Secure Mobile Subscription of Sensor-Encrypted Data

Cheng-Kang Chu
Cryptography and Security Department
Institute for Infocomm Research
1 Fusionopolis Way
Singapore 138632
ckchu@i2r.a-star.edu.sg

Wen Tao Zhu
State Key Laboratory of Information Security
Graduate University of Chinese Academy of Sciences
19A Yuquan Road
Beijing 100049, China
wtzhu@ieee.org

Sherman S. M. Chow
Department of Combinatorics and Optimization
University of Waterloo
200 University Avenue West
Waterloo, Ontario
Canada N2L 3G1
smchow@uwaterloo.ca

Jianying Zhou
Cryptography and Security Department
Institute for Infocomm Research
1 Fusionopolis Way
Singapore 138632
jyzhou@i2r.a-star.edu.sg

Robert H. Deng
School of Information Systems
Singapore Management University
80 Stamford Road
Singapore 178902
robertdeng@smu.edu.sg

ABSTRACT

In an end-to-end encryption model for a wireless sensor network (WSN), the network control center preloads encryption and decryption keys to the sensor nodes and the subscribers respectively, such that a subscriber can use a mobile device in the deployment field to decrypt the sensed data encrypted by the more resource-constrained sensor nodes. This paper proposes `SMS-SED`, a provably secure yet practically efficient key assignment system featuring a discrete time-based access control, to better support a business model where the sensors deployer rents the WSN to customers who desires a higher flexibility beyond subscribing to strictly consecutive periods. In `SMS-SED`, a node or a mobile device stores a secret key of size independent of the total number of sensor nodes and time periods. We evaluated the feasibility of deploying 2000 nodes for 4096 time periods at 1024-bit of security as a case study, studied the trade off of increasing the storage requirement of a node to significantly reduce its computation time, and provided formal security argument in the random oracle model.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; E.3 [**Data Encryption**]; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

ASIACCS '11, March 22–24, 2011, Hong Kong, China.

General Terms

Security, Algorithms

Keywords

sensor network security, subscription-based key management, compact key size, data confidentiality, access control, weak computational device

1. INTRODUCTION

A wireless sensor network (WSN for short) is a large-scale, self-organized network consisting of a number of low-cost, resource-constrained sensor nodes, which monitors ambient environments in a certain deployment field. By means of ad hoc routing, the sensor nodes send wirelessly the obtained data (typically the sensor readings) to a base station, which may further process the data and forward the result to a control center (e.g., for decision making). In certain scenarios, the base station and the control center may be implemented in the same physical server, which is usually a powerful control center assumed to take over operating management including user (i.e., customer) authentication.

1.1 Motivation

We envision a potential business model where the network provider deploys the sensor nodes and rents the WSN to commercial customers. The network as a whole is provided as a certain kind of infrastructure service (e.g., for scientific research purposes), and each user just subscribes to the service for data acquisition without necessarily deploying his or her own sensor nodes. A sensor node in general can be multi-functional, and thus there can be multiple types of data services (e.g., temperature, pressure, and humidity reports) and accordingly, various types of subscriptions regarding all possible combinations (e.g., a user may subscribe to temperature and humidity but no pressure reports). For simplicity, we focus on the consideration of just one such data service (and thus only one type of subscription), while our

developed solution can be directly extended to accommodate a scalable, multi-service scenario, and be further adapted to other data provision applications such as self-navigation for vehicles, as well as military applications [19]. Data are collected by mobile units (e.g., unmanned aerial units) that access the sensor network at locations which may be "hard" to predict.

For the benefit of the network provider, a commercially viable data provision service needs a certain kind of access control so that any user can only acquire the entitled data according to the subscription. Such a subscription is typically managed on a time basis, e.g., a user is charged daily, or pays per hour. This subscription-based payoff model has been well supported by economic research results on owner-side strategies for maximizing profits of information goods. For example, research at NYU and MIT concluded that content bundling and fixed fees can generate greater profits per good [2]. This work is partially motivated by the understanding that a business model based on subscription is more economically beneficial than others.

1.2 Sensor Network Data Encryption Models

To ensure data confidentiality, sensor nodes should encrypt the obtained data before sending them over the air. To realize this goal efficiently, sensor networks adopt symmetric-key cryptography, where the same key is employed for both encryption (i.e., data protection) and decryption (i.e., data access). For a WSN, the encryption can be done in either a hop-by-hop manner, or an end-to-end one.

- In the hop-by-hop encryption model, a sensor node locally shares a pairwise key with (almost) each of its neighbor nodes within its communication range, so that a neighbor node can decrypt (and possibly aggregate) the received data before re-encrypting and forwarding the data to the next hop (i.e., the neighbor node's neighbor on the path towards the base station).

- On the other hand, in the end-to-end encryption model, a sensor node only holds one encryption key (no matter how many neighbors it has) secretly shared with the base station; the encrypted data can be decrypted by the base station but not the other nodes, while the neighbor nodes just propagate the data that are kept intact in the encrypted form.

One advantage of the end-to-end encryption model is that the sensed data can keep confidential even if some sensor nodes are untrustworthy (e.g., corrupted). Another advantage is that the data can be locally accessed by a mobile user authorized by the control center. That is, the user can roam in the WSN deployment field and directly access the localized data at any sensor node, as long as the user holds respective decryption keys (i.e., the same with those employed by the base station). In this article, we are interested in such an end-to-end encryption model.

1.3 Time-based Access Control

Apart from who can access the data, another dimension to consider is when can the data be accessed. Time-based access control can be incarnated by time-bound key assignment (e.g., [23, 7]). Each sensor node encrypts the obtained data with a time-variant session key. Different nodes (identified by their node indices ß $= \{1, 2, \cdots, n\}$) employ different session keys even in the same time period. Note that we assume sensor nodes are (at least loosely) synchronized with a secure time synchronization scheme [20].

On the other hand, a user registers at the WSN control center to subscribe to the data service for a set J of subscription time periods. At any (discrete) time $j \in J$, any user who "subscribes to J" can access the sensed data directly from the sensor nodes. In a nutshell, a sensor node encrypts the data with a session key $k_{i,j}$, which is determined by the node index i and the current period of time j, and a mobile user in the WSN deployment field can access the localized data with the same session key $k_{i,j}$, as long as j is within his or her subscription set J.

The above idea can be realized trivially if the control center creates $n \cdot l$ keys, pre-loads l keys to each node and assigns $n \cdot |J|$ keys to each user, where l is the total number of time periods and $|J|$ is the number of time periods subscribed by the user. Apart from the key management issues of a large number of keys, there is a cost issue since secret key is preferably stored in secure storage which is considerably more expensive than normal storage. Having a bulky (collection of) key is probably not a good idea for resource-constrained mobile devices and even more constrained sensor nodes.

Preferably, we want a "two-dimensional" time-based key management scheme which works as follows. The control center issues a node key $k_{i,*}$ to the node i, and issues a user key $k_{*,J}$ to a user who subscribed to the set of time periods J, as depicted in Fig. 1(a). A time-variant session key $k_{i,j}$ can either be computed by the node using a node key $k_{i,*}$ according to the current discrete time index j, or by a mobile user using a user key $k_{*,J}$ if $j \in J$, as depicted in Fig. 1(b). For example, in the simplest case, a mobile user only subscribes to a single time period j (i.e., $|J| = 1$) is assigned with the user key $k_{*,j}$, from which $k_{i,j}$ for any node i can be derived.

Now we have a conceptual idea which just needs to utilize a constant size of cryptographic secret keying material. A practical solution following this framework should allow the key derivations to be done efficiently. In particular, the computation required in deriving a session key should be independent of the total number of nodes.

1.4 Our Contribution

1.4.1 Generalized Time-Based Access Control

Without loss of generality, we assume the entire subscription time is partitioned into l equal units referred to as time periods, where each period can typically be a day or an hour. Our idea of time-based access control system features that any user is allowed to subscribe to an *arbitrary* set $J \subseteq \{1, 2, \cdots, l\}$ of these time periods, where the $|J|$ periods can be either consecutive or intermittent. Such flexibility makes our mechanism distinguished from an existent technique known as time-bound access control [23, 7], where the $|J|$ time periods only start at a certain j and end at $(j + |J| - 1)$, i.e., the time periods are strictly consecutive, which may be a limit for many real-world applications.

While our motivating applications is providing subscription services of data encrypted by WSN to mobile users, our system provides a useful and lightweight primitive for time-based access control in general. Moreover, the idea of discrete time periods generalizes to classifiers of different kinds

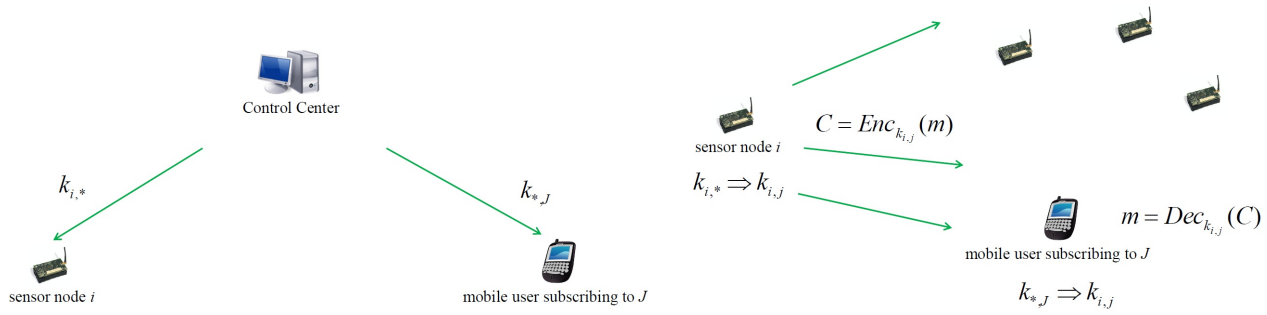

Figure 1: Overview of the proposed time-based key management scheme for data access control in a WSN – $k_{i,j}$ is the symmetric session key employed by sensor node i at time j for encryption and also employed by the mobile user for decryption, that is either derived from $k_{i,*}$ by the node i, where $k_{i,*}$ is the node key initially issued by the control center; or derived from his $k_{*,J}$, where $k_{*,J}$ is the user key also issued by the control center and $j \in J$.

of data, and our system thus naturally leads to application or domain specific subscription of data.

1.4.2 Provable Security Guarantee

A major challenge of the posed key management problem lies in the collusion of corrupted sensor nodes and (legitimate but malicious) mobile users. Since sensor nodes are not made tamper-proof, an adversary can easily reveal the pre-loaded node key from a captured node. On the other hand, by subscribing to the data service, the adversary can also play the role of a mobile user, and thus acquire user keys for certain time periods at his will. To thwart any possible collusion, we pursue a secure solution where the adversary is prevented from accessing localized data at any uncompromised sensor node at any time period beyond his subscription.

We will define the above intuition of security requirement with respect to a threshold t, which models the adversary's ability to capture the sensor nodes. As long as no more than t sensor nodes are compromised, the whole system still remains secure. We remark that the threshold t only bounds the number of revealed node keys, but not the number of acquired user keys. Actually, an adversary is allowed to obtain by means of subscription as many user keys as he would like. Even so, our time-based scheme can still achieve provable security. We believe this is important for a security-oriented protocol, especially when previous time-bound key management schemes [23, 7] have been found vulnerable to collusion attacks [26, 25].

1.4.3 Our Design Principles

Our scheme can be seen as borrowing the idea of the broadcast key assignment protocol proposed by Benaloh and de Mare [5]. We applied two design principles in our construction. One of the first principles in Computer Engineering is to make the common case fast. In our context, we want to make the computation at the node, i.e., the node key derivation, as lightweight as possible. Looking ahead, our system (and [5]) assigns different prime exponents for different "atomic units" of the access control policy (time period in our case) and creates a session key by doing repeated exponentiations. One may consider further adapting this idea also on the node identifiers, similar to the treatment of time periods in our algorithm. However, this will result in more exponentiation in the node key derivation (those exponentiations regarding the n primes associated with the n nodes), which is undesirable. Our solution can be seen as making the "common" case fast by shifting the burden of node to that of the mobile user. A novelty in our system is that, the user key derivation is computed from an interpolation of t elements, which means the "apparent" shifted burden is *not* dependent on the number of nodes as one may imagine. This leads to our second principle. We get a higher efficiency for the overall system since our system `SMS-SED` provided a "reasonable" level of security – resilience against a threshold t of nodes where t is a system parameter. On the other hand, an adoption of Benaloh and de Mare's idea in our scenario can possibly provide a very high level of security which makes the system remains secure even when all but one of the sensor nodes of the whole WSN are compromised.

1.4.4 Symmetric Key Aggregation

Our system `SMS-SED` can also be seen as providing a mechanism to aggregate many symmetric keys together. The size of either a node key $k_{i,*}$ or a user key $k_{*,J}$ are independent of either the total number of sensor nodes n or the total number of time periods l, yet each of them can be used to derive keys for all l time periods or keys used by all n nodes respectively. Using symmetric keys, only a little overhead is required to simultaneously obtain confidentiality and authenticity (integrity) by using a block cipher mode of operation which supports authenticated encryption (e.g., [4].) While there exist public-key signature schemes which feature a very short signature size and signature transmission would not required too much energy from the transceivers, its verification may be time-consuming when computation power is limited, unless extra measures are taken (for example, outsourced yet verifiable computation, e.g., [8]).

1.5 Paper Organization

Related work will be discussed in the next section. Section 3 presents some cryptography basics. Section 4 formally presents the notion of subscription-based key management scheme for discrete time periods, followed by the description of SMS-SED, our proposed key management system in Section 5. Section 6 evaluates the feasibility of the scheme with respect to resource-constrained sensor nodes. Finally, we conclude in Section 7. For readers' convenience, the symbols and parameters employed throughout this article are summarized in an alphabetical order in Table 1.

2. RELATED WORK

2.1 Mechanisms for Key Distribution

Establishing pairwise keys is always an important issue for WSN. Eschenauer and Gligor [13] first proposed a key pre-distribution scheme for WSN. In their scheme, each sensor node randomly picks a key set from a big key pool. If two nodes have a key in the intersection of their key sets, they can use this key as a secret key. Chan *et al.* [6] then proposed a scheme using the same procedure [13] but the key between two nodes is available if and only if the intersection of the key sets contains a number of keys. Otherwise, they have to communicate to each other via intermediate node(s). Subsequently, there are many key pre-distribution schemes proposed, e.g., [11, 16, 12, 15, 18]. However, these schemes are not suitable for our scenario because the keys are established according to the data receivers rather than time periods. In our scenario, the user is issued a key for (continuous or discrete) time periods and is restricted to access the nodes within the time periods.

2.2 Key Assignments Supporting Hierarchy

Time-bound access control is usually realized with key assignment scheme supporting a partial order hierarchy. A comprehensive overview of the time-bound hierarchical key assignment can be found in [27]. While one may extend the time-hierarchy to also cover the nodes (e.g., the upper level represents time periods and the lower level represents node indices), it is unclear how to support the two-dimensional key derivation as we described in Section 1.3. Even though the structure of a public-key scheme may be rich enough to support "fancy" hierarchy (e.g., a two-dimensional hierarchy [24]), these encryption schemes are impractical for low-cost, resource-constrained sensor nodes, unless other arrangements are made such as storing pre-computed results and utilizing online/offline encryption techniques (e.g. [9]).

One may view that our system is essentially providing a mean for a node i and a user subscribed to j to agree upon a session key $k_{i,j}$, which bears a similarity with the key agreement problem. First, it is "agreed" in a non-interactive or "offline" manner between a sensor node and a mobile user, without interacting with each other or interacting with the control center other than the initial assignments. While non-interactive key agreement exists, we still need to satisfy our design goal requiring the size of either $k_{i,*}$ or $k_{*,J}$ to be independent of either the total number of sensor nodes n or the total number of time periods l, which rules out the straightforward adoption of key agreement protocol.

3. TECHNICAL PRELIMINARIES

3.1 Quick Review of RSA Cryptosystem

An RSA cryptosystem can be setup in the following way. Choose two large primes p and q. Let $N = pq$ be the public RSA modulus and $\lambda(N) = \text{lcm}(p-1, q-1)$, where $\text{lcm}(\cdot,\cdot)$ denotes the least common multiple, be the Carmichael function (which is a secret similar to the factorization of N)[§]. Find integers $e > 1$ and d such that $ed \equiv 1 \pmod{\lambda(N)}$ (this implicitly requires $e, d \in Z^*_{\lambda(N)}$; therefore, $e \geq 3$). Publish e as the encryption exponent, and keep d as the (secret) decryption exponent. For any integer $x \in \mathbb{Z}_N$, its encrypted form is $y = x^e \bmod N$. With the knowledge of d, the plaintext x can be recovered from the ciphertext y following $x = y^d \bmod N = x^{ed} \bmod N = x^{z\lambda(N)+1} \bmod N = x$, where z is an integer.

Generally, given the public-key (e, N) and the ciphertext y, it is computationally infeasible for an adversary to recover the plaintext x. Nevertheless, given e_1, e_2, $y_1 = x^{e_1} \bmod N$ and $y_2 = x^{e_2} \bmod N$, where the greatest common divisor of the exponents is 1, i.e., $\gcd(e_1, e_2) = 1$, it is feasible for an adversary to recover the plaintext x. This is done as follows. Employ the Euclidean algorithm to compute integers u and v such that $e_1u + e_2v = 1$. Then $y_1^u y_2^v \equiv x^{e_1 u} x^{e_2 v} \equiv x \bmod N$. This is known as the common modulus attack, which we will employ in our security proof.

3.2 Quadratic Residue

We call an integer $u \in \mathbb{Z}^*_N$ a *quadratic residue* modulo N if there exists an integer v such that $u = v^2 \bmod N$ (otherwise, u is called a quadratic non-residue modulo N). We define the group of quadratic residues in $\mathbb{Z}^*_N$ as $\mathbb{QR}_N = \{u \in \mathbb{Z}^*_N \mid u = v^2 \bmod N, v \in \mathbb{Z}_N\}$. It is sufficient to generate $\mathbb{QR}_N$ by only considering $v \in \mathbb{Z}^*_N \cap \mathbb{Z}_{\lfloor \frac{N}{2} \rfloor + 1}$.

We say a prime p is a *safe prime* if $p = 2p' + 1$, where p' itself is a (large) prime (hence we immediately have $p \equiv 3 \pmod 4$). We say an N is a safe RSA modulus if $N = pq$ is the product of two distinct safe primes (hence N is a Blum integer). For such an N, the order of the group of quadratic residues is $|\mathbb{QR}_N| = \frac{(p-1)(q-1)}{4} = p'q' = \frac{\lambda(N)}{2}$. That is, there are $p'q'$ such quadratic residues in $\mathbb{Z}^*_N$.

It has been proved that, $u \in \mathbb{QR}_N$ is a generator for $\mathbb{QR}_N$, if and only if $\gcd(u-1, N) = 1$ [17]. Therefore, $\mathbb{QR}_N$ is a cyclic group of order $p'q'$. If p and q are significantly large, any random $u \in \mathbb{QR}_N \backslash \{1\}$ shall be a generator for $\mathbb{QR}_N$ except for a negligible probability $\varepsilon = \frac{p'+q'-2}{p'q'-1}$. One approach to compute this ε is to employ the group theory (regarding subgroup); another approach is to enumerate $u \in \mathbb{QR}_N \backslash \{1\}$ satisfying $\gcd(u-1, N) \neq 1$ (particularly, with the Chinese Remainder Theorem).

3.3 Strong QR-RSA Assumption

The strong RSA assumption is first introduced by [3]. Its variant and itself have been very useful in the construction of many efficient cryptographic functions (e.g. signature scheme in [10]). In this article, we consider a variant of this (standard) strong RSA problem, the strong QR-RSA

[§]We follow PKCS#1 version 2.1, which specifies using the Carmichael function $\lambda(N)$ instead of the Euler's totient function $\phi(N) = (p-1)(q-1)$. Observe that $\lambda(N)$ is always a divisor of $\phi(N)$. PKCS#1 version 2.1 is the public-key cryptography standard published by RSA laboratories in 2002 which was also republished as an Internet standard (RFC 3447).

problem [17], in the group of quadratic residues in $\mathbb{Z}_N^*$, where N is the product of two safe primes. It has been shown that this variant is not any easier than the standard one [10]. Therefore, we have the following strong QR-RSA assumption.

Definition 1. The strong QR-RSA assumption holds if no polynomial time adversary $\mathcal{A}$ has non-negligible advantage in solving the following strong QR-RSA problem: given the instance (y, N), where N is a safe RSA modulus and $y \in \mathbb{QR}_N$, output (x, e) such that $y = x^e \bmod N$.

3.4 Lagrange Interpolation

A polynomial $f(x)$ of degree t can be uniquely recovered from $(t+1)$ sample values $f(x_0), f(x_1), \cdots, f(x_t)$, where no two x_a are the same, $0 \le a \le t$. The interpolation polynomial in the Lagrange form is

$$f(x) = \sum_{a=0}^{t} \lambda_{x x_a} f(x_a), \text{ where } \lambda_{x x_a} = \prod_{b=0, b\neq a}^{t} \frac{x - x_b}{x_a - x_b}.$$

Each $\lambda_{x x_a}$ itself is also a polynomial of degree t (instead of a coefficient). Later in Section 5.1 we shall employ a special case where $x_a = a, 0 \le a \le t$. Thus we know for any (integer) i, $\lambda_{ia} = \prod_{b=0,b\neq a}^{t} \frac{i-b}{a-b}$ and then $f(i) = \sum_{a=0}^{t} \lambda_{ia} f(a)$. Let $\alpha_{ia} = \prod_{b=0,b\neq a}^{t} (i-b)$ and $\beta_a = \prod_{b=0,b\neq a}^{t} (a-b)$. Then we have $\lambda_{ia} = \frac{\alpha_{ia}}{\beta_a}$ for any (integer) i. In this article, we call such λ_{ia}'s the Lagrange interpolation coefficients.

The Lagrange interpolation also works in $\mathbb{Z}_m$, where $m = p'q'$ is the product of two (large) primes [22]. That is, for any subset of t points in $\{1, 2, \cdots, n\}$, the sample values of $f(x) \bmod m$ at these points uniquely determine the value of $f(x) \bmod m$ at any other point in $\{1, 2, \cdots, n\}$. (This follows from the fact that the corresponding Vandermonde matrix is invertible modulo m, since its determinant is relatively prime to m [22].) We will apply this idea in Section 5.2.

4. FORMAL DEFINITIONS

4.1 Framework

A time-based key management scheme consists of the following five algorithms:

- **Setup**$(1^\mu, n, t, l)$: On input an unary string 1^μ for an integer μ which acts as the security parameter, an integer n as the total number of sensor nodes, an integer t as the maximum number of nodes that the adversary may corrupt and an integer l as the total number of time periods for user subscription, output the master secret key K (along with certain public parameters). This is the only probabilistic one among all five algorithms.

- **NodeKeyGen**(K, i): On input the master secret key K and a node index i, output the secret key $k_{i,*}$ for node i.

- **UserKeyGen**(K, J): On input the master secret key K and a set J of time periods, output the secret key $k_{*,J}$ for a mobile user subscribing to all the time periods $j \in J$.

- **NodeKeyDer**$(k_{i,*}, j)$: On input a secret key $k_{i,*}$ and a time period j, output the encryption key $k_{i,j}$ for node i.

- **UserKeyDer**$(k_{*,J}, i, j)$: On input a secret key $k_{*,J}$ of a set J of time periods, output the decryption key $k_{i,j}$ for the user subscribing to J.

The control center performs **Setup** to select the master secret key, and then issues the node keys and user keys to sensor nodes and mobile users by invoking **NodeKeyGen** and **UserKeyGen**, respectively. After that, a sensor node performs **NodeKeyDer** to derive its encryption key for the current time period j, and an entitled mobile user performs **UserKeyDer** at time period $j \in J$ with respect to a corresponding node to derive the same key for decryption. That is, **NodeKeyDer** and **UserKeyDer** independently generate the same $k_{i,j}$ with regard to any given node-time index pair (i, j) for $i \in \{1, 2, \cdots, n\}$, $j \in J \subseteq \{1, 2, \cdots, l\}$.

4.2 Security Model

The security of a (t, n)-threshold time-based key management scheme is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Setup. $\mathcal{C}$ invokes **Setup**$(1^\mu, n, t, l)$ to select the master secret key K.

Query Phase 1. $\mathcal{A}$ can query the following oracles:

- NodeExt(i): $\mathcal{C}$ responds with

  $$k_{i,*} \leftarrow \textbf{NodeKeyGen}(K, i).$$

  This oracle can be only queried for up to t different i's.

- MobiExt(J): $\mathcal{C}$ responds with

  $$k_{*,J} \leftarrow \textbf{UserKeyGen}(K, J).$$

Challenge. $\mathcal{A}$ selects $(\hat{i}, \hat{j})$, where $\hat{i}$ has never been called in NodeExt and any J with $\hat{j} \in J$ has never been called in MobiExt. Then $\mathcal{C}$ flips a fair coin $b \in \{0, 1\}$. If $b = 0$, $\mathcal{C}$ responds with a random element chosen from the key space. If $b = 1$, $\mathcal{C}$ responds with the real session key $k_{\hat{i},\hat{j}} \leftarrow \textbf{NodeKeyDer}(k_{\hat{i},*}, \hat{j})$.

Query Phase 2. $\mathcal{C}$ responds to the queries from $\mathcal{A}$ as in Query Phase 1, but neither NodeExt$(\hat{i})$ nor any MobiExt(J) with $\hat{j} \in J$ is permitted; otherwise, $\mathcal{A}$ trivially obtains $k_{\hat{i},\hat{j}}$.

Guess. $\mathcal{A}$ outputs his guess b' on b. If $b' = b$, $\mathcal{A}$ wins the game.

The above game has modeled the potential attacks on the scheme: the adversary can corrupt at most t nodes and extract their node keys (via NodeExt), and acquire an arbitrary number of user keys (via MobiExt). Therefore, if a scheme is provably secure in the above model, it can resist collusion attacks.

Definition 2. A (t, n)-threshold time-based key management scheme is secure if for any polynomial time algorithm $\mathcal{A}$, $\mathcal{A}$'s advantage in the above game $Adv_{\mathcal{A}} = |\Pr[b' = b] - \frac{1}{2}|$ is negligible.

For this work, we consider the slightly restricted "static" security of a (t, n)-threshold time-based key management scheme, where $\mathcal{A}$ chooses a set of at most t nodes he may corrupt at the very beginning of the game. That is, in Query Phase 1 and Query Phase 2, NODEEXT is only made to these pre-determined nodes. Such a static model is the actual one we shall employ in the security proof of the proposed scheme.

5. OUR PROPOSED SYSTEM

5.1 Protocol Specification

Let n be the total number of sensor nodes in the WSN, t be the maximum number of nodes that the adversary may corrupt, and l be the total number of time periods for user subscription. The symbols and parameters employed are summarized in an alphabetical order in Table 1.

- **Setup**$(1^\mu, n, t, l)$:
  1. Choose two distinct primes p' and q' of length μ, so that $p = 2p' + 1$ and $q = 2q' + 1$ are two large safe primes.
  2. Let $e_1, e_2, \cdots, e_l$ be the enumeration of the l primes after n (i.e., $e_l > \cdots > e_2 > e_1 > n$). (Note that μ is large enough so that even e_l is still far less than both p' and q'.)
  3. Let $N = pq$ and $m = p'q'(= \frac{\lambda(N)}{2})$.
  4. Let $H(\cdot) : \mathbb{QR}_N \to \mathcal{K}$ be a cryptographic one-way hash function, where $\mathcal{K}$ is the key space associated with the symmetric encryption to be employed for protecting the sensed data.
  5. Let F be the factorial of n, i.e., $F = n!$.
  6. Randomly choose a secret $s \in \mathbb{QR}_N \backslash \{1\}$, and a polynomial $f(x)$ of degree t in $\mathbb{Z}_m$.
  7. Publish $(N, H(\cdot), F)$ as the public system parameters and securely store $K = (s, f(x))$ as the master secret key .
  8. For speeding up **NodeKeyGen** and **UserKeyGen**, s^{F^2}, and $s^{\frac{f(0)F^2}{\beta_0}}, \cdots, s^{\frac{f(t)F^2}{\beta_t}}$ can be precomputed, where $\beta_a = \prod_{b=0, b\neq a}^{t}(a-b)$.
- **NodeKeyGen**(K, i): For $K = (s, f(x))$, output
$$k_{i,*} = s^{f(i)F^2} \bmod N.$$
- **UserKeyGen**(K, J): For $K = (s, f(x))$, output
$$k_{*,J} = (s^{f(0)E_J\frac{F^2}{\beta_0}}, s^{f(1)E_J\frac{F^2}{\beta_1}}, \cdots, s^{f(t)E_J\frac{F^2}{\beta_t}}) \bmod N,$$
where $E_J = \prod_{z=1, z\notin J}^{l} e_z$, and $\beta_a = \prod_{b=0,b\neq a}^{t}(a-b)$ which is always a factor of F^2.¶
- **NodeKeyDer**$(k_{i,*}, j)$: Output
$$k_{i,j} = H(k_{i,*}^{E_j} \bmod N), \text{ where } E_j = \prod_{z=1, z\neq j}^{l} e_z.$$
- **UserKeyDer**$(k_{*,J}, i, j)$: Let $k_{*,J} = (k_0, k_1, \cdots, k_t)$. For $j \in J$, output
$$k_{i,j} = H((k_0^{\alpha_{i0}} k_1^{\alpha_{i1}} \cdots k_t^{\alpha_{it}})^{\prod_{z\in J\backslash\{j\}} e_z} \bmod N),$$
where $\alpha_{ia} = \prod_{b=0,b\neq a}^{t}(i-b)$ for $a = 0, 1, \cdots, t$. As discussed, these α_{ia}'s are the numerators of the Lagrange interpolation coefficients involved in computing $f(i) = \sum_{a=0}^{t} \lambda_{ia} f(a)$.

Correctness.
Let $\lambda_{ia} = \frac{\alpha_{ia}}{\beta_a} = \prod_{b=0,b\neq a}^{t} \frac{i-b}{a-b}$, be the Lagrange interpolation coefficients. For $j \in J$,

$$\begin{aligned}
& k_{i,*}^{E_j} \bmod N \\
= \ & (s^{f(i)F^2})^{E_J \prod_{z\in J\backslash\{j\}} e_z} \bmod N \\
= \ & s^{(\lambda_{i0}f(0)+\cdots+\lambda_{it}f(t))E_J F^2 \prod_{z\in J\backslash\{j\}} e_z} \bmod N \\
= \ & s^{(\frac{\alpha_{i0}}{\beta_0}f(0)E_J F^2+\cdots+\frac{\alpha_{it}}{\beta_t}f(t)E_J F^2)\prod_{z\in J\backslash\{j\}} e_z} \bmod N \\
= \ & (k_0^{\alpha_{i0}} k_1^{\alpha_{i1}} \cdots k_t^{\alpha_{it}})^{\prod_{z\in J\backslash\{j\}} e_z} \bmod N
\end{aligned}$$

That is, the encryption key derived by the node i is the same as the decryption key derived by the mobile user at time period $j \in J$.

5.2 Security Proof

We claim the static security (as defined in Section 4.2) of SMS-SED with the following theorem.

THEOREM 1. *SMS-SED is a secure (t, n)-threshold time-based key management scheme when $H(\cdot)$ is modeled as a random oracle, and under the strong QR-RSA assumption.*

PROOF. Suppose there is an adversary $\mathcal{A}$ breaking our scheme. Given a strong QR-RSA problem instance (y, N), where N is a safe RSA modulus and $y \in \mathbb{QR}_N$, we can construct another polynomial time algorithm $\mathcal{B}$ to break the strong QR-RSA assumption. This is done as follows. After $\mathcal{A}$ chooses the set $\hat{S}$ of t nodes he may corrupt, $\mathcal{B}$ chooses $i' \in_R \{1, 2, \cdots, n\}\backslash\hat{S}$ and $j' \in_R \{1, 2, \cdots, l\}$, and enumerates the l primes $e_1, e_2, \cdots, e_l$ after n. Then $\mathcal{B}$ prepares

$$(s^{f(0)e_{j'}\frac{F^2}{\beta_0}}, s^{f(1)e_{j'}\frac{F^2}{\beta_1}}, \cdots, s^{f(t)e_{j'}\frac{F^2}{\beta_t}}) \bmod N$$

by the following four steps:

1. Since $y \in \mathbb{QR}_N$, there exist s and $\hat{y}$ such that $s^{\hat{y}} \equiv y \pmod N$. Since $e_{j'} \ll p'$ and $e_{j'} \ll q'$, ${e_{j'}}^{-1} \bmod m$ exists (this still holds even if $e_{j'}$ itself is not prime because $m = p'q'$ has no common divisor with $e_{j'}$). Let $f(i') = {e_{j'}}^{-1}\hat{y} \bmod m$ such that $s^{f(i')e_{j'}} \equiv y \pmod N$. Note that $\mathcal{B}$ does not know $s^{f(i')}$ due to not knowing the factorization of N (and thus not being able to compute $e_{j'}^{-1} \bmod m$), but this still implicitly determines one sample point for $f(x) \bmod m$ at $x = i'$.
2. Randomly choose $r_i \in \mathbb{Z}_{\lfloor N/4 \rfloor}$. Let $s^{f(i)} \equiv y^{r_i} \equiv s^{f(i')e_{j'}r_i} \pmod N$ for $i \in \hat{S}$. That is, for these $|\hat{S}| = t$ sample points, it is implicitly defined that $f(i) \equiv f(i')e_{j'}r_i \pmod m$.
3. Compute $\beta_a = \prod_{b=0,b\neq a}^{t}(a-b)$ for $a = 0, 1, \cdots, t$.

¶F^2 is not only for canceling β_a such that division in the exponent is not necessary, but also for the security proof.

Table 1: Quick-reference notation: symbols and parameters

e_z	z-th prime after n (i.e., $e_l > \cdots > e_2 > e_1 > n$), $z \in \{1, 2, \cdots, l\}$
$k_{i,*}$	node key secretly issued by the control center
$k_{*,J}$	user key assigned for those subscribing to J
$k_{i,j}$	symmetric session key for data protection as well as data access
l	total number of time periods for user subscription (e.g., 4096)
m	product of two distinct large primes p' and q'
n	total number of sensor nodes in the WSN (e.g., 2000)
t	maximum number of sensor nodes that may be corrupted (e.g., 100)
F	factorial of n (i.e., $n!$)
$H(\cdot)$	a cryptographic one-way hash function of appropriate range and domain
J	user subscription set, an arbitrary subset of time indices $\{1, 2, \cdots, l\}$
N	product of two distinct safe primes $p\,(= 2p' + 1)$ and $q\,(= 2q' + 1)$
$\lambda(N)$	Euler's totient of n, $(= \mathrm{lcm}(p-1, q-1))$

4. Apply the Lagrange interpolation to the $(t+1)$ values of $f(x) \bmod m$ sampled at the $(t+1)$ points: i' (Step 1) and $i \in \hat{S}$ (Step 2). Thus $\mathcal{B}$ has

$$f(x) \equiv \lambda_{xi'} f(i') + \sum_{i \in \hat{S}} \lambda_{xi} f(i) \pmod{m},$$

where $\lambda_{xi'} = \prod_{j \in \hat{S}} \frac{x-j}{i'-j}$ and $\lambda_{xi} = \prod_{j \in \{i'\} \cup \hat{S}, j \neq i} \frac{x-j}{i-j}$ for $i \in \hat{S}$. Therefore,

$$\begin{aligned} s^{f(x)e_{j'}} &\equiv s^{\lambda_{xi'} f(i')e_{j'}} \cdot \prod_{i \in \hat{S}} s^{\lambda_{xi} f(i) e_{j'}} \pmod{N} \\ &\equiv y^{\lambda_{xi'}} \cdot \prod_{i \in \hat{S}} y^{r_i \lambda_{xi} e_{j'}} \pmod{N}. \end{aligned}$$

For $a = 0, 1, \cdots, t$, $\mathcal{B}$ evaluates $(s^{f(x)e_{j'}})$ at a and then raises it to the power of $\frac{F^2}{\beta_a}$,

$$s^{f(a)e_{j'} \frac{F^2}{\beta_a}} \equiv y^{\lambda_{ai'} \frac{F^2}{\beta_a}} \cdot \prod_{i \in \hat{S}} y^{r_i \lambda_{ai} e_{j'} \frac{F^2}{\beta_a}} \pmod{N}.$$

As in the actual scheme, $F^2 = (n!)^2$ cancels β_a. Similarly, F^2 also cancels the denominators in $\lambda_{ai'}$ and λ_{ai}, so $\mathcal{B}$ can prepare the above $(t+1)$ values $s^{f(a)e_{j'} \frac{F^2}{\beta_a}} \bmod N$, $a = 0, 1, \cdots, t$, without knowing the factorization of N. Also note that $\gcd(e_z, F^2) = 1$ for any $1 \leq z \leq l$ since $e_1 > n$.

Query Phase 1. For the static security setting, since $\mathcal{A}$ may corrupt the nodes with indices in $\hat{S}$, $\mathcal{B}$ can just send to $\mathcal{A}$ the keys $k_{i,*} = s^{f(i)F^2} \bmod N = y^{r_i F^2} \bmod N$ for all $i \in \hat{S}$. $\mathcal{B}$ then simulates the other oracles for $\mathcal{A}$. Without loss of generality, we assume each query is issued once only.

- MobiExt(J): If $j' \in J$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ responds with

$$k_{*,J} = (s^{f(0)E_J \frac{F^2}{\beta_0}}, \cdots, s^{f(t)E_J \frac{F^2}{\beta_t}}) \bmod N.$$

As long as $j' \notin J$, such $k_{*,J}$ can be computed from $(s^{f(0)e_{j'} \frac{F^2}{\beta_0}}, \cdots, s^{f(t)e_{j'} \frac{F^2}{\beta_t}}) \bmod N$ prepared before by raising each of the $s^{f(a)e_{j'} \frac{F^2}{\beta_a}} \bmod N$ term to the power of $\prod_{z=1, z \neq j', z \notin J}^{l} e_z$.

- Random oracle $H(w)$: $\mathcal{B}$ first checks if $w^{e_{j'}} \equiv y^{E_{j'} F^2} \bmod N$. If not, $\mathcal{B}$ responds with a random number from the key space $\mathcal{K}$. If the condition holds, since $s^{f(i')e_{j'}} \equiv y \pmod{N}$, $\mathcal{B}$ shall have $w^{e_{j'}} \equiv s^{f(i')e_{j'} E_{j'} F^2} \pmod{N}$. That is, $\mathcal{B}$ encounters a "special" w from $\mathcal{A}$ as a random oracle query: $w \equiv s^{f(i')E_{j'} F^2} \pmod{N}$. Denote the known $s^{f(i')}$ by x. Then $\mathcal{B}$ has both $y = x^{e_{j'}} \bmod N$ and $w = x^{E_{j'} F^2} \bmod N$. Note that $E_{j'} = \prod_{z=1, z \neq j'}^{l} e_z$ and thus $\gcd(e_{j'}, E_{j'} F^2) = 1$. Therefore, with y and w, $\mathcal{B}$ can employ the common modulus attack to compute $x \bmod N$, and then provide $(x, e_{j'})$ as a solution to the given instance of the strong QR-RSA problem.

Challenge. Once $\mathcal{A}$ decides that Query Phase 1 is over, he selects $(\hat{i}, \hat{j})$ that he targets. If $\hat{i} \neq i'$ or $\hat{j} \neq j'$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ responds with a random value in $\mathcal{K}$.

Query Phase 2. $\mathcal{B}$ responds to the queries as in Query Phase 1.

Guess. $\mathcal{A}$ outputs a guess b' for b. $\mathcal{B}$ terminates.

$\mathcal{A}$'s guess is "ignored". However, for $\mathcal{A}$ to do any thing "useful", $\mathcal{A}$ should have issued $s^{f(\hat{i})F^2 E_{\hat{j}}} \bmod N$ (i.e., the "special" w) to the random oracle to get the correct $k_{\hat{i},\hat{j}}$, which helps the computation of x and $e_{\hat{j}}$.

The simulation succeeds when $(\hat{i}, \hat{j}) = (i', j')$. If $\mathcal{A}$ has non-negligible advantage ε to break our scheme, $\mathcal{B}$ can solve the strong QR-RSA problem with probability $\frac{\varepsilon}{(n-t)l}$, which is also non-negligible. □

6. FEASIBILITY EVALUATION

There are three kinds of entities involved in our time-based key management scheme, namely, the sensor nodes (performing **NodeKeyDer**), the mobile users (performing **UserKeyDer**), and the control center (performing **Setup**, **NodeKeyGen**, and **UserKeyGen**). The control center is probably a more powerful server, and the knowledge of factorization of $N = pq$ can significantly accelerate its operations like computing $k_{i,*} = s^{f(i)F^2} \bmod N$ by exploiting the Chinese remainder theorem. A mobile user is usually

equipped with a device like a PDA or a smart phone. While definitely weaker than a server, the user key derivation is not an expensive operation for them. The sensor node is the one with really limited computing power. In this section, we evaluate the processing cost of deriving a session key $k_{i,j} = H(k_{i,*}^{E_j} \bmod N)$ by a node i at time period j. Note that the computation or storage cost for a sensor node is mainly affected by l, which indicates that our scheme is applicable to a very large WSN.

6.1 Case Study Settings

We assume an adversary may corrupt up to 100 out of 2000 sensor nodes, and there are a total of 4096 time periods allocated by the control center for user subscription. That is, we set $n = 2000$, $t = 100$, and $l = 4096$ as a case study. Having $l = 4096$ is large enough for most real-life applications. For examples, if each period corresponds to one hour, $l = 4096$ corresponds to nearly half a year; if each period corresponds to one day, $l = 4096$ corresponds to more than 11 years. We take the safe RSA modulus to be $|N| = 1024$ bits in length, which is the recommended security level nowadays.

Since $H(\cdot)$ is a one-way hash function (usually cost-efficient), the major workload for computing $k_{i,j} = H(k_{i,*}^{E_j} \bmod N)$ may consist of computing the product $E_j = \prod_{z=1,z\neq j}^{l} e_z$ regarding $1 \leq j \leq l$ and then computing the (modular) exponentiation $k_{i,*}^{E_j} \bmod N$. This may be computationally expensive. However, we shall show that the computation can be done in a more efficient "divide-and-conquer" manner. Let us begin with how to obtain each e_z in E_j.

6.2 Prime Enumeration

In our case study, a sensor node needs $l = 4096$ primes after n $(e_1, e_2, \cdots, e_{4096})$ to derive the session keys. There are 303 primes within $n = 2000$ (i.e., $2, 3, 5, \cdots, 1997, 1999$), and thus a sensor node actually needs to know the first $l' = 4399$ primes. To obtain all these l' primes, at first glance a sensor node may choose to employ the Sieve of Eratosthenes with $\mathcal{O}((d \log d)(\log \log d))$ time complexity and $\mathcal{O}(d)$ space complexity [21], where d is the total number of integers to sieve ($l' \approx d/\ln d$ according to the prime number theorem). Even though there is a segmented version of the Sieve of Eratosthenes with $\mathcal{O}(d)$ time complexity and $\mathcal{O}(\sqrt{d} \log \log d/\log d)$ space complexity [1], this is still too expensive for a sensor node when d (essentially, l') becomes significantly large. So it is unfavorable (if not impractical) for a sensor node to enumerate the primes by itself.

As a result, we suggest to pre-load the enumeration of the l primes after n $(e_1 = 2003, e_2 = 2011, \cdots, e_{4096} = 42071)$ into each sensor node's memory, specifically, read only memory (ROM). In other words, a sensor node only needs to look up in a static table to immediately obtain any of the $l = 4096$ primes. The storage for all the $l = 4096$ primes occupies 59738 bits = 7468 bytes (ignoring the small encoding overhead), which is fairly acceptable for the current generation of sensor nodes[‖].

6.3 Computation/Storage Tradeoff

[‖]For example, a MICAz node has 128K bytes of ROM but only 4K bytes of RAM. Detailed datasheet of the product is available for download at `http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf` (12/28 '10).

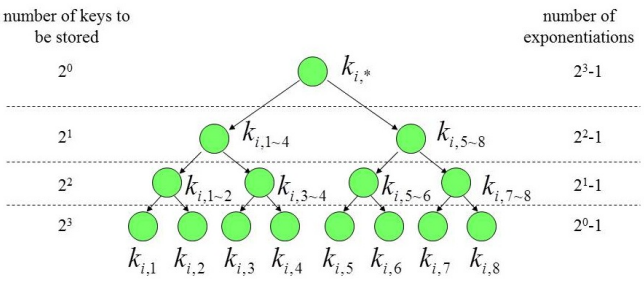

Figure 2: Sensor's computation/storage tradeoff for $l = 8$. Each stripe (separated by the dashed lines) suggests a possible approach to computing the desired key by only using exponentiation with an exponent at most $|e_l|$ bits.

We know that the product of all $l = 4096$ primes is of length 57581 bits and e_1, the first prime greater than $n = 2000$, is 11-bit long. To compute $k_{i,*}^{E_j} \bmod N$ for a certain j, a sensor node needs to compute $k_{i,*}$ to the power of a certain E_j of up to 57570 bits in length (when $|e_j| = 11$). It reminds us of potential implementation challenges at low-cost, resource-constrained sensor nodes. We address this problem by doing the exponentiations separately, and optimize the performance with the help of a key-tree structure, which trades storage for computation efficiency.

The tradeoff, depicted in Fig. 2 with $l = 8$ as a simple illustration, is to have each sensor node pre-loaded with, in addition to the primes enumeration, certain pre-computed keys in the form of $k_{i,a\sim b} = s^{f(i)F^2 \prod_{z\in\{1,\cdots,l\}\setminus\{a,\cdots,b\}} e_z} \bmod N$, so that a node can compute a session key for a specific time period more efficiently. When a node is additionally pre-loaded with 2^r pre-computed keys, it only needs to take one of these keys to the power of the product of $(\frac{l}{2^r} - 1)$ e_i's. Assuming the sensor node's processor can only perform viable exponentiations where the exponent is of at most $|e_l|$ bits in length, the computation can be implemented as $(\frac{l}{2^r} - 1)$ such viable exponentiations.

In our case study where $l = 4096$, we have the tradeoff graph as shown in Fig. 3. We can see that when $r = 1$, a node has to store the keys of total length $2^1 \times 1024/8 = 256$ bytes. When $r = 2$, a node needs to store the keys of length $2^2 \times 1024/8 = 512$ bytes, and so on. For the computational cost, we consider the heaviest case: a pre-loaded key to the power of the product of the most lengthy $(\frac{l}{2^r} - 1)$ primes. For example, assuming that a sensor node is preloaded with 2^8 pre-computed keys (i.e., $2^r|N|/8 = 256 \times 1024/8 = 32$K bytes), it needs to compute a key to the power of a 231-bit exponent. According to the recent evaluation on sensor nodes [14], this kind of exponentiation can be done in one minute even in software implementation (and less than one seconds in hardware implementation). This seems to be a reasonable balance for a sensor node. Similarly, one can appropriately trade storage for computation according to different application conditions. Furthermore, note that the key-tree as illustrated in Fig. 2 can be reused by a node i when computing $k_{i,j}$'s for different j's, and thus certain intermediate results can be cached for better computation efficiency. For example, caching $k_{i,1\sim2}$ when $k_{i,1}$ is computed enables the computation of $k_{1,2}$ in the next time period

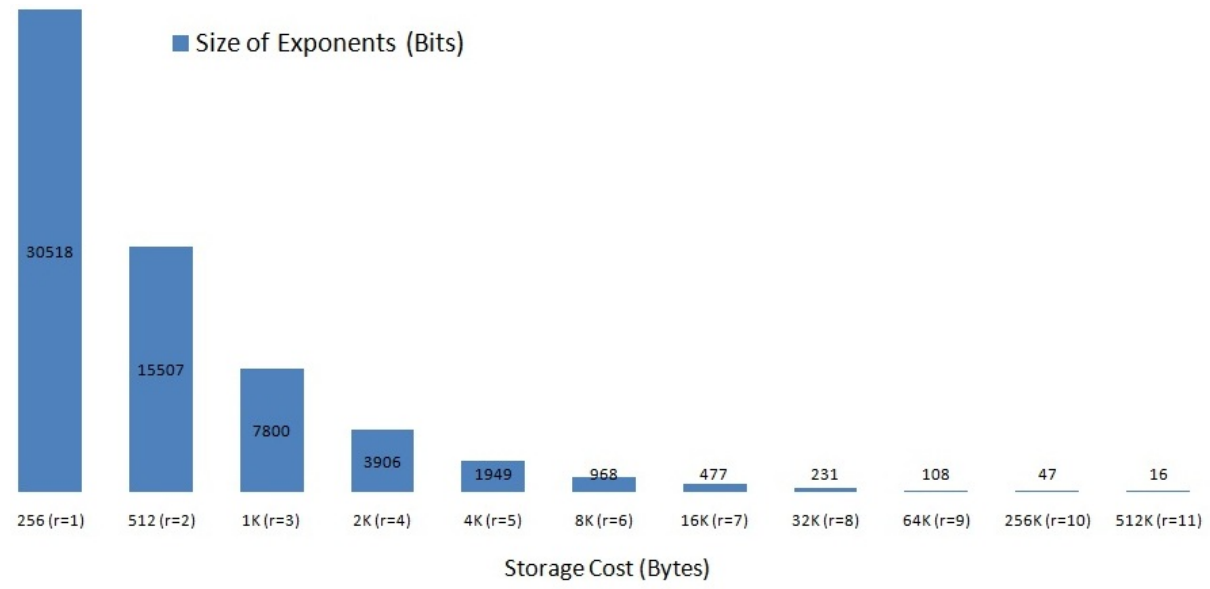

Figure 3: The computation/storage tradeoff for 4096 time periods

to be done in one small-exponent exponentiation.

Finally, we want to stress that while the storage size may look logarithmic to the total number of time periods, it is still independent of the total number of nodes, as we discussed in Section 2.2.

7. CONCLUSION

Economic research has demonstrated that a subscription-based payoff model is more profitable than a traffic-based one for the service owner. We envision a business model for data provision services in wireless sensor networks, where a time-based access control is adopted to protect data confidentiality. As an instantiation of the access control mechanism, we proposed `SMS-SED`, a secure (t, n)-threshold time-based key management system for secure mobile subscription of sensor-encrypted data. The primary feature of the scheme lies in that the control center can delegate the data access rights to mobile users with respect to their subscription time periods. Our cryptographic construction achieves provable security even if an adversary can capture up to t sensor nodes and reveal the stored node keys. We show that our system is practical for the current generation of wireless sensor networks and a tradeoff study for balancing between the processing overhead and the storage cost of a sensor node. The high efficiency of our system is from the design principles of making common cases fast and supporting a right level of security.

8. ACKNOWLEDGMENTS

This work was supported by the Singapore A∗STAR project SEDS-0721330047 and by the National Natural Science Foundation of China under Grant 60970138.

9. REFERENCES

[1] A. O. L. Atkin and D. J. Bernstein. Prime sieves using binary quadratic forms. *Mathematics of Computation*, 73(246):1023–1030, 2004.

[2] Y. Bakos and E. Brynjolfsson. Bundling information goods: Pricing, profits, and efficiency. *Management Science*, 45(12):1613–1630, 1999.

[3] N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology - EUROCRYPT'97*, volume 1233 of *LNCS*, pages 480–494, 1997.

[4] M. Bellare, P. Rogaway, and D. Wagner. The EAX mode of operation. In *Fast Software Encryption (FSE'04)*, volume 3017 of *LNCS*, pages 389–407, 2004.

[5] J. Benaloh and M. de Mare. Efficient broadcast time-stamping. Technical Report TR-MCS-92-1, Clarkson University Department of Mathematics and Computer Science, April 1992.

[6] H. Chan, A. Perrig, and D. X. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy (S&P'03)*, pages 197–213, 2003.

[7] H.-Y. Chien. Efficient time-bound hierarchical key assignment scheme. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(10):1301–1304, 2004.

[8] S. S. M. Chow, M. H. Au, and W. Susilo. Server-aided signatures verification secure against collusion attack. In *6th ACM Symposium on Information, Computer and Communications Security (AsiaCCS'11)*, 2011. To appear.

[9] S. S. M. Chow, J. K. Liu, and J. Zhou. Identity-based online/offline key encapsulation and encryption. In *6th ACM Symposium on Information, Computer and Communications Security (AsiaCCS'11)*, 2011. To appear.

[10] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *6th ACM Conference on Computer and Communications Security (CCS'99)*, pages 46–51, 1999.

[11] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K.

Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *IEEE International Conference on Computer Communications (INFOCOM'04)*, pages 597–607, 2004.

[12] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Transactions on Information and System Security (TISSEC)*, 8(2):228–258, May 2005.

[13] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *9th ACM Conference on Computer and Communications Security (CCS'02)*, pages 41–47, 2002.

[14] W. Hu, P. I. Corke, W. C. Shih, and L. Overs. secFleck: A public key technology platform for wireless sensor networks. In *6th European Conference on Wireless Sensor Networks (EWSN'09)*, volume 5432 of *LNCS*, pages 296–311, 2009.

[15] D. Liu and P. Ning. Improving key predistribution with deployment knowledge in static sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 1(2):204–239, Nov 2005.

[16] D. Liu, P. Ning, and R. Li. Establishing pairwise keys in distributed sensor networks. *ACM Transactions on Information and System Security (TISSEC)*, 8(1):41–77, Feb 2005.

[17] D. Micciancio. The RSA group is pseudo-free. In *Advances in Cryptology - EUROCRYPT'05*, volume 3494 of *LNCS*, pages 387–403, 2005.

[18] A. Mohaisen, D. Nyang, and K. Lee. Hierarchical grid-based pairwise key pre-distribution in wireless sensor networks. *Intl. Journal of Network Security (IJNSEC)*, 8(1):282–292, 2009.

[19] B. Parno, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. In *IEEE Symposium on Security and Privacy (S&P'05)*, pages 49–63, 2005.

[20] R. Poovendran, C. Wang, and S. Roy, editors. *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*. Advances in Information Security. Springer, 2006.

[21] P. Pritchard. Linear prime-number sieves: A family tree. *Science of Computer Programming*, 9(1):17–35, 1987.

[22] V. Shoup. Practical threshold signatures. In *Advances in Cryptology - EUROCRYPT'00*, volume 1807 of *LNCS*, pages 207–220, 2000.

[23] W.-G. Tzeng. A time-bound cryptographic key assignment scheme for access control in a hierarchy. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 14(1):182–188, 2002.

[24] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *11th ACM Conference on Computer and Communications Security (CCS'04)*, pages 354–363, 2004.

[25] X. Yi. Security of Chien's efficient time-bound hierarchical key assignment scheme. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(9):1298–1299, 2005.

[26] X. Yi and Y. Ye. Security of Tzeng's time-bound key assignment scheme for access control in a hierarchy. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 15(4):1054–1055, 2003.

[27] W. T. Zhu, R. H. Deng, J. Zhou, and F. Bao. Time-bound hierarchical key assignment: An overview. *IEICE Transactions on Information and Systems*, E93-D(5):1044–1052, 2010.