A Hierarchical Key Pre-distribution Scheme

Mahalingam Ramkumar
Dept. of CSE, Mississippi State University

Nasir Memon
Dept. of CIS, Polytechnic University

Rahul Simha
Dept. of CS, The George Washington University

Abstract

We present an efficient, scalable, and renewable hierarchical random key predistribution (KPD) scheme as an enabler for a low complexity security infrastructure. As the proposed KPD scheme employs only symmetric cryptographic primitives it permits resource constrained nodes to take part in the deployment. Further, two nodes need to exchange only their IDs before they can establish a shared secret. The proposed KPD scheme provides scalable security for different levels of the hierarchy depending on available resources. In addition higher levels of the hierarchy are protected from security breaches in lower levels.

1 Introduction

For many applications, for example wireless communication devices employed by soldiers belonging to a regiment to communicate with each other, with their officers, and perhaps soldiers belonging to other regiments, suggest a natural hierarchy of devices deployed. Smooth operation of such deployments calls for a security infrastructure which "mirrors" the organization of the devices. In other words, a hierarchical *key distribution scheme* is called for. This would, for instance, permit devices (or personnel controlling such devices) higher up in the hierarchy, to monitor, control and set permissions and policies for communications involving devices at lower levels.

A common approach for securing such hierarchical deployments is by using a public key infrastructure (PKI). However, solutions based on PKI may not be very suitable for all applications. For instance, the use of public key cryptography may be impractical due to computational demands and bandwidth overhead (due to the need to exchange signed public keys) necessitated by asymmetric cryptography.

KPD schemes are inherently *trade-offs between security and resource utilization*. Their significantly reduced resource requirements (as compared to PKI) would permit even resource constrained devices to be part of the deployment. Their limitation in security is the need to restrict sizes of attacker coalitions, perhaps through *some assurance of tamper-resistance* of the devices with preloaded secrets.

However, the increasing need for *autonomous* operation of devices implies that tamper resistance is not optional. This realization is already driving technology to improve tamper-resistance of devices. While tamper resistance may not be able to provide *unconditional guarantees* in the face of an attacker with unlimited time and resources, it may be able to provide "limited guarantees." Fortunately (we argue), a combination of limited guarantees provided by tamper resistance, and periodic renewal of keys can dramatically enhance the security of KPD schemes. However, even though many KPD schemes have been proposed in literature, not all of them can effectively utilize the advantage provided by key renewal. Further they do not extend readily to hierarchical deployments.

In this paper, we propose a hierarchical random key predistribution scheme. In the proposed scheme "child" nodes are provided with a subset of keys belonging to the "parent" nodes. Further, the subset of keys preloaded in child nodes, are hashed (a variable number of times) using a cryptographic, pre-image resistant hash function. The proposed scheme enables a tree-hierarchical deployment, and protects higher levels of the hierarchy from security breaches in lower levels. Under some realistic assumptions, we show that an attacker may need to tamper with a few hundred thousand nodes (in a finite period of time) in order to compromise the proposed KPD scheme.

In the Section 2 we provide a brief overview of various KPD schemes, and divide them into 2 categories. Various extents of attacks an attacker can inflict on KPD schemes (by compromising nodes) are discussed. In Section 3 we introduce a hierarchical random KPD scheme and provide a quantitative analysis of its security. The proposed scheme is an hierarchical extension of a previously proposed KPD scheme, HARPS [1]. In addition to the hierarchical extension, this paper takes into consideration the effect of partial read-proofing of secrets on the security of the scheme. In

Section 4 we provide some justification for the assumption of partial read-proofing.

2 Key Pre-distribution

A KPD scheme consists of a trusted authority (TA), and N nodes with unique IDs (say $ID_1 \cdots ID_N$). The TA chooses P secrets $\mathcal{R}$ and two operators $f()$ and $g()$. The operator $f()$, is used to determine the secrets $\mathcal{S}_i$ that are preloaded in node i. Any two nodes i and j, with preloaded secrets $\mathcal{S}_i$ and $\mathcal{S}_j$ can discover a unique shared secret K_{ij} using a *public* operator $g()$ without further involvement of the TA. The restrictions on $f()$ and $g()$ in order to satisfy these requirements can be mathematically stated as follows:

$$\begin{aligned} \mathcal{S}_i &= f(\mathcal{R}, ID_i) \\ K_{ij} &= g(\mathcal{S}_i, ID_j) = g(\mathcal{S}_j, ID_i) \\ &= f(\mathcal{R}, ID_j, ID_i) = f(\mathcal{R}, ID_i, ID_j). \end{aligned} \quad (1)$$

As $g()$ is public, it possible for two nodes, just by exchanging their IDs, to execute $g()$ 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.

The established trust is based on the assumption that no one else, apart from node j has access to the secrets $\mathcal{S}_j$. Note that the main difference between KPD schemes and conventional KDSs (like Kerberos, PKI) is that the preloaded keys in different nodes are *not* independent - they are all derived from the same set of P secrets $\mathcal{R}$. Thus, if an attacker manages to expose secrets buried in a finite number of nodes, he may be able use this "knowledge" to "compromise the system." However, the phrase "compromising a KPD scheme," may have different meanings, depending on the attacker's motivation (and capabilities).

With access to secrets $\mathbb{S}_a = \{\mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n\}$ exposed from n nodes (say by tampering with them), The attacker may be able to determine K_{ij}, which allows him to masquerade as node i for his interactions with node j (or vice-versa). Some possible motivations then, of the attacker, would be to determine K_{ij} for the following cases:

1. **Eavesdropping Attack (AE):** A specific i, j. The ability to engineer this attack (by compromising $n = n_e$ nodes) permits the attacker to *eavesdrop* on any communication between nodes i and j.
2. **Synthesis Attack (AS):** A specific i, when j is the TA. A successful attack (by compromising $n = n_s$ nodes) permits the attacker to effectively *synthesize* a (non-compliant) node that can impersonate node i for *any* interaction

There is thus a notion of "extent of damage" that an attacker can inflict, depending on the capability and the efforts of the attacker to expose secrets.

KPD schemes, are inherently trade-offs between security and resource constraints in nodes. In general, more the available resources in each node, more is the effort needed by the attacker to compromise the system. However, different KPD schemes employ different *mechanisms* of trade-offs. For instance, for some KPD schemes (say category 1), the effort needed (number of nodes that need to be compromised) for accomplishing any of the attacks (eavesdropping, synthesis, consummate) is the same. For other KPD schemes (category 2), it may be substantially easier to accomplish eavesdropping attack and increasingly difficult to accomplish synthesis and consummate attacks.

Category 1 KPDs that could resist compromises of up to n nodes, are referred to as n-secure KPDs. Typically, the category 1 KPD schemes are based on finite field arithmetic techniques [2]-[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 (or less) are compromised, the system is completely secure. But with $n + 1$ compromised nodes the *entire system* is compromised - or all three attacks become feasible. Moreover, they are also computationally more expensive due to the need for finite-field arithmetic. In addition, extension of category 1 schemes to hierarchical deployments typically results in significant increase in complexity. Further, with category I schemes, the *same* set of preloaded secrets that enable mutual authentication, *cannot* be used for other security associations needed for multicast security.

The concept of n-secure KPDs, however does not readily extend to describing the category 2 KPD schemes which include random KPD schemes [1], [6] - [8] provide only *probabilistic guarantees* of security - in which case a more appropriate characterization would be (n_e, n_s)-secure with probabilities of compromise (p_e, p_s) respectively (n_e, n_s nodes need to be compromised to engineer eavesdropping and synthesis attacks with probabilities p_e and p_s respectively). For example, a random KPD scheme may provide an assurance that it could "resist" eavesdropping attack even when n_e nodes have been compromised - however with a probability of failure of say $p_e = 10^{-20}$.

3 A Hierarchical Random KPD Scheme

HARPS is a simple random KPD where each node is preloaded with a *hashed* subset of keys belonging to its parent. A *hierarchical* deployment of HARPS starts with a node R with P secrets $\mathcal{R} = \{K_1 \cdots K_P\}$, at the root of the tree (see Figure 1). The root node has many (say N_0) children, with IDs α_i, $1 \leq i \leq N_0$, at level 1. A node α_i has a set of k_1 secrets $\mathbb{A}_i$ - which is a subset of the P secrets $\mathcal{R}$, repeatedly hashed a variable number of times. The choice of

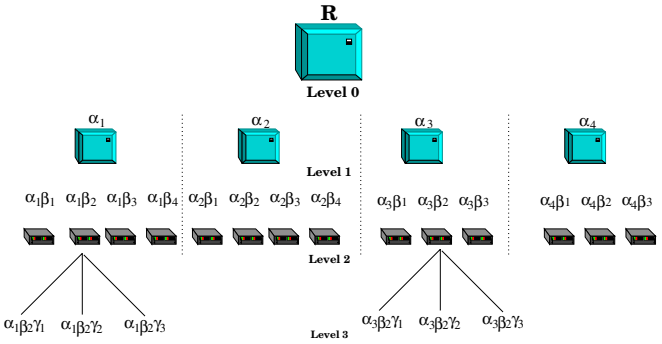

Figure 1. A 4-Level Hierarchical Deployment of HARPS

the subset of keys, and the number of times each chosen key is hashed, is determined by a public operator $f_1()$, and the node ID. Or, $\{(I_1, d_1), (I_2, d_2), \ldots, (I_k, d_k)\} = f_1(\alpha_i)$.

In other words, the first coordinate $\{I_1, I_2, \ldots, I_k\}$ indicates the indexes of the preloaded keys (between 1 and P) in node α_i, and the second coordinates $\{d_1, d_2, \ldots, d_k\}$, their corresponding "hash-depths" - or the number of times each chosen key is hashed. The hash depths of the keys in level 1 nodes are uniformly distributed between 1 and L_1. Thus $\mathbb{A}_i = F(f_1(\alpha_i), \mathcal{R})$, where the function $F()$, "chooses" the keys dictated by the set of first coordinates of $f_1(\alpha_i)$, and hashes each key the required number of times (as dictated by the set of second coordinates of $f_1(\alpha_i)$). As a concrete example, $I_1 = 23, d_1 = 31$ implies that the first preloaded key in node α_i is the the key indexed 23 (or K_{23} of the TA) repeatedly hashed 31 times.

The level 2 children of a node α_i are $\alpha_i\beta_j$, $1 \leq j \leq N_i$, and the level 3 children of the node $\alpha_i\beta_j$ are $\alpha_i\beta_j\gamma_l$, $1 \leq j \leq N_{i,j}$. A public operator $f_2(\beta_j)$ determines the indexes and hash depths of the keys preloaded in $\alpha_i\beta_j$ (w.r.t the parent device α_i) and a similar $f_3(\gamma_l)$ determines the indexes and hash depths of the keys preloaded in $\alpha_i\beta_j\gamma_l$ (w.r.t the parent device $\alpha_i\beta_j$). The secrets preloaded in nodes $\alpha_i\beta_j$ and $\alpha_i\beta_j\gamma_l$ are therefore

$$\mathbb{A}_i\,\mathbb{B}_j = F(f_2(\beta_j), \mathbb{A}_i) \text{ and } \mathbb{A}_i\,\mathbb{B}_j\,\mathbb{G}_l = F(f_3(\gamma_l), \mathbb{A}_i\,\mathbb{B}_j). \quad (2)$$

The indexes of the k_2 preloaded keys in level 2 devices range between 1 and k_1, and their hash depths between $L_1 + 1$ and L_2 ($L_2 > L_1$). Similarly the indexes of the k_3 preloaded keys in level 3 range between 1 and k_2 and the hash depths between $L_2 + 1$ and L_3 ($L_3 > L_2$). Note that as long as the hash function used is pre-image resistant, compromise of secrets in lower levels of the hierarchy does not affect the higher levels.

3.1 Security Analysis

The analysis of security of HARPS involves estimation of the probability of success of attacks AE and AS by an attacker who has managed to expose secrets from m nodes. We shall assume that the attacker can expose only a fraction ρ ($0 \leq \rho \leq 1$) of secrets[1] by tampering with any node (we shall provide a justification for this assumption in Section 4). Further, by doing so, the node is *rendered unusable* in future.

The extent of damage an attacker can inflict, depends not just on the value of m (the number of nodes tampered with) and the fraction ρ, but also on the "position" of the nodes in the hierarchy. Let us assume that the (partially) compromised nodes belong to the level 2, and that they are all children of one level 1 node, say α_1 - or let the m nodes be $\alpha_1\beta_i \cdots \alpha_1\beta_{i+m}$. While knowledge of level 2 secrets has no effect on the security of level 1 and level 0 secrets, they could potentially be used for compromising communications involving other level 2 and level 3 nodes.

By tampering with m nodes in level 2, $\alpha_1\beta_i \cdots \alpha_1\beta_{i+m}$, the attacker exposes a total of $m\rho k_2$ secrets. The exposed secrets are hashed versions (hash depths uniformly distributed between $L_2 + 1$ and L_3) of a subset of secrets of the root node R. With the exposed secrets, he could discover the shared secrets between various nodes (to accomplish attack AE) or synthesize nodes (to accomplish attack AS).

3.1.1 Resistance of HARPS to Attack AE

What we are interested in now, is the probability p_e that an attacker can discover the shared secret between say,

- SS nodes $\alpha_1\beta_1$ and $\alpha_1\beta_2$ (two siblings attempting to discover their secret and the compromised nodes are also siblings of both nodes).
- CS nodes $\alpha_1\beta_1$ and $\alpha_2\beta_1$ (two cousins try to discover a secret when the compromised nodes are siblings of one of the nodes).
- SC nodes $\alpha_2\beta_1$ and $\alpha_2\beta_2$ (two siblings try to discover a shared secret, and the compromised nodes are cousins).
- CC nodes $\alpha_2\beta_1$ and $\alpha_3\beta_1$ (two cousins try to discover a secret when the compromised nodes are *not* siblings of either node).

For the four different scenarios, the probability of eavesdropping p_e (or the probability of success of attack AE) is given by $p_e(m) = (1 - \varepsilon)^P$, where

$$\begin{aligned} \varepsilon &= \mathcal{P}_s \textstyle\sum_{u=0}^{m} \mathcal{P}_a(m, u)\mathcal{P}_q(u, L). \\ \mathcal{P}_q(u, L) &= \textstyle\sum_{i=1}^{L} \frac{2i-1}{L^2} \left(\frac{L-i}{L}\right)^u. \end{aligned} \quad (3)$$

[1] See for instance "partially openable chips" in [8].

Table 1. Expressions for $\mathcal{P}_s$ and $\mathcal{P}_a(m,u)$ in Eq (3) for scenarios SS, CS, SC and CC.

SS	$\alpha_1\beta_1 \leftrightarrow \alpha_1\beta_2$	$\mathcal{P}_s = \xi_1\xi_2^2$	$\mathcal{P}_a(m,u) = \binom{m}{u}(\rho\xi_2)^u(1-\rho\xi_2)^{m-u}$
CS	$\alpha_1\beta_1 \leftrightarrow \alpha_2\beta_1$	$\mathcal{P}_s = (\xi_1\xi_2)^2$	$\mathcal{P}_a(m,u) = \binom{m}{u}(\rho\xi_2)^u(1-\rho\xi_2)^{m-u}$
SC	$\alpha_2\beta_1 \leftrightarrow \alpha_2\beta_2$	$\mathcal{P}_s = \xi_1\xi_2^2$	$\mathcal{P}_a(m,u) = \binom{m}{u}(\rho\xi_2\xi_1)^u(1-\rho\xi_2\xi_1)^{m-u}$
CC	$\alpha_2\beta_1 \leftrightarrow \alpha_3\beta_1$	$\mathcal{P}_s = (\xi_1\xi_2)^2$	$\mathcal{P}_a(m,u) = \binom{m}{u}(\rho\xi_2\xi_1)^u(1-\rho\xi_2\xi_1)^{m-u}$

Table 2. Resistance of hierarchical HARPS to attack AE.

ξ_2 / ξ_3	p_e for SS	p_e for CS	p_e for SC	p_e for CC
	0.5 / 10^{-6} / 10^{-20}	0.5 / 10^{-6} / 10^{-20}	0.5 / 10^{-6} / 10^{-20}	0.5 / 10^{-6} / 10^{-20}
0.075 / 0.75	2400 / 505 / 190	2050 / 420 / 125	3150 / 675 / 253	2750 / 560 / 168
0.075 / 0.50	2400 / 505 / 190	1690 / 312 / 40	4700 / 1010 / 380	3400 / 625 / 81
0.050 / 0.75	2940 / 584 / 151	2560 / 470 / 60	3850 / 780 / 203	3400 / 625 / 81

The term $\mathcal{P}_s$ is the probability that two nodes trying to discover a shared secret, share a particular key (say key index i, $1 \le i \le P$). The term $\mathcal{P}_a(m,u)$ is the probability that exactly u instances of the i^{th}key is present in the set of all keys exposed by the attacker (by revealing a fraction ρ of secrets from m nodes). The term $\mathcal{P}_q(u,L)$ is the probability that *maximum* of the hash depths of the i^{th}key shared by the two nodes attempting to discover shared secret, is smaller that the *minimum* hash depth of the u instances of the i^{th}key that the attacker has access to. The expressions for $\mathcal{P}_s$ and $\mathcal{P}_a(m,u)$ for the four different scenarios are given in Table 1. In the table, $\xi_2 = \frac{k_2}{k_1}$, and $\xi_1 = \frac{k_1}{P}$. Table 2 depicts the relationship between p_e and m, for $\rho = 0.1$. In other words, Table 2 depicts the number of nodes the attacker may need to tamper with (m) to eavesdrop on communications between two nodes (for the four different cases SS, CS, SC and CC), with probabilities of $p_e = 0.5$, $p_e = 10^{-6}$ and $p_e = 10^{-20}$ respectively. The results are based on the choice of $\rho = 0.1$, $k = 1500$, and $L = 512$, for three different choices of ξ_2 and ξ_3. As an example, for $\xi_2 = 0.075, \xi_3 = 0.50$ (row 2), for a scenario where two cousins interact and the compromised nodes are siblings of the interacting nodes (CS), the attacker needs to tamper with 1690 nodes to be able to obtain the shared key between the interacting cousins with a probability of 0.5.

3.1.2 Resistance of HARPS to Node Synthesis

For accomplishing attack AS, the attacker should be able to expose *all* keys buried in a node. If he has exposed a fraction ρ of keys from m nodes, the probability of synthesis of a *specific* node is $p_s = (1-\varepsilon\prime)^P$, where

$$\begin{aligned} \varepsilon\prime &= \mathcal{P}\prime_s \textstyle\sum_{u=0}^{m} \mathcal{P}\prime_a(m,u)\mathcal{P}\prime_q(u,L) \\ \mathcal{P}\prime_q(u,L) &= \tfrac{1}{L}\textstyle\sum_{i=1}^{L}\left(\tfrac{L-i}{L}\right)^u \end{aligned} \tag{4}$$

and $\mathcal{P}\prime_s = \frac{k_1}{P} = \xi_1$, the probability that a targeted node *has* a particular key (say index i, $1 \le i \le P$). $\mathcal{P}\prime_a(m,u)$ is the probability that exactly u instances of the i^{th}key is present in the set of all keys exposed by Oscar. The term

Table 3. Resistance of hierarchical HARPS to attack AS (or node synthesis).

ξ_2	S		C	
	$\xi_1 = 0.75$	$\xi_1 = 0.50$	$\xi_1 = 0.75$	$\xi_1 = 0.50$
0.075	101000	101000	133000	196000
0.050	181000	181000	238000	353000

$\mathcal{P}\prime_q(u,L)$ is the probability that hash depth of the i^{th}key in the target node is smaller that the minimum hash depth of the u instances of the i^{th}key that the attacker has access to. For the case of synthesis of a node by attacking other sibling nodes (S) or cousin nodes (C),

$$\mathcal{P}\prime_a(m,u) = \begin{cases} \binom{m}{u}(\rho\xi_2)^u(1-\rho\xi_2)^{m-u} & \text{(S)} \\ \binom{m}{u}(\rho\xi_2\xi_1)^u(1-\rho\xi_2\xi_1)^{m-u} & \text{(C)} \end{cases} \tag{5}$$

where $\xi_2 = \frac{k_2}{k_1}$. With access to m nodes, the attacker can *expect* to successfully synthesize **one** node if $m \approx \frac{1}{p_s}$. Table 3 illustrates the number of nodes the attacker needs to tamper with in order to successfully engineer attack AS for two different scenarios (the compromised nodes may be siblings (S) or cousins (C)), for $k = 1500, L = 512, \rho = 0.1$) for four different values of ξ_2, ξ_1. Note that he has to compromise over 100000 nodes for all situations.

It should be pointed out here that the results reported in Table 3 based on Eq (4) assume that in order to synthesize a node, the necessary keys are obtained by tampering with *other* nodes. However, a fraction ρ of the desired keys are obtained by tampering with the target node itself. Thus we only need to obtain the remaining fraction $1-\rho$ by tampering with *other* nodes. The revised estimate m_r (of m, the number of nodes an attacker needs to tamper with) can be easily obtained by adjusting the estimates of m in Table 3 as $m_r = (1-\rho)m + 1$.

However, for small ρ, $m_r \approx m$. Obviously, if $\rho = 1$, an attacker needs to tamper with only one node in order to synthesize one node.

3.2 Summary of Properties

The performance of hierarchical HARPS obviously depends on many factors like k (the size of the key ring, which is different for each level), L (the range of hash depths in each level - which may be different for each level), the fraction ρ (the "guarantee" provided by tamper resistance) and the choice of the ratios of key ring sizes between levels (like ξ_3 and ξ_2 in our discussion) and the desired probability of failure. Due to limitations of space we have restricted ourselves to reporting the values for a fixed k_2, L and ρ. However the estimates of m for other cases can be easily extrapolated.

In order to increase m (or resistance to attacks), apart from optimizing the values of ξ_2 and ξ_3, we could increase the value of k_2 and / or strive to reduce ρ (by improving technology for tamper resistance). An increase in k_2 by a factor t results in a t fold increase in the value of m. Also, an increase in ρ by a factor t reduces m by the same factor. In other words $m \propto \frac{k_2}{\rho}$.

The value of m can also be increased by increasing L - however the relationship is not linear. Numerical evaluations show that m increases approximately 3 fold as L increases from 32 to 512. The exact nature of the relationship is currently being investigated.

The value of ξ_3 determines the security of interactions between nodes in different branches of the deployment. A small value of ξ_3 results in reduced security of inter branch interactions. However, the loose coupling between branches results in increased resistance to a compromises in other branches. However, any desired level of security can always be obtained at the expense of increased complexity (increasing the number of preloaded keys).

4 Partial Read-Proofing

A fundamental difference between KPDs and conventional KDSs is that KPDs typically have multiple secrets in each node, while a KDS based on Kerberos or PKI have only one secret for each node. In other words, for PKI and Kerberos, there are times when the entire secret needs to be stored in RAM[2] (whenever the key is needed for some computation). However, for KPDs, only *one* of the k keys stored in a node is actually needed for computation at any point in time. For instance, even though a shared secret K_{AB} between two nodes A and B may be a function of m keys, the actual calculation of the secret K_{AB} may need only one key at a time. This difference provides a significant advantage for KPDs - and a more practical way of realizing "partial read-proofness" - it is easier to keep *most* KPD secrets "protected" at any point in time!

[2]This is not strictly true for asymmetric ciphers like RSA which need to perform exponentiation using the private key. At any time only one bit of the private exponent is needed.

In other words, we can ensure that all keys are *always* encrypted at *any* point in time except for one, (or a small set of keys) currently in use - or we *d*ecrypt *o*nly *w*hen *n*ecessary![3] A single key K_V used for encrypting (and decrypting) the KPD keys could itself be stored in a volatile CPU register. Whenever some intrusion is sensed, as long as the key K_V is erased, very few (or just one) KPD keys which are currently unprotected could be exposed by tampering.

For example, an attackers strategy may be to suddenly immerse a functioning chip in liquid nitrogen and freeze all "bits" in their current state. As long as a sensor (which would perhaps sense sudden changes in temperatures) could react fast enough to erase the single volatile key K_V stored in RAM, the attacker may be able to expose only a small fraction of keys which are *currently* unprotected. In theory, there is no reason why it should not be possible to ensure that no more than one secret is "exposed" at any point in time (if a node has 1000 secrets in its key-ring this translates to $\rho = 0.001$).

While it may be easy to protect a single key K_V when a device is in use, protecting the key when the device is at rest may be more difficult. However physical unclonable functions (PUFs) [10] offer a nice solution to this problem. PUFs employ uncontrollable and unique delays in fabricated chips (even though many chips may be manufactured with identical masks, each chip would have some differences in the delays in their circuits). The delays could be used to implement a *unclonable* one-way function. Thus a one-way function $\mathcal{H}$ provided by the PUF could be provided a challenge X to obtain a response K_X. The secret to be protected at rest could be encrypted with K_X and stored in non-volatile memory, along with X. When the device is powered on, the key K_X could be regenerated by challenging the PUF with X. However, when the device is powered off, there is no way for an attacker to determine K_X from X!

However, the use of PUFs may not be feasible for very low powered devices like wireless sensors. But for such applications, typically, the effective lifetime of a device may be limited by the battery life. In other words, such devices may remain powered-on throughout their useful lifetime. When the battery dies down, the secrets stored in the device would be rendered unusable as K_V (which is used to encrypt all secrets) is lost.

4.1 Node Synthesis and Key Renewal

Accomplishment of attack AS (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 AS) can be used advantageously by *periodic renewal of keys*. For renewal, each node would authenticate itself to its parent using *all* its preloaded secrets, and receive a set of

[3]In [9] this is referred to as the DOWN policy.

new keys. If $\rho < 1$ the attacker can only expose a fraction of the keys by tampering with a target node. For determining other keys, the attacker has to carry out a synthesis attack. As long as synthesis attack is prohibitively expensive (with the down policy in place - or $\rho = \frac{1}{k}$, the attacker may have to compromise one secret from a few *million* nodes in order to engineer attack AS) the attacker cannot take part in key renewal with a synthesized node.

After key updates, the efforts of an attacker to gather secrets that made it possible for him to perform attack AE, are rendered useless. Category 2 KPD schemes benefit from such a key renewal infrastructure, which obviously is not nearly as useful for category 1 KPD schemes. Thus (for category 2 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.

The tree-hierarchical deployment also helps avoid *update floods* for the renewal process. Note that each node only needs to approach its parent for renewal. The renewal process could start with the root node replacing some of its P secrets with new secrets. It is also possible that some secrets may be replaced with their pre-images (under the one-way hash used for HARPS[4]). Nodes in Level 1 approach the root node for key updates. After a level 1 node has renewed its secret it child nodes (in Level 2) can renew its secrets by interacting with the parent node.

5 Conclusions

The main motivation of this paper is the extension of a KPD scheme to hierarchical deployments, and an analysis of its security under some "reasonable" assumptions of guarantees provided by tamper resistance. Our analysis demonstrates that it may be impractical for an attacker to compromise a security infrastructure based on hierarchical HARPS.

While dependence on tamper resistance has, and perhaps will continue to be a controversial issue [11]-[14] among cryptographers, there is no denying the fact that it is indeed *mandatory* for deployments with autonomous devices. We could therefore expect technology, driven by need, to provide suitable solutions to this problem.

References

[1] M. Ramkumar, N. Memon, "An Efficient Random Key Pre-distribution Scheme for MANET Security," IEEE Journal on Selected Areas of Communication, March 2005.

[4] In order to do this the root node should have generated a one-way chain of such secrets and used the last key in the chain as the corresponding root secret - this could be done for each of the P secrets.

[2] R. Blom, "An Optimal Class of Symmetric Key Generation Systems," *Advances in Cryptology: Proc. of Eurocrypt 84*, Lecture Notes in Computer Science, **209**, Springer-Verlag, Berlin, pp. 335-338, 1984.

[3] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung, "Perfectly-Secure Key Distribution for Dynamic Conferences," Lecture Notes in Computer Science, vol 740, pp 471–486, 1993.

[4] T. Matsumoto, M.E.Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, **IT-22**(6), Dec. 1976, pp.644-654.

[5] D. R. Stinson, T. van Trung, "Some New Results on Key Distribution Patterns and Broadcast Encryption," *Designs, Codes and Cryptography,* **14** (3) pp 261–279, 1998.

[6] M. Dyer, T. Fenner, A. Frieze and A. Thomason, "On Key Storage in Secure Networks," *Journal of Cryptology,* **8**, 189–200, 1995.

[7] M. Ramkumar, N. Memon, R. Simha, "Pre-Loaded Key Based Multicast and Broadcast Authentication in Mobile Ad-Hoc Networks," Globecom-2003.

[8] T. Leighton, S. Micali, "Secret-key Agreement without Public-Key Cryptography,"*Advances in Cryptology* - CRYPTO 1993, pp 456-479, 1994.

[9] M. Ramkumar, "DOWN with Trusted Devices," submitted to the New Security Paradigms Workshop (NSPW 2005).

[10] B. Gassend, D. Clarke, M. van Dijk, S. Devadas, "Controlled Physical Random Functions," 18th Annual Computer Security Applications Conference, San Diego, CA, Dec 2002.

[11] M.G. Zapata, "Secure Ad hoc On-demand Distance Vector Routing," Mobile Computing and Communications Review, **6**(3), 2001.

[12] R. Anderson, M. Kahn, "Tamper Resistance - a Cautionary Note," Second USENIX Workshop on Electronic Commerce Proceedings, pp 1-11, Oakland, CA 1996.

[13] Semiconductor Insights Inc., "Tamper Resistance - A Second Opinion," available at http://www.smartcard.co.uk/resources/articles/tamper-res.html.

[14] E. Auer, "Tamper Resistant Smart Cards - Attacks and Counter Measures," available at http://www-krypt.cs.uni-sb.de/teaching/ seminars/ss2000/auer.pdf, Sep 2000.