Realizing Stateful Public Key Encryption in Wireless Sensor Network

Joonsang Baek, Han Chiang Tan, Jianying Zhou and Jun Wen Wong

Abstract In this paper, we present our implementation of a stateful public key encryption (stateful PKE) scheme in the wireless sensor network (WSN) environment. In order to reduce the communication overhead of the stateful PKE scheme we implement, which is of prime importance in WSN, we introduce a technique called "indexing". The performance analysis of our implementation shows that there are significant advantages of using stateful PKE in WSN in terms of computation and communication costs, compared with normal public key encryption.

1 Introduction

1.1 Motivation

Wireless sensor networks (WSNs) are useful in a variety of domains, including monitoring the integrity of buildings and building automation, early discovery of catastrophes (like forest fires and earthquakes), medical surveillance and remote diagnosis, pollution control and the battlefield and perimeter defense.

In the typical setting, a WSN consists of numerous tiny nodes communicating with a few base stations. Among those tiny nodes, there can be some nodes which have more computation and/or communication capacity. The base stations are often

Joonsang Baek
Institute for Infocomm Research, Singapore, e-mail: jsbaek@i2r.a-star.edu.sg

Han Chiang Tan
Institute for Infocomm Research, Singapore, e-mail: hctan@i2r.a-star.edu.sg

Jianying Zhou
Institute for Infocomm Research, Singapore, e-mail: jyzhou@i2r.a-star.edu.sg

Jun Wen Wong
Institute for Infocomm Research, Singapore, e-mail: jwwong@i2r.a-star.edu.sg

assumed to be powerful enough to perform computationally intensive tasks such as cryptographic computations. The sensor nodes, on the other hand, have constrained resources in terms of computation, memory and battery power.

Although WSN brings us a great variety of applications as mentioned above, it is fairly vulnerable to attacks such as eavesdropping and impersonation as sensor nodes are often deployed in physically accessible areas and often interact with environments and people. As a result it has become of prime importance to provide security services for WSNs including data encryption and node authentication.

Not long ago public key cryptography (PKC) was believed to be unsuitable for providing security services in WSN as PKC usually requires computationally-intensive cryptographic operations while sensor nodes are severely resource constrained [16]. Contrary to this common belief, it has recently been reported that PKC is in fact feasible to be realized in WSNs [9][20][21].

In this paper, we focus on the realization of PKC in WSN, specifically, efficient implementation of public key encryption for the *confidentiality* service in WSN. Before presenting our contribution, we review the previous work in this line of research.

1.2 Related Work

Although its realization on WSN is challenging, PKC will bring great simplicity and efficiency in providing a number of essential security services [10]. In this section we briefly survey the related work on implementation of PKC in WSNs.

Watro et al. [21] designed and implemented public key based protocols that allow authentication and key agreement between a sensor network and a third party as well as between two sensor networks. The specific public key algorithm they used is RSA [15] whose key size varies (512, 768 and 1024 bits). Their protocols were implemented on UC Berkeley Mica2 motes using the TinyOS [19] environment.

Wander et al. [20] presented implementation of authentication and key exchange protocols based on public-key cryptography on Atmel ATmega128L low-power 8-bit microcontroller platform. Two base algorithms for their work are RSA-1024 (RSA with 1024-bit key size) and ECC-160 (Elliptic Curve Cryptography with 160-bit key size). It was reported in their paper that ECC has a significant advantage over RSA as it reduces computation time and also the amount of data transmitted and stored.

Bellare et al. [4] discussed how to significantly speed-up the public key encryption (PKE) by simply allowing a sender to maintain "state" that is re-used across different encryptions. This new type of PKE is called *stateful PKE*. As an efficient construction, Bellare et al. presented a stateful PKE scheme based on the Diffie-Hellman assumption (Given g^a, g^b, it is computationally infeasible to compute g^{ab}).

1.3 Our Contributions

From the literature review in the previous subsection, one can notice that due to the efficiency that it could provide, ECC can be a good candidate for the algorithm that realizes PKE in WSN, which consists of sensor nodes with limited resources for computation/communication. One can also notice that stateful PKE could bring further improvement on the realization of PKE in WSN.

Having these in mind, we make the following contributions in this paper:

- In order to enhance *communication* efficiency of Bellare et al.'s [4] Diffie-Hellman (DH) based stateful PKE scheme, we modify it using a simple but useful "indexing" technique whereby the repeated part of ciphertext, which is usually long, is replaced by a short string.
- We implement the modified version of the DH based stateful PKE scheme on the MicaZ [8] platform and analyze its security and performance. To our knowledge, this is the first implementation of stateful PKE in WSN.

2 Our Modified DH-Based Stateful PKE for WSN

2.1 Basic Setting

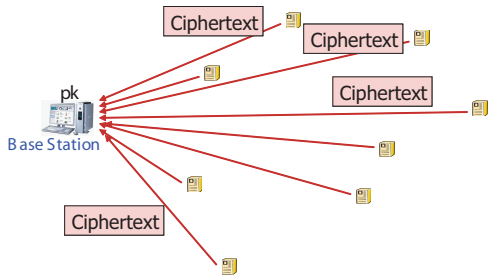

Fig. 1 Overview of Basic Setting

Security services for WSNs can vary depending on the specific requirements of each application. For our implementation, we consider a simple (single-hop) but widely applicable security service architecture in which each sensor node can encrypt data using a base station's public key *pk* as depicted in Figure 1. We assume that the public key of the base stations are embedded in each sensor node when they are deployed. On receiving each ciphertext from each sensor node, the base station uses the corresponding private key *sk* to decrypt it.

Like the case for general WSNs, we assume that the base station is powerful enough to perform computationally intensive cryptographic operations, and the sen-

sor nodes, on the other hand, have constrained resources in terms of computation, memory and battery power. We also assume that the private key of the base station is safely stored, e.g., using smart card.

One of the advantages of employing public key encryption in this setting, where the sensor nodes do not have to perform decryption, is that a long-term private key does not need be stored inside each sensor node. In contrast, if one wants to run some key exchange protocol to share symmetric key between the base station and the sensor node and encrypts subsequent messages using the shared key, the shared key of the sensor node ought to be protected securely. (Otherwise, an attacher that has obtained the *long-term* shared key from the sensor node can freely decrypt the subsequent ciphertexts as well as the ciphertexts obtained before.)

Of course other more complex settings such as multi-hop exist and the security services for these settings should be difficult to realize. However, the focus of this paper is mainly realizing stateful PKE in the basic WSN setting we have described above, which itself is challenging due to resource constraints of sensors, rather than advocating that our scheme can solve every security (confidentiality) problem in WSN.

2.2 Some Preliminaries

Not surprisingly, there are few *public key* cryptographic tools that we can easily employ to realize the security services in WSN due to the high level of resource-constraints in WSN. Nonetheless, the following cryptographic primitives can be useful:

- Elliptic curve cryptography (ECC): Since its introduction in late 80's [13][14], ECC has attracted much attention as the security solutions for wireless networks due to the small key size and low computational overhead. It is an established fact that ECC-160 offers a similar level of security of RSA-1024. As mentioned earlier, Wander et al. [20] showed that ECC has significant advantages over RSA in the WSN setting.
- Hybrid encryption: Recall that in our basic setting presented previously, each sensor node has to send encrypted data to the base station. It is a well-known fact that normal PKE schemes solely constructed from number-theoretic primitive are too slow to encrypt a large amount of data. Hence hybrid encryption is used in practice. In a hybrid encryption scheme, a session key is generated by a public key algorithm called "Key Encapsulation Mechanism (KEM)" [12] and actual data is encrypted by a symmetric encryption algorithm specifically called "Data Encapsulation Mechanism (DEM)" [12] under the generated session key. It is shown [7] that this hybrid encryption scheme is secure against chosen ciphertext attack (CCA-secure) if both KEM and DEM are CCA-secure.

One of the PKE schemes that are based on the above primitives is Abdalla et al.'s [1] DHIES (Diffie-Hellman Integrated Encryption Scheme). But Bellare et al. has

shown that DHIES can further be improved using the "stateful encryption" concept, which will be explained shortly.

2.3 Diffie-Hellman Based Stateful PKE with Indexing

Stateful PKE [4] could be understood as a special type of hybrid encryption which significantly speeds up the KEM-part of hybrid encryption by allowing a sender to maintain state which is reused across different encryptions. For example, Bellare et al's [4] DH based stateful PKE scheme, which is a stateful version of DHIES, works as follows. To encrypt a message M, the encryption algorithm computes $(rP, \mathsf{E}_K(M))$, where r is chosen at random from $\mathbb{Z}_q$ (q, a prime), $K = \mathsf{H}(rP, Y, rY)$ (P, a generator of the ECC-group of order q; H, a hash function; E, a CCA-secure symmetric encryption function) and $Y = xP$ (x, a private key; Y, a public key). Now, the value r is kept as state and rP and K do not need to be computed every time a new message is encrypted. In this way, stateful PKE brings computational efficiency gains.

But we argue that this scheme can further be improved to save energy for communication. As the sensor nodes lack sufficient amount of energy, reducing communication overhead is also of prime importance. (According to [20], power to transmit *one* bit is equivalent to approx. 2,090 clock cycle of execution on the microcontroller.) Hence, the repeated transmission of the same value $U = rP$ for a number of different sessions would be a waste of the communication resource.

Our approach to resolve this problem is to employ a natural but useful "indexing" technique whereby the value U is replaced by a much shorter string. – In our implementation, for example, the length of U is 21 bytes and the index for this which we denote by id_U is only 3 bytes. To uniquely identify U using id_U, we use an identity of a sensor node and a sequence number. (This will be explained in detail in Section 3.1.) Also, to protect id_U from modification by attackers, we hash it with a KEM-key. More precisely we describe our modified DH based stateful PKE scheme as follows. - Along with this description, readers are refered to Figure 2.

- **Setup**: The base station does the following:
  Pick a group $\mathbb{G}$ of prime order q;
  Pick a generator P of $\mathbb{G}$;
  Pick a hash function H;
  Pick a symmetric encryption scheme $\mathsf{SYM} = (\mathsf{E}, \mathsf{D})$;
  Pick x at random from $\mathbb{Z}_q^*$ and compute $Y = xP$;
  Return $pk = (q, P, Y, \mathsf{H}, \mathsf{SYM})$ and $sk = (pk, x)$ // pk and sk denote public key and private key resp.
- **I-Phase** (Indexing Phase): Using pk, a sensor node performs the following to encrypt a plaintext M:
  Pick $r \in \mathbb{Z}_q^*$ at random and compute $U = rP$;
  Pick an index $id_U \in \{0,1\}^*$ for U in such a way that id_U uniquely identifies U;

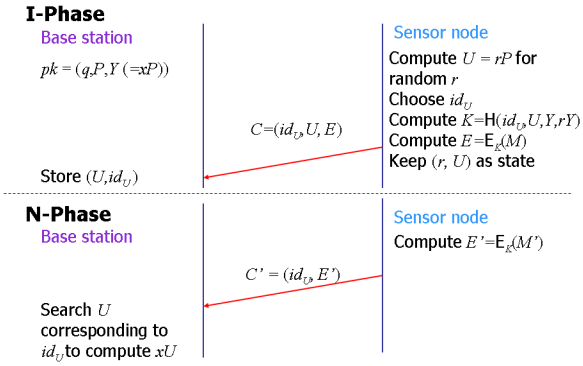

Fig. 2 Overview of DH-Based Stateful PKE with Indexing

Compute $K = \mathsf{H}(id_U, U, Y, rY)$;
Compute $E = \mathsf{E}_K(M)$; // $\mathsf{E}_K(\cdot)$ denotes symmetric encryption function under key K
Keep (r, U) as state;
Return $C = (id_U, U, E)$ as ciphertext

Note that the size of id_U is much smaller than that of U. Note also that the node can cache K to save computation further.

Upon receiving $C = (id_U, U, E)$ from the sensor node, the base station performs the following to decrypt it:

Compute $xU = xrP$ and $K = \mathsf{H}(id_U, U, Y, xU)$;
Compute $M = \mathsf{D}_K(E)$; // $\mathsf{D}_K(\cdot)$ denotes the symmetric decryption function under key K
Return M

Note in the above algorithm that M can be $\perp$ (meaning "reject").

- **N-Phase** (Normal Phase): In this phase, the sensor node performs the following to encrypt a plaintext M':

  Compute $E' = \mathsf{E}_K(M')$;
  Return $C' = (id_U, E')$ as ciphertext

  Upon receiving $C' = (id_U, E')$, the base station conducts the following to decrypt C':

  Search its database for U that corresponds to id_U;
  If the corresponding U does not exists, return $\perp$
  Else compute $xU = xrP$, $K = \mathsf{H}(id_U, U, Y, xU)$ and return $M' = \mathsf{D}_K(E')$

We remark that the choice of id_U is very important. For instance, if it were chosen at random, it would collide with $id_{U'}$ that other sensor node has chosen for other "U'" value. In this case, the base station cannot decrypt a given ciphertext as there is an ambiguity as to which one is correct. For this reason, id_U ought to uniquely identify the value U. In Section 3, we will describe how to select id_U in details.

2.4 Security Analysis

The security against chosen ciphertext attack (CCA) for stateful PKE is defined in [4], which extends the usual IND-CCA (Indistinguishability under CCA [5]) notion of normal PKE. The essence of this security definition is that an adversary does not get any significant advantage in breaking the confidentiality of ciphertext even though he uses the same state to encrypt messages for multiple receivers.

We now prove that our modified DH based stateful PKE scheme is also secure under this security definition. A basic idea of the proof[1] is that even though an attacker can replace the index of a challenge ciphertext (a ciphertext that the attacker wants to break the confidentiality) with its own, it cannot break the confidentiality (indistinguishability of encryption) since, intuitively, the hash function H prevents id_U from alteration. Formally we prove the following thorem.

Theorem 1. *Assume that the underlying symmetric encryption scheme* E *is IND-CCA secure and the hash function* H *is random oracle [6]. Then our stateful PKE scheme proposed above is secure against CCA in the sense defined in [4] under the assumption that the Gap Diffie-Hellman (GDH) problem is computationally intractable. (The GDH problem refers to a computational problem where an adversary, given* (P, aP, bP) *for random* $a, b \in \mathbb{Z}_q$*, tries to compute a DH-key* abP *with the help of DH-oracle, which, given tuple* (P, aP, bP, cP)*, can decide whether* $c = ab$ *or not.)*

Proof. (Sketch) Let A and B be a CCA adversary and an adversary for GDH respectively. Assume that B is given (P, aP, bP) as instance. B sets $U^* = aP$ and $Y_1 = bP$, where Y_1 denotes the receiver 1's public key. B picks K^* at random from the appropriate key space and sets $K^* = \mathsf{H}(id_{U^*}, U^*, Y_1, ?)$, where ? denotes "indeterminate". When A queries receiver i's public key Y_i, where $2 \leq i \leq n$, as public key registration query, B picks K_i at random from the appropriate key space and defines $K_i = \mathsf{H}(id_{U^*}, U^*, Y_i, ?)$.

Now, when A queries (id_U, U, Y, D) to H, B answers as follows:

- Pick K at random
  If $id_U = id_{U^*}, U = U^*$ and D is a DH-key of (U, Y) then

  If $Y \neq Y_1$ then
  If $Y = Y_i$ for some $i \in [2, n]$ then return K_i (which was selected by B in the beginning) as answer
  Else pick K_i at random, set $K_i = \mathsf{H}(id_{U^*}, U^*, Y_i, D)$. and return K_i as answer
  Else abort the game and return D as DH-key of $U^* (= aP)$ and $Y_1 (= bP)$

Note that in the above simulation of H, B keeps a query-answer list, which we denote by H-List. B deals with the rest of the queries from A as follows.

When A asks for encryption of (i, M), B searches appropriate K_i, computes $E = \mathsf{E}_{K_i}(M)$ and returns (id_{U^*}, U^*, E) to A as answer. When A queries (M_0, M_1)

[1] Note that the "proof" here means the "reductionist proof" [2] widely used to provide security arguments for various schemes and protocols.

as a challenge, B picks $b \in \{0,1\}$ at random, computes $E^* = \mathsf{E}_{K^*}(M_b)$ and returns (id_{U^*}, U^*, E^*) to A as a challenge ciphertext.

There are two types of decryption queries. When A queries (id_U, U, E) for decryption, B first checks whether U is an element of group $\mathbb{G}$. If it is not, B sends off $\perp$ to A, otherwise it conducts the following:

If $id_U = id_{U^*}$ and $U = U^*$ then return $\mathsf{D}_{K^*}(E)$
Else search $K = \mathsf{H}(id_U, U, Y_1, ?)$ from the H-List

If it exists return $\mathsf{D}_K(E)$
Else pick K at random and return $\mathsf{D}_K(E)$ and update the query-answer list for H with K

When A queries (id_U, E) for decryption, B searches the H-List for U. If such U does not exist, B returns $\perp$ to A, otherwise, it conducts the same procedure as described above.

3 Our Implementation

3.1 Implementations of Symmetric Encryption and Index id_U

As shown in the preceding section, for the modified DH based stateful PKE scheme to be secure, we need to use an IND-CCA secure symmetric encryption scheme. We select the IND-CCA secure symmetric encryption scheme (DEM 3) recommended by ISO standard [12], which can be described as follows.

- Encryption: First, this algorithm splits the key $K(= \mathsf{H}(id_U, U, Y, rY))$ into K_1 and K_2 such that $K = K_1||K_2$, where the length of K_1 is the same as the length of a plaintext M and the length of K_2 is appropriate for the key length of Message Authentication Code function MAC. Next, this algorithm computes $S = K_1 \oplus M$ and $\sigma = \mathsf{MAC}_{K_2}(S)$. Finally, it outputs a ciphertext $E = (S, \sigma)$. (Note that the particular MAC scheme used in our implementation is HMAC [3].)
- Decryption: On input $E = (S, \sigma)$ and the key K, this algorithm computes $K = K_1||K_2$ and checks whether $\sigma = \mathsf{MAC}_{K_2}(S)$. If it is, this algorithm returns $M = S \oplus K_1$ otherwise, returns $\perp$.

As remarked at the end of Section 2.3, selecting id_U so that it uniquely identifies the value U is important. In our implementation, we construct id_U as follows:

$$id_U = ID_{\text{node}}||N,$$

where ID_{node} denotes a unique identity of a sensor node and N denotes a sequence number for the current value U. The size of ID_{node} and N is 2 bytes and 1 byte respectively. In our implementation, the base station is set to replace id_U (in its

database) with new one whenever N changes. - Consequently, only *one* id_U exists for *each* sensor node.

Note that since each sensor node has a different identity ID_{node}, it is not possible to find a collision if less than $2^{16}(=65536)$ sensor nodes exist. Hence, if constructed in the way described above, id_U uniquely identifies U depending on how many sensor nodes should be deployed. (One can of course enlarge the size of ID_{node} to increase the number of sensor nodes that can be deployed. Even if one byte is stretched, the number of deployable senor nodes increases dramatically.)

Item	Value
Transmission Frequency	2450 MHz
Transmission Power	0 dBm (=1 mW)
Data Rate	250 kbps
Energy to Transmit (Measured)	**1.56 μJ/byte**

Table 1 Characteristic Data for MicaZ

3.2 Performance Analysis

The WSN platform on which our implementation is based is MicaZ, developed by Crossbow Technology [8]. The RF transceiver for this MicaZ complies with IEEE 802.15.4/ZigBee, and the 8-bit microcontroller is Atmel ATmega128L, which is the major energy consumer. We use a laptop PC (Lenovo T60 1.83GHz (Intel Core 2) CPU, 512MB RAM) as a base station.

In Table 1, we summarize some characteristic data for the MicaZ platform, which include the energy to transmit one byte, which we measured.

Fig. 3 Data Format

The programming languages we used for our implementation are nesC, C and Java (mainly used for interface design on base station). The base operating system for the MicaZ platform is TinyOS [19]. The ECC component of our stateful PKE is based on TinyECC [18], which we modify for our implementation. The size of key for ECC is 160 bits.

Figure 3 illustrates the data format of a packet in our implementation. We assume that the size of each packet be 50 bytes, 5 bytes for the header and 45 bytes for the payload.

We now analyze the computational overhead of our implementation. In Table 2, we summarize and compare the energy consumptions of our implementation when a plaintext message of 20 bytes is encrypted in I-Phase and N-Phase respectively. – It would not be surprising that I-Phase needs much more energy than N-Phase since I-Phase includes two point-multiplications in order to compute $U(=rP)$ and rY, which are *not* needed in N-Phase where U and rY are reused.

	Energy cost
Encryption in I-Phase	46.76 mJ
Encryption in N-Phase	0.89 mJ

Table 2 Comparison between the energy consumptions for encryption of a 20-byte plaintext in I-Phase and N-Phase

Another interpretation of this result is that the modified DH based stateful PKE scheme we implement actually saves significant amount of energy compared with its non-stateful version in which U and rY should be computed according to the randomness of r every time a new message is encrypted. In other words, in the non-stateful version, I-Phase is repeated whenever a new plaintext message is inputted to the encryption function. – In fact, I-Phase of our modified DH based stateful PKE scheme is almost the same[2] as Bellare et al.'s DHIES [1], a normal PKE scheme based on the GDH problem, from which the DH based stateful PKE scheme in [4] is derived.

Based on this observation, we can compare the encryption cost of our modified DH based stateful PKE scheme with that of the non-stateful version. Using the measured energy costs for encryption presented in Table 2 and assuming encryption is conducted 10 times, we demonstrate the comparison between the energy cost of encryption in our modified DH based stateful PKE scheme (simply termed "stateful PKE") and that of its non-stateful version (simply termed "non-stateful PKE") in Figure 4. Notice that the non-stateful version consumes roughly 8.5 times more energy than stateful one.

Before analyzing the communication overhead, we remark that the performance of our implementation is comparable to those of the ECDSA implementation on MicaZ presented in [18] and the DH-key exchange implementation on Mica2dot presented in [20]. According to [18], the measured energy costs of signature generation and verification of the ECDSA implementation are 46.2 mJ and 58.4 mJ respectively when using 160-bit key. Also, it is reported in [20] that the energy cost for the DH-key generation is 22.3 mJ. (Note that the DH-key generation involves one point multiplication while our implementation of the DH based stateful PKE scheme needs two point multiplications in encryption.)

We now analyze the communication overhead. Recall that we use a packet size of 50 bytes. However, 50 bytes are not enough to send all the required data in I-Phase

[2] The only difference is that id_U should be chosen and hashed together with the Diffie-Hellman key.

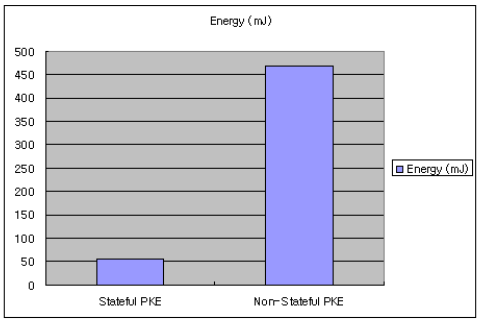

Fig. 4 Comparison between the energy costs of encryption in our modified DH based stateful PKE and its non-stateful version assuming that encryption is conducted 10 times

as the value U, which takes up 21 bytes, needs to be transmitted in this phase. So, in the actual implementation, we make a sensor node send actually two packets in I-Phase. Since U is replaced by id_U which is 3 bytes in length in N-Phase, we do not need to send two packets. On the other hand, if this "indexing" method is not used, U should be transmitted every time a new message is encrypted and hence, two packets should be transmitted every time. More precisely, we can obtain the energy consumption for transmitting two packets 1 time and one packet $n_t - 1$ times by computing

$$\begin{aligned} 1.56\mu\text{J/byte} \times \big((50+50)+50(n_t-1)\big) \text{ bytes} \\ = 78(n_t+1)\mu\text{J} \end{aligned}$$

and the energy for transmitting two packets n_t times by computing

$$1.56\mu\text{J/byte} \times (2 \cdot 50n_t) \text{ bytes} = 156n_t \ \mu\text{J},$$

where n_t denotes the total number of transmissions.

The implication of this result is that our indexing method can save at least 45% of transmission energy when using stateful PKE as illustrated in Figure 5.

4 Concluding Remarks

In this paper, we presented another positive result regarding the feasibility of public key cryptography (PKC) in WSNs: We successfully implemented a statful PKE scheme on MicaZ node [8].

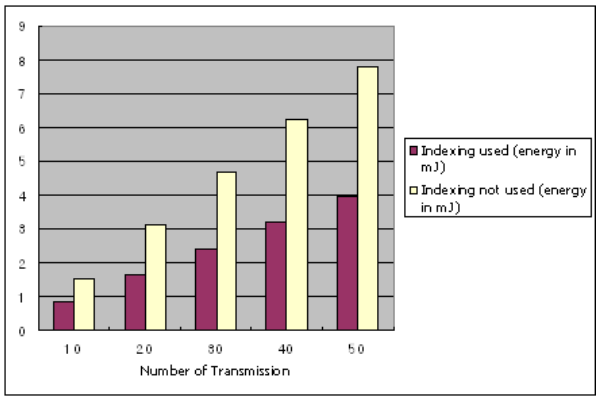

Fig. 5 Comparison between the energy costs of communication when the indexing method is used and when it is not

To enhance the communication efficiency of the stateful PKE, which is very important in WSN, we introduced a technique called "indexing". The performance analysis of our implementation showed that our indexing technique reduced the communication overhead significantly. An interesting direction of research on the indexing technique would be to provide different designs of index (id_U) for different purposes.

Finally, we note that this work focused only on how to provide confidentiality service for WSNs using PKC. How to provide authentication service for WSN using PKC is interesting future work.

Acknowledgements The authors are grateful to the anonymous referees of IFIP SEC '08 for their helpful comments. This work is partially funded by the EU project SMEPP-033563.

References

1. M. Abdalla, M. Bellare and P. Rogaway, *The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES*, In CT-RSA '01, LNCS 2020, pp. 143–158, Springer-Verlag, 2001.
2. M. Bellare, *Practice-Oriented Provable-Security*, Lectures on Data Security – Modern Cryptology in Theory and Practice, LNCS 1561, pp. 1–15, Springer-Verlag 1999.
3. M. Bellare, R. Canetti and H. Krawczyk, *Keying Hash Functions for Message Authentication*, In Crypto '96, LNCS 1109, pp. 1–15, Springer-Verlag, 1996.
4. M. Bellare, T. Kohno and V. Shoup, *Stateful Public-Key Cryptosystems: How to Encrypt with One 160-bit Exponentiation*, In ACM-CCS 2006, pp. 380–389, 2006.
5. M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, *Relations Among Notions of Security for Public-Key Encryption Schemes*, In Crypto '98, LNCS 1462, pp. 26–45, Springer-Verlag, 1998.
6. M. Bellare and P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, In ACM-CCS '93, pp. 62–73, ACM, 1993.

7. R. Cramer and V. Shoup, *Design and Analysis of Practical Public-key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack*, SIAM Journal of Computing 33, pp. 167–226, 2003.
8. MicaZ Wireless Sensor Network Platform, Crossbow Technology, http://www.xbow.com/
9. G. Gaubatz, J.-P. Kaps and B. Sunar, *Public Key Cryptography in Sensor Networks Revisited*, In European Workshop on Security in Ad-Hoc and Sensor Networks 2004 (ESAS '04), LNCS 3313, pp. 2–18, Springer-Verlag, 2005.
10. G. Gaubatz, J.-P. Kaps, E. Oztruk and B. Sunar, *State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks*, In IEEE International Workshop on Pervasive Computing and Communication Security 2005 (PerSec '05), 2005.
11. J. Hoffstein, J. Pipher, J. Silverman, *NTRU: A Ring-Based Public Key Cryptosystem*. In Algorithmic Number Theory (ANTS III), LNCS 1423, pp. 267–288, Springer-Verlag, 1998.
12. ISO 18033-2, *An Emerging Standard for Public-Key Encryption*, Working Group 2 of ISO/IEC JTC 1/SC27, 2004.
13. N. Koblitz, *Elliptic Curve Cryptosystems*, Mathematics of Computation 48, pp. 203–209, 1987.
14. V. Miller, *Use of Elliptic Curves in Cryptography*, In Crypto '85, LNCS 218, pp. 417–426, Springer-Verlag, 1986.
15. R. Rivest, A. Shamir, and L. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM 21 (2), pp. 120–126, 1978.
16. A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, D.E. Culler, *SPINS: Security Protocols for Sensor Networks*. Wireless Networks 8 (2002), pp. 521–534, 2002.
17. M.O. Rabin, *Digitalized signatures and Public Key Functions as Intractable as Factorization*. Mit/lcs/tr-212, Massachusetts Institute of Technology, 1979.
18. TinyECC, http://discovery.csc.ncsu.edu/software/TinyECC/
19. TinyOS, http://www.tinyos.net/
20. A. Wander, N. Gura, H. Eberle, V. Gupta and S Shantz, *Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks*, In IEEE International Conference on Pervasive Computing and Communication 2005 (PerCom '05), pp. 324–328, IEEE Computer Society, 2005.
21. R. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn and P. Kruus, *TinyPK: securing sensor networks with public key technology*, In ACM workshop on Security of ad hoc and sensor networks 2004 (SASN '04), pp. 59–64. ACM Press, 2004.