Strongly-Resilient and Non-Interactive Hierarchical Key-Agreement in MANETs

Rosario Gennaro* Shai Halevi* Hugo Krawczyk* Tal Rabin*
Steffen Reidt† Stephen D. Wolthusen†

April 4, 2008

Abstract

Key agreement is a fundamental security functionality by which pairs of nodes agree on shared keys to be used for protecting their pairwise communications. In this work we study key-agreement schemes that are well-suited for the mobile network environment. Specifically, we describe schemes with the following characteristics:

- *Non-interactive:* any two nodes can compute a unique shared secret key without interaction;
- *Identity-based:* to compute the shared secret key, each node only needs its own secret key and the identity of its peer;
- *Hierarchical:* the scheme is decentralized through a hierarchy where intermediate nodes in the hierarchy can derive the secret keys for each of its children without any limitations or prior knowledge on the number of such children or their identities;
- *Resilient:* the scheme is fully resilient against compromise of *any number of leaves* in the hierarchy, and of a threshold number of nodes in each of the upper levels of the hierarchy.

Several schemes in the literature have three of these four properties, but the schemes in this work are the first to possess all four. This makes them well-suited for environments such as MANETs and tactical networks which are very dynamic, have significant bandwidth and energy constraints, and where many nodes are vulnerable to compromise. We provide rigorous analysis of the proposed schemes and discuss implementations aspects.

*1. IBM, T.J. Watson Research Center Hawthorne, NY 10532, USA
†2. Royal Holloway, Department of Mathematics, Royal Holloway, University of London, United Kingdom

1 Introduction

Key agreement is a fundamental tool for secure communication; it lets two nodes in a network agree on a shared key that is known only to them, thus allowing them to use that key for secure communication.

In environments where bandwidth is at a premium, there is a significant advantage to *non-interactive* schemes, where two nodes can compute their shared key without any interaction. The classical (static) Diffie-Hellman key-agreement protocol [4] is an example of a non-interactive scheme: in that protocol, node A can compute a shared key with node B knowing only the public key of B (and its own secret key). But the nodes in this protocol must still learn each other's public keys which requires direct communication between them or some other form of coordination.

To minimize the required coordination, one may use *identity-based* key-agreement, where the public key of a node is just the node's name. Such schemes rely on a central authority with a master secret key, that provides each node with a secret key that corresponds to that node's name. In this setting, the non-interactive identity-based scheme of Sakai et al. [13] (which is based on bilinear maps) allows node A to compute a shared key with node B knowing only B's name (and A's own secret key).

However, it is often unrealistic to expect all nodes to register with just one central authority as required by Sakai et al. [13]. For example, in mobile ad-hoc networks (MANETs), one expects frequent communication between nodes from different organizational units. One would therefore prefer a *hierarchical* system, where a root authority only needs to distribute keys to a small number of large organizations, and each of these can further distribute keys to smaller and smaller units, until finally the end-nodes get their secret keys from their immediate orgamizational unit. Such a hierarchical scheme would serve well also for military applications where the organization of the network is already hierarchical in nature. (Indeed, key-agreement for MANETs and tactical networks served as our motivation and the starting point for this work.)

Our goal in this paper is to propose schemes that have all the above functional properties and are secure in a strong sense. That is, they are *non-interactive* to save on bandwidth, *identity-based* to save on coordination and support ad-hoc communication (see more on this below), and *hierarchical* to allow for flexible provisioning of nodes. At the same time, we design these schemes to be *fully* resilient to the compromise of *any number of end-users (leaf nodes)* and resilient to the compromise of a "threshold" of nodes in the upper levels of the hierarchy.

One elegant scheme that has the above three "functional" properties (but weaker security guarantees) was proposed by Blundo et al. [2] following the earlier work of Blom [1]. ([2] mainly deals with the non-hierarchical setting, but they also discuss an extension to the hierarchical case.) In this scheme (see Section 2.3), each node has a secret polynomial (in the role of a secret key), and the shared key between two leaf nodes is computed by evaluating the polynomial held by one node at a point that corresponds to the identity of the other. An alternative approach to building a hierarchical scheme is to start from a randomized key-predistribution schemes as in Eschenauer and Gligor [7], and extend it to a hierarchical scheme as in Ramkumar et al. [12] (see Section 2.4).

Both hierarchical schemes, however, have a significant limitation in applications where the end-users, or leaves, in the hierarchy are at a high risk of compromise (as in a MANET or military application). They guarantee security only as long as not too many of these nodes are compromised. Once the number of compromised nodes grows above some threshold, an attacker can learn keys of uncompromised nodes, and may even learn the master secret key of the whole system.

On the other hand, the identity-based key agreement scheme of Sakai et al. [13] provides resilience against the compromise of any number of leaf nodes, but, as mentioned earlier, it requires a central authority to hand out keys to each and every participant in the network including any participants joining the network at a later point.

1.1 Our Contribution

The main contribution of this work is in combining the best properties of the above schemes in order to offer a highly-functional and dynamic key agreement scheme that enjoys a very high level of resilience against node compromise. Specifically, we show how to combine a large class of hierarchical schemes (that includes the schemes from [2, 12])[1] with the (non-hierarchical) scheme of Sakai et al. [13], and obtain a hierarchical key-agreement scheme (KAS) that is fully resilient against compromise of any number of leaf nodes. In the upper levels of the hierarchy we preserve the property of the original hierarchical scheme, namely, resilience to the compromise of a threshold of nodes. We provide a rigorous security analysis for our modified hierarchical scheme in terms of the security of the original one. Namely, we prove that if the original hierarchical scheme was secure, then our modified scheme is also secure, *but this time also with respect to compromise of arbitrary number of leaf nodes.*

In many cases, this combination of threshold resilience in the upper levels with full resilience in the leaves is the right security trade-off: It is often the case that upper-level nodes are better protected (e.g., they are less mobile, have better physical security, etc.), while leaf nodes are both more numerous and much more vulnerable to attack.

For a hierarchy of depth $L+1$ and "security-threshold" t, the amount of secret information in the schemes in the literature (and thus also in our solutions) grows in the order of $(t^2/2)^L$. Hence these solutions can be used for moderate values of t and small values of L. However for many practical applications this is not necessarily a concern. For example, consider a military scenario where a central authority resides at the headquarters, the leaf nodes belong to individual soldiers, and the intermediate nodes are the various units. In this case the number of levels is likely to be relatively small and the same holds for the branching factor of the tree (except for the lowest level in the hierarchy where the number of leaves can be arbitrarily large). In this case the threshold t (which is never larger than the branching factors at levels above the leaves) and depth L are both relatively small.

Another very important property of our solution is that nodes can be added to the hierarchy, by their parents, without requiring any further coordination with other nodes and without changing the information held by other nodes. In particular, there is no limitation on the number of children a node can have. Furthermore, our scheme allows for a threshold of siblings to add a new sibling without requiring the parent participation. These properties highlight the decentralized and dynamic nature of our schemes which is central for many of the ad-hoc networking applications that motivate this work.

Another source of flexibility in our schemes comes through the use of identity-based techniques. As said, these techniques free the key-agreement schemes from the need for distribution of public keys (and of revocation lists). Next, we discuss these (and other) benefits in more detail.

Benefits of our identity-based solutions. As we pointed out earlier, non-interactive key agreement can be achieved without resorting to the bilinear maps used in [13] by using traditional static Diffie-Hellman exchange. This, however, requires each party to have the peer's public key before

[1]To wit, the hierarchical schemes that can be combined in this way are those in which all the secret keys are obtained as *linear combinations* of some base elements selected by the root authority (see Definition 1 in Section 3).

they can compute a shared key. Depending on the setting, this may necessitate of a centralized certification authority or, in hierarchical settings as ours, it requires the ability of nodes to cross-certify each other or verify a certificate chain. Moreover, most systems will require some form of large-scale coordination and communication (possibly on-line) to propagate certificate revocation information. Identity-based schemes significantly simplify deployment by eliminating the certification issues. All a party needs to know in order to generate a shared key is its own secrets and the identity of the peer (clearly, the need to know the peer's identity exists in any scheme including a certificate-based one where certificates bind identities to public keys). In particular, in identity-based systems, identities may have a semantic meaning that identifies their function and attributes without need for further certification. For example, in a vehicular system a service point in the road may be identified by the location of that point (e.g., "traffic monitor at coordinate points x and y"), or in a military application the identity could be "commander of xyz unit", etc. A device that needs to communicate securely with such points only needs to know their "functional identities". In addition, functional identities can include other attributes such as a date of validity; the latter makes keys short-lived and hence less dependent on revocation. When, for instance, party P's identity includes a time period, P will need to obtain a new secret key from its parent when the period expires; this however does not require coordination or information exchange with any other node[2].

Simulative Validation For MANETs, particularly tactical networks, performance is a prime concern. However, key factors contributing to the communication complexity of a protocol are difficult to capture analytically. We have therefore implemented the distribution scheme and simulated its performance in a platoon-level operation in an urban area to adequately represent the impact of limited and fluctuating connectivity on key distribution performance. It should be noted, however, that the performance estimates given in Section 4 are only a qualitative guide to performance on typical MANET devices.

1.2 Related Work

In the context of non-interactive identity-based key agreement, we already mentioned the works of Sakai et al. [13], Blundo et al. [2], and Eschenauer and Gligor [7] (and its extension by Ramkumar et al. [12]), which play a central role in our construction.

There were also a few prior attempts to improve the resilience of the scheme of Blundo et al. Hanaoka et al. [8] show that in a sparse system (where most pairs of nodes never need to communicate) the threshold can be increased by a significant factor (possibly up to 16 fold) without adversely effecting the performance. That solution is applicable in relatively static networks where one can partition the nodes into disjoint sets and have no inter-set communication, but it is not applicable in settings where every pair of nodes may potentially need to communicate.

Another technique for improving the resilience of the Blundo et al. scheme was proposed by Zhang et al. [18], using random perturbations in order to randomize the polynomials used in Blundo et al. However, a practical instantiation of the parameters for the protocol enables the parties to agree on a small number of bits (say 12) in each execution of the protocol. Thus, in order to generate enough secret keying material about ten independent executions of the protocol need to be carried out. Furthermore, this scheme does not provide the hierarchical capabilities.

[2]As an example, when our scheme is instantiated with multivariate polynomials, each leaf could get from its parent, once every period, a secret derived by evaluating a polynomial on a point of the form $Hash(\mathrm{LeafId}||date)$.

Matt [11] described some trade-offs between resilience and performance, and even proposed a combination of the schemes of Blundo et al. and Sakai et al. However, his scheme requires that each node *communicates directly with the central authority*, and hence it is not a hierarchical scheme.

Following the identity-based encryption scheme of Boneh and Franklin [3], Horwitz and Lynn [9] initiated a study of hierarchical identity-based encryption. Interestingly, their scheme combines a pairing-based scheme and a polynomial-based one as we do. However, they only use two levels where the pairing-based scheme is placed at the top level and the polynomial-based scheme at the second level. In this work we reverse the order, using the polynomial-scheme for all the top levels and the pairing-based scheme only for the leaves to obtain a solution that supports non-interactive key agreement (encryption functionality as in [9] can support key agreement but requires interaction).

Open question. It would be interesting to have a hierarchical *non-interactive* key agreement scheme where resilience is achieved not only against any number of corruptions in the leaves (as we do) but also against any number of corruptions in the higher levels of the hierarchy. Note that this can be achieved with our solution by setting the threshold in upper levels of the hierarchy to the number of children in each level. The drawback of this solution is that it becomes impractical with large thresholds (see above). Also, such a scheme loses one of the important benefits of our scheme, namely, the possibility of adding new nodes to the hierarchy without influencing or changing the information held by other nodes. One hopes that a better solution could be achieved by developing a full hierarchical scheme solely based on pairing cryptography similar to known schemes for hierarchical identity-based encryption. The search for such a solution is one of the more interesting problems left open by our work.

1.3 Alternatives to Non-Interactive Key Agreement

One can argue that using non-interactive key agreement does not really eliminate interaction since the shared key must be used for communication at some point (or else why compute it at all). According to this view, the effect of non-interactive key agreement can also be obtained with encryption and signatures: Simply have the initiator of the communication send an encrypted (under the recipient's public key) and signed secret key along with the first communication flow, and thereafter the nodes can use that key to secure further communication.

We point out, however, that using non-interactive key agreement offers some important advantages, most significantly the saving of bandwidth (and energy). Indeed, using encryption and signatures as above entails additional communication of at least a few dozen (or a few hundred) bytes with the first communication flow. In environments where bandwidth and energy are very limited, this additional overhead may be significant. In tactical networks another benefit of our non-interactive solution is reducing the detectability (e.g., via RF emissions) of mobile nodes.

In addition, one can envision applications where the shared key is used for purposes other than just securing a traditional communication channel between the two peers. For example, consider using the shared key to establish a steganographic channel between the peers, trying to hide not only the content of communication but also its very presence. In this case, one cannot simply use encryption, since that first encrypted message would be detected. Having a shared key already in place allows the peers to establish a steganographic channel between them.

Another case where non-interactive key agreement is needed, is when the shared key provides a shared randomness between the peers, even though the two end points are never meant to interact directly with each other. For illustration, consider two nodes A and B that need to perform some measurement and report it to node C. Node C needs to compute the average of the two values, but we want to hide from it the actual measurements. One way to achieve this is for A and B to "blind" their measurement by adding/subtracting a blinding factor that is derived from their shared secret key. Since they both use the same number then C can still compute the average. But since C does not know the blinding factor then it cannot recover the original measurements.

2 Preliminaries

Our key-agreement schemes (KAS) are built by combining the identity-based key agreement protocol of Sakai et al. [13] with hierarchical schemes that use linear operations, such as the polynomial-based key distribution system of Blundo et al. [2] or the random-subset-based scheme. Below we present some background material and recall these schemes.

2.1 Bilinear Maps and the BDDH Assumption

Let G_1 and G_2 be two cyclic groups of order q for some large prime q. Let e be a mapping $e : G_1 \times G_1 \rightarrow G_2$. The mapping e is:

1. Bilinear if $e(P^a, Q^b) = e(P, Q)^{ab}$ for any $P, Q \in G_1$, $a, b \in Z_q$.
2. Non-degenerate if e does not send all pairs to the identity in G_2.
3. Computable if there is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.

Bilinear mappings that can be computed efficiently are known based on Weil and Tate pairings in Abelian varieties.

Bilinear Decisional Diffie-Hellman Problem (BDDH).
The central hardness assumption on which we base our schemes is the following BDDH assumption introduced by Boneh and Franklin [3]. Let G_1, G_2 and e be as above. Given a random $P \in G_1$, $P^a, P^b, P^c \in G_1$ for random $a, b, c \in Z_q$, and given $h \in G_2$, it is hard to distinguish the case where $h = e(P, P)^{abc}$ from the case where $h = e(P, P)^r$ for a random and independent $r \in Z_q$. Formally, an algorithm $\mathcal{A}$ has advantage ϵ in solving the BDDH in $\langle G_1, G_2, e\rangle$ if

$$Pr[\mathcal{A}(P, P^a, P^b, P^c, e(P, P)^{abc}) = 1] - Pr[\mathcal{A}(P, P^a, P^b, P^c, e(P, P)^r) = 1] \geq \epsilon$$

where the probability is over the random choice of $P \in G_1$, $a, b, c, r \in Z_q$, and the internal randomness of $\mathcal{A}$. The BDDH assumption (with respect to $\langle G_1, G_2, e\rangle$) states that feasible adversaries can have only an insignificant advantage.[3]

2.2 Non-Interactive Identity Based Key Agreement

Sakai et al. [13] propose the following non-interactive (but not hierarchical) key-agreement scheme. The central authority sets up the parameters for an identity based public key system, by fixing two cyclic groups G_1, G_2 and the bilinear map $e : G_1 \times G_1 \rightarrow G_2$. Furthermore, it chooses a cryptographic hash function $H : \{0, 1\}^* \rightarrow G_1$. It then chooses a secret key $s \in Z_q$ and provides a node with identity ID with the secret key $S_{ID} = H(ID)^s \in G_1$.

[3]In this extended abstract we forgo the asymptotic notations that are needed to make this formal. Instead we take the "concrete security" approach, directly relating the advantage of an adversary against our scheme to the advantage in solving BDDH over the relevant group.

The shared key between two nodes with identities ID_1 and ID_2 is $K = e(H(ID_1), H(ID_2))^s \in G_2$, which party ID_1 computes as $K = e(S_{ID_1}, H(ID_2))$ and ID_2 computes as $K = e(H(ID_1), S_{ID_2})$.

The security of this scheme can be reduced to the BDDH assumption in the random-oracle model, as was shown in [6].

2.3 Polynomial Based KAS

Our generic key-agreement scheme (KAS) presented in Section 3 can be instantiated using different hierarchical systems. Here and in the next subsection we describe two instantiations of such hierarchical systems. The first is based on multivariate polynomials and follows Blundo et al. [2] (we will refer to it as Blundo's scheme). Let L be the depth of the hierarchy, i.e., the nodes are arranged in a tree with L levels. Each node's identity corresponds to the path from the root to the node (thus a node at level i will have as identity a vector with i components $\langle I_1, \ldots, I_i \rangle$ where each I_j is an integer).

For desired threshold parameters $\{t_i : i \leq L\}$, the root authority chooses a random polynomial (over Z_q for a large enough prime q) $F(x_1, y_1, \cdots, x_L, y_L)$, where the degree of x_i, y_i is t_i. F is chosen such that $F(x_1, y_1, \cdots, x_L, y_L) \equiv F(y_1, x_1, \cdots, y_L, x_L)$, i.e. F is symmetric between the x'es and y's. One way to choose such polynomial is to choose a random polynomial f on the same variables, and then set $F(x_1, y_1, \cdots, x_L, y_L) = f(x_1, y_1, \cdots, x_L, y_L) + f(y_1, x_1, \cdots, y_L, x_L)$. We note that the size of the description of F (number of coefficients) is $\prod_{i=1}^{L} \frac{(t_i+1)(t_i+2)}{2}$ (the half is due to the symmetry of the polynomial), so this scheme can only be used with moderate thresholds t_i and small values of L.

The master secret key of the system is the polynomial F itself. The secret key of node with identity I in the first level of the hierarchy is the polynomial $F_I = F(I, y_1, x_2, y_2, \cdots)$ that has $2L - 1$ variables. Similarly, the secret key of a node at level i with identity $\vec{I} = \langle I_1, \ldots, I_i \rangle$ is the polynomial $F_{\vec{I}} = F(I_1, y_1, \cdots, I_i, y_i, x_{i+1}, y_{i+1}, \ldots)$ that has $2L - i$ variables, and the secret key of the leaf with identity $\langle I_1, \ldots, I_L \rangle$ is the polynomial in L variables $F(I_1, y_1, \cdots, I_L, y_L)$.

The shared key between the two leaf nodes $\langle I_1, \ldots, I_L \rangle$ and $\langle J_1, \ldots, J_L \rangle$ is the value of the polynomial $F(I_1, J_1, \ldots, I_L, J_L) = F(J_1, I_1, \ldots, J_L, I_L)$, that each node can compute by evaluating its secret polynomial on the points that correspond to its peer's identity.

Blundo's scheme provides information theoretic security for uncompromised nodes in the following important way. We call a node *compromised* if the attacker has learned *all* of the node's secrets (i.e., all the coefficients of the polynomial the node holds, and hence all of its descendants' shared keys), otherwise we call it *uncompromised*. Blundo's scheme guarantees that the key shared between any two uncompromised nodes is information theoretically secure, namely, all values of the key are equally possibly given the attacker's view.

Note that a node N in the hierarchy can be compromised (i.e., all its secrets learned) by directly breaking into N and finding its secrets or by breaking into other nodes from which the information in N can be reconstructed. For example, one can learn all of N's secrets by breaking into an ancestor of N or by breaking into $t + 1$ of its children (where t is the node's threshold). Here, the word "secrets" can refer to the coefficients of the polynomial held by a node N or, equivalently, to the set of pairwise shared-keys known to N and its descendants (i.e., the set of keys shared by these nodes with every other node in the hierarchy). In general, since pairwise keys are derived by evaluating a polynomial, the knowledge of a set of secrets (coefficients and/or pairwise keys) can allow an attacker to derive the value of additional secrets. Given a set of secrets S, we say that a key K (e.g., between parties I and J) is *independent from S* if no attacker (even if computationally unbounded) can learn anything about K from the set S; we say that a set of keys S is independent if each key in it is independent of the other keys in the set. It can be shown that in a Blundo's hierarchy with $L+1$ levels (with the root being at level 0 and the leaves at level L) and threshold t_i at level i, an attacker that wants to learn all the secrets of a node N in level ℓ must learn (at least) a set of T independent keys where $T = \prod_{i=\ell+1}^{L} \frac{(t_i+1)(t_i+2)}{2} \prod_{i=1}^{\ell}(t_i+1)$. In particular, the attacker must learn *at least* this many number of keys (or coefficients) in the system before it can learn all of N's secrets.[4]

2.4 Subset Based KAS

A different instantiation of our KAS uses subset-based key pre-distribution schemes, which were first studied by Eschenauer and Gligor [7]. Roughly, in this protocol the root authority chooses a large number of secret keys for its key-ring, the key-ring of every node contains a random subset of these keys, and the shared key for two nodes is computed from the intersection of the keys in their respective key-rings.

Extending it to a hierarchical ID-based scheme is fairly straightforward: a parent node in the tree gives to each child a random subset of its key-ring, and that subset is computed deterministically from the child's name (using a cryptographic hash function). Such a hierarchical scheme was described by Ramkumar et al. [12].

In a few more details, the scheme would work as follows:

- The parameters of the system are the number of keys at the root (denoted N), and for each level i in the tree a probability $p_i \in (0,1)$ that says what fraction of the key-ring of the parent is forwarded to the children.
- The root node chooses N secret keys at random for its key-ring. For our purposes, we think of these keys as integers modulo a fixed large prime number q.
- Let $n = \langle I_1, \ldots, I_i \rangle$ be an internal node at level i with key ring $R_n = \{K_1, K_2, \ldots\}$, and let $c = \langle I_1, \ldots, I_i, I_{i+1} \rangle$ be a child of n in the tree. The node n uses a cryptographic hash function to derive a sequence of numbers from the child's name, say by computing: $r_j \leftarrow H(c, j)$, where r_j's are numbers between 0 and 1. The child c gets all the keys $K_j \in R_n$ for which $r_j < p_i$. Namely, its key-ring is $R_c = \{K_j \in R_c \ : \ r_j < p_i\}$.
- For two leaf nodes $\langle I_1, \ldots, I_L \rangle$ and $\langle J_1, \ldots, J_L \rangle$ the nodes repeat the hash calculations from above to determine the intersection of their key rings, and the shared key is computed (say) as the sum modulo q of all the keys in the intersection.

It is not hard to show that in order to withstand up to t_i compromised nodes at level i, the optimal setting for the parameter p_i is $p_i = 1/(t_i+1)$. And given all the t_i's and p_i's, the parameter N should be set large enough to ensure the required level of security. Specifically, to ensure that an attacker that compromises up to t_i nodes in each level i will not have more than e^{-m} probability of learning the shared key between two specific uncompromised nodes, the parameter N should be set to $N = \lceil m / \prod_i p_i^2 (1-p_i)^{t_i} \rceil \approx m e^L \cdot \prod_i t_i(t_i+1)$. To ensure that the attacker will have probability at most e^{-m} to learn the key of *any* pair of uncompromised nodes, we need to add to the number N above $2 \log M$ where M is the number of nodes in the system.

[4]When all t_i's are equal to the same number t we have $T = (\frac{(t+1)(t+2)}{2})^{L-\ell}(t+1)^{\ell}$.

3 Our Fully Leaf-Resilient KAS

Our goal is to provide a hierarchical identity-based key agreement scheme that is secure against compromise of any number of nodes at the lowest level of the hierarchy. Namely, we consider a key-agreement scheme (KAS) in the form of a tree-like hierarchy of authorities that issue keys to nodes lower in the hierarchy, where any two leaf nodes can compute *without interaction* a shared key unique to these two leaves. (That is, each leaf computes the shared key from its own secret key, its peer's identity, and potentially some other public information).

We want this hierarchy to be secure in the sense that an attacker that compromises some of the nodes in the hierarchy cannot learn the keys shared by leaves that are not in the subtree of a compromised nodes. Typically, the above guarantee of security will only hold as long as the attacker does not compromise too many nodes, and we will extend this guarantee even in the face of unlimited number of compromised leaves.

Technically, our scheme is a combination of *linear* hierarchical schemes (of which the schemes from Sections 2.3 and 2.4 are special cases) with the identity-based scheme of Sakai et al. that was described in Section 2.2. In the rest of this section we formalize the linear requirement from the underlying hierarchical KAS and then present our hybrid scheme.

Definition 1 (Linear Hierarchical KAS) *A hierarchical key-agreement scheme is called* linear *if it satisfies the following properties with respect to some linear space V and an integer parameter N: (i) The root authority selects N random elements from V to be used as the* master secret keys. *(ii) The secret key of each node in the hierarchy consists of a set of values $v_1, v_2, \ldots \in V$, each of which is a* linear combination *(over V) of the master secret keys. (iii) The shared key between every two nodes is an element of V which is also a linear combination over V of the master secret keys. (iv) The number of values v_i in each node and the coefficients in the linear combinations that determine these values are derived* deterministically *from public information such as the position of a node in the hierarchy and its identity.*

We note that in typical hierarchical schemes, an internal node will provide its children with values that are linear combination of its own values (which thus must be linear combinations of the master secret keys). This is indeed the case for the two schemes from Sections 2.3 and 2.4.

3.1 A Leaf-Resilient Hybrid Hierarchical KAS

We now show how to combine a linear hierarchical KAS $\mathcal{H}$ with the bilinear identity-based scheme of [13] (Section 2.2), resulting in a hybrid scheme, $\mathcal{H}'$, that is as resilient to attack on the *internal* nodes as $\mathcal{H}$ is, but which is *fully resilient* against leaf compromise. Roughly, a leaf node with identity ID can compute the shared key "in the exponent", thereby obtaining the secret $H(ID)^s$ as needed for the scheme of Sakai et al.

In more details, let $\mathcal{H}$ be an L-level linear hierarchical KAS, and we construct an $L+1$-level hybrid KAS $\mathcal{H}'$ as follows:

- The root authority of $\mathcal{H}'$ sets up and publishes the parameters for an identity based public key system, by fixing two cyclic groups G_1, G_2 of order q and the bilinear map $e : G_1 \times G_1 \to G_2$, as well as a hash function $H : \{0,1\}^* \to G_1$.

  In addition, the root authority carries the same actions as the root authority of $\mathcal{H}$, where the linear space over which $\mathcal{H}$ is defined is set to Z_q.

- For any internal node other than the root, a leaf or a parent of a leaf, all actions are identical to the scheme $\mathcal{H}$.
- A node F that is a parent of a leaf has secret values $v_1, \ldots, v_n \in Z_q$ as in $\mathcal{H}$. For each child leaf ℓ with identity ID_ℓ,[5] the values that F provides to ℓ are the elements $H(ID_\ell))^{v_i} \in G_1$, $i = 1, \ldots, n$.
- The shared key between leaf nodes ℓ, ℓ' with identities ID, ID' whose parents are F, F', respectively, is computed as follows:

  Let $v_1, \ldots, v_n$ be the secret key of F, and let $\alpha_1, \ldots, \alpha_n$ be the coefficients of the linear combination that F would have used in $\mathcal{H}$ to compute a shared key with F'. (In other words, F would have computed the shared key with F' in $\mathcal{H}$ as $s = \sum_i \alpha_i v_i \pmod q$.) Recall that the secret key of ℓ are the group elements $V_1 = H(ID)^{v_1}, \ldots, V_n = H(ID)^{v_n} \in G_1$, and that the coefficients α_i can be computed from publicly available information. The leaf ℓ computes

  $$U_1 \leftarrow \prod_i V_i^{\alpha_i} \ \left(= H(ID)^{\sum_i \alpha_i v_i} = H(ID)^s\right)$$

  and $U_2 \leftarrow H(ID')$, and sets the shared key to $K \leftarrow e(U_1, U_2) = e(H(ID), H(ID'))^s$. Similarly the leaf ℓ' with secret key $V'_1, \ldots, V'_{n'}$ determines the coefficients $\beta_1, \ldots, \beta_{n'}$ that F' would have used in $\mathcal{H}$, then computes $U'_1 \leftarrow H(ID)$ and $U'_2 \leftarrow \prod_i (V'_i)^{\beta_i}$ and sets $K \leftarrow e(U'_1, U'_2) = e(H(ID), H(ID'))^s$.

(For example, when applying this hybrid to the subset scheme from 2.4, the two leaves will determine the set of indexes I for which they both received keys, and then the leaf ℓ will compute $U_1 \leftarrow \prod_{i \in I} V_i$ and the leaf ℓ' will compute $U'_2 \leftarrow \prod_{i \in I} V'_i$.)

Security. A rigorous analysis and proof of the above generic hybrid scheme is presented in Section 5. We first discuss practical implementation issues.

4 Implementation

There are many trade-offs that one can make when choosing a key-agreement scheme for a particular application. Below we describe some of these trade-offs:

4.1 Setting the thresholds

The complexity of the schemes that we present here depends on the product $\prod_i t_i$, so to get a realistic scheme one must choose the t_i's as small as the security considerations allow. As was explained in the introduction, if the hierarchy is expected to only have a very small branching factor (except for the leafs) then one can set the t_i's to that expected branching factor. Otherwise, it sometimes makes sense to assume that higher-level nodes are better protected than lower-level nodes, and thus the thresholds t_i should increase as we go down the tree.

Below we demonstrate the complexity that we get for two settings, both of which correspond to a hierarchy that has two levels of intermediate nodes (i.e., the leaves are three levels below the

[5]We assume that the identity includes the entire path from the root of the hierarchy to the leaf, so no two leaves have the same identity.

root). The first setting is applicable to a very small tree, where we set $t_1 = t_2 = 3$. The second setting is applicable to large tree, where we use $t_1 = 7$ and $t_2 = 31$. The resulting key-sizes and number of operations to compute the shared key are summarized in Table 1.

4.2 Polynomials vs. Subsets

The two underlying hierarchical schemes from Sections 2.3 and 2.4 offer quite different characteristics. The main advantage of the polynomial scheme is that the secret keys are considerably smaller: for the same setting of the thresholds, the polynomial scheme has the leafs holding keys of size $\prod_i (t_i + 1)$ group elements, and the root holding a key of size $\prod_i \frac{(t_i+1)(t_i+2)}{2}$ (see Section 2.3). In the subset scheme, on the other hand, the size of the keys at the root is larger by roughly a factor of $m(2e)^L$ for security level of e^{-m} (in the leaves the factor is me^L). In our examples with $L = 2$, and assuming $m = 20$ (which seems a reasonable value), this means that the keys in the subset scheme are larger by about two orders of magnitude.

On the other hand, computing the shared key between two leaves may be faster using the subset construction. This is because in the polynomial scheme the leaves have to do one elliptic-curve multiplication for every group element in their key, whereas in the subset scheme they only need to do an elliptic-curve addition for every element in the intersection of the two sets (which is a small fraction of the entire key of each of them).

Another difference is the security behavior: the polynomial scheme ensures security as long as the adversary does not exceed the threshold of nodes compromised, but can break completely once the threshold is exceeded. The subset construction, on the other hand, provides a gradual degradation of security, with the probability of a break monotonically increasing as the adversary compromises more nodes.

Finally, we comment that one can also use hybrids between the two schemes, such as using the subset construction on one level and the polynomial construction on the other. Such hybrids are discussed in the works of Du et al. [5] and Liu and Ning [10].

4.3 Choosing the curves

Another non-trivial choice is the elliptic curves for the pairing operation. As is the case for the scheme of Sakai et al., our schemes require using "symmetric" pairing, where the two arguments to the pairing function are taken from the same group. These pairing are known for super-singular elliptic curves. Using Tables 1-2 from [14] as our point of reference, the curves to use for different security levels and their respective performance characteristics are summarized in Table 2.

Scheme:	*Polynomial scheme*		*Subset scheme*	
Thresholds:	$t_1 = t_2 = 3$	$t_1 = 7, t_2 = 31$	$t_1 = t_2 = 3$	$t_1 = 7, t_2 = 31$
Key-size (# of group elements)	Root: 100 Leaves: 16	Root: 19008 Leaves: 256	Root: 28768 Leaves: 1800	Root: 8930800 Leaves: 35000
Shared key Computation	1 pairing 16 EC mult's	1 pairing 256 EC mult's	1 pairing 450 EC add's 1800 hashing	1 pairing 1100 EC add's 35000 hashing

Table 1: Performance characteristics of hierarchical schemes: Subset numbers are with respect to security level $e^{-20} \approx 2 \times 10^{-9}$. (Add's and mult's stand for 'additions' and 'multiplications', resp.)

4.4 Concrete implementations

Combining the numbers from Tables 1 and 2 (and assuming that the SHA256 hashing operation takes about one microsecond on a 2.4 GHz Pentium-4 platform), the running times and storage requirements for the various schemes are summarized in Table 3. As is evident by these tables, the polynomial scheme offers much smaller keys, while the subset construction is faster for the leaves (but slower for the parents of the leaves).

In addition to the operations listed in Table 3, one also needs to implement the key-generation by the root and key-delegation between internal nodes. At key-generation time the root just needs to choose random numbers between 1 and q, one for every group element, where the prime number q is the order of the elliptic curve over which this scheme is implemented. For the curves that we deal with, the prime q is in the range from $q \approx 2^{160}$ to $q \approx 2^{300}$. For the polynomial scheme the time and space requirements are insignificant, and even for the subset scheme this is manageable. (At worse, with the parameters $t_1 = 7, t_2 = 31$ and working over a large curve, the root needs to generate 327 MByte of random data.)

Delegating between intermediate nodes in the subset scheme consists only of hashing (in order to determine which keys to delegate). With the parameter $t_1 = 7$, the root needs to do one hash calculation for every about 85 numbers in its key ring (since we only need three bits per number in order to select it with probability 1/8, and one application of SHA256 yields 256 bits). Hence the root needs to do only about 100000 hashing operations, which can be done in about 0.1 seconds. The intermediate nodes need to do even less work to compute the key delegation. (On the other hand, the keys in the subset scheme are large so key delegation may take considerable bandwidth.)

Delegating between intermediate nodes in the polynomial scheme requires evaluating polynomials modulo q. Specifically, every element that a node at level i delegates to a child is obtained as the result of evaluating a polynomial of degree t_i modulo q, which roughly means performing t_i modular multiplications. Since we work with small t_i's and moderate values of q, and since the secret keys in the polynomial schemes consists of at most a few thousand elements, then this is a rather short calculation. In our more computationally-demanding example, with parameters $t_1 = 7, t_2 = 31$, the root needs to evaluate 8192 polynomials of degree seven (for a total of about 60000 multiplications modulo a 160-bit to 300-bit prime). Extrapolating from reported speeds of modular exponentiations, this can be done in well under one second. (For example, the implementation of DSA in `openssl` was reported to performs one 160-bit exponentiation modulo a 512-bit prime in 0.8 millisecond on a 2.4 GHz Pentium-4. Hence a multiplication modulo a 300-bit prime should take no more than 2-3 microsecond, and 60000 of them can be done in under 0.2 seconds.)

Security	EC	Addition	Multipl.	Pairing	El.-size
RSA-912	SS(163,-)	0.1ms	15ms	57ms	260
RSA-1080	SS(193,-)	0.12ms	22ms	86ms	307
RSA-1976	SS(353,-)	0.3ms	94ms	355ms	561

Table 2: Elliptic-curve parameters from [14]. Security level is the (very rough) equivalence in RSA security. $SS(n, -)$ is the curve $Y^2 = X^3 - X - 1$ over $GF(3^n)$. Running times are in milliseconds on a 2.4GHz Pentium 4. Addition time is an estimate based on the timing of multiplication. Element-size is the number of bits to represent a point on the curve.

Scheme:	*Polynomial scheme*		*Subset scheme*	
Security level:	$t_1 = t_2 = 3$, RSA-912	$t_1 = 7, t_2 = 31$, RSA-1976	$t_1 = t_2 = 3$, RSA-912	$t_1 = 7, t_2 = 31$, RSA-1976
Key size	Root: 2000 bytes Leaves: 520 bytes	Root: 713 KByte Leaves: 17952 bytes	Root: 575 KByte Leaves: 58500 bytes	Root: 327 MByte Leaves: 2.34 MByte
Key-delegation to leaves	0.24 sec	24.1 sec	27 sec	3290 sec
Shared key computation	0.3 sec	24.4 sec	0.1 sec	0.7 sec

Table 3: Timing/storage of hierarchical schemes: the numbers were computed from Tables 1 and 2, assuming 1μs for computing SHA256.

4.5 Simulations

We complement our discussion of implementation issues with a report on a specific simulation scenario presented in Appendix A.

5 Security

In this section we show that when combining any secure linear scheme with the Sakai et al. scheme as above, the result is a secure scheme that is resilient to compromise of arbitrarily many leaf nodes. We start by recalling the security model for a hierarchical KAS.

5.1 Security model for Hierarchical KAS

Setup. The KAS root chooses and publishes a set of public parameters for the scheme. (These may include information about the maximal depth of the hierarchy, number of nodes, security parameters, cryptographic functions, domain of keys, etc.). It also chooses at random the master secret keys and keeps them secret.

Attacker. The attacker is given all public parameters of the system. It may then perform two type of actions:

- *Compromise:* The attacker names a node and obtains all the secret values held by the node.
- *Test query:* The attacker names two leaves and obtains a value z chosen as follows: A bit σ is chosen at random in $\{0, 1\}$. If $\sigma = 1$ then the attacker gets the secret key shared between the two leaves, and if $\sigma = 0$ it gets a key chosen at random from the set of all possible shared keys.

  We refer to the two leaves specified in the Test query as the *target leaves*, and the value returned to the attacker is the *target key.*

The attacker ends its run by outputting a bit σ' (which represents its guess of the bit σ, i.e., whether the seen test key is real or random).

Informally, Definition 2 below states that the attacker is deemed successful if the guess is correct, i.e., $\sigma = \sigma'$, and the scheme is deemed secure if no attacker can succeed with probability much better than 1/2.

In some KAS schemes, including the ones presented here, a hash function is used by the scheme which is modeled as a "random oracle" in the security analysis. In this case, the attacker will issue an additional form of query, namely, a *random-oracle query* on a given value for which it receives the result of applying the random oracle on that value.

Attacker's compliance. A security model for a KAS sets some restrictions on the attacker's queries. For example, how many nodes it can compromise and in what order. Typically, the restrictions will include a bound on the number of compromised nodes in each level. It is also common to restrict the adaptiveness of the queries. This may range from a fully non-adaptive strategy where the attacker makes all its choices at the start of its run, to a fully-adaptive case where each query can be decided by the attacker after seeing the responses to previous queries.

Two restrictions that appear in every model are that (i) only one test query is allowed to the attacker and (ii) neither of the leaves named in the test query or any of their ancestors can be compromised. We will refer to an attacker that follows the model's restrictions as a compliant attacker. When talking about an attack model for a specific KAS model $\mathcal{M}$, we will refer to the attacker as $\mathcal{M}$-compliant.

Definition 2 (KAS-security) *A hierarchical KAS is called secure for model $\mathcal{M}$ if the KAS-advantage of any $\mathcal{M}$-compliant attacker $\mathcal{A}$ is negligible, where KAS-advantage is defined as:*

$$|\ \Pr[\mathcal{A} \textit{ outputs 1} \mid \sigma = 1] - \Pr[\mathcal{A} \textit{ outputs 1} \mid \sigma = 0]\ |$$

where the probability is over the randomness of the scheme as well as the internal randomness of $\mathcal{A}$.

Definition 3 (Ordered attacker) *We say that an attacker against a hierarchical KAS is* ordered *if it uses all the Compromise queries for internal nodes before any leaf Compromise. (Note that this constitutes a limitation on the* adaptiveness *of the attacker.)*

5.2 Security of the Hybrid Scheme

Here we prove that the hybrid scheme $\mathcal{H}'$ is as resilient to internal-node compromise as the original scheme $\mathcal{H}$, and in addition $\mathcal{H}'$ is resilient to compromise of any number of leaf nodes. Note that the attacker model for the hybrid scheme is the same as for any hierarchical KAS except that now we have another level in the hierarchy, and we do not restrict the number of compromised nodes in this level (as long as the attacker does not compromise the test leaves). Below we denote by $\mathcal{M}$ the KAS model for the original scheme $\mathcal{H}$, and by $\mathcal{M}'$ the KAS model for $\mathcal{H}'$.

Theorem 1 *Let G_1, G_2, e be two groups of order q and a bilinear mapping that together satisfy the BDDH assumption; Let $\mathcal{H}$ be a linear hierarchical KAS over $GF(q)$ that is secure for model $\mathcal{M}$; and let the hash function H used in the bilinear scheme be modeled as a random oracle. Then, the resultant hybrid scheme $\mathcal{H}'$ is secure against any $\mathcal{M}'$-compliant and ordered attacker.*

The rest of this section is dedicated to prove the theorem. We show a reduction from the security of our hybrid scheme $\mathcal{H}'$ to the BDDH assumption. Specifically, given any $\mathcal{M}'$-compliant and ordered attacker $\mathcal{A}'$ that breaks the scheme $\mathcal{H}'$ with some advantage, we build an attacker $\mathcal{B}$ that breaks the BDDH assumption with essentially the same advantage. (Hence if the BDDH assumption holds then $\mathcal{A}'$ advantage must be negligible.)

In the description below we refer to $\mathcal{B}$ as "the simulator" (since it will try to simulate for $\mathcal{A}'$ a run of the system). $\mathcal{B}$ is initialized with the BDDH parameters $\langle G_1, G_2, e\rangle$ and the points $(P, P_a = P^a, P_b = P^b, P_c = P^c, g)$ and it needs to decide if $g = e(P,P)^{abc}$ or $g = e(P,P)^r$. The idea of the proof is that $\mathcal{B}$ will embed its BDDH input into the test query issued by $\mathcal{A}'$ such that a successful distinction by $\mathcal{A}'$ between a real or random key in $\mathcal{H}'$ implies an equally successful guess of the real/random instance in the BDDH input.

$\mathcal{B}$ starts by choosing security and public parameters for a hybrid scheme $\mathcal{H}'$ using its input parameters (groups G_1, G_2 and map e) for the scheme parameters. $\mathcal{B}$ also chooses a random set of N master secret keys from Z_q, denoted $k_1, \ldots, k_N$, exactly as the root authority of the scheme $\mathcal{H}'$ would do. From these keys $\mathcal{B}$ can compute each of the secret values v_i held by each node in the $\mathcal{H}'$ hierarchy. $\mathcal{B}$ invokes the $\mathcal{H}'$-attacker $\mathcal{A}'$, providing it with the public parameters of the system $\mathcal{H}'$.

The attacker $\mathcal{A}'$, which is assumed to be $\mathcal{H}'$-compliant and ordered, begins with a sequence of internal nodes Compromise actions. For each such internal-node compromise, $\mathcal{B}$ provides $\mathcal{A}'$ with all the secret values v_i held by that node as they derive from the master secret keys that were chosen by $\mathcal{B}$.

For simplicity of presentation, we assume for now that $\mathcal{A}'$ uses the Test query before it compromises any leaf nodes. (We will remove this assumption later.) When $\mathcal{A}'$ makes a Test query for leaves ℓ_1 and ℓ_2, the simulator $\mathcal{B}$ returns to $\mathcal{A}'$ the value $g \in G_2$ from its BDDH input as the target key.

Random-oracle queries by $\mathcal{A}'$, if they existed in the underlying scheme $\mathcal{H}$, are answered by $\mathcal{A}'$ with random values (in the range of the corresponding hash function). Random-oracle queries that pertain the bilinear scheme at the leaves, namely values of the form $H(ID_\ell)$ where ℓ some leaf node, are answered as follows: If the leaf is the first target leaf ℓ_1, set $H(ID_{\ell_1}) = P_a$ where P_a is one of the BDDH inputs. Similarly, for the second target leaf ℓ_2, $\mathcal{B}$ answer with sets $H(ID_{\ell_2}) = P_b$. For any other leaf ℓ, $\mathcal{B}$ chooses at random $\rho \in Z_q$, answers the query with $H(ID_\ell) = P^\rho$, and stores the pair (ID_ℓ, ρ).

After compromising internal nodes and using the Test query, the attacker $\mathcal{A}'$ may start compromising leaves. As said, the idea is to let $\mathcal{B}$ embed the BDDH input in the response to the test query (i.e., in the key generated by test leaves ℓ_1 and ℓ_2). This however requires that $\mathcal{B}$ orchestrates things to match the value g that it gave to $\mathcal{A}'$ in the Test query (assuming that this value was set as $g = e(P, P)^{abc}$), so a distinction between a real/random key by $\mathcal{A}'$ will translate into a similar distinction for the BDDH input.

If $g = e(P, P)^{abc}$ then the identities of the target leaves are hashed to P^a, P^b, and the target key is $g = e(P^a, P^b)^c$. This means that the key that was supposed to be shared between the parents of the target leaves in the original scheme $\mathcal{H}$ is c (namely, the discrete logarithm of the BDDH input P_c with respect to P). Although $\mathcal{B}$ does not knows c, we will see that the linearity of the scheme allows $\mathcal{B}$ to still generate a consistent view for $\mathcal{A}'$.

Since $\mathcal{H}$ is linear then every value that $\mathcal{B}$ gave to $\mathcal{A}'$ as a result of node compromise can be expressed as a linear combination in the master secret keys with publicly known coefficients. Let $v_1, \ldots, v_m$ be all these values, and let $v_j = \sum_{i=1}^{N} \alpha_{i,j} k_i$ be the linear relation for the value v_j. We denote the set of all these equations by E. Note that E describes the full information that $\mathcal{A}'$ has on the values k_i. From the point of view of $\mathcal{A}'$, any set of values k_i that satisfy E is equally probable as the set of master secret keys. Also note that this system has a solution, for example the one induced by the values of k_i chosen by $\mathcal{B}$.

Similarly, we denote the linear relation for the key that the parents of ℓ_1, ℓ_2 would have shared in $\mathcal{H}$ is denoted $s = \sum_{i=1}^{n} \beta_i k_i$. Note that if the vector $\bar{\beta} = (\beta_1, \ldots, \beta_N)$ is linearly dependent dependent on the matrix of coefficients $[\alpha_{i,j}]$ then $\mathcal{A}'$ could find the shared key between the parents of the target nodes, contradicting the security of the underlying scheme $\mathcal{H}$. Therefore, this "dependence event" happens with only negligible probability. If in the run of $\mathcal{A}'$ by $\mathcal{B}$ the above "dependence event" indeed occurs, then $\mathcal{B}$ aborts this run and outputs a random bit.

Otherwise, we can add the relation for c to the system E thus getting a linear system $\langle v_1, \ldots, v_m, c\rangle = M \times \langle k_1, \ldots, k_N\rangle^T$ (where M consists of the $\alpha_{i,j}$'s and the β_i's), and this linear system has a solution *for any value of* s. In particular, there exist k_i's that solve this system for the v_i's that $\mathcal{B}$ gave to $\mathcal{A}'$ and for $s = c$. If this linear system is still underspecified then $\mathcal{B}$ extends it in a random way to a fully specified system (i.e., completes M into a rank-N matrix $\bar{M}$ and randomly sets the additional v_i's that may be needed). Denote the resulting fully specified linear system by $\bar{E}$. Since $\mathcal{B}$ does not know c it writes it into the system of equations $\bar{E}$ as indeterminate c. Solving this system, $\mathcal{B}$ gets a solution that can be written as $k_i = y_i + z_i c$ for all i, where $\mathcal{B}$ knows the scalars $y_i, z_i \in Z_q$.

When $\mathcal{A}'$ compromises a leaf ℓ, $\mathcal{B}$ responds as follows: We recall that $\mathcal{B}$ stored the pair (ID_ℓ, γ) such that $H(ID_\ell) = P^\gamma$. (This is how $\mathcal{B}$ responded to random-oracle queries on leaf identities). The secret values held by ℓ are of the form $V = H(ID_\ell)^v$ where $v = \sum_{i=1}^N t_i k_i$ for known coefficients t_i. Re-writing the k_i's in terms of their linear combination in c, we get

$$v = \sum_i t_i(y_i + z_i c) = (\sum_i t_i y_i) + c(\sum_i t_i z_i)$$

where $\mathcal{B}$ can compute the scalars $\tau = \sum t_i y_i$ and $\rho = \sum t_i z_i$. Hence $\mathcal{B}$ can return to $\mathcal{A}'$ the group element

$$V \leftarrow P^{\gamma\tau} \cdot (P_c)^{\gamma\rho} = (P^\gamma)^{\tau + c\rho} = H(ID_\ell)^v.$$

The run of $\mathcal{B}$ concludes when $\mathcal{A}'$ outputs its guess bit σ', in which case $\mathcal{B}$ outputs the same bit.

In all, we have that the view of $\mathcal{A}'$ in its run under $\mathcal{B}$ is identical to its view in a real run against the hybrid scheme $\mathcal{H}'$ where root keys are chosen at random. Moreover, the test query is answered by $\mathcal{B}$ with its BDDH input g, so if $g = e(P, P)^{abc}$ in the BDDH instance then

$$g = e(P^a, P^b)^c = e(H(ID_{\ell_1}), H(ID_{\ell_2}))^{\sum_{i=1}^N \alpha_i k_i}$$

which corresponds to the real key, i.e., $\sigma = 1$, in the run of $\mathcal{A}'$ under $\mathcal{B}$. On the other hand, when $g = e(P, P)^r$, for random r, then the response by $\mathcal{B}$ to the test query of $\mathcal{A}'$ is a random element in G_2, which corresponds to $\sigma = 0$. In other words, $\mathcal{B}$'s response is correct whenever $\mathcal{A}'$ is correct which means that the advantage of $\mathcal{B}$ against the BDDH problem is the same as the advantage of $\mathcal{A}'$ against the hybrid scheme, except in the case that $\mathcal{B}$ stopped its run earlier due to a "dependence event" as defined above and which we showed to have a negligible probability to occur. Thus, we have that, up to a negligible difference, the KAS-advantage of $\mathcal{A}'$ against $\mathcal{H}'$ is the same as the BDDH-advantage of $\mathcal{B}$ and hence both must be negligible.

The only thing left now is to remove the assumption we made earlier in the proof that the pair of test leaves is chosen by the attacker at the beginning of its run. Without that assumption we let $\mathcal{B}$ simply guess these two leaves and abort in the case that $\mathcal{A}'$ chooses a different test pair. This results in an additional factor of m^2 in the cost of reduction where m is the number of leaves in the hierarchy. □

Remarks. We comment that since we work in the random-oracle model, we could replace the BDDH assumption with the computational BDH assumption, by replacing the key $e(I, J)^s$ by its hash $H'(e(I, J)^s)$ with H' yet another random oracle.

6 Conclusions

In this paper we have proposed, and analyzed in detail, a hierarchical, non-interactive key agreement protocol which is particularly suitable for use in mobile and tactical networks, with an emphasis on being resilient to compromises of arbitrary numbers of leaf nodes (which are considered the most vulnerable). While the schemes are limited in their efficiency as the thresholds grow, this is not an impediment for networks with the number of nodes and limited hierarchies typically found, for example, in tactical networks. The proposed schemes are intended to minimize the communication complexity both in terms of the number of bits transmitted and the number of protocol runs; the use of identity-based schemes provides an implicit benefit since no directory look-up protocols or related services are required. This benefits both the energy efficiency and also the undetectability (based on RF emissions) of mobile nodes.

Acknowledgment.

Research was sponsored by US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] Rolf Blom, An Optimal Class of Symmetric Key Generation Systems. EUROCRYPT '84, pages 335–338, 1984

[2] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung. Perfectly Secure Key Distribution for Dynamic Conferences. *Information and Computation*, 146(1):1–23, 1998.

[3] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM. J. Computing*, 32(3):586–615, 2003.

[4] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[5] Wenliang Du, Jing Deng, Yunghsiang S. Han, Pramod K. Varshney, Jonathan Katz, and Aram Khalili. A Pairwise Key Pre-Distribution Scheme for Wireless Sensor Networks. ACM Transactions on Information and System Security 8(2): 228-258, 2005.

[6] R. Dupont and A. Enge. Practical Non-Interactive Key Distribution Based on Pairings. http://eprint.iacr.org/2002/136, 2002.

[7] L. Eschenauer and V. D. Gligor A key-management scheme for distributed sensor networks Proceedings of the *9th ACM conference on Computer and communications security*, ACM-CCS'02, pages 41–47. ACM, 2002.

[8] Goichiro Hanaoka, Tsuyoshi Nishioka, Yuliang Zheng, and Hideki Imai. A Hierarchical Non-interactive Key-Sharing Scheme with Low Memory Size and High Resistance against Collusion Attacks. *Comput. J.*, 45(3):293–303, 2002.

[9] Teremy Horwitz and Ben Lynn. Towards Hierarchical Identity-Based Encryption. In *Eurocrypt '02*, pages 466–481, 2002. LNCS No. 2332.

[10] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, ACM-CCS'03. Pages 52–61, 2003, ACM.

[11] B. Matt. Toward Hierarachical Identity-Based Cryptography for Tactical Networks . In *Military Communications Converence, MILCOM 2004*, pages 727–735. IEEE, 2004.

[12] M. Ramkumar, N. Memon, and R. Simha A hierarchical key pre-distribution scheme *Electro/Information Technolgy Conference*, EIT 2005, IEEE, 2005.

[13] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems Based on Pairings. In *Proceedings of SCIS 2000*, 2000.

[14] D. Page, N. P. Smart, and F. Vercauteren, A comparison of MNT curves and supersingular curves. *Appl. Algebra Eng., Commun. Comput.*, vol. 17, no. 5, pp. 379–392, 2006.

[15] S. Balfe, K. D. Boklan, Z. Klagsbrun, and K. G. Paterson, Key Refreshing in Identity-based Cryptography and its Applications in MANETS Milcom 2007.

[16] NS-2: Open Source Network Simulator. http://www.isi.edu/nsnam/ns/

[17] S. Reidt, P. Ebinger, and S. D. Wolthusen, Resource-Constrained Signal Propagation Modeling for Tactical Networks. Manuscript, 2006.

[18] A Compromise-Resilient Scheme for Pairwise Key Establishment in Dynamic Sensor Networks. W. Zhang, M. Tran, S. Zhu, G. Cao, MobiHoc 2007.

A Simulating the Key Distribution

Although the main advantage of a non-interactive key-agreement scheme lies in applications where the distribution of keys to the leaves can be done in an off-line manner, there are still many applications where one needs to refresh the keys "in the field", and in this section we examine the feasibility of our key-distribution scheme in such an environment. Specifically, we consider a small tactical network, where complete key refreshments may be mandatory, e.g., to merge networks with different security parameters.

To get a feel for the time and feasibility for an ad-hoc key refreshment in a tactical network, we set up a simulation scenario illustrating a platoon of 37 nodes in a city area. For this simulations we chose to work with the setting of $t_1 = t_2 = 3$, and using small parameters also for the elliptic curve. The simulation was done using the network simulator NS-2 [16]. Figure 1 shows a snapshots of the scenario, a full video is available from the authors.

In the simulation scenario, the platoon is splitting between buildings in two and temporarily three subgroups which are represented by squads. Two of the squads remain connected nearly all time, while the connection to the third squad is disrupted several times due to buildings and distances between the squads. The platoon is intended to be part of a bigger network, e.g. a network of several platoons. The green node represents the only level 1 node, the black ones the level 2 nodes and the gray ones the leaf nodes in the hierarchy of depth 2. The simulation examines a key distribution process, which is executed in a frequency of 50 seconds. For this purpose the green

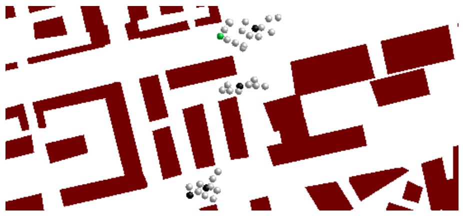
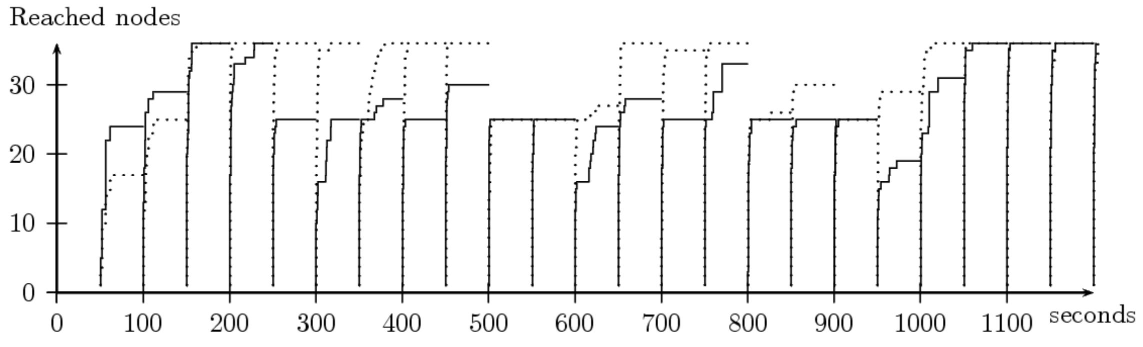


Figure 1: Snapshot of simulation scenario illustrating a platoon of 37 nodes in a city area.



Reached nodes:	29	36	25	25	25	24	25	25	25	19	36	36
Duration:	11.6	29.0	17.5	3.5	1.6	24.0	2.5	4.0	2.0	22.1	3.5	2.9

Data Packets (size including header: 383 to 583 byte)

Sent:	38	79	37	47	26	31	31	48	31	28	48	65
Received:	32	58	31	30	25	26	26	35	27	21	38	55
Dropped:	6	16	6	17	1	4	5	12	4	4	10	8

AODV Packets (average size including header: 87 byte)

Sent:	79	149	69	75	85	51	55	76	50	47	102	97
Received:	32	66	31	31	25	26	26	35	27	22	38	55
Dropped:	33	47	29	38	46	19	24	36	19	18	48	25

Figure 2: Simulation result of key distribution in an interval of 50 seconds. The tables beneath show the results for every second distribution process.

node disseminates the key material to all level 2 and leaf nodes. As part of a bigger network, the green node can either calculate the respective keys on its own or receive it from a key distribution center. The simulation starts at the point where the green node already holds the key material and simulates its distribution at several points in time. The distribution process is performed 24 times, illustrating for different topologies whether the key material can be disseminated successfully or not. The simulation does not incorporate response messages from internal and leaf nodes or retries in case of a transmission failures. Sophisticated communication protocols for key distribution and further security related protocols will be part of future work.

The keys for level-two nodes in this scheme consist of $4^2 = 16$ numbers modulo $q \approx 2^{160}$, and the size of leaf nodes consist of 16 points on the elliptic curve. Hence the size of the key material packets for level-two nodes is $16 \times 160 = 2560$ bit (320 bytes), and the size of the key material packets for leaf nodes is $16 \times 260 = 4160$ bits (520 bytes).[6]

The graph in figure 2 shows the number of nodes that received their new key at the various points in time. New keys are distributed at a frequency of once every 50 seconds (which is not chosen as a realistic period but to provide a larger set of simulation results). The dotted line illustrates

[6]To transmit the key material securely, it can be encrypted by the old private key as proposed by Balfe et al. [15].

the results under the usage of the standard Two-ray-ground propagation model as incorporated in NS-2 [16] which does not consider the buildings. The figure shows that the distance between the nodes in the periods of 150 to 450, 650 to 800 and 1000 to 1200 seconds is small enough to reach all nodes in the network. The solid line illustrates the results under the consideration of buildings as obstacles and reflexions on house walls up to a depth of 2 by a ray-optical propagation model [17]. Most of the time in the period of 250 to 1000 seconds the green node can only reach 25 nodes, which is the size of the upper two squads in the simulation. Despite of several situations of inter-visibility between the squads, it happens rarely that some of the remaining nodes receive their complete key material. In second 350 and respectively 750 for example the squads are connected by different nodes for at least 20 seconds, but the routing protocol seems to react too slowly to take full advantage of these temporarily existing routes. These results show that we cannot expect a successful key distribution in a weekly connected network similar to the illustrated one in second 250 to 1050.

Beneath the graph in figure 2 the first two rows list the maximum number of nodes that could be reached after the respective time. For the purpose of clarity, all values are only listed for points in time corresponding to multiples of 100 seconds, and only for the ”solid line” simulation which incorporates the influences by buildings. The *duration* values show that the routing process keeps sending packets up to a period of 30 seconds. If the links in the network are stable, the distribution process is completed after 1.5 to 5 seconds. This situation occurs in second 150 and 1000 to 1200 when all nodes have a line of sight contact, as well as in seconds 400, 500 to 550 and 800 to 900 where the upper two squads are properly connected, but the third one is disconnected from these two due to buildings or the distance. The values for the data and routing packets give an overview of the amount of data that is sent during a key distribution. The high amount of dropped routing packets highlights the potential for sophisticated communication strategies, which will be investigated in future work. However, distribution times of 1.5 to 5 seconds and the computation period of less than one second show that an ad hoc key refreshment in our hierarchical key generation scheme is feasible for hierarchy of depth 3.