (two transmissions and one reception) $((2\times 2.9)+1.4) = 7.2$ mJ of energy. This cost is one time cost as once the shared secret key is derived, it could be used for the entire lifetime of the network unless the key is exposed. For node to node communication, the sender needs one transmission (2.9 mJ) and the receiver needs two receptions and one transmission $(((2\times 1.4)+2.9) = 5.7$ mJ). As a whole, the entire scheme could be well-afforded by the energy resources of the current generation sensor nodes.

Finally, it should be mentioned that, our approach is well-scalable as any number of new sensors could be added to an existing wireless sensor network based on the requirements. Any newly added node could derive a shared secret key with the base station using the key handshaking process and then it could be used for node-to-node secure communications.

VI. CONCLUSIONS AND FUTURE WORKS

On constructing a complete security architecture for wireless sensor networks, in this paper we have presented an efficient approach which uses the asymmetry of public key cryptosystem for secure communications in the network. We have used different keys for encryption and decryption of the messages for node-to-node communications. Our simulation results and analysis have shown a considerable level of security which is viable with the current generation sensor node platforms. As the keys are generated after deployment of the sensors, no pre-assignment of keys is required and there is no need to store a lookup key table in our approach. Our PKC-based architecture does not require any centralized certificate authority and thus it is free from managing and verifying huge computations associated with certificates. In future, we will combine our work with other security mechanisms to construct a large-scale security architecture for Wireless Sensor Networks.

REFERENCES

[1] Wood, A. D., Stankovic, J. A., and Son, S. H., "JAM: A Jammed-Area Mapping Service for Sensor Networks," *Proceedings of the 24th IEEE International Real-Time Systems Symposium (RTSS'03)*, 2003, pp. 286-297.

[2] Pathan, A.-S. K., Lee, H.-W., and Hong, C. S., "Security in Wireless Sensor Networks: Issues and Challenges," *Proceedings of 8th IEEE ICACT 2006*, Volume II, 20-22 February, Phoenix Park, Korea, 2006, pp. 1043-1048.

[3] Malan, D.J., Welsh, M., and Smith, M.D., "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," *Proceedings of IEEE SECON 2004*, 4-7 October, 2004, pp. 71 – 80.

[4] Wander, A.S., Gura, N., Eberle, H., Gupta, V., and Shantz, S.C., “Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks”, *Proceedings of PerCom 2005*, pp. 324-328.

[5] Jing, Q., Hu, J., and Chen, Z., “C4W: An Energy Efficient Public Key Cryptosystem for Large-Scale Wireless Sensor Networks”, *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, Oct. 2006, pp. 827-832.

[6] Gaubatz, G., Kaps, J.-P., Ozturk, E., and Sunar, B., "State of the Art in Ultra-Low Power Public Key Crytography for Wireless Sensor Networks," *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops, 2005*, pp. 146-150.

[7] Bertoni, G., Breveglieri, L., and Venturi, M., “Power Aware Design of an Elliptic Curve Coprocessor for 8 bit Platforms,” *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06)*, p. 337.

[8] Xbow Sensor Networks, Available at: http://www.xbow.com/

[9] Watro, R., Kong, D., Cuti, S.-f., Gardiner, C., Lynn, C. and Kruus, P., "TinyPK: Securing Sensor Networks with Public Key Technology," *ACM SASN'04*, Washington, DC, USA, 2004, pp. 59-64.

[10] Du, W., Wang, R., and Ning, P., "An Efficient Scheme for Authenticating Public Keys in Sensor Networks," *Proceedings of ACM MobiHoc'05*, Illinois, USA, 2005, pp. 58-67.

[11] Nyang D. and Mohaisen A., “Cooperative Public Key Authentication Protocol in Wireless Sensor Network,” *UIC 2006, LNCS 4159*, Springer-Verlag, pp. 864-873, 2006.

[12] Mykletun, E., Girao, J., and Westhoff, D., "Public Key Based Cryptoschemes for Data Concealment in Wireless Sensor Networks," *Proceedings of IEEE International Conference on Communications*, 2006, pp. 2288-2295.

[13] Israel, A.B., and Greville, T.N. E., *Generalized inverses: theory and applications*, John Wiley & Sons, New York, 1974.
[14] Boullion, T.L., and Odell, P.L., *Generalized inverse matrices*, Wiley-Interscience, New York, 1971.
[15] Rhee, M. Y., *Internet Security: Cryptographic Principles, Algorithms and Protocols*, WILEY, 2003.
[16] Menezes, A. J., Oorschot, P. C. V., and Vanstone, S. A., *Handbook of applied Cryptography*, CRC Press, 1997.