# An Intrusion and Random-Number-Leakage Resilient Scheme in Mobile Unattended WSNs

Tatsuro Iida, Keita Emura, Atsuko Miyaji and Kazumasa Omote

*Japan Advanced Institute of Science and Technology (JAIST)*
*1-1, Asahidai, Nomi, Ishikawa, 923-1292, Japan*
*Email: {s0910001,k-emura,miyaji,omote}@jaist.ac.jp*

***Abstract*—In INFOCOM 2010, Pietro, Oligeri, Soriente, and Tsudik (POST) proposed an intrusion-resilient system with forward and backward secrecy in mobile Unattended Wireless Sensor Networks (UWSNs), where sensors move according to some mobility model (random jump model and random waypoint model). In the POST scheme, each sensor encrypts its ephemeral key $K$ as a plaintext by using the sink's public key, and sends this ciphertext and the encrypted sensed data by $K$. Although the POST scheme recommends the hybrid encryption, it does not follow the conventional hybrid encryption usage, i.e., the POST scheme is not necessarily secure. More concretely, $K$ must be regarded as a plaintext of the underlying public key system, and therefore the POST scheme requires at least one more encryption procedure (i.e, encryptions for both $K$ and the data) compared with the conventional hybrid encryption procedure. In this paper, we scrutinize the original POST intrusion-resilient system. We set deployed information as a seed used for generating a random number (which is applied for public key encryption). This procedure follows the conventional hybrid encryption usage, and random-number-leakage problem does not occur. In conclusion, we improve the POST scheme from the viewpoint of both security and efficiency without spoiling significant benefit points of the original one.**

***Keywords*-wireless sensor networks (WSNs), hybrid encryption, random-number-leakage problem**

## I. Introduction

### A. Wireless Sensor Networks (WSNs)

Wireless Sensor Networks (WSNs) are ad hoc networks that can consist of a lot of sensor nodes and trusted third party (sink). Each sensor is a small battery-powered device with limited memory and computational ability, and a short-range wireless radio. The sensors might be deployed within a certain area and their activity is usually measured and forwarded to sink. Much of the prior research for WSNs included routing, security, powerawareness, data abstraction, and so on.

A common assumption in most previous research of WSNs has been that data collection is performed in a more or less real time fashion. In such WSNs, a sink is assumed to be always present. The sink is able to query the WSNs in real time and obtain a reply. While it may well be true that many WSNs operate in such general settings. However, there remains a segment of WSNs and applications that do not apply the real time model. It envisions WSNs operating in unattended and hostile environments [1], [2], [3]. In this unattended WSNs (UWSNs), the sink visits the network with irregular and even unpredictable frequency. Consequently, each sensor must retain its data for a considerable time. Intervals between successive sink visits represent periods of vulnerability. While the sink is away, the adversary can easily compromise a number of sensors, learn all memory (e.g., sensed data and secret key) and all communication.

To protect the sensed data or transmitted information, many prior UWSNs schemes used cryptographic techniques. As above, sensor resource is limited. Therefore, public key encryption was shunned by the sensor security community because of its high cost, and most cryptography for WSNs is symmetric cryptography. However, recent developments make public key encryption feasible on commodity sensors [4], [5].

### B. Intrusion-Resilience in Mobile Unattended WSNs

In INFOCOM2010, Pietro, Oligeri, Soriente, and Tsudik [6] proposed intrusion-resilient system in mobile unattended wireless sensor networks (WSNs), where sensors move according to some mobility model (random jump model and random waypoint model), called the POST scheme. Briefly, the flow of the POST scheme is described as follows: First, a sensor collects the data considered in the underlying system. Next, a sensor encrypts its data by using an ephemeral key $K$, which is generated by the internal state contained in the sensor. Finally, the internal state is updated by using all neighbor sensor's deployment information on wireless network. Since an adversary cannot expect each sensor's behavior, the adversary cannot guess the ephemeral key. Then, each sensor encrypts its ephemeral key $K$ as a plaintext by using the sink's public key, and sends this ciphertext and the encrypted sensed data by the ephemeral key to the sink.

The POST scheme handles forward secrecy and backward secrecy, and self-healing properties. In the case of WSNs, the three properties are important to recover from attacks on WSNs. "Forward secrecy" is a property to prevent a past

secret information leakage from a present secret information. If it satisfies this property, adversary cannot compute any secrets used in prior rounds from present secrets. "Backward secrecy" is a property to prevent a future secret information leakage from a present secret information. If it satisfies this property, adversary cannot compute any secrets used in future rounds from present secrets. "Self healing" is a property to recover compromised secret information. Thus, even if adversary compromise node's secret information, node recover own secret information, and revoke compromised secret information.

### C. Problems of the POST scheme

Even if a cryptographic primitive is secure, the system is not necessarily secure if how to use it is wrong. It is necessary to use a secure cryptographic primitive in an appropriate manner.

**Hybrid Encryption**: The POST scheme recommends hybrid encryption [7], [8], where the encryption of the sensed data is done by a symmetric encryption and the symmetric key is encrypted by a asymmetric encryption. Hybrid encryption is also known as the KEM/DEM framework (KEM stands for key encapsulation mechanism, and DEM stands for data encapsulation mechanism). In the conventional hybrid encryption, first an ephemeral key $K$ and its encapsulation $C_1$ is computed such that $(K, C_1) \leftarrow \mathsf{Encaps}(pk)$, where $pk$ is a receiver's public key. Next, the actual data $M$ is encrypted such that $C_2 \leftarrow \mathsf{E}(K, M)$ by using the ephemeral secret key $K$, where $\mathsf{SKE} = (\mathsf{E}, \mathsf{D})$ is a symmetric key encryption scheme. The receiver can compute $M$ such that $K \leftarrow \mathsf{Decaps}(sk, C_1)$ and $M \leftarrow \mathsf{D}(K, C_2)$, where $sk$ is a receiver's secret key.

However, the POST scheme does not follow the conventional hybrid encryption, since $K$ is decided by the previous round's ephemeral key and all sensor's deployment information. So, we cannot provide $K$ as an output of the Encaps algorithm. Hence $K$ of the POST scheme cannot be used as an ephemeral key. So, $K$ must be regarded as a "plaintext" of the underlying public key system, and therefore the POST scheme requires at least one more encryption procedure (i.e, encryptions for both $K$ and the data itself) under the conventional hybrid encryption procedure[1]. In the WSN environment, the computation ability of sensor is limited. So, it is desirable to reduce the computational costs as small as possible. This public key encryption phase is the most costly part of a sensor in the POST scheme, and therefore we reconsider how to apply hybrid encryption in the POST scheme.

**Random-number-leakage problem**: In the encryption algorithm (or the Encaps algorithm also), a random number is chosen to guarantee the semantic security of the underlying encryption scheme. So, the POST scheme implicitly requires that each sensor has to choose a random number for public key encryption. However, how to choose a random number has not been clearly decided in POST scheme. Actually, since the adversary can capture sensors, and can obtain "all" internal states of captured sensors in the WSN usage, the seed of each sensor for generating random numbers is disclosed after a sensor is compromised. This means that the node-capture adversary can run several attacks, e.g., state compromise extension attacks [16] since random number generation is started in an guessable/detectable insecure state. Note that even the underlying public key cryptosystem is semantic secure (i.e., any information of the encrypted data is not exposed from the ciphertext), an adversary can easily extract the corresponding plaintext from the ciphertext when the adversary knows the random number which was used for public key encryption. Even if the underlying public key encryption is secure against chosen ciphertext attack (CCA), there is nothing to take away this fact.

Recently, leakage-resilient cryptography has been considered [10], [11], [12], where an adversary can obtain a "part" of the internal state (containing the random number) through the leakage function whose range is bounded to some bits (e.g., the Kiltz-Pietrzak public key scheme [11] is secure as long as the leakage is sufficiently smaller than $\log(p)/2$, where $p$ is the prime order of the underlying group). In the conventional public key encryption usage, leakage-resilient cryptography is an effective countermeasure against side-channel attacks (e.g., [13], [14], [15]). However, leakage-resilient cryptographic schemes cannot be applied for the WSN usage because a sensor has no tamper resistant hardware. That is, there is a vulnerability of the POST scheme to node-capture attack. So, we insist that this random-number-leakage problem is quite serious in the POST scheme.

### D. Our contribution

In this paper, we propose an elaborate methodology to improve the POST scheme from the viewpoint of both security and efficiency without spoiling significant benefit points of the original one. We focus on the fact that the POST scheme applies public key encryption as a "black box manner". Our methodology is totally different from the black box usage, where:

- We set deployed information as a seed used for generating a random number (which is applied for the Encaps algorithm), instead of the ephemeral key for generation in the POST scheme.

[1] That is, for the ephemeral key $K$ of the POST scheme (computed from deproyment information), the actual transfered ciphertext is $(C_1, C_2, C_3)$, where $(K', C_1) \leftarrow \mathsf{Encaps}(pk)$, $C_2 \leftarrow \mathsf{E}(K, M)$, and $C_3 \leftarrow \mathsf{E}(K', K)$. We insist that $C_3$ is redundant from the viewpoint of both encryption costs and channel capacity. Note that a ciphertext consists at least two group elements in probabilistic public key encryption (e.g., the ElGamal encryption). Although deterministic encryption (e.g., the conventional RSA scheme [9]) can be implimented with the one-group element ciphertext, such scheme never achieve semantic security. So, information of a plaintext (the encryption key $K$ in the POST scheme case) might be leaked from the ciphertext.

- The actual encryption of data is done by using the KEM key.

Namely, we pay attention to the internal state of the underlying hybrid encryption, and this kind of approach has never been considered before to the best of our knowledge. Briefly, for the ephemeral key $K$ of the POST scheme (computed from deployment information), we set $K$ as the random number (that is, $(K', C_1) \leftarrow \mathsf{Encaps}(pk; K)$, which stands for the Encaps algorithm is executed with a random number $K$), and the actual ciphertext is $(C_1, C_2)$ only, where $C_2 \leftarrow \mathsf{E}(K', M)$. Our methodology follows the conventional hybrid encryption usage, and therefore we can reduce both the encryption cost and the size of the ciphertext compared with the POST scheme. In addition, our improvement also captures the random-number-leakage problem by applying the crucial advantage of the POST scheme, where the adversary cannot expect each sensor's behavior. It is particularly worth nothing that:

- We improve the efficiency (both the encryption cost and the size of the ciphertext).
- We repair the vulnerability of the POST scheme to node-capture attack.
- We achieve the above results without detracting the benefit points of the POST scheme (i.e., intrusion resilience, forward secrecy, backward secrecy, and self healing property).

We summarize our results in Fig.1.

## II. Preliminaries

In this section, we define system model, and KEM/DEM framework.

### A. System Model

Here, by borrowing notations [6], we define the system model as follows. Let $\mathbf{N} = \{s_1, s_2, \ldots, s_N\}$ be the set of all sensors. At initial deployment, sensor $s_j$ position is $cp_j^0$. At round $r$, $s_j$ obtains data $d_r^j$. We assume that sensors have a common one-way hash function $H(\cdot)$ used as a pseudo-random number generator (PRNG), and assume that $s_j$ has a unique random secret seed $\mathcal{K}_j^r$ which is chosen by the sink and loaded onto each sensor upon each sink visit. Let $K_j^r$ be a random number which was used for public key encryption (Note that $K_j^r$ "was" a random key which was used for public key encryption of the actual data in the POST scheme).

Sensors are free to move over the deployment area according to a network-wide mobility model. As in the POST scheme, we consider two mobility models called random jump model and random waypoint model. We apply the functions $cp \leftarrow \mathsf{RANDOM\text{-}JUMP}(cp)$ and $cp \leftarrow \mathsf{RANDOM\text{-}WAYPOINT}(cp, wp, m)$ to decide the sensor position at the next round, where $wp$ is the waypoint (next destination) and $m$ is a distance (i.e., all sensors move with the same constant speed and can cover $m$ in a single round). These functions have been introduced in the POST scheme (in [6] algorithm 1 and 2, respectively). Since our target is the public key encryption phase of the POST scheme, we omit the detailed definitions of these functions, and we use the RANDOM-JUMP in our proposed scheme. Of course, we can also use the RANDOM-WAYPOINT function.

### B. Hybrid Encryption

Here, we define public key encryption with the KEM/DEM framework. Let $\mathsf{SKE} = (\mathsf{E}, \mathsf{D})$ be a symmetric key encryption scheme. The encryption algorithm $\mathsf{E}$ takes a secret key $K \in \{0,1\}^k$ and a plaintext $M$, and returns a ciphertext $C$. The decryption algorithm $\mathsf{D}$ takes a secret key $K \in \{0,1\}^k$ and a ciphertext $C$, and returns a plaintext $M$ or $\perp$. Let $\mathsf{KEM} = (\mathsf{KEM.KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ be a KEM scheme. The key generation algorithm KeyGen takes a security parameter $\lambda \in \mathbb{N}$, and returns a pair of public/secret key $(pk, sk)$. Note that $\lambda$ indicates the size of the underlying group (e.g., the bit length of a prime order). The encapsulation algorithm Encaps takes $pk$, and returns an ephemeral key $K \in \{0,1\}^k$ and its encapsulation $C$. When a random number is explicitly indicated, then we denote $\mathsf{Encaps}(pk; r)$ which stands for the Encaps algorithm is executed with a random number $r$[2]. According to our usage, we assume that $r \in \{0,1\}^k$. The decapsulation algorithm Decaps takes $sk$ and $C$, and returns $K$ or $\perp$. Next, we describe hybrid encryption as follows:

*Definition 1 (Hybrid Encryption):*

$\mathsf{KeyGen}(1^\lambda)$:
: Run $(pk, sk) \leftarrow \mathsf{KEM.KeyGen}(1^\lambda)$ of the underlying KEM scheme. Output $(pk, sk)$.

$\mathsf{Enc}(pk, M)$:
: Run $(C_1, K) \leftarrow \mathsf{Encaps}(pk)$ and $C_2 \leftarrow \mathsf{E}(K, M)$. Output $(C_1, C_2)$.

$\mathsf{Dec}(sk, C)$:
: Run $K \leftarrow \mathsf{Decaps}(sk, C_1)$ and $M \leftarrow \mathsf{D}(K, C_1)$. Output $M$. Note that if either Decaps or D outputs $\perp$ (i.e., either $C_1$ or $C_2$ is a invalid ciphertext), then output $\perp$.

As in the Encaps algorithm, we denote $\mathsf{Enc}(pk, M; r)$ when a random number $r \in \{0,1\}^k$ is explicitly indicated. Then, $\mathsf{Encaps}(pk; r)$ is called in the Enc algorithm.

If the random oracle model [17][3] is admitted, then we recommend the Abe-Kiltz-Okamoto KEM/DEM scheme [18], which can be implemented with the smallest costs to the best of our knowledge. However, if sensors can use such fully random function, then sensors do not have to compute

[2]Note that the notation $r$ is used as the round counter in our proposal and the POST scheme. Here, only for the explanation of the public key encryption, we use $r$ as a random number as a matter of principle.

[3]In the random oracle model, all parties (including the adversary) have oracle access to a fully random function.

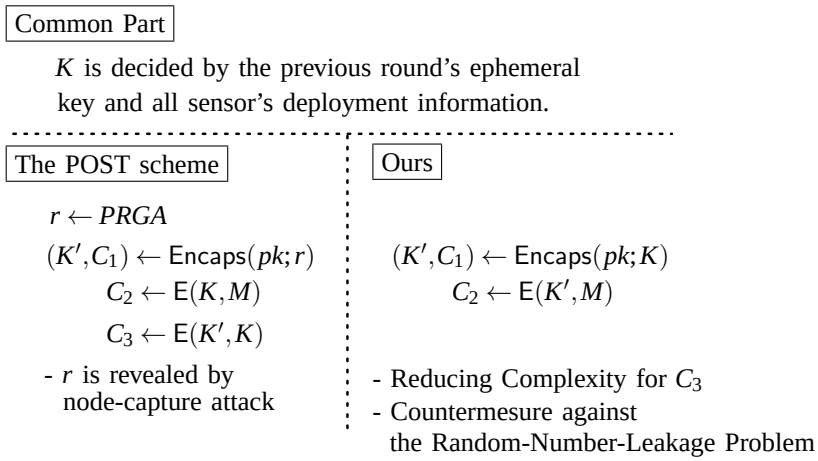


Figure 1. Our Contribution

its ephemeral key or random number based on the deployed information in the first place. In addition, it is desired to construct cryptographic schemes without random oracles. For example, Canetti et. al. [19] shows that there exist signature and encryption schemes, where are secure in the random oracle model, but are insecure when replaces random oracles with actual hash functions. So, we recommend the Kurosawa-Desmedt KEM/DEM scheme [7] which can be implemented with the highest efficiency (in the case without applying random oracles), to the best of our knowledge. An adversary cannot control the result of data collections since the Kurosawa-Desmedt KEM/DEM scheme is CCA secure.

If sensor ability is poor (and usually it is natural assumption in WSNs), then we recommend the ElGamal KEM/DEM scheme. Although it is secure against chosen plaintext attack (CPA), two scalar multiplications over a group is required for encryption. In this case, the sink must check the decryption result. For example, if the classical CPA secure ElGamal encryption scheme [20] is applied, then an adversary can pass off a wrong data to the sink as follows: Let $C_1$ and $C_1'$ be an encapsulation of the ephemeral key $K$ and $K'$, respectively, and $C_2$ be a ciphertext of the actual data under $K$. An adversary can obtain $(C_1, C_2, C_1')$ on the public channel. Then, the adversary can obtain a ciphertext (say $C_1''$) of $K \cdot K'$ from $C_1$ and $C_1'$ by applying the homomorphic property of the ElGamal encryption. If $(C_1'', C_2)$ is sent to the sink, then the sink obtains a wrong data such that $M' \leftarrow \mathsf{D}(K \cdot K', C_2)$. So, it is required that the sink can verify whether the decryption result is correct or not (i.e., format check, range check, and so on).

## III. The Proposed Scheme

In this section, we propose a modified POST scheme. Intuitively, our scheme is described as follows. From deployed information $(R_j^r[0], \ldots, R_j^r[c])$, a random number $K_j^r$ (which was used for public key encryption such that $\mathsf{Enc}(pk, d_j^r; K_j^r)$[4]) is generated, and the actual encryption of data is done by using the KEM key. Function $\mathsf{RandomNumberGeneration}(\cdot)$ is used the sensor's current secret state to generate a random number (it is corresponding to the KeyGeneration function in the POST scheme).

*Protocol 1 (The Proposed Scheme):* For sensor $s_j$ (with a secret seed $\mathcal{K}_j^r$) at round $r$,

1) Move $cp_j^r$ from $cp_j^{r-1}$ according to $cp_j^r \leftarrow \mathsf{RANDOM\text{-}JUMP}(cp_j^{r-1})$.
2) Pick a new value $t_j^r$, and broadcast $t_j^r$ to neighbors/peers such that $\mathsf{broadcast}(t_j^r)$.
3) Obtain data $d_r^j$ from the around.
4) Generate a new random value from secret state such that $K_j^r \leftarrow \mathsf{RandomNumberGeneration}(\mathcal{K}_j^r)$.
5) Encrypt $d_r^j$ (by using $K_j^r$ as the random number) such that $(C_1, C_2) \leftarrow \mathsf{Enc}(pk, d_j^r; K_j^r)$, and store current data $(C_1, C_2)$. Note that $C_1$ is the encapsulation of the KEM key, and $C_2$ is the ciphertext of the actual data $d_r^j$ under the KEM key. See definition1 for detailed notations.
6) Receive peer contributions such that

   **While** $(roundTimer)$ **do**
   a) Receive $t_r$ from $s_p$.
   b) Set $R_j^r[c] = t_p^r$.
   c) Increment $c = c + 1$.
7) Generate new secret state such that $\mathcal{K}_j^{r+1} \leftarrow H(\mathcal{K}_j^r || R_j^r[0] || \cdots || R_j^r[c])$.
8) Delete $K_j^r$ and $\mathcal{K}_j^r$.

## IV. Discussion

Since we just improve the public key encryption part (the 4th and 5th steps in the proposed scheme) of the POST

[4]As in the POST scheme, we emphasize that sensors do not have their own public/secret keys and do not perform any public key decryption.

scheme only (i.e., we do not touch the other part of the POST scheme), our modified POST scheme inherits crucial advantages of the original POST scheme. That is, our modified POST scheme satisfies forward secrecy and backward secrecy since the original POST scheme satisfies these. In addition, we reduce the computation and transformation cost of the ciphertext of the ephemeral secret key $K$[5]. In other words, we improve the efficiency (both the encryption cost and the size of the ciphertext), and repair the vulnerability of the POST scheme to node-capture attack "without detracting the benefit points of the POST scheme".

Next, for the sake of clarity, we briefly explain that our scheme satisfies forward secrecy and backward secrecy as follows. A secret state $\mathcal{K}_j^r$ is updated periodically, using the one-way hash function $H$. If we assume that time is divided in rounds and let $\mathcal{K}_j^0$ be an initial secret state, the secret state for round $r \geq 1$, say $\mathcal{K}_j^r$, is computed as $\mathcal{K}_j^r \leftarrow H(\mathcal{K}_j^{r-1}||\cdots)$. Hence, if adversary learns secret at $r$ round, it cannot compute any secrets used in prior rounds due to the one-wayness of $H$[6]. In addition, for updating a secret state, neighbor's contributions $R_j^r[0],\cdots||R_j^r[c]$ are used. When, since the contribution used is random in each round, no adversary can expect the contribution. Hence, if adversary learns secret at $r$ round, it cannot compute any secrets used in future rounds.

## V. Conclusion

In this paper, we improve the POST scheme from the viewpoint of both security and efficiency. Our methodology follows the conventional hybrid encryption usage, and therefore we can reduce both the encryption cost and the size of the ciphertext compared with the POST scheme. By applying the magnificent idea of the POST scheme (i.e., deployment information is set as a seed for PRGA) to choose a random number used in the underlying hybrid encryption scheme, we also repair the vulnerability of the POST scheme to node-capture attack. Our proposal is valuable to implement the POST scheme in the real WSNs environments.

## References


[1] Di Ma and Gene Tsudik. Dish: Distributed self-healing. In *SSS*, pp. 47–62, 2008.

[2] Roberto Di Pietro, Di Ma, Claudio Soriente, and Gene Tsudik. Posh: Proactive co-operative self-healing in unattended wireless sensor networks. In *SRDS*, pp. 185–194, 2008.

[3] Roberto Di Pietro, Luigi V. Mancini, Claudio Soriente, Angelo Spognardi, and Gene Tsudik. Catch me (if you can): Data survival in unattended sensor networks. In *PerCom*, pp. 185–194, 2008.


[5] We mentioned this ciphertext as $C_3 \leftarrow \mathsf{E}(K', K)$, and the size of $C_3$ is at least the size of $K$ (e.g., 128 bit).

[6] Although one-wayness does not mean semantic security, we practically use a hash function such as SHA1 or SHA2. Of course, we can theoretically use a hard core as a hash value since a hard core exists in an one-way hash function.


[4] Gerard D. Murphy, Emanuel M. Popovici, and William P. Marnane. Area-efficient processor for public-key cryptography in wireless sensor networks. In *SENSOR-COMM*, pp. 667–672, 2008.

[5] Ronghua Wang, Wenliang Du, Xiaogang Liu, and Peng Ning. Shortpk: A short-term public key scheme for broadcast authentication in sensor networks. *TOSN*, Vol. 6, No. 1, 2009.

[6] Roberto Di Pietro, Gabriele Oligeri, Claudio Soriente, and Gene Tsudik. Intrusion-resilience in mobile unattended WSNs. In *INFOCOM*, pp. 2303–2311, 2010.

[7] Yvo Desmedt, Rosario Gennaro, Kaoru Kurosawa, and Victor Shoup. A new and improved paradigm for hybrid encryption secure against chosen-ciphertext attack. *J. Cryptology*, Vol. 23, No. 1, pp. 91–120, 2010.

[8] Victor Shoup. Using hash functions as a hedge against chosen ciphertext attack. In *EUROCRYPT*, pp. 275–288, 2000.

[9] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, Vol. 21, No. 2, pp. 120–126, 1978.

[10] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pp. 293–302, 2008.

[11] Eike Kiltz and Krzysztof Pietrzak. Leakage resilient ElGamal encryption. In *ASIACRYPT*, pp. 595–612, 2010.

[12] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In *ACM Conference on Computer and Communications Security*, pp. 141–151, 2010.

[13] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In *EUROCRYPT*, pp. 37–51, 1997.

[14] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *CHES*, No. Generators, pp. 251–261, 2001.

[15] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO*, pp. 104–113, 1996.

[16] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Cryptanalytic attacks on pseudorandom number generators. In *FSE*, pp. 168–188, 1998.

[17] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pp. 62–73, 1993.

[18] Masayuki Abe, Eike Kiltz, and Tatsuaki Okamoto. Compact CCA-secure encryption for messages of arbitrary length. In *Public Key Cryptography*, pp. 377–392, 2009.

[19] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, Vol. 51, No. 4, pp. 557–594, 2004.

[20] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, Vol. 31, No. 4, pp. 469–472, 1985.

[21] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact E-cash. In *EUROCRYPT*, pp. 302–321, 2005.

[22] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In *VIETCRYPT*, pp. 193–210, 2006.

[23] Giuseppe Ateniese, Jan Camenisch, and Breno de Medeiros. Untraceable RFID tags via insubvertible encryption. In *ACM Conference on Computer and Communications Security*, pp. 92–101, 2005.

[24] Man Ho Au, Qianhong Wu, Willy Susilo, and Yi Mu. Compact E-cash from bounded accumulator. In *CT-RSA*, pp. 178–195, 2007.

[25] Atsuko Miyaji, Masaki Nakabayashi, and Shunzo Takano. Characterization of elliptic curve traces under FR-reduction. In *ICISC*, pp. 90–108, 2000.

[26] Gerard D. Murphy, Emanuel M. Popovici, and William P. Marnane. Area-efficient processor for public-key cryptography in wireless sensor networks. In *SENSORCOMM*, pp. 667–672, 2008.

## APPENDIX

Here, we introduce the ElGamal KEM/DEM scheme as follows. Let $\mathbb{G}$ be a cyclic group with prime order $p$ and we assume that the decisional Diffie-Hellman (DDH) problem is infeasible over $\mathbb{G}$. If $\mathbb{G}$ is a subgroup of $\mathbb{Z}_q$ with $p \mid q-1$, then we need to require 1024-bit prime $q$. If elliptic curve is used, then we just require 170-bit prime $q$. Note that a bilinear map (i.e., pairing) can be seen as the DDH oracle, and therefore we need to require the external Diffie-Hellman (XDH) assumption [21], [22] or symmetric XDH (SXDH) [23], [24], where the DDH problem over $\mathbb{G}$ is hard even the pairing is implemented. The XDH assumption hold in certain subgroups of MNT elliptic curves [25]. Note that a versatile public-key cryptographic processor suitable for WSNs has been considered [26].

*Protocol 2 (The ElGamal KEM/DEM scheme):*

- $\mathsf{KeyGen}(1^\lambda)$: Choose a generator $g \in \mathbb{G}$ and $x \in \mathbb{Z}_p$ ramdomly, compute $h = g^x$, and output $pk = (g,h)$ and $sk = x$.
- $\mathsf{Enc}(pk, M)$: Choose $r \in \mathbb{Z}_p$ ramdomly, and compute $u = g^r$, $K = \mathsf{KDF}(h^r)$, and $e = \mathsf{E}(K,M)$, and output $C = (u,e)$.
- $\mathsf{Dec}(sk; C)$: Compute $K = \mathsf{KDF}(u^x)$ and $M = \mathsf{D}(K,e)$, and output $M$.

The ElGamal KEM/DEM scheme is CPA secure if all followings hold; (1) the DDH assumption holds over $\mathbb{G}$, (2) $\mathsf{SKE}$ is a one-time semantically secure symmetric encryption scheme, and (3) $\mathsf{KDF}$ is a secure key derivation function.

Obviously, if the random number $r$ used in the $\mathsf{Enc}$ algorithm is revealed, then $K$ is also revealed. In our usage, the encryption algorithm is executed such that $\mathsf{Enc}(pk, M; K)$. Then, instead of choosing a random number internally, $K$ is indicated in the outside of the algorithm as the random number. So, a ciphertext is described as $(g^K, e = \mathsf{E}(\mathsf{KDF}(h^K), M))$ only. This is our main methodology to improve the POST scheme.