A Distributed Group Rekeying Scheme for Wireless Sensor Networks

Seyed Hossein Nikounia, Amir Hossein Jahangir
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
nikoonia@ce.sharif.ir, jahangir@sharif.ir

Vanesa Daza
Department of Information and Communication Technologies
Pompeu Fabra University
Barcelona, Spain
vanesa.daza@upf.edu

***Abstract*—In applications that require group communication and clustering, there is usually a single key for all members of the group. This key should be updated in order to support dynamic nature of the groups and also to handle possible node compromise attack. In this paper, we propose a new distributed group rekeying scheme with t-revocation capability that is based on local collaboration of group members. Our proposed scheme provides t-wise backward and forward secrecy. It can be used with any key size. This scheme, in contrast to centralized schemes, does not require a centralized rekeying server, so the rekeying process is handled locally in the group itself and the communication overhead is reduced. The security of this scheme is analyzed. We have also implemented our proposal for TinyOS and have used Avrora to simulate the compiled binary for MICA2 motes. Simulation results show that compared to the only published distributed scheme, our scheme consumes less energy and has lower communication overhead.**

***Keywords*-Security; Key Management; Group Rekeying; Wireless Sensor Networks.**

I. Introduction

Wireless sensor networks consist of many small low-cost and low-power nodes that sense their environment, process data, and communicate through wireless links [1]. These networks are often deployed in adverse or even hostile environments. Nodes are resource-constrained and they are often deployed in unattended manner. Due to cost limitations, it is not practical to use tamper-proof hardware for all nodes. Hence, an adversary can mount a physical attack on a node and read, probably secret, data from its memory. These issues make providing security services a challenging task.

Grouping is a technique to do localized computation and to reduce communication overhead in wireless sensor networks. The most common grouping technique is clustering. Cluster head usually do coordination and some aggregation to send the results back to the sink.

There is usually a group-wide key, called the *group key*, shared between group members. When a node become compromised, we remove the compromised node by not revealing the new group key to that node. The process of renewing the group key is called *group rekeying*. This is also referred to as *group key revocation* in some literature.

In this paper, we review the existing schemes for group rekeying. We propose a new group rekeying scheme which is not based on a centralized rekeying server. We have compared our proposed scheme with other group rekeying schemes using various performance parameters including communication and computation overhead.

The rest of this paper is organized as follows. Section I-A presents preliminaries including notations and definitions as well as description of Shamir's secret sharing scheme. Section II reviews existing techniques for group rekeying in sensor networks. In Section III, we describe our proposed scheme. Sections IV and V presents simulation results and performance analysis, respectively. We compare our proposed scheme with a distributed scheme in Section VI. Finally, this paper ends with conclusions in Section VII.

A. Goals

The general goal is to develop an efficient and unconditionally secure rekeying scheme for wireless sensor networks. This scheme should be able to tolerate node compromise.

Due to hardware constraints of sensor nodes, the harsh environments in which sensor networks are often deployed and also security requirements, a suitable rekeying scheme should provide:

- t-revocation capability (See Definition 1).
- t-wise forward secrecy (See Definition 2).
- t-wise backward secrecy (See Definition 2).
- On-demand rekeying: a suitable scheme should provide a mechanism for revoking a compromised node from the group on-demand.
- and also low communication, computation and low storage overhead.

In this section, we define some notations and definitions for our proposed scheme. We also describe Shamir's Secret Sharing scheme. The idea of Shamir's secret sharing scheme is usually used in group rekeying schemes.

B. Notations and Definitions

We assume a group of n sensor nodes, deployed closely to each other within a large scale sensor network. A group consists of $n - 1$ group members and one group controller.

The group controller is responsible for the management of the group. Each group member has an ID $i > 0$, and a secret key shared with the group controller. There is a group key K shared with all group members. They use this key to get confidentiality and/or integrity of their communication.

When needed, the group controller uses a rekeying scheme to update K. Let us call the j-th group key (group key of session j) K_j. Each node stores a personal secret. Node i, stores S_i as its personal secret. S_i is used in the rekeying process. This secret is known only by the node itself.

The group controller renew the current group key when, for example, a node become compromised. Hence, the rekeying mechanism should have the ability not to reveal the new group key to the compromised nodes.

In a rekeying event, there might be w nodes that should be revoked (i.e., the new group key should not be revealed to these nodes). After the rekeying process, those nodes are no more members of the group.

There are mainly three pieces of information that are used in the rekeying process:

- The personal secrets, S_i, that every sensor hold.
- Rekeying materials that should not be revealed and are used by the group controller (or the rekeying server) to compute a broadcast message.
- The broadcast message. Group members use this message plus S_i to compute the new group key.

There are three types of actors:

- Group controller which acts like a coordinator in a rekeying event.
- Group members $\mathcal{U} = \{U_1, \ldots, U_n\}$, which are the group of sensonr nodes. When a member is not accepted to be part of the group anymore, it is called revoked.
- Network manager is a person who initialize the nodes offline (i.e., before deployment)

Please note that we are assuming an adversary that can do the node compromise attack and the network IDS is capable of detecting it. The attacker can also eavesdrop the wireless communications. To further clarify our goals, we give the following definitions.

To further clarify our goals, we give the following definitions.

Definition 1. *(Group rekeying with revocation capability) Let $t, i \in \{1, \ldots, n\}$ and p be a prime number. In a group rekeying Ξ, the group controller seeks to establish a new $K \in \mathbb{F}_p$ with each group member U_i through a broadcast message and some personal information S_i it owns. In detail:*

1) *Ξ is a group rekeying scheme if*
    a) *For any group member U_i, K is determined by S_i and B.*
    b) *For any set $M \subset \mathcal{U}$, $|M| \leq t$, and any $U_i \notin M$, the members in M are not able to learn anything about S_i.*
    c) *No information is leaked from either the broadcast message or the S_i alone.*
2) *Ξ has t-revocation capability if given any set of revoked group members $R \subset \mathcal{U}$ such that $|R| \leq t$, the group controller can generate a broadcast message B such that $U_i \notin R$, U_i can recover K but the revoked group members cannot recover K.*

Definition 2. *(t-wise backward and forward secrecy) Let $t, i \in \{1, ..., n\}$, $j \in \{1, ..., m\}$ and $K_j \in \mathbb{F}_p$ be the group key of session j.*

1) *A rekeying scheme guarantees t-wise forward secrecy if for any set $R \subseteq \{U_1, ..., U_n\}$, where $|R| \leq t$ and all $U_i \in R$ are revoked before session j, the members in R together cannot get any information about K_j, even with the knowledge of group keys before session j.*
2) *A rekeying scheme guarantees t-wise backward secrecy if for any set $R \subseteq \{U_1, ..., U_n\}$, where $|R| \leq t$ and all $U_i \in R$ joined after session j, the members in R together cannot get any information about K_j, even with the knowledge of group keys after session j.*

Similar definitions are also used in [2].

C. Shamir's Secret Sharing Scheme

The idea of Shamir's secret sharing scheme is usually used in group rekeying schemes. It is used in our scheme as well. The goal is to share a secret S between n people so that $t + 1$ (or more) of them can recover S. For this purpose, a random t-degree polynomial in which $S = P(0)$ is generated. This polynomial is evaluated over $\mathbb{F}_p$, where p greater than n. $P(i)$ is given to person $i > 0$ as his/her share. Now, $t + 1$ (or more) person can recover the original polynomial, and hence S. But having less than $t + 1$ shares do not give any information about S. See Theorem 1.

Theorem 1. *Suppose the opponent knows t shares of this polynomial. For each candidate value $S' \in [0, p-1]$ he can construct one and only one polynomial $P'(x)$ of degree t such that it satisfies conditions of shares and also $P'(0) = S'$.*

By construction, these p possible polynomials are equally likely, and thus there is absolutely nothing the opponent can deduce about the real value of S.

Proof: Refer to [3] for the proof. ∎

II. Related Work

Mainly, there are two categories of schemes for group rekeying in sensor networks: there are some distributed schemes which do not rely on a rekeying server and there are some centralized schemes which require a rekeying server to construct the broadcast message. It is assumed that the rekeying server is secured and could not be compromised. Here we review both distributed and centralized schemes, but before that let us review some basic methods of sharing a secret to only legitimate members that are used in some of the schemes described later.

A. Underlying Methods

In this section, we review some general solutions to the problem of revealing a secret to non-revoked group members. These solutions are underlying method for most of the group rekeying schemes for wireless sensor networks.

1) Method 0 (Naive): Every group member should have a secret key shared with the group controller. The group controller can encrypt K_j with this secret key for each member and send the message to the appropriate member. The communication overhead of this scheme is $O(n)$ where n is the number of group members.

2) Method 1: In this scheme, each member has an ID $i > 0$. A t-degree random polynomial $P(x)$ which is evaluated over $\mathbb{F}_p$ (p is prime) is constructed and shares of it (i.e., $P(i)$) are pre-distributed to group members. The secret to be revealed is $K = P(0)$.

Suppose that $w = t$. The group controller reveals shares of revoked members. At this point, every non-revoked group members have $t + 1$ shares and can recover the original polynomial so non-revoked group members can evaluate $K = P(0)$.

This scheme has been proposed in [4]. It can also be used for $w < t$ if the group controller reveals shares of w revoked-members plus shares of arbitrarily selected $w - t$ dummy members.

3) Method 2: In [2], the group controller randomly picks a $2t$-degree masking polynomial $h(x) = h_0 + h_1x + ... + h_{2t}x^{2t}$ over a finite field $\mathbb{F}_p$ where p is prime. Each group member i gets its personal secret $S_i = h(i)$ from the group controller.

Given a set of revoked group members $R = \{r_1, r_2, ..., r_w\}, w \leq t$, the group controller randomly picks a t-degree polynomial $p(x)$ and constructs $q(x) = K - p(x)$. Then the controller distributes the shares of the t-degree polynomials $p(x)$ and $q(x)$ to non-revoked sensors using the following broadcast message:

$$\begin{aligned} B = \{R\} \\ \cup\{P(x) = g(x)p(x) + h(x)\} \\ \cup\{Q(x) = g(x)q(x) + h(x)\} \end{aligned}$$

where $g(x) = (x - r_1)(x - r_2)...(x - r_w)$. If any non-revoked group member i receives such a broadcast message, it evaluates polynomials $P(x)$ and $Q(x)$ at point i. and gets $P(i) = g(i)p(i) + h(i)$ and $Q(i) = g(i)q(i) + h(i)$.

Because member i knows $h(i)$ and $g(i) \neq 0$, it can compute $p(i) = \frac{P(i)-h(i)}{g(i)}$ and $q(i) = \frac{Q(i)-h(i)}{g(i)}$. Member i can then compute the new group key $K = p(i) + q(i)$. The revoked members (which are not member of the group anymore) cannot compute K because $g(i) = 0, \forall i$ revoked.

As it is proved in [2], this schemes is unconditionally secure rekeying scheme with t-revocation capability. It also provides t-wise backward and forward secrecy.

B. Distributed Schemes

In [5], a group rekeying protocol has been proposed. In this protocol, rekeying materials are preloaded into each node. Each member distributes encrypted shares of its rekeying materials to other nodes which will be returned back to the node in a rekeying event. There is also some improvements to their basic protocol, B-PCGR, which improves its security. To the best of our knowledge, this is the only published distributed group rekeying scheme for sensor networks.

C. Centralized Schemes

In [6], Danio and Savio have proposed a group key revocation protocol for wireless sensor networks that has communication overhead of $O(\log n)$, instead of $O(n)$ in naive scheme (see Subsection II-A1). This protocol also provides a lightweight key authentication using one-way hash chains. In this protocol, each node has a symmetric key shared with the keying server. They have proposed to use a (binary) tree of hash chains. Leaves are assigned to group members and each group member has the current key in the hash chain of the nodes which are in the path between this leaf and the root.

Authors have shown that using this structure, the number of messages are reduced to $O(\log n)$ but some of the messages should be sent to more than one member. But the drawback of this scheme is that it does not provide backward secrecy and that is due to the use of hash chains of keys.

In [7], a self-healing group key revocation has been proposed. In this protocol, lifetime of the group is divided into some intervals and nodes can authenticate the new group key using a dual hash chain. There is no communication overhead for revocation and it can tolerate rekeying message loss (the self-healing property of this scheme).

But there are some drawbacks. The revocation could not be done on-demand, network manager should plan for the revocation time in advance. And also there is an implicit assumption that the adversary is not able to compromise group nodes and hence, could not read the keying materials in their memory.

Another protocol for updating group key has been proposed in [8]. They adapt the secret-sharing revocation scheme that is explained in Subsection II-A2 for sensor networks by reducing the computation overhead. A centralized group rekeying scheme has been proposed in [9]. The underlying rekeying scheme is very similar to Method 2 (see Subsection II-A3) that is also used in [10].

D. Motivation

While centralized schemes require a secured rekeying server and also a secure connection between the group controller and the rekeying server, the only published distributed scheme [5] has a large communication overhead.

A secured rekeying server might not be applicable for some applications in which the network is being deployed in adverse environments. In addition to computation and communication overhead of the aforementioned distributed scheme, it has another drawback. In order to provide t-revocation capability, the underlying IDS should provide each group member with the information of compromised nodes.

We propose to distribute shares of required materials for a rekeying event (which are stored in the rekeying server in centralized solutions) between group members using a method inspired by [5]. The group controller uses these shares and constructs a broadcast message similar to [2] (see Subsection II-A3). In this scheme, a rekeying server is not required and as we show in the following sections, the energy consumption of the proposed scheme is less than [5].

III. A New Group Rekeying Scheme

In this section, we describe our proposed group rekeying scheme for wireless sensor networks. The goal is to build a distributed group rekeying scheme with t-wise backward and forward secrecy without the need of a secured rekeying server. A group consists of $n-1$ group members and one group controller. Let $R = \{r_1, r_2, ..., r_w\}$ be the set of group members to be revoked in rekeying event j.

In this scheme, shares of required rekeying materials are pre-distributed between group nodes. In a rekeying event, they deliver their shares to the group controller and the group controller uses these shares to compute the broadcast message. Group members renew the group key using this message. Note that all polynomials are evaluated over $\mathbb{F}_p$ where p is prime. In this scheme, $t < n$, $1 \leq \lambda \leq 2t$ and $\mu \geq t$ are system parameters. $\mu \leq n$ is assumed. We'll discus how to choose these parameters later.

A. Details

The initialization process is as follows. These operations are done offline by the network manager:

1) Generate the random polynomial $h(x, y)$. The degree of x and y are $2t$ and λ, respectively.
2) Generate the random polynomial $e(x, y, u)$. The degree of x, y and u are $2t$, λ and μ, respectively and $t \leq \mu \leq 2t$.
3) Let $h'(x, y)$ be a polynomial defined as $h'(x, y) = h(x, y) + e(x, y, 0)$;
4) Then $h(i, y)$ and $e(x, y, i)$ are predistributed (or sent by the sink) to group member i. Actually, $\lambda+1$ group members should have $e(x, y, i)$, but for the sake of fault tolerance, we may distribute to more than $\lambda + 1$ members. Note that $h(i, y)$ and $e(x, y, i)$ are one and two variate polynomials, respectively. They are the result of evaluation of polynomials h and e for each group member i.
5) $h'(x, y)$ is kept by the group controller.

The j-th rekeying process (revealing j-th group key, K_j) is as follows:

1) The group controller sends a request to $\mu + 1$ group members for sending their shares of e. Note that we've assumed $\mu \leq n$.
2) They send back $e(x, j, i)$ to the group controller, where i is the ID of the member. To prevent eavesdropping, encryption might be used in this step. Shares can be encrypted with a pairwise key between group member i and the group controller. In this case, any encryption scheme might be used. Based on the used encryption algorithm and key length, group members and the group controller consume energy for this process.
3) As the group controller receives shares, it follows the steps:
   - The group controller constructs $e(x, j, u)$ by solving $\mu+1$ $(\mu+1)$-variable linear equations. It then computes $h(x, j) = h'(x, j) - e(x, j, 0)$.
   - Let $g(x) = (x - r_1)(x - r_2)...(x - r_w)$;
     Generate a t-degree random polynomial $p(x)$;
   - Let $q(x) = K_j - p(x)$.
     Broadcast the following message to the group members:
$$B = \{R\}$$
$$\cup\{P(x) = g(x)p(x) + h(x, j)\}$$
$$\cup\{Q(x) = g(x)q(x) + h(x, j)\}$$
4) Non-revoked group member i could evaluate polynomials $P(x)$ and $Q(x)$ at point i, and gets $P(i) = g(i)p(i) + h(i, j)$ and $Q(i) = g(i)q(i) + h(i, j)$. Since for non-revoked group members $g(i) \neq 0$, they can compute $p(i) = \frac{P(i)-h(i,j)}{g(i)}$ and $q(i) = \frac{Q(i)-h(i,j)}{g(i)}$. The new group key is $K_j = p(i) + q(i)$.

B. Example

Here's a simple example of our proposed scheme with four group members and one group controller. We assumed $t = \lambda = \mu = 2$. Figure 1 shows their location. From the initialization process (See Figure 2), group member i has $h(i, y)$ and $e(x, y, i)$ and the group controller has $h'(x, y)$. Assume that in $7th$ rekeing member 4 is the one to be removed; That is $R = \{4\}$. Group controller asks non-revoked members to send their shares. Member i sends back $e(x, 7, i)$ (See Figure 3). As in Figure 4 Group controller computes the broadcast message B and broadcast it to all members (See Figure 5). Non-revoked members (i.e., member 1, 2 and 3) can get the new group key K_7 while revoked member 4 cannot (See Figure 6).

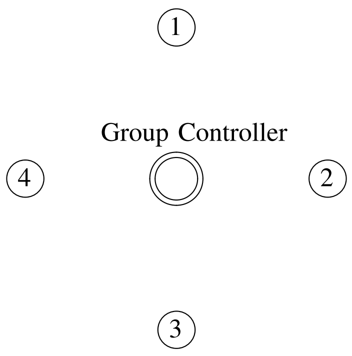

Figure 1. Location of a group controller and 4 group members

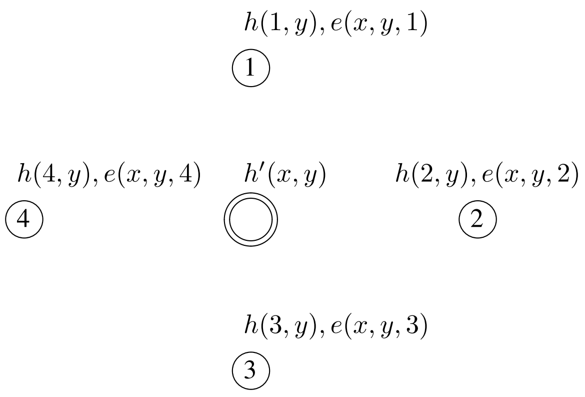

Figure 2. After initialization process; group member i has $h(i,y)$ and $e(x,y,i)$ and the group controller has $h'(x,y)$.

C. How to Choose System Parameters

This scheme can handle up to t revocations in one rekeying event. In order to compromise the whole group, an adversary should compromise the group controller plus $\mu+1$ (or more) group members. In order to guarantee t-wise backward and forward secrecy, $t \leq \mu$ should be considered.

Having larger μ does not straighten backward and forward secrecy. However it makes it harder for the adversary to compromise e polynomial. The adversary should capture $\mu+1$ group members in addition to the group controller.

Group members do not reveal their original share to the group controller. Instead, they send a *session share* for that specific session. Although they send it encrypted, in order to enhance the security of this scheme, we put a constrain

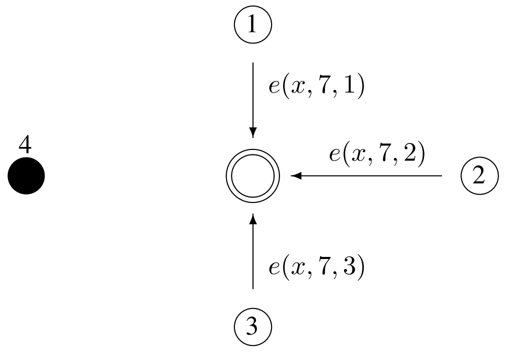

Figure 3. 7th rekeying: Group members sending their shares to the group controller; Member 4 to be revoked; In other word $R = \{4\}$

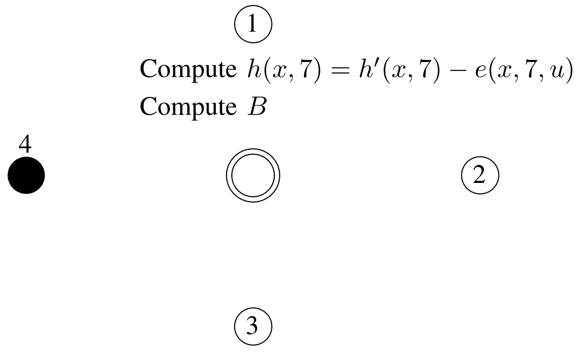

Figure 4. 7th rekeying: Group controller receives shares and compute $h(x,7)$ and the broadcast message B.

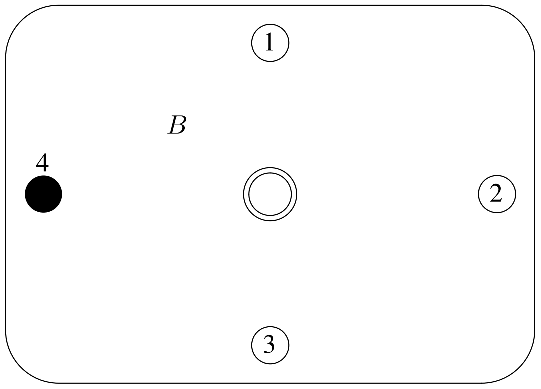

Figure 5. 7th rekeying: Group controller broadcast B; Anyone can receive B.

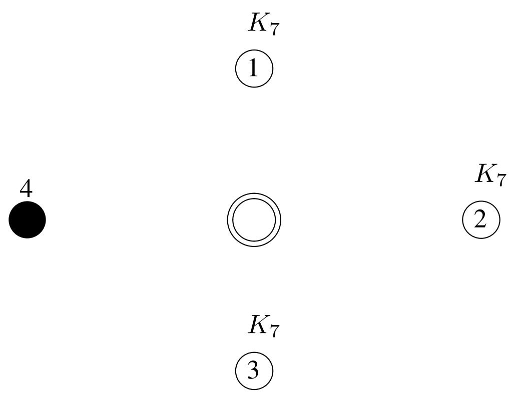

Figure 6. 7th rekeying: Group members (members 1,2 and 3) can compute the new group key but revoked member(s) (member 4) cannot get the new group key since $g(4) = 0$.

that knowing less than $\lambda + 1$ session share of a node do not provide any information about the original stored share. Based on the cryptographic algorithm that is used for sending the shares, $1 \leq \lambda \leq 2t$ should be chosen.

D. Fault Tolerance

In the proposed scheme, like other schemes where the group controller is responsible for sending the broadcast message, the group controller is a single point of failure. In order to tolerate k failures in the group controller, it could be possible to have k group controllers (only one of them is active at a time). But group nodes should be able to trust k group controllers instead of one.

If h becomes compromised, the whole rekeying mechanism is compromised. In order to tolerate k compromises of h polynomial, it is possible to have k distinct instances of the scheme with k group controllers. So each group member has k personal secrets.

E. Security Analysis

According to Theorem 4, this scheme has t-revocation capability. It also provides t-wise backward and forward secrecy.

Theorem 2. *$h(x, y)$ is compromised if and only if*

1) *$\mu + 1$ (or more) shares of e are compromised and $h'(x, y)$ is also compromised.*
2) *or $2t + 1$ of group members become compromised.*

Proof: Having $\mu + 1$ (or more) shares of e, one can find the original $e(x, y, u)$ by solving $\mu + 1$ $(\mu + 1)$-variable linear equations. Knowing less than $\mu + 1$ share, $e(x, y, u)$ could not be constructed. It is clear that $h(x, y)$ could be constructed if and only if $e(x, y, u)$, and $h'(x, y)$ are also available.

Table I
MAXIMUM AND MINIMUM AMOUNT OF MEMORY CONSUMPTION IN THE GROUP CONTROLLER (BYTES). $\mu = \lambda = t$

$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$	
400	812	1440	2332	3536	Minimum
656	1200	1976	3032	4416	Maximum

Table II
MAXIMUM AND MINIMUM AMOUNT OF MEMORY CONSUMPTION IN THE GROUP MEMBER (BYTES). $\mu = \lambda = t$

$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$	
112	184	272	376	496	Minimum
164	256	364	488	628	Maximum

If $2t + 1$ group members become compromised, actually $2t+1$ of $h(i, y)$ are compromised which are enough material to reconstruct h polynomial. ■

Theorem 3. *This scheme has t-revocation capability. It also provides t-wise backward and forward secrecy.*

Proof: The proof is similar to the proof of Theorem 2. Note that we have assumed $t \leq \mu$. ■

IV. IMPLEMENTATION AND SIMULATION

We have implemented our proposed scheme for TinyOS-2.1.0 [11]. We installed TinyOS on Ubuntu Linux with 2.6.24-16-server kernel. We have used Avrora (Beta 1.7.10) [12] to simulate the implemented code. Simulation results are give bellow.

In our implementation, we have used dynamic memory allocation for storing polynomials coefficients. Although this code has been tested on a real MICAz mote [13], we are not claiming that it is perfectly optimized. Since the largest integer data type in TinyOS is `uint64_t`, we used `uint32_t` for coefficients in polynomials[1]. So the key length is 32 bits. We have also implemented the basic version of the only published distributed scheme, B-PCGR [5], using the same computation engine as ours.

A. Memory Usage

To have a better understanding of memory usage of our implementation, we logged the amount of memory allocated (`malloc()`) and freed (`free()`) for each phase of the rekeying procedure.

The initially allocated memory (minimum) and the maximum amount of allocated memory during execution of rekeying process in the group controller and a group member are shown in Table I and II, respectively.

Note that these figures are only the amount of dynamically allocated memories and does not contain memory usage of function codes, local variables, etc.

[1]to be able to have multiplication of two 32-bit integers

Table III
ENERGY CONSUMPTION OF COMPUTATION IN EACH PHASE OF THE PROPOSED SCHEME ON A MICA2 MOTE. THE GROUP CONTROLLER DOES STEP 3 ONCE. $\mu + 1$ GROUP MEMBERS DO STEP 2. ALL GROUP MEMBERS DO STEP 4

Step 3	Step 2	Step 4	
491.85 μJ	56.6 μJ	129.83 μJ	$t = \lambda = \mu = 2$
861.73 μJ	104.59 μJ	146.63 μJ	$t = \lambda = \mu = 3$
1334.56 μJ	167.32 μJ	163.39 μJ	$t = \lambda = \mu = 4$
1946.38 μJ	244.75 μJ	191.38 μJ	$t = \lambda = \mu = 5$

Table IV
ENERGY CONSUMPTION OF COMPUTATION IN EACH PHASE OF B-PCGR ON A MICA2 MOTE. COMPUTING SHARES IS DONE IN EACH GROUP MEMBER $\mu + 1$ TIMES. EACH GROUP MEMBER COMPUTES K ONCE

Computing the group key	Computing shares	
272.9 μJ	0.5 μJ	$t = \mu = 2$
644.71 μJ	0.62 μJ	$t = \mu = 3$
1178.03 μJ	0.79 μJ	$t = \mu = 4$
1944.11 μJ	0.99 μJ	$t = \mu = 5$

B. Energy Consumption

Although radio communications consume most of the motes' energy, energy consumption of computations should also be considered. We have used Avrora to measure the energy consumption of computations of each phase of the rekeying procedure for our proposed scheme as well as B-PCGR [5] in MICA2 motes.

For this purpose, codes of each phase of the rekeying process have been run in a `for` loop for 100 times. The energy consumption has been measured with and without running loop and the difference divided by 100 is reported for the energy consumption of that phase.

Table III and IV demonstrate the measured figures for our proposed scheme and B-PCGR, respectively. As it is clear, the most power hungry part of our scheme runs in the group controller. While each phase of B-PCGR consumes an small mount of energy, these phases should be run several times.

Figure 7 shows total energy consumption of computations of our scheme with $\mu = \lambda = t$ and B-PCGR with $\mu = t$ for $n = 10$. Our scheme consumes less energy compared to B-PCGR, and the difference becomes more significant for larger ts.

Figure 8 demonstrates how growth of n affects the total energy consumption of computations in our scheme with $\mu = \lambda = t = 3$ and B-PCGR with $\mu = t = 3$.

C. Computation Time

We have measured computation time of our scheme using Avrora simulator for MICA2 motes. Table V presents the results.

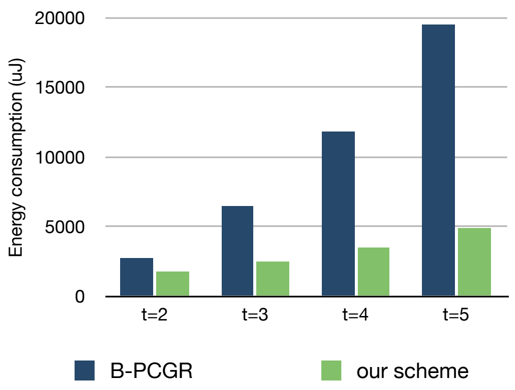

Figure 7. Total energy consumption (μJ) of our proposed scheme and B-PCGR for $n = 10$

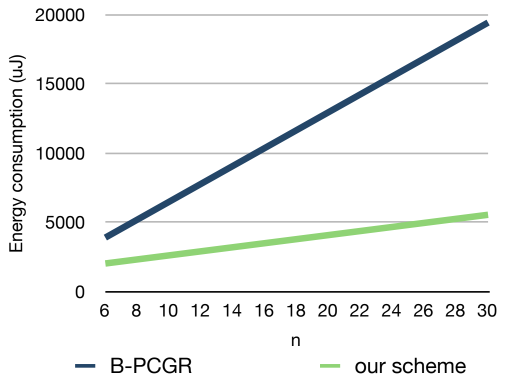

Figure 8. Total energy consumption (μJ) of our proposed scheme with $\mu = \lambda = t = 3$ and B-PCGR with $\mu = t = 3$

D. Communication Overhead

Table VI and VII shows total size of the payloads that should be sent in our scheme and B-PCGR, respectively for $n = 10$. Figure 9 compares total payload size of the sent packets in our scheme and B-PCGR for different values of n.

Table V
COMPUTATION TIME OF OUR IMPLEMENTATION ON A MICA2 MOTE. THE GROUP CONTROLLER DOES STEP 3 ONCE. $\mu + 1$ GROUP MEMBERS DO STEP 2. ALL GROUP MEMBERS DO STEP 4

Step 3	Step 2	Step 4	
22.22 ms	2.68 ms	6.39 ms	$t = \lambda = \mu = 2$
39.06 ms	5 ms	7.77 ms	$t = \lambda = \mu = 3$
60.68 ms	8.09 ms	9.48 ms	$t = \lambda = \mu = 4$
88.74 ms	11.98 ms	12.11 ms	$t = \lambda = \mu = 5$

Table VI
TOTAL PAYLOAD SIZE IN OUR SCHEME

total size	share request packet	no.	share packet	no.	broadcast message	no.	
120 Byte	= 12 Byte	×1	+20 Byte	×3	+48 Byte	×1	$t = \lambda = \mu = 2$
196 Byte	= 16 Byte	×1	+28 Byte	×4	+68 Byte	×1	$t = \lambda = \mu = 3$
288 Byte	= 20 Byte	×1	+36 Byte	×5	+88 Byte	×1	$t = \lambda = \mu = 4$
396 Byte	= 24 Byte	×1	+44 Byte	×6	+108Byte	×1	$t = \lambda = \mu = 5$

Table VII
TOTAL PAYLOAD SIZE IN B-PCGR FOR $n = 10$

total size	share request packet	no.	share packet	no.	
240 Byte	= 12 Byte	×10	+4 Byte	×30	$t = \mu = 2$
320 Byte	= 16 Byte	×10	+4 Byte	×40	$t = \mu = 3$
400 Byte	= 20 Byte	×10	+4 Byte	×50	$t = \mu = 4$
480 Byte	= 24 Byte	×10	+4 Byte	×60	$t = \mu = 5$

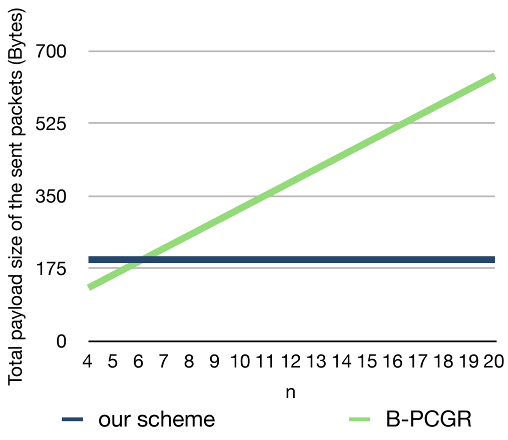

Figure 9. Comparison of total payload size of sent packets in our scheme and B-PCGR

E. Source Code

The source code is available in `http://ce.sharif.edu/~nikoonia`

V. PERFORMANCE ANALYSIS

In this section, we analytically evaluate the performance of the proposed scheme. We compare the performance of our scheme and the only published distributed scheme in Section VI. In the next sub-sections, we assume a group of n sensor nodes that do their key management computations in $\mathbb{F}_p$. Hence, the key size is $\lceil \log q \rceil$.

A. Communication Cost

In a rekeying event, $\mu + 1$ nodes should send their session share of size $(2t + 1) \times \lceil \log q \rceil$ bit to the group controller; The group controller computes a broadcast message of size $(w + 2 \times (2t + 1)) \times \lceil \log q \rceil$ bits[2]. In the worst case, $w = t$. So in the worst case, the broadcast message size is $(5t + 2) \times \lceil \log q \rceil$ bits

B. Computation Overhead

In a rekeying event, $\mu + 1$ nodes must evaluate e polynomial at an specific point (i.e., j) which has a computation overhead of $O(t\lambda)$ modular arithmetic operation.

The group controller has to rebuild e polynomial and evaluate it for $u = 0$ from $\mu + 1$ shares using Gaussian elimination which together requires $O(\mu^2)$ arithmetic operation. Computing $P(x)$ and $Q(x)$ for the broadcast message requires $O(t^2)$ modular arithmetic operation.

Finally, group members need to do $O(t)$ modular arithmetic operation to recover K.

So the computation overhead of a rekeying operation for the group controller is $O(t^2 + \mu^3)$ and the average computation overhead for each group member is $O(t) + \frac{\mu+1}{n} O(t\lambda)$.

C. Storage Requirements

The group controller stores $h'(x, y)$ which is $(2t + 1) \times (\lambda + 1) \times \lceil \log q \rceil$ bits. Group member i should store $h(i, y)$ and $e(x, y, i)$ which needs $(\lambda + 1) \times \lceil \log q \rceil$ and $(2t + 1) \times (\lambda + 1) \times \lceil \log q \rceil$ bits, respectively.

VI. COMPARISON

In this section, we conclude our comparison between our proposal and the only published distributed scheme. Table VIII provides a comparison between our proposed scheme and B-PCGR [5]. Note that by the *communication overhead*, we mean the number of bits that should be sent and not the traffics that are forwarded by the nodes due to the routing process. Both schemes provide t-wise backward and forward secrecy. They also provide on-demand rekeying.

[2]We assume $\lceil \log q \rceil$ bit IDs

Table VIII
COMPARISON OF GROUP REKEYING SCHEMES. $L = \lceil \log q \rceil$ IS THE KEY SIZE.

	Our Scheme	B-PCGR [5]
Attacker Model	Active	Active
On-demand rekeying	Yes	Yes
t-wise forward secrecy	Yes	Yes (for $\mu > t$)
t-wise backward secrecy	Yes	Yes (for $\mu > t$)
System Model	Distributed	Distributed
Total point-to-point communication overhead (bits)	$(\mu+1) \times (2t+1) \times L$	$n \times (\mu+1) \times L$
Broadcast communication overhead (bits)	$(5t+2) \times L$	none
Computation overhead for each nodes	$O(t^2) + \frac{\mu+1}{n} O(\lambda^2)$ Modular arithmetic operation	$O(\mu^3 + (n+1) \times t^2)$ modular arithmetic operation
Computation overhead for the group controller	$O(\mu^3 + t^2)$ Modular arithmetic	none
Storage overhead for each node (bits)	$(2t+1) \times (\lambda+1) \times L$	$(n+1)(t+1) \times L$

In order to provide these features, B-PCGR needs to have an underlying IDS with the capability to inform all group members about the compromised nodes which costs more complexity of the IDS and more communication overhead. While in our proposal, only the group controller needs to have such information.

Our scheme consumes less energy in its computations (see Section IV-B). It also have lower communication overhead (see Section IV-D).

VII. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a new distributed group rekeying scheme which does not require a secure rekeying server and is based on local collaboration of group members. We have evaluated analytically the performance and the security of this scheme.

We have also implemented our proposal for TinyOS and used Avrora to simulate the compiled binary for MICA2 motes. Energy consumption, memory usage and communication overhead have been reported. Simulation results show that comparing to the only published distributed scheme, our scheme consumes less energy in its computations and has lower communication overhead.

Most of the group rekeying schemes, including our proposed scheme, rely on one group controller. A failure in the group controller could damage the whole group. This problem is not addressed in the literature. Our future work includes the study of the impact of multiple group controllers. Choosing the optimum key size in order to minimize energy consumption of the encryption, decryption and rekeying processes is another issue that we will study in our future work.

ACKNOWLEDGMENT

The work of the third author was partially supported by the Spanish Ministry of Education and Science under Projects TEC2009-13000 and CONSOLIDER CSD2007-00004 (ARES).

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[2] D. Liu, P. Ning, and K. Sun, "Efficient self-healing group key distribution with revocation capability," in *Proceedings of the 10th ACM conference on Computer and communications security (CCS '03)*, 2003, pp. 231–240.

[3] A. Shamir, "How to share a secret," *CACM*, vol. 22, no. 11, pp. 612 – 613, 1979.

[4] M. Naor and B. Pinkas, "Efficient trace and revoke schemes," in *LNCS, Vol. 1962*, 2001, pp. 1–20.

[5] W. Zhang and G. Cao, "Group rekeying for filtering false data in sensor networks: a predistribution and local collaboration-based approach," in *Proc. of INFOCOM*, 2005, pp. 503–514.

[6] G. Danio and I. M. Savino, "An efficient key revocation protocol for wireless sensor networks," in *Proceedings of WOWMOM'06*, 2006, pp. 450–452.

[7] Y. Jiang, C. Lin, M. Shi, and X. S. Shen, "Self-healing group key distribution with time-limited node revocation for wireless sensor networks," *Ad-Hoc Networks*, vol. 5, no. 1, pp. 14–23, 2007.

[8] F. I. Khan, H. Jameel, S. M. K. Raazi, A. M. Khan, and E. N. Huh, "An efficient re-keying scheme for cluster based wireless sensor networks," in *LNCS No. 4706*, 2007, pp. 1028–1037.

[9] Y. Wang and B. Ramamurthy, "Group rekeying schemes for secure group communication in wireless sensor networks," in *Proc. of ICC'07*, 2007, pp. 3419–3424.

[10] Y. Wang, B. Ramamurthy, and X. Zou, "Keyrev: An efficient key revocation scheme for wireless sensor networks," in *IEEE International Conference on Communications (ICC '07)*, 2007, pp. 1260–1265.

[11] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "Tinyos: An operating system for sensor networks," *Ambient Intelligence*, pp. 115–148, 2005.

[12] B. L. Titzer, D. K. Lee, and J. Palsberg, "Avrora: scalable sensor network simulation with precise timing," in *Proceedings IPSN '05*, 2005, pp. 67–71.

[13] Crossbow. http://xbow.com, accessed on aug. 31, 2011.