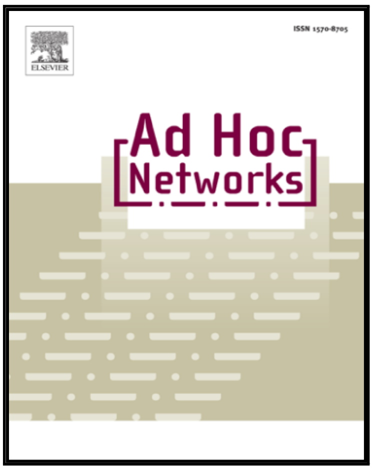
Accepted Manuscript



DKEM: Secure and Efficient Distributed Key Establishment Protocol for Wireless Mesh Networks

Duygu Karaoğlan Altop, Muhammed Ali Bingöl, Albert Levi,
Erkay Savaş

PII: S1570-8705(16)30276-1
DOI: 10.1016/j.adhoc.2016.10.001
Reference: ADHOC 1469

To appear in: *Ad Hoc Networks*

Received date: 8 January 2016
Revised date: 9 August 2016
Accepted date: 5 October 2016

Please cite this article as: Duygu Karaoğlan Altop, Muhammed Ali Bingöl, Albert Levi, Erkay Savaş, DKEM: Secure and Efficient Distributed Key Establishment Protocol for Wireless Mesh Networks, *Ad Hoc Networks* (2016), doi: 10.1016/j.adhoc.2016.10.001

DKEM: Secure and Efficient Distributed Key Establishment Protocol for Wireless Mesh Networks[☆]

Duygu Karaoğlan Altop[a], Muhammed Ali Bingöl[a,*],
Albert Levi[a], Erkay Savaş[a]

[a] *Faculty of Engineering and Natural Sciences, Sabancı University, Istanbul, TURKEY*
E-mail: {duyguk,mabingol,levi,erkays}@sabanciuniv.edu

Abstract

In this paper, we propose an efficient and secure key establishment protocol that is tailored for Wireless Mesh Networks. The protocol is based on identity-based key establishment, but without the utilization of a trusted authority for private key generation. Instead, this task is performed by the collaboration of mesh nodes; a number of users exceeding a certain threshold form a coalition to generate private keys for the network users. We performed simulative performance evaluation in order to show the effect of both the threshold value and the network size, i.e., total number of nodes, on the latency of key establishment and on the success percentage of user private key generation. Results reveal a trade-off between resiliency and efficiency; increasing the threshold value also increases the resiliency of the network, but negatively effects its latency and success percentage. For the threshold values that are smaller than 10 and for a minimum of 40 mesh nodes, at least 93% of the user private keys can be computed within at most 2 min. We also discuss the security of our protocol. We show that our protocol is secure against both outsider malicious and insider semi-honest adversaries.

Keywords: Wireless Mesh Networks, Key Establishment, Identity-based Cryptography, Threshold Secret Sharing

[☆] This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant 104E071 and part of this work is presented in Q2SWINET 2010 [1].

[*] Corresponding author

1. Introduction

Wireless Mesh Networks (WMNs) [2] are radio frequency based networks, within which wireless communication is carried out through multihop mesh routing. They consist of mesh routers and mesh clients, as depicted in Figure 1. Mesh routers, being stationary, form the backbone of the network and serve as routing devices or access points. Mesh clients, being either mobile or stationary, are often laptops, cell phones or other such wireless devices with the goal of connecting the network. They also collaborate to relay the information being transmitted by the other nodes to improve the network's coverage. The difference between these two types of nodes is not only in their mobility, but also in their energy consumption constraints. Mesh clients are known to be more limited in energy as compared to the mesh routers. Therefore, functionalities that require high computational power and bandwidth can be delegated to the mesh routers.

WMNs are dynamically self-organized, self-configured and self-healing, meaning that the network operates in a plug-and-play manner. Furthermore, they offer both low-cost and high-speed network services for the end users. Along with the ease of their deployment, WMNs provide mobility, flexibility, high robustness and increased coverage with an effective level of scalability. Hence, their utilization is favored, especially in rough and challenging terrains. Example areas of WMN usage include health and security surveillance systems, neighborhood and enterprise networking, building automation and transportation [3].

Nevertheless, WMNs are vulnerable to both passive and active attacks by the very nature of the wireless multihop communication [4]. In a WMN, a passive attack results in the violation of confidentiality, whereas an active attack further compromises authentication, integrity and non-repudiation [2]. Therefore, it is essential to build a security mechanism for the protection of the exchanged information. In order to maintain mutual trust and secure communication among the users, a key establishment service must be provided. It is generally assumed that the conventional solutions are inapplicable for WMNs due to the energy, storage and bandwidth constraints of the mesh nodes. However, recent research has shown that implementing asymmetric cryptography in resource constrained devices can actually be feasible [5, 6, 7, 8].

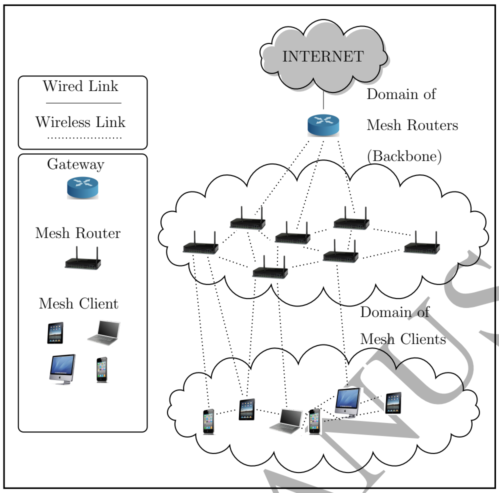


Figure 1: Infrastructure of a Wireless Mesh Network

Furthermore, the utilization of Identity-based Cryptography (IBC) in conjunction with Elliptic Curve Cryptography (ECC) found prominent usage in that context. The reasons behind this are that IBC eliminates the certificate-based public key distribution and ECC provides a similar level of security with RSA [9] for much more smaller key sizes [10, 11, 12]. Hence, EC-IBC (Elliptic Curve based IBC) reduces both the computation necessary to join the network and the utilized network bandwidth, considerably. However, the corresponding constructions assume the existence of a trusted third party (TTP), by whom the users' private keys are generated and distributed. Unfortunately, using TTP in a security protocol is usually neither rational nor practical because of the fact that such a system will be prone to single point of failures. Besides, due to the fact that WMNs are plug-and-play networks by their very nature, security should be guaranteed dynamically without any kind of third party intervention. Hence, the TTP assumption does not fit in the unique characteristics of WMNs, i.e., being dynamically self-organized, self-configured and self-healing.

In this paper, we propose a secure and efficient key establishment protocol (Distributed Key Establishment for Mesh Networks (DKEM)) that is designed to cope with the unique characteristics of WMNs. In this solution, EC-IBC is utilized to decrease the communication and computational complexities of the proposed protocol. Moreover, the role of the TTP is distributed using secret sharing in order to increase the resiliency of the network. In DKEM, the master private key of the network is distributed among the mesh nodes using both threshold secret sharing (ThSS) and additive secret sharing (AdSS) constructions. Our DKEM protocol aims at providing security only for the operational fragment of WMNs. Hence, the attack models that are specifically designed for the protocol stack of WMNs are out of the scope of this paper, and we measure the resiliency of the network only against adversaries that physically capture the mesh nodes. Performance evaluation results show that the threshold value should be increased in order to have a more resilient network at the expense of higher key establishment latency and lower success percentage of user private key generation.

1.1. Related Work

Matsumoto [13], Diffie et al. [14] and Girault [15] propose key establishment schemes that are based on the basic Diffie-Hellman (DH) key exchange protocol [16], and You and Xie [17] propose a mechanism that is based on the multi-linear DH key exchange protocol [18]. Each of these schemes adopts a different approach to bind the exchanged random values to the identities of the communicating parties. Besides, both Chatterjee et al. [19] and Shi and Gong [20] propose approaches to mutually authenticate the nodes with each other, using the well known ECC constructions. The protocol ISA [21], proposed by Li, utilizes EC-IBC to solve the secure key management problem of the WMNs. On the other hand, Zhou and Hass [22] propose a key establishment protocol that is based on the conventional public key infrastructure, in which a group of nodes share the role of the Certification Authority (CA) using ThSS. In their scheme, any k partially server-signed certificates can be used to construct a signed certificate that is identical to a CA-signed certificate. Similar approaches are proposed by Kong et al. [23], in which the RSA certificate signing key is distributed among *all the nodes* of the network, and by Dahshan and Irvine [24], in which the private key of the central authority, assumed in the ECC constructions, is distributed among *a group of nodes*. However, in [22, 23, 24], shares of the certificate signing keys are generated and distributed by the help of a TTP. Unfortunately, none of the protocols mentioned above is applicable for WMNs due to their assumptions on the existence of a trusted server.

One of the approaches to the elimination of the TTP assumption is the utilization of frequency hopping spread spectrum (FHSS). In this method, nodes rapidly switch a carrier among many frequency channels and transmit their data through the channel they are currently using. The constructions rely on the fact that the communicating parties will be on the same channel at some point of time. For instance, Strasser et al. [25] propose to use FHSS to establish a secret key in the presence of a jammer. Similarly, Miller and Vaidya [26] utilize FHSS for the establishment of the shared keys by exploiting channel diversity to create link keys for the neighbors of the nodes. Alternatively, Zan and Gruteser [27] propose a protocol, in which one of the communicating nodes stays on a randomly selected channel, while the other continuously selects channels and transmits a pre-key information until the corresponding channels match. In general, FHSS is utilized for the agreement of the pairwise keys. If the network in consideration is dense, then the number of channels should be large enough to handle collisions. Unfortunately, this negatively affects the performance of the solution.

Another approach to the elimination of the TTP assumption is the utilization of secret sharing [4, 28, 29]. For instance, Khalili et al. [4] and Deng et al. [28] propose to use EC-IBC together with ThSS to manage the cryptographic keys within wireless ad hoc networks (WAhNs). These works offer a collaborative generation of both the secret and its shares, without assuming any trusted authority. As a result, they enable flexible, efficient and fully distributed key management for WAhNs. Our DKEM solution is based on the proposal of Deng et al. [28], which is described in detail as the baseline protocol in the following subsection.

1.2. The Baseline Protocol

The baseline protocol has two phases: (i) *distributed key generation phase*, and (ii) *identity-based authentication phase*. In the former, nodes collaboratively generate the master key of the network and construct their private keys, while in the latter, they authenticate each other and secure their communication using the keys computed in the first phase.

At the beginning of the first phase, users collaboratively generate their shares of the master private key. Then, they compute their master public key shares by simply multiplying their master private key shares with the generator of the elliptic curve group, and publish them. As soon as a user receives sufficient number of master public key shares, it reconstructs the master public key of the network. Thereafter, each user broadcasts a request message on the shares of its own private key. Users, receiving such a request message, compute the corresponding shares by multiplying their own master private key shares with the public key of the requester, and transmit them. When the requesting user receives sufficient number of shares, it reconstructs its own private key.

1.3. Disadvantages of the Baseline Protocol and Roadmap to DKEM

Although the baseline protocol is fully distributed, it has two crucial drawbacks: (i) large transmission delays, and (ii) the protocol suffers from security vulnerabilities. First, the number of users contributing to the master key generation process directly affects the utilized network bandwidth. If we assume that n users are in collaboration for the master private key share computations, then at least $(n \times (n-1))$ packets will be transmitted among them. Thereafter, these users will publish their master public key shares, which requires another n packets to be transmitted. Second, the protocol is not secure against an insider attacker. An insider attacker can obtain the master private key shares of another node and more importantly, can recover the master private key of the system (details in Section 3). Since the attacker is able obtain the secret keys without physically capturing or compromising any user, the baseline protocol is also not resilient.

In DKEM, we address the aforementioned deficiencies of the baseline protocol by exploiting node hierarchy with respect to both the limitations of the mesh clients and the provided security level. Since the mesh routers can be distinguished from the mesh clients by the parameters they hold and by the operations they perform, we can impose the burden of the master key generation process on the mesh routers. Moreover, due to the fact that neither the master public key nor its shares are used among the nodes, we can eliminate the related computations and communications. These modifications reduce both the number of nodes present in the master key generation process and the number of packets transmitted during this process. In addition, we assume that it is harder to compromise the mesh routers than compromising the mesh clients. With this assumption, we can increase the threshold required in a reconstruction process by increasing the number of shares that the mesh routers hold. Consequently, the resiliency of the system increases without increasing the number of required neighboring nodes as opposed to the baseline protocol, which is shown to be non-resilient (details in Section 3).

The rest of this paper is organized as follows. Section 2 elaborates on DKEM. In Section 3, we analyze the security of both the baseline protocol and our DKEM protocol. In Section 4, we explore to what extent the network is resilient. Section 5 provides the communication, computational and energy complexities of our DKEM solution and Section 6 evaluates its performance. Finally, in Section 7, we conclude the paper.

2. DKEM: Distributed Key Establishment Protocol for Wireless Mesh Networks

We propose Distributed Key Establishment for Mesh Networks (DKEM), a secure and efficient key establishment protocol that is specifically designed for WMNs. In the following subsections, we introduce the assumptions of our DKEM solution, describe its methodology and present the *repeat request after timeout* method, which is proposed to improve the performance of our DKEM protocol.

2.1. Assumptions of DKEM

First of all, we assume that *the mesh nodes can misbehave or collude with each other to reveal the private key of any other mesh node only when they are under attack; if there is no attack, we expect proper behavior.* DKEM does not rely on the existence of a trusted authority and there is no predefined mutual trust among the mesh nodes in the sense that a particular node cannot generate the private key of any other node directly. All of the keys are generated collaboratively by the mesh routers and distributed accordingly to the mesh clients. Here, we assume that the mesh nodes will not misbehave on their own or conspire with either of the parties unless they are captured by an adversary. Hence, we measure the resiliency of the network only against adversaries that physically capture the mesh nodes.

Secondly, we assume that *it is harder to compromise the mesh routers than compromising the mesh clients.* Due to the fact that the mesh routers form the backbone of WMNs, they should be deployed in such a way that they cover the network area, so as to maintain continuous connectivity. In other words, mesh routers must be carefully placed according to a plan. At this point, we assume that the physical locations of the mesh routers are selected in such a way that the physical capture of these nodes becomes hard. For example, mesh routers can be placed at the top of the street lamps or at the roofs of the properties, where physical capture requires an effort that is equal to break-in.

Thirdly, we assume that *during the key establishment, mesh nodes are authenticated with each other and each can establish an authenticated tunnel with any of the mesh routers.* This assumption is also required by the baseline protocol [28]. In the baseline protocol, it is assumed that "*When a new node joins a network, it presents its identity, self-generated temporary public key,* *and some other required physical proof (depending on key issuing policy). To make sure the generated shares are securely transmitted, the requesting node may also present its self-generated temporary public key when sending request.*". This assumption is relevant in order to prevent the trivial impersonation and spoofing attacks [30]. Alternatively, the authentication infrastructure can be accomplished with 802.1x authentication server infrastructure or Blom key predistribution [31, 32] can be used to establish the secure channels. This assumption of authentication and the first assumption above about having no mutual trust may seem contradictory. However, they are not contradictory since trust and authentication are two different security concepts and one can be held without the other.

Fourthly, we assume that *the identities of the mesh nodes are unique.* As IBC implies, DKEM utilizes the identities of the mesh nodes as their public keys. Therefore, these identities should be unique. In order to easily overcome this uniqueness issue, the identities of the nodes are selected to be their IP (Internet Protocol) addresses. This can be simply obtained through dynamic address allocation such as DHCP (Dynamic Host Configuration Protocol), where a centralized server ensures the uniqueness of the node addresses.

Finally, we assume that *each mesh node has a mechanism to discover its one hop neighbors and the communication among the mesh nodes is limited to neighborhood.* Any adversary can simply decrease the bandwidth share of a node by increasing the number of hops in a route between the source and the destination nodes that a packet will traverse [33, 34]. In order to prevent this type of action, and thus to improve the capacity of the network, nodes should only communicate with their neighbors, as the analytical upper and lower bounds of a network capacity imply [35]. This is accomplished by broadcasting messages instead of using unicast messaging unless it is not required.

2.2. Methodology of DKEM

DKEM consists of three phases: (i) *master private key share generation phase*, (ii) *master private key share distribution phase*, and (iii) *user private* *key generation phase*. In the first phase, mesh routers collaboratively generate the shares of the master private key, while in the second phase, these shares are distributed to the mesh clients. As soon as a mesh client constructs its share of the master private key, it can also contribute to the distribution process. Last phase provides a user private key generation service, by which each mesh node obtains its own private key. This service is carried out by a collaboration of a number of mesh nodes. The minimum number of mesh nodes involved in this process is determined by the level of threat, which depends on the application requirements. Precisely, the level of threat determines a security threshold, which in turn determines the number of collaborating mesh nodes. For more details about the selection of the threshold value, one can profitably refer to Section 4.

In DKEM, we have different levels of trust among the mesh nodes. As mentioned in Section 2.1, we assume that it is harder to compromise a mesh router than compromising a mesh client. Accordingly, we can trust mesh routers more than mesh clients. Hence, we may distribute a larger number of ThSS-shares to the mesh routers: e.g., if each mesh router holds x shares, then each mesh client will hold y shares such that $x > y$. Therefore, the scheme becomes an ($mx + cy$, k)-ThSS, where m is the number of mesh routers, c is the number of mesh clients and k is the threshold value. However, as mentioned above, the first phase of DKEM is carried out *only* by the mesh routers. Hence, the total number of ThSS-shares present in the system depends only on the number of mesh routers and the number of shares each holds. Therefore, the ThSS construction utilized in DKEM becomes (mx, k)-ThSS. Furthermore, we may also additively share the master private key of the network, r, as defined in Equation 1, where r^m is known by all of the mesh routers and r^u is distributed among the mesh nodes using the approach described above. With this method, collaboration required for the user private key construction operations includes a number of mesh nodes providing a total of k shares and a mesh router.

$$r = r^m + r^u \pmod q \tag{1}$$

Symbols used in the protocol definition are listed in Table 1. Throughout the paper, a value with a capital letter (except the abbreviations for entities M, C and U) represents a point on the elliptic curve. In Appendix A, we present some example network settings of our DKEM protocol.

2.2.1. Master Private Key Share Generation Phase

The first phase of DKEM is the master private key share generation phase, in which the mesh routers collaboratively generate the shares of the master private key. It starts with the mesh routers deciding on the global parameters of the elliptic curve cryptosystem and the threshold value to be utilized. The upcoming operations performed by the mesh routers are given in Algorithm 1.

First of all, each mesh router M_i selects x secrets $t_{i,\,z}$ and x polynomials $f_{i,\,z}(a)$ of degree $(k-1)$ over $\mathbb{F}_q$ such that $f_{i,\,z}(0) = t_{i,\,z}$, where $1 \leq z \leq x$ (Step 3). Secondly, each of them computes the subshares of the master private key, $\sigma_{j,\,i,\,z}$, by evaluating the generated polynomials for each mesh router (Step 5), and exchanges these information with their correspondents (Step 7). Finally, when mesh router M_i receives $((m-1)\times x)$ subshares of the master private key, it computes its master private key shares, $\gamma_{i,\,z}$, using Equation 2 (Step 13).

$$\gamma_{i,\,z} = \sum_{j=1}^{m} \sigma_{i,\,j,\,z} \pmod q \quad 1 \leq i \leq m\,,\, 1 \leq z \leq x \tag{2}$$

As soon as a mesh router computes its shares of the master private key, it broadcasts a message that indicates the end of the first phase (Step 15). This message will be referred as "*FINISH* message" hereafter.

In Appendix A, Figure A.9 depicts an illustration for the master private key share generation phase of DKEM.

2.2.2. Master Private Key Share Distribution Phase

The second phase of DKEM is the master private key share distribution phase, in which the shares of the master private key are distributed to the mesh clients. The procedures followed by the mesh nodes are given in Algorithm 2.

Table 1: Symbols used in DKEM Protocol Definition

Symbol	*Description*
U_i	i^{th} user
M_i	i^{th} mesh router
C_i	i^{th} mesh client
m	Number of mesh routers
c	Number of mesh clients
x	Number of shares for M_i for ThSS
y	Number of shares for C_i for ThSS
k	Threshold value of ThSS
B_i	User public key of U_i
V_i	User private key of U_i
r	Master private key of the network
r^m	Additive share of r, owned only by M_i
r^u	Additive share of r, threshold-wise shared among U_i
$\gamma_{i,\,z}$	z^{th} share of r^u owned by U_i
ℓ	Total number of computed $\gamma_{i,\,z}$ for U_i
$\sigma_{i,\,j,\,z}$	z^{th} subshare of r^u generated by M_j for M_i
$\rho_{i,\,j}$	Partial share of r^u generated by U_j for U_i
$\Gamma^u_{i,\,j,\,z}$	z^{th} share of V_i generated by U_j using r^u
$\Gamma^m_{i,\,j}$	Share of V_i generated by M_j using r^m

Algorithm 1 Master Private Key Share Generation

```
1:  procedure Phase 1: Mesh Router $M_i$ ($m$, $x$, $k$)
2:      for all $z \in \{1, ..., x\}$ do
3:          select $t_{i,\,z}$ & $f_{i,\,z}(a)$
4:          for all $j \in \{1, ..., m\}$ do
5:              compute $\sigma_{j,\,i,\,z}$
6:              if $j \neq i$ then
7:                  transmit $\sigma_{j,\,i,\,z}$ to $M_j$
8:              end if
9:          end for
10:     end for
11:     if $((m-1) \times x)$ $\sigma_{i,\,j,\,z}$'s received then
12:         for all $z \in \{1, ..., x\}$ do
13:             compute $\gamma_{i,\,z}$
14:         end for
15:         broadcast *FINISH*
16:     end if
17: end procedure
```

When mesh client C_i receives a *FINISH* message from one of its neighbors, it broadcasts a request message on its master private key share (Step 14). Mesh node U_j, receiving this request message, can contribute to the distribution process if and only if it has already computed its (at least two) shares of the master private key. If this is the case, then the mesh node U_j transmits a message to the mesh client C_i that includes the number of shares it can contribute with (Steps 3, 23, and 27). Otherwise, mesh node U_j saves this request in order to handle it after computing its master private key share(s) (Steps 5 and 29).

When mesh client C_i receives a number of replies that indicates a total contribution of at least k partial shares, it randomly selects sufficient number of contributing mesh nodes, and broadcasts another message indicating the selected nodes that will contribute to the distribution operation (Step 18). Upon

Algorithm 2 Master Private Key Share Distribution

```
procedure Phase 2: Mesh Router $M_j$ (x)
    if share request by $C_i$ & $\gamma_{i,\,x}$ is computed then
        transmit a contribution reply to $C_i$
    else if share request by $C_i$ then
        save request
    else if contribution required by $C_i$ then
        compute $\rho_{i,\,j}$
        transmit $\rho_{i,\,j}$ to $C_i$
    end if
end procedure
procedure Phase 2: Mesh Client $C_i$ $(x,\,k)$
    $\ell \leftarrow 1$
    if FINISH message received for the first time then
        broadcast share request
    else if $y > 1$ & $\ell < y$ & $\gamma_{i,\,\ell-1}$ is computed then
        broadcast share request
    else if sufficient contributions received then
        broadcast share request with contributors
    else if $k$ $\rho_{i,\,j}$'s received then
        compute $\gamma_{i,\,\ell}$
        $\ell \leftarrow \ell + 1$
        for all saved share requests from $C_j$ do
            transmit a contribution reply to $C_j$
        end for
    else if share request by $C_j$ then
        if $\ell > 1$ & $\gamma_{i,\,\ell-1}$ is computed then
            transmit a contribution reply to $C_j$
        else
            save request
        end if
    else if contribution required by $C_j$ then
        compute $\rho_{j,\,i}$
        transmit $\rho_{j,\,i}$ to $C_j$
    end if
end procedure
```

receipt of this second request message, if mesh node U_j recognizes that its contribution is not required, then it simply discards this message. Otherwise, it computes the master private key partial share of the mesh client C_i, $\rho_{i,\,j}$, via Equation 3, where $l_{j,\,z}(i, \ell)$ is the Lagrange basis polynomial and ℓ is the number of computed master private key shares (Steps 7 and 32).

$$\rho_{i,\,j} = \sum_{z} \gamma_{j,\,z} \times l_{j,\,z}(i, \ell) \pmod q \begin{cases} 1 \leq z \leq y & U_j = C_j \\ 1 \leq z \leq x & U_j = M_j \end{cases} \tag{3}$$

Thereafter, mesh node U_j transmits the computed partial share to mesh client C_i (Steps 8 and 33). When mesh client C_i receives a total of k partial shares, it computes its master private key share by simply adding up all the received data, as given in Equation 4 (Step 20). If $y > 1$, after computing their first master private key shares, i.e., $\ell = 1$, mesh clients broadcast another request message on their second master private key shares, i.e., $\ell = 2$, and this procedure is repeated until all y master private key shares are computed, i.e., $\ell = y$ (Step 16).

$$\gamma_{i,\,z} = \sum_{j} \rho_{i,\,j} \pmod q \quad 1 \leq z \leq \ell \tag{4}$$

In Appendix A, Figure A.10 depicts an illustration for the master private key share distribution phase of DKEM.

2.2.3. User Private Key Generation Phase

The last phase of DKEM is the user private key generation phase, in which each mesh node constructs its own private key with the collaboration of the other mesh nodes. The operations performed by the mesh nodes are given in Algorithm 3.

After mesh node U_i finishes computing its share(s) of the master private key, it broadcasts a request message on the construction of its own private key (Step 8). Another mesh node U_j, receiving this request message, can contribute to the user private key generation process if and only if it has already computed its share(s) of the master private key. Otherwise, it will save this request in order to handle it once it has its master private key share(s) (Step 26).

Algorithm 3 User Private Key Generation Phase

```
procedure PHASE 3: MESH NODE $U_i\,(x,\, y,\, k)$
    if $U_i$ is a mesh router then
        $e \leftarrow x$
    else if $U_i$ is a mesh client then
        $e \leftarrow y$
    end if
    if $\gamma_{i,\,z}$ is computed then
        broadcast share request message
    else if sufficient contributions received then
        broadcast share request with contributors
    else if request by $U_j$ and $\gamma_{i,\,1}$ is computed then
        transmit a contribution reply to $U_j$
    else if $(k - e)$ $\Gamma^{u}_{i,\,j,\,z}$'s received then
        compute $V_i$
    else if contribution required with $p$ shares then
        if $\gamma_{i,\,z}$ is computed then
            for all $z \in \{1,\, \ldots,\, p\}$ do
                compute $\Gamma^{u}_{j,\,i,\,z}$
                transmit $\Gamma^{u}_{j,\,i,\,z}$ to $U_j$
            end for
            if additive share requested then
                compute $\Gamma^{m}_{j,\,i}$
                transmit $\Gamma^{m}_{j,\,i}$
            end if
        else
            save request
        end if
    end if
end procedure
```

If the mesh node U_j has already computed its share(s) of the master private key, then it transmits a message to the requesting mesh node U_i that includes the number of shares it can contribute with (Steps 12).

When mesh node U_i receives a number of replies that indicates a total contribution of $(k - x)$ or $(k - y)$ user private key shares, depending on being a mesh router or a mesh client, respectively, it randomly selects sufficient number of contributing mesh nodes. If the requesting mesh node U_i is a mesh client, one of these contributing mesh nodes must be a mesh router due to the fact that the additive share of its user private key can only be provided by a mesh router. Since the mesh routers are arranged specifically so as to cover the network area, as discussed in Section 2.1, at least one mesh router will be in the communication range of the requesting mesh client. Hence, this additional share will always be received. On the other hand, if the requesting mesh node U_i is a mesh router, then it can compute this additional share by itself.

Thereafter, mesh node U_i broadcasts another request message indicating the selected nodes that will contribute to the user private key generation process, specifying the mesh router that will also contribute with the additive share of its user private key (Step 10). Upon receipt of this second request message, if mesh node U_j recognizes that its contribution is not required, it simply discards the message. Otherwise, it computes the user private key share(s) of the requesting mesh node U_i, $\Gamma^u_{i,\,j,\,z}$, using Equation 5, where B_i is the public key of the mesh node U_i (Step 18). If the mesh node U_j is the mesh router from which the mesh node U_i requested the additive share of its user private key, then it also computes this additional share using Equation 6 (Step 22).

$$\Gamma^u_{i,\,j,\,z} = \gamma_{j,\,z} \times B_i \quad \begin{cases} 1 \leq z \leq y & \text{if } U_j = C_j \\ 1 \leq z \leq x & \text{if } U_j = M_j \end{cases} \tag{5}$$

$$\Gamma^m_{i,\,j} \quad = \quad r^m \times B_i \tag{6}$$

After that, mesh node U_j delivers the computed user private key share(s) to the mesh node U_i (Step 19 and 23). As soon as the requesting mesh node

U_i receives all of the required data, it constructs its own private key using Equation 7, where $l_{j,\,z}(i)$ is the Lagrange basis polynomial (Step 14).

$$V_i = \Gamma^{m}_{i,\,j} + \sum_{j=1}^{t_1}\sum_{z=1}^{t_2} \Gamma^{u}_{i,\,j,\,z} \times l_{j,\,z}(i) \quad \begin{cases} t_1 \times t_2 = k \\ t_1\text{: collaborators} \\ t_2\text{: provided shares} \end{cases} \tag{7}$$

In Appendix A, Figure A.11 depicts an example scenario for the user private key generation phase of DKEM.

2.3. Repeat Request After Timeout

DKEM necessitates collaboration of mesh nodes during the computations in its last two phases. In the second phase, master private key shares are collaboratively distributed to the mesh clients, while in the last phase, mesh nodes generate their private keys with the contribution of the other mesh nodes. These constructions require request messages to be broadcast. Therefore, if a requesting mesh node does not have sufficient number of neighbors that can contribute with a total of k shares, where k is the threshold value, it simply cannot complete these phases since the corresponding requests cannot provide it with sufficient number of shares for the required computations. However, a request message may also drop due to collisions. As a result, the requesting mesh node cannot compute either of the related keys in spite of having sufficient number of neighbors.

This problem cannot be resolved using a MAC (Medium Access Control) layer mechanism because of the facts that the corresponding messages are broadcast messages and they are not retransmitted. In order to overcome collision based packet drops of the request messages, DKEM adopts *repeat request after time-out* method. In this method, each of the requesting mesh nodes sets a timer associated with their requests, just after transmitting them. If a requesting mesh node cannot receive all of the required data within a predefined time interval, it retransmits its request message and sets another timer associated with this retransmitted-request. Mesh nodes retransmit their request messages only for a predefined number of times to avoid indefinite message retransmissions in case that they may not have enough neighbors to provide them with sufficient number of shares for the requested operation. In our implementations, mesh nodes wait for 3 sec to repeat their requests and they retransmit their request messages at most 10 times. We determined these values via simulation over various values to optimize the performance of our solutions with respect to both key establishment latency and successful user private key generation rate.

3. Security Analysis of Baseline and DKEM Protocols

A WMN may come under attack from both *outsider* and *insider* attackers. Outsider attackers are limited to attacking the network without having access to the network resources; this includes eavesdropping, injection, modification, and replay of packets. Insider attackers, having obtained access to authentication material (e.g., cryptographic keys), can pose as authorized participants and attack the network from the inside. Insider attackers are either nodes that have been compromised or honest nodes that have turned malicious.

The outsider attacker could be either passive or active, but for the insider attacker, we consider the semi-honest type adversary, which is defined as follows.

Definition 3.1 (Semi-Honest Party)**.** *A Semi-Honest Party (a.k.a. honest-but-curious party) is an insider attacker that properly follows the prescribed actions of the protocol, but can keep a record of all its intermediate communications and analyze in order to deduce some additional knowledge about the other entities' private information sets.*

In this model, the insider attacker adheres to the predefined protocol steps in order to remain undetectable. This assumption is relevant because considering a WMN, the sustainability of the system is based on trustworthiness of the entities, and detecting any malicious behavior is not tolerated and may result in a ban.

For a key establishment protocol, the most worrying attack that should be prevented is revealing the secret key of an entity or the system by either outsider or insider attacker since the key establishment is done on the wireless environment. In this section, we first discuss the security of the baseline protocol proposed by Deng et al. [28]. We show that the baseline protocol is not secure against a semi-honest insider attacker. We prove that the attacker can obtain master private shares of other nodes and more crucially, can obtain the master private key of the system results in compromising the system completely. Thereafter, we show that our protocol is secure against both outsider and semi-honest insider adversaries.

3.1. Security Analysis of Baseline Protocol

We keep the same notation used in the original paper [28], in order to avoid any confusion with DKEM protocol.

Theorem 3.1. *Considering the protocol defined in [28], a semi-honest node can obtain a master private share of another mesh node.*

Proof. Let C_p be a semi-honest node that wants to generate its share of the master private key and let C_i be another node that contributes to this process by computing the partial share of the node C_p as in Equation 8, where $l_i(p) = \prod_{j\in\mathcal{G},\, j\neq i}^{k} \frac{p-j}{i-j}$ and $\mathcal{G}$ is the set of the contributors. C_i sends the subshare $s_{i,\,p}$ to C_p.

$$s_{i,\,p} = S_i \times l_i(p) \pmod q \tag{8}$$

At this point, C_p should not recover the secret share S_i from this subshare. However, C_p can easily compute $l_i(p)$, which is a publicly known value, and it obtains the secret share of C_i as in Equation 9, where a^{-1} mod q denotes the multiplicative inverse of a in $\bmod q$.

$$S_i = s_{i,\,p} \times l_i(p)^{-1} \pmod q \tag{9}$$

Therefore, a semi-honest client can obtain the master private key share of a mesh router. □

Theorem 3.2. *A semi-honest client C_p can obtain the master secret of the system and can compute all the mesh nodes' private keys.*

Proof. As a result of Theorem 3.1, a semi-honest client can compute the master secret of the system by collecting k number of subshares from other nodes and obtain the master private key of the system, SK_M, given in Equation 10, where $k-1$ is the degree of the secret polynomial and SK_M is the master private key of the system.

$$SK_M = \sum_{i=1}^{k} S_i \times l_i(0) \pmod q \tag{10}$$

Hence, C_p can compute the private key of any mesh node by $sk^j = SK_M \times Q_{ID}^j$, where sk^j and Q_{ID}^j are the secret and the public key of node j respectively. □

Note that, since the attacker is able to obtain the secret keys without physically capturing or compromising any user, the baseline protocol is also not resilient. Therefore in Section 4 we do not examine the resiliency of the baseline protocol.

3.2. Security Analysis of DKEM Protocol

The security goals of our DKEM protocol are providing secret data confidentiality from both outsider (both passive and active) and semi-honest insider adversaries. We assume that q is a large prime, thus the probability of guessing an arbitrary number over the finite field $\mathbf{F}_q$ is negligible. In what follows, we examine the security of the DKEM protocol and show that our constructions are secure in the semi-honest model.

Theorem 3.3. *Considering the first phase of the DKEM protocol, an outsider attacker $\mathcal{A}_o$ can obtain a mesh router's master private key share with a negligible probability.*

Proof. Let there be m mesh routers in the system, and let x be the number of shares that a mesh router holds. Besides, let assume $\mathcal{A}_o$ can observe all of the interactions among the mesh routers. So, $\mathcal{A}_o$ learns the values of

$\sigma_{i,j,z}$, $\forall i, j$ with $i \neq j$ and $1 \leq i, j \leq m$, and $\forall z$ with $1 \leq z \leq x$. If $i' \in_R \{1, \ldots, m\}$ and $z' \in_R \{1, \ldots, x\}$. In order to calculate the value of $\gamma_{i',z'} = F(\sigma_{i',1,z'}, \ldots, \sigma_{i',i',z'}, \ldots, \sigma_{i',m,z'})$, the adversary has to figure out the value of $\sigma_{i',i',z'}$. However, the distribution of σ is uniform, thus the success probability of $\mathcal{A}_o$ is $\frac{1}{|q|}$ which is negligible. Since i' and z' are arbitrary, this argument holds for any master private key share. □

Theorem 3.4. *Considering the first phase of DKEM protocol, an insider semi-honest attacker can obtain a mesh router's master private key share with a negligible probability.*

Proof. There are two types of attackers to consider: (i) a semi-honest router, and (ii) a semi-honest client.

(i) Let M_s be a semi-honest mesh router. Note that, the only advantage of M_s over $\mathcal{A}_o$ is the knowledge of the values $\sigma_{i,s,z}$, where $1 \leq i \leq m$ and $1 \leq z \leq x$. However, M_s still needs to calculate the values of $\gamma_{i,z}$ for $i \neq s$. As indicated above, for any fixed i' and z', the success probability of the attacker is $\frac{1}{|q|}$, which is negligible.

(ii) It is obvious that a semi-honest mesh router is more powerful than a semi-honest client C_s. Thus, a semi-honest mash node cannot either obtain mesh router's master private key share.

□

Theorem 3.5. *Considering the second phase of DKEM protocol, a semi-honest client C_s can obtain a master private key share of another mesh node with a negligible probability.*

Proof. There are two cases to consider: A semi-honest mesh client try to obtain (i) master private key share of another mesh router (M_i), or (ii) master private key share of another mesh client (C_i).

(i) Let C_s be a semi-honest client that wants to generate its master private key share according to Algorithm 2. Let M_i be a mesh router that contributes to this process by computing the z-th master private key partial share of the mesh client C_s via Equation 11, where $l_{i,j}(s) = \prod_{j \in \mathcal{G},\, j \neq i}^{k} \frac{s-j}{i-j}$ and $\mathcal{G}$ is the set of the contributors.

$$\sum_{1<z\leq x} \rho_{s,\,i,\,z} = \sum_{z\leq x} \gamma_{i,\,z} l_{i,j}(s) \pmod q \tag{11}$$

M_i sends this sum of subshares to C_s. In order to complete the proof, we have to show two sub-cases such that (a) C_s cannot not recover any single secret share $\gamma_{i,\,z}$ and (b) C_s cannot compute any other point in the secret polynomial.

(a) The problem is equivalent to solving linear equations with z-unknowns, where $1 < z \leq x$. Since in this case there is only one equation, the solution is infeasible. Therefore, C_s cannot obtain any $\gamma_{i,\,z}$ value.

(b) C_s cannot compute any other point in the secret polynomial, i.e., $f_i(a)$ for $\forall a \geq 0$ such that $a \neq s$. Since Equation 11 is calculated using $l_{i,j}(s)$ with different j values and from the first sub-case (a) C_s cannot obtain any $\gamma_{i,\,z}$, it cannot compute any other point in the secret polynomial $f_i(a)$.

(ii) It is easy to see that the above mentioned reasoning also works when the contributor is a mesh client (C_i). The sum of subshares calculation is similar as in Equation 11 for $z = 2$. Therefore, a semi-honest mesh client cannot obtain master private key share of another mesh client (C_i).

□

Note that, considering the second phase of the DKEM protocol, a mesh node U_j can contribute to the share generation process of a mesh client C_i if it has at least two master private key shares. The partial share is computed by summing the shares as given in the Equation 11. U_j can provide more than one partial shares if and only if the provided linear equations ensure an *underdetermined linear system* (i.e. the unknowns outnumber the number of linearly independent equations). Otherwise, providing a set of equations that is linearly solvable simply reveals U_j's shares.

Theorem 3.6. *Considering the third phase of the DKEM protocol, a semi-honest node U_s cannot obtain a master private key share of another mesh node.*

Proof. Let U_j be a node that contributes for the secret key generation of the mesh node U_s. Using Equation 12, U_j computes the following share.

$$\Gamma^{u}_{s,\,j,\,z} = \gamma_{j,\,z} \times B_s \quad \begin{cases} 1 \leq z \leq y & \text{if } U_j = C_j \\ 1 \leq z \leq x & \text{if } U_j = M_j \end{cases} \tag{12}$$

Obtaining $\gamma_{j,\,z}$ from the above equation is equivalent to solving the Elliptic Curve Discrete Logarithm Problem (ECDLP). The same security reduction is also applicable for the case of computing $\Gamma^{m}_{i,\,j} = r^m \times B_i$. Furthermore, obtaining any number of these shares from other users does not provide an additional advantage to the attacker. □

4. Resiliency Analysis of DKEM

Resiliency of a network is defined as the maximum number of compromised nodes for which the security of the network is not affected. As described in Section 2.2, DKEM uses AdSS, in which one of the additively shared secrets is known by all of the mesh routers and the other one is shared among the mesh nodes using (mx, k)-ThSS, where m is the number of mesh routers, x is the number of shares each holds and k is the number of shares required for the reconstruction processes. In this scheme, the generated master private key shares are distributed to the mesh clients in such a way that each mesh client holds *at most* y shares, where $y < x$. Our scheme ensures that a mesh router will always contribute to the construction of a user private key. Hence, an adversary must capture at least one mesh router in order for him to be successful. In other words, as long as a mesh router is not compromised, no matter how many mesh clients are captured, the resiliency of the network is conserved.

Although being harder than compromising a mesh client (as mentioned in Section 2.1), if at least one mesh router is compromised, then the resiliency of the network depends on the total number of captured shares. In this case, collaborations among the captured mesh nodes, which aim at obtaining the shares of any other mesh node, should also be taken into account.

Let assume an adversary compromises $a \geq 1$ mesh routers and b mesh clients. The number of *directly* captured shares is $(ax + by)$. On the other hand, the number of *indirectly* captured shares depends on obtaining a set of equations (in the form of Equation 3) that are linearly solvable. In this context, the resiliency of the network is preserved when the system satisfies Equation 13, where E is the expected number of *indirectly* captured shares.

$$k > ax + by + E \quad (a \geq 1) \tag{13}$$

In order to show how E varies with respect to the number of compromised nodes, we implemented a simulator. We make the following basic assumptions in order to model our simulator: (i) partial shares of clients are transmitted by summing up two master private key shares (in the form of Equation 3), (ii) a client obtains each share pair from a uniformly random selected router. For the second assumption, in real case, the share transmission may depend on the geographic distribution of the nodes and the communication ranges. However, this assumption is relevant when the clients are mobile or the communication range is long enough.

Now we describe how our simulator works. After the mesh routers obtain their master private key shares, each client receives their master private key shares from a set of routers selected uniformly random. A router can provide more than one equation to the same client given that none of the single shares are common among the provided equations. After that, the simulator randomly selects a given number of nodes to be compromised and then extracts the indirectly captured shares. In order to find the indirectly captured shares, the simulator uses a *graph theoretical* approach. For each mesh router, the simulator constructs an undirected and unweighted graph (in total, m graphs), such that each vertex represents a master private key share of the router. Simulator generates an edge between two verticies if the corresponding shares are provided to a client with an equation. In each graph, the simulator checks for a cycle through the corrupted edges (equations) of the compromised clients. If a cycle is found, the vertices on that cycle are marked. Finding a cycle means that the shares on the cycle can be captured by solving the linear equations. In addition, the vertices that are directly connected to a captured vertex are also marked as captured. This is because of the fact that if the attacker has an equation (originally with two unknowns) and one of the unknown is captured, then the other unknown can be easily found. In the resulting graph, the marked vertices yield the number of indirectly captured shares for a mesh router. The total number of indirectly captured shares is equivalent to the total number of marked vertices for all routers.

Figure 2 depicts the results of simulations comparing the number of compromised clients (i.e. b) versus the expected number of indirectly captured shares (i.e. E). We run the experiments for the combinations of the following parameters: the number of mesh routers: 25 and 50; the number of master private key share of routers: 4, 8 and 16; the number of master private key share of clients: 2. Note that compromising clients are easier than compromising routers therefore minimizing the number of client share is reasonable. Therefore we fix the number of client share to the minimum number of 2, since doubling both the number of client share and router share is not meaningful. For each given number of compromised clients (in x axis), the experiments are run for 10.000 times and the expected numbers of indirectly captured shares are calculated by taking the average.

In Figure 2, the results show that the number of indirectly captured shares decreases as the number of routers increases. This is because of the fact that when there are more supplier routers, clients can receive their shares from more alternative routers. Note that the number of graphs in the simulator is equivalent to the the number of routers. Thus increasing the number of routers also increases the total number of graphs. Assuming a uniformly distributed system,

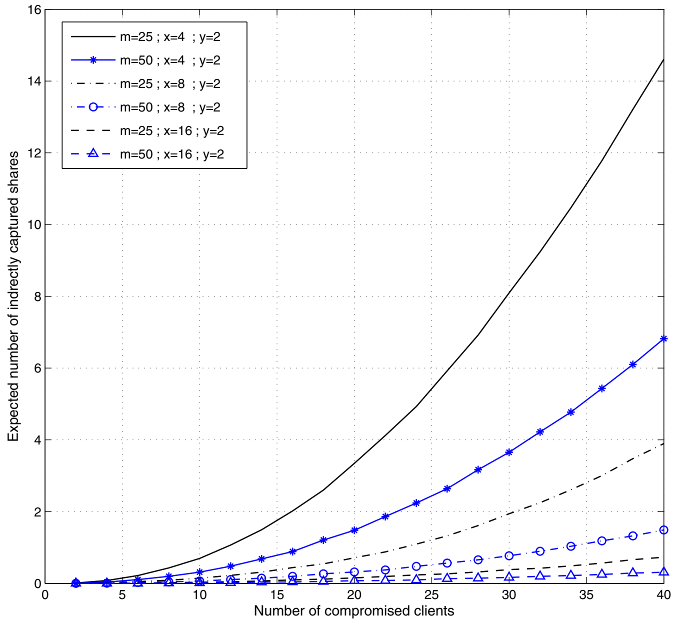


Figure 2: Number of compromised clients vs. the expected number of indirectly captured shares for different m, x and y value combinations. Considering each number of compromised clients, the experiment is run for $10,000$ times and the result is calculated by taking the averages.

this will decrease the probability of having a cycle for a given graph. Similarly the number of indirectly captured shares is inversely proportional to the number of router share. Considering our simulator, the number of vertices in each graph increases as number of shares to be distributed increases. Therefore, this causes a decrease in the probability of having a cycle in a more sparse graph.

One another important result of Figure 2 is that the expected number of indirectly captured shares is relatively small comparing the directly captured shares. According to the first (and worst) setting (i.e. $m = 25, x = 4, y = 2$), for example, after compromising 15 clients the adversary obtains 32 shares in total; 30 of them are directly captured (i.e., each client has 2 master private key shares $15 \times 2 = 30$) and only 2 of them are indirectly captured. Therefore, the number of indirectly captured shares is limited as compared to the number of directly captured shares. Thus we conclude that the resiliency of DKEM mostly depends on the number of physical corruptions.

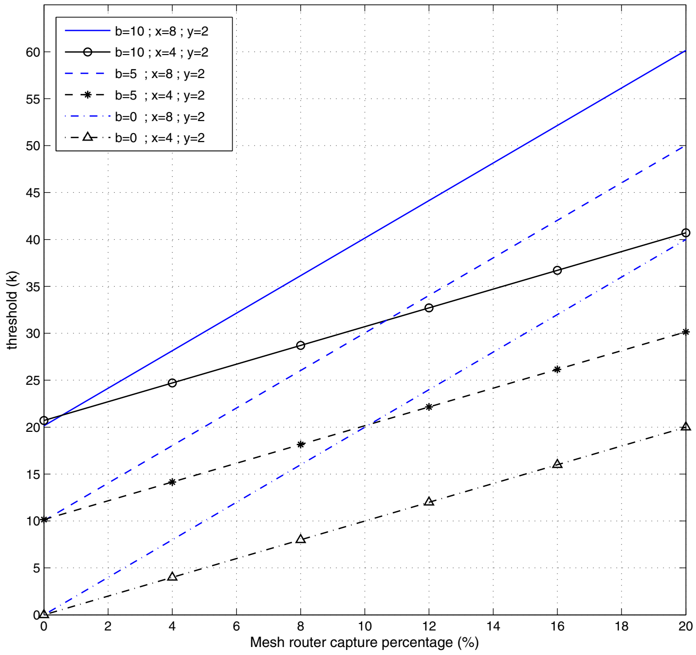


Figure 3: Mesh router capture percentage vs. threshold parameter for different b, x, y combinations

560 Figure 3 depicts mesh router capture percentage versus the required threshold value using the inequality $k > ax + by + E$ given in Equation 13. We consider various numbers of corrupted clients ($b = 0, 5, 10$) and number of shares ($x = 8, y = 2$ and $x = 4, y = 2$). Here the number of mesh routers is taken as 25. Each line is linear; this is expected because of the linearity of Equation 13. 565 Note that DKEM ensures that as long as a mesh router is not compromised, no matter how many mesh clients are captured, the resiliency of the network is preserved. Furthermore, according to assumption mentioned in Section 2.1,

it is assumed that capturing a mesh router is much harder than capturing a mesh client. Hence, in a typical system, a router has more shares than a client. Therefore, if the number of captured routers increases, the total number of shares compromised also increases dramatically, depending on the value of x. This, in turn, causes higher threshold values. For a system that has a high risk of attack, the threshold value should be selected as high accordingly; the consequences of this are discussed in Section 6.1.

5. Communication, Computational and Energy Overhead of DKEM and the Baseline Protocol

Both DKEM and the baseline protocol incur communication overhead in all of their phases, whereby the first phases dominate the others. The computational overhead of DKEM is introduced by the utilization of EC-IBC, AdSS and ThSS, while the computational overhead of the baseline protocol is introduced by the utilization of EC-IBC and ThSS. In the following subsections we comparatively examine the communication, computational and energy overhead of DKEM and the baseline protocol.

5.1. Communication Overhead of DKEM and the Baseline Protocol

In DKEM, the master private key of the network is generated collaboratively to eliminate the TTP assumptions of both IBC and secret sharing constructions. As described in Section 2.2.1, in the first phase of DKEM, the shares of the master private key are generated with the contribution of *all the mesh routers*. This process requires $(m \times (m-1))$ unicast messages to be transmitted, where m is the number of mesh routers. In the last two phases of DKEM, the number of transmitted packets are affected only by the number of the mesh nodes that have already computed their master private key shares. Nevertheless, none of those operations introduces larger amount of packet transmissions as compared to that of the first phase. Besides, the sizes of all message payloads are approximately the same with each other. Hence, the communication overhead introduced by the message transfers in the last two phases are negligible. As a consequence, the communication complexity of DKEM is $O(m^2)$, in terms of the number of packets transmitted.

In the baseline protocol, the communication overhead of generating the shares of the master private key is far more than the communication overhead of the upcoming operations performed in the first phase. In other words, analogous to that of DKEM, the communication overhead introduced by the message transfers after the master private key share generation process is negligible. On the other hand, as described in Section 1.2, this specific process requires $(n \times (n-1))$ unicast messages to be transmitted, where n is the number of mesh nodes. Consequently, the communication complexity of the baseline protocol becomes $O(n^2)$, in terms of the number of packets transmitted. In this regard, we can state that the communication complexity of the baseline protocol is greater than the communication complexity of DKEM, considering the fact that $n > m$.

5.2. Computational Overhead of DKEM and the Baseline Protocol

In the first phase of DKEM, mesh routers collaboratively generate the shares of the master private key. They start with computing the corresponding subshares, which includes $((m+1) \times x \times (k-1))$ modular additions and $(m \times k \times (k-1)/2)$ modular multiplications, where m is the number of mesh routers, x is the number of shares each holds and k is the threshold value. Then, after receiving all of the relevant data, they perform another modular addition of $(m \times x)$ values to compute their shares of the master private key. In the second phase of DKEM, these shares are distributed to the mesh clients. The corresponding partial shares are computed using k modular multiplications and k modular additions. When a mesh client receives k subshares, it computes its master private key share after $(3k \times (k-1))$ modular multiplications, $(k \times (k-1))$ modular inverse operations, k elliptic curve (EC) scalar point multiplications and k EC point additions. All of the second phase operations are performed at most y times, where y is the number of shares each mesh client holds. Finally, in the last phase of DKEM, mesh nodes construct their own private keys. Mesh routers compute user private key ThSS-shares using x EC scalar point multiplications, while mesh clients compute such shares using at most y EC scalar point multiplications. Besides, mesh routers compute the additional share for the requested user private key using $(x + 1)$ EC scalar point multiplications. Moreover, mesh nodes compute their private keys using at most $(3k \times x \times (k - 1))$ modular multiplications, $(k \times x \times (k - 1))$ modular inverse operations, $(k \times x)$ EC scalar point multiplications and $(k \times x + 1)$ EC point additions.

On the other hand, in the baseline protocol, which is described in Section 1.2, nodes perform $((n + 1) \times (k - 1))$ modular additions and $(n \times k \times (k - 1)/2)$ modular multiplications to compute their subshares, where n is the total number of nodes within the system. After that, each performs another modular addition of n values to compute their shares of the master private key. Then, each node computes their share of the master public key using one EC point multiplication and reconstructs the actual value of the master public key using n EC point additions. Thereafter, nodes construct their own private keys using at most $(3k \times (k - 1))$ modular multiplications, $(k \times (k - 1))$ modular inverse operations, $(k + 1)$ EC scalar point multiplications and $(k + 1)$ EC point additions.

Table 2 gives the computational overhead of DKEM and the baseline protocol in terms of the types and the numbers of operations performed by each of the nodes. In this table, c is the number of mesh clients and p is the number of mesh clients that the mesh routers help.

5.3. Energy Overhead of DKEM and the Baseline Protocol

In order to calculate the energy consumed by the nodes of DKEM and the baseline protocol, we have made a number of assumptions and used relevant information provided in the literature, as detailed below. In brief, we have reduced the field and EC point operations into field multiplication operations and assuming 256-bit security, we have approximated the energy consumed by the nodes of the network, given specific values for the number of nodes and the number of shares hold by each of these nodes.

Table 2: Computational Overhead of DKEM and the Baseline Protocol

	DKEM		**Baseline Protocol**
	Mesh Router	**Mesh Client**	
Modular Addition	$x(k(m+1)-1)$ $+ pk$	$k(cy-p)$	$k(n+1)-1$
Modular Multiplication	$3kx(k-1)+kp$ $+ mk(k-1)/2$	$k(cy-p)$ $+ 3k(k-1)(x+y)$	$k(k-1)(3+n/2)$
Modular Inverse	$kx(k-1)$	$k(k-1)(x+y)$	$k(k-1)$
EC Point Addition	$kx+1$	$kx+1$	$n+k+1$
EC Point Multiplication	$x(k+2)+1$	$kx+y$	$k+2$

Using Modified Jacobian Coordinates, an EC doubling operation is equivalent to 4 field multiplication and 4 field squaring operations, while an EC point addition operation is equivalent to 9 field multiplication and 5 field squaring operations [36]. At this point, we assume that a squaring operation is equivalent to a multiplication operation in field arithmetic, for the sake of simplicity. Besides, we further assume that the total cost of modular addition and EC point addition operations are negligible as compared to the cost of all the other operations. Under these circumstances, from computational point of view, we can reduce each one of the modular exponentiation, modular inverse and EC point multiplication operations (without side-channel protection) into a certain number of modular multiplication operations. In this regard, a modular exponentiation would correspond to $3b$ modular multiplications, a modular inverse would correspond to $3b$ modular multiplications (due to Fermat's little theorem [37]), and a EC point multiplication would correspond to $30b$ modular multiplications, providing b-bit security.

Considering 256-bit security, mesh routers and mesh clients in DKEM, and the nodes in the baseline protocol perform approximately $37k^2+83k$, $36k^2+234k$ and $56k^2-26k$ field multiplications, respectively, with the following network specifications: $m = 25$, $c = 75$, $n = 100$, $x = 4$ and $y = 2$, where k is the threshold value. Having regard to the appraised values provided in [38] and [39], each one of the mesh routers and mesh clients in DKEM, and the nodes in the baseline protocol consume approximately $(384k^2+863k)\,nJ$, $(374k^2+2434k)\,nJ$ and $(582k^2-270k)\,nJ$ of energy, respectively. At this point, we can conclude that the nodes in DKEM consume less energy as compared to the nodes in the baseline protocol, as proposed by the lower coefficients of k^2, although the energy consumption complexity of both DKEM and the baseline protocol are $O(k^2)$. Furthermore, the communication complexity of DKEM is preferable as discussed in Section 5.1 and the baseline protocol is not secure as discussed in Section 3.1.

6. Performance Evaluation of DKEM

We analyzed the performance of our DKEM protocol using Network Simulator 2 (ns2) [40] version 2.35. For simulation scenarios, we modeled the network as having $n = 40, 60, 80, 100$ nodes within 2000×2000 square meters. Our network model includes 25 mesh routers, i.e., $m = 25$, that are deployed in gradual manner. In this deployment model, each mesh router is in the transmission range of its neighboring mesh routers. Additionally, each of them has 4 shares of the master private key, i.e., $x = 4$. On the other hand, mesh clients are deployed within the network area using bivariate uniform random distribution, and each holds at most 2 shares of the master private key, i.e., $y = 2$. Besides, we simulated the performance of our network for the threshold values $k \in \{6, 8, 10, 12, 14, 16\}$.

All of the simulations are run on a personal computer with the following configuration: Windows 8 (64-bit), Intel Core 2 Duo Quad CPU Processor at 2.40 GHz, 4 GB RAM and GCC 4.3.3 on Cygwin 2.859. In addition, we used MAC and network interface types of 802.11p [41], omni-directional antenna with 375 meter transmission range, priority queue defined under drop tail queue, DSDV (Destination-Sequenced Distance-Vector) routing and TCP (Transmission Control Protocol). Besides, we used EC-IBC implementation provided by MIRACL (Multiprecision Integer and Rational Arithmetic C/C++ Library) [42]. In this implementation, the non-singular elliptic curve is defined a " $y^2 = x^3 + 1 \mod (p)$ " and the cryptographic curve parameters have the following properties: (i) $p \bmod 3 = 2$, (ii) $p \bmod 8 = 3$, (iii) q is a prime factor of $(p + 1)$, (iv) $q > 3$, and (v) $q = 2^{159} + 2^{17} + 1$.

In order to evaluate the performance of our DKEM solution, we consider two metrics: (i) *key establishment latency*, and (ii) *success percentage of user private key generation*. Our first metric is defined as the elapsed time between the initial deployment and the final user private key computation. Our second metric is the ratio of the number of mesh nodes that have computed their own private keys to the network size.

The following subsections include discussions on the performance results of our DKEM protocol with respect to the defined metrics, and a comparison of those results with the respective evaluation results of the baseline protocol.

6.1. Key Establishment Latency

We evaluated the key establishment latency of DKEM, i.e., time elapsed between the start of the first phase and the end of the last phase, for different values of both the network size, n, and the threshold value, k. Figure 4 shows the corresponding results.

In the last two phases of DKEM, in which the master private key shares are distributed and the user private keys are generated, communication among the mesh nodes includes request messages, as described in Sections 2.2.2 and 2.2.3, respectively. As also discussed in Section 2.3, to handle collision based packet drops, mesh nodes adopt a *repeat request after timeout* method, by the use of which requests are repeated. When the threshold value increases, the probability of having sufficient number of neighbors decreases. Accordingly, the

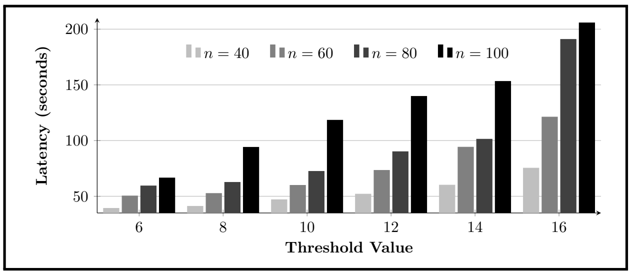


Figure 4: Key Establishment Latency in DKEM

probability of receiving sufficient number of shares for a specific construction operation decreases. Hence, the adopted method becomes favorable, in spite of *not* having sufficient number of neighbors. In other words, the requesting mesh node assumes that its request message has collided with another request message and repeats its request. As a result, key establishment latency increases as the threshold value increases. For instance, in a network with 40 nodes, the latency of key establishment is 39.2 sec at the threshold value of 6, while it is 51.9 sec at the threshold value of 12.

Furthermore, when the network size increases, the probability of a request message being collided with another request message increases, since all of the mesh routers compute their shares of the master private key almost at the same time, and thus, most of the mesh clients broadcast their master private key share request messages simultaneously. This increases the utilization of the adopted *repeat request after timeout* method. As a consequence, latency of key establishment increases as the network size increases. For instance, at the threshold level of 6, it takes 39.2 sec for 40 mesh nodes to compute their private keys, while it takes 66.4 sec when the network size is 100. Besides, at the threshold value of 12, an increase in the number of mesh nodes from 40 to 100 increases the key establishment latency from 51.9 sec to 139.7 sec.

Although these latency values might seem high, one should note that the key establishment procedure is performed only once. Besides, it is also important to note that the scenario of the measured key establishment latency involves all of the mesh clients to be deployed exactly at the same time. As discussed above, the first cause of this high latency is the utilized *repeat request after timeout* method, which gives rise to the increase in the packet drops due to packet collisions. Therefore, if we were to deploy the mesh clients fragmentarily, the key establishment latency would be lower since the packet collisions would be diminished. In this way, we can achieve higher threshold values needed to thwart against heavy node capture attacks discussed in Section 4.

6.2. Success Percentage of User Private Key Generation

We evaluated the success percentage of user private key generation, i.e., the number of mesh nodes that are capable of computing their own private keys over the network size, for different values of both the network size, n, and the threshold value, k. Figure 5 shows the corresponding results.

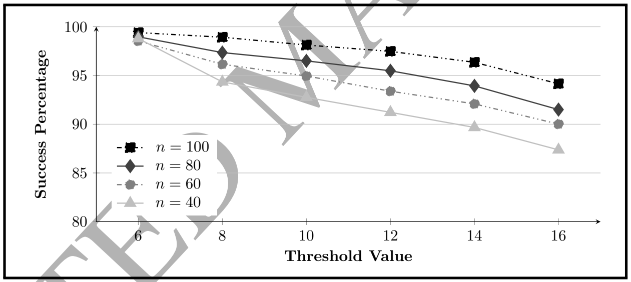


Figure 5: Success Percentage of User Private Key Generation in DKEM

In the last phase of DKEM, mesh routers can compute their own private keys if and only if they receive at least $(k - x)$ user private key shares, as discussed in Section 2.2.3, where x is the number of shares that a mesh router holds. Likewise, mesh clients should receive $(k-y)$ user private key shares for the proper construction of their own private keys, where y is the number of shares that a mesh client holds. However, a request receiving mesh node can provide the private key share(s) of a requesting mesh node if and only if it has already computed its share(s) of the master private key. Therefore, success percentage of user private key generation actually depends on the number of neighboring nodes that have already computed their master private key shares.

When the threshold value increases, the probability of a mesh node computing its master private key share decreases, since the received partial shares will not be sufficient for the corresponding construction process. Accordingly, for a specific mesh node, the number of neighbors that have already computed their master private key shares decreases. This results in a decrease in the number of shares received by a mesh node sending request to compute its own private key. Hence, an increase in the threshold value decreases the success percentage of user private key generation. Results show that, in a network with 40 nodes, while 99% of the mesh nodes can compute their own private keys when the threshold value is 6, at least 91% of them can perform the corresponding construction successfully when it is 12, and the success percentage is 87% at the threshold value of 16.

Besides, when the network size increases, the number of neighboring nodes increases. Consequently, the probability of a mesh client being able to compute its master private key share also increases, due to the increase in the number of accessible partial shares. As a result, for a specific mesh node, the number of neighbors that have already computed their master private key shares increases. This results in an increase in the number of user private key shares received by a requesting mesh node. Hence, an increase in the network size increases the success percentage of user private key generation. Results show that, at the threshold value of 12, at least 91% of the mesh nodes can compute their own private keys when the network size is 40, while at least 93% of them can perform the corresponding construction successfully when it is 60, and the success percentage is 97% when the network size is 100.

6.3. Comparison with the Baseline Protocol

We adapted the baseline protocol to WMNs in order to have a proper comparison with our DKEM solution. As described in Section 1.2, in the baseline protocol, the master key of the network is generated with the collaboration of all the users. Hence, in the baseline protocol, the first phase of DKEM, in which the master private key shares are generated, is executed by *all of the mesh nodes*. Since there is no other mesh node left to compute its master private key share, the second phase of DKEM is not included in the baseline protocol. However, different from DKEM, in the baseline protocol, mesh nodes also compute the shares of the master public key, publish them and reconstruct the corresponding public key. Finally, they execute the last phase of DKEM (except for the computations related to the additional share of the master private key) to compute their own private keys. It is important to note that in the baseline protocol each node holds only one share of the master private key.

Figure 6 shows *DKEM key establishment latency speed-up* as compared to the baseline protocol, which is defined as the ratio of the key establishment latency of the baseline protcol to the key establishment latency of DKEM, i.e., $\frac{\textit{Key Establishment Latency of the Baseline Protocol}}{\textit{Key Establishment Latency of DKEM}}$. As discussed in Section 6.1, an increase in either the threshold value or the network size also increases the utilization of the introduced *repeat request after timeout* method. This, in turn, increases the latency of key establishment. In the baseline protocol, after computing their shares of the master private key, mesh nodes compute their master public key shares and publish them, as mentioned above. This increases the use of the adopted *repeat request after timeout* method considerably, even more than that of DKEM.

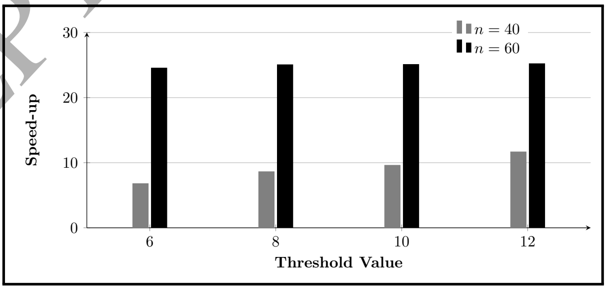


Figure 6: DKEM Key Establishment Latency Speed-up as Compared to the Baseline Protocol - This is the ratio of the key establishment latency of the baseline protocol to the key establishment latency of DKEM

When compared with DKEM, in the baseline protocol, it takes much longer for the final mesh node to finish computing its own private key. For instance, in a network with 40 nodes operating at the threshold value of 6, the key establishment latency is approximately 39 sec in DKEM, while it is approximately 266 sec in the baseline protocol, which is approximately 6.8 fold of DKEM's latency. When the network size is 40, as we increase the threshold value from 6 to 8, the latency of key establishment becomes approximately 41 sec in DKEM, while it becomes approximately 8.6 fold of that value, which is 353 sec, in the baseline protocol. Besides, at the threshold level of 6, when we increase the network size from 40 to 60, the key establishment latency becomes approximately 50 sec in DKEM, while it becomes approximately 1234 sec in the baseline protocol, which is approximately 25 fold of DKEM's latency. In fact, in the baseline protocol, we cannot further increase either the network size over 60 nodes or the threshold value over 12; the increase observed in the latency of key establishment is so excessive that the protocol becomes inapplicable. Nevertheless, the least elapsed time observed in the baseline protocol, which is approximately 266 sec ($n = 40$, $k = 6$), is higher than the highest elapsed time of DKEM, which is approximately 206 sec ($n = 100$, $k = 16$). Hence, we can conclude that the key establishment latency of DKEM outperforms the key establishment latency of the baseline protocol.

On the other hand, Figure 7 shows the success percentage of user private key generation of the baseline protocol. As discussed in Section 6.2, the corresponding ratio depends on the number of the mesh nodes that have already computed their master private key shares. However, in the baseline protocol, the shares of the master private key are generated with the contribution of all the mesh nodes. At the end of the first phase, all of the mesh nodes finish computing these shares. Therefore, in the baseline protocol, this ratio depends only on the number of neighboring nodes. Consequently, the success percentage increases when either the threshold value decreases or the network size increases.

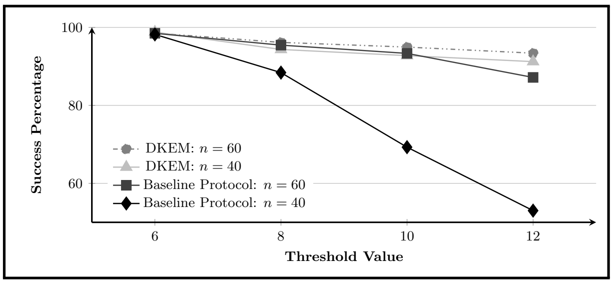


Figure 7: Success Percentage of User Private Key Generation in the Baseline Protocol

As compared with DKEM, the number of the mesh nodes that can construct their own private keys is fewer in the baseline protocol. For instance, in a network with 40 nodes operating at the threshold value of 10, success percentage of user private key generation is approximately 93% in DKEM, while it is approximately 69% in the baseline protocol. When the network size is 40, as we increase the threshold value from 10 to 12, the success rate becomes approximately 91% in DKEM, while it becomes approximately 53% in the baseline protocol. Besides, at the threshold level of 10, when we increase the network size from 40 to 60, the success percentages of user private key generation become approximately 95% and 93% in DKEM and in the baseline protocol, respectively. Hence, we can conclude that the successful user private key generation rate of DKEM outperforms the successful user private key generation rate of the baseline protocol as the threshold value goes beyond 6.

7. Conclusions

WMNs are one of the important research areas that provide low-cost and high-speed network services for the end users. As in all types of wireless networks, key management is their most important and critical security concern. Unfortunately, the conventional solutions for the key establishment problem do not fit in the unique characteristics of WMNs, i.e., being dynamically self-organized, self-configured and self-healing.

In this paper, we propose an efficient and secure distributed key establishment protocol that is specifically designed for WMNs. Our scheme makes use of EC-IBC, which decreases both the computation necessary to join the network and the bandwidth consumed within the network by eliminating the necessity of the certificate-based public key distribution. Moreover, the problems arising from the TTP assumption of the EC-IBC constructions are eliminated using ThSS, through which the trusted authority is eliminated via collaborative generation of the shared secrets. In fact, due to the assumption of not having mutual trust among the mesh nodes, we need to use a verifiable ThSS construction [43]. However, we implemented DKEM without considering this requirement. If we were to use a verifiable ThSS, then the key establishment latency would be higher, but the success percentage of user private key generation would not be affected.

In our solution, we utilize a variant of Shamir's threshold secret sharing. Since the mesh routers can be distinguished from the mesh clients by both the parameters they hold and the operations they perform, we can delegate the master private key share generation process to the mesh routers. With this construction, we can decrease the total number of nodes present in the master private key share generation process, which decreases the communication and the computational complexities of the system, as compared to the baseline protocol. Moreover, we assume that it is harder to compromise the mesh routers than compromising the mesh clients. With this assumption, we can both increase the number of shares required in the reconstruction processes by increasing the number of shares that the mesh routers hold and enforce the contribution of a mesh router in the reconstruction operations. As a consequence, the resiliency of the network can be increased without increasing the number of required neighboring nodes.

We explored the security of both baseline protocol and DKEM. Our analysis exposed that the baseline protocol is not secure against an insider semi-honest attacker. We showed that how an adversary can easily obtain the master private shares of other nodes and more vitally can capture the master private key of the system. Thereafter, we proved that DKEM is secure against both outsider and semi-honest insider adversaries in which the success probability of the attacker is only negligible.

We performed performance evaluation in order to show the effect of both the threshold value and the network size on the latency of key establishment and on the success percentage of user private key generation by using simulation. Results show that an increase in the network size increases both the latency of key establishment and the success percentage of user private key generation. For instance, at the threshold value of 8, an increase in the number of mesh nodes from 40 to 100 results in 5% increase in the successful user private key generation rate and 129% increase in the elapsed time. Our simulation results also show that increasing the threshold value decreases the success percentage of user private key generation and increases the key establishment latency. For instance, at the threshold value of 6, almost all of the mesh nodes can compute their own private keys within at most 1 min regardless of the network size. Moreover, for the worst case network scenario with 40 nodes performing at the threshold level of 16, at least 87% of the mesh nodes can compute their private keys within approximately 2.5 min. Hence, there is a trade-off between resiliency and efficiency: increasing the threshold value also increases the resiliency of the network but negatively effects its efficiency (e.g., key establishment latency increases and success percentage of user private key generation decreases).

Finally, we also simulated the baseline protocol. Results show that DKEM outperforms the baseline protocol when either latency of key establishment, success percentage of user private key generation or network resiliency is of concern. In conclusion, DKEM is much more efficient than the baseline protocol in all performance metrics.

References

References

[1] D. Karaoglan, A. Levi, E. Savas, A distributed key establishment scheme for wireless mesh networks using identity-based cryptography, in: Proceedings of the 6th ACM Workshop on QoS and Security for Wireless and Mobile Networks (Q2SWinet'10), ACM, New York, USA, 2010, pp. 11–18.

[2] M. S. Siddiqui, C. S. Hong, Security issues in wireless mesh networks, in: Proceedings of the International Conference on Multimedia and Ubiquitous Engineering, IEEE Computer Society, Washington, 2007, pp. 717–722.

[3] I. F. Akyildiz, X. Wang, W. Wang, Wireless mesh networks: a survey, Computer Networks 47 (4) (2005) 445–487.

[4] A. Khalili, J. Katz, W. A. Arbaugh, Toward secure key distribution in truly ad-hoc networks, in: Proceedings of the Symposium on Applications and the Internet Workshops, IEEE Computer Society, Washington, 2003, pp. 342–350.

[5] A. S. Wander, N. Gura, H. Eberle, V. Gupta, S. C. Shantz, Energy analysis of public-key cryptography for wireless sensor networks, in: Proceedings of the IEEE International Conference on Pervasive Computing and Communications, IEEE Computer Society, Washington, 2005, pp. 324–328.

[6] W. Du, R. Wang, P. Ning, An efficient scheme for authenticating public keys in sensor networks, in: Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing, ACM, New York, 2005, pp. 58–67.

[7] TR-2001-95, From Euclid's GCD to Montgomery Multiplication to the Great Divide, Sun Microsystems, Mountain View, CA, USA (2001).

[8] E. O. Blass, M. Zitterbart, Towards acceptable public-key encryption in sensor networks, in: Proceedings of the International Workshop on Ubiquitous Computing, INSTICC Press, Setubal, 2005, pp. 88–93.

[9] R. L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Commununications of the ACM 21 (1978) 120–126.

[10] WM-CS-2005-12, Telosb implementation of elliptic curve cryptography over primary field, College of William and Mary, Department of Computer Science, Williamsburg, VA (2005).

[11] B. Lo, G. Z. Yang, Key technical challenges and current implementations of body sensor networks, in: Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks, IEEE Computer Society, Washington, 2005, pp. 1–5.

[12] K. Malasri, L. Wang, Design and implementation of a secure wireless mote-based medical sensor network, in: Proceedings of the International Conference on Ubiquitous Computing, Springer-Verlag, New York, 2008, pp. 172–181.

[13] Y. T. T. Matsumoto, H. Imai, On seeking smart public-key distributions systems, Transactions of the IECE (Japan) E69-E (2) (1986) 99–106.

[14] P. V. O. W. Diffie, M. Wiener, Authentication and authenticated key exchanges, Designs, Codes and Cryptography 2 (2) (1992) 107–125.

[15] M. Girault, Self-certified public keys, in: Proceedings of Eurocrypt, Springer-Verlag, London, 1991, pp. 490–497.

[16] W. Diffie, M. E. Hellman, New directions in cryptography, IEEE Transactions on Information Theory IT-22 (6) (1976) 644–654.

[17] Z. You, X. Xie, A novel group key agreement protocol for wireless mesh network, Computers and Electrical Engineering 37 (2) (2011) 218–239.

[18] E. Bresson, O. Chevassut, D. Pointcheval, Provably secure authenticated group Diffie-Hellman key exchange, ACM Transactions on Information and System Security 10 (3) (2007) 10:1–10:45.

[19] K. Chatterjee, A. De, D. Gupta, Mutual authentication protocol using hyperelliptic curve cryptosystem in constrained devices, Network Security 15 (1) (2013) 9–15.

[20] W. Shi, P. Gong, A new user authentication protocol for wireless sensor networks using elliptic curves cryptography, Distributed Sensor Networks 2013 (2013) 2013:1–2013:7.

[21] G. Li, An Identity-based security architecture for wireless mesh networks, in: Proceedings of the IFIP International Conference on Network and Parallel Computing Workshops, IEEE Computer Society, Washington, 2007, pp. 223–226.

[22] L. Zhou, Z. J. Haas, Securing ad hoc networks, IEEE Network Magazine 13 (6) (1999) 24–30.

[23] J. Kong, P. Zerfos, H. Luo, S. Lu, L. Zhang, Providing robust and ubiquitous security support for mobile ad-hoc networks, in: Proceedings of the International Conference on Network Protocols, IEEE Computer Society, Washington, 2001, pp. 251–260.

[24] H. Dahshan, J. Irvine, An elliptic curve distributed key management for mobile ad hoc networks, in: Proceedings of Vehicular Technology Conference, IEEE Computer Society, Washington, 2010, pp. 1–5.

[25] M. Strasser, S. Capkun, C. Popper, M. Cagalj, Jamming-resistant key establishment using uncoordinated frequency hopping, in: Proceedings of the IEEE International Symposium on Security and Privacy, IEEE Computer Society, Piscataway, 2008, pp. 64–78.

[26] M. J. Miller, N. H. Vaidya, Leveraging channel diversity for key establishment in wireless sensor networks, in: Proceedings of the IEEE International Conference on Computer Communications, IEEE Computer Society, Piscataway, 2006, pp. 1–12.

[27] B. Zan, M. Gruteser, Random channel hopping schemes for key agreement in wireless networks, in: Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, IEEE Computer Society, Washington, 2009, pp. 2886–2890.

[28] H. Deng, A. Mukherjee, D. P. Agrawal, Threshold and Identity-based key management and authentication for wireless ad hoc networks, in: Proceedings of the International Conference on Information Technology: Coding and Computing, IEEE Computer Society, Washington, 2004, pp. 107–111.

[29] P. Guo, J. Wang, X. H. Geng, J.-U. Kim, A variable threshold-value authentication architecture for wireless mesh networks, Journal of Internet Technology 15 (6) (2014) 929–935.

[30] H. C. v. Tilborg, Encyclopedia of Cryptography and Security, Springer-Verlag, New York, Inc., Secaucus, NJ, USA, 2005.

[31] R. Blom, Non-public key distribution, in: Advances in Cryptology, Springer, US, 1983, pp. 231–236.

[32] R. Blom, An optimal class of symmetric key generation systems, in: Proceedings of the EUROCRYPT 84 Workshop on Advances in Cryptology: Theory and Application of Cryptographic Techniques, Springer-Verlag, New York, 1985, pp. 335–338.

[33] N. Ben Salem, J.-P. Hubaux, Securing wireless mesh networks, IEEE Wireless Communications 13 (2) (2006) 50–55.

[34] J. Jun, M. L. Sichitiu, The nominal capacity of wireless mesh networks, IEEE Wireless Communications 10 (5) (2003) 8–14.

[35] P. Gupta, P. R. Kumar, The capacity of wireless networks, IEEE Transactions on Information Theory 46 (2) (2000) 388–404.

[36] H. Cohen, A. Miyaji, T. Ono, Efficient elliptic curve exponentiation using mixed coordinates, in: Proceedings of the International Conference on

the Theory and Applications of Cryptology and Information Security: Advances in Cryptology, Springer-Verlag, London, UK, UK, 1998, pp. 51–65.

[37] D. M. Burton, The History of Mathematics / An Introduction, McGraw-Hill, New York City, NY, USA, 2011.

[38] R. de Clercq, L. Uhsadel, A. V. Herrewege, I. Verbauwhede, Ultra low-power implementation of ECC on the ARM Cortex-M0+, in: 51st ACM/EDAC/IEEE Design Automation Conference, 2014, pp. 1–6.

[39] C. K. Koc, T. Acar, B. S. Kaliski, Analyzing and comparing montgomery multiplication algorithms, IEEE Micro 16 (3) (1996) 26–33.

[40] T. V. Project, The Network Simulator NS-2: Documentation, Available at `http://www.isi.edu/nsnam/ns/ns-documentation.html` (2011).

[41] IEEE draft standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 7: Wireless access in vehicular environments, IEEE unapproved draft std P802.11p/D7.0 (2009).

[42] M. Scott, MIRACL: A Multiprecision Integer and Rational Arithmetic C/C++ Library, Shamus Software Ltd, Dublin, Ireland (2003).

[43] T. P. Pedersen, A threshold cryptosystem without a trusted party, in: Proceedings of the International Conference on Theory and Application of Cryptographic Techniques, Springer-Verlag, London, 1991, pp. 522–526.

Duygu Karaoğlan Altop is a Ph.D. candidate in Computer Science and Engineering at Sabancı University. She received the B.S. degree in Telecommunications Engineering and M.S. degree in Computer Science and Engineering from Sabancı University, in 2007 and 2009, respectively. She served as logistics chair of CSW (Computer Science Student Workshop) in both 2010 and 2011, and as publications chair of CSW in 2012. Her research interests include computer and network security, data and communication security, cryptography and biometrics.

Muhammed Ali Bingöl is a Ph.D. candidate in Computer Science and Engineering at Sabancı University. He received B.S. degree in Telecommunications Engineering and M.S. degree in Electronics and Communication Engineering from Istanbul Technical University, in 2008 and 2012, respectively. He has been also working as a senior researcher at TUBITAK BILGEM UEKAE (National Research Institute of Electronics & Cryptology) since 2008. His primary research interests include designing and analyzing cryptographic protocols, information security & privacy, lightweight cryptography and secure multi-party computation.

Albert Levi received B.S., M.S. and Ph.D. degrees in Computer Engineering from Boğaziçi University, Istanbul, Turkey, in 1991, 1993 and 1999, respectively. He served as a visiting faculty member in the Department of Electrical and Computer Engineering, Oregon State University, OR, between 1999 and 2002. He was also a postdoctoral research associate in the Information Security Lab of the same department. Since 2002, Dr. Levi is a faculty member of Computer Science and Engineering in Sabancı University, Faculty of Engineering and Natural Sciences, Istanbul, Turkey and founding co-director of the Cryptography and Information Security

Group (CISec) at Sabancı University. He has been promoted to associate professor in January 2008 and to full professor in May 2015. His research interests include computer and network security with emphasis on mobile and wireless system security, public key infrastructures (PKI), privacy, and application layer security protocols. Dr. Levi has served in the program committees of various international conferences. He also served as general and program co-chair of ISCIS 2006, general chair of SecureComm 2008, technical program co-chair of NTMS 2009, publicity chair of GameSec 2010 and program co-chair of ISCIS 2011. He is editorial board member of The Computer Journal published by Oxford University Press and Computer Networks published by Elsevier.

Erkay Savaş received the BS (1990) and MS (1994) degrees in electrical engineering from the Electronics and Communications Engineering Department at Istanbul Technical University. He completed the Ph.D. degree in the Department of Electrical and Computer Engineering (ECE) at Oregon State University in June 2000. He had worked for various companies and research institutions before he joined Sabancı University as an assistant professor in 2002. He is the director of the Cryptography and Information Security Group (CISec) of Sabancı University. His research interests include cryptography, data and communication security, privacy in biometrics, trusted computing, security and privacy in data mining applications, embedded systems security, and distributed systems. He is a member of IEEE, ACM, the IEEE Computer Society, and the International Association of Cryptologic Research (IACR).

Appendix A. An Example Network

We present below a simple WMN setting in order to provide an illustration of our DKEM protocol. During descriptions and explanations of the protocol, we use the example network setting depicted in Figure A.8, which consists of 2 mesh routers and 2 mesh clients. In this setting, mesh routers M_1 and M_2 are in the communication ranges of each other, mesh client C_3 is in the communication range of all the other mesh nodes, and mesh client C_4 is in the communication ranges of mesh router M_1 and mesh client C_3. For the sake of easy reading, in the example, each mesh router holds 2 master private key shares, each mesh client holds 1 master private key share and the threshold value is 4. However, note that due to the security reasons, a client should hold at least 2 master private key shares to be able to help the other clients (see Section 3).

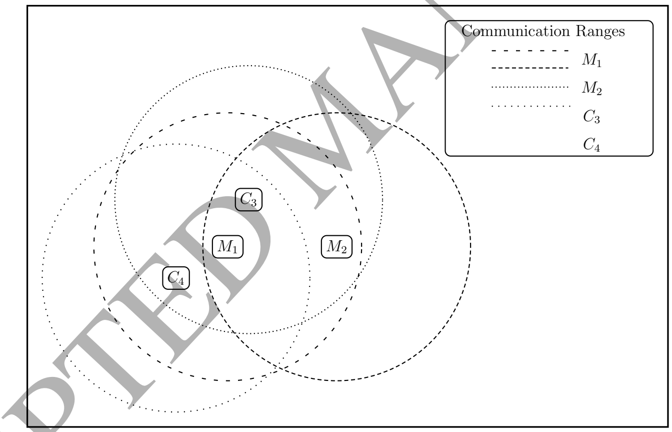


Figure A.8: An Example Network Model

Appendix A.1. Master private key share generation phase

Figure A.9 depicts an example for the master private key share generation phase of DKEM. After computing the subshares of the master private key, mesh routers M_1 and M_2 exchange these subshares with each other. When they receive all of their missing data, they compute their shares of the master private key and broadcast the *FINISH* message.

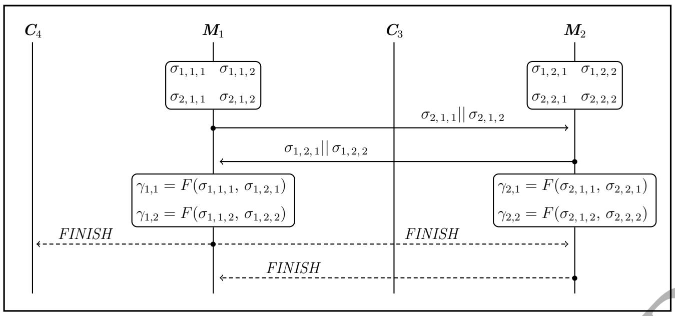

C_4
M_1
C_3
M_2
$\sigma_{1,1,1}$ $\sigma_{1,1,2}$
$\sigma_{2,1,1}$ $\sigma_{2,1,2}$
$\sigma_{1,2,1}$ $\sigma_{1,2,2}$
$\sigma_{2,2,1}$ $\sigma_{2,2,2}$
$\sigma_{2,1,1}||\sigma_{2,1,2}$
$\sigma_{1,2,1}||\sigma_{1,2,2}$
$\gamma_{1,1} = F(\sigma_{1,1,1}, \sigma_{1,2,1})$
$\gamma_{1,2} = F(\sigma_{1,1,2}, \sigma_{1,2,2})$
$\gamma_{2,1} = F(\sigma_{2,1,1}, \sigma_{2,2,1})$
$\gamma_{2,2} = F(\sigma_{2,1,2}, \sigma_{2,2,2})$
FINISH
FINISH
FINISH

Figure A.9: An Instance for the Master Private Key Share Generation Phase of DKEM - Figure A.8 depicts the corresponding network model, the number of shares that a mesh router holds is 2 (i.e., $x = 2$), and the dashed lines represent broadcast messages while the solid lines represent unicast messages.

Appendix A.2. Master private key share distribution phase

Figure A.10 depicts an example for the master private key share distribution phase of DKEM. After receiving a *FINISH* message from one of its neighbors, mesh client C_4 broadcasts a request message on its master private key share. This request message is received by mesh router M_1 and mesh client C_3. Since mesh client C_3 has not computed its share of the master private key yet, it saves this request. On the other hand, mesh router M_1 has already computed its shares of the master private key; thus, it replies to the request of the mesh client C_4 indicating that it can contribute to the distribution process with 2 partial shares. Then, mesh client C_3 also recognizes the end of the first phase and broadcasts a request message on its master private key share. This request message is received by all of the mesh nodes. Mesh client C_4 saves this request because it cannot send an immediate reply. However, having computed their master private key shares, mesh routers M_1 and M_2 reply to the request of the mesh client C_3 indicating that they can contribute to the distribution process with 2 partial shares. At this point, mesh client C_3 knows that it can receive 4 partial shares of the master private key; therefore, it broadcasts another request message indicating that the contribution of the mesh routers M_1 and M_2 are required.

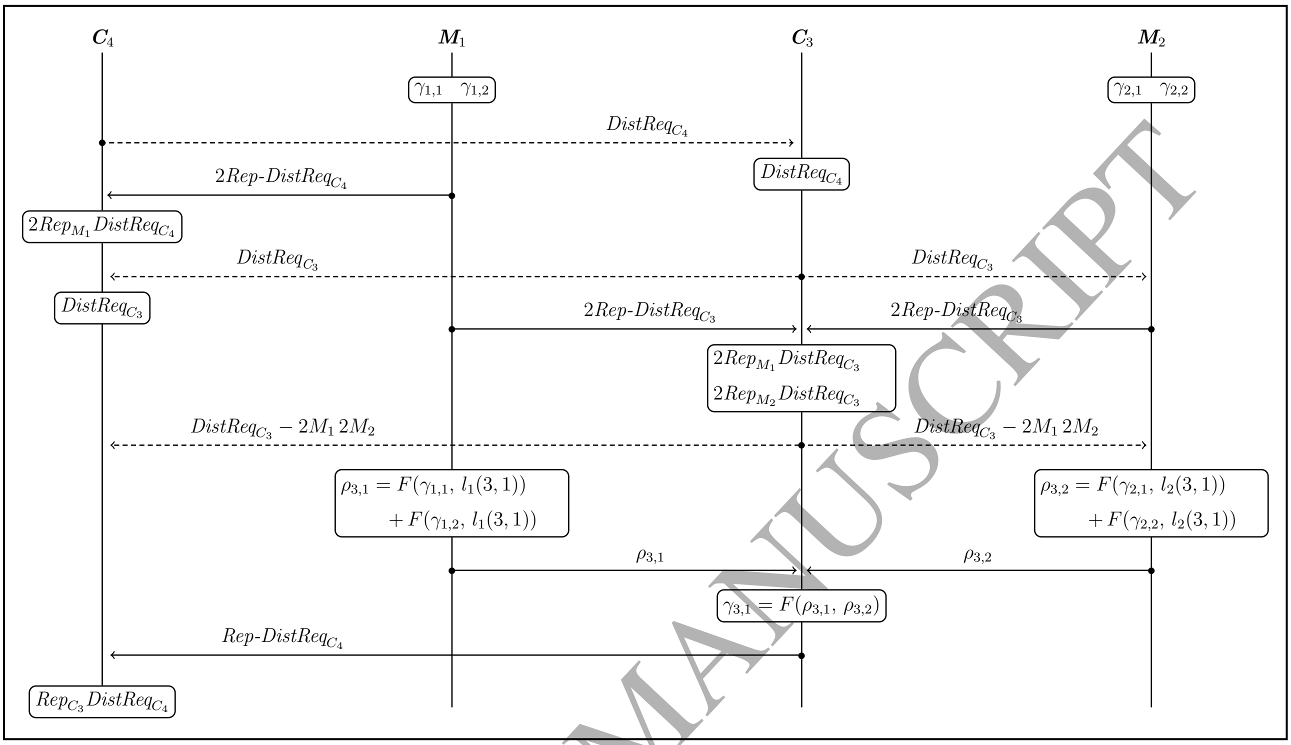
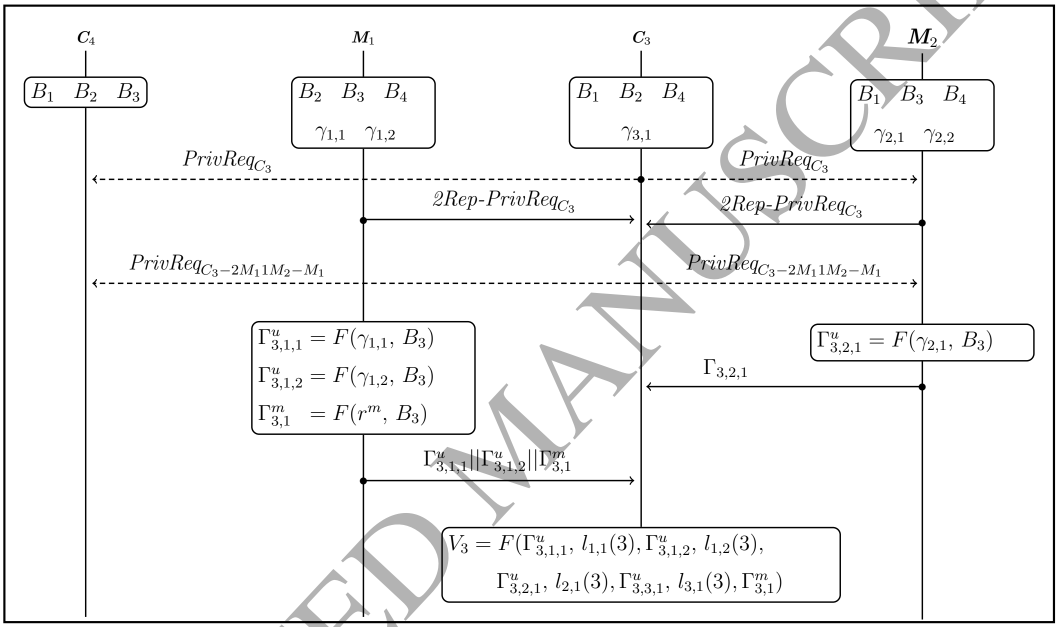

C_4
M_1
C_3
M_2
$\gamma_{1,1}$ $\gamma_{1,2}$
$\gamma_{2,1}$ $\gamma_{2,2}$
$DistReq_{C_4}$
$DistReq_{C_4}$
$2Rep\text{-}DistReq_{C_4}$
$2Rep_{M_1}DistReq_{C_4}$
$DistReq_{C_3}$
$DistReq_{C_3}$
$DistReq_{C_3}$
$2Rep\text{-}DistReq_{C_3}$
$2Rep\text{-}DistReq_{C_3}$
$2Rep_{M_1}DistReq_{C_3}$
$2Rep_{M_2}DistReq_{C_3}$
$DistReq_{C_3} - 2M_1\,2M_2$
$DistReq_{C_3} - 2M_1\,2M_2$
$\rho_{3,1} = F(\gamma_{1,1},\, l_1(3,1)) + F(\gamma_{1,2},\, l_1(3,1))$
$\rho_{3,2} = F(\gamma_{2,1},\, l_2(3,1)) + F(\gamma_{2,2},\, l_2(3,1))$
$\rho_{3,1}$
$\rho_{3,2}$
$\gamma_{3,1} = F(\rho_{3,1},\, \rho_{3,2})$
$Rep\text{-}DistReq_{C_4}$
$Rep_{C_3}DistReq_{C_4}$

Figure A.10: An Instance for the Master Private Key Share Distribution Phase of DKEM - Figure A.8 depicts the corresponding network model, the number of shares that a mesh router holds is 2 (i.e., $x = 2$), the number of shares that a mesh client holds is 1 (i.e., $y = 1$), the threshold value is 4 (i.e., $k = 4$), and the dashed lines represent broadcast messages while the solid lines represent unicast messages.

This second request message is also received by all of the mesh nodes. Upon receipt, mesh client C_4 discards the first request message of the mesh client C_3 that it has previously saved to respond after computing its master private key share, since its contribution is not required anymore. On the other hand, mesh routers M_1 and M_2 compute the partial shares of the mesh client C_3 and transmit them. As soon as mesh client C_3 receives the required partial shares, it computes its master private key share. Thereafter, mesh client C_3 replies to the request of the mesh client C_4, which it has previously saved, indicating that it can contribute to the distribution process with 1 partial share. Unfortunately, mesh client C_4 cannot compute its share of the master private key because of the fact that it can receive only 3 master private key partial shares from its neighbors.

Appendix A.3. User private key generation phase

Figure A.11 illustrates an example scenario for the user private key generation phase of DKEM. After computing its share of the master private key, mesh client C_3 broadcasts a request message on the generation of its own private key. This request message is received by all the other mesh nodes. Having computed their shares of the master private key, mesh routers M_1 and M_2 respond to this request indicating that they can contribute with 2 shares. At this point, mesh client C_3 knows that it can receive 3 shares of its own private key and the additional share. Thus, it broadcasts another request message indicating that the contribution of the mesh router M_1 with 2 shares along with the additive share and the mesh router M_2 with 1 share are required. This second request message is also received by all of the mesh nodes. Upon receipt, mesh routers M_1 and M_2 compute the corresponding shares and transmit them to the mesh client C_3. Finally, after receiving all the required data, mesh client C_3 constructs its user private key.



C_4
M_1
C_3
M_2
B_1 B_2 B_3
B_2 B_3 B_4 $\gamma_{1,1}$ $\gamma_{1,2}$
B_1 B_2 B_4 $\gamma_{3,1}$
B_1 B_3 B_4 $\gamma_{2,1}$ $\gamma_{2,2}$
$PrivReq_{C_3}$
$PrivReq_{C_3}$
$2Rep\text{-}PrivReq_{C_3}$
$2Rep\text{-}PrivReq_{C_3}$
$PrivReq_{C_3-2M_11M_2-M_1}$
$PrivReq_{C_3-2M_11M_2-M_1}$
$\Gamma^u_{3,1,1} = F(\gamma_{1,1}, B_3)$
$\Gamma^u_{3,1,2} = F(\gamma_{1,2}, B_3)$
$\Gamma^m_{3,1} = F(r^m, B_3)$
$\Gamma^u_{3,2,1} = F(\gamma_{2,1}, B_3)$
$\Gamma_{3,2,1}$
$\Gamma^u_{3,1,1}||\Gamma^u_{3,1,2}||\Gamma^m_{3,1}$
$V_3 = F(\Gamma^u_{3,1,1}, l_{1,1}(3), \Gamma^u_{3,1,2}, l_{1,2}(3), \Gamma^u_{3,2,1}, l_{2,1}(3), \Gamma^u_{3,3,1}, l_{3,1}(3), \Gamma^m_{3,1})$

Figure A.11: An Instance for the User Private Key Generation Phase of DKEM - Figure A.8 depicts the corresponding network model, the number of shares that a mesh router holds is 2 (i.e., $x = 2$), the number of shares that a mesh client holds is 1 (i.e., $y = 1$), the threshold value is 4 (i.e., $k = 4$), and the dashed lines represent broadcast messages while the solid lines represent unicast messages.