Broadcast-enhanced Key Predistribution Schemes

Michelle Kendall, Keith M. Martin, Siaw-Lynn Ng,
Maura B. Paterson, Douglas R. Stinson*

May 24, 2012

Abstract

We present a formalisation of a category of schemes which we call *Broadcast-enhanced Key Predistribution Schemes*. These schemes can be used instead of a key predistribution scheme in any network which has access to a trusted base station and broadcast channel. In such networks, broadcast-enhanced key predistribution schemes can provide advantages over key predistribution schemes including flexibility and more efficient revocation. There are many possible applications and ways to implement broadcast-enhanced key predistribution schemes, and we propose a framework for describing, comparing and analysing them.

In their paper 'From key predistribution to key redistribution', Cichoń, Gołębiewski and Kutyłowski propose a scheme for 'redistributing' keys to a wireless sensor network using a broadcast channel after an initial key predistribution. We classify this as a broadcast-enhanced key predistribution scheme and analyse it in that context. We provide simpler proofs of some results from their paper, give a precise analysis of the resilience of their scheme, and discuss modifications based on defining a suitable keyring intersection threshold. In the latter half of the paper we suggest two particular scenarios where broadcast-enhanced key predistribution schemes may be particularly desirable and the relevant design goals to prioritise in each case. For each scenario we propose a suitable family of broadcast-enhanced key predistribution schemes and our analysis demonstrates their effectiveness in achieving their aims in resource-constrained networks.

1 Introduction

In [6], Cichoń et al. propose a 'key-redistribution' scheme for wireless sensor networks which makes use of a trusted base station and broadcast channel to distribute and update keys. This can provide benefits over 'static' key predistribution schemes which are typically designed for networks which do not have access to a broadcast channel or a trusted base station after deployment. There is a variety of potential applications for such 'redistribution' schemes, and accordingly, many different ways to design them. We therefore propose the category of *broadcast-enhanced key predistribution schemes* (BEKPSs), and define a framework for their design and analysis.

We will introduce BEKPSs in this section and suggest motivations for studying them. We begin by introducing the ideas and terminology of two closely related concepts, key predistribution schemes and broadcast encryption, before defining BEKPSs themselves.

1.1 Key predistribution schemes

A *key predistribution scheme* (KPS) is a method for a trusted base station to preload symmetric keys onto devices or 'nodes' before they are deployed into an environment to create a network.

*Research supported by NSERC grant 203114-2011

Key predistribution is a technique particularly suited to resource-constrained environments where public key cryptography is infeasible and there is no method for distributing symmetric keys once the network is operational, in particular for networks where a secure channel cannot be established between the network and base station after deployment. A major drawback of KPSs is that once the keys have been predistributed, subsequent key management operations are challenging to conduct [2]. We will present examples of KPSs in Section 2.1.

1.2 Broadcast encryption

There are many applications where it is possible for a trusted base station to use a broadcast channel to communicate with nodes during the operational phase of the network. This broadcast channel can be used not only to distribute content, but also to conduct key management operations. Such applications have been widely studied in the context of *broadcast encryption.* The first broadcast encryption schemes were given in [1, 10]. A classic example of a broadcast encryption application is pay-TV systems, where a key predistribution scheme is used to install keys into set-top boxes during the initialisation phase. The access to content is then managed by broadcasting the encrypted content along with a key management 'header' whose purpose is to provide an additional key 'layer' of content keys. The combined use of the predistributed keys and content keys defines the set of users which are able to decrypt and hence view the content. Note that whilst a pair of users may at times share keys, there is no motivation in the design of the scheme for users to be able to communicate with each other; the purpose of a broadcast encryption scheme is to control access to content.

1.3 Broadcast-enchanced key predistribution

We define a *broadcast-enhanced key predistribution scheme* (*BEKPS*) to be a key distribution scheme designed for a network where a trusted base station and broadcast channel will be available. We distinguish between the *underlying keys* which are predistributed to the nodes, and the *temporal keys* which are broadcast by the base station and which the nodes may use for communication until the next broadcast. The base station broadcasts the temporal keys by encrypting them using underlying keys, as we will describe in Section 2.

Broadcast encryption can be regarded as a type of BEKPS, where temporal key sharing is incidental. In this paper we consider BEKPSs for applications where communication *between* nodes is important, for example in networks of data-gathering nodes, and so temporal key sharing is one of the primary design goals. Such differences of purpose create substantial differences between the designs of typical BE schemes and general BEKPSs.

There are several advantages of deploying broadcast-enhanced key predistribution as opposed to the use of a basic KPS:

- **Flexibility:** Underlying keys may be allocated in a way which is undesirable, either because there was a lack of control over the initial deployment, or because the purpose or priorities of the network have changed over time. For example, as node batteries become drained, it may be desirable to reduce the burdens on remaining nodes by distributing fewer temporal keys and maintaining a connected network. A BEKPS enables the base station flexibility to ensure that some undesirable properties of the underlying key predistribution do not persist in the temporal key distribution. This is clear when we consider that the number of temporal keys shared by two nodes can be greater than, equal to, or less than the number of underlying keys which they share. Changes may be temporary, and hence only effective between a small number of updates, or permenantly sustained by future updates.

- **Ease of revocation:** In any BEKPS it is possible to revoke a node by ensuring it does not receive any future temporal keys. This can be done simply by omitting that node's underlying keys from the set of underlying keys used to encrypt all future temporal key broadcasts. This straightforward approach benefits from reducing the base station's future broadcast load. However, it has the potential to reduce the connectivity and resilience of the remaining network (we define these terms in Section 2), and repeated revocations may lead to rapid degeneration of the remaining network [6]. In Section 4 we discuss a practical way to design a BEKPS for efficient revocation, where repeated revocations increase the broadcast load but do not lead to network degeneration.

- **Creating hierarchy in the temporal key distribution:** The distribution of the temporal keys may coincidentally or deliberately feature 'imbalances', for example, certain nodes may store more keys than average. Nodes which store extra keys may be desirable for efficient routing of information through the network, and indeed KPSs have been proposed for heterogenous networks; see [3, 12] for a brief survey. In homogeneous networks (where all nodes have identical hardware), having comparatively more keys brings with it the disadvantages of increased communication burdens and quicker battery drainage. One way to reduce the damage that this causes to the network is to change at regular intervals the nodes which are required to store extra keys, as in the election of cluster heads in a network - see [16].

  In any network where some nodes store extra keys, the compromise of such a node will be more detrimental to the resilience of the network than the compromise of the average node. In Section 5 we propose a family of BEKPSs which provide the benefits of efficient routing found in hierarchical networks, whilst frequent temporal key updates reduce the resilience risks and battery drainage.

In the next section we will see an example of a BEKPS and define the model, setting and metrics for analysing BEKPSs in more detail. Section 3 provides some simpler versions of proofs from [6] and discusses modifications of the Cichon et al. BEKPS based on defining a suitable intersection threshold.

We then present two families of BEKPSs designed for particular scenarios: in Section 4 we present BEKPSs designed for efficient revocation of nodes, and in Section 5 we present BEKPSs which harness some of the advantages of hierarchical networks in a homogeneous network of nodes. We include these two families of schemes as examples of the many applications of BEKPSs, and to demonstrate that explicit formulae can be derived for the connectivity, resilience and broadcast load, allowing for precise and efficient analysis of BEKPSs. We conclude in Section 6 and present ideas for future work.

2 Framework

In this section we propose the framework for a BEKPS protocol by describing our model and setting, defining the relevant notation and metrics for our analysis, and providing examples.

First, we briefly revisit key predistribution schemes, introducing notation and examples which will be referred to throughout the rest of the paper.

2.1 Key predistribution schemes

A *key pool* $\mathcal{K}$ of n symmetric keys $\{K_1, K_2, \ldots, K_n\}$ is selected from the space of all possible keys, and each node $N_i \in \{N_1, N_2, \ldots, N_v\}$ is allocated a subset of keys from the key pool. A KPS is a method for allocating a set of keys to each node. The size of the key pool and the number of keys allocated to each node are chosen to supply a trade-off between the following conflicting metrics:

- **Key storage:** the number of keys which a node is required to store. It is usually desirable to minimise key storage.
- **Connectivity:** a measure of the proportion of nodes which share keys. In common with many other papers, we will denote connectivity by Pr_1, the probability that a randomly picked pair of nodes are 'connected', or 'form a link'. In many schemes, this is simply the probability that a randomly picked pair of nodes share a single common key. Some schemes, an example of which is given in Example 2.2, require a pair of nodes to share $\eta > 1$ keys before they are connected. We generally want to maximise connectivity.
- **Resilience:** a measure of the network's ability to withstand node compromise. We calculate resilience by fail_s, the probability that a link between two uncompromised nodes N_i, N_j is insecure after s other nodes are compromised. This measure is used in [5, 7], though the notation fail_s developed later. For 'good' resilience we wish to minimise fail_s.

The following example is a KPS to which we will frequently refer throughout this paper: the random key predistribution scheme proposed in the seminal paper by Eschenauer and Gligor [9].

Example 2.1 (Eschenauer Gligor KPS)**.** Every node is assigned a random σ-subset of keys chosen from a given pool $\mathcal{K}$ of n symmetric keys. Two nodes N_i and N_j are connected if they have at least one common key. If they have more than one common key, they should randomly select one of them to encrypt their communications. The probability of two nodes sharing at least one key is

$$\mathsf{Pr}_1 = 1 - \frac{\binom{n-\sigma}{\sigma}}{\binom{n}{\sigma}} \ . \tag{1}$$

The resilience is given by

$$\mathsf{fail}_s = 1 - \left(1 - \frac{\sigma}{n}\right)^s \tag{2}$$

after s nodes have been compromised [5].

To improve the resilience of the Eschenauer Gligor KPS, we can make use of the fact that some nodes may share more than one common key. Suppose that N_i and N_j have exactly $\omega \geq 1$ common keys, say $K_{i_1}, \ldots, K_{i_\omega}$, where $i_1 < i_2 < \cdots < i_\omega$. Then they can each compute the same pairwise secret key,

$$K_{ij} = h(K_{i_1} \parallel \ldots \parallel K_{i_\omega} \parallel i \parallel j),$$

using an appropriate public key derivation function h, which has suitable input and output sizes. Such key derivation functions could be constructed from a secure public hash function, e.g., SHA-1. This leads to a modification of the Eschenauer Gligor scheme:

Example 2.2 (η-composite scheme)**.** Chan, Perrig and Song [5] propose the η-composite scheme, a modification of the Eschenauer Gligor scheme. Each pair of nodes may compute a pairwise key only if they share at least η common keys, where the integer $\eta \geq 1$ is a pre-specified *intersection threshold*. Given that two nodes have at least η common keys, they use all their common keys to compute their pairwise key, by means of an appropriate key derivation function, as described above. If the size of the key pool is kept constant, this modification will reduce the connectivity and increase the resilience of the network.

The other main approach used for key predistribution is deterministic. There are many different deterministic KPSs, including those based on combinatorial designs, graph constructions, Blom schemes, and Reed Solomon codes. See [3, 13, 15] for overviews of these kinds of schemes. As well as providing different trade-offs between the metrics, deterministic schemes can provide advantages such as more efficient shared key discovery (defined in Section 2.2). The BEKPSs which we present and analyse in Sections 4 and 5 are based on the random key predistribution scheme of Example 2.1, but in principle we could instead have used any KPS.

We now define our model for a BEKPS.

2.2 BEKPS model

We propose BEKPSs for networks of v nodes, $N_1, N_2, \ldots, N_v$, a trusted authority that preloads the underlying keys onto the nodes, and a (not necessarily distinct) trusted base station that can broadcast to all nodes using a broadcast channel.

A BEKPS protocol is comprised of the following phases:

1. **Underlying key predistribution:** Each node N_i is allocated a set of *underlying keys* from the underlying key pool $\mathcal{K}_v = \{u_1, u_2, \ldots, u_n\}$ before deployment, according to a key predistribution scheme. Underlying keys are solely for the purpose of encrypting and decrypting temporal keys, and should not be used for node to node communication. Once the nodes are deployed, we assume that the underlying keys are fixed and cannot be altered by the base station or overwritten by the nodes. (We justify this assumption by noting that if it were possible to securely supply nodes with new underlying keys, then the resulting system could simply be considered as an entirely new BEKPS and analysed within our model.)

2. **Temporal key distribution:** After the nodes are deployed, the base station broadcasts *temporal keys* to them in order for them to communicate. Each node is allocated a set of temporal keys from a temporal key pool $\mathcal{K}_\tau = \{t_1, t_2, \ldots, t_m\}$. The temporal keys are broadcast to the nodes encrypted by underlying keys, so that a node learns a temporal key if and only if the temporal key is encrypted by an underlying key known to that node.

3. **Shared key discovery:** Once the temporal keys have been broadcast, a shared key discovery protocol such as one of those given in [3, 19] can be used so that each node establishes the set of other nodes with which it shares keys. As in KPSs, if the temporal keys are assigned in a way known to all the nodes, then a node N_i can broadcast information about its identity, its *node identifier*, from which any node N_j can derive the list of temporal key identifiers $\{ID(t_{i1}), ID(t_{i2}), \ldots\}$ which correspond to N_i's temporal key set $\{t_{i1}, t_{i2}, \ldots\}$. It then remains for each node to look up whether any of these temporal keys are also known to them. If temporal keys are not assigned in a deterministic or publicly known way, then each node has to broadcast its whole list of key identifiers in order to perform shared key discovery.

   All BEKPSs described in this paper use variations on the random KPS of Example 2.1, so shared key discovery requires all nodes to broadcast their key identifiers, unless the assignment of keys is made public. Since this does not vary throughout the paper, we will generally omit a description of this phase when defining our BEKPSs.

4. **Temporal key update:** A new temporal key pool may be generated and new sets of temporal keys broadcast as often as desired, according to the constraints of the network.

We now present an example of a BEKPS: the 'key redistribution' scheme of Cichoń et al. [6].

Example 2.3. Underlying keys are distributed randomly, as in an Eschenauer Gligor KPS [9]: a key pool of underlying keys $\mathcal{K}_\upsilon = \{u_1, u_2, \ldots, u_n\}$ is generated, and each node is allocated a random k-subset of $\mathcal{K}_\upsilon$. A temporal key pool $\mathcal{K}_\tau = \{t_1, t_2, \ldots, t_m\}$ of $m = n/c$ temporal keys is generated, where c is a small constant. Each temporal key is encrypted using c underlying keys, and the base station then broadcasts the encrypted temporal keys to the network.

In general, the choice of which underlying keys should encrypt each temporal key can be made randomly or deterministically. In the Cichoń et al. scheme, the underlying keys which should be used to encrypt each temporal key are chosen in a pseudorandom way. That is, we take a pseudorandom bijection π between $\{1, \ldots, m\} \times \{1, \ldots, c\}$ and $\{1, \ldots, n\}$ and encrypt t_i using $u_{\pi(i,1)}, \ldots, u_{\pi(i,c)}$. Presumably, a new π is chosen before each update. To simplify the notation, we relabel the underlying keys so that $u_1 = u_{\pi(1,1)}, u_2 = u_{\pi(1,2)}, \ldots, u_n = u_{\pi(n/c,c)}$ so that the first c underlying keys encrypt t_1 and so on. Then the base station broadcasts:

$$\begin{array}{cccc}
E_{u_1}(t_1), & E_{u_2}(t_1), & \ldots, & E_{u_c}(t_1) \\
E_{u_{(c+1)}}(t_2), & E_{u_{(c+2)}}(t_2), & \ldots, & E_{u_{2c}}(t_2) \\
\vdots & & & \\
E_{u_{(n-c)}}(t_{n/c}), & E_{u_{(n-c+1)}}(t_{n/c}), & \ldots, & E_{u_n}(t_{n/c})
\end{array}$$

We now describe in more detail the setting for which we design BEKPSs.

2.3 Setting

The networks which we consider are comprised of nodes which are static and homogeneous. In many applications, the nodes will not all be within communication range of each other. To fully analyse and set the parameters for a particular network, therefore, it is necessary to consider both the *key graph* (a representation of the nodes which share keys) and the *communication graph* (representing the nodes within communication range of each other). The intersection of these two graphs indicates the pairs of nodes which can communicate directly and cryptographically protect their link. A common approach used in the KPS literature is to assume a random communication graph, typically the Erdős Rényi model [8] or random geometric model, as in [4]. However, our contributions in this paper relate to the properties of the key graph, and so this is where we perform our analysis. Applying our results to a particular scenario with its corresponding communication graph can be done in the same way as with any key predistribution scheme; for an example, see [4] where a random geometric graph is used to model the sensors' locations.

We assume the existence of a strong adversary who is able to compromise nodes to learn both temporal and underlying keys, and to keep records of all previous transmissions. It should be noted that a BEKPS does not provide backwards and forwards security against such an adversary, though it would do so under a weaker adversary who is only able to obtain temporal keys, for example in networks where underlying keys are stored in tamper-resistant hardware. We suppose that the adversary compromises each node with equal probability.

As with KPSs, we consider BEKPSs for resource-constrained environments where asymmetric cryptography is infeasible. If there were no other constraints on resources then it would be trivial to design a BEKPS with almost any properties:

- If there were no limit to the number of keys a node may store, then every pair of nodes could share a unique underlying key, or indeed every possible subset of nodes could share a unique underlying key. This would make it possible to achieve temporal key sharing across any arbitrary group of nodes, though with the potential for high broadcast requirements.
- If the broadcast size were unlimited, each node could store a single, unique underlying key. The base station could then individually target nodes when broadcasting temporal keys, and achieve any desired combination of shared temporal keys amongst the nodes.

However, such high requirements will not always be feasible. Our focus in this paper will be on BEKPSs for constrained environments such as wireless sensor networks, where key storage and broadcast capability are limited, and where it is desirable for the longevity of the network to minimise the communication and computational requirements of the nodes.

2.4 Metrics

As in KPSs, the resource constraints dictate that there is a trade-off to be made between minimising key storage and maximising connectivity and resilience. For BEKPSs these metrics need to be defined in a little more detail, and we identify two further metrics to consider.

Key storage: The number of keys σ which a node is required to store in its memory is the sum of the number of underlying keys, k, and the number of temporal keys, κ. As in KPSs, key storage should be minimised. We note that in BEKPSs the number of temporal keys that a node is required to store is not necessarily constant over time.

Connectivity: As with KPSs, we measure connectivity by Pr_1, the probability that a randomly picked pair of nodes are connected. That is, the probability that they share at least η keys, where η is the required number of keys specified by the KPS. It is usually desirable to maximise Pr_1 in BEKPSs. Connectivity in the underlying key predistribution is not necessarily required.

Resilience: As with KPSs, resilience is measured by fail_s, the probability that a temporal link between two uncompromised nodes N_i, N_j is insecure after s other nodes are compromised. In this paper we confine our analysis to the computation of fail_1 for the ease of comparing schemes.

Broadcast load: We quantify the number of encrypted temporal keys to be broadcast by the base station at each update, and consider ways to minimise this broadcast load.

Revocation efficiency: Since nodes may develop faults and we assume the presence of an adversary, the ability to revoke keys and/or nodes adds robustness to a network. We will describe nodes which are to be revoked, that is, nodes suspected to be compromised by an adversary, displaying irregularites, or otherwise weakening the network, as ‘compromised nodes’. We will refer to the remaining nodes which (as far as the base station can tell) have not been compromised and are functioning as they should, as ‘uncompromised’.

We analyse a BEKPS’s capability to revoke compromised nodes by the metrics:

- size of broadcast required to revoke r nodes during a temporal key update
- number of uncompromised nodes which lose keys because of the revocation of r compromised nodes

We note that for many subsets of these metrics it is trivial to devise a BEKPS which optimises them. For example, storage, connectivity and broadcast load can be optimised by all nodes storing a single underlying key u_1, with which a single temporal key t_1 is encrypted and broadcast. Nodes would be connected with probability $\mathsf{Pr}_1 = 1$, but resilience would be minimised and revocation of a strict subset of nodes would be impossible. Therefore we are interested in schemes which provide suitable trade-offs between all of these metrics.

3 The BEKPS of Cichoń et al.

We noted in Section 1 that Cichoń, Gołębiewski and Kutyłowski present a technique for 'key redistribution' in sensor networks [6], which we classify as a BEKPS. The details of their scheme are given in Example 2.3. In this section we provide simpler proofs of some of their results (Section 3.1), refine the estimates for the expected number of shared underlying and temporal keys between two nodes (Section 3.2) and give a precise analysis of the resilience (Section 3.3). In Section 3.4 we present some numerical values of our formulae, and finally in Section 3.5 we discuss a modification to the scheme based on defining a suitable intersection threshold.

3.1 Simplifying proofs from [6]

In this section, we give some simplified proofs of results from [6]. We begin by establishing a combinatorial framework.

We have noted that each node contains a k-subset of keys from $\mathcal{K}_v$. The indices of these keys form a k-subset of $X = \{1, \ldots, n\}$ that we term a *block*. Thus, each node can be identified with the block that is associated with the keys that the node holds; henceforth we will use the terms 'node' and 'block' interchangably. Note that every block is a k-subset of $\{1, \ldots, n\}$ that is chosen independently and uniformly at random from the set of all $\binom{n}{k}$ possible k-subsets.

In [6, Theorem 1], formulae are proven for the expected number of shared underlying keys and the expected number of shared temporal keys for two nodes. The proofs given in [6] use some heavy machinery involving generating functions. However, this theorem has a quick, simple proof based on the linearity of expectation of random variables.

First we consider [6, Theorem 1 (part 2)], which asserts that the expected number of temporal keys shared by two nodes is $\frac{n}{c}\left(1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right)^2$. Suppose that $G_1, \ldots, G_{n/c}$ partition the n-set $\{1, \ldots, n\}$ into $m = n/c$ disjoint c-sets. A and B are random blocks. The number of temporal keys shared by A and B is

$$\omega_{A,B} = |\{i : A \cap G_i \neq \emptyset \text{ and } B \cap G_i \neq \emptyset\}|.$$

For $1 \leq i \leq n/m$, define a random variable $\tilde{\mathsf{X}}_i = 1$ if $A \cap G_i \neq \emptyset$ and $B \cap G_i \neq \emptyset$, and define $\tilde{\mathsf{X}}_i = 0$, otherwise. Let $\tilde{\mathsf{X}} = \sum_{i=1}^{n/c} \tilde{\mathsf{X}}_i$. Then $\tilde{\mathsf{X}}$ computes $\omega_{A,B}$ and $E[\tilde{\mathsf{X}}]$ is the expected value of $\omega_{A,B}$. It is obvious that

$$\mathsf{Pr}[A \cap G_i \neq \emptyset] = \mathsf{Pr}[B \cap G_i \neq \emptyset] = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}$$

and hence

$$E[\tilde{\mathsf{X}}_i] = \mathsf{Pr}[A \cap G_i \neq \emptyset \text{ and } B \cap G_i \neq \emptyset] = \left(1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right)^2.$$

By linearity of expectation,

$$E[\tilde{\mathsf{X}}] = \frac{n}{c}\left(1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right)^2, \tag{3}$$

which proves [6, Theorem 1 (part 2)].

To prove [6, Theorem 1 (part 1)] which states that the expected number of underlying keys shared between two nodes is $\frac{k^2}{n}$, we just set $c = 1$ in the formula derived above. We have

$$E[\text{number of shared underlying keys}] = \frac{n}{1}\left(1 - \frac{\binom{n-1}{k}}{\binom{n}{k}}\right)^2 = n\left(1 - \frac{n-k}{n}\right)^2 = \frac{k^2}{n},$$

which proves the desired result.

3.2 Refining estimates

We have reproved the exact formula for the expected number of shared temporal keys. In [6, Corollary 1], an estimate for Equation (3) is given when k is roughly $\sqrt{n}$. However, we can also estimate Equation (3) when $k \neq \sqrt{n}$.

First, we estimate

$$\frac{\binom{n-c}{k}}{\binom{n}{k}} \approx \frac{(n-c)^k}{n^k} = \left(1 - \frac{c}{n}\right)^k,$$

so

$$E[\tilde{\mathsf{X}}] \approx \frac{n}{c}\left(1 - \left(1 - \frac{c}{n}\right)^k\right)^2.$$

Next,

$$\left(1 - \frac{c}{n}\right)^k \approx 1 - \frac{kc}{n} + \frac{k^2c^2}{2n^2},$$

so

$$E[\tilde{\mathsf{X}}] \approx \frac{n}{c}\left(\frac{kc}{n} - \frac{k^2c^2}{2n^2}\right)^2 = \frac{k^2c}{n}\left(1 - \frac{kc}{2n}\right)^2.$$

Finally, if we expand the square and ignore the last term, we get

$$E[\tilde{\mathsf{X}}] \approx \frac{k^2c}{n}\left(1 - \frac{kc}{n}\right). \tag{4}$$

If $k = \sqrt{n}$, then our estimate (4) is

$$\frac{k^2c}{n} - \frac{k^3c^2}{n^2} = \frac{k^2c}{n} - \frac{c^2}{\sqrt{n}}.$$

The estimate given in [6] is

$$\frac{k^2c}{n} + O\left(\frac{1}{\sqrt{n}}\right).$$

However, in [6], c is assumed to be fixed and the big-oh hides an unspecified constant that depends on c.

Our estimate is quite accurate for reasonable values of the parameters. For example, when $n = 10000$, $k = 100$ and $c = 16$, the exact value of $E[\tilde{\mathsf{X}}] = 13.81$ while the estimate (4) is 13.44. Here is another data point: when $n = 10000$, $k = 50$ and $c = 16$, the exact value of $E[\tilde{\mathsf{X}}] = 3.718$ while the estimate (4) is 3.680.

3.3 Refining the calculation of resilience

For the analysis in this section we consider the resilience of the Cichoń et al. BEKPS during a single broadcast phase, that is, during a time period where each node's set of temporal keys is not updated. Thus we are concerned with the compromise of temporal keys; an adversary's knowledge of underlying keys is irrelevant to the analysis.

Cichoń et al. [6] study the resilience of their BEKPS but they make several simplifying assumptions. Here we give a much more general analysis and we derive general formulae for resilience. In [6, Theorem 2], it is assumed that two nodes A and B have exactly c temporal keys in common. In view of the estimates provided in the last section, this is roughly the expected number of common temporal keys when $k = \sqrt{n}$. Under this assumption, [6, Theorem 2] estimates the probability that a random node C contains these c common temporal keys to be $(kc/n)^c$. We calculate the resilience when $k \neq \sqrt{n}$.

3.3.1 Temporal key sets

As before, suppose that $G_1, \dots, G_{n/c}$ partition an n-set $X = \{1, \dots, n\}$ into $m = n/c$ disjoint c-sets. Suppose A is a random block (i.e., a k-subset of X) and define

$$\mathcal{I}(A) = \{i : A \cap G_i \neq \emptyset\}.$$

$\mathcal{I}(A)$ is the set of indices of the temporal keys held by A. Then let

$$\kappa_A = |\mathcal{I}(A)|;$$

κ_A is the number of temporal keys held by A.

Fix any i-subset $I \subseteq \{1, \dots, m\}$. Define

$$M(i) = |\{A : \mathcal{I}(A) = I\}|.$$

Note that $M(i)$ counts the number of possible nodes whose set of temporal keys is equal to I. The value $M(i)$ does not depend on the particular i-subset I that was chosen.

It is easy to see that

$$|\{A : \mathcal{I}(A) \subseteq I\}| = \binom{ic}{k}. \tag{5}$$

We can derive a formula for $M(i)$ from (5) by applying the principle of inclusion-exclusion.

Lemma 3.1. *For $i \geq 1$, we have*

$$M(i) = \sum_{j=0}^{i-1} (-1)^j \binom{(i-j)c}{k} \binom{i}{j}. \tag{6}$$

Next, define

$$N(i) = |\{A : \kappa_A = i\}|.$$

$N(i)$ is the number of possible nodes holding exactly i temporal keys. The following is an immediate consequence of (6).

Lemma 3.2. *For $i \geq 1$, we have*

$$N(i) = \binom{m}{i} M(i) = \sum_{j=0}^{i-1} (-1)^j \binom{m}{i} \binom{(i-j)c}{k} \binom{i}{j}. \tag{7}$$

3.3.2 Intersection of two blocks

Next, we consider intersections of two blocks. For $\omega \geq 1$, define a *ω-link* to be an ordered pair of two nodes that contain exactly ω common temporal keys. Let $P(\omega)$ denote the number of possible ω-links; then

$$P(\omega) = |\{(A, B) : |\mathcal{I}(A) \cap \mathcal{I}(B)| = \omega\}|.$$

We have the following formula for $P(\omega)$:

Lemma 3.3. *For $\omega \geq 1$, we have*

$$P(\omega) = \sum_{i=\omega}^{k} \sum_{j=\omega}^{k} \binom{m-i}{j-\omega} \binom{i}{\omega} N(i) M(j). \tag{8}$$

For $\omega = 0$, we have

$$P(0) = \sum_{i=1}^{k} \sum_{j=1}^{k} \binom{m-i}{j-\omega} N(i) M(j). \tag{9}$$

Proof. Let $i = \kappa_A$ and $j = \kappa_B$. We can choose A in $N(i)$ ways. For each choice of A, choose ω indices in $\mathcal{I}(A)$ and choose $j - \omega$ indices in $\{1, \ldots, m\} \backslash \mathcal{I}(A)$. Let the set of the j chosen indices be denoted by J. Then choose B such that $\mathcal{I}(B) = J$; there are $M(j)$ ways to do this. $\square$

Remark 3.1. *We can "numerically" verify the formulae (8) and (9) by checking that the following equations hold for various values of n, c and k:*

$$\sum_{\omega=0}^{k} P(\omega) = \binom{n}{k}^2$$

and

$$\frac{\sum_{\omega=1}^{k} \omega P(\omega)}{\binom{n}{k}^2} = \frac{n}{c} \left(1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right)^2.$$

3.3.3 Compromised links and resilience

We can now find expressions for the number of nodes which will compromise a given link, and derive the formula for fail_1. Suppose that (A, B) is a ω-link. Then define

$$S(\omega) = |\{C : \mathcal{I}(A) \cap \mathcal{I}(B) \subseteq \mathcal{I}(C)\}|.$$

$S(\omega)$ denotes the number of possible nodes that will compromise the ω-link (A, B), and it does not depend on the particular choices of A and B.

Lemma 3.4. *For any $\omega > 0$, we have*

$$S(\omega) = \sum_{i=\omega}^{k} \binom{m-\omega}{i-\omega} M(i). \tag{10}$$

Proof. Let $i = \kappa_C$. Choose $i - \omega$ indices in

$$\{1, \ldots, m\} \backslash (\mathcal{I}(A) \cap \mathcal{I}(B)).$$

Let J denote the i-set consisting of the $i - \omega$ chosen indices along with $\mathcal{I}(A) \cap \mathcal{I}(B)$. Then choose C such that $\mathcal{I}(C) = J$; there are $M(i)$ ways to do this. $\square$

Finally, define

$$T(\omega) = |\{(A, B, C) : |\mathcal{I}(A) \cap \mathcal{I}(B)| = \omega \text{ and } \mathcal{I}(A) \cap \mathcal{I}(B) \subseteq \mathcal{I}(C)\}|.$$

$T(\omega)$ counts triples (A, B, C) where (A, B) is a ω-link compromised by C. It is clear, applying (10), that the following formula holds.

Lemma 3.5. *For any $\omega > 0$, we have*

$$T(\omega) = P(\omega)S(\omega) = \sum_{i=\omega}^{k} \binom{m-\omega}{i-\omega} M(i)P(\omega).$$

Now we are in a position to compute some resilience parameters. Recall that the failure probability fail_1 denotes the probability that a random link (A, B) is compromised by a random node C.

Theorem 3.6. *The failure probability is given by*

$$\mathsf{fail}_1 = \frac{\sum_{\omega=1}^{k} T(\omega)}{\sum_{\omega=1}^{k} P(\omega)\binom{n}{k}}. \tag{11}$$

Proof. The total number of possible ω-links with $\omega \geq 1$ is

$$\sum_{\omega=1}^{k} P(\omega),$$

so the total number of triples (A, B, C) where (A, B) is a link is

$$\sum_{\omega=1}^{k} P(\omega)\binom{n}{k}.$$

The total number of triples (A, B, C) where (A, B) is a link and C compromises this link is

$$\sum_{\omega=1}^{k} T(\omega).$$

The resilience is just the quotient of these two quantities. □

Define $\mathsf{fail}_1(\omega)$ to denote the probability that a random ω-link (A, B) is compromised by a random node C. We have the following obvious result.

Lemma 3.7. *For any $\omega \geq 1$, we have*

$$\mathsf{fail}_1(\omega) = \frac{S(\omega)}{\binom{n}{k}}. \tag{12}$$

Lemma 3.7 provides another way to derive the formula (11) for fail_1. Let λ_ω denote the probability that a random link is a ω-link. It is clear that

$$\lambda_\omega = \frac{P(\omega)}{\sum_{i=1}^{k} P(i)} \tag{13}$$

and

$$\mathsf{fail}_1 = \sum_{\omega=1}^{k} \lambda_\omega \mathsf{fail}_1(\omega). \tag{14}$$

Then, from (12), (13) and (14), we have

$$\begin{aligned}
\mathsf{fail}_1 &= \sum_{\omega=1}^{k} \lambda_\omega \mathsf{fail}_1(\omega) \\
&= \sum_{\omega=1}^{k} \frac{P(\omega)S(\omega)}{\sum_{i=1}^{k} P(i)\binom{n}{k}} \\
&= \frac{\sum_{\omega=1}^{k} T(\omega)}{\sum_{\omega=1}^{k} P(\omega)\binom{n}{k}},
\end{aligned}$$

agreeing with (11).

3.4 Numerical examples

We now provide some numerical examples of our formulae. First, we give an example to illustrate the computation of resilience parameters.

Example 3.1. Suppose $n = 1000$, $c = 4$ and $k = 31$. Then the expected number of temporal keys shared by a pair of nodes, given by (3), is $\omega = 3.511857771$, which is a bit less than $\omega = 4$. [6, Theorem 2] estimates $\mathsf{fail}_1(4)$ by computing the quantity

$$\left(\frac{kc}{n}\right)^c = 0.0002364213760.$$

A more accurate estimate for $\mathsf{fail}_1(4)$ based on the analysis in [6], would be

$$\frac{\binom{m-c}{k-c}}{\binom{m}{k}} = 0.0001980391200.$$

However, from (12), the exact value of $\mathsf{fail}_1(4) = 0.0001651542962$.

The overall resilience of the scheme determined from (11) is $\mathsf{fail}_1 = 0.01330121549$. This is quite a bit higher than $\mathsf{fail}_1(4)$, primarily because links consisting of fewer than four temporal keys (which occur frequently) are compromised with higher probability. This can be seen in the following tabulation of values λ_ω and $\mathsf{fail}_1(\omega)$:

ω	λ_ω	$\mathsf{fail}_1(\omega)$
1	0.08756777557	0.1185218591
2	0.1843995070	0.01364696407
3	0.2407996311	0.001524883082
4	0.2188569817	0.0001651542962
5	0.1472998707	0.00001731603382
6	0.07626527018	0.000001755184555

Our next example considers the effect of varying the parameter k.

Example 3.2. Suppose $n = 1000$ and $c = 4$. We compute the values of fail_1 for various choices of k:

k	fail_1
5	0.01925413575
10	0.03349126556
15	0.03904935504
20	0.03548705708
25	0.02588255435
30	0.01518790238
35	0.007187785428
40	0.002776219702
45	0.0008938567010
50	0.0002464139425

It is interesting to observe that fail_1 at first increases, and then decreases, as k increases. The higher values of fail_1 for small values of k reflect the fact that the network has fewer links and the links that do exist are more easily compromised.

Our next example considers the effect of varying the parameter c.

Example 3.3. Suppose $n = 5040$ and $k = 50$. We compute the values of fail_1 for various choices of c:

c	fail_1
2	0.01182106347
3	0.01334509061
4	0.01321072373
5	0.01211454057
6	0.01055743581
7	0.008871668296
8	0.007256884957
9	0.005816967246
10	0.004592239563

The interesting thing to note here is that fail_1 decreases as c increases beyond 3, but the decrease is gradual and not very dramatic.

3.5 Intersection thresholds

We discussed the idea of an intersection threshold in Example 2.2. Basically, as η increases, resilience increases and connectivity decreases. We now develop formulae for these metrics, that depend on the intersection threshold of the scheme.

Recall from Section 2.4 that the connectivity of a scheme is measured by computing the probability Pr_1 that a random pair of nodes is connected. The following result gives a formula for Pr_1.

Theorem 3.8. *For a scheme with intersection threshold η, we have that*

$$\mathsf{Pr}_1 = 1 - \frac{\sum_{i=0}^{\eta-1} P(i)}{\binom{n}{k}^2}. \tag{15}$$

Proof. There are $\binom{n}{k}^2$ possible pairs of nodes, of which $\sum_{i=0}^{\eta-1} P(i)$ are not connected. □

The formula (11) for resilience is generalised as follows.

Theorem 3.9. *For a scheme with intersection threshold η, the failure probability is given by*

$$\mathsf{fail}_1 = \frac{\sum_{\omega=\eta}^{k} T(\omega)}{\sum_{\omega=\eta}^{k} P(\omega)\binom{n}{k}}. \tag{16}$$

Proof. The proof is a straightforward modification of the proof of Theorem 3.6. □

We now revisit Example 3.1.

Example 3.4. Suppose $n = 1000$, $c = 4$ and $k = 31$, as in Example 3.1. We compute the connectivity and failure probabilities for various values of η.

η	Pr_1	fail_1
1	0.9809852766	0.01330121549
2	0.8950825780	0.003202999469
3	0.7141893766	0.0005577036219
4	0.4779684839	0.00007970558807
5	0.2632730072	0.00001002335465

The use of an intersection threshold allows a suitable tradeoff between connectivity and resilience. Observe that resilience increases substantially as η increases; however, connectivity decreases at the same time. For $\eta > 5$, the connectivity is too low to be practical. In this example, $\eta = 2$ or 3 provides a good way to "balance" connectivity and resilience.

4 Revocation

In this section we consider how to design a BEKPS where the highest priority is to be able to revoke nodes efficiently. Revocation of compromised nodes can be achieved in any BEKPS simply by avoiding using their underlying keys to encrypt new temporal keys. However, this can have the undesired effect of reducing the connectivity amongst uncompromised nodes, because they will receive fewer temporal keys if their underlying keys become disused. In general, it is possible to recover the level of connectivity Pr_1 after revocation by selecting future temporal keys from a smaller pool, so that each temporal key will be known to a higher proportion of the nodes. However, this lowers the resilience. We therefore design a BEKPS which enables the revocation of compromised nodes whilst retaining the connectivity and resilience in the remaining network, and keeping key storage and broadcast load low.

Clearly, the most precise way to be able to revoke individual nodes without causing any damage to the rest of the network is to assign a unique underlying key to each node of the network. If each node is given a single, unique underlying key, then this also has the benefit of achieving minimum key storage per node. However, an update requires a broadcast of $(v - r)\kappa$ temporal keys when r nodes have been revoked, which is infeasibly large for many applications.

If it is not the case that each node stores a unique underlying key, then revocation cannot be precise: uncompromised nodes will also be increasingly affected as the number of revocations increases. For example, if each node stores k underlying keys, then when a single node is revoked, k underlying keys are taken out of use by the base station. We denote this as $R(1) = k$ and derive a general formula for the number of redundant underlying keys after i revocations, when underlying keys are distributed using the random KPS of Example 2.1.

Lemma 4.1. *Suppose that each node stores k underlying keys, selected randomly from a key pool of n underlying keys. Let $R(i)$ denote the expected number of underlying keys removed from use when i nodes have been revoked. Then,*

$$R(i) = n\left(1 - \left(1 - \frac{k}{n}\right)^i\right) \tag{17}$$

Proof. Let the i nodes which have been revoked be denoted by $N_1, \ldots, N_i$. For $1 \leq j \leq n$ define a random variable

$$X_j = \begin{cases} 1 & \text{if key } k_j \text{ is known to at least one of } N_1, \ldots, N_i \\ 0 & \text{otherwise} \end{cases}$$

and let $X = \sum_{j=1}^{n} X_j$. We want to find $R(i) = E[X]$.

The expected value of X_1 is $E[X_1] = 1 - \mathsf{Pr}[k_1 \text{ is in none of } N_1, \ldots, N_i]$, so

$$E[X_1] = 1 - \left(1 - \frac{k}{n}\right)^i .$$

Linearity of expectation gives $E[X] = nE[X_1]$, which gives the result. $\square$

This means that an uncompromised node is unintentionally revoked with probability $\frac{\binom{R(i)}{k}}{\binom{n}{k}}$ as it can no longer learn any temporal keys in future broadcasts, and so after $i \geq 1$ revocations the network size $v(i)$ is

$$v(i) = (v - i)\left(1 - \frac{\binom{R(i)}{k}}{\binom{n}{k}}\right)$$

where v is the original network size.

We propose a BEKPS where precise revocation is possible, that is, $v(i) = v - i$, and which provides a choice of trade-offs between key storage and broadcast load which are likely to be suitable for a wide range of network scenarios. To achieve this, we use *LKH schemes* for the underlying layer and random key predistribution for the temporal layer.

4.1 LKH

Logical key hierarchy (LKH) schemes [11, 17, 18] are used in the literature of broadcast encryption for effective revocation. Each of the $v = 2^{d-1}$ nodes is allocated d keys, one of which is unique, and the other keys are known to $2^1, 2^2, \ldots, 2^{d-1}$ nodes respectively. In Figure 1 we demonstrate this on a network of $v = 16 = 2^{5-1}$ nodes. Each node stores five keys: a unique key, a key δ_i shared with another node, a key γ_i shared with 3 other nodes, a key β_i shared with 7 other nodes, and the key α known to all nodes. The key known to all nodes is called the 'root' key.

If a message is to be broadcast to all nodes it can be encypted using the root key. If a set of r nodes is to be revoked, then the message should be broadcast using the smallest set of keys known only to the $v - r$ uncompromised nodes. The size of a broadcast is then logarithmic in the size of the network.

We use LKH for the underlying layer of our BEKPS because it allows fine-grained revocation with low key storage and logarithmic broadcast load. Other revocation schemes may also be

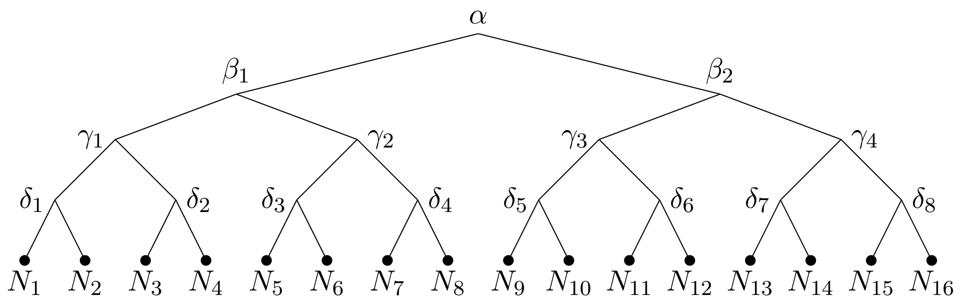

Figure 1: LKH tree on 16 nodes

adapted to form the underlying layer for specific BEKPS scenarios, but the analysis will remain largely similar to that given here. Since the compromise of a node exposes the temporal key(s) known to all other nodes in its LKH tree, we use multiple, smaller trees to balance this resilience risk with the ability to quickly revoke nodes and broadcast new temporal keys.

4.2 BEKPS for revocation

We propose the following BEKPS for scenarios where revocation is a high priority.

4.2.1 Underlying key predistribution

We assume that for a given application, each node can store σ keys in total, where σ is constant. To distribute underlying keys, we partition the v nodes into sets of size $\lambda = 2^{d-1}$, and we do this μ times. For ease of analysis, we will assume that these partitions are not arbitrary but are chosen such that any tree from partition Π_i intersects any tree from partition Π_j in at most one node.

Within each set in each partition, nodes are allocated keys according to an LKH scheme, where the LKH tree has depth d. Thus each node belongs to μ different LKH trees of depth d and therefore must store $k = \mu d$ underlying keys, where μ and d are chosen so that $k < \sigma$. The total number of LKH trees is $L = \frac{\mu v}{\lambda}$.

4.2.2 Temporal key distribution

From our assumption that total key storage is constant, it follows that each node can store at most $\kappa = \sigma - \mu d$ temporal keys. For each of the μ partitions of the nodes, a temporal key pool $\mathcal{K}_{\tau_i}$ of m keys is generated. We require that these key pools are disjoint, that is $\mathcal{K}_{\tau_i} \bigcap \mathcal{K}_{\tau_j} = \emptyset$ for all $1 \leq i < j \leq \mu$, so that we have μ independent Eschenauer Gligor schemes. Relaxing this requirement would allow for improvements in connectivity at the cost of decreased resilience.

A random set of temporal keys is allocated to every tree in the following way. For each partition i and each underlying LKH tree $T_{i,j}$ belonging to that partition (where $1 \leq i \leq \mu$ and $1 \leq j \leq \frac{v}{\lambda}$), a set of $\lfloor \frac{\kappa}{\mu} \rfloor$ temporal keys is chosen at random from the key pool $\mathcal{K}_{\tau_i}$ and encrypted using the underlying root key for $T_{i,j}$.

We assume that nodes require only one temporal key in common in order to establish a link, that is, the intersection threshold is $\eta = 1$. For ease of analysis, we also create the following rules:

- if two nodes are in the same tree T_i in any partition then they will share the set of temporal keys broadcast to T_i. The single temporal key which they use for communication should be randomly selected from this set. Any other keys which they may coincidentally share should be ignored (for example, they may share other keys from other partitions.) Note that there is no ambiguity because it is not possible for two nodes to be in more than one common tree.

- if two nodes are not in the same tree in any partition, they may form a link if they have at least one key in common. They should select just one of these keys at random to secure the link.

If these rules were relaxed and a pair of nodes could use any combination of their shared temporal keys to secure their link, the connectivity and resilience of the network would increase, and so our analysis produces lower bounds.

In summary, if a pair of nodes are in the same tree then they have probability $\mathsf{Pr}_1 = 1$ of being connected. If not, the probability of them being connected is proportional to the Eschenauer Gligor value of Pr_1, as we explain below.

4.2.3 Temporal key update

New temporal keys may be broadcast to the network as often as desired. To revoke a node N_i, the base station should broadcast new temporal keys as normal to all nodes that are not in a common tree to N_i. For any nodes that are in a common tree to N_i, the base station can broadcast new temporal keys encrypted using the smallest set of LKH keys so that all uncompromised nodes receive the new temporal keys but N_i is unable to decrypt any temporal keys from the broadcast.

4.3 Analysis

We now perform analysis of this BEKPS for the cases $\mu = 1$ and $\mu = 2$, and consider the effect of varying d and hence λ, the size of each LKH tree.

4.3.1 LKH trees where $\mu = 1$

We begin by considering $\mu = 1$, that is each node is in exactly 1 tree, and the total number of LKH trees is $L = \frac{v}{\lambda}$.

Lemma 4.2. *When $\mu = 1$, the connectivity is given by*

$$\mathsf{Pr}_1 = \frac{\lambda - 1}{v-1}.1 + \frac{v-\lambda}{v-1}\left(1 - \frac{\binom{m-(\sigma-d)}{\sigma-d}}{\binom{m}{\sigma-d}}\right) \ .$$

Proof. Fix a single node N_i, belonging to a single LKH tree T_j (since $\mu = 1$). We consider the probability of N_i being connected to another of the $v-1$ nodes, which fall into two categories:

- N_i will share temporal keys with the other $\lambda - 1$ nodes of its tree T_j with probability 1
- N_i will share at least one key with the remaining $v - \lambda$ nodes with the Eschenauer Gligor probability given in Example 2.1, and $\kappa = \sigma - d$.

□

We now calculate the resilience metric fail_1. We must consider the probability of a random link being compromised when it is a link between nodes of the same tree, and when it is a link between nodes which are not in a common tree.

Lemma 4.3. *When $\mu = 1$, the resilience is given by*

$$\mathsf{fail}_1 = \frac{f_1\left[\frac{\lambda-2}{v-2}.1 + \frac{v-\lambda}{v-2}\left(\frac{\sigma-d}{m}\right)\right] + f_2\left[\frac{2\lambda-2}{v-2}.1 + \frac{v-2\lambda}{v-2}\left(\frac{\sigma-d}{m}\right)\right]}{f_1+f_2} \quad ,$$

where $f_1 = \binom{\lambda}{2}\frac{v}{\lambda}$ *and* $f_2 = \left(1 - \frac{\binom{m-(\sigma-d)}{\sigma-d}}{\binom{m}{\sigma-d}}\right)\left(\binom{v}{2} - \binom{\lambda}{2}\frac{v}{\lambda}\right)$.

Proof. There are $f_1 = \binom{\lambda}{2}\frac{v}{\lambda}$ pairs of nodes in the network where both nodes are in the same tree. After compromising a single node, the adversary can break a link between one of these pairs with probability

$$\mathsf{fail}_{1,f1} = \frac{\lambda-2}{v-2}.1 + \frac{v-\lambda}{v-2}\left(\frac{\sigma-d}{m}\right) \quad ,$$

since for a given link, there are $\lambda - 2$ nodes which, if compromised, would break that link with certainty by virtue of being in the same tree. A compromise of one of the remaining $v - \lambda$ nodes would reveal the desired key with probability $\frac{\sigma-d}{m}$.

The total number of pairs of nodes which are in different trees is $\binom{v}{2} - \binom{\lambda}{2}\frac{v}{\lambda}$, and each pair is connected with the Eschenauer Gligor probability. Therefore we have that the expected number of links between pairs of nodes from different trees is

$$f_2 = \left(1 - \frac{\binom{m-(\sigma-d)}{\sigma-d}}{\binom{m}{\sigma-d}}\right)\left(\binom{v}{2} - \binom{\lambda}{2}\frac{v}{\lambda}\right)$$

and for these,

$$\mathsf{fail}_{1,f2} = \frac{2\lambda-2}{v-2}.1 + \frac{v-2\lambda}{v-2}\left(\frac{\sigma-d}{m}\right) \quad .$$

This is because, if we fix a link between uncompromised nodes N_i and N_j from different trees, there are $2(\lambda - 1)$ nodes which are in a common tree with either N_i or N_j and therefore know their shared key with certainty. This leaves $v - 2\lambda$ nodes which each have a probability of $\frac{\kappa}{m}$ of knowing their shared key. □

Finally, we consider b_r, the broadcast load required to revoke r nodes. We derive the general formula here for $\mu > 0$.

Lemma 4.4. *For all $\mu > 0$ and $r \geq 1$, the broadcast load to revoke r nodes is given by*

$$b_r \leq (\sigma - \mu d)\left(\frac{v}{\lambda} + rd - 2r\right) \quad .$$

Proof. We begin by calculating b_1, the broadcast load required to revoke a single node N_i. Notice that N_i belongs to μ trees. Calculating b_1 requires the number of Eschenauer Gligor keys per tree $\frac{\sigma}{\mu} - d$, the broadcast to the $L - \mu$ trees of uncompromised nodes, and the LKH revocation for the μ trees containing the compromised node, which requires $d - 1$ broadcasts per tree. Thus if the number of nodes per tree is greater than 1,

$$b_1 = \left(\frac{\sigma}{\mu} - d\right)(L - \mu + \mu(d-1)) = (\sigma - \mu d)\left(\frac{v}{\lambda} + d - 2\right) \quad .$$

If the number of nodes per tree is 1 (ie. $d = 1$) then we assume that $\mu = 1$ so that the number of trees L equals the number of nodes v. Then $b_1 = (\sigma - 1)(L - 1)$.

The value of b_r for $r > 1$ will depend upon whether any of the r nodes are in the same tree(s). However, we observe that the broadcast load is largest when each of the nodes to be revoked is in a different tree, hence $b_r \leq \left(\frac{\sigma}{\mu} - d\right)(L - r\mu + r\mu(d-1)) = (\sigma - \mu d)\left(\frac{v}{\lambda} + rd - 2r\right)$. □

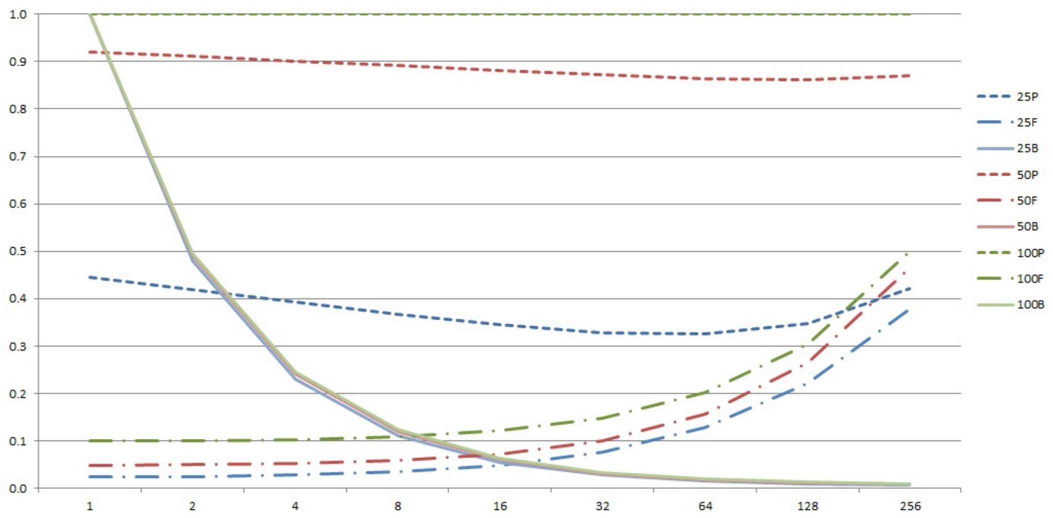

Figure 2: Plot of the values of Pr_1, fail_1 and b'_1 when $\mu = 1$ and there are $1, 2, 2^2, \ldots, 2^8$ nodes per tree for key storage $\sigma = 25, 50$ and 100 respectively

We demonstrate these formulae in Figure 2. For comparison, we consider a fixed network size of $v = 1024$ nodes, and temporal key pool size of $m = 1000$ keys. We consider nodes which can store 25, 50 or 100 keys respectively, and plot the corresponding values of Pr_1 and fail_1 for underlying LKH layers of $1, 2, 4, \ldots, 256$ nodes per tree. In order to plot the broadcast load on the same axes, we plot b_1 as a fraction of the number of keys to be broadcast when there is only one node per tree ($d = 1$), ie. we plot $b'_1 = \frac{b_1}{(\sigma-\mu)\left(\frac{v}{\lambda}-1\right)}$.

We see that if nodes can store 100 keys then $\mathsf{Pr}_1 \approx 1$. If nodes can store only 50 or 25 keys then the connectivity decreases significantly, but this has the advantage of lowering fail_1. Finally, we note that the broadcast load b'_1 decreases exponentially as the number of nodes per tree increases. That is, for fixed network size v and key storage σ, the broadcast load can be decreased by increasing the number of nodes per tree. The plots show that b'_1 is almost identical across different values of σ, however, as we know from the formula, the actual broadcast size b_1 does of course increase with σ. For example, when there are 8 nodes per tree, the broadcast to revoke one node is $b_1 = 2730$ when $\sigma = 25$, $b_1 = 5980$ when $\sigma = 50$ and $b_1 = 12480$ when $\sigma = 100$.

4.3.2 LKH trees where $\mu = 2$

We now consider the case where $\mu = 2$, that is, each node is a member of two trees, one from each partition. Each node therefore stores $2d$ underlying LKH keys, leaving space for it to store $\sigma - 2d$ temporal keys. The base station may broadcast a set of at most $\lfloor\frac{\sigma}{2}\rfloor - d$ temporal keys to each tree. Indeed, in general the base station may broadcast at most $\lfloor\frac{\sigma}{\mu}\rfloor - d$ to the root of each tree. For ease of notation we will omit the floor symbols.

Lemma 4.5. *When $\mu = 2$, the connectivity is given by*

$$\mathsf{Pr}_1 = \frac{2(\lambda-1)}{v-1}.1 + \frac{v-1-2(\lambda-1)}{v-1}\left(1 - \left[\frac{\binom{m-(\frac{\sigma}{2}-d)}{\frac{\sigma}{2}-d}}{\binom{m}{\frac{\sigma}{2}-d}}\right]^2\right).$$

The proof follows in the same way as that of Lemma 4.2. The Eschenauer Gligor probability contains a squared term because the probability of two nodes from different trees *not* being connected is the probability of them not sharing any keys from partition Π_1 multiplied by the probability of them not sharing any keys from partition Π_2.

Lemma 4.6. *When $\mu = 2$, the resilience is given by*

$$\mathsf{fail}_1 = \frac{f_1\left[\frac{\lambda-2}{v-2}.1 + \frac{v-\lambda}{v-2}\left(\frac{\frac{\sigma}{2}-d}{m}\right)\right] + f_2\left[\frac{2\lambda-2}{v-2}.1 + \frac{v-2\lambda}{v-2}\left(\frac{\frac{\sigma}{2}-d}{m}\right)\right]}{f_1+f_2},$$

where $f_1 = \binom{\lambda}{2}L$ and $f_2 = \left(1 - \left[\frac{\binom{m-(\frac{\sigma}{2}-d)}{\frac{\sigma}{2}-d}}{\binom{m}{\frac{\sigma}{2}-d}}\right]^2\right)\left(\binom{v}{2} - \binom{\lambda}{2}L\right).$

Proof. As in the proof of Lemma 4.3, we calculate fail_1 by considering the two cases:

1. If the link is between two nodes in a common tree in partition P_i, $\lambda - 2$ other nodes from that tree can break the link with probability 1, and $v - \lambda$ nodes can each break the link with probability $\frac{\frac{\sigma}{2}-d}{m}$ using their knowledge of keys from the key pool $\mathcal{K}_{\mathcal{T}_i}$. There are $f_1 = \binom{\lambda}{2}L$ such links.
2. If the link is between two nodes which are not in a common tree, $2\lambda - 2$ other nodes in their respective trees can break the link, and $v - 2\lambda$ other nodes can break the link with probability $\frac{\frac{\sigma}{2}-d}{m}$. The expected number of such links is $f_2 = \left(1 - \left[\frac{\binom{m-(\frac{\sigma}{2}-d)}{\frac{\sigma}{2}-d}}{\binom{m}{\frac{\sigma}{2}-d}}\right]^2\right)\left(\binom{v}{2} - \binom{\lambda}{2}L\right).$

□

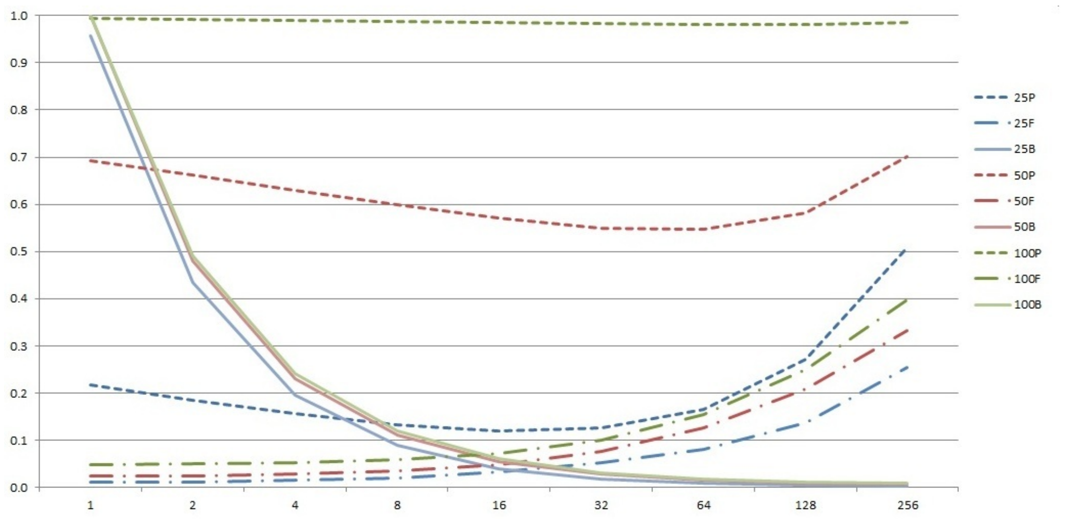

Figure 3: Plot of the values of Pr_1, fail_1 and b'_1 when $\mu = 2$ and there are $1, 2, 2^2, \ldots, 2^8$ nodes per tree for key storage $\sigma = 25, 50$ and 100 respectively

In Figure 3 we demonstrate some numerical values using these formulae. As in Figure 2, we set $v = 1024$ and $m = 1000$ for each of the key pools. Again, we see that Pr_1 is highest when 100 keys are stored per node, at the cost of a slightly increased value of fail_1. Comparing Figures 2 and 3 we find that, when all other variables are fixed, the higher value of μ gives lower values of Pr_1 and

fail_1, with little effect on b_1'. For comparison, we note that when there are 8 nodes per tree, the broadcast to revoke one node is $b_1 = 2080$ when $\sigma = 25$, $b_1 = 5460$ when $\sigma = 50$ and $b_1 = 11960$ when $\sigma = 100$, that is, a little lower than when $\mu = 1$. We therefore suggest that if the lower value of Pr_1 can be tolerated for the network's purposes, then $\mu = 2$ should be chosen to give higher resilience and lower broadcast for revocation.

When $\mu > 2$ the analysis becomes increasingly complex, and it remains an open problem to determine whether there are any advantages to higher values of μ. It seems likely that as μ increases Pr_1 will decrease, because nodes within the same tree will always be connected (unless revoked), but nodes which are not in a common tree can only be connected if they know keys from the same Eschenauer Gligor scheme, of which there are μ different schemes. Since each node can only store a fixed number of keys σ, as μ increases the number of temporal keys per Eschenauer Gligor scheme will decrease, and so Pr_1 will decrease accordingly. By the same argument, it seems likely that fail_1 would also decrease, giving higher resilience against an adversary

We have thus constructed an effective BEKPS protocol which allows efficient revocation and where, given key storage σ, there is some freedom to choose an appropriate trade-off between the parameters b_r, Pr_1 and fail_1, not only by varying the size of the key pool (as with any KPS), but also by varying the size λ of LKH trees in the underlying layer, and the number of trees μ to which each node belongs.

5 Hierarchical temporal key distribution

In Section 1.3 we introduced the idea of using red a BEKPS to create hierarchy in the temporal layer, by broadcasting extra keys to certain nodes. This can provide more efficient routing of information through a network. The flexibility which a BEKPS provides to *change* which nodes have the extra keys reduces both the damage caused by extra battery usage and the risk posed to the resilience of the network. We will refer to nodes which are allocated extra keys as *primary nodes*, whilst the remaining *secondary nodes* have fewer keys.

Regularly changing the set of nodes that are primary will mean that the burdens of being a primary node are spread across the network over time. Random allocation of primary nodes reduces the risk of an adversary launching a targeted attack to reveal a high number of keys through a small number of node compromises.

5.1 BEKPS for hierarchical temporal key distribution

We now consider the question of how to create a BEKPS so that any node can be chosen as a primary node, and so that at any time period between broadcasts there should be p primary nodes and $v - p$ secondary nodes. (Note that the number of primary nodes p may be changed at any broadcast, so that there are p_i primary nodes after update i. However, since each update can be analysed without reference to the number of primary nodes which have gone before, we simply write p in the analysis which follows, for ease of notation.)

5.1.1 Underlying key predistribution

We propose that the best choice of KPS for the underlying keys is again one based on a revocation scheme such as LKH. We justify this with the following observations. Suppose that a node N_i with underlying key set U_i is to be chosen as a primary node. The base station must broadcast a higher proportion of temporal keys to it than to secondary nodes.

1. If at least one of the underlying keys in U_i is known uniquely to node N_i, then the base station can simply use this key to encrypt the extra temporal keys.

2. If none of N_i's underlying keys is known uniquely to N_i, that is, for each $u_j \in U_i$ there exists a node N_k with $u_j \in U_k$, then in broadcasting extra temporal keys, it will happen that some other nodes learn some extra temporal keys too. This will have the effect of creating a multiple-layered hierarchical network, where p nodes are primary nodes but amongst the remaining $v - p$ nodes there is variety in how many temporal keys are received. Whilst this may be desirable for some applications, in others it would cause some unnecessary battery drainage amongst the $v - p$ nodes and complicate routing protocols. We therefore restrict our study to a strictly two-layer hierarchy of primary and secondary nodes.

We conclude that to efficiently create primary and secondary nodes and avoid unnecessarily burdening non-primary nodes, it is desirable that each node stores a unique underlying key. For similar reasons to those given in Section 4.1, we propose an underlying layer based on LKH. As in Section 4, using a single LKH scheme minimises broadcast load but maximises underlying key storage, and so we partition the nodes into several underlying LKH trees, each of size $\lambda = 2^{d-1}$.

5.1.2 Temporal key distribution

A straightforward way to allocate temporal keys in order to create p primary nodes is to use a slight modification of the Eschenauer Gligor KPS [9], where each primary node is allocated κ_1 temporal keys and each secondary node is allocated κ_2 temporal keys from a key pool $\mathcal{K}_\tau$ of m temporal keys. We will demonstrate that this allows the connectivity and resilience parameters to be easily altered with each broadcast, though of course many other KPSs would be suitable. For ease of analysis, we choose an intersection threshold of $\eta = 1$, that is, two nodes may form a link if they have one key in common. We will also assume that if two nodes have more than one key in common then they randomly select one of those keys to secure the link. Relaxing this assumption would increase the resilience of the scheme.

The choice of primary nodes could be made deterministically or randomly, as desired. The benefits of choosing them deterministically are:

- more efficient shared key discovery
- the possibility of node identity authentication
- a node will not be required to be a primary node twice until necessary, ie. when all other nodes have been used as primary nodes at least once.

On the other hand, choosing the primary nodes at random may increase the difficulty for an adversary to target them for compromise. Given the increased risk to the resilience of the network which primary nodes cause, the unpredictability of the choice of primary nodes is an important security consideration. In our analysis we will assume that the adversary compromises nodes at random, and therefore our analysis is applicable to deterministic and random allocations of primary nodes.

The base station may choose how to broadcast the temporal keys to secondary nodes in order to achieve a particular trade-off between connectivity, resilience and broadcast load. We consider this in more detail in Section 5.2.2.

5.2 Analysis

5.2.1 Connectivity

We now derive formulae for the connectivity probabilities in terms of the size m of the temporal key pool and the number of temporal keys assigned to primary and secondary nodes, κ_1 and κ_2 respectively. We use $\mathsf{Pr}_{1,1}$ to denote the probability of two primary nodes being connected, $\mathsf{Pr}_{1,2}$ for the probability of a primary node and secondary node being connected, and finally $\mathsf{Pr}_{2,2}$ for the connectivity probability between a pair of secondary nodes.

Using the Eschenauer Gligor probability of connectivity given in Example 2.1, we have that

$$\mathsf{Pr}_{1,1} = 1 - \frac{\binom{m-\kappa_1}{\kappa_1}}{\binom{m}{\kappa_1}}$$

and

$$\mathsf{Pr}_{1,2} = 1 - \frac{\binom{m-\kappa_1}{\kappa_2}}{\binom{m}{\kappa_2}} \ .$$

Similarly, it can be seen that

$$\mathsf{Pr}_{2,2} \geq 1 - \frac{\binom{m-\kappa_2}{\kappa_2}}{\binom{m}{\kappa_2}} \ .$$

when we consider that $1 - \binom{m-\kappa_2}{\kappa_2}/\binom{m}{\kappa_2}$ is the probability that two secondary nodes with different temporal key sets are connected. Two secondary nodes which are given the same set of temporal keys because they were encrypted with a shared LKH key will certainly be connected, and this is why a lower bound for $\mathsf{Pr}_{2,2}$ is given. The exact value of $\mathsf{Pr}_{2,2}$ will depend on choices which the base station makes regarding how to use the LKH tree(s) to distribute the temporal keys, as we describe in Section 5.2.2. In Section 5.2.3 we derive an estimate for $\mathsf{Pr}_{2,2}$ using an assumption about the temporal key distribution.

m	κ_1	κ_2	$\mathsf{Pr}_{1,1}$	$\mathsf{Pr}_{1,2}$	$\mathsf{Pr}_{2,2}$
500	50	10	0.9962	0.6548	$\geq$ 0.1844
500	50	15	0.9962	0.7990	$\geq$ 0.3709
1000	85	15	0.9996	0.7388	$\geq$ 0.2041
1000	85	25	0.9996	0.8945	$\geq$ 0.4731
1000	60	30	0.9783	0.8481	$\geq$ 0.6045
5000	100	50	0.8701	0.6377	$\geq$ 0.3965

Table 1: Examples of connectivity parameters (to four decimal places) for different key pool sizes n and sizes of temporal key pool for primary and secondary nodes, κ_1 and κ_2 respectively.

Thus the base station can choose the parameters m, κ_1 and κ_2 to achieve different levels of the connectivity probabilities. Some example values are given in Table 1. Observe that connectivity between secondary nodes may not be necessary or even desirable; for example, to conserve resources whilst maintaining a connected network, it may be preferable to have a very low value of $\mathsf{Pr}_{2,2}$ as long as $\mathsf{Pr}_{1,2}$ is high enough to ensure that almost every secondary node is connected to at least one primary node, and $\mathsf{Pr}_{1,1}$ is high enough to ensure that almost every primary node is connected to all other primary nodes. Finally, we note that m, κ_1 and κ_2 are independent of the network size v, and can be changed at each broadcast if desired.

As with any random KPS, higher connectivity in this BEKPS results in lower resilience. In particular, the compromise of a primary node will reveal κ_1 of the total m keys. This risk will be reduced by dynamically changing the choice of primary nodes to lower the risk of their compromise, and by choosing $\mathsf{Pr}_{2,2}$, $\mathsf{Pr}_{1,2}$ and $\mathsf{Pr}_{1,1}$ to be as small as possible whilst retaining functional connectivity across the network. We calculate fail_1 in Section 5.2.3 after considering the different options available for the base station for the broadcast.

5.2.2 Broadcast load

The following example considers how the temporal keys could be distributed to the secondary nodes.

Example 5.1. Suppose we have a network of $v = 16 = 2^{5-1}$ nodes arranged in an LKH tree so that each node has to store $d = 5$ keys, as illustrated in Figure 4.

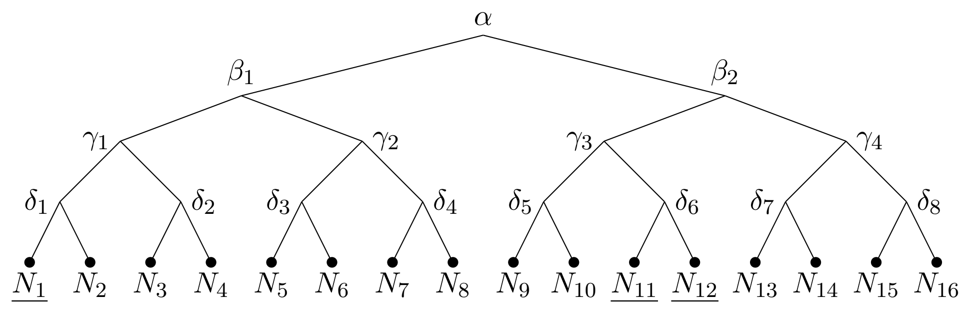
α
β_1
β_2
γ_1
γ_2
γ_3
γ_4
δ_1
δ_2
δ_3
δ_4
δ_5
δ_6
δ_7
δ_8
$\underline{N_1}$ N_2 N_3 N_4 N_5 N_6 N_7 N_8 N_9 N_{10} $\underline{N_{11}}$ $\underline{N_{12}}$ N_{13} N_{14} N_{15} N_{16}

Figure 4: LKH tree on 16 nodes

Suppose that we wish to create $p = 3$ primary nodes, and at random we pick these to be nodes N_1, N_{11} and N_{12} (underlined in Figure 4). The base station would broadcast κ_1 temporal keys to each of these primary nodes, using their unique underlying keys. For the secondary nodes, there is a choice to be made about the temporal key broadcast.

1. The key centre could broadcast a separate temporal key set to each of the secondary nodes using their underlying keys. This creates the maximum broadcast load, the highest resilience, and $\mathsf{Pr}_{2,2}$ achieves its lower bound:

$$\mathsf{Pr}_{2,2} = 1 - \frac{\binom{m-\kappa_2}{\kappa_2}}{\binom{m}{\kappa_2}} \ .$$

2. The key centre could minimise the broadcast by using the smallest set of LKH keys not known to the primary nodes, that is, by using the LKH keys associated with the minimal covering set of the secondary nodes. In this example, temporal keys would be broadcast to N_5, N_6, N_7, and N_8 encrypted by their shared key γ_2; to nodes N_3 and N_4 using δ_2; and to N_2 using its unique underlying key. Similarly, the broadcast to nodes N_{13}, N_{14}, N_{15} and N_{16} would be encrypted by their shared key γ_4, and nodes N_9 and N_{10} would be broadcast temporal keys encrypted by underlying key δ_5. The number of temporal key sets to be broadcast is then reduced from 16 to 8. The probability that a pair of secondary nodes have at least one common key is then

$$\mathsf{Pr}_{2,2} = \frac{2\binom{4}{2} + 2\binom{2}{2} + (1 \times 2 \times 4 \times 2 \times 4)\left(1 - \frac{\binom{m-\kappa_2}{\kappa_2}}{\binom{m}{\kappa_2}}\right)}{\binom{13}{2}} \ ,$$

which is greater than if unique underlying keys were used, at the cost of reduced resilience.

3. In order to find a trade-off between the above options, the key centre could choose not to use the smallest set of LKH keys unknown to the primary nodes, for example by using the six δ_i keys with $i \in \{1,2,\ldots,8\} \setminus \{1,6\}$, plus the unique key known to N_2. Then

$$\mathsf{Pr}_{2,2} = \frac{6\binom{2}{2} + (1 \times 12 + 2 \times (10+8+6+4+2))\left(1 - \frac{\binom{m-\kappa_2}{\kappa_2}}{\binom{m}{\kappa_2}}\right)}{\binom{13}{2}} \quad .$$

This example illustrates that there are choices to be made about the broadcast within each LKH tree, as well as about the number of underlying LKH trees $\frac{v}{\lambda}$. In our analysis we will assume that, on average, each set of temporal keys is broadcast to x secondary nodes, where $x < \lambda$ and $x \to \lambda$ as $p \to 0$ if the base station is using the minimum broadcast load. Then to broadcast a set of κ_1 temporal keys to each primary node and κ_2-sets of keys to each secondary node requires a broadcast of size

$$b \approx \kappa_1 p + \kappa_2 \frac{(v-p)}{x} \quad .$$

Using our assumption that each set of temporal keys is broadcast to x secondary nodes, we can revisit the expression we derived for $\mathsf{Pr}_{2,2}$ and use weighted probability to derive the estimate

$$\mathsf{Pr}_{2,2} \approx \frac{x-1}{v-p-1} + \frac{v-p-x}{v-p-1}\left(1 - \frac{\binom{m-\kappa_2}{\kappa_2}}{\binom{m}{\kappa_2}}\right) \quad .$$

5.2.3 Resilience

We can make an estimate of fail_1 using Equation 2 from Section 2 with a weighted probability for primary and secondary nodes: the expected number of keys known to an adversary after the compromise of one node is $\kappa_1 \frac{p}{v} + \kappa_2 \frac{(v-p)}{v}$, and so we have that

$$\mathsf{fail}_{1,est} = \frac{\kappa_1 p + \kappa_2 (v-p)}{vm} \quad ,$$

since there is exactly one key securing each link. However, this method does not take into account the proportions of the three different types of links.

We now extend the definition of fail_1 to the hierarchical network setting. We retain our assumption that the adversary compromises all nodes with equal probability, and give each type of link in the network the same weight. In Table 2 we see comparisons between the approximation $\mathsf{fail}_{1,est}$ and our more detailed calculation of fail_1.

Lemma 5.1. *The resilience is given by*

$$\mathsf{fail}_1 = \frac{1}{T}\left(\binom{p}{2}\mathsf{Pr}_{1,1}\mathsf{fail}_{1,1} + p(v-p)\mathsf{Pr}_{1,2}\mathsf{fail}_{1,2} + \binom{v-p}{2}\mathsf{Pr}_{2,2}\left[\frac{v-p-x}{v-p-1}\mathsf{fail}_{2,2,a} + \frac{x-1}{v-p-1}\mathsf{fail}_{2,2,b}\right]\right) \quad ,$$

where $\mathsf{Pr}_{1,1}, \mathsf{Pr}_{1,2}$ *and* $\mathsf{Pr}_{2,2}$ *are as given above,*

$$\mathsf{fail}_{1,1} = \frac{\kappa_1(p-2) + \kappa_2(v-p)}{m(v-2)}, \quad \mathsf{fail}_{1,2} \approx \frac{1}{v-2}\left(\frac{\kappa_1(p-1) + \kappa_2(v-p-x)}{m} + x - 1\right),$$

$$\mathsf{fail}_{2,2,a} \approx \frac{1}{v-2}\left(\frac{\kappa_1 p + \kappa_2(v-p-2x)}{m} + 2(x-1)\right), \quad \mathsf{fail}_{2,2,b} \approx \frac{1}{v-2}\left(\frac{\kappa_1 p + \kappa_2(v-p-x)}{m} + x - 2\right),$$

and

$$T = \binom{p}{2}\mathsf{Pr}_{1,1} + p(v-p)\mathsf{Pr}_{1,2} + \binom{v-p}{2}\mathsf{Pr}_{2,2} \ .$$

Proof. We begin by finding the total number of links in the network, before any compromise, which is

$$T = \binom{p}{2}\mathsf{Pr}_{1,1} + p(v-p)\mathsf{Pr}_{1,2} + \binom{v-p}{2}\mathsf{Pr}_{2,2} \ .$$

Now we consider each type of link and its resilience.

- **Primary-primary links**

  There are $\binom{p}{2}\mathsf{Pr}_{1,1}$ primary node to primary node links. Fix such a link between some primary nodes N_i and N_j, and consider the advantage to an adversary of compromising a single node $N_k \notin \{N_i, N_j\}$. If N_k is a primary node, the adversary will learn κ_1 keys; if N_k is secondary it will reveal κ_2 keys. Thus the adversary breaks the link with probability

$$\begin{aligned}\mathsf{fail}_{1,1} &= \frac{1}{m}\left(\kappa_1\frac{p-2}{v-2} + \kappa_2\frac{v-p}{v-2}\right)\\ &= \frac{\kappa_1(p-2)+\kappa_2(v-p)}{m(v-2)} \ .\end{aligned}$$

- **Primary-secondary links**

  The number of primary node to secondary node links is $p(v-p)\mathsf{Pr}_{1,2}$. Fix such a link between primary node N_i and secondary node N_j. Suppose that the base station is using less than the maximum broadcast load. Then the adversary can certainly break the link if it compromises a secondary node which is ‘near’ to N_j in the LKH tree, such that it stores the same set of temporal keys as N_j. That is, if we assume that on average, each set of temporal keys is broadcast to x secondary nodes, then an adversary who compromised a secondary node N_j will certainly be able to break the $p(x-1)$ links between primary nodes and the $x-1$ secondary nodes with which N_j shares the underlying LKH key used for the broadcast. Therefore, we have that a primary-secondary node link is broken with probability

$$\mathsf{fail}_{1,2} \approx \frac{1}{v-2}\left(\frac{\kappa_1(p-1)+\kappa_2(v-p-x)}{m} + x - 1\right),$$

  where the approximation comes from x being an average value.

- **Secondary-secondary links**

  There are $\binom{v-p}{2}\mathsf{Pr}_{2,2}$ secondary node to secondary node links. Fix such a link between secondary nodes N_i and N_j. As with primary-secondary links, the adversary can break the link with certainty if the broadcast load is less than the maximum and the adverary compromises a secondary node N_k which has received the same temporal key set as one (or both) of N_i and N_j. Suppose that N_i and N_j have different temporal key sets $\mathcal{K}_{N_i}$ and $\mathcal{K}_{N_j}$. Then the probability of the link being broken after the compromise of a single node is

$$\mathsf{fail}_{2,2,a} \approx \frac{1}{v-2}\left(\frac{\kappa_1 p+\kappa_2(v-p-2x)}{m} + 2(x-1)\right) \ ,$$

  and finally, if $\mathcal{K}_{N_i} = \mathcal{K}_{N_j}$, then the probability of breaking the link is

$$\mathsf{fail}_{2,2,b} \approx \frac{1}{v-2}\left(\frac{\kappa_1 p+\kappa_2(v-p-x)}{m} + x - 2\right) \ .$$

Combining these results gives the stated formula. □

We illustrate some example values of fail_1 in Table 2.

x	p	m	κ_1	κ_2	$\mathsf{Pr}_{1,1}$	$\mathsf{Pr}_{1,2}$	$\mathsf{Pr}_{2,2}$	fail_1	b
2^2	50	500	50	10	0.9962	0.6548	0.1870	0.0290	4875
2^3	50	500	50	10	0.9962	0.6548	0.1905	0.0357	3687.5
2^4	50	500	50	10	0.9962	0.6548	0.1973	0.0492	3093.75
2^3	100	500	50	10	0.9962	0.6548	0.1908	0.0383	6125
2^3	250	500	50	10	0.9962	0.6548	0.1921	0.0476	13437.5
2^3	50	1000	50	10	0.9280	0.4027	0.1027	0.0236	3687.5
2^3	50	500	80	10	0.9999	0.8281	0.1905	0.0384	5187.5
2^3	50	500	50	20	0.9962	0.8835	0.5683	0.0554	4875
2^3	50	500	50	30	0.9962	0.9617	0.8536	0.0744	6062.5

Table 2: Examples of connectivity and resilience metrics (to four decimal places) and broadcast load for fixed network size $v = 1000$ and varying: the average number of secondary nodes to which a single temporal key set is sent, x; the number of primary nodes p; the number of keys in the key pool n; and the number of keys given to primary and secondary nodes, κ_1 and κ_2 respectively.

We observe that

- increasing x reduces the broadcast load and creates a marginal increase in $\mathsf{Pr}_{2,2}$, leaving the other connectivities unchanged. However, it noticeably increases fail_1, that is, it substantially reduces the resilience.
- for most applications the number of primary nodes need not be large; $\mathsf{Pr}_{1,1}$ and $\mathsf{Pr}_{1,2}$ can be set to be high independently of p, whilst increasing p reduces the resilience and significantly increases the broadcast load.
- as we would expect, increasing m lowers the connectivity probabilities and fail_1, increasing the resilience. The broadcast load is unaffected.
- increasing κ_2 substantially increases the connectivity $\mathsf{Pr}_{2,2}$, whilst increasing the broadcast load and reducing the resilience to a lesser extent. It may seem, therefore, that a comparatively high value of κ_2 will be desirable for most network applications. However, secondary node to secondary node communication may be unnecessary as long as $\mathsf{Pr}_{1,2}$ is high enough to ensure that most secondary nodes are connected to at least one primary node. It may therefore be desirable to keep κ_2 very low in order to increase resilience, reduce broadcast load and conserve battery power in anticipation of secondary nodes becoming primary nodes in the future.

6 Concluding remarks

We have introduced the term *broadcast-enhanced key predistribution schemes* (BEKPS) to describe schemes which combine key predistribution with a trusted base station and broadcast channel, and discussed some of the many motivations for using BEKPSs. We developed a framework for the design and analysis of BEKPSs, and demonstrated its use throughout our paper. In Section 3 we provided simpler proofs for some of the results given by Cichoń et al. in [6] for their scheme, which we classify as a BEKPS. We derived more general formulae to calculate the resilience and explained how intersection thresholds can be used to increase resilience at the cost of decreasing connectivity.

In Sections 4 and 5 we proposed appropriate BEKPS protocols for specific applications. In Section 4, we demonstrated a practical BEKPS where revocation can be performed without any uncompromised nodes losing keys. We showed that for a given key storage parameter σ, suitable trade-offs can be found between the connectivity, resilience and broadcast load by varying the size of the temporal key pool, and the number and size of LKH trees used to distribute underlying keys. In Section 5 we demonstrated a BEKPS which creates a network with two-layer hierarchy. This brings the benefit of more efficient data routing. The ability to dynamically change the connectivity probabilities and the allocation of primary nodes reduces the risks of battery drainage and lowered resilience from which other hierarchical networks suffer.

For future work, there are many variations of BEKPSs which can be studied. We note the following open questions:

- Are there BEKPS scenarios where an underlying key predistribution based on a revocation scheme is not the best choice?
- Can other revocation schemes provide advantages over LKH in the underlying key predistribution of a BEKPS?
- Are there advantages to assigning temporal keys deterministically (other than aiding shared key discovery)?
- How can a BEKPS design be adapted to be more efficient if the locations of nodes are known to the base station?

References

[1] Shimshon Berkovits. How to broadcast a secret. In *Proceedings of EUROCRYPT '91, Lecture Notes in Computer Science*, pages 535–541, 1991.

[2] Simon R. Blackburn, Keith M. Martin, Maura B. Paterson and Douglas R. Stinson. Key refreshing in wireless sensor networks. In *ICITS 2008 Proceedings, Lecture Notes in Computer Science* pages 156–170, 2008.

[3] Seyit Ahmet Camtepe and Bulent Yener. Key distribution mechanisms for wireless sensor networks: a survey. *Rensselaer Polytechnic Institute, Computer Science Department, Tech. Rep. TR-05-07*, 2005.

[4] Seyit Ahmet Camtepe, Bulent Yener, and Moti Yung. Expander graph based key distribution mechanisms in wireless sensor networks. In *ICC 06, IEEE International Conference on Communications*, pages 2262–2267, 2006.

[5] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 197–213, Washington, DC, USA, 2003. IEEE Computer Society.

[6] Jacek Cichoń, Zbigniew Gołębiewski and Mirosław Kutyłowski. From key predistribution to key redistribution. In *(ed. Christian Scheideler) ALGOSENSORS'10, Proceedings of the 6th international conference on Algorithms for sensor systems, wireless adhoc networks, and autonomous mobile entities*, pages 92-104, 2010. Springer-Verlag.

[7] Wenliang Du, Jing Deng, Yunghsiang S. Han, Pramod K. Varshney, Jonathan Katz and Aram Khalili. A pairwise key predistribution scheme for wireless sensor networks. ACM Transactions on Information System Security, pages 228–258, 2005.

[8] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

[9] Laurent Eschenauer and Virgil D Gligor. A key-management scheme for distributed sensor networks. In *CCS '02: Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 41–47, New York, NY, USA, 2002. ACM.

[10] Amos Fiat and Moni Naor. Broadcast encryption. In *Proceedings of CRYPTO '93, Lecture Notes in Computer Science*, pages 480–491, 1994.

[11] Hugh Harney and Eric Harder. Logical key hierarchy protocol. *Internet Draft, Internet Engineering Task Force*, March 1999.

[12] Kejie Lu, Yi Qian and Jiankun Hu. A framework for distributed key management schemes in heterogeneous wireless sensor networks. *2006 IEEE International Performance Computing and Communications Conference*, pages 513–520, 2006.

[13] Keith M. Martin. On the applicability of combinatorial designs to key predistribution for wireless sensor networks. *Coding and Cryptology (IWCC2009), Lecture Notes in Computer Science 5557*, pages 124-145, 2009.

[14] Carles Padró, Ignacio Gracia, Sebastià Martín and Paz Morillo. Linear broadcast encryption schemes. *Discrete Applied Mathematics*, pages 223–238, 2003.

[15] Maura B. Paterson and Douglas R. Stinson. A unified approach to combinatorial key predistribution schemes for sensor networks. *Cryptology ePrint Archive*, Report 076, 2011.

[16] Stanislava Soro and Wendi B. Heinzelman. Cluster head election techniques for coverage preservation in wireless sensor networks. *Ad Hoc Networks* pages 955–972, 2009.

[17] Debby M. Wallner, Eric J. Harder and Ryan C. Agee. Key management for multicast: issues and architectures. *Technical report, IETF draft*, July 1997.

[18] Chung Kei Wong, Mohamed Gouda and Simon S. Lam. Secure group communications using key graphs. In *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 68–79, 1998.

[19] Yang Xiao, Venkata Krishna Rayi, Bo Sun, Xiaojiang Du, Fei Hu and Michael Galloway. A survey of key management schemes in wireless sensor networks. In *Computer Communications*, pages 2314–2341, 2007.