Proceedings of the 2004 IEEE
Workshop on Information Assurance and Security
United States Military Academy, West Point, NY, 10–11 June 2004

On the Security of Random Key Pre-distribution Schemes

Mahalingam Ramkumar, Nasir Memon

Abstract— **Key pre-distribution (KPD) schemes, which are inherently trade-offs between security and complexity, are perhaps well suited for securing large-scale deployments of *resource constrained* nodes without persistent access to a trusted authority (TA). However, the need to offset their inherent security limitations, calls for some degree of *tamper-resistance* of nodes. Obviously, if absolute tamper-resistance is guaranteed, KPD schemes are rendered secure. In practice, however, tamper-resistance will have some limitations which will be exploited by attackers. In this paper, we analyze the security of deployments of *random* key pre-distribution schemes based on some assumptions on the "extent of tamper-resistance." We argue that a "limited extent of tamper resistance" when used in conjunction with a mechanism for "periodic key updates," drastically improves the security of (especially random) KPD schemes.**

I. Introduction

In many evolving applications, autonomous, highly resource constrained (probably battery operated wireless communication devices in sensor networks or mobile ad hoc networks) nodes are expected to be deployed in large numbers. For smooth operation of such devices, there is a need for a *low complexity* security infrastructure, which would permit devices (or nodes) to *authenticate* each other, and ensure *confidentiality* of inter-nodal exchanges.

Limited resources may imply that a security infrastructure based on asymmetric key cryptography (or PKI) would not be a suitable option. Lack of a persistent "channel" to a central trusted authority (TA) renders solutions like Kerberos impractical. In such situations, a possible solution is a security infrastructure based on key pre-distribution (KPD). In a KPD scheme, some secrets are *preloaded* in each node by a TA prior to deployment. These keys are then used by the nodes to authenticate each other and communicate securely. KPDs are effectively *trade-offs between security and resource utilization*, and would thus permit even severely resource constrained devices to participate in the deployment. Their limitation in security is the need to *control sizes of attacker coalitions*, perhaps by providing some assurance of *tamper-resistance* of the devices with preloaded secrets - which is the main reason they have not been given serious attention.

However, the need for *autonomous* operation of the devices, implies that dependency on tamper resistance is *not optional*. Deployments based on PKI for instance, would still need mechanisms to protect the preloaded private keys. After all, every device that is deployed is expected to operate *without human intervention*. Even though many devices may have a "human controller" at hand, it is not practical for the person to store the key just in his / her head and supply it to the device when needed (for each instance of communication)! This realization, is already driving technology to improve tamper-resistance of devices - as is evidenced by a slew of tamper-resistant security modules that have been announced recently[1] - after all, it has always been necessity that has motivated technology.

If tamper-resistance is perfect, KPD schemes are rendered secure. In practice, any form of tamper-resistance (whatever advances technology may bring about), can perhaps be broken, given a motivated attacker with unlimited time and resources. Hence, in this paper we examine a significantly weaker model for tamper resistance. That is, we assume that an attacker is able to compromise only a fraction of the secret bits from each node. Furthermore, we assume an update mechanism that periodically refreshes the secret keys within each node. Under these two assumptions, we perform a detailed analysis of the security of KPD schemes. For reasons which shall be explained in the next section, we restrict ourselves to a class of KPD schemes called *random* KPD schemes. For one such scheme, HARPS [1], we show that under reasonable assumptions, an attacker may have to tamper with many *tens of thousands* of nodes to compromise a security infrastructure. This result implies that perhaps random KPD schemes are an attractive and practical option for large distributed systems, if they are engineered in the right manner.

The rest of this paper is organized as follows. In Section II we provide a very brief overview of KPD schemes and justify limiting our focus to random KPD schemes. We explain why a combination of limited tamper resistance and periodic renewal can dramatically improve the security of random KPD schemes. Three random KPD schemes, LM (Leighton-Micali) [2], RPS (Random Preloaded Subsets) [3] and HARPS (HAshed RPS) [1] are explored in more detail.

M. Ramkumar, Department of Computer Science and Engineering, Mississippi State University, MS.
N. Memon, Department of Computer and Information Science, Polytechnic University, Brooklyn, NY.

[1]See for instance eracom-tech.com, spyrus.com.au, ncipher.com, aepsystems.com, rainbow.com, thales-esecurity.com, cryptomathic.com, h20138.www2.hp.com.

As RPS and LM are special cases of HARPS, the description focuses on HARPS and points out the relationship between the three schemes. In Section III we lay down the assumptions for the model employed for tamper-resistance. Based on this model, we re-evaluate the security provided by the three random KPD schemes. Conclusions and scope for further work can be found in Section IV.

II. Key Pre-distribution

A KPD scheme consists of a trusted authority (TA) who chooses P secrets, and N nodes with unique IDs (say $ID_1 \cdots ID_N$). We shall represent the collection of P secrets with the TA as $\mathcal{R}$, where $\mid \mathcal{R} \mid = P$. Further, the TA chooses two functions $f()$ and $g()$. The function $f()$,

$$\mathcal{S}_i = f(\mathcal{R}, ID_i), \qquad (1)$$

is used to calculate the secrets $\mathcal{S}_i$ that are preloaded in node i with ID ID_i. Two nodes i and j, with preloaded secrets $\mathcal{S}_i$ and $\mathcal{S}_j$ can discover a unique shared secret K_{ij} as

$$K_{ij} = g(\mathcal{S}_i, ID_j) = g(\mathcal{S}_j, ID_i), \qquad (2)$$

without further involvement of the TA.

Obviously, there are restrictions on functions $f()$ and $g()$ in order to satisfy the requirements in Eqs (1) and (2). Also, the function $g()$ is public. This makes it possible for two nodes, just by exchanging their IDs, to execute the function $g()$ and discover a unique shared secret. As the shared secret is a function of their IDs, their ability to arrive at the shared secret provides mutual assurances to i and j that the other node possesses the necessary secrets $\mathcal{S}_j$ and $\mathcal{S}_i$, respectively, and can thus be "trusted". The secrets preloaded in each node is referred to as the node's *key-ring*. We shall represent by k, the size of the key ring, or the number of preloaded secrets preloaded in each node - $\mid \mathcal{S}_i \mid = k$.

Note that the established trust is based on the assumption that no one else, apart from node j has access to the secrets $\mathcal{S}_j$. Typically, if an attacker manages to "expose" secrets buried in a finite number of nodes - say he manages to expose secrets $\mathbb{S}_{\mathbb{A}} = \{\mathcal{S}_{\nVdash} \cup \cdots \cup \mathcal{S}_{\ltimes}\}$ - he may be able use this "knowledge" $\mathbb{S}_{\mathbb{A}}$ to "compromise the system."

The phrase "compromising a KPD scheme," may have different meanings, depending on the motivation of the attacker. An attacker with access to exposed secrets $\mathbb{S}_{\mathbb{A}}$, may be able to "masquerade" as some node i, for the purposes of his interactions with node j. He achieves this by "discovering" the shared secret K_{ij} between the two nodes (by employing his "knowledge" $\mathbb{S}_{\mathbb{A}}$ - the attacker also simultaneously gains the ability to convince node i that he is node j). Some possible motivations (by no means an exhaustive list) then, of an attacker, would be to determine K_{ij} for the following cases

A1 a specific i, j;
A2 a specific i, when j is the TA;
A3 for all i when j is the TA.

There is thus a notion of "extent of damage" that an attacker can do, indicated by levels A1 to A3, depending on the capability and the efforts of the attacker to expose secrets $\mathbb{S}_{\mathbb{A}}$.

A. KPD Schemes

KPD schemes, are trade-offs between security and resource constraints in nodes. In general, more the available resources in each node (as most KPDs use only symmetric cryptographic primitives the "available resources" is a function of the size of the key-ring), more is the effort needed by an attacker to compromise the system. However, different KPD schemes employ different *mechanisms* of trade-offs. For instance, for some KPD schemes (say category I), the effort needed by the attacker for accomplishing any of the attacks A1 to A3 is the same. For other KPD schemes (say category II), it may be substantially easier to accomplish A1 and increasingly difficult to accomplish attacks A2 and A3.

Category I: Category I KPDs that could resist compromises of up to n nodes, are referred to as n-secure KPDs. Typically, the category I KPD schemes are based on finite field arithmetic techniques [4] - [5]. They need only $k = O(n)$ preloaded keys in each node in order to be n-secure. But they suffer from problems of catastrophic onset of failure (as long as n nodes are compromised the system is completely secure - but when a single additional compromised node the *entire system* is compromised - or all attacks A1, A2 and A3 become feasible). Moreover, they are also typically computationally more expensive due to the need for finite-field arithmetic (however they are still considerably less expensive than asymmetric key cryptography).

Category II: The concept of n-secure KPDs, however does not readily extend to describing the category II KPD schemes - a more accurate representation of category II KPDs would be as a (n_1, n_2, n_3)-secure KPD (or n_1, n_2, n_3 nodes need to be compromised to engineer attacks A1, A2 and A3 respectively). Many such KPD schemes [6] - [7], based on *subset intersections* (SI) have been proposed which solve the problems of catastrophic onset of failure and computational complexity associated with category I KPD schemes. In SI schemes, a subset k of the TA's pool of P keys are distributed in a deterministic fashion to each node (each node gets a different subset). The shared secret between any two nodes is a function of the keys the nodes share. However, SI schemes introduce a new disadvantage - the dependence of k on the network size N, which severely restricts their scalability. For SI schemes, typically $k = O(n_1\sqrt{N})$ to $k = O(n_1 \log N)$, (where n_1 is the number of nodes that need to be compromised for accomplishing attack A1). While for the efficient SI schemes (with $k = O(n_1 \log N)$) scalability is not a problem, they employ complex constructions for the deterministic allocation of subsets. This either renders the function $g()$ computationally intensive, or calls for the nodes to exchange long messages consisting of the indexes of the keys they possess before they can discover the shared secret. In addition to introducing significant bandwidth overheads, the main disadvantage of the latter approach is that it *does not provide implicit authentication* of the node IDs.

Category III - Random Key Pre-distribution: Yet another category of KPD schemes (category III or random KPD schemes) [1] - [3], [8] - [9], provide only *probabilistic guarantees* of security - in which case a more appropriate characterization would be (n_1, n_2, n_3)-secure with probabilities of compromise (p_1, p_2, p_3) respectively. For example, a random KPD scheme may provide an assurance that it could "resist" attack A1 when n_1 nodes have been compromised - however with a probability of failure of say $p_1 = 10^{-20}$.

At fist sight, permitting a finite probability of compromise, may seem like a serious disadvantage. *In practice, it is not.* Even category I KPDs which provide deterministic assurances (say to resist n compromised nodes), the final shared secret is a "key" with a *finite number of bits.* For instance, if the shared secret is a 64-bit key, there does exist a finite probability ($\frac{1}{2^{64}} > 10^{-20}$) that an attacker can "pull the secret out of a hat" (without the need to compromise any node). Thus permitting a probability of compromise is not a disadvantage as long as it is comparable to the security offered by the key-length of the final shared key (say $p_1 = \approx 10^{-20}$ for 64 bit keys).

By utilizing this freedom to permit finite compromise probabilities, random KPD schemes perform significantly better than other KPD schemes. The scheme proposed by Leighton and Micali [2] for instance, achieves $k < O(n_1^3)$. Two other random KPD schemes RPS [3] and HARPS [1] do even better by achieving $k = O(n_1)$. RPS and HARPS achieve this by permitting a small "outage probability." In other words, there might exist a finite probability that two nodes *cannot* discover a shared secret! But as long as the outage probability is kept small (which we shall see is indeed the case), this is not really an issue.

B. Tamper Resistance and Key Renewal

Deterrence of the attacker from exposing secrets calls for some assurance of tamper-resistance of devices. Obviously, if tamper-resistance is perfect, KPD schemes are rendered secure. In practice, any form of tamper resistance can perhaps be broken by a motivated attacker with unlimited time and resources. However, it may be be reasonable to expect tamper resistance to provide some limited extent of guarantees [10] - [11]. As a model for limited extent of assurances provided by tamper-resistance, we assume that tamper-resistance can ensure that only a *fraction* of the secrets can be exposed by tampering with any node. The existence of this guarantee, affects different KPD schemes in different ways.

Consider a n-secure category I KPD, where $n = 20$. If the tamper-resistance property guarantees that only 10% of the keys buried in each node can be compromised, then an attacker needs to tamper with more than $10n = 200$ nodes to engineer a successful attack. On the other hand for a category II KPD, with comparable complexity, the attacker may need to tamper with only 50 nodes for accomplishing attack A1 but probably 10000 nodes for accomplishing attack A3. For a category III KPD, (with comparable complexity), an attacker may have to tamper with 120 nodes to accomplish the attack A1 with a probability of 10^{-20}, and probably 500 nodes to accomplish A1 with a probability of 0.5, and say 20,000 nodes to accomplish attack A2 with a probability of 0.5, and perhaps 25,000 nodes to accomplish attack A3 with a probability of 0.5.

Accomplishment of attack A2 (the ability to "fool" the TA), implies successful "synthesis" of a node by an attacker. Increased resistance of KPD schemes to node synthesis (or attack A2) can be used advantageously by *periodic renewal of keys.* For renewal, each node would authenticate itself to the TA using *all* its preloaded secrets, and receive a set of new keys[2]. After key updates, the efforts of an attacker to gather secrets that made it possible for him to perform attack A1, are rendered useless. Categories II and III KPD schemes benefit from such a key renewal infrastructure, which obviously is not nearly as useful for category I KPD schemes. Thus (for category II and III KPD schemes) a combination of "some extent of tamper resistance" and "periodic renewal" of keys has the ability to render them a lot more secure. We shall illustrate this in a quantitative and analytical fashion in the following sections of this paper.

C. HARPS, RPS and LM

In this section we briefly review HARPS [1]. A brief description of the analysis of HARPS presented in [1] (without a model for tamper resistance) is also included as it is necessary for further analysis (with "limited" tamper resistance) carried out in later sections. HARPS is a simple random KPD scheme, the performance of which is dictated by 3 parameters - P the size of the key-pool, k the size of the "key-ring" in each node, and L the maximum "hash depth." A HARPS deployment consists of a TA who chooses P secrets. Each node is given a unique ID. The ID of each node determines the subset of k keys that the node receives (through a public one-way function). Each key in the key-ring of every node is further hashed a variable

[2]This calls for the ability to communicate with the TA periodically. However note that this is also needed for PKI in order to renew revocation lists.

number of times (uniformly distributed between 1 and L, the exact values of which are also determined by the ID of the node through a public one-way function). Two nodes just need to exchange their IDs to determine the secrets they share.

From their IDs, they can arrive at a unique secret (with a very high probability) by determining the keys they share, and the corresponding hash depths of the shared keys. The node which has a lower hash depth for a particular shared key needs to hash forward (that specific key) to reach the same hash depth as the other node. Once both nodes, in this fashion, have reached a common hash depth for all shared keys, they can hash the shared keys (now with same hash depths) together to calculate the shared secret independently. Under proper selection of parameters of the system (P, k, L) we can ensure that the probability of compromise of shared secrets by coalitions of n nodes (which includes the outage probability) is kept to vanishingly small levels. HARPS is a generalization of RPS [3] and LM [2]. LM is a special case of HARPS when $P = k$. RPS is a special case of HARPS when $L = 0$.

Security Analysis: The probability ξ that an arbitrary node is loaded with an arbitrary key from the TA's key pool (of P keys) follows a binomial distribution, where $\xi = \frac{k}{P}$. The probability $\mathcal{P}(\xi, n, u)$, that 2 nodes (the pair trying to establish a shared secret) "pick" a specific key, that is *also picked* by u out of n nodes in the attacker's coalition is given by

$$\mathcal{P}(\xi, n, u) = \binom{n}{u} \xi^{u+2}(1-\xi)^{n-u} \tag{3}$$

In such a situation, the hash depths of the specific key picked by the 2 nodes are say, α_1, α_2, which are uniformly distributed between 1 and L. So are the hash depths $\beta_1 \cdots \beta_u$ of the keys picked by the u nodes in the attacker's coalition. We represent by

$$\begin{aligned} \mathcal{Q}(L, u) &= \Pr\{\min(\beta_1 \cdots \beta_u) > \max(\alpha_1, \alpha_2)\} \qquad (4) \\ &= \begin{cases} \sum_{i=1}^{L} \frac{2i-1}{L^2} \left(\frac{L-i}{L}\right)^u & 1 \leq u \leq n \\ 1 & u = 0 \\ 0 & L = 0, \end{cases} \end{aligned}$$

An attacker who has compromised n nodes, can compromise the shared secret with a probability given by [1]

$$p(n) = (1 - \varepsilon)^P. \tag{5}$$

where

$$\varepsilon = \sum_{u=0}^{n} \mathcal{P}(\xi, n, u)\mathcal{Q}(L, u). \tag{6}$$

In the equation above, the term ε can be interpreted as the "elemental" security offered by each (of P) root-key, from which Eq (5) follows.

For the special cases of RPS and LM respectively,

$$\varepsilon = \begin{cases} \varepsilon_R = \xi^2(1-\xi)^n & L = 0, \xi \neq 1 \\ \varepsilon_L = \mathcal{Q}(L, n) & \xi = 1, L \neq 0 \end{cases}. \tag{7}$$

The suffixes R and L in Eq (7) represent RPS and LM respectively.

From Eqs (5) - (7), it can be shown that $k > O(n^2)$ for LM and $k \approx O(n)$ for RPS and HARPS. More specifically $k \approx 128n$ for RPS and $k \approx 75n$ for HARPS for $p(u) < 10^{-20}$ $\forall u \leq n$ and large n). As a numerical example, a HARPS systems designed to ensure that an attacker who has managed to compromise 20 nodes can only eavesdrop on any arbitrary communication with a probability less than 10^{-20}, would need $k = 1610$ keys (with $P = 19390$ and $L = 64$). For larger n, k increases linearly. The *outage probability*, or the probability that two nodes cannot discover a shared secret is equal to the probability $p(0)$, which is about 5.5×10^{-59} for HARPS with $P = 19390, k = 1610$ and $L = 64$. HARPS outperforms the other two random KPD schemes significantly. For instance, for the same requirements ($p \leq 10^{-20}$ for $n \leq 20$), RPS needs $k = 2565$ and LM, $k = 12659$.

III. Security Analysis of Random KPD Deployments

A. Model for Tamper-Resistance and Assumptions

The issue of "tamper-resistant" hardware modules is perhaps one of the more controversial issues in cryptographic applications [10] - [12]. Some authors [12] have even indicated that reliance on tamper-resistance should not even be considered as an option, based on previously reported low complexity attacks on smart cards [10]. Not withstanding the rebuttal of their claim [13], the same authors (of [10]) have however later pointed out [14] that it may indeed be possible, with simple modifications to smart-cards, to reasonably resist invasive attacks. However, for deployment of *autonomous* nodes tamper resistance is not optional. The only issue here is the "extent" of protection required. For deployments based on PKI, compromise of private keys of one node does not affect other nodes. On the other hand, for systems based on key pre-distribution this is not true.

In this section we propose and examine a significantly weaker model for tamper proofing and analyze the security of the three KPD schemes described in the previous section under this model. In our model, we first assume that an attacker is able to compromise only a fraction of the secret bits from each node [15], [11]. Even though the statement that "only a fraction of bits can be compromised" does not immediately translate to "a fraction of preloaded keys can be compromised," there is conceivably a relationship between the two. The exact nature of the relationship would depend on the underlying hardware and implementation details.

In the remainder of this section, we present an analysis of the security of deployments of random KPD schemes based on the following assumptions:

- The nodes are resistant to passive sniffing.
- The nodes are tamper-resistant. However, a motivated

attacker, can compromise a fraction ρ (and no more) of the secrets from any node. By doing so, the partially compromised node is destroyed.

- The preloaded keys are refreshed periodically (say at intervals T).
- For key updates, the nodes authenticate themselves to the TA by deriving a session key that is based on *all* the k keys they possess.
- The TA maintains a "black-list" or "revocation-list" of nodes.
- The TA maintains the current update status of each node.

B. Attacker's Goals

If the attacker's (Oscar - from here on we shall refer to the attacker thus) motivation is just the ability to *temporarily*[3] compromise communications between nodes (or attack A1), he can do so by compromising partial secrets from many nodes. For example, Oscar could compromise an *equivalent* of n nodes by partially compromising $\frac{n}{\rho}$ nodes. For such a scenario the probability of eavesdropping reported in the previous section apply. For instance, for a HARPS deployment designed for $p = 10^{-20}$ for $n = 20$ (or $P = 19390, k = 1610, L = 64$)[4], Oscar has to compromise an equivalent of 220 nodes to ensure that he can eavesdrop on *any* conversation with probability greater than 0.5. If $\rho = 0.1$, Oscar would have to tamper with (and destroy) 2200 nodes in order to achieve his goal. However, all his hard work is rendered useless after a round of key updates.

The main motivation of Oscar, would therefore be to "synthesize" as many nodes as possible. In other words, if Oscar is able to compromise *all* secrets buried in a node (by tampering with *many* nodes), he can synthesize a node that can even "fool" the TA (attack A2), and hence, *remain in sync with future updates* too. With many such synthesized nodes, Oscar can, at will, compromise the entire system.

First let us assume, that Oscar has managed to collect m nodes. Further, by tampering with m nodes, Oscar is able to collect a pool of ρmk keys. From the ρmk keys, he may without much trouble, be able to arrive at k distinct keys (derived from k unique root keys). However, any such set of k keys does not correspond to the keys of a *valid* node. After all, there are $\binom{P}{k} \times L^k$ unique "key-rings." For example, for $P = 19390, k = 1610, L = 64$, this translates to about 2.1×10^{5315} possible key rings[5]! If, for instance, the ID of each node is represented by 32 bits, only about 4 billion out of 2.1×10^{5315} key rings are permissible.

Let us further assume that Oscar succeeds in finding a particular node ID (any one out of 2^{32}) for which *all* preloaded secrets can be derived by using the pool of ρmk secrets he has accumulated. However, even though the possible network size is 4 billion, only a few million of them may actually be deployed. Thus the particular ID that Oscar arrived at, is probably not even in "circulation".

In such a scenario, even with the synthesized node, Oscar cannot take part in updates (the TA maintains a list of nodes and their update status). Another possibility is that Oscar may end up with a node which is perhaps *not under his control.* It may belong to some other user. In this case, Oscar may be able to participate in one update. However, when the original owner of the node also approaches the TA for an update, the TA would recognize a disparity in update status of the node and thus black-list (or revoke) the node. A black-listed node will not be allowed to participate in updates, and therefore loose their ability to communicate with other nodes. Obviously, if the original owner of the node performed the update, before Oscar (with his synthesized node), approached the TA, Oscar would not be able to participate even in one update.

Thus it is not enough if Oscar manages to compromise a node with *any* ID. It is necessary for him to compromise a node with a *particular* ID - a node which is under his control (he can make sure that that node does not *ever* communicate with the TA). Of course with many nodes under his control Oscar's target is to synthesize one or more nodes from the pool of m nodes under his control.

C. Probability of "Synthesis" of Nodes

We can now calculate the probability that Oscar, who has access to (and is willing to tamper with and thereby, destroy) m nodes, will be successful in synthesizing a node under his control. The probability p_S that he can synthesize a *specific* node by tampering with m nodes is

$$p_S(m) = (1 - \varepsilon_0)^P, \tag{8}$$

where

$$\varepsilon_0 = \sum_{l=0}^{m} \xi \binom{m}{l} (\rho\xi)^l (1 - \rho\xi)^{m-l} \mathcal{Q}(L, 1, l) \tag{9}$$

In Eq (9) above the term $\xi = \frac{k}{P}$ is the probability with which the node that Oscar is attempting to synthesize (or the "target" node) "picks" each key from the pool. Oscar, (by "picking" keys from the m nodes under his control) has a probability of only $\rho\xi$ to pick any arbitrary key from the pool from each of his m nodes. Thus the probability that Oscar is able to pick the same key from exactly l out of m nodes is $\binom{m}{l}(\rho\xi)^l(1 - \rho\xi)^{m-l}$. Further, the hash depth α of the key picked by the target node is uniformly distributed between 1 and L. The l picks of Oscar for the same key are also uniformly distributed between 1 and L. Let the minimum of those l picks be β. Now $\mathcal{Q}_s(L, l)$ is the

[3]Until the next round of key updates.

[4]For the same design goals RPS would need $k = 2565$ keys in each node and LM, $k = 12559$.

[5]This also gives an indication of the level of scalability of HARPS. Taking "birthday paradox" into account, HARPS (with $P = 19390, k = 1610, L = 64$) could potentially support 4.6×10^{2657} nodes!

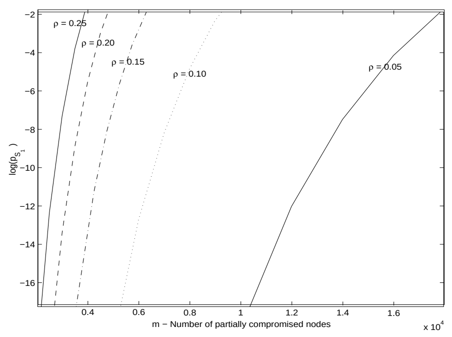

Fig. 1. Plot of the probability of successful synthesis of at least one node and the number of nodes in attacker's control for values of ρ ranging from 5% to 25%. The plots are for HARPS with $P = 19391, k = 1610, L = 64$.

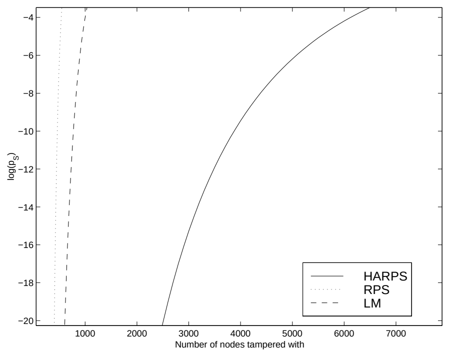

Fig. 2. Plot of the probability of successful synthesis of a specific node p_S, and the number of nodes in attacker's control for $\rho = 0.1$ for HARPS, RPS and LM. For HARPS $P = 19391, k = 1610, L = 64$, $P = 53840, k = 2565$ for RPS and $k = 12559$ for LM.

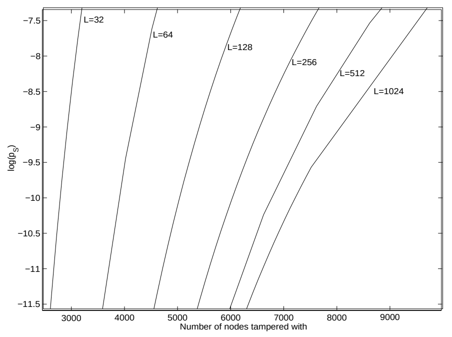

Fig. 3. Plot of the probability of successful synthesis of a specific node p_S, and the number of nodes m in attacker's control for $\rho = 0.25$ for $L = 32$ to $L = 1024$. The plots are for HARPS with $P = 19391, k = 1610$.

probability that $\beta > \alpha$. From Eq (5),

$$\mathcal{Q}_s(L,l) = \sum_{i=1}^{L} \frac{1}{L}\left(\frac{L-i}{L}\right)^l. \tag{10}$$

For the special cases of RPS ($L = 0$) and LM ($\xi = 1$) respectively

$$\varepsilon_{0R} = \xi(1-\rho\xi)^m \tag{11}$$

and

$$\varepsilon_{0L} = \sum_{l=0}^{m} \binom{m}{l} \rho^l (1-\rho)^{m-l} \mathcal{Q}_s(L,l) \tag{12}$$

However, with m nodes under his control, Oscar only needs to synthesize *any* of the m nodes. The probability that Oscar *cannot* synthesize a specific node is $1 - p_S$. Thus the probability that he *can* synthesize at least one node is

$$p_{S_1} = 1 - (1 - p_S(m))^m \approx m p_S(m) \text{ for } p_S(m) << 1. \tag{13}$$

Now the probability that Oscar can synthesize at least d nodes (out of the m nodes under his control) is

$$p_{S_d} = \sum_{i=d}^{m} \binom{m}{i} (p_S(m))^i (1 - p_S(m))^{m-i} \tag{14}$$

Figure 1 depicts plots of $\log(p_{S_1})$ vs m for various values of ρ (for HARPS with $P = 19390, k = 1610, L = 64$). Note that for the same value of p_{S_1}, a reduction in ρ by a factor t increases m by the same factor t, which is intuitive. For example, if ρ reduces five fold Oscar needs to tamper with five times as many nodes (or $m\rho$ is a constant for a fixed p_S).

From the figure, for $\rho = 0.1$ for instance, Oscar has to compromise over 9,800 nodes to ensure that he can synthesize a node with probability of 0.5. However, subsequent synthesis of nodes become much simpler. Oscar has already amassed significant *reusable* "wealth" by compromising nodes - provided his "wealth" is re-used within the same update period. At this stage Oscar needs only about 3000 more nodes before can synthesize about 100 nodes. So by tampering with 12,800 nodes (all within the same update period), Oscar can wreak significant damage to the system (for $\rho = 0.1$). If $\rho = 0.05$ on the other hand, Oscar may need to destroy over 25,600 nodes to achieve this goal (synthesize 100 nodes).

Figure 2 is a comparison of security of HARPS, RPS and LM for $\rho = 0.25$ in terms of resistance to synthesis of nodes. In the figure, the parameters for the three schemes have been chosen so that all three methods have the same eavesdropping probability of $p = 10^{-20}$ for $n = 20$. Specifically, $k = 1610$ for HARPS, $k = 2565$ for RPS, and $k = 12559$ for LM. Thus strictly speaking, the comparison is not "fair" for HARPS. In spite of this, HARPS manages to outperform the other two by a very large margin. Figure 3 depicts plots of $\log(p_S)$ vs m for various values of L for $\rho = 0.25$. Note a three-fold increase in the value of m as L is increases from 64 to 512.

Effect of Additional Update Key: We shall now assume that an additional "update" key is shared between the TA and each node (different for each node) - or each node employs the special update key, in *addition* to all keys in its key ring, to authenticate itself to the TA. For the model for compromising the update key, we assume that Oscar can *either* compromise the update key *or* a fraction of the key ring (if Oscar chooses to compromise the key rings, he destroys the node and cannot compromise the update keys, and vice-versa). The main rationale for ascribing a different level of protection for the update keys are 1) the update keys are unique for each node (unlike the keys in the key-ring); 2) the update key may be fixed while the key ring is renewed periodically; 3) the update key may be longer than the other keys.

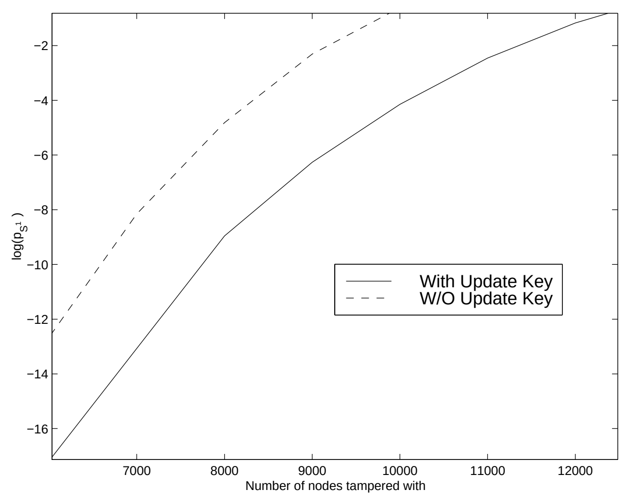

Fig. 4. Plot of the probability of successful synthesis of at least one node p_{S_1}, and the number of nodes in attacker's control for $\rho = 0.1$ with and without the additional update key. The plots are for HARPS with $P = 19391, k = 1610, L = 64$.

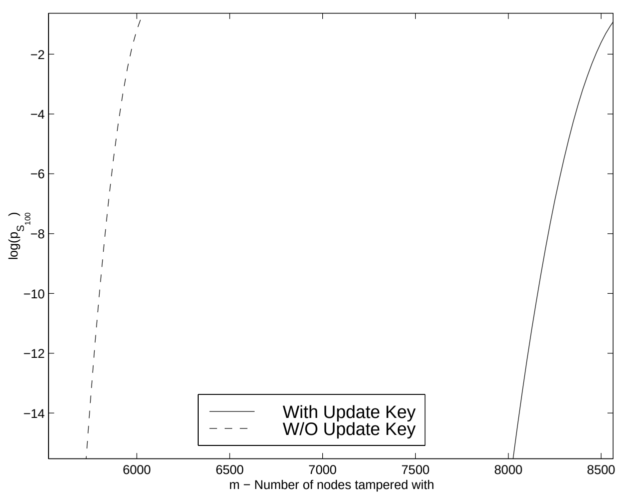

Fig. 5. Plot of the probability of successful synthesis of at least 100 nodes $p_{S_{100}}$, and the number of nodes the attacker has to tamper with, for $\rho = 0.1$, with and without the additional update key. The plots are for HARPS with $P = 19391, k = 1610, L = 64$.

Under such a scenario, Oscar has to divide the set of m nodes he has under his control into two groups. He tampers with the *first* group of $m_1 < m$ nodes, to arrive at his pool of $\rho m_1 k$ secrets. Using this pool, he would try to synthesize a node belonging to the *second* group of m_2 nodes ($m = m_1 + m_2$). After having arrived at all keys belonging to one (or more) of the nodes from the second group of m_2 nodes, Oscar proceeds to tamper with those nodes to expose the update key. Obviously, there is an optimal way of choosing the size of the two groups m_1 and m_2, $m = m_1 + m_2$, depending on the fraction ρ that can be compromised. The expressions for ε_0 in this case would be identical to that of equations (9), (11) and (12), except that m would be replaced by $m_1 < m$. In the expression for p_{S_1} in Eq (13), m would be substituted by $m_2 < m$. For instance for HARPS with $P = 19390, k = 1610, L = 64$, Oscar, with $m = 6000, 8000, 10000, 12000$ would choose roughly $(m_1 = 5800, m_2 = 200), (m_1 = 7650, m_2 = 350), (m_1 = 9500, m_2 = 500), (m_1 = 11250, m_2 = 750)$ respectively.

Figure 4 is a comparison of the two cases - with and without a special update key - depicted as plots of $\log(p_{S_1})$ vs $m = m_1 + m_2$. Note that the addition of a single update key is able increase m by about 25%. Or Oscar's job is rendered 25% harder (for synthesizing one node).

Additionally, the update key also offers improved resistance to *subsequent synthesis* of nodes. In this case, the probability that Oscar can synthesize at least d nodes (from the pool of m_2 nodes, by tampering with m_1 nodes) is

$$p_{S_d} = \sum_{i=d}^{m_2} \binom{m_2}{i} (p_S(m_1))^i (1 - p_S(m_1))^{m_2 - i} \qquad (15)$$

Figure 5 is a plot of m vs the probability $p_{S_{100}}$ of synthesis of 100 nodes for the cases with and without the update key for $\rho = 0.25$ (for any other value of ρ we could just scale the x-axis correspondingly). With the update key, while Oscar needed to tamper with 25% more nodes for synthesizing one node (compared to the scenario without update keys), for synthesis of 100 nodes, Oscar needs to tamper with about 42% more nodes compared to the scenario without update keys.

For all the plots in this section, we have used the same values of $P = 19390$ and $k = 1610$ for HARPS. Obviously the probability of synthesis also depends on the values of P and k. It is easy to see that for the same ratio of $\xi = \frac{k}{P}$, p_S is exponentially related to k. One can thus always increase the resistance to node synthesis by increasing k (if resources permit). For a given k[6], the discussions in this section indicate the following other ways to ensure that it is not worth-while for Oscar to attempt node synthesis:

- Ensuring low values of ρ by sophisticated tamper-proofing technology.
- Increasing L (at the expense of an increase in computational complexity to arrive at shared secrets).
- Using a highly protected update key in addition to the key ring for updates.
- Practical mechanisms to ensure that it is extremely difficult for an attacker to accumulate tens of thousands of nodes under his control.

We conclude this section with another numerical example. For $\rho = 0.05$ (or if Oscar can only expose 5% of the keys by tampering with any node), for the same values of $P = 19390, k = 1610$, but with $L = 512$, and employing the additional update key, Oscar has to tamper with around 75,000 nodes to have a "reasonable shot" at synthesizing a

[6]The probabilities of compromise reduce exponentially with increasing k.

single node. For synthesizing over 100 nodes he might need to tamper with (and destroy) more that $110,000$ nodes!

IV. Conclusions and Scope for Further Work

In this paper we have addressed the possibility of employing random key pre-distribution schemes for securing large scale deployments of possibly resource constrained nodes. The analysis in this paper shows that, a *combination* of a limited extent of tamper resistance and periodic renewal of keys dramatically improves the security of KPD schemes - especially random KPD schemes. Even though it is not very clear if *partial tamper resistance* is achievable, presumably it should be easier to achieve than *complete* tamper resistance. Perhaps, until now there has not been a *need* for security hardware vendors to even consider such an approach. At the risk of sounding repetitive, it has always been need that has driven technology.

The main advantage of a security infrastructure based on KPDs (as opposed to PKI, which a more common approach), is that KPD schemes also permit resource constrained nodes to participate in the deployment. Another advantage of KPD schemes is that in order to discover the shared key two nodes just need to exchange their IDs. Compare this with the case of a security infrastructure based on public keys. For the latter, nodes have to exchange signed certificates (probably running up to a few thousand bits) before they can establish a shared secret. Thus for applications where the bandwidth of the messages exchanged in a session is very small (say 100s of bits), the use of a PKI introduces a very large overhead. The most important limitation of resource constrained devices, is typically their battery life. Improvements in technology may permit security modules capable of performing asymmetric key cryptography, to shrink to sizes that may enable them to be part of any conceivable device that may need to take an active part in any deployment. However, the increased resources (both computation and bandwidth) for PKI based deployments translate to faster draining of battery life. Improvements in battery technology, however, have lagged behind those of improvements in semiconductor technology significantly. KPD schemes would help in conserving this valuable resource.

The obvious disadvantage of KPD schemes is the increased reliance on technology to provide tamper-resistance. But the need for autonomous devices is already driving technology to render tamper-resistance feasible. Another disadvantage of KPD schemes is that authentication mechanisms based on shared keys cannot be used for signature schemes (at least without involving a trusted third party). However, if signatures are rarely used, this (involving a third party) is not a very serious disadvantage. Our ongoing work is focused on more realistic models for the extent of tamper-resistance, involving perhaps more realistic assumptions, firmly rooted on actual hardware design issues.

References

[1] M.Ramkumar and N.Memon, "An efficient key pre-distribution scheme for manet security," *Submitted to the IEEE Journal on Selected Areas of Communication*, 2003.

[2] T. Leighton and S. Micali, "Secret-key agreement without public-key cryptography," *Proceedings of CRYPTO 1993*, pp. 456–479, 1994.

[3] M.Ramkumar, N.Memon, and R.Simha, "Pre-loaded key based multicast and broadcast authentication in mobile ad-hoc networks," *Proceedings of Globecom 2003*, December 2003.

[4] R. Blom, "An optimal class of symmetric key generation systems," in *Lecture Notes in Computer Science*, (Berlin), pp. 335–338, Springer-Verlag, 1984.

[5] T. Matsumoto and M.E.Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, pp. 644–654, December 1986.

[6] P. Erdos, P. Frankl, and Z. Furedi, "Families of finite sets in which no set is covered by the union of 2 others," *Journal of Combinatorial Theory, Series A*, vol. 33, pp. 158–166, 1982.

[7] M. Dyer, T. Fenner, A. Frieze, and A. Thomason, "On key storage in secure networks," *Journal of Cryptology*, vol. 8, pp. 189–200, 1995.

[8] L. Eschenauer and V. Gligor, "A key-management scheme for distributed sensor networks," *Proceedings of the Ninth ACM Conference on Computer and Communications Security*, pp. 41–47, November 2002.

[9] W. Du, J. Deng, Y. Han, and P.K.Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," *Proceedings of the 10th ACM Conference on Computer and Communication Security*, pp. 42–51, 2003.

[10] R. Anderson and M. Kahn, "Tamper resistance - a cautionary note," *Second USENIX Workshop on Electronic Commerce Proceedings*, pp. 1–11, 1996.

[11] A. Clark, "Tamper resistance and crypto variable protection." (revised Jan 1998), based on "Physical Protection of Cryptographic Devices," Lecture Notes in Computer Science, 1987, 2001.

[12] M. Zapata, "Secure ad hoc on-demand distance vector routing," *Mobile Computing and Communications Review*, vol. 6, no. 3, 2001.

[13] S. I. Inc., "Tamper resistance - a second opinion," *available at http://www.smartcard.co.uk/resources/articles/tamper-res.html.*

[14] O. Kummerling and M. Kuhn, "Design principles for tamper resistant smartcard processors," *USENIX Workshop on Smartcard Technology*, 1999.

[15] C. Koc, D. Naccache, and C. Paar, "Fast primitives for internal data scrambling in tamper resistant hardware," *Cryptographic Hardware and Embedded Systems*, vol. CHES2001, pp. 16–28, 2001.