DSKE: Dynamic Set Key Encryption

Galen E. Pickard, Roger I. Khazan, Benjamin W. Fuller, Joseph A. Cooley
MIT Lincoln Laboratory
244 Wood St
Lexington MA, 02451
Email: {gpickard, rkh, bfuller, cooley}@ll.mit.edu

Abstract—**In this paper, we present a novel paradigm for studying the problem of group key distribution, use it to analyze existing key distribution schemes, and then present a novel scheme for group key distribution which we call "Dynamic Set Key Encryption," or DSKE. DSKE meets the demands of a tactical environment while relying only on standard cryptographic primitives. Our "set key" paradigm allows us to focus on the underlying problem of establishing a confidential communication channel shared by a group of users, without concern for related security factors like authenticity and integrity, and without the need to consider any properties of the group beyond a list of its members. This separation of concerns is vital to our development and analysis of DSKE, and can be applied elsewhere to simplify the analyses of other group key distribution schemes.**

I. Introduction

Many modern applications for internet-based group communication rely on a multicast paradigm, in which one user may send data to a large number of other users in a single message. This is often much more efficient than the unicast paradigm, in which a user wishing to communicate with a group sends a distinct point-to-point message to each other group member. In situations where network bandwidth is at a premium, the increased efficiency provided by multicast allows for the use of applications which might otherwise be too costly. However, the multicast paradigm makes security a more challenging problem, especially as groups evolve. We are motivated by a scenario in which warfighters use a text chat application to communicate with a dynamic group over a tactical network. In this scenario, users need to maintain the confidentiality of transmitted data, yet have a very limited communication bandwidth. Existing methods for maintaining dynamic group security are generally either too inefficient to be useful, or suffer from requirements (e.g. availability of a centralized server) which make them untenable solutions in tactical environments. We present a novel solution to this group security problem, which we call Dynamic Set Key Encryption (DSKE), and show that it is a viable method for maintaining confidentiality of dynamic groups over tactical networks.

One end-goal of a group-security protocol is to create a shared private channel among a group of users. We define a *shared private channel* as a method by which any group member can send a message whose contents can be read by any other member, but cannot be read by anyone who is not in the group. The most straightforward method for creating a shared private channel is through distribution of a shared secret symmetric key. A *shared secret symmetric key* is a piece of cryptographic material which can be used to both encrypt and decrypt messages, and is known only to group members. Once a secret symmetric key is shared among all group members, any message encrypted using it can be decrypted by all group members and only by group members; encryption using this key provides a shared private channel for the group. An important component of a group security protocol, then, is a solution to the problem of *group key distribution*, in which a symmetric key becomes shared among the group.

This work was sponsored by the Department of Defense under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

When the membership of a group is static, or when there is a small number of possible variants on group membership, this does not pose a challenging problem. In these nearly-static situations, pre-configured keys can be distributed to the relevant users, and these keys can be used for communication when appropriate. However, this solution does not scale well when the groups are fully dynamic — that is, when users join and leave at will, causing membership to change unpredictably over time. In a situation in which there are n possible users of a system, there are a total of 2^n possible group membership configurations which could occur, so using pre-configured keys is not a scalable solution to the problem of key management for dynamic groups.

While the problem of dynamic group key management is fraught with complexity, we observe that there is a conceptually simple problem at its heart. When the membership of a group changes, a new secret key must be created and distributed to each member of the group. Some policy dictates who is responsible for creating this key (be it a centralized server, a group member, or a coalition of multiple group members) and what channels are available with which to propagate this key to the other members of the group. Once these decisions have been made, the properties of the "group" become unimportant — we are left with a key, a set of members needing to receive this key, and a set of channels which can be used to transmit it. We call this problem *set key distribution*. In this paper we make the notion of set keys concrete and explicit, and use it to build a novel conceptual framework which we use as a context for discussing group key management. We use this framework to analyze existing group key distribution systems, and present and analyze a number of variants on a novel key distribution system, Dynamic Set Key Encryption.

Specifically, this paper makes the following contributions:

- We define the *set key* paradigm for analysis of key distribution schemes.
- We present *Dynamic Set Key Encryption*, a novel key distribution scheme which is practical for dynamic groups using tactical networks while relying on standard cryptographic primitives.
- We analyze the performance of DSKE, and present a method of *randomized provisioning* which can be used to further lower the cost of key distribution.

In Section 2, we make explicit our definitions and assumptions. In Section 3, we discuss existing key distribution schemes and their lack of applicability to tactical networks. In Section 4, we introduce the set key framework, and in sections 5 and 6 we discuss key provisioning. In Section 7, we present some results relating to the performance of DSKE.

II. Definitions and Assumptions

Although we are motivated by practical examples, many details of actual network behavior are difficult to model accurately while also having very little impact on the comparative properties of various key distribution schemes. We have performed tests on actual networks with actual users, including several deployments of our key distribution system at military exercises [1], but our goal in this paper is to present a more abstract analysis. In support of that analysis, we discuss the underlying assumptions we make regarding communication, computation, information storage, and security.

A. Communication

In this paper, we address issues relating to confidentiality. We note that authenticity and integrity are important concerns as well, but do not treat them in depth. We can act as part of a solution for ensuring integrity, as such a solution is easy to build on top of communication with symmetric keys, but these details are beyond the scope of this paper. Hence, throughout our discussion, we assume that each user has the ability to communicate with each other user in an authentic and guaranteed (but not necessarily secure) manner. We will focus on "broadcast" networks, in which a message sent from any single user to any other user is automatically received by all other users. This choice is motivated by our focus on tactical networks, since these networks are often based on a wireless communication substrate that is inherently broadcast-based.

We consider two fundamental problems as solved: establishing a secret key shared between two users who share an authentic, guaranteed, but public channel (e.g. using Diffie-Hellman key agreement [2]); and secure communication between any number of users who share an authentic, guaranteed, but public channel, and also share a secret key (e.g. using any of a number of symmetric key ciphers). Among these two problems, the latter has solutions that scale well as the number of users increase, while the former does not.

We focus on applications in which communication efficiency is paramount, and thus the naive solution is insufficient. We are motivated by military applications involving severely disadvantaged links, and we observe that in these applications, communication bandwidth is a much more severe bottleneck than either computational overhead or storage requirements. The links we aim to use are limited in terms of both bandwidth and latency; not only is the cost per bit high, but it may take a very long time for a message to reach its recipient [3], [4], [5]. As a result of this, we consider schemes which require multiple sequential rounds of communication to be untenable for our use in tactical situations.

B. Computation and Storage

We make the standard cryptographic assumption that any polynomial-time computation is free, while any algorithm requiring super-polynomial time has no functionality in practice. Implicitly, this follows from the assumption that an adversary's computational power can be estimated, and that factors like key size can be chosen in such a manner as to cause an insignificant increase in cost to legitimate users while guaranteeing that an adversary cannot use brute-force methods within a time-frame deemed allowable. Within the polynomial-time domain, we do not seek to optimize for ease of computation, as we believe that even on the most low-capacity mobile computing hardware, communication is a much more significant constraint than computation.

Similarly, we assume that data storage is free, and that data at rest is secure. These assumptions should both be considered as tempered by specific operational concerns, and in cases where either of these assumptions cannot be made, the resulting complications can be mitigated through periodic deletion of cryptographic material. We consider these concerns to be a facet of group management, and thus outside the domain of this paper.

C. Security

We consider three standard security requirements: group forward secrecy, group backward secrecy, and key independence [6]. *Group forward secrecy* requires that a user who leaves a group (or has its access revoked) cannot decrypt future messages sent to the group. *Group backward secrecy* requires that a user who joins a group cannot use its new group key to decrypt messages sent prior to its arrival. *Key independence* requires that the compromise of a single key by an adversary does not allow that adversary access to other keys.

In the scenarios which motivate our research, group forward and backward secrecy are vital, as without them, the sender of a message has no guarantees as to its eventual recipients. Key independence, on the other hand, we consider as a luxury. Most existing group key distribution systems rely on long-term secrets (e.g. PKI) whose compromise would lead to the loss of other keys, but maintain key independence in terms of group keys. We consider these group keys to be of a more limited lifespan, and thus we do not see violating key independence on group keys as a significant increase in practical attack surface. In fact, it is our relaxation of key independence that allows us to provide a significant reduction in network cost. Thus, we consider group forward and backward secrecy as requirements, but we argue that key independence, while desirable, should not be seen as critical in the case of dynamic groups in tactical environments.

III. Existing Solutions

There is a rich body of research regarding group keying, and many keying schemes have been proposed and implemented. Unfortunately, nearly all of this research has focused on wired networks like the internet, and hence most results can not be applied directly to tactical networks; in particular, there are three general classes of solutions, each of which has a requirement which is unrealistic for tactical networks:

The first class are centralized schemes (e.g. GKMP [7], [8], LKH [9], and OFT [10]). These schemes vary in how they distribute cryptographic material, but share the common property that there is a single entity which is solely responsible for the creation and distribution of all keys. These schemes assume that this entity is always available, and will not function in its absence. In tactical situations, we cannot guarantee that an external server will always be available, and further we cannot guarantee that any given group member will always be present, which would allow them to act as the central node. Thus, schemes which rely on centralization are not useful to us.

The second class of group keying schemes are decentralized schemes (e.g. IGKMP [11], Hydra [12], Baal [13], and SAKM [14]) in which one user sends a key to a number of intermediate nodes, which then relay it to the rest of the group. These schemes inherently rely on multiple sequential rounds of communication. When dealing with networks with high latency (e.g. satellite links), this imposes delays which are often too long for practical use.

The third class of schemes are distributed key agreement schemes (e.g. D-LKH, D-OFT [15], and CKA [16]) in which each user contributes to the computation of the group key. In addition to requiring multiple rounds of communication, these schemes require that all users be present and active in order to create the key. In cases where links are intermittent, it is unrealistic to expect all users to be able to contribute in a timely fashion.

Because of these constraints, little of the previous work on group keying is directly applicable to tactical networks. The best known solution which can be used in tactical networks is the naive fully pairwise solution, which will be discussed and analyzed in section 5. We will present an improvement over this naive solution in section 6. In support of this analysis, we proceed by presenting a new framework for discussing the group keying problem.

IV. The Set Key Framework

We define a *set key* as a piece of cryptographic material (a key) which is shared by a specific set of users, and is known to no one else. We will focus on applications using symmetric cryptography, in which the same key is used to both encrypt and decrypt data. A set key can be used as the basis for a shared private channel among the members of its set, since any member of the set can encrypt a message using the set key and then broadcast it publicly, knowing that only other members of the set will be able to decrypt it. We also avoid the notion of an abstract "dynamic group:" conceptually, if a certain user has its access revoked, existing sets are not modified to account for this, but instead new sets which exclude it are created. These new sets should be thought of as completely separate entities, and we thus routinely encounter sets co-existing separately with different but overlapping membership and mathematically unrelated set keys. In this manner, we can study set key distribution without having to worry about tangential issues like access control and authentication. Given a solution to the set key distribution problem, a group key management protocol (which also solves the related problems) can simply execute a set key distribution scheme to the newly defined set which is created whenever group membership changes. In this regard, we maintain an abstraction between set key distribution and the very notion that there exists a changing group.

A. Set Key Distribution

We define specifically the problem of set key distribution as follows. We have a universe of n users $U = \{u_1, \ldots, u_n\}$ which share a reliable, authentic communications substrate. Additionally, a number of shared private channels $K = \{k_1, k_2, k_3, \ldots\}$ have been established among these users. These channels can take the form of previously distributed set keys which have been provisioned for future use, or of pair-wise channels which we assume to exist (or to be creatable at little cost) *a priori*. For some channel k, we define $u(k) \subset U$ to be the users who share k; equivalently, k is the shared private channel for $u(k)$. We define a target set $T \subset U$ of users who will be forming a new shared private channel, and hence need to be recipients of a new set key, and an initiating user $u^* \in T$ who is responsible for the generation and distribution of the key. We consider solutions to the set key distribution problem to take the form of algorithms which take as input u^*, T, and K, and produce a set of actions to be taken by users which result in a new set key being known to all $u \in T$ and to no $u \notin T$. Since we are focusing on applications in which only a single round of communication is allowed, the only user which performs actions is u^*. Since we are focusing on broadcast networks, we consider that u^* performs some series of computations, then produces a single message which it then broadcasts. Thus, we can describe any set key distribution scheme in terms of the single message which u^* broadcasts. In most cases, u^* will begin by producing a new random key which will be used as (or from which will be derived) the new set key, and we will denote such a key as k^*. We largely ignore the details of our cryptography, but we assume we have an encryption function $E_k(s)$ which should be understood as "encrypt object s using channel k."

V. Static Provisioning

To begin, we will consider solutions to the set key distribution problem for which the set of private channels K is static; we also term these "stateless" solutions, as the properties of K do not change between instantiations of the scheme.

A. Fully Pairwise Key Distribution

To illustrate, we present what we consider to be the simplest solution, in which every pair of users share a private channel, and no other set keys have been provisioned. That is, $\forall u_i, u_j \in U, \exists k_{ij} \in K$ such that $u(k_{ij}) = \{u_i, u_j\}$ and $\forall k \in K, |u(k)| = 2$. We note that these channels can be established through a number of means, including Diffie-Hellman, or if the proper infrastructure is in place, a solution based on RSA encryption [17] or Identity-Based Encryption[18].

To distribute a set key, u^* generates a random key k^*. Then, for each other $u_i \in T$, u^* encrypts k^* using k_{i*} (the private channel from K shared by u_i and u^*) to create $E_{k_{i*}}(k^*)$. u^* packs all of these encrypted keys into one message, and sends it to all other users. We assume this message contains some header providing a list of recipients who should be able to decrypt the message, and we treat the size of such a header as negligble. All other members of T will be able to decrypt one of the encapsulations, and anyone not in T will be unable to decrypt any of the encrypted keys. Hence, all and only members of T will know k^*, satisfying our requirements for set key distribution. The cost of this set key distribution depends only on the size of T: Since every member receives a key through pairwise encryption, a total of $|T| - 1$ encryptions of k^* must be broadcast.

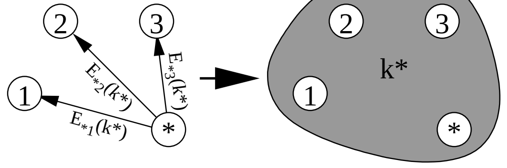

Fig. 1. Distributing k^* requires 3 pairwise encryptions.

B. Further Static Provisioning

We consider pairwise private channels to be either extant or cheap, since regardless of the expense of creating such a channel, it serves as a basis for both pairwise and group security of all kinds, and we thus consider the expense as a necessary start-up cost. However, further wide-scale static provisioning is much more costly. We can, for example, statically provision all channels corresponding to all possible sets of three users. There are $O(u^3)$ such sets, and there is no obvious method for creating channels for them without paying a total cost of $O(u^3)$. If there is no third party who can distribute to each user the $O(u^2)$ keys they need out-of-band, then each user will need to do some amount of work to establish shared private channels, requiring at least $O(u)$ messages at an average size of $O(u^2)$ each. In general, provisioning all channels of size m and below scales at a cost of $O(u^m)$.

If we do have statically provisioned all channels of size m and below, set key distribution can be carried out at a cost of $\left\lceil \frac{|T|}{m-1} \right\rceil$. To accomplish this, u^* generates a random key k^*, then partitions all other members of T into sets of size at most $m - 1$. For each of these subsets of T, K contains a channel shared among only u^* and the other members of the subset, and u^* can encrypt k^* using each of these $\left\lceil \frac{|T|}{m-1} \right\rceil$ channels to achieve set key distribution.

VI. Set Key Provisioning

To improve upon the static provisioning scheme, we observe that we can use existing set keys to aid in the distribution of new set keys. We require, then, that all users maintain a database of all set keys they have created and received, as well as a mapping from these keys to their memberships, and back. We note that we are effectively provisioning set keys for future use, in addition to using them for communication.

When a user u^{*1} wishes to distribute a set key to a group T, it first searches its database for shared private channels corresponding to a set $S \subset T$, and chooses the largest such subset. u^* encrypts k^* using k_S to create $E_{k_S}(k^*)$, and for every other $u_i \in T, \notin S$, uses pairwise channels to encrypt $E_{k_{i*}}(k^*)$. The members of S can decrypt the key encapsulated using k_S, and all others can decrypt using their pairwise channels. Again, all and only members of T will be able to decrypt the

[1]Note that this may be a different u^* than was used previously; in principle, any user can distribute keys, and we always refer to the user initiating a distribution as u^*

new key k^*. We note that it is critical that S is a subset of T, since if there are any members of S that are not also members of T, they will be able to decrypt the message and receive k^*, violating our constraint that only members of S know k^*.

If there exists at least one $k \in K$ such that $u(k) = S \subset T$ has a least 3 members, then the cost of this set key distribution is strictly smaller than the cost of an analogous fully pairwise distribution. A total of $|T| - |S| + 1$ encapsulations are broadcast: $|T| - |S|$ pairwise encapsulations and a single encapsulation to T.

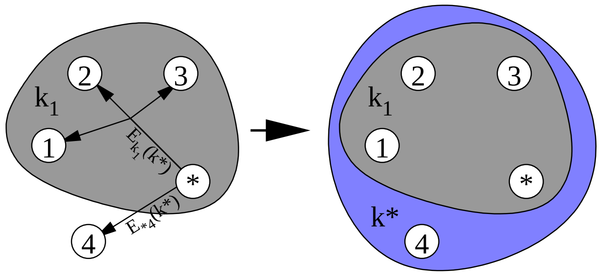

Fig. 2. Using k_1, k^* can be distributed using a total of 2 encryptions.

A. Dynamic Set Key Encryption

Here, we introduce a further improvement on key reuse: using multiple existing shared private channels to form a set cover of the new set. We introduce the following framework, which we call Dynamic Set Key Encryption (DSKE). Consider the universe U of n users, and also the set of all subsets of U which share a private channel $K \subset 2^U$. We note that K necessarily contains all subsets of 2^U of size 2, which correspond to the pairwise channels, and that we do not consider subsets of size 1 or 0, as they are meaningless in the context of the set key distribution problem.

When we consider the problem of a user u^* distributing a set key to some set T, we can easily determine which channels are relevant. u^* can only use channels it shares, and it is not allowed to use any which are shared with users outside T, so we define the set of relevant keys K_R as $\forall k \in K, k \in K_R \Leftrightarrow u^* \in u(k), u(k) \subset T$. In the above examples, for example, this set of relevant keys will contain all pairwise channels between u^* and other members of T, as well as all subsets of T which have been keyed previously, but in general, any set in S meeting these requirements could be relevant if its members share a private channel.

The problem of set key distribution, then, can be described entirely in terms of finding a set cover of T using sets in K_R. Any set cover of T drawn from K_R will correspond to a set of channels which can be used to key T, and any valid keying of T will correspond to some set cover. We can describe the fully pairwise scheme as “create a set cover using only sets of size 2,” and the first improvement as “create a set cover using the largest set in K_R and some number of sets of size 2.” Ideally, one would like to find the minimal set cover, but unfortunately that is an NP-hard problem. In practice, a greedy algorithm can be used to find good set covers. The greedy algorithm begins with an initially empty set of sets X, and adds to it sequentially the set in K_R which has the largest intersection with $T - \cup(X)$. In the worst case, the greedy algorithm performs worse than optimal by a factor of $O(\log |T|)$ [19], but on real-world inputs and randomized inputs with sufficient entropy, the greedy algorithm tends to perform within a sub-logarithmic factor of optimal [20]. We refer to the basic idea of distributing a key using a set cover of existing keys as DSKE, and we will proceed with a discussion of how existing work fits into the DSKE framework, as well as possible improvements beyond basic DSKE.

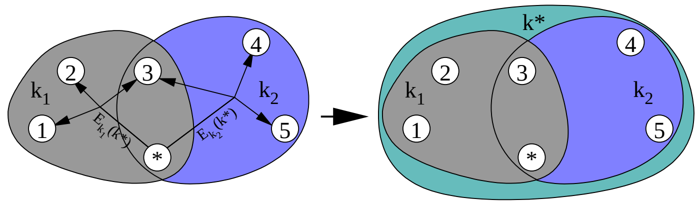

Fig. 3. k_1 and k_2 can be used together to distribute k^*.

B. Dynamic Provisioning

When distributing set keys to a set T, we have thus far only considered various methods of encrypting the key which will be used by T itself. In principle, there is no reason u^* could not create and distribute secondary keys, which could be used in current as well as future key distributions. For example, we could use a scheme such as the following. Consider a scenario in which $T = \{u^*, u_1, u_2, u_3\}$, and K_R contains only sets of size 2. u^* can create an “intermediate” key k_i, then encrypt k_i using k_{1*} and k_{2*}. If u^* then encrypts k^* using both k_i and k_{3*} and sends all four of $E_{k_{1*}}(k_i), E_{k_{2*}}(k_i), E_{k_i}(k^*)$, and $E_{k_{3*}}(k^*)$, then all three recipients will be able to decrypt k^*. This scheme involves sending 4 encapsulations instead of the 3 that would have been required by a fully pairwise scheme, but in return, an intermediate key is created that can then be used for future key distribution.

This scheme generalizes easily to a case where $T = \{u^*, u_1, u_2, u_3, \ldots, u_t\}$, and u^* sends out a “chain” of intermediate keys $k_{i_2}, k_{i_3}, \ldots, k_{i_{t-1}}$ in which k_{i_2} is encrypted in the manner of k_i above, and k_{i_n} for $n > 2$ are each encrypted using $k_{i_{n-1}}$ and also k_{n*}. Finally, k^* is encrypted using k_{t-1} and k_{t*}, and all members of T are able to decrypt k^*. This generalized scheme creates a chain of $t-2$ intermediate keys such that a user u_n receives all intermediate keys $k_{i_2}, \ldots, k_{i_n}$ in addition to k^*. Distributing these $t-2$ extra keys costs one encapsulation per key, increasing the total cost from $|T| - 1$ to $2|T| - 3$ when compared to fully pairwise distribution. In cases where the membership of subsequently keyed sets is strongly correlated, this is perhaps a worthwhile tradeoff. We refer to the creation of intermediate keys as *dynamic provisioning*, and we will evaluate both how existing schemes relate to the provisioning framework and present and evaluate novel provisioning schemes.

C. Logical Key Hierarchy as Dynamic Provisioning

To demonstrate the power of this framework, we show how to rephrase and analyze the performance of the Logical Key Hierarchy (LKH) Protocol as a specific instance of a dynamic provisioning scheme. Throughout this analysis, we will continue to assume prior establishment of pairwise channels. LKH is a centralized system, though, so in this case we have a single user which is always responsible for distributing set keys; no keys which do not contain this single u^* need ever be considered.

In LKH, the n members of a group are arranged in some order, and a balanced binary tree of height $\log_2(n)$ is constructed on that ordering. Each node in this tree corresponds to a key that is known only to those users which are children of that node, with the root node of the tree corresponding to the set key. If we have (in order) users $u_1, u_2, \ldots, u_n$, and we assume for simplicity that $n = 2^k$ for some integer k, then set keys for $\{u^*, u_1, u_2\}, \{u^*, u_3, u_4\}, \ldots, \{u^*, u_{n-1}, u_n\}$ correspond to the first layer of internal nodes $\{u^*, u_1, u_2, u_3, u_4\}, \ldots, \{u^*, u_{n-3}, u_{n_2}, u_{n-1}, u_n\}$ correspond to the second layer, and so on.

When group membership changes, the new key can be distributed efficiently because of this tree-structure of provisioned keys. In order to maintain the structure, though, keys other than the overall set key must be distributed. In particular, when membership changes, either a user at a particular leaf left the group, or a user joined and was added to the structure at a leaf. In either case, there are $\log_2(n)$ internal nodes which are an ancestor of the affected leaf. Starting at the leaf and working towards the root, the root user generates a new key for each of these ancestors and encrypts it to the relevant users. In any case, we have provisioned keys such that two existing keys serve to cover the recipient set for each new key to be distributed.

In this manner, we can see that LKH is simply a special case of Dynamic Provisioning. It is a very efficient scheme, requiring only $O(\log(n))$ encryptions to distribute a key to a new set that deviates by the presence or absence of a single user from the "current state," but, unfortunately, LKH relies on a single user to provide these keys. In reliable networks where the presence of a centralized server is a reasonable requirement, LKH is a fine solution. Distributed variants on LKH exist, but are generally contributory keying schemes, and require more than one round of communication.

D. *Randomized Provisioning*

We now present a simple method of dynamic provisioning that improves the long-term performance of DSKE in return for a small startup cost, which we call *DSKE with Randomized Provisioning* (RP-DSKE). RP-DSKE follows similarly to the scheme described in the above introduction to dynamic provisioning, except that only one intermediate key is produced and distributed. Specifically, u^* chooses some random subset $R \subset T$ with $u^* \in R$. If there exists a channel k in K such that $R \subset u(k) \subset T$, u^* does not provision and proceeds normally. If no such channel exists, u^* performs the usual DSKE set-covering using members of K, calling the set of channels used $D = \{K_{D_1}, K_{D_2}, \ldots\}$. We note here that an implementation of DSKE would now broadcast $E_{K_{D_1}}(k^*).E_{K_{D_2}}(k^*)\ldots$. Instead, we find some proper subset $D' \subset D$ with the property that $R \subset \cup u(D')$. If no such subset exists, u^* proceeds with normal DSKE. If it finds such a subset, though, it computes a random $k^{*\prime}$ which it will distribute to $\cup u(D')$. We note that we have dynamically provisioned a new random subset of T at a cost of one encryption: u^* encrypts $k^{*\prime}$ using all $k \in D'$, encrypts k^* using all $k \in D - D'$, and also with $k^{*\prime}$. Because it cost so little to produce, the randomly provisioned key needn't be used often or with high probability in order to cause a net reduction in bandwidth usage. The actual performance of this algorithm depends highly on the specifics of the target sets (similarly, LKH as stated is worse than the fully pairwise solution when subsequent target sets differ by a large number of users).

VII. Results

Here, we present results demonstrating the performance of DSKE. We note that schemes in which previous set keys are used to distribute future set keys have an inherent statefulness that creates a situation in which the performance of a scheme depends on which sets are keyed, and in what order. To account for this, we explicitly consider the group dynamics which underlie the formation and change of group membership. We analyze the performance of DSKE on a corpus of dynamic group membership data collected from the internet, and also on two types of simulated group membership patterns. As a baseline for comparison, we use the fully pairwise distribution scheme; we present the performance of our novel schemes in terms of the ratio of the cost of our schemes to the cost of fully pairwise distribution.

A. *DSKE Results and Real-World Data*

Figure 4 shows DSKE operating on real-world data. We have collected logs of public internet chatroom use, and extracted room membership information [2]. These chatrooms are unencrypted, but we simulate the cost of adding security by calculating which sets of users were actually formed and in what order, and then applying the DSKE algorithm to that group dynamic. Specifically, we look at the sets of users to which messages are sent, and simulate distributing keys to each of these sets. We do not assume that a new key need be distributed every time a user joins or leaves the chatroom, and we do not consider the group membership to change until a message is sent. Hence, each group membership change corresponds to a set key distribution, and each change could potentially represent a drastic modification of group membership. Figure 4 shows the average cost of distributing keys over 2000 randomly-sampled 500-membership-change long intervals drawn from our data. Four different chatrooms are shown, with average populations varying between 5.1 and 13.1 concurrent users and average total number of users varying between 13.1 and 33.6 users. Additionally, the total performance over all four rooms (which is effectively the average of the rooms weighted based on relative population) is shown. All results shown are cumulative, so we see, for example, that the total cost of the first 500 key distributions is only 30% of the fully pairwise cost when using DSKE.

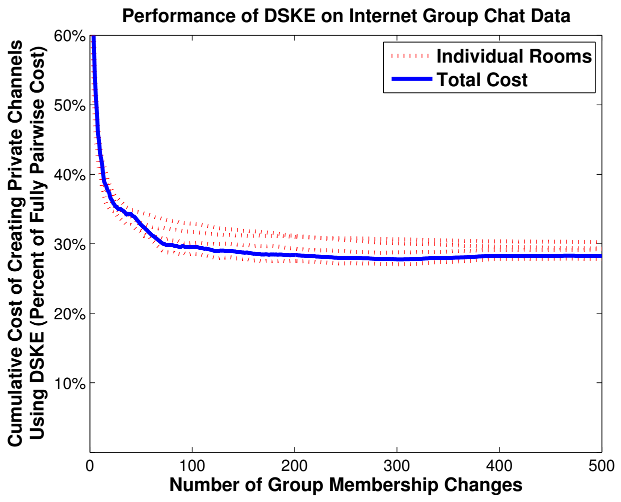

Fig. 4. Performance of DSKE on Real-World Data

These results show that, on average, DSKE performs approximately 70% better than fully pairwise key distribution (that is, the cost of DSKE is 30% of the cost of fully pairwise). The four rooms are of differing sizes and show a range of group dynamics characteristics, but the performance improvement garnered by DSKE is consistent across rooms. Over the first 500 key distributions, the four rooms show total costs of 29.3%, 27.8%, 29.2%, and 30.3%, yielding a mean cost of 29.1% with a standard deviation of 1.0%. Additionally, the results show that these improvements in performance were seen very rapidly: the cost is reduced below 50% within the first 5-10 key distributions and a steady-state around 30% was reached within the first 100, after which no further improvement is seen.

B. *RP-DSKE Results and Group Dynamics Simulations*

We use simulated data to investigate the specific properties of group dynamics that impact the performance of DSKE, and also to reduce the noise inherent in real-world data, allowing us to better compare DSKE and RP-DSKE. We consider two different types of group dynamics: in the first, a set of users U is defined, and each target set is an independently chosen random subset of U. In the second, a set of users U and an

[2]These data will be made available for public use at our website, and will be presented in more depth in a forthcoming publication.

initial $T_0 \subset U$ are chosen, and each subsequent T_i differs from T_{i-1} by the addition or deletion of only a single member. When considering the usual framework of group key distribution, these two models represent the two extremes of volatility. In the sequential set model, only one membership change occurs between each key distribution, while the independent model simulates situations where enough change happens between key distributions that the sets are functionally independent.

We make two high-level observations: first, we expect the performance of DSKE to improve over time, since a larger number of provisioned sets should result in a smaller number of encryptions; second, we should expect DSKE to perform better in the sequential set model, in which subsequent sets are more closely correlated. In this model, provisioned sets are very likely to be used immediately upon the next set key distribution, while that does not hold true in the independent set model.

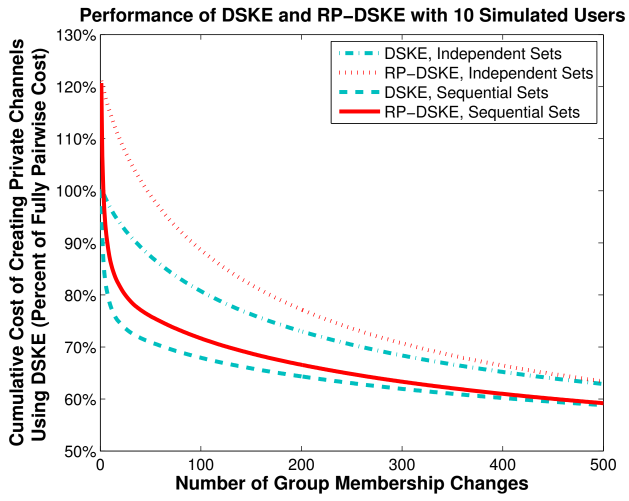

Fig. 5. Performance of DSKE with 10 users

Figure 5 shows the performance of DSKE with and without Randomized Provisioning under the two set models in a universe of 10 users. In the sequential case, DSKE shows a significant performance improvement (30% reduction in cost) over fully pairwise distribution within the first 100 key distributions. Randomized provisioning actually hurts our performance until approximately 500 set key distributions have occurred, but provides modest gains after that point. As expected, performance is worse under the independent model, but we do see significant reductions in cost over the course of the first 500 set key distributions, although they take much longer to appear than in the sequential model.

Figure 6 shows the same results for a universe of 40 users. Using the sequential model, the initial gain from DSKE occurs with approximately the same speed as with 10 users, reaching a cost of approximately 55% within the first 300 key distributions. However, the benefit of DSKE flattens out significantly, showing almost no relative improvement between the 400th and 4000th distributions. With randomized provisioning, however, performance continues to improve, showing a further 5% improvement over DSKE within the first 4000 key distributions. Under the independent set model, DSKE fares poorly with 40 users, showing performance gains of less than 1% when compared with fully pairwise. With the addition of randomized provisioning, however, performance gains are seen from around the 500th key distribution, and the improvement exceeds 10% by the 4000th.

Our results serve to illustrate the ability of DSKE to facilitate key distribution among dynamic groups in tactical environments. Without relying on centralized servers, multiple rounds of communication, or

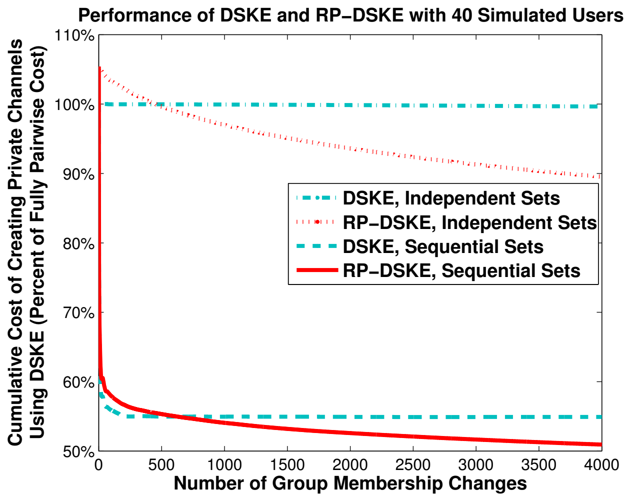

Fig. 6. Performance of DSKE with 40 users

non-standard cryptographic primitives, DSKE yields improvements up 55% compared to fully pairwise key distribution on simulated data, and up to 70% on data drawn from real-world use of a chat application. This represents a significant savings in tactical situations where communication efficiency is a serious concern, and could allow for the practical use of group keying in situations where it might not otherwise be possible.

C. *Future Work*

In none of our simulated group dynamic models do we see the 70% reduction we found when using our real-world data. We hypothesize that this has to do with specific inter-member correlations between users. We observe that many real-world groups exhibit a "core and fringe" pattern in which there is a small "core" of users who are present with high probability, and a larger "fringe" of users who are present much more rarely. This type of dynamic seems as though it should be beneficial to DSKE, as keys shared among this "core" are likely to be re-used many times. Specific analysis of the properties of group dynamics and their effects on key distribution is ongoing, and will be presented in a future paper. It is also worth noting that the point at which RP-DSKE broke even with DSKE was around 500 key distributions in all four of the conditions presented here. Given the ways in which other properties of the cost of DSKE depend on the universe size and user model, the fact that this break-even point remains relative constant merits further investigation.

VIII. Conclusion

In this paper, we have presented the framework of "Set Key Distribution," simplifying analysis of the group key distribution problem. Using this framework, we have presented and analyzed a novel method for distributing set keys, which we term Dynamic Set Key Encryption. DSKE is fully decentralized and uses only one round of communication, making it appropriate for use over disadvantaged links and in situations where communication is unreliable and expensive. Other existing schemes for key distribution rely either on a central server or multiple rounds of communication, and hence DSKE represents an advancement in the state of the art of key distribution in constrained environments.

References

[1] R. Khazan, J. Cooley, G. Pickard, and B. Fuller, "GROK Secure multi-user Chat at Red Flag 2007-03," in *Military Communications Conference, 2008. MILCOM 2008. IEEE*, Nov. 2008, pp. 1–7.

[2] W. Diffie and M. E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644–654, 1976.

[3] Inmarsat. [Online]. Available: http://www.inmarsat.com

[4] Milstar. [Online]. Available: http://www.lockheedmartin.com/products/Milstar/index.html

[5] Iridium. [Online]. Available: http://www.iridium.com/about/about.php

[6] Y. Challal and H. Seba, "Group key management protocols: A novel taxonomy," *International Journal of Information Technology*, vol. 2, no. 2, pp. 105–118, july 2005.

[7] H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) Specification," RFC 2093 (Experimental), Internet Engineering Task Force, Jul. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2093.txt

[8] ——, "Group Key Management Protocol (GKMP) Architecture," RFC 2094 (Experimental), Internet Engineering Task Force, Jul. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2094.txt

[9] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," in *IEEE/ACM Transactions on Networking*, 1998, pp. 68–79.

[10] D. A. McGrew and A. T. Sherman, "Key establishment in large dynamic groups using one-way function trees," 1998.

[11] B. DeCleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhang, "Secure group communications for wireless networks," vol. 1, 2001, pp. 113–117 vol.1.

[12] S. Rafaeli and D. Hutchison, "Hydra: a decentralised group key management," 2002, pp. 62–67.

[13] A. Schaff, "Dynamic group communication security," in *ISCC '01: Proceedings of the Sixth IEEE Symposium on Computers and Communications*. Washington, DC, USA: IEEE Computer Society, 2001, p. 49.

[14] Y. Challal, H. Bettahar, and A. Bouabdallah, "Sakm: a scalable and adaptive key management approach for multicast communications," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 55–70, 2004.

[15] O. Rodeh, K. Birman, and D. Dolev, "Optimized group rekey for group communication systems," Tech. Rep. TR99-1764, 1999. [Online]. Available: citeseer.ist.psu.edu/rodeh00optimized.html

[16] C. Boyd, "On key agreement and conference key agreement," in *Information Security and Privacy: Australasian Conference, LNCS(1270):294302*. Springer-Verlag, 1997, pp. 294–302.

[17] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[18] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing." Springer-Verlag, 2001, pp. 213–229.

[19] C. Lund and M. Yannakakis, "On the hardness of approximating minimization problems," *J. ACM*, vol. 41, no. 5, pp. 960–981, 1994.

[20] O. A. Telelis and V. Zissimopoulos, "Absolute o(logm) error in approximating random set covering: an average case analysis," *Information Processing Letters*, vol. 94, no. 4, pp. 171 – 177, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/B6V0F-4FPJB8Y-1/2/bd27bec0555f5e8167e6884c18eb2afa