A Template Approach for Group Key Distribution in Dynamic Ad-hoc Groups

Bruhadeshwar Bezawada[‡*] Alex X. Liu[*†] Xiaojiang Liang[*] Rui Li[§]
[*]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China,
[†]Dept. of Computer Science and Engineering, Michigan State University, USA
[‡]School of Engineering Sciences, Mahindra École Centrale, Hyderabad, India
[§]College of Computer Science and Networking Security, Dongguan University of Technology
Emails: bru@mechyd.ac.in, alexliu@cse.msu.edu, xjliang2007@gmail.com, ruili@dgut.edu.cn

***Abstract*—The ubiquitous wireless communication medium poses serious threats to the confidentiality and integrity of communication in constrained networks like Internet-of-Things (IoT), sensor nodes or ad-hoc groups of soldiers on a battlefield. Data encryption is essential for ensuring the confidentiality and integrity of the wireless network communication. We focus on the problem of group key distribution in dynamically formed groups of network nodes. One major challenge is that the neighborhood of a node is not known prior to deployment and therefore, the group key establishment protocol needs to be independent of the physical network topology. In this paper, we describe an efficient template based approach for group key establishment using only shared symmetric secrets. Our template is a logical shared secret distribution built on the network nodes prior to deployment, which ensures that any node shares a distinct set of secrets with any dynamic group of nodes and thus, is able to establish a secure group key. We illustrate our template using two secret distribution protocols: sub-set and dual one-way hash chain distributions. Compared to existing algorithms, our template approach gives flexibility to the network administrator to choose from different secret distribution protocols for fine-grained control over the security levels and performance.**

I. Introduction

A. Motivation

Rapid advances in technology and hardware have led to the development of versatile portable computing devices, which don't need robust communication infrastructure for networking and can establish ad-hoc node to node communication to remain connected to the network. Examples of such networks are: (a) Soldiers in a battlefield who communicate with their colleagues regarding battlefield updates and (b) Internet-of-Things (IoT), where any device can connect to the Internet with the assistance of other devices. These networks operate in an hostile environment using an open wireless communication medium with negligible centralized monitoring, which makes them vulnerable to attacks [11] from malicious entities. Hence, the security of such ad-hoc network communication is critical for the reliability of network applications.

This work is supported in part by the National Natural Science Foundation of China under Grant Numbers 6167215, 61370226, 61472184, 61321491 and 61272546, the National Science Foundation under Grant Number CNS-1318563, CNS-1524698, and CNS-1421407, the Jiangsu High-level Innovation and Entrepreneurship (Shuangchuang) Program, and the China Postdoctoral Science Foundation. Bruhadeshwar was visiting scholar at Nanjing University.

ISBN 978-3-901882-94-4

In this work, we focus on the problem of securing dynamic group communication among ad-hoc nodes where the nodes need not be within the transmission range of each other. The term *dynamic* group means that the group is not predefined, either by logical association or by physical location, but is decided by a sending node *"on-the-fly"*. A shared session secret, in short, *group key*, is used to encrypt all the communication exchanged within the dynamic group. Most importantly, the sending node needs to be able to exchange the group key securely using confidential channels shared with the group members. Therefore, there is a need to instantiate efficient key distribution protocols, which enable *any* node to form a dynamic secure group consisting of *any* chosen set of nodes, regardless of the underlying network conditions.

B. Problem Overview

We focus on the problem of *group key establishment* in wireless ad-hoc networks using shared secret distribution protocols. Shared secret distribution [9] offers several advantages, such as computational efficiency in session key establishment and ability to operate without an on-line controller. In these protocols, prior to deployment, the network administrator loads each node with a random set of secrets drawn from a common pool in such a way that each secret is likely to be shared by two or more nodes. We use the generic term *"secret"* to avoid placing any restrictions on the length or format of the secrets, but assume that it is a sufficiently long random string.

Now, to demonstrate the technical challenges in this problem, we consider two simple shared secret distributions. For a network of n nodes, the total number of possible groups is, $O(2^n)$, and each node is part of $O(2^{\frac{n}{2}})$ groups. A naïve secret distribution protocol is to store $O(2^{\frac{n}{2}})$ distinct secrets at each node, *i.e.*, one secret for each group to which the node belongs to. This solution allows any sending node to establish a group key using only one encrypted message, regardless of the size of the group. However, this communication efficiency is offset by the unreasonable storage at each node and furthermore, several of these secrets may never be utilized. Another solution is to deploy a pair-wise [2], [9] symmetric secret protocol where every distinct pair of nodes share a distinct secret. But this solution requires a sender to individually encrypt and send the group key to each group member, *i.e.*, $O(|G|)$ computational and communication complexity, which is high even for moderately large groups. Furthermore, adding a new network node incurs a costly $O(n)$ pair-wise key distribution.

From these two examples, we note that, it is possible to balance storage and communication if a sender pre-shared a small set of auxiliary secrets with some group members. These additional secrets not only reduce the cost of establishing the group key, but also keep the storage cost at the nodes manageable. But, a major hurdle in this approach is that, in dynamically formed groups, it is difficult to predict the group member identities with non-negligible probability. Therefore, the design of shared secret distribution protocols, which achieve an ideal balance of storage and communication for dynamic group key establishment, is a challenging problem.

C. Limitation of Prior Art

Previous works in this domain have considered this problem under two different system and network models. In the first model [7], [14], [17]–[20], [25], a centralized group controller is assumed to be in charge of establishing the necessary group key and for distributing the group key securely to the group members. However, this model is unrealistic for ad-hoc networks due to two important concerns. First, the centralized group controller needs to be on-line at all times to perform the group keying, which is a central point of failure for the protocol. Second, the span of the ad-hoc group is assumed to be limited to a geographic region, which is not always possible, especially for ad-hoc networks distributed across a wide region.

In the second model, solutions in [3], [5], [6], [10] describe collaborative group key agreement protocols to enable multiple parties to agree upon a common group key. While group key agreement protocols are distributed in nature and avoid the issues of centralized key management, they suffer from other drawbacks. First, the group members need to exchange several messages before agreeing upon the group key, which is a considerable burden on the ad-hoc nodes operating in unreliable network conditions. Second, each group member performs computationally expensive exponentiation operations proportional to the size of the group, which makes these solutions impractical for large groups.

D. Proposed Approach

Our approach is based on two important building blocks: *a logical template* –which creates full logical connectivity among the ad-hoc nodes, and *shared secret distribution* – which ensures that a sender can establish secure channels for group key establishment. The template is a *logical* hierarchical shared secret structure overlaying the ad-hoc network nodes independent of their physical deployment. A logical channel corresponds to the ability of two nodes to communicate securely even if they are not in each other's vicinity.

At the highest level, the network administrator arranges the nodes into two mutually exclusive partitions and performs shared secret distribution by treating each partition as a single logical entity. To secure the communication for a given two sibling partitions, the network administrator loads each node with two types of secrets: a *group shared* secret, to exchange group keys among a group of nodes, typically, belonging to the same partition and a *pair-wise* secret, to exchange group keys between two nodes, typically, belonging to different partitions. At system initialization, the administrator associates a distinct master secret for each partition and distributes this secret to all nodes in that partition. Furthermore, for each node in a partition, the administrator computes a *blinded* secret using the node's unique identifier and the master secret of the sibling partition, and gives this value to the node. Therefore, any two nodes, where each belongs to a different partition, can use this common secret information to establish a pair-wise session key. Next, to distribute group shared secret in a partition, the administrator computes a blinded secret from the master secret of the other partition and gives this value to all nodes in the partition. This group shared secret can be used by a node in one partition to broadcast a message securely to nodes in the sibling partition. Proceeding thus, the basic template can be recursively applied, by splitting each higher level partition into two lower level mutually distinct sibling partitions, to achieve efficient group key establishment among any group of nodes in the network. By construction, the template can effectively reduce the cost of group key establishment by logically securing groups of nodes in a structured manner and still keep the storage at nodes small.

E. Technical Challenges and Solutions

The first challenge is if the group of receivers are sparsely distributed in the template, then the sender may need to perform $O(N)$ encryptions to establish the group key. To address this, we distribute additional shared secrets within a partition so that the nodes share additional secrets. But, at the same time, distributing additional secrets can increase node storage and it is desirable to keep the key storage as low as possible.

Towards this, we describe two storage efficient key distribution protocols to enable a high level of secret sharing among the keys while keeping the storage cost poly-logarithmic in network size. In the first protocol, called the *sub-set keying* scheme, the administrator distributes a pool of secrets to each partition instead of one single group shared secret. For each distinct node in the other partition, the administrator distributes a unique sub-set of keys from this pool of keys. This key distribution has all the properties of our original template with the advantage that the nodes within a single partition share additional common secrets. In the second protocol, called the *one-way hash chain* scheme, the administrator instantiates two one-way hash chains for each partition. The nodes in a partition are ordered in a logical or numerical sequence. Based on the node position in the ordering, each distinct node in a partition receives one unique hash value from each of the one-way hash chains belonging to the other partition. The distribution is performed in such a way that *combination* of these two hash chain values is known only to one node in the partition.

The second challenge is due to collusion among nodes, *i.e.*, two or more nodes can share their secret information and try to compromise the communication of other nodes in the network. This is due to the deterministic sharing patterns of the secrets among nodes, which can help a colluding node in determining an ideal set of colluding nodes. To address this, we instantiate another template on the nodes, which is a randomized version of the original template and diffuses the secret sharing patterns among the nodes. For secure communication, a combination of secrets from both templates are used and provide higher collusion resistance. We show that our collusion resistance solution does not increase the storage at the nodes substantially and provides higher collusion resistance as well.

Our Contributions. (a) We describe an efficient template based approach to secure dynamic group communication in ad-hoc networks, which does not require the presence of a dynamic group controller. Our protocols achieve the dual goals of confidentiality and integrity of communication without additional storage. Our template approach provides a useful place-holder for secure key distributions that exist in literature as well as those that might be proposed in future. Any such key distribution protocol can be easily "plugged-in" depending on the network administrator's choice, system storage and computational trade-offs that can be tolerated. (b) We describe two instantiations of our template: *sub-set keying* and *dual one-way hash chain* schemes, which offer varying trade-offs in storage and computation. These instantiations validate the utility of our template based approach and show the feasibility of further viable instantiations. (c) We describe an algorithmic approach to strengthen the collusion resistance of our key distribution protocols. Our collusion resistance algorithm is simple to implement and improves the collusion resistance of the network without increasing the associated overheads substantially.

II. Preliminaries

We describe the system model and threat model in this section. Without loss of generality, we use the terms "nodes" and "users" interchangeably.

A. System Model

We consider that the ad hoc nodes are deployed in an open wireless environment without no or limited networking infrastructure in place and the nodes are not constrained with respect to mobility or region. All the nodes have equal computational ability and have the capability to perform cryptographic operations. Any node can communicate with one or more nodes where the network communication is single hop or multi-hop and each ad-hoc node acts as a receiver of the message or as a router forwarding the data. A network administrator is responsible for the initial configuration of the shared secret distribution protocol and deployment of the nodes, but has a limited role after the network is deployed.

B. Threat Model

The ad-hoc node communication is vulnerable to eavesdropping and tampering by external adversaries. We differentiate between two kinds of attacks: compromise and collusion. A compromise attack happens when a node is captured by external adversaries. In such attacks, the secrets held by the mobile device might be protected by a tamper proof mechanism or by other stronger authentication approaches and these devices may not be useful to the adversary. A collusion attack is when some internal malicious nodes pool their secrets and try to gain information from other nodes' communication. This internal attack is difficult to detect and the network administrator can only try preventive measures or limit the damage due to such attacks. Also, in the absence of collusion, the ad-hoc nodes assumed to be *semi-honest* in nature, *i.e.*, the nodes follow proper network operation but try to gain additional information by using whatever information at their disposal.

C. Related Work

There has been a significant amount of research in the area of secure group communication in ad-hoc networks, which can be broadly categorized into three types: conference key agreement, centralized and distributed group key establishment.

In conference key agreement protocols [3], [5], [6], all the nodes of the group contribute some secret values to generate the group key. These protocols are perfectly secure and allow for arbitrary dynamic groups. However, since all the nodes of the group need to participate in the group key generation, these protocols cannot scale well for even medium sized groups. Although efficient variations, supporting inexpensive computations, have been proposed recently [12], the communication complexity remains unchanged. Our approach does not require group key agreement as the group initiator chooses the group key and distributes it securely to the other group members.

The centralized group key management protocols [14], [17] assume the presence of a central administrator to assist the formation and maintenance of secure dynamic groups in the network. Also, these solutions assume that there is only sender for the entire network, which is usually the administrator and that there is a multicast key-tree [23] to distribute the group key. In [14], the authors maintain a physical multicast tree of the group members and use public-key certificates for group key establishment. However, this is an expensive approach as the data is encrypted on a per link basis and is not secure in the model considered in our work. In [17], the authors propose an approach to align the physical topology to the key tree for reducing cost of group key distribution. Furthermore, some works [7], [25] have focused on achieving reliability for the group key distribution process as ad-hoc networks are inherently unreliable. The work in [19] considers the problem of dynamic ad-hoc group management within a well-defined geographic region. This work allows arbitrary nodes to form a group, but still requires the group controller to establish the group key. Our key distribution protocols do not require an online dynamic group controller for managing keys. Once the administrator deploys the group, the nodes need not contact the administrator for any key distribution.

In the distributed group key establishment setting, the work in [24] considers the problem addressed in our paper. This work considers dynamic group formation and pre-distributes secrets to achieve group key establishment. The assumption is that the nodes are assigned to random groups before deployment. The nodes collaborate at run-time to generate the group keys. However, this approach is inefficient as it requires a collaborative effort from neighboring nodes and hence, cannot support arbitrarily large groups. A good survey of key management issues in ad-hoc networks may be found in [1], [8], [11], [13], [21], [22]. Our key distribution protocols support arbitrary group formation and do not place restrictions on the location or neighborhood of a given node.

III. Our Basic Template for Group Key Establishment

Consider two groups of users, X and Y. Let users in Y be named $y_1, y_2, \cdots, y_n$ and the users in X named $x_1, x_2, \cdots, x_m$. We proceed as follows: we create a secret, say s, and a large random value R_x, and give them to every user in X. Now, user y_j from Y gets the secret $f(s, j)$ where f is a one-way function, *i.e.*, it is not possible for y_j to extract s from $f(s, j)$. In addition, all users $y \in Y$, also receive $f(s, R_x)$ from X. We call this scheme as $hs(X, Y)$. The secret s is called the master secret.

Theorem 1. Any message encrypted by $f(s, j)$ can be decrypted only by y_j or any user in X.

Proof. The secret $f(s, j)$ is given to y_j only and therefore, only $y_j \in Y$ can decrypt a message encrypted with $f(s, j)$. Also, since s is known to every user in X, any user in X can generate this secret and decrypt any messages encrypted with this secret. □

Now, we use this scheme for secure group key establishment between arbitrary users. Towards this end, consider the case where the system users are divided into two parts: X_1 and X_2, i.e., $X_1 \cap X_2 = \phi$ and $X_1 \cup X_2 =$set of all users. Apply the scheme $hs(X_1, X_2)$ and $hs(X_2, X_1)$. Let s_1 and s_2 be the respective master secrets used above and let R_1 and R_2 be the respective random secret values.

Theorem 2. Given two users, $x_j \in X_1$ and $x_k \in X_2$, they can communicate securely using secrets $f(s_2, j) XOR f(s_1, k)$ and establish a secure group key.

Proof. Based on Theorem 1, users that can generate $f(s_2, j)$ are those in X_2 and x_j. Likewise, users that can generate $f(s_1, k)$ are those in X_1 and x_k. Thus, x_j and x_k are the only users that can generate both the secrets. □

Theorem 3. Given user $x_j \in X_1$ is a group sender and all the users in X_2 are the group receivers, then x_j can use the above scheme to securely establish the group key.

Proof. Based on Theorem 1, users that can generate $f(s_2, j)$ are those in X_2, and the user x_j knows this secret by construction. Also, users in X_2 receive $f(s_1, R_1)$, and x_j can generate this secret. Thus, x_j and the users in X_2 are the only users that can generate both the secrets. □

The above scheme maintains three secrets per user. It allows a user in X_1 to communicate with user in X_2 and vice versa. However, it does not allow two users in X_1 (respectively, X_2) to communicate with each other. This problem can be solved recursively by applying the same algorithm for X_1 and X_2 independently. For instance X_1 is split into two mutually disjoint partitions X_{11} and X_{12} such that $|X_{11}| = |X_{12}| = \lceil |X_1|/2 \rceil$. The $hs(X, Y)$ scheme is instantiated on X_{11} and X_{12} and hence, secures the communication among nodes in X_1. Proceeding recursively, the leaf nodes of the hierarchy contain partitions with individual users.

Theorem 4. The number of secrets in the recursive scheme is $3 \log N$.

Proof. At each level, a user needs to store two keys. One key is its local key, say $\langle s_1, R_1 \rangle$, with which it generates keys for the other partition of users. Another two keys, say, $f(s_2, j)$ and $f(s_2, R_2)$, which the user gets from the other partition. As there are $\log N$ levels, the total storage is only $3 \log N$ keys. □

Theorem 5. The above scheme allows any sender to securely establish a group key with any dynamic ad-hoc group.

Proof. There are two scenarios possible for a sender: (a) the sender and group receivers are in the same partition, and (b) the sender and group receivers are in different partitions. If the sender and the receivers are in the same partition, the sender uses the secrets at the lower layers of the template, which provide security between the sender and the receivers, to establish the group key. If the sender and receivers are in different partition, then the sender uses the pair-wise secrets to establish the group. In a special case, if all the receivers are in the other partition, then the cost of group key establishment is optimal: *i.e.*, one message and one encryption. Any other scenario can be composed as a combination of these two scenarios. □

We have shown that it is possible to achieve group key establishment in dynamic ad-hoc groups with our template based shared secret distribution. However, the worst case complexity of group key establishment is $O(N)$ encryptions and messages, especially when the group size is between $\frac{N}{2}$ and $\frac{3N}{4}$. In the following, we demonstrate that the template structure can lead to more efficient shared secret distributions, which support communication efficient group key establishment. Now, we show how this secret distribution can be used for group key establishment between these two partitions.

Dynamic Addition of Nodes. Our approach supports dynamic addition of new nodes in a straightforward manner. We assume that the perceived additions are bounded within a percentage of total population and we instantiate the protocol by leaving certain *vacancies* in the logical partitions. When a node is to be added, the node is first placed in a vacancy at the lowest level of the template and the corresponding keys are loaded. Proceeding recursively, the node moves up the hierarchy into its corresponding vacancies and is loaded with all secrets according to its position. Most importantly, with this approach, the key distribution in the existing nodes remains unchanged and the new node can establish secure groups with any subset of existing nodes.

IV. Sub-set Secret Distribution

Our sub-set secret distribution is an enhanced version of the scheme from [4], which considers one sender and a set of receivers. The sender chooses a pool of k distinct secrets and each receiver is given a unique sub-set of l secrets from this pool. By construction, no two users share identical subsets of the secrets. We instantiate this basic scheme on our template to construct an efficient secret distribution protocol for group key establishment. In [4], the authors show that for a group size of n, a sender maintains only $k = \log n + 1/2 \log \log n + 1$ secrets and each user receives a unique subset of $|k|/2$ secrets.

Now, consider two partitions S and R as shown in Figure 1. The users in the S-partition receive a *pool* of secrets $\{K_1, K_2, K_3, K_4, s\}$ and the users in R-partition receive $\{C_1, C_2, C_3, C_4, r\}$. The random values s and r are used to blind the secrets before they are given to the users in the other partition. To load the secrets to the users in partition R, the administrator generates blinded values of the secrets held by the S-partition as follows: $hK_i = \mathcal{HMAC}(K_i, s) \forall K_i \in S$ where $\mathcal{HMAC}$ is a secure one-way hash function. Each user in R receives two sets of secrets: one set for secure pair-wise communication and other for secure group communication. For secure pair-wise communication, each user receives a unique sub-set of the blinded secrets, e.g., R_1 receives hK_1, hK_2. This distribution ensures that the combination of these two secrets is known only to R_1 and users in S, but not known to any user in R. For secure group communication, R_1 receives XOR value of the secrets that are not part of the unique subset, e.g., R_1

receives $hK_3 \oplus hK_4$. This distribution allows $R1$ to generate a shared group secret as follows: $hC_1 \oplus hC_2 \oplus hC_3 \oplus hC_4$ combined with secrets hK_1 and hK_2 received from S. This secret can be used by R_1 to send a message securely to all users in S. Furthermore, due to the properties of XOR, this secret can be generated by all the users in R-partition with their respective secrets.

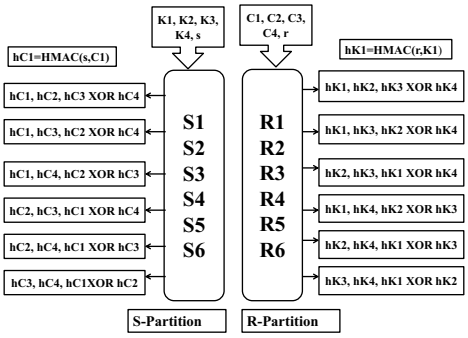


Fig. 1: Sub-set Secret Distribution

Now, we show how this secret distribution can be used for group key establishment between these two partitions. As an example, we let $S_1 \in S$ be the sender of the group and choose three different group membership instances. In the first group instance, only one user R_1, from R-partition, is a member of the group. In this case, S_1 uses a combination of the secrets: $hC_1 \oplus hC_2$, which are known to R_1, and $hK_1 \oplus hK_2$, which are loaded, to generate a unique session traffic encryption key as follows:

T_k= $hC_1 \oplus hC_2 \oplus hK_1 \oplus hK_2$

Next, S_1 generates a session group key k_g and encrypts k_g with T_k using a symmetric encryption algorithm ENC, such as AES, and transmits the encrypted message $ENC(T_k, k_g)$ to R_1 after attaching the necessary meta-data. For message integrity, using T_k, S_1 can add message authentication code (MAC) to authenticate the message as follows: $\mathcal{HMAC}(T_k, k_g)$. The cost of the group key establishment with authentication is one encryption, one signature, and one message transmission.

In the second group instance example, several users from R-partition are chosen by S_1 as group members. For instance, R_4, R_5, R_6 are chosen as the group members. To establish the group key, S_1 first determines the *minimum* set of keys that are shared among these nodes. This problem is known as the key-cover problem, as identified in [23], and reduces to the general set-cover problem, which is known to be NP-complete. We use a greedy heuristic algorithm to solve this problem as follows. We create a list of the secrets and for each secret we list the users sharing that secret. Now, we select the secrets that are shared by the maximum number of the group users. In this case, hK_4 is shared among all the nodes and is not known to the other members in R. Therefore, S_1 generates the traffic encryption key T_k by using XOR to combine hK_4, hC_1 and hC_2 and uses T_k to transmit the group key k_g. The algorithm is shown in Algorithm 1.

While the cost of group key establishment is only one, the cost of authenticating the message is higher in this scenario. The sender S_1 needs to generate multiple MACs because it is essential that each receiver should be able to verify independently the source of the messages transmitted. For instance, if hK_4 were to be used to sign the message, then it is not possible to distinguish between S_1 and the receivers as any of the parties could have generated the signature. The sender follows the authentication approach of [15], in which the sender generates multiple MACs for the same message using different secrets and each receiver can verify a sub-set of these signatures with the secrets in possession. Therefore, the sender generates the following MAC secrets, which are used to sign the group key messages: $hK_1 \oplus hC_1 \oplus hC_2$; $hK_2 \oplus hC_1 \oplus hC_2$; $hK3 \oplus hC_1 \oplus hC_2$; and $hK_4 \oplus hC_1 \oplus hC_2$. The cost of authentication is therefore, $O(\log |R|)$, as this is the number of secrets held by the sender for a receiver set R.

In the final group instance, all users in partition R are the group members. The sender S_1 generates the traffic encryption secret T_k by computing XOR of the values, $hC_3 \oplus hC_4$, received from R, and $hK_1 \oplus hK_2 \oplus hK_3 \oplus hK_4$, which can be generated by all the users in R. Note that, the combination of these secrets is known only to S_1 and no other user in S knows these values. Also, the sender generates signatures as in the previous scenarios, *i.e.*, the sender S_1 generates $\mathcal{HMAC}$ signatures on the message with all the secrets in its possession.

Algorithm 1: Key Selection Algorithm

Input: $R = \{R_1, R_2, \cdots, R_N\}$
Output: KC : Set of Keys Covering R
Let $Keys(R)$ denote set of keys known to R;
Let $Users(K) := \{R_i | K \in Keys(R_i)\}$; *#set of users knowing K*
KC:=ϕ;
while R *is not empty* **do**
  $KC = KC \cup \{K_i | Max(|Users(K_i)| \forall K_i \in Keys(R))\}$; *#greedy Max function selects the key that covers maximum number of receivers*
  $R = R - Users(K_i)$; *#update receiver set*
return KC;

The above group variations solve the problem of group key establishment if the sender and receivers are in different partitions. But if S_1 wishes to transmit the message to another user in S then the above scheme is not sufficient as the users in S do not share any unique secrets. For this case, we partition S in two equal sized mutually disjoint sets, S_a, S_b and perform the secret distribution for each partition using the above scheme. Furthermore, this partitioning is repeated until at the leaf node only the individual users remain. As any group size can be decomposed into a variation of the above mentioned group instances, the sub-set secret distribution enables secure group key establishment for dynamic ad-hoc groups.

The number of secrets stored at each user is as follows. Assuming that there are n total users in the network, the initial partitions consist of $n/2$ nodes each. A user stores $\log \frac{n}{2} + \frac{1}{2} \log \log \frac{n}{2} + 1$ secrets for the partition it belongs to and $\frac{1}{2}(\log \frac{n}{2} + \frac{1}{2} \log \log \frac{n}{2} + 1)$ secrets received from the other partition. Proceeding further, the total number of secrets stored at each user is: $\frac{3}{2}((\log n - 1)(\frac{\log n + 4}{2}) + \log \log n)$.

V. Dual One-way Hash Chain Secret Distribution

Our second template instantiation is based on the use of one-way hash chains first described by Leslie Lamport [16] for authentication. A one-way hash chain is generated from a random seed s by repeated application of a hash function on the seed as follows: $h(s), h(h(s)), h(h(h(s)))$ and so on, where h is a one-way hash function such as $\mathcal{HMAC}$. The notation, $h^t(s)$ denotes that the one-way hash function h has been applied t times over the random seed s. The important property of one-way hash chains is that one can efficiently compute $h^{t+1}(s), h^{t+2}(s), \cdots$, from $h^t(s)$, but it is infeasible to compute $h^{t-1}(s), h^{t-2}, \cdots$ from $h^t(s)$.

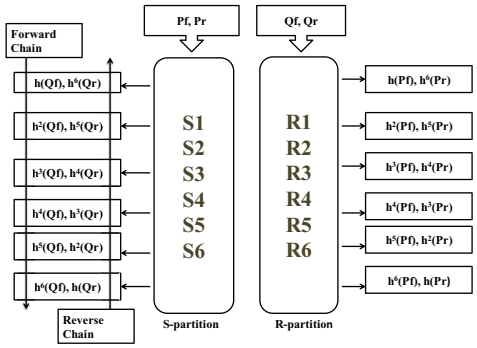


Fig. 2: Dual One-way Hash-Chain Distribution

To describe our secret distribution, we again consider the two partitions S and R consisting of users as shown in Figure 2. To the users in a given partition, say S, the administrator distributes two random seed values, P_f and P_r. Now, the administrator instantiates two one-way hash chains of the form $h(P_f), h^2(P_f), \cdots, h^{|R|}(P_f)$ called the *forward* chain and $h(P_r), h^2(P_r), \cdots, h^{|R|}(P_r)$, called the *reverse* chain, each of length $|R|$. Note that any user in S-partition can generate these values at run-time and need not store them explicitly. A user $R_i \in R$ receives one hash value from each of these chains, such that the combination of the hash values is only known to this user. From the forward chain R_i receives $h^i(P_f)$ and from the reverse chain R_i receives $h^{|R|-i+1}(P_r)$. Based on the secret distribution and property of the one-way hash chains, no other user in R knows both these secrets and neither can any user generate these values. For instance, user R_4, knowing the secret $h^4(P_f)$, cannot derive the secret $h^3(P_f)$ known to R_3 as it is not possible to generate $h^3(.)$ using $h^4(.)$. Vice-versa, R_3 cannot use the loaded secret $h^4(P_r)$ to generate the secret $h^3(P_r)$ known to R_4. The complete distribution of secrets is shown in Figure 2. Now, we show how this secret distribution enables a sender, say $S_1 \in S$, to achieve secure group key establishment in the three types of group instances discussed previously.

In the first group instance type, the group consists of one user from the R-partition, say R_1. The sender S_1 generates the secrets $h(P_f), h^6(P_r)$ and combines them with $h(Q_f), h^6(Q_r)$ received from the R-partition, to form the traffic encryption key T_k as follows: $h(P_f) \oplus h^6(P_r) \oplus h(Q_f) \oplus h^6(Q_r)$. The T_k resulting from above is known only to S_1 and R_1 and no other user in either partitions. Next, to generate the MAC secret, S_1 combines the hash values it received from R with the secrets given to R_1 as follows: $h(Q_f) \oplus h(P_f) \oplus h^6(P_r)$ and $h^6(Q_r) \oplus h(P_f) \oplus h^6(P_r)$ and generates the $\mathcal{HMAC}$ signatures with both these secrets. This is an essential step for authentication because of the nature of one-way hash chains, any other user in S can also generate the individual hash values that can be generated by S_1, but at the same time, any other user in S cannot know the values received by S_1 from R. Therefore, the cost of group key establishment is one encryption, two signatures and one message transmission.

In the second group instance, several users from R-partition are group members. For instance, R_1, R_2, R_3 are chosen as the group members. To generate the traffic encryption secret T_k, the sender S_1 computes the secrets known by these users. In this scenario, since the logical identifiers of the users are known to S_1, the identifiers of the secrets known to these users are known to S_1 as well. Since the users have consecutive identifiers, S_1 determines that all these users either know or can generate the secret: $h^3(P_f)$ and $h^6(P_r)$. For instance, R_2 knows $h^5(P_r)$ and can compute $h^6(P_r)$ and so on. Therefore, S_1 generates the traffic encryption secret by combining these common secrets with the hash values it received from R as shown, to generate the T_k: $h^3(P_f) \oplus h^6(P_r) \oplus h(Q_f) \oplus h^6(Q_r)$. This instance shows that one-way hash chains offer more opportunities for optimization and this is validated through the experimental results as well. However, these optimizations do have a tradeoff in terms of the user computation, which is only a few additional fast hashing operations. The message authentication is similar to the previous group instance, *i.e.*, S_1 combines the common secrets, used in the traffic encryption key generation, with each of the secrets it received from R to generate the MAC secret. The MAC secrets generated are: $h(Q_f) \oplus h^3(P_f) \oplus h^6(P_r)$ and $h^3(P_f) \oplus h^6(P_r) \oplus h^6(Q_r)$. Thus, the cost of group key establishment is: one encryption, two signatures and one transmission. The worst case scenario occurs when the receiver size is $\frac{R}{2}$ and none of the identifiers are consecutively numbered, *i.e.*, the identifiers are R_1, R_3, R_5 and so on. In this scenario, the sender needs to securely unicast the group key to each of these users and therefore, the group key establishment cost is $O(|R|)$ in the worst case. In practice, however, the probability of such a situation is very low.

In the third and final group instance, all the users in R are selected as the group users. For this scenario, the sender S_1 generates the traffic encryption key by combining two types of secrets: (a) a secret that can be generated by all users in R from their pre-loaded values and (b) the secrets received from R. We note that, all the users in R can generate the secrets: $h^6(P_f)$ and $h^6(P_r)$ from their pre-loaded values. Therefore, the traffic encryption key T_k is generated as follows: $h^6(P_f) \oplus h^6(P_r) \oplus h(Q_f) \oplus h^6(Q_r)$. Next, for generating the signatures on this message, S_1 uses $h(Q_f) \oplus h^6(P_f) \oplus h^6(P_r)$ and $h^6(Q_r) \oplus h^6(P_f) \oplus h^6(P_r)$, which is similar to the previous scenario. Thus, the cost of group key establishment for broadcast is one encryption, two signatures and one message transmission.

Finally, using the decomposition technique discussed in the previous section, the one-way hash chain secret distribution can achieve group key establishment across any dynamic ad-hoc group. The total number of secrets stored at each user can be computed as follows. Each user stores two random seeds for each partition and receives two hash values from the other partition. Since there are $\log n$ partitions, the number of secrets stored at each user is $4 \log n$. However, this approach requires that the users generate the necessary hash values at run-time

and this complexity is $O(n)$ in the worst-case. Thus, compared to the sub-set secret distribution, the one-way hash chain technique involves more computations from the user and we illustrate this impact in our experimental analysis. On the other hand, the sub-set secret distribution incurs more transmissions than those required in one-way hash chain distribution.

VI. Enhancing Collusion Resistance

The secret distribution protocols discussed can be vulnerable to user collusion, *i.e.*, two or more users combining their secrets to compromise the communication of other users. In this section, we describe a collusion resistance scheme to handle this situation and reduce the impact of collusion.

To highlight the tradeoffs involved, we note that, perfect collusion resistance is achieved only if each user shares a unique secret with every other user. In this secret distribution, a sender needs to send as many messages as the size of the dynamic group to establish the group key. Since our protocols offer avenues for optimizing this cost, we focus on enhancing collusion resistance. We only consider collusion attacks, *i.e.*, insider attacks and not compromise attacks, in which an external adversary captures the nodes. We surmise that the hardware is sufficiently tamper proof and an adversary cannot really gain much information by collecting random devices. In a collusion attack, two or more users combine their secrets and try to compromise the secure channels used by a sender for group key establishment. An attack is successful if the sender generates a traffic encryption key using secrets that are found in the combined pool of secrets of the colluding users.

In our protocols, the key vulnerability is that the secrets distributed to the users are based on a deterministic protocol and it is possible that the user identifiers provide useful information to a colluding adversary to identify suitable targets. Thus, to prevent collusion, we use two intuitions: (a) first, increase the number of secrets with the users and, (b) second, randomize the distribution of these additional secrets. Based on these two ideas we describe an approach to distribute additional secrets to the users in a nearly random manner. The effect of such a secret distribution is that the adversary needs to select many target users for effective collusion attack and such efforts are eventually detected by other nodes. To describe our approach, we consider two partitions of users A and B as shown in Figure 3, which have been secured by one of the secret distributions. The total number of ad-hoc nodes in the group is assumed to be $|A| + |B| = N$. We assume that all the sub-partitions of A and B, which are not shown in the figure, are also secured based on our template approach.

Now, in our original secret distribution, the A-partition would be decomposed and the sub-partitions will be secured and so on. But, as discussed earlier, this kind of deterministic distribution is susceptible to better collusion attacks. To prevent this, we instantiate a second secret distribution on the A and B partitions. Instead of partitioning A, we randomly select several user blocks from A, as shown in Figure 3 and exchange them with equal sized random user blocks from B. For example, we may randomly choose three user blocks A_1, A_2, A_3 from A such that $|A_1+A_2+A_3| = |A|/2 = |N|/4$, *i.e.*, we randomly select half of the users in the partition. In similar manner, we randomly choose user blocks B_1, B_2, B_3 from B. Now, we exchange these user blocks across the

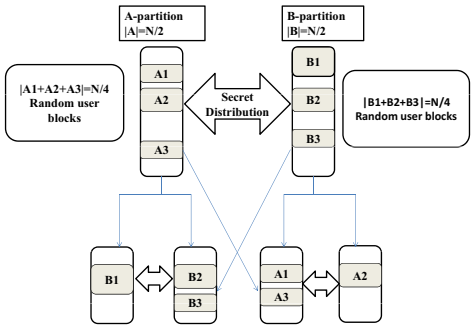


Fig. 3: Random Displacement of User Blocks A_1, A_2, A_3, B_1, B_2 and B_3

partition, *i.e.*, A_1, A_2, A_3 are placed into B and B_1, B_2, B_3 are placed into A. Finally, our secret distribution is performed on the re-arranged partitions. This approach ensures that the original set of users in A will now share a different set of secrets and the same applies to the users in B. Since the distribution of users is random, it is difficult for an adversary to try to select the best possible nodes for collusion. Furthermore, after the secret distribution at the top-level, we decompose the individual partitions into sub-partitions as shown in Figure 3. We repeat the random exchange of users for the sub-partitions and perform the secret distribution.

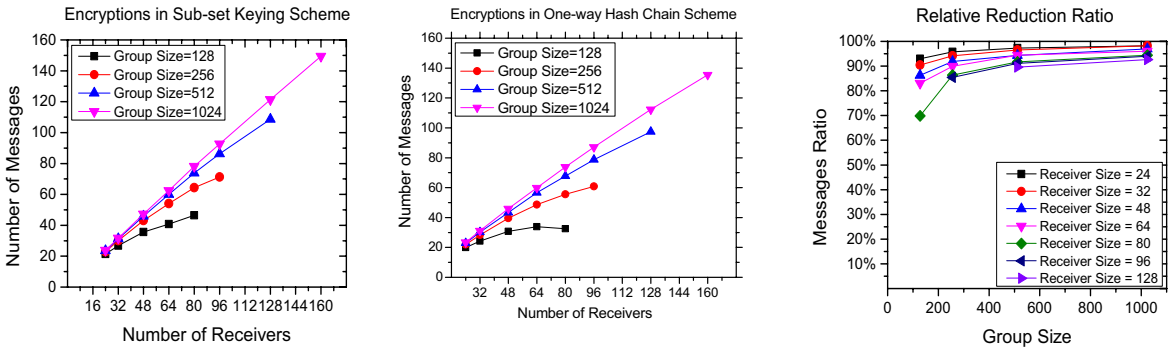
This distribution ensures the users are now randomly distributed and that several users share additional secrets that will enhance the collusion resistance as compared to the original secret distribution. The storage at the users is two times higher as they need to store secrets from both the secret distributions, but still poly-logarithmic in complexity. The algorithms for group key establishment remain the same except that a sender will apply secrets from both the distributions to send a secure message. From experimental results we show that, the cost of the group key establishment does not increase by the same amount and is slightly lesser due to the optimizations offered by the underlying secret distribution protocols.

VII. Performance Evaluation

We evaluate the performance of our protocols on different group instances focusing on the cost of group key establishment and relative tradeoffs in the two secret distribution protocols. We use the following metrics for this purpose: the number of group key establishment messages and the number of signatures by the sender. We also measure the impact of collusion on our protocols and evaluate the effectiveness of our collusion resistance mechanisms.

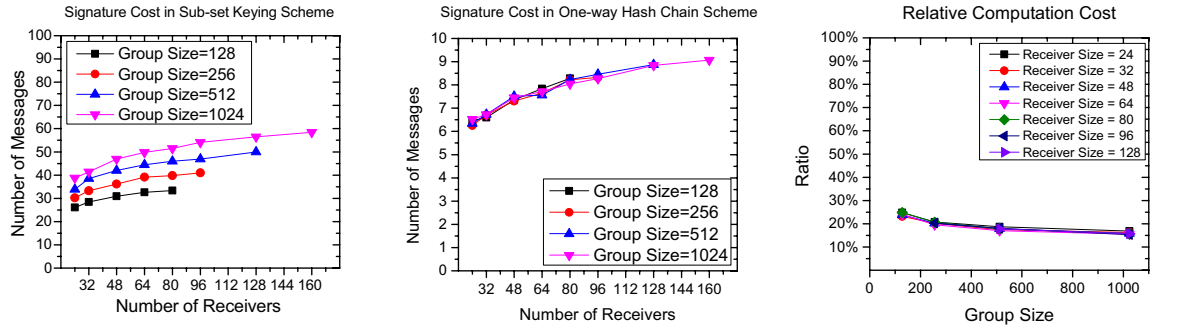
A. Experimental Methodology

We do not make any assumptions regarding the topology of the network and only assume that there are routing mechanisms in place to deliver the packets to any receiver. We focus only on the cost of the sender and the receivers during the group key establishment process. We performed each experiment by choosing a fixed network size, then electing a random sender and choosing random users as group members from this network. For our experiments, we considered the network group sizes with, 128, 256, 512 and 1024 users and groups with, 24, 32, 48, 64, 80, 96, 128 and 160 receivers. In each experiment, the sender chooses a random set of receivers to form the



(a) Sub-set Scheme (b) One-way Hash Scheme (c) Relative Communication of One-way/Sub-set Ratio

Fig. 4: Encryptions Performed in Sub-set and One-way Hash Schemes and Relative Comparison



(a) Sub-set Scheme (b) One-way Scheme (c) Relative Computation of One-way/Sub-set Ratio

Fig. 5: Signature Cost in Sub-set and One-way Hash Schemes and Relative Comparison

group and establishes the group key among these receivers. We averaged each experiment over 5 different senders and for each sender repeated the experiment over 10 trials by choosing a different set of receivers for the given sender. We focus only on the number of operations involved in achieving a specific task, e.g., the number of encryptions performed and the number of transmissions, and do not consider specific implementation details such as the encryption algorithms, key lengths or hash functions involved. Our implementation was done in Java and experiments were conducted on PC running Windows 7 operating system on a *3.4GHz Intel(R) Core(TM) i5-3570* processor.

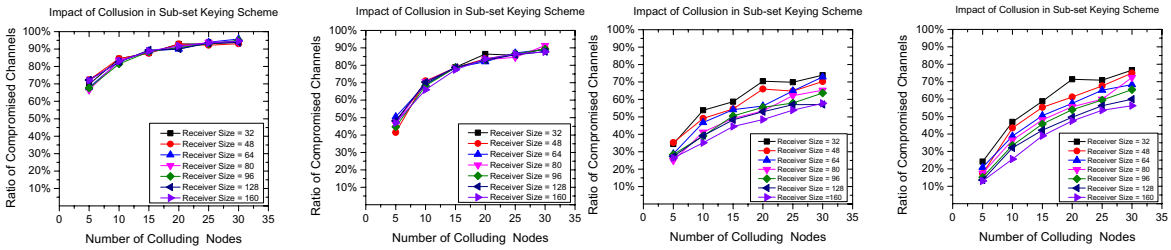
B. Group Key Establishment Cost

Our group key establishment cost reduces the number of sender messages for small and large sized groups, while keeping the cost reasonable for medium sized groups. In Figure 4, we show the number of encryptions performed by the sender in the sub-set and one-way hash chain schemes. Each unique encryption of the group key is counted as one message for this computation. From this figure, we note that, the cost is lowest if the receiver set is nearly proportional to the group size. If the group size is large and receiver set is small, the sub-set scheme, Figure 4a does not perform very well and the reduction in cost is less than 10%. The one-way hash chain scheme, Figure 4b, performs better in such cases, but requires a higher computation from the users. In Figure 4c, we show the reduction in number of messages achieved by the one-way hash chain scheme compared to the sub-set scheme.

The signature cost is shown in Figure 5, in which the one-way hash chain performs much better than the sub-set scheme. Overall both the schemes require that the number of signatures is no higher than half the group size. Figure 5c, shows that the one-way hash chain outperforms the sub-set scheme in terms of signatures.

C. Collusion Resistance Evaluation

Our collusion resistance approach reduces the impact of collusion by nearly 50%. In Figures 6, we evaluate the effect of collusion on the group key establishment process for sub-set keying scheme as the results for one-way hash chain scheme are similar. We choose the number of colluding users as: 5, 10, 15, 20, 25 and 30 users, where a larger set is likely to be detected or noticed by other nodes in the network. For this experiment, the colluding users combine their secret information and try to compromise the secure channels, both confidentiality and authentication, between different pairs of users. A *compromised channel* indicates that for any given pair of users, the secrets known to this pair of users is a subset of the secrets known to the colluding users and therefore, the communication between the pair of users is not secret. The results for the "plain" version of the protocols are shown in Figures 6a, and 6c, which show that the colluding nodes can compromise up to 90% of the secure channels. Figures 6b and 6d, labeled "Collusion Resistant", show that by adopting our collusion resistance approach we are able to reduce the impact of collusion by nearly 50% for the same scenarios.



(a) Plain Confidentiality (b) Collusion Resistant Confidentiality (c) Plain Authentication (d) Collusion Resistant Authentication

Fig. 6: Effect of Collusion and Mitigation by Collusion Resistance in Sub-set Scheme

VIII. Conclusion

We have addressed the problem of secure group key establishment for dynamic ad-hoc groups. Our major contribution is a structured template based approach to enable a sender to communicate securely with any dynamically identified group in the network. We performed two instantiations of our template and showed that these two instantiations reduce the cost of group key establishment while keeping the storage at the users manageable. Furthermore, to mitigate the effect of collusion, we have described a randomized approach to distribute additional secrets to the users. Our collusion resistance approach has been able to reduce the impact of collusion by 50%, as seen from our experimental validation. The future work in this domain is to explore further instantiations of our template in more network settings, such as the Internet-of-Things. Our structured template enables administrators to have more control on the secret distribution and storage at the users.

References

[1] David Airehrour, Jairo Gutierrez, and Sayan Kumar Ray. Secure routing for internet of things: A survey. *Journal of Network and Computer Applications*, 66:198 – 213, 2016.

[2] Amitanand S. Aiyer, Alvisi Lorenzo, and Mohammed G. Gouda. Key grids: A protocol family for assigning symmetric keys. In *IEEE ICNP*, pages 178–186, 2006.

[3] R.K. Balachandran, B. Ramamurthy, Xukai Zou, and N.V. Vinodchandran. Crtdh: an efficient key agreement scheme for secure group communications in wireless ad hoc networks. In *IEEE ICC*, pages 1123–1127, May 2005.

[4] Bruhadeshwar Bezawada and Sandeep S. Kulkarni. An optimal symmetric secret distribution of star networks. Technical Report MSU-CSE-07-196, Department of Computer Science, Michigan State University, East Lansing, Michigan, November 2007.

[5] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung. Perfectly-secure key distribution for dynamic conferences. CRYPTO '92, pages 471–486. Springer-Verlag, 1993.

[6] Mike Burmester and Yvo Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology EUROCRYPT'94*, pages 275–286. Springer, 1995.

[7] Reza Curtmola and Seny Kamara. A mechanism for communication-efficient broadcast encryption over wireless ad hoc networks. *Electron. Notes Theor. Comput. Sci.*, 171(1):57–69, April 2007.

[8] R. Di Pietro, S. Guarino, N. V. Verde, and J. Domingo-Ferrer. Review: Security in wireless ad-hoc networks - a survey. *Comput. Commun.*, 51:1–20, September 2014.

[9] Laurent Eschenauer and Virgil D Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM CCS*, pages 41–47. ACM, 2002.

[10] Mohammad Sabzinejad Farash, Muhamed Turkanovi, Saru Kumari, and Marko Hölbl. An efficient user authentication and key agreement scheme for heterogeneous wireless sensor network tailored for the internet of things environment. *Ad Hoc Networks*, 36, Part 1:152 – 176, 2016.

[11] Jorge Granjal, Edmundo Monteiro, and Jorge Sá Silva. Security for the internet of things: a survey of existing protocols and open research issues. *IEEE Communications Surveys & Tutorials*, 17(3):1294–1312, 2015.

[12] T.R. Halford, T.A. Courtade, and K.M. Chugg. Energy-efficient, secure group key agreement for ad hoc networks. In *IEEE Communications and Network Security (CNS)*, pages 181–188, Oct 2013.

[13] A.M. Hegland, E. Winjum, S.F. Mjolsnes, C. Rong, O. Kure, and P. Spilling. A survey of key management in ad hoc networks. *IEEE Communications Surveys Tutorials*, 8(3):48–66, rd 2006.

[14] T. Kaya, G. Lin, G. Noubir, and A. Yilmaz. Secure multicast groups on ad hoc networks. In *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, SASN '03, pages 94–102, New York, NY, USA, 2003. ACM.

[15] Sandeep S. Kulkarni, Bruhadeshwar Bezawada, and Mohamed Gouda. Optimal key distribution for secure communication. Technical Report MSU-CSE-07-189, Department of Computer Science, Michigan State University, East Lansing, Michigan, July 2007.

[16] Leslie Lamport. Password authentication with insecure communication. *Comm. of the ACM*, 24(11):770–772, 1981.

[17] Loukas Lazos and Radha Poovendran. Energy-aware secure multicast communication in ad-hoc networks using geographic location information. In *ICASSP*, volume 4, pages 201–204. IEEE, 2003.

[18] Albert Levi and Salim Sarimurat. Utilizing hash graphs for key distribution for mobile and replaceable interconnected sensors in the iot context. *Ad Hoc Networks*, pages 3–18, 2016.

[19] Kui Ren, Wenjing Lou, Bo Zhu, and S. Jajodia. Secure and efficient multicast in wireless sensor networks allowing ad hoc group formation. *Vehicular Technology, IEEE Transactions on*, 58(4):2018–2029, May 2009.

[20] Rodrigo Roman, Cristina Alcaraz, Javier Lopez, and Nicolas Sklavos. Key management systems for sensor networks in the context of the internet of things. *Computers & Electrical Engineering*, 37(2):147 – 159, 2011.

[21] Pitipatana Sakarindr and N. Ansari. Security services in group communications over wireless infrastructure, mobile ad hoc, and wireless sensor networks. *Wireless Communications, IEEE*, 14(5):8–20, October 2007.

[22] Aggeliki Sgora, Dimitrios D. Vergados, and P. Chatzimisios. A survey on security and privacy issues in wireless mesh networks. *Security and Communication Networks*, pages 1877–1889, 2013.

[23] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, 8(1):16–30, 2000.

[24] Wensheng Zhang, Sencun Zhu, and Guohong Cao. Predistribution and local collaboration-based group rekeying for wireless sensor networks. *Ad Hoc Netw.*, 7(6):1229–1242, August 2009.

[25] Sencun Zhu, Sanjeev Setia, Shouhuai Xu, and Sushil Jajodia. Gkmpan: An efficient group rekeying scheme for secure multicast in ad-hoc networks. *J. Comput. Secur.*, 14(4):301–325, July 2006.