UNIVERSITY OF CALIFORNIA
RIVERSIDE

# Secure On-line Key Generation for MANETs by Fragment Assembly

A Project report submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Computer Science

by

John P. Jones

March 2005

The Project Report of John P. Jones is approved:

---

---
Committee Chairperson

University of California, Riverside

# Secure On-line Key Generation for MANETs by Fragment Assembly

John P. Jones
Computer Science & Engineering Department
Marlan & Rosemary Bourns College of Engineering
University of California, Riverside
Riverside, California 92521–0144
Email: jjones@cs.ucr.edu

**Abstract**

Traditional key-management requires either the use of public key cryptography or that an on-line trusted third party arbitrate the selection and distribution of symmetric keys to communicating nodes. Recently much work has been published on random key predistribution methods, in which keys are predistributed between a limited set of node pairs and are used to bootstrap other required node pairs. We propose an alternate scheme, in which key fragments rather than entire cryptographic keys are predistributed to nodes. This modification leads to a different security analysis model and yields substantially better security guarantees than basic random key predistribution for a fixed amount of per node storage.

## I. Introduction

We propose a key-management scheme that allows nodes to communicate securely without requiring access to an on-line trusted key-management system or the use of public key cryptography. Such a scheme is suitable for deployment in wireless ad-hoc networks, which exhibit low reliability and are composed of resource constrained devices. We focus on defending against adversaries capable of capturing a bounded number of network nodes and using the information present in these nodes to subvert system operations. Resource requirements increase with the number of colluding adversaries.

A relatively small symmetric key (on the order of $128$ bits) is sufficient to secure a communications channel with a high level of security. It is, however, impractical in a large network for each pair of nodes to share this amount of information. We seek to maximize security given a constraint on the amount of storage available at a node for the storage of such cryptographic information.

A trusted key-management service could be used to distribute pairwise keys, however, this service must be constantly available throughout the lifetime of the network. State of the art solutions use public key cryptography and signed key certificates to provide high resilience against colluding adversaries but require a significant amount of storage at each node. Unfortunately, assymetric cryptographic systems are very expensive, so that, despite continued improvements in computational power, low- and medium-end devices will continue to experience difficulty implementing assymetric cryptosystems secure against high-end devices for the foreseable future.

In [10] the authors present a randomized key pre-distribution method for Distributed Sensor Networks (DSNs) that enables nodes to be pre-configured with a set of pre-chosen cryptographic keys. This scheme was designed to protect only against adversaries without knowledge of secret information at other nodes. Our scheme is related, but allows nodes to communicate securely in the presence of colluding adversaries.

Our scheme functions by a dealer choosing a master secret, and distributing to nodes in the network a small fraction of this master secret. Any pair of initialized nodes can then construct a symmetric shared key from the intersection of their fractions of the master secret. We determine rules for choosing the size of the master secret and the size of the fragments distributed to nodes to maximize the security achieved against an adversary with a given number of colluding partners.

| | |
|---|---|
| $N$ | The set of nodes in the network. |
| $S$ | The amount of storage available at each node |
| $n_i$,$n_j$ | A pair of nodes in $N$ |
| $a_i$ | The well known identifier of node $n_i$ |
| $\mathcal{D}$ | The dealer |
| $\mathcal{A}$ | The adversary |
| $t$ | The number of nodes the adversary can compromise |
| $\Psi$,$\psi$ | The set of symmetric keys and its size |
| $k$ | The bit-length of a symmetric key |
| $\kappa_{ij}$ | The symmetric key shared between $n_i$ and $n_j$ |
| $\Sigma$,$\sigma$ | The set of key fragments and its size |
| $m$ | The bit-length of a key fragment |
| $\mathcal{K}$ | The master sequence of key fragments |
| $K$ | The number of fragments in $\mathcal{K}$ |
| $q$ | The number of fragments given to each node |
| $r$ | The ratio $q/K$ |
| $\mathcal{T}_i$ | The template for node $n_i$ |
| $\mathcal{K}_i$ | The projection held by node $n_i$ |
| $\mathcal{K}_\mathcal{A}$ | The projection held by the adversary |
| $D$ | Dealer's chosen system instance identifier |
| $salt_i$ | The public salt associated with $n_i$ |
| $\mathrm{MGF}$ | The mask generation function used by nodes |
| $\mathrm{Adv}[\mathcal{A}]$ | $\mathcal{A}$'s advantage over a $0$-adversary |

Fig. 1. Notations used throughout this paper.

While the scheme presented in [10] is subject to attacks from an active adversary (via a man-in-the-middle attack) our scheme uses a mechanism for binding a node's key to its name in an authenticated manner that is not feasible in assymetric cryptosystems. Our scheme achieves results similar to that of Identity Based Encryption, but with a traditional symmetric cipher. This idea has been subsequently suggested in [9].

Figure 1 gives a summary of the notations used in this paper. The remainder of this paper is structured as follows. In section II we give a brief overview of related work in the area of key distribution with an emphasis on the applicability of these methods to networks with limited computational resources such as sensor networks. In section III we provide our system model and describe the assumptions and constraints under which our scheme was constructed. In section IV we describe our approach. In section V we show how the parameters of our approach can be chosen to provide maximum security. In section VI we present computational and analytical results of the security of our approach under various operating conditions. In section VII we present a few practical considerations for the deployment of our approach. Finally, in section VIII we present our conclusions and suggest some ideas on future work.

## II. Related Work

Previous approaches to key-management include the following. The use of a single shared key among all members of a group. Not resilient in the face of malicious members. The exhaustive pre-distribution of shared keys to all pairs of nodes. The use of a trusted on-line key management service (KMS) to construct and distribute keys to pairs of nodes. This requires that every node share a symmetric key with

the KMS and that the KMS is constantly available [20]. Diffie-Hellman key exchange, in which nodes construct a shared key without divulging the secret by blinding secrets using modular exponentiation. This approach is insecure against an active adversary launching a man-in-the-middle attack [7]. Public Key Infrastructures (PKI), in which a Certificate Authority (CA) creates signed key certificates, which bind assymetric key pairs to nodes [11]. Diffie-Hellman and PKIs both require the use of expensive assymetric cryptography operations. We feel that none of these approaches are adequate for use in mobile ad-hoc networks consisting of a large number of very low-range devices communicating in a dynamic, unreliable environment.

In [10] the authors propose a key pre-distribution protocol, in which a large pool of symmetric keys is chosen, and each of these keys is given a key identifier. Prior to deployment, sensors are loaded with subsets of the key pool. After deployment, neighboring sensors exchange lists of the key identifiers corresponding to the keys in their key pool, and agree on using one of the keys that they share. We note the following deficiencies of this scheme.

- Although multiple keys may be shared by a pair of nodes only one of them is used.
- The system does not provide security against colluding adversaries.
- Sensors agree on each other's key pools without any authentication mechanism, thus a man-in-the-middle attack is possible.
- An intrusion detection scheme is suggested to identify compromised nodes. No details are given on how a network of low-power sensors can successfully detect intrusions.
- The keys distributed to compromised nodes must be revoked by an on-line central authority that digitally signs a revocation list that must be verified by the sensor nodes.
- Pairs of sensors that do not directly share a key rely on their neighboring nodes to securely transmit data between them.

Our scheme corrects each of these deficiencies.

Since the publication of [10] several papers have suggested modifications that improve the security obtained from the scheme. In [4] the authors suggest several modifications to improve upon the performance of [10] including hashing multiple keys shared by a pair of nodes and using neighbors to make introductions between pairs of unsecured nodes. In CCS 2003 two papers [9], [12] introduced similar schemes to establish pairwise keys using predistributed information. These schemes make use of linear algebra techniques to allow any two nodes to compute a shared key as long as the adversary does not learn a threshold number of nodes' shares. [13], [8] both utilize partial knowledge of sensor deployment to help the dealer distribute keys biased such that nearby nodes are more likely than random nodes to share keys. In [5] the authors introduce a new scheme in which any pair of nodes may be introduced by intermediary nodes that shares keys with both nodes. This scheme effectively distributes the functionality of a key management service. In this scheme an adversary's strength grows slowly since each compromised node only compromises $N$ pairs of nodes, regardless of adversary strength. Unforutnately the adversary can choose their target in order to disrupt communications between a pair of nodes. The necesity to utilize an intermediary node in order to establish a key between a pair of nodes results in increased communications overhead that can be particularly damaging since this intermediary node may be far away and potentially unavailable. While the work of [5] shows particular promise we present an alternate advancement of the work in [10] in which we explore sharing key fragments rather than entire keys and develop methods for selecting the dealer's distribution parameters and to approximate the security obtained in this scheme.

Our scheme uses concepts of probabilistic set intersection similar to those of probabilistic masking quorum systems [14] and a projection mechanism similar to the use of random projections in the discovery of patterns in biological sequences [3].

## III. System Model

We consider a set of nodes $N$ in a dynamic network topology such that any pair of nodes may need to communicate securely. We assume that each node $n_i \in N$ can be identified by a unique identifier $a_i \in \{0,1\}^*$.

Nodes communicate securely using a symmetric cipher that has an associated key space, $\Psi$, of cardinality $\psi = |\Psi|$. We assume that $\psi$ is a power of 2 and that elements of $\Psi$ are encoded by $k = \lg(\psi)$-bit strings. We require the cipher used to be a probabilistic encryption scheme, since non-probabilistic encryption schemes allow the adversary to detect repeated messages. We have constructed our scheme with the Advanced Encryption Standard (AES) [16], [6] (used with an appropriate padding scheme) in mind, in which $k = 128$. We do not use a public key cryptosystem to ensure secure communications. While public key cryptography provides scalable key management, secure against powerful adversaries, it is often too expensive for nodes with limited resources.

We also assume the availability of a secure hash function such as SHA or MD5 [15], [18]. We will use this hash function for simple hashes, secure pseudo-random number generation, and to construct a mask generation function. A mask generation function hashes an input string of arbitrary length onto a pseudo-random bit string of a specified length. We suggest using the mask generation function MGF1 given in [19], [1], which can be constructed from a secure hash function.

Each node has a limited amount of storage at its disposal. We suppose that the maximum amount of storage available at a node for the storage of cryptographic information is $S$-bits. Our scheme makes use of an off-line trusted dealer, which we will denote $\mathcal{D}$. This dealer authenticates nodes upon their entry to the network and initializes them so that they can communicate securely with each other.

### A. The Adversary

We focus on adversaries who seek to eavesdrop on, or interject communications messages into, the network. These adversaries may capture nodes in $N$ and collect the cryptographic information distributed to them by the dealer. We define a $t$-adversary as an adversary who has access to the information distributed to at most $t$ nodes. Our scheme is parameterized by $t$. We will use $\mathcal{A}$ to denote a $t$-adversary.

We define the *advantage* of a $t$-adversary, $\mathrm{Adv}[\mathcal{A}]$, as the increase in the probability of its determining the key shared between any pair of non-compromised nodes, over that of a 0-adversary. We assume that the symmetric cipher used is perfectly secure, meaning that the best strategy for a 0-adversary is to exhaustively search the key space $\Psi$ to break the cipher. Our goal is to minimize the advantage of a $t$-adversary.

## IV. Our Approach

In our scheme a trusted dealer $\mathcal{D}$ first chooses a secret master sequence $\mathcal{K}$ of $K$ $m$-bit fragments. $\mathcal{D}$ securely initializes each node with a sequence of $q$ fragments. The fragment sequence distributed to a node is determined by projecting the secret master sequence $\mathcal{K}$ onto a template specific to that node. The template $\mathcal{T}_i$ specifies which $q$ of the $K$ fragments in $\mathcal{K}$ are included in node $n_i$'s projection $\mathcal{K}_i$. To communicate securely, a pair of nodes will independently use their key fragment sequences to construct a shared key. The system parameters $K$, $q$ and $m$ are selected so that it is very unlikely that a $t$-adversary will hold enough of these key fragments to succeed in constructing their shared key.

Our scheme is computationally efficient since only symmetric encryption operations, secure hash operations and bit masking must be performed to initiate secure communications between two nodes. The binding between a node's name and the sequence of key fragments it is dealt prevents the adversary from launching a successful man-in-the-middle attack without requiring expensive authentication protocols.

Figure 2 depicts an example of the master fragment sequence $\mathcal{K}$ chosen by the dealer, two projections $\mathcal{K}_i$ and $\mathcal{K}_j$ of this sequence $\mathcal{K}$ distributed to nodes $n_i$ and $n_j$ respectively, and the corresponding templates $\mathcal{T}_i$ and $\mathcal{T}_j$. The key fragments that $n_i$ and $n_j$ hold in common form a shared secret, used to construct a symmetric encryption key. Also shown is the union of the key fragments held by $n_i$ and $n_j$, an adversary that has captured these two nodes discovers all of these key fragments.

| $\mathcal{D}$ | $\mathcal{K}$ | thisisthedealerschosenkey |
|---|---|---|
| $n_i$ | $\mathcal{T}_i$ | 01001110010001101000010001 |
| | $\mathcal{K}_i$ | h ist d er c n y |
| $n_j$ | $\mathcal{T}_j$ | 01000110101000100011010010 |
| | $\mathcal{K}_j$ | h st e e r os n e |
| Shared | $\mathcal{T}_{i\wedge j}$ | 01000110000000100000001000 |
| Fragments | $\mathcal{K}_{i\wedge j}$ | h st r n |
| Adversary's | $\mathcal{T}_{i\vee j}$ | 01001110111001101011010011 |
| Knowledge | $\mathcal{K}_{i\vee j}$ | h ist ede er c os n ey |

Fig. 2. An example with the dealer distributing shares of his chosen secret to a pair of nodes and how they can use those secrets, for good and for ill.

### A. Key Fragments

We will denote the set of key fragments as $\Sigma$ and the cardinality of this set as $\sigma = |\Sigma|$. We assume that $\sigma$ is a power of 2, and that key fragments are encoded by binary strings of length $m = \lg(\sigma)$. Sets that do not meet this requirement do not yield efficient binary representations. If each node has $S$ bits of storage, at most $q = \lfloor S/m \rfloor$ $m$-bit key fragments can be stored at a node.

Although many choices are possible for $\Sigma$, we will focus our analysis on the two extremal cases.

1) $\Sigma = \Psi$: Here each key fragment is an actual key of the symmetric cipher. One fragment provides the full security of the cipher. This is the approach used in all previous key predistribution schemes [10], [4], [9], [12], [13], [8], [5].
2) $\Sigma = \{0, 1\}$: Here each key fragment is one bit long, and $k = \lg(|\Psi|)$ fragments are required to match the cipher's security.

In the first case, two nodes $n_i$ and $n_j$ can communicate securely if they share a single key fragment not known to the adversary. If they share several key fragments they XOR the shared fragments to generate a shared key $\kappa_{ij} \in \Psi$. The communications between $n_i$ and $n_j$ will be secure if there is a single key fragment shared by these nodes unknown to the adversary. This approach is broadly similar to the scheme proposed in [10] although their nodes $n_i$ and $n_j$ randomly select one of the shared keys rather than using them all to compose a more secure key. The small change we propose completely changes the degree of security obtained from the system and the selection of optimal parameters.

In the second case, two nodes must share $k$ 1-bit key fragments that are not known to the adversary to communicate securely. In this case, key fragments must be combined into a secure key using a more complex mask generation function. If, for example, nodes $n_i$ and $n_j$ share $100$ key fragments but $28$ of these are known to the adversary, the adversary can reduce the search space from $100$ to $72$ bits. If the adversary breaks the key, he will also learn $72$ new key fragments to use in future attacks.

### B. Sequences, Templates & Projections

The master secret $\mathcal{K}$ chosen by the dealer is a key fragment sequence; the dealer uses $\mathcal{K}$ to initialize nodes.

*Definition 1 (Key Fragment Sequence):* A key fragment sequence of length $K$ is a string over $\Sigma$, the set of key fragments. Positions in this string are indexed by elements of $\mathbb{Z}_K$.

A template, represents a set membership function that identifies a subset of the fragments in a sequence.

*Definition 2 (Template):* A template is a binary string of length $K$.

The dealer associates every node $n_i$ with a corresponding template $\mathcal{T}_i$, which specifies the key fragments from $\mathcal{K}$ that the node will be given.

The templates of order $K$ form a lattice of height $K + 1$, in which the top element $\top$ corresponds to the template $\top = 1^K$ and the bottom element $\bot$ corresponds to the template $\bot = 0^K$. In this lattice the *meet* of two templates $\mathcal{T}_i$ and $\mathcal{T}_j$ is the template $\mathcal{T}_{i\wedge j}$ corresponding to the bitwise AND of the two templates. Similarly the *join* of these two templates is $\mathcal{T}_{i\vee j}$ corresponding to their bitwise OR.

A projection is an application of a template to a given sequence. It retains some elements of the original sequence, and discards others.

*Definition 3 (Projection):* A projection is a string over $\Sigma \cup \varepsilon$. Elements of a projection are either key fragments or the null value $\varepsilon$. Projections are key fragment sequences that have been masked by a template to conceal information, by replacing selected key fragment values by null values. Elements of a projection of length $K$ are indexed by $\mathbb{Z}_K$.

The dealer distributes to every node $n_i$ a projection $\mathcal{K}_i$ formed by projecting the master sequence $\mathcal{K}$ onto the template $\mathcal{T}_i$ corresponding to that node.

A projection of length $K$ that contains $q$ key fragments and $K - q$ null elements, can be represented by a key fragment sequence of length $q$. The indices in such a compacted projection can be determined using the template corresponding to that projection.

Throughout this paper we will use the notations $\mathcal{T}_i$ and $\mathcal{K}_i$ to denote the template and projection, respectively, distributed to a node $n_i$. Furthermore, we will denote the template and projection that are shared by nodes $n_i$ and $n_j$ as $\mathcal{T}_{i\wedge j}$ and $\mathcal{K}_{i\wedge j}$ respectively. The template and projection of a 2-adversary comprising nodes $n_i$ and $n_j$ will be denoted as $\mathcal{T}_{i\vee j}$ and $\mathcal{K}_{i\vee j}$, while those of a general $t$-adversary $\mathcal{A}$ will simply be denoted $\mathcal{T}_{\mathcal{A}}$ and $\mathcal{K}_{\mathcal{A}}$.

### C. System Initialization

The dealer $\mathcal{D}$ first chooses a random key fragment sequence $\mathcal{K}$ of order $K$. $\mathcal{D}$ will distribute projections of $\mathcal{K}$ to nodes at initialization. $\mathcal{D}$ also selects an instance identifier $D \in \{0,1\}^*$ to identify the current instance of the scheme, so that nodes can verify that they have both been initialized from the same $\mathcal{K}$.

### D. Key Fragment Distribution

The dealer binds each node $n_i \in N$ publicly to a template $\mathcal{T}_i$. This template is determined by the instance identifier $D$, which identifies this instantiation of the scheme, the node's unique identifier $a_i$, and a random nonce, $salt_i$. Without the salt, the template would be determined by the name alone, and an adversary could choose the key fragments it receives by choosing a name that will receive those fragments. These three values $(D, a_i, salt_i)$ are used to initialize a hash based secure pseudorandom number generator. The output stream of this generator is parsed as a stream of values from $\mathbb{Z}_K$. The first $q$ distinct elements from this stream determine the $q$ bits of $\mathcal{T}_i$ that are set to 1.

Upon constructing the template $\mathcal{T}_i$ corresponding to a node $n_i$ the dealer projects $\mathcal{K}$ onto $\mathcal{T}_i$ to determine $\mathcal{K}_i$. The dealer then distributes $\mathcal{K}_i$, $salt_i$ and $D$ to the newly initialized node $n_i$.

Since $\mathcal{T}_i$ can be determined from the triple $(a_i, D, salt_i)$ the null values of $\mathcal{K}_i$ need not be stored, so it suffices to store the $q$ non-null values of $\mathcal{K}_i$ in sequential order. Their proper location in $\mathcal{K}_i$ can be determined using $\mathcal{T}_i$. This encoding of $\mathcal{K}_i$ allows node $n_i$ to use only $q \cdot m$ bits of storage to represent $q$ key fragments each of length $m$ bits.

The node $n_i$ must also store the system wide constant $D$ and its salt value $salt_i$. We will disregard the storage required for $D$ and $salt_i$ since it is a small constant storage cost that does not vary with the system parameters.

### E. Key Fragment Usage

In our scheme two initialized nodes $n_i$ and $n_j$ can independently discover the shared secret $\mathcal{K}_{i\wedge j}$, from which they may construct a shared key $\kappa_{ij} \in \Psi$. Nodes $n_i$ and $n_j$ first exchange $D$ and their salt values $salt_i$ and $salt_j$ respectively. They then verify that they were initialzed by the same dealer and construct one another's projection templates $\mathcal{T}_i$ and $\mathcal{T}_j$. Node $n_i$ constructs $\mathcal{K}_{i\wedge j}$ by first computing $\mathcal{T}_j$ from $(a_j, D, salt_j)$ and then projecting his projection $\mathcal{K}_i$ onto $\mathcal{T}_j$. Similarly node $n_j$ constructs $\mathcal{K}_{i\wedge j}$ by projecting $\mathcal{K}_j$ onto $\mathcal{T}_i$. Nodes $n_i$ and $n_j$ can then construct a shared key $\kappa_{ij}$ from $\mathcal{K}_{i\wedge j}$ by using the mask generation function

MGF. This is done by setting $\kappa_{ij} = \mathrm{MGF}(\mathcal{K}_{i\wedge j}, k)$. This key $\kappa_{ij} \in \Psi$ can now be used by $n_i$ and $n_j$ to communicate securely.

An adversary attempting to launch a man-in-the-middle attack by substituting salts will fail since both the nodes's names and their salts are combined to form the template. It would be very difficult for an adversary to find a salt that maps another node's name to a template that is covered by its key fragments. The probability that the adversary holds a significant fraction of the key fragments specified in a template chosen from a salt it chooses will be no better than that of a template chosen by the dealer.

## V. Parameter Selection

For strong security against a $t$-adversary, the various parameters of our scheme must be carefully chosen.

- The cardinality of key fragments set $\Sigma$ shared by nodes, must be balanced against the storage required to store these key fragments.
- The size $K$ of the dealer's master sequence $\mathcal{K}$ must be tuned to minimize the probability that a $t$-adversary can compromise the communications of nodes using the system.
- The amount of security obtained from the limited storage $S$ for key fragment storage must be maximized to make efficient use of this valuable resource.

In this section we consider the proper ways to design the system parameters $K$, $q$, and $\sigma$ to provide maximum security in an environment with the parameters $t$ (the adversary strength), and $S$ (the storage limitation of the nodes).

### A. The Adversary's Advantage

An adversary $\mathcal{A}$, having compromised a set of nodes and learned their key fragments, is given a pair of nodes $n_i$ and $n_j$ and is asked to determine the key $\kappa_{ij} \in \Psi$ they share.

We define the *advantage* of a $t$-adversary $\mathcal{A}$ to be the increased probability of it correctly determining $\kappa_{ij}$ using the key fragments it has learned over the probability that it by chance determines the key.

$$\mathrm{Adv}[\mathcal{A}] \stackrel{\mathrm{def}}{=} \Pr[\mathcal{A} \text{ given } \mathcal{T}_{i\wedge j} \text{ outputs } \kappa_{ij}] - \psi^{-1}$$

Figure 3 shows the template lattice with the templates of the adversary $\mathcal{A}$, holding $s$ key fragments and a pair of communicating nodes $n_i$ and $n_j$, sharing $l$ key fragments. The lattice depicts the situation in which the adversary holds $n'$ of the fragments shared between $n_i$ and $n_j$. The remaining $n$ key fragments must be guessed by the adversary for it to compute the shared key. The template $\mathcal{T}_{i\wedge j}$ can be regarded as the join of two disjoint projections $\mathcal{T}_{i\wedge j\wedge\mathcal{A}}$ and $\mathcal{T}_{i\wedge j\wedge\overline{\mathcal{A}}}$. $\mathcal{T}_{i\wedge j\wedge\mathcal{A}}$ is the subset of key fragments shared by nodes $n_i$, $n_j$ and the adversary. $\mathcal{T}_{i\wedge j\wedge\overline{\mathcal{A}}}$, is the subset of key fragments shared by nodes $n_i$ and $n_j$, but not known to $\mathcal{A}$. It is the second projection, $\mathcal{K}_{i\wedge j\wedge\overline{\mathcal{A}}}$, that the adversary must guess in order to determine $\mathcal{K}_{i\wedge j}$, and compute $\kappa_{ij} = \mathrm{MGF}(\mathcal{K}_{i\wedge j}, k)$.

### B. Distribution of the Number of Safe Key Fragments

Safe key fragments are those unknown to the adversary. We will now determine the probability distribution on the number of safe key fragments held by two nodes.

To clarify our presentation we will solve the equivalent problem of determining the size of the intersection of sets $S_x$ that are $q$ size subsets of a common set $T$ of size $K$. Using this notation we can express the distribution of the number of safe keys as the following.

$$|\mathcal{K}_{i\wedge j\wedge\overline{\mathcal{A}}}| = \left| S_i \cap S_j \cap \overline{\bigcup_{x\in\mathcal{A}} S_x} \right|$$

$$\forall x((S_x \subset T) \wedge (|S_x| = q) \wedge (|T| = K))$$

In the remainder of this section we will use this set based notation to simplify our discussion.

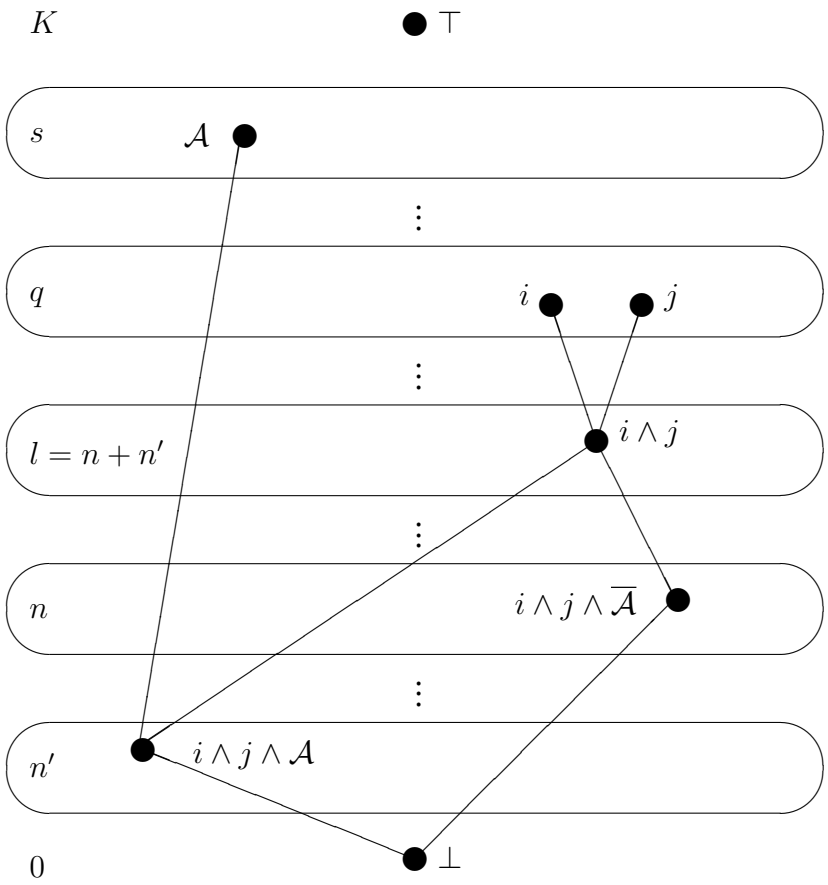


Fig. 3. The template lattice with the templates of an adversary $\mathcal{A}$ and a pair of communicating nodes $n_i$ and $n_j$ shown.

By DeMorgan's Theorem,

$$\left| S_i \cap S_j \cap \overline{\bigcup_{x \in \mathcal{A}} S_x} \right| = \left| S_i \cap S_j \cap \bigcap_{x \in \mathcal{A}} \overline{S}_x \right|$$

Thus, the distribution of safe key fragments can be computed as the intersection of $t+2$ sets, two of which are of size $q$ and correspond to the key fragments held by each node in the communicating pair. The other $t$ sets, of size $K-q$, correspond to the key fragments that are unknown to each of the $t$ nodes compromised by the adversary.

To compute the distribution of the size of this $t+2$-way intersection we must first compute the distribution of the size of a 2-way intersection of sets with sizes described by a probability distribution.

$$\begin{aligned} \Pr[|S_1 \cap S_2| = l] = \sum_{q_1, q_2} & \Pr[|S_1 \cap S_2| = l : |S_1| = q_1, |S_2| = q_2] \\ & \cdot \Pr[|S_1| = q_1] \cdot \Pr[|S_2| = q_2] \end{aligned}$$

An intersection of two subsets $S_1$ and $S_2$ of sizes $q_1$ and $q_2$ can be computed from the following combinatorial counting argument.

$$\Pr[|S_1 \cap S_2| = l : |S_1| = q_1, |S_2| = q_2] = \frac{\binom{q_1}{l} \cdot \binom{K-q_1}{q_2-l}}{\binom{K}{q_2}}$$

The direct computation of the distribution of the number of safe key fragments is expensive for large values of $K$, $q$ and $t$. We have developed approximations to this distribution, which allow us to select the system parameters optimally and to estimate the performance of the system.

In the limit of large $K$ we can make the change of variables $r = q/K$, and solve for the fraction of a node's key fragments that are safe, rather than the number of safe key fragments. The fragments held by each node cover a fraction $r$ of the fragment space. Using the independence of the fragments held by nodes, the probability of a key being in the intersection of these sets is $r^2(1-r)^t$. In the limit it is clear that $r^2(1-r)^t$ of the $K$ key fragments will be shared safely between two nodes. By the central limit theorem, for large K the distribution of safe keys will be normally distributed with mean $r^2(1-r)^t K = r(1-r)^t q$.

We can approximate the variance of the distribution of the number of safe key fragments by approximating it as a binomial distribution with $q$ trials and a trial success probability of $p = r(1-r)^t$. The mean of this approximated distribution is $\mu = p \cdot q = r(1-r)^t q$, which agrees with our previous estimate. This approximation yields an estimate for the variance of $\sigma^2 = p(1-p)q$ [2]. We will use these estimates for the mean and variance of the distribution of the number of safe key fragments to optimize system parameters and estimate the security of the system. In section VI we will compare these estimates with the actual mean and variance of the distribution.

### *C. Security derived from $n$ Safe Keys*

Consider a pair of nodes sharing $l$ key fragments. Let $l-n$ of these fragments be known to the adversary. The adversary can try to guess the other $n$ key fragments that it does not know by selecting a sequence $\hat{x}_n \in \Sigma^n$. The adversary can combine $\hat{x}_n$ with the key fragments it does know, $\hat{x}_{l-n} \in \Sigma^{l-n}$ to form a guess $\hat{x} \in \Sigma^l$. If this guess is correct or, by chance, the key $\hat{\kappa} = \mathrm{MGF}(\hat{x}, k)$ is equal to the actual key shared by the communicating nodes, $\kappa = \mathrm{MGF}(x, k)$, then the adversary has succeeded in breaking the communications between the pair of nodes.

If $x = \hat{x}$ then $\kappa$ will surely be equal to $\hat{\kappa}$ and if $x \neq \hat{x}$ then the pseudo-random property of $\mathrm{MGF}$ assures us that the probability that $\kappa = \hat{\kappa}$ will be $\psi^{-1}\ldots$

$$\begin{aligned} \Pr[\mathrm{MGF}(x,k) = \mathrm{MGF}(\hat{x},k)] &= \sigma^{-n} + (1-\sigma^{-n})\psi^{-1} \\ &\leq \sigma^{-n} + \psi^{-1} \end{aligned}$$

We expect the term $\sigma^{-n} \cdot \psi^{-1}$ to be very small, neglecting this term safely overestimates the adversary's success probability, thus we can approximate the probability $\Pr[\mathrm{MGF}(x,k) = \mathrm{MGF}(\hat{x},k)]$ by neglecting it.

### *D. Approximating the Adversary's Advantage*

The advantage $\mathrm{Adv}[\mathcal{A}]$ of an adversary $\mathcal{A}$ is the increased probability of it correctly determining the key $\kappa_{ij}$ shared between two nodes $n_i$ and $n_j$ using the key fragments it has learned. In section V-C we determined the probability that an adversary succeeds in guessing the key shared between nodes given the number $n$ of safe keys shared between the nodes. We can now solve for the adversary's probability of guessing the key by summing over the distribution of safe key fragments discussed in section V-B

$$\begin{aligned} &\Pr[\mathcal{A} \text{ given } \mathcal{T}_{i\wedge j} \text{ outputs } \kappa_{ij}] \\ &= \sum_n \begin{array}{c} \Pr[\mathrm{MGF}(x,k) = \mathrm{MGF}(\overline{x},k)] \\ \cdot \Pr[|\mathcal{K}_{i\wedge j\wedge\overline{\mathcal{A}}}| = n] \end{array} \\ &= \sum_n \sigma^{-n} \cdot \Pr[|\mathcal{K}_{i\wedge j\wedge\overline{\mathcal{A}}}| = n] + \psi^{-1} \end{aligned}$$

Subtracting the probability of guessing the key $\psi^{-1}$ yields the adversary's advantage.

$$\mathrm{Adv}[\mathcal{A}] = \sum_n \sigma^{-n} \cdot \Pr[|\mathcal{K}_{i\wedge j\wedge\overline{\mathcal{A}}}| = n]$$

We have presented a binomial distribution approximation of the distribution of the number of safe key fragments. This binomial approximation has $q$ trials and a trial success probability $p = r(1-r)^t$, which corresponds to a mean $\mu = p \cdot q$ and a variance $\sigma^2 = p(1-p) \cdot q$. The central limit theorem tells us that for large $q$ this will be approximately normally distributed. We can use this normal approximation to determine the mean, $\overline{\mu}$, and variance, $\overline{\sigma^2}$, of $\mathrm{Adv}[\mathcal{A}]$.

$$\begin{aligned} \overline{\mu} &= E[2^{-mn}] &&= \textstyle\sum_n 2^{-mn} \Pr[|\mathcal{K}_{i \wedge j \wedge \overline{\mathcal{A}}}| = n] \\ &&&\approx \textstyle\int_{-\infty}^{\infty} 2^{-mx} N\left[\mu, \sigma^2\right](x) dx \\ &&&= 2^{-m(\mu - \frac{\ln(2)}{2} m\sigma^2)} \end{aligned}$$

$$\begin{aligned} \overline{\sigma^2} &= E[(2^{-mn} - \overline{\mu})^2] &&= E[2^{-2mn}] - \overline{\mu}^2 \\ &\approx 2^{-2m\mu}(2^{2\ln(2)m^2\sigma^2} &&- 2^{\ln(2)m^2\sigma^2}) \end{aligned}$$

These integrals were computed using Wolfram Research's Integrator web site [21].

### *E. Minimizing The Adversary's Advantage*

We define two related measures of the security of the system, the *benefit* and the *cost*. The benefit is defined to be $-\lg(\overline{\mu})/S$, and represents the effective key length that the system provides per bit of storage required per node. The cost is defined to be the inverse of the benefit, and represents the number of bits required for the system to provide each bit of effective key strength. Using the above expressions for $\overline{\mu}$, $\mu$, $\sigma^2$, and $S = mq$ we arrive at the following expression for the benefit of the system...

$$\begin{aligned} \mathrm{BENEFIT} &= (1 - \frac{\ln(2)}{2} m(1 - r(1-r)^t)) \cdot r(1-r)^t \\ \mathrm{COST} &= \mathrm{BENEFIT}^{-1} \end{aligned}$$

These measures will allow us to quantify the storage efficiency and scaling properties of our scheme.

### *F. Optimal Parameter Selection*

We will now discuss the optimal choice of the system parameters.

*1) Optimal Selection of $\sigma$:* Given storage $S$, we must determine whether it is better to use a small number of large key fragments or a large number of small key fragments. The two extremal cases that represent these conflicting philosophies are $\Sigma = \Psi$ and $\Sigma = \{0, 1\}$. In the former case $m = k$, and each key fragment is an entire key, so a single securely shared key fragment provides full security. In the latter case $m = 1$, so each key fragment is a single bit, and many key fragments are required. We show that the appropriate selection of $\sigma$ is $\sigma = 2$, corresponding to the use of many small key fragments.

We have seen that for a fixed distribution constant $r = q/K$, the expected fraction of key fragments stored at a node that are not held by a $t$-adversary is $r(1-r)^t$. Regardless of the size of key fragments the expected number of bits shared between two nodes is $r(1-r)^t S$. When large key fragments are shared, however, the variance of the distribution has a negative effect on the security of the system. Although the security provided by sharing exactly $X$ bits of information provides the adversary with an advantage of $2^{-X}$, the expected amount of security provided by sharing a distribution of bits with mean $X$ is not $2^{-X}$. This is due to the fact that the adversary's advantage grows exponentially with decreasing $X$. Thus the variance of the distribution of the number of bits securely shared between two nodes has an effect on the security provided by this shared information. As we saw in the previous section, if the number of secretly shared key fragments has mean $\mu$ and variance $\sigma^2$ then the expected security obtained from these fragments is $\overline{\mu} \approx 2^{-m(\mu - \frac{\ln(2)}{2} m\sigma^2)}$. The term being subtracted from $\mu$ has a negative effect on the benefit. Since $\sigma^2$ is independent of $m$ we see that the expected security decreases with increasing $m$. Thus the expected security is maximized when $m = 1$ and we use the smallest possible key fragments.

The intuitive reasoning behind this result is that the granularity of information sharing increases with smaller key fragments. When $m$ is large the effect of having less than the mean number of key fragments safely shared is increased. Allowing $m$ to be small however allows nodes to hold a few fragments less than the mean and suffer a smaller impact on security.

| Parameter | Optimal Value |
|---|---|
| $m$ | $1$ |
| $\sigma$ | $2$ |
| $\Sigma$ | $\{0,1\}$ |
| $q$ | $S$ |
| $K$ | $(t+1)\cdot S$ |
| $r$ | $\frac{1}{t+1}$ |

TABLE I
SUMMARY OF OPTIMAL PARAMETER SELECTIONS

*2) Optimal Selection of $K$:* Each node can store $q = \lfloor S/m \rfloor$ $m$-bit key fragments. The optimal number of key fragments $K$ chosen by the dealer affects the likelihood that the key fragments held by nodes coincide, and also affects the number of distinct key fragments available to the adversary. We wish to choose $r = q/K$ to maximize the security obtained from our bounded storage S, thus we will choose the $r$ that maximizes $\mathrm{BENEFIT}$. Recall from section V-E that

$$\begin{aligned}\mathrm{BENEFIT} &= (1 - \frac{\ln(2)}{2} m(1 - r(1-r)^t)) \cdot r(1-r)^t \\ &= (1 - \frac{\ln(2)}{2} m) \cdot r(1-r)^t + \frac{\ln(2)}{2} m r^2 (1-r)^{2t}\end{aligned}$$

For a given $m$ and $t$ we can numerically solve for the maximum value with $r \in [0,1]$. We observe however that the final term that is $O(r^2(1-r)^{2t})$ has a small effect on the position of the maximum of this function, and by neglecting this term we are conservatively estimating the actual benefit of the system. We can thus approximate $\mathrm{BENEFIT}$ as $C_1 \cdot r(1-r)^t$ for some constant $C_1$. This corresponds to the expected fraction of key fragments that are unknown to a $t$-adversary. Taking the derivative of $r(1-r)^t$ and solving for the maximum, we determine that $r = \frac{1}{1+t}$. This yields the optimal parameter selection $K = r^{-1} \cdot q = (t+1)q$. Intuitively, this optimal value for $r$ is reasonable, since key fragments must be distributed more conservatively to counteract a stronger adversary.

Table I summarizes the optimal parameter values we have computed for our system.

*3) System Scaling With $t$:* Substituting the optimal parameter choices into the $\mathrm{BENEFIT}$ equation we can observe that it is a function only of the adversary strength $t$. This shows that under optimal operating conditions the effective key strength of our system grows linearly with the storage, and that the slope of this linear growth is determined by the strength of the adversary. To observe how the benefit of the system grows for strong adversaries we take a limit of $\mathrm{BENEFIT}$ as $t$ tends toward infinity.

$$\begin{aligned}\lim_{t\to\infty} \mathrm{BENEFIT} &= \frac{2-\ln(2)}{2} \cdot \frac{(\frac{t}{t+1})^t}{t+1} + \frac{\ln(2)}{2} \cdot \frac{(\frac{t}{t+1})^{2t}}{(t+1)^2} \\ &= \frac{2-\ln(2)}{2e(t+1)} + \frac{\ln(2)}{2e^2(t+1)^2} \\ &= O(1/t)\end{aligned}$$

Thus the amount of storage required to achieve a constant level of security against a $t$-adversary increases linearly with $t$.

## VI. SECURITY ANALYSIS

In section V-E we determined the optimal choice of parameters for our key distribution scheme. In this section we will analyze how well our scheme performs. We verify the feasibility of our scheme, and validate our analysis, by numerically computing both the exact security, and our approximations of it.

| Parameters | | Actual | | Approximation | |
|---|---|---|---|---|---|
| S | t | $\mu$ | $\sigma^2$ | $\mu$ | $\sigma^2$ |
| 32 | 1 | 8.000 | 4.047 | 8.000 | 6.000 |
| 64 | 1 | 16.000 | 8.047 | 16.000 | 12.000 |
| 128 | 1 | 32.000 | 16.047 | 32.000 | 24.000 |
| 256 | 1 | 64.000 | 32.047 | 64.000 | 48.000 |
| 32 | 2 | 4.741 | 3.356 | 4.741 | 4.038 |
| 64 | 2 | 9.481 | 6.692 | 9.481 | 8.077 |
| 128 | 2 | 18.963 | 13.364 | 18.963 | 16.154 |
| 256 | 2 | 37.926 | 26.709 | 37.926 | 32.207 |
| 32 | 3 | 3.375 | 2.674 | 3.375 | 3.019 |
| 64 | 3 | 6.750 | 5.337 | 6.750 | 6.038 |
| 128 | 3 | 13.500 | 10.663 | 13.500 | 12.076 |
| 256 | 3 | 27.000 | 21.315 | 27.000 | 24.152 |
| 32 | 4 | 2.621 | 2.198 | 2.621 | 2.407 |
| 64 | 4 | 5.243 | 4.390 | 5.243 | 4.813 |
| 128 | 4 | 10.486 | 8.774 | 10.486 | 9.627 |
| 256 | 4 | 20.972 | 17.542 | 20.972 | 19.254 |

TABLE II

EXACT AND APPROXIMATE VALUES FOR THE MEAN AND VARIANCE OF THE DISTRIBUTION OF THE NUMBER OF SAFE KEY FRAGMENTS SHARED BETWEEN TWO NODES. ALL RESULTS USE THE OPTIMAL PARAMETERS GIVEN IN TABLE I.

| $t$ | Storage(bits) | Approx. Storage(bits) |
|---|---|---|
| 1 | 461 | 692 |
| 2 | 852 | 1226 |
| 3 | 1238 | 1759 |
| 4 | 1621 | 2292 |
| 5 | 2004 | 2824 |
| 6 | 2386 | 3357 |
| 7 | | 3889 |
| 8 | | 4422 |

TABLE III

STORAGE REQUIRED TO PROVIDE SECURITY, AGAINST A $t$-ADVERSARY, COMPARABLE TO A 128-BIT SHARED SYMMETRIC KEY. THE LEFT COLUMN GIVES ACTUAL RESULTS COMPUTED NUMERICALLY. THE RIGHT COLUMN GIVES APPROXIMATIONS FROM THE EQUATIONS DERIVED IN SECTION V-D. ALL RESULTS USE THE OPTIMAL PARAMETERS GIVEN IN TABLE I. NOT ALL OF THE ACTUAL RESULTS WERE COMPUTED DUE TO THE EXPENSIVE OF THESE CALCULATIONS.

### *A. Distribution of Safe Key Fragments*

In section V-B we estimated the mean and variance of the distribution of the number of safe key fragments shared between two nodes as $r(1-r)^t \cdot q$ and $(r(1-r)^t - r^2(1-r)^{2t}) \cdot q$ respectively. Also, in section V-B we show how the actual distribution can be computed. We now compare our approximations to the actual mean and variance calculated from the distribution for several values of $S$ and $t$, where we choose the system parameters optimally as described in section V-E.

Table II shows both the actual and estimated mean and variance. These values show that our estimate for the mean is very accurate, and that our estimate for the variance is consistently larger than the actual variance. Even for these relatively small parameters the asymptotic bounds are already manifest and the mean and variance grow linearly with $S$ and inversely with $t$. Our approximations scale well to the large values of $S$ and $t$ that are useful in practice. Unfortunately, the numerical methods we use to determine the actual mean and variance scales as $O((t \cdot S)^{3 \lg(t)})$.

### B. The Adversary's Advantage

We present results from our numeric calculations, and our approximations, on the advantage of a $t$-adversary. Table III shows the amount of storage required to achieve a level of security equivalent to that given by a $128$-bit symmetric key for several values of $t$. These results show that our approximations consistently overestimate the storage required. Both the actual and approximate values agree with our observation in section V-F.3, that the cost of achieving constant security grows linearly with $t$.

An alternative approach to estimating the security of the system based solely on our approximations is to compute the actual results for a smaller set of system parameters and extrapolate the desired system security from the result. Although this approach potentially achieves more accurate results than our loose approximations, we do not recommend it since it does not yield a conservative estimate. If an exact result is required for an implementation with tight design constraints our approximations provide a good starting point for narrowing the search space.

## VII. Implementation Considerations

In the previous sections we have described the operation of our system in an relatively simple environment. In this section we extend our scheme with several features that make it more practical.

First, we describe extensions to allow nodes to dynamically join the network. If they can be initialized by $\mathcal{D}$, joins are covered by the basic scheme. If, however, nodes must be added when the dealer is not available, we describe a method for existing members of the network to work together to simulate the initialization of a node by the dealer.

Second, we describe how to extend our scheme to allow for heterogeneous node populations. In our basic scheme the device with the smallest available storage dictates security system-wide. In networks with widely variable device characteristics this may be highly undesirable. We describe briefly how to distribute varying numbers of key fragments to nodes, and describe how the assumptions on the adversary must change to justify this change.

### A. Dynamic Network Joins

We will now describe a mechanism by which dynamic network membership can be supported by allowing a group of initialized nodes to corporate to distributively initialize a new node $n_i$ by helping it construct a valid projection $\mathcal{K}_i$.

Our dynamic join algorithm requires the joining node $n_i$ to identify a set of nodes $\mathcal{S}_i$ that are able and willing to authenticate $n_i$ for admission to the network. It can send each node in $S_i$ a join request, specifying itself, any necessary authentication information required for authentication and a list of the members of $\mathcal{S}_i$. Each node in $S_i$ first authenticates $n_i$, aborting the join if it is unwilling to participate in $n_i$'s initialization. Nodes in $S_i$ must then agree on a projection template $\mathcal{T}_i$ for the initialized node by choosing a salt $salt_i$, and determine whether they have all of the key fragments specified in $\mathcal{T}_i$. If they can reconstruct $\mathcal{K}_i$ they will accept the node's join request. Otherwise, they will choose another salt and try again. If after a fixed number $k$ of such trials they fail to construct a complete projection template for $n_i$ they will deny the join request. If they succeed in finding a $\mathcal{K}_i$ each member holding a key fragment from this projection securely distributes this information to the joining node along with the corresponding salt value. If $n_i$ receives a complete projection map and corresponding salt, it is able to join the network. Otherwise it deletes all shares and tries again later.

We must determine the size of $S_i$ for a given system $(\sigma, q, K)$ to yield a non-negligible probability that the join process will succeed after trying only a reasonable number of different salts. Nodes joining the network may have a difficult time authenticating themselves with enough initialized nodes, if $S_i$ is large. to successfully join. Unfortunately, a small $S_i$ diminishes system security. This is an inherent trade-off between the system security and the cost associated with dynamic joins.

Approximately $1621$ bits of storage are required to provide security equivalent to a $128$ shared key, against a $4$-adversary. With these parameters $29$ nodes must be in $S_i$ in order to successfully admit nodes

into the system. To protect against an $8$-adversary twice the amount of storage is needed and the size of a successful $S_i$ doubles to $61$.

### *B. Heterogeneous Share Distribution*

Distributing equal shares of $\mathcal{K}$ to all nodes produces a system in which all pairs of nodes achieve the same level of security. While this is desirable in a network consisting of a large number of relatively homogeneous devices it does present problems in a network of heterogeneous nodes. An alternate approach is to classify nodes by their storage resources, and their susceptibility to attack by the adversary. Nodes capable of protecting their key fragments better against the adversary can be given larger shares of $\mathcal{K}$. This will increase the security of the communications between them and any other node in the system.

It is also important to limit the adversary's ability to masquerade as a node with a smaller than standard number of key fragments, tricking a node into using an insecure key. This could result in the adversary learning key fragments from successful attacks. The most secure way of preventing this is for node $n_i$ to know exactly how many key fragments node $n_j$ has been given before attempting to construct a secure key with $n_j$.

## VIII. Conclusion

We have presented a novel approach that allows pairs of nodes to generate a shared key using a limited amount of stored information, distributed to them by an off-line dealer. Our approach provides security against an adversary capable of capturing a bounded number of network nodes. We accomplish this without requiring the use of expensive assymetric cryptosystems, which are infeasible for low- and medium-range devices. Our scheme is efficient since it makes use of symmetric key cryptography and secure hash functions. Since the trusted authority in our scheme is off-line, it is particularly well suited for deployment in MANETs, where connectivity is not guaranteed. Against a $6$-adversary (capable of capturing at most $6$ nodes), for example, our scheme provides security comparable to a $128$ bit symmetric key strength using approximately $299$ bytes of information stored at each node. As the adversary's strength grows, our scheme requires a linear storage increase, to provide a constant level of security. Our scheme provides an important alternative to existing approaches, which fail to meet the requirements of efficiency, and security against colluding adversaries, in a network with poor connectivity.

## Acknowledgment

The authors would like to thank our colleagues working on the DARPA funded SWiFT project for their valuable input into the development and analysis of our work.

# REFERENCES


[1] M. Bellare, P. Rogaway, “Optimal Assymetric Encryption – How to Encrypt with RSA”, EUROCRYPT’94, LNCS 950, pages 92–111. Springer-Verlang, Berlin, 1995.

[2] G. Bloch, S. Greiner, H. de Meer, K. Trivedi, *Queuing Networks and Markov Chains*, John Wiley & Sons, 1998.

[3] J. Buhler, M. Tompa, “Finding Motifs Using Random Projections”, In *proceedings of the 5th Int’l Conference on Computational Molecular Biology*, April 2002. pp. 67-74.

[4] H. Chan, A. Perrig, and D. Song, “Random key predistribution schemes for sensor networks”, *IEEE Symposium on Security and Privacy*, 2003.

[5] H. Chan, A. Perrig, “Pike: Peer intermediaries for key establishment in sensor networks”, *INFOCOM*, 2005.

[6] J. Daemen, V. Rijmen, “AES Proposal: Rijndael”, AES Algorithm Submission, September 1999.

[7] W. Diffie, M.E. Hellman, “New Directions in Cryptography,” *IEEE Transactions on Information Theory*, v. IT-22, n. 6, November 1976, pp. 644-654.

[8] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney, “A key management scheme for wireless sensor networks using partial deployment knowledge”, *INFOCOM*, 2004.

[9] W. Du, J. Deng, Y. Han, and P. Varshney, “A pairwise key predistribution scheme for wireless sensor networks”, *ACM Conference on Computer and Communications Security*, 2003.

[10] L. Eschenauer, V. Gligor, “A Key-Management Scheme for Distributed Sensor Networks”, *ACM conference on Computer and communications security*, 2002.

[11] Kohnfelder, “Toward a Practical Public Key Cryptosystem”, Bachelor’s thesis, MIT Department of Electrical Engineering, May 1978.

[12] D. Liu, P. Ning, “Establishing pairwise keys in distributed sensor networks”, *ACM Conference on Computer and Communications Security*, 2003.

[13] D. Liu, P. Ning, “Location-based pairwise key establishments of static sensor networks”, *ACM workshop on Security in Ad Hoc and Sensor Networks*, 2003.

[14] D. Malkhi, M.K. Reiter, A. Woll, and R.N. Wright. “Probabilistic Byzantine Quorum Systems”. Brief announcement in *Proceedings of the 16th ACM Symposium on Principles of Distributed Computing*, June 1998.

[15] National Institute of Standards and Technology (NIST), Secure Hash Standard, Federal Information Processing Standards Publication 180-1, April 1995.

[16] National Institute of Standards and Technology (NIST), Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, November 2001.

[17] R. Rivest, A. Shamir, L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems” CACM 21(2): 120-126 (1978).

[18] R. Rivest, “The MD5 Message-Digest Algorithm”, RFC 1321, April 1992.

[19] RSA Data Security, Inc. Public Key Cryptography Standards – PKCS.

[20] B. Schneier, *Applied Cryptography, 2nd edition.* John Wiley & Sons, 1995.

[21] Wolfram Research, Inc. The Integrator, http://integrals.wolfram.com.