Proceedings of the 2005 IEEE
Workshop on Information Assurance and Security
United States Military Academy, West Point, NY

Safe Renewal of a Random Key Pre-distribution Scheme for Trusted Devices

Mahalingam Ramkumar
Department of Computer Science and Engineering
Mississippi State University, Mississippi State, MS 39762

Abstract—**Evolving application scenarios involving ubiquitous, heterogeneous devices (some of which may be severely resource constrained) forming co-operative ad hoc networks, calls for a different model for "trust." It is the devices that are trusted - not the operators or the "owners" of the devices. Any security solution based on *trusted devices* demands mechanisms for *read-proofing* the secrets stored in *tamper-resistant* devices. However, as perfect tamper-resistance may not be feasible, for long-lived security of such deployments, it is essential that the stored secrets are *renewed* periodically. This paper addresses issues involved *safe renewal* of secrets stored in trusted devices. For safe renewal of keys, (irrespective of the key distribution scheme used) some assurances from technology for tamper-resistance is needed. In this paper we address issues involved in safe renewal of a recently proposed random key pre-distribution scheme, HARPS (hashed random preloaded subsets) [1]. We discuss 1) some "reasonable" assurances that technology could provide (like *partial tamper resistance* and *circuit-delay based authentication*), and 2) possible security precautions and policies (like use of a pass-phrase, use of an additional stored secret, and rest encryption), and their effect on the security of HARPS.**

I. Introduction

In evolving application scenarios like smart homes, intelligent spaces, medibots [2], or more generally, application scenarios catering for *anywhere / anytime connectivity to anything*, ubiquitous and heterogeneous (and mostly wireless) devices, for very different purposes, and with varying capabilities, are expected to organize themselves [3] into ad hoc, pervasive, computing / communication networks[1]. While such ad hoc networks (AHNs), may not *rely* on fixed infrastructure, they may themselves *become* crucial infrastructures for our day to day computing and communication needs, and therefore need to be protected from malicious intents aimed at sabotaging the infrastructure. What is sorely needed, for smooth functioning of such deployments, is a mechanism for establishing **trust** between any two devices.

A. Trust

In conventional client-server applications, it is always an *end-user* that is trusted (for example, not to disclose their password). However, for ubiquitous computing / communication applications (the principal characteristic of which is *mutual co-operation*), placing trust on end-users is not enough. For example, nodes forming mobile ad hoc networks (MANETS) have to co-operatively build routing tables, and relay messages destined for other nodes. In such a scenario, malicious action by a single node (or a few colluding nodes) could have a potentially disruptive effect over the entire network - for an attacker "controlling" one or more nodes can inflict significant harm to *other* nodes. It is therefore vital that the nodes (or devices) people possess (or operate) "behave responsibly." While it may not be possible to force the owners / operators of the nodes to behave in a responsible fashion, it may be possible to force the *devices themselves* to do so. In other words, *it is the devices that have to be trusted - not the users*!

Two devices can trust each other if there exists some means of convincing each other that they "play by the rules," or are "compliant" (to some pre-imposed rules) [4]. For instance, in a MANET scenario, compliant nodes may guarantee that they will faithfully forward messages as required, and will not advertise false routing information.

From a cryptographic perspective, two nodes can trust each other if they can establish an *authenticated* shared secret. This could facilitated by a key distribution scheme (KDS), which provides each node with one or more secrets. The KDS secrets would then used to establish or discover inter-nodal shared secrets. The KDS secrets provided to a node could however, be used as a *hook* for compliance - only nodes (or devices) that have been *checked* for compliance would be provided with the necessary secrets. Thereafter, the ability of any two nodes to establish an authenticated shared secret, indirectly provides a means for *verification of compliance*.

For applications demanding compliance of nodes (or trusted devices), tamper resistance and read-proofing [5] are *mandatory*. Just as a user is trusted not to reveal their secrets in client-server applications, trusted devices are trusted not to reveal *their* secrets. In the absence of the assurance of read-proofness, secrets that serve as a hook for compliance could be *transferred* to non-compliant devices. In the absence of the assurance of tamper-resistance, the components (or software) that ensure compliance of a device could be modified. The need for tamper resistance in evolving application scenarios is already driving technology - as is evidenced by a slew of companies manufacturing tamper resistant modules.

B. Key Pre-distribution

Privacy constraints dictate that interactions between any two nodes, for purposes of establishing mutual trust, should not need external mediators - or trust should be established in an *ad hoc* manner. This eliminates Kerberos [6] (or any variant of the

[1] Or even as a single virtual Global Ubiquitous Computer (GUC) [2].

0-7803-9290-6/05/$20.00 ©2005 IEEE.

Needham-Schroeder symmetric key protocol [7]) as a viable alternative. A more common approach, under these circumstances, is the use of public key infrastructure (PKI) [8]. However, solutions based on PKI may not be very suitable for all applications due to the accompanying computational and bandwidth overheads.

The requirement of ad hoc establishment of shared secrets, with low overheads, is however facilitated by key predistribution (KPD) schemes. KPD schemes are inherently *tradeoffs between security and resource utilization*. Their significantly reduced resource requirements (compared to PKI) would permit even resource constrained devices to be part of the deployment. On the other hand, unlike conventional KDSs (for which the secrets stored in each node are *independent*), for KPDs, the secrets assigned to each node are *not independent*. This implies that compromise of secrets in a finite number of nodes may result in the compromise of the entire KPD. While this may be unacceptable for conventional (client-server) applications, it is not necessarily so for emerging applications, for the following reasons:

1. An attacker controlling a finite number of devices could wreak havoc on the entire system. In such an event (if a conventional KDS is used), the fact that the *KDS is not compromised* while the *deployment is*, is not of much help!
2. The fact that tamper-resistance / read-proofing are *mandatory*, would severely limit an attackers ability to compromise secrets from "many" nodes.

In other words, it is mandatory in evolving applications scenarios to take proactive steps to limit the number of compromised devices, *irrespective* of the KDS used to secure the deployment. While it may be naive to assume that technology for tamper resistance / read-proofing may be able to provide *unconditional guarantees*, even in the face of an attacker with unlimited time and resources, it may be reasonable to expected some "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 HARPS[2].

C. Organization of the Paper

Having provided the motivation for the suitability of KPD schemes for securing trusted devices forming co-operative ad hoc networks, the rest of the paper is organized as follows.

In the Section II we provide a brief introduction to KPD schemes and classify them based on the *extent* of harm, an attacker can inflict, by compromising secrets from a finite number of nodes. We argue that the property of "high resistance to node synthesis" (of a KPD), [9] is very useful for long-lived security of the deployment. Section II also contains a brief description of HARPS (a summary of results from Refs [1] and [9]).

The primary contribution of this paper is in Section III, where we address the issues involved in *safe renewal* of the KPD over open channels. For practicality, renewal has to occur over open channels - in which case, the devices need to authenticate themselves to the trusted authority (TA or manufacturer - who distributes the secrets to the trusted devices in the first place) using their *current* secrets, in order to receive new secrets. Obviously, if an attacker has compromised all secrets stored in a device, he can also take part in the renewal process and gain access to the new secrets - or the renewal is *not* safe. Therefore, some assurances from technology for tamper-resistance / read-proofing is necessary to ensure safe renewal.

We consider two such "reasonable" assurances

1. partially openable chips (see for instance Leighton and Micali [10]), and
2. an unexposable (and possibly weak) secret based on circuit-delays (Gassend et al [11]).

and briefly discuss some practical techniques to realize such assurances. Further, we also consider the effect of some additional security measures like

1. Use of additional password for renewal
2. Use of a *unique* (additional) stored secret for renewal, and
3. Rest encryption [12],

on the security of the renewal process. If safe renewal is practical (or if it is impractical for an attacker who has exposed old secrets to take part in the renewal process and thereby discover new secrets), then it significantly reduces the motivation of potential attackers to compromise the system. For the fruits (ability to perform malicious tasks) of their labor (effort involved in exposing secrets) are short-lived, as the exposed keys are rendered useless after key updates. Conclusions are presented in Section IV.

II. 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:

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

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

[2]This is not necessarily true for all KPDs

(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 (hereafter referred to as Oscar) 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 Oscar'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), Oscar 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 Oscar, would be to determine K_{ij} for the following cases:

1. **Eavesdropping Attack:** A specific i, j. The ability to engineer this attack (by compromising $n = n_e$ nodes) permits Oscar to *eavesdrop* on any communication between nodes i and j.
2. **Synthesis Attack:** A specific i, when j is the TA. A successful attack (by compromising $n = n_s$ nodes) permits Oscar to effectively *synthesize* a (non-compliant) node that can impersonate node i for *any* interaction[3].

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

A. KPD Schemes

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 Oscar 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 for accomplishing any of the attacks (eavesdropping or synthesis) is the same. For other KPD schemes eavesdropping attack can be substantially easier than synthesis attack.

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 [13], [14], [15], [16]. 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 attacks become feasible.

The concept of n-secure KPDs, however does not readily extend to describing the category II KPD schemes - a more accurate representation which would be as a (n_e, n_s)-secure KPD. In other words, n_e, n_s nodes need to be compromised to engineer eavesdropping and synthesis attacks respectively. Many such KPD schemes [17], [18], [19] based on *subset intersections* (SI) have been proposed. In SI schemes, a subset of cardinality k, of the TA's pool of P keys, are distributed in a *deterministic* fashion to each node (each node gets a different subset), most of them, motivated by the seminal work of [20] and [21]. The shared secret between any two nodes is a function of the keys the nodes share (or the intersection of the subsets).

Category III, or *random* KPD schemes 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. 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 $p_e = 10^{-20}$. Noting that the "probability" that one can break a 64-bit encryption by "guessing" the key is $\frac{1}{2^{64}} > 10^{-20}$, probabilistic guarantees are not necessarily inferior - as long as the probabilities of compromise are small.

Most random KPD schemes (with the exception of [10]) are simple extensions of the category II KPD schemes, with the twist that the secrets are assigned *randomly* [22], [23], [24] or *pseudo-randomly* [1], [25], [26] (instead of the deterministic fashion in category II schemes). We refer to all such methods as random preloaded subsets (RPS). Unlike RPS schemes, LM [10] is based on repeated hashing of keys distributed to different nodes. HARPS [1], is a generalization of both LM and RPS. In general, Category II and III methods offer a higher resistance to node synthesis attack ($n_s >> n_e$) than Category I methods (for which $n_s = n_e$).

B. HARPS

HARPS is a simple random KPD where each node is preloaded with a *hashed* subset of keys belonging to its parent. HARPS (like SI schemes and RPS) consists of a TA with P secrets $\mathcal{R} = \{K_1 \cdots K_P\}$, and N nodes with unique IDs $\alpha_1 \cdots \alpha_N$. A node with ID α_i is preloaded with a set of k secrets $\mathbb{A}_i$ - which is a subset of the P secrets $\mathcal{R}$, repeatedly hashed a variable number of times using a public cryptographic hash function $h()$.

The choice of the subset of keys, and the number of times each chosen key is hashed, is determined by a public operator $f()$, and the node ID. Or,

$$\{(I_1, d_1), (I_2, d_2), \ldots, (I_k, d_k)\} = f(\alpha_i). \quad (1)$$

In other words, the first coordinate $\{I_1, I_2, \ldots, I_k\}$ indicates the indexes[4] 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 $d_i, 1 \leq i \leq k$ are integers uniformly distributed between 1 and L (L is the maximum hash-depth). If we represent by K_i^j the result of repeated hashing of K_i, j times, or

$$K_i^j = h^j(K_i) = h(h(\cdots j \text{ times } \cdots (K_i)) \cdots) \quad (2)$$

[3] While Oscar could synthesize a target node A by tampering with and exposing all secrets from the node A itself, for reasons that shall be explained later, the synthesis attack represents exposing all secrets buried in a target node by tampering with *other* nodes.

[4] $\{I_1, I_2, \ldots, I_k\}$ could be realized as a *partial* random permutation of $\{1 \cdots P\}$.

the k preloaded keys in node α_i can be represented as $\mathbb{A}_i = \{K_{I_1}^{d_1} \cdots K_{I_k}^{d_k}\}$.

As mentioned earlier, HARPS is a generalization of RPS [26] and LM [10]. Specifically, RPS is HARPS with $L = 0$ (keys are not hashed before pre-loading) and LM is HARPS with $P = k$ (all TA's keys are preloaded in every node, but are however hashed a variable number of times).

A group of g nodes desirous of discovering a shared group secret just need to exchange their IDs (for pairwise secrets $g = 2$). From their IDs, the nodes can discover shared secrets and their corresponding hash depths in the nodes by application of the public function $f(.)$. If for instance, if two nodes A and B share q keys with indexes $i_1 \cdots i_q$ and the hash depth of the q keys in nodes A and B are $a_1 \cdots a_q$, and $b_1 \cdots b_q$, respectively, and if $e_j = \max(a_j, b_j), 1 \leq j \leq q$, the shared secret K_{AB} between the nodes A and B, is a function of $[K_{i_1}^{e_1} \parallel \cdots \parallel K_{i_q}^{e_q}]$. Similarly a shared group secret between g nodes would depend on the keys common to all g nodes, and their respective *maximum* hash depths.

An attacker, by exposing secrets from many nodes, can discover the shared secret K_{AB} if he is able to discover the secrets $[K_{i_1}^{d_1} \cdots \parallel K_{i_q}^{d_q}]$, where $d_j \leq e_j, 1 \leq j \leq q$ (the attacker can discover $K_{i_j}^{e_j}$ from $K_{i_j}^{d_j}$ by hashing $K_{i_j}^{d_j}$ repeatedly, $d_j - e_j$ times).

B.1 Analysis of HARPS

We shall assume that an attacker (Oscar) can expose only a fraction $0 \leq \rho \leq 1$ of secrets by tampering with a node (see for instance "partially openable chips" in [10]). The analysis however, does not preclude the possibility of $\rho = 1$. Oscar, by tampering with many (say m) nodes $\mathfrak{O}_m$ will be able to expose some secrets buried in them, which could potentially be used to "compromise the system". By compromising a fraction ρ of keys from m nodes Oscar accumulates a collection $\mathbb{O}$ of ρkm secrets. However, in general, not all secrets in $\mathbb{O}$ are unique.

The analysis of HARPS involves determining

1. $p_e(m, g, \rho)$ - the probability of eavesdropping, or the probability that Oscar, using $\mathbb{O}$, can discover the secret shared by g nodes in the set $\mathfrak{G}_g$, where $\mathfrak{O}_m \cap \mathfrak{G}_g = \emptyset$.
2. $p_s(m, \rho)$ - the probability that Oscar, using $\mathbb{O}$, can synthesize a node (or discover all secrets stored in a specific node) $A \notin \mathfrak{O}_m$.

It can be easily shown that

$$p_e(m, g, \rho) = (1 - \varepsilon(m, g, \rho))^{(\frac{k}{\xi})}, \tag{3}$$

$$p_s(m, \rho) = p_e(m, g = 1, \rho), \tag{4}$$

where

$$\varepsilon(m, g, \rho) = \sum_{u=0}^{n} \xi^g (\rho\xi)^u (1 - \rho\xi)^{m-u} Q_L^g(u)$$

$$Q_L^g(u) = \sum_{i=1}^{L} \frac{i^g - (i-1)^g}{L^g} \left(\frac{L-i}{L}\right)^u.$$

TABLE I

RESISTANCE OF HARPS TO *eavesdropping* ATTACK IN TERMS OF THE NUMBER OF NODES AN ATTACKER NEEDS TO TAMPER WITH TO ENGINEER AN ATTACK WITH PROBABILITIES $p_e = 0.5$, $p_e = 10^{-6}$ AND $p_e = 10^{-20}$ RESPECTIVELY FOR $\rho = 0.1, 0.05$.

ξ	$\rho = 0.1$			$\rho = 0.05$		
	p_e			p_e		
	0.5	10^{-6}	10^{-20}	0.5	10^{-6}	10^{-20}
0.050	2920	584	151	5841	1169	302
0.075	2400	505	190	4800	1014	382

TABLE II

RESISTANCE OF HARPS TO *synthesis* ATTACK, IN TERMS OF m, THE EXPECTED NUMBER OF NODES AN ATTACKER NEEDS TO TAMPER WITH SYNTHESIZE *any one* OF THE m NODES, FOR $\rho = 0.2, 0.1$ AND 0.05.

ξ	$\rho = 0.2$	$\rho = 0.1$	$\rho = 0.05$
0.050	93,000	181,000	353,000
0.075	52,200	101,000	196,000

Eq (4) readily follows from Eq (3) by noting the fact that discovery of *all* secrets in a node is equivalent to discovering the secrets a node shares with *itself*!

B.2 Resistance to Eavesdropping Attack

From Eq (3) it is easy to estimate the number of nodes m Oscar needs to tamper with in order to discover the shared secret between two arbitrary nodes (or $g = 2$) with some probability p_e, for a given set of parameters P, k, L of HARPS, and the "assurance" $\rho \leq 1$ provided by technology for read-proofing (if no such assurance is provided, $\rho = 1$).

Table I depicts the calculated value of m for $p_e = 0.5$, $p_e = 10^{-6}$ and $p_e = 10^{-20}$, for the cases of $\rho = 0.2, 0.1$ and 0.05, for two values of $\xi = \frac{k}{P}$ - 0.05 and 0.075. The parameters k and L have been fixed at $k = 1500$, $L = 512$. It is easy to see the linear dependence of m and $\frac{1}{\rho}$ - which is intuitive. In general

$$p_e(m, \rho_1, g) \approx p_e(m\frac{\rho_1}{\rho_2}, \rho_2, g). \tag{5}$$

As ξ is reduced, the rate of deterioration of security reduces - which is also intuitive - as keys in one a node provide less information about keys in *other* nodes.

B.3 Resistance to Node Synthesis

In [9] we showed that while random KPD schemes in general exhibit a higher resistance to node synthesis (compared to other KPDs), HARPS [1] in particular, is about two *orders of magnitude* better than other *random* KPD schemes in this respect. It was also indicated in [9] that this useful property could be leveraged for realization of long-lived security of HARPS, through periodic renewal.

From Eq (4), we can easily estimate the number of nodes (m) Oscar needs to tamper with to achieve a desired probability of

node synthesis p_s. With access to m nodes, Oscar can *expect* to successfully synthesize **one** node if $m \approx \frac{1}{p_s}$ (or if he is able to achieve a synthesis probability of $p_s = \frac{1}{m}$). Table 2 illustrates the number of nodes Oscar needs to tamper with in order to successfully engineer the synthesis attack for $\xi = 0.05$, and 0.075 and $\rho = 0.2, 0.1$ and 0.05. Once again, k and L are fixed at 1500 and 512 respectively.

III. Safe Renewal

While initial pre-loading of keys can be performed by *physical contact* [27] we cannot expect the owners / operators of devices to take them back to the manufacturer (parent, or TA) every once in a while for key renewal. Therefore practicality constraints dictate that renewal has to occur over open communication channels (say over the Internet). Keys can be renewed periodically by direct interaction with the TA, for which nodes have to authenticate themselves to the TA using all their *current* secrets, in order to receive a new set of secrets. Obviously, if Oscar has managed to expose all secrets in a node, he can also take part in subsequent key renewals - which raises the following questions:

1. How does key renewal help? Or is "safe renewal" possible over open channels?
2. What guarantees (provided by technology for read-proofing / tamper-resistance) are required in order to realize safe renewal?

In this section we consider the effect of three "reasonable" assumptions to cater for safe renewal:

A0 Keys can be exposed only by tampering with nodes (cryptanalytic attacks are infeasible). Further, nodes that are tampered with are rendered unusable in future.

A1 Assurance of (only) "partially open-able chips" (or $\rho < 1$).

A2 Existence of a unique (possibly weak) secret for every node that *cannot* be exposed by tampering.

In particular, we shall consider two cases.

1. Case 1: Assumptions A0 and A1 hold (and A2 does not) - or A0↑A1↑A2↓
2. Case 2: Assumptions A0 and A2 hold (and A1 does not) - or A0↑A1↓A2↑

Apart from the possible guarantees A0, A1 and A2 provided by technology, we shall also investigate the effect of

1. S1: the use of a pass-phrase, (say V_A for node A) in addition to all KPD secrets
2. S2: the use of a unique (strong) secret, say U_A for node A (known only to the node and the TA), and
3. S3: rest encryption [12]

on the security of key renewal.

A. Realization of Assurances

A.1 Assurance A1 - Partially Openable Chips

The assurance A1 is a guarantee that only a *fraction* of the preloaded secrets can be "read" even by a sophisticated attacker, by tampering with a chip. This could perhaps be provided by sensors in chips which recognize intrusive attempts, and erase all secrets.

For example, if we have s such sensors, and each sensor could successfully detect intrusion with a probability ν, then the probability that the attacker would be able to circumvent *all* sensors is $(1-\nu)^s$. It may be reasonable to assume that the attacker may succeed in circumventing one or two sensors, but not more. This assumption (presence of active sensors) may imply a need for a built in source of power (for the sensors to be activated) inside the chip. However, it is not entirely inconceivable that we could do *without* an energy source - for instance the residual potential differences (between 1s and 0s) in stored media could possibly be used to power the process of erasing bits. Another possibility is the use of MEMS for this purpose.

A.2 DOWN (Decrypt Only When Necessary) Policy

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, while 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.

Thus 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 [28]. A single key K_V used for encrypting (and decrypting) the KPD keys could itself be stored in volatile RAM. Whenever some intrusion is sensed, as long as the key K_V is erased, just one of the KPD keys which is 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 volatile memory) 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. For $k = 1000$ this translates to $\rho = \frac{1}{k} = 0.001$).

A.3 Assurance A2 - Delay-based Circuit Authentication

At first sight, it might appear that if the password (V_A for node A), shared by the owner of the node and the TA / parent node, is sufficient for providing assurance A2 - after all, V_A is stored in the owner's head, and therefore *cannot be exposed by tampering* with node A. However, for trusted devices, the intention is to ensure that even the owner of the device cannot determine the secrets buried in her device. More specifically, even if the owner has exposed all key-ring secrets in her device, she should still not be able to take part in *subsequent key renewals*. So what we need is a secret *not* known to the owner, but which the node has access to - but however, cannot be exposed by tampering with the node.

A possible solution to this problem is the use of Physical Unclonable Functions (PUF) [11], which employ delay-based cir-

cuit authentication. The TA (or the parent node or the manufacturer of the node), would be able to identify a node based on immeasurable (and uncontrollable) hardware delays in the circuits of any fabricated chip - which results in a unclonable hash function in every chip. The manufacturer could store some challenge-response pairs of the PUF. This responses to specific challenges could be used as secrets known only to the manufacturer.

However, the delays may be sensitive to environmental changes. Gassend et al [11] further argue that this could be easily overcome by using error correction codes. In practice, this may imply that the secret generated from this phenomenon may not have a very high entropy - for example it may provide an equivalent of say 128 bit security for a particular range of operating temperatures and only say 32 bit security for a larger range of operating temperatures. So we shall make a pessimistic assumption that this additional secret (say W_A for node A) is "weak" - or susceptible to practical brute-force attacks.

B. Safe Renewal with Assurances

B.1 Case 1: A0↑A1↑A2↓

If the assurance A1 holds, only a fraction ρ of the desired keys can be 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 estimates for m (the number of nodes Oscar needs to tamper with for node synthesis) in Table 2 are based on the assumption that *all* keys are exposed by tampering with *other* nodes. The revised estimate m_r can be easily obtained by adjusting the estimates of m as $m_r = (1-\rho)m + 1$. However, for small ρ, $m_r \approx m$.

The reported value of m can be easily extrapolated to other values of ρ. For instance if $m = 100,000$ when $\rho = 0.1$, then 90,000 nodes need to be tampered with for synthesizing a node. For $\rho = 0.2$, m reduces to 50,000. In this case, roughly $(1 - 0.2) \times 50,000 = 40,000$ nodes need to be tampered with when only assurance of partial tamper resistance (A1) holds.

If knowledge of the additional pass-phrase V_i is required for renewal of keys in node i, then things are even more difficult for Oscar. Out of the m_r nodes that he has access to, he may know the renewal password only for m_p nodes. In practice it may be much more difficult for Oscar to accumulate the m_p nodes (for each of which he would need the co-operation of the owners) than the other $m_r - m_p$ nodes (which could even be "stolen" nodes). The figures of $m_r = 90,000$ for $\rho = 1$ assumes that it is sufficient for Oscar to synthesize *any* of the 90,000 nodes (or achieve synthesis probability of $p_s = \frac{1}{m_r}$. However, with the additional password requirement, he is forced to synthesize one of the $m_p << m_r$ nodes (or achieve $p_s \approx \frac{1}{m_p}$). For example, if $m_p = 200$, he may need $m_r > 150,000$ for $\rho = 0.1$.

If in addition to the key ring and the pass-phrase, a unique key (U_i for node i) stored in the nodes is also needed for renewal, then, Oscar will have to expose the unique keys too from the m_p nodes. If, for example, Oscar is able to successfully expose the unique key only from a fraction ρ' of the nodes, or from $m'_p = \rho' m_p$ nodes, then Oscar is forced to synthesize one of the m'_p nodes (or achieve a synthesis probability of $p_s = \frac{1}{m'_p}$). If (for example) $m'_p = 20$, Oscar needs to tamper with roughly $m_r = 230,000$ nodes!

With rest-encryption [12], as the name indicates, the stored secrets are encrypted for rest (before the node is powered off). For instance, before powering off, the set of preloaded keys may be divided into two sets L_0 and R_0, or $\mathbb{S}_A = [L_0 \parallel R_0]$, followed by two Fiestel-like encryption rounds,

$$L_1 = R_0, R_1 = E_{R_0}(L_0) \text{ and } L_2 = R_1, R_2 = E_{R_1}(L_1),$$

which can be easily reversed when the node is powered on. However, even if T bits of $[L_2 \parallel R_2]$ are *not* recoverable after tampering, the complexity involved in recovery of $\mathbb{S}_A$ by an attacker is equivalent to that of breaking a T-bit cipher.

Thus a combination of assurance A1 (partially open-able chips) along with rest-encryption (S3) can guarantee that *no* secrets can be exposed. However, it may be possible to partially circumvent rest-encryption during transition from "in-use" to "rest" state[5]. The property of high resistance to node synthesis would still be useful in such an event.

The encrypted keys $[L_2 \parallel R_2]$ could further be encrypted using keys derived from biometric signals and / or a password. This could prevent the attacker from gainfully using sacrificial nodes to discover secrets (which would thus severely limit Oscar's ability increase m_r)

B.2 Case 2: A0↑A1↓A2↑

In this case, assurance A2 guarantees a weak secret in every node, that cannot be exposed by tampering with the node. Let,
1. $\mathbb{S}_A^0$ represent the keys initially stored in node A.
2. $\mathbb{S}_A^i$, the keys stored in node A after the i^{th}renewal.
3. W_A, the weak additional authentication key (which cannot be exposed by tampering with A).
4. $h()$, a secure cryptographic hash function, and
5. R_{i+1} a random *strong* secret generated by the TA during the process of the $i+1^{\text{th}}$renewal.

The interaction between the node A and the TA for the next $(i+1)^{\text{th}}$round of renewal could proceed as follows:

$$\begin{array}{lll}
A \rightarrow \text{TA}: & M_1 = E_{K_1^{i+1}}(ID_A) & \left\{K_1^{i+1} = h(\mathbb{S}_A^i)\right\} \\
\text{TA} \rightarrow \text{A}: & M_2 = E_{K_2^{i+1}}(\tilde{\mathbb{S}}_A^{i+1}) & \left\{K_2^{i+1} = h(\mathbb{S}_A^i \parallel W_A)\right\} \\
 & & \left\{\tilde{\mathbb{S}}_A^{i+1} = E_{R_{i+1}}(\mathbb{S}_A^{i+1})\right\} \\
A \rightarrow \text{TA}: & M_3 = E_{K_3^{i+1}}(ID_A) & \left\{K_3^{i+1} = h(\mathbb{S}_A^i \parallel \tilde{\mathbb{S}}_A^{i+1})\right\} \\
\text{TA} \rightarrow \text{A}: & M_4 = E_{K_3^{i+1}}(R_{i+1}) &
\end{array}$$

If Oscar had exposed $\mathbb{S}_A^i$ by *tampering with other nodes* (or by performing synthesis attack), he can still take part in the renewal process. From $M_2 = E_{K_2^{i+1}}(\tilde{\mathbb{S}}_A^{i+1})$ and $M_3 = E_{K_3^{i+1}}(ID_A)$, Oscar could brute-force all candidate W_As. If successful, he has everything needed to synthesize node A (both W_A and $\mathbb{S}_A^{i+1}$).

However if node synthesis is prohibitively expensive, Oscar would need to expose secrets $\mathbb{S}_A^i$ by *tampering* with device A,

[5]Once again, perhaps by sudden freezing of the chip before it has the chance to complete the shut-down sequence.

TABLE III

NUMBER OF NODES m AN ATTACKER NEEDS TO TAMPER WITH IN ORDER TO SYNTHESIZE A NODE THAT CAN TAKE PART IN SUBSEQUENT KEY RENEWALS UNDER VARIOUS ASSURANCES (A0, A1 AND A2) AND ADDITIONAL SECURITY MEASURES (S1, S2, S3). AN UPWARD FACING ARROW FOLLOWING AN ASSUMPTION INDICATES THAT THE ASSUMPTION IS VALID (DOWNWARD FACING ARROW - INVALID, AND UP-DOWN ARROW - IRRELEVANT).

	Assumptions	m
1.	A0↑A1↑A2↓S1↓S2↓S3↓	90,000
2.	A0↑A1↑A2↓S1↑S2↓S3↓	150,000
3.	A0↑A1↑A2↓S1↑S2↑S3↓	230,000
4.	A0↑A1↑A2↓S1↕S2↕S3↑	∞
5.	A0↑A1↓A2↑S1↑S2↓S3↓	14,500
6.	A0↑A1↓A2↑S1↕S2↑S3↓	∞
7.	A0↑A1↑A2↑S1↓S2↓S3↓	100,000
8.	A0↑A1↑A2↑S1↑S2↓S3↓	165,000
9.	A0↑A1↑A2↑S1↑S2↑S3↓	250,000

in which process, he would have destroyed the device A, and therefore would not be able to generate a proper response M_3. As a result Oscar would have no way to determine W_A. The only way is to "try" out different candidate W_As, obtain the corresponding estimates of M_2 and M_3. However a wrong estimate of W_A would result in the failure of authentication of the corresponding M_2 by the TA (so the TA would not respond with M_4). So each "brute-forcing" attempt needs the *involvement of the TA*. If the attacker tries too many times the TA would "smell a rat" and refuse to honor subsequent attempts by node A.

Assurance A2 thus has the ability of inhibiting Oscar from tampering with *the node that he desires to synthesize* - or he has to engineer a pure synthesis attack to be successful. The number of nodes Oscar needs to compromise in this case could again be derived by adjusting the estimates in Table 2. In Table 2 the estimates are for $\rho = 0.1$. However if assumption A1 does not hold, then the estimates have to be adjusted for $\rho = 1$.

Under this scenario, Oscar may need to tamper with the order of about 36,000 nodes (expose *all* secrets buried in 36,000 nodes) in order to synthesize a *specific* node (which Oscar cannot tamper with). However, with access to m nodes, Oscar may not need to synthesize a *specific* node. He can divide the set of nodes he has into two groups of m_p and m_r nodes ($m_p + m_r = m$). He tampers with the group of m_r "sacrificial" nodes to reveal secrets, and uses the revealed secrets to compromise all keys belonging to one of the m_p nodes (he cannot tamper with any of the m_p nodes as he has to engage them in an exchange with the TA to determine the weak secret based on circuit delays).

For the first group of m_r nodes even "stolen" nodes would suffice. But the second group of m_p nodes *cannot* include stolen nodes if the additional pass-phrase V_i is used for renewal. For example, with $m_p = 200$ (for $k = 1500, \xi_2 = 0.075, L = 512$), Oscar needs roughly $m_r = 14,500$ to achieve his goal. Of course, other combinations of m_p and m_r are also possible.

However, if the additional *unique* strong secret (U_i for node i) is also used for renewal, then the attacker faces a dilemma. If he has to tamper with node A for exposing U_A then there is no way for him to determine W_A. Simultaneously, there is no way for Oscar to determine U_A by tampering with *other* nodes! Thus with the assurance A2 and the use of the additional unique secret U_i, safe renewal is assured irrespective of the number of nodes that Oscar can compromise! In other words "a prohibitively high resistance to node synthesis" is *not mandatory* in this case.

In practice, both forms of assurances (partial tamper resistance and additional weak key for authentication) may be present. The results reported in Table 2 are based on the assumption that in order to synthesize a node, the necessary keys are obtained by tampering with *other* nodes, *and* only a fraction of keys can be compromised from each node. In other words, the results in Table 2 are for the case when *both* assurances A1 and A2 hold.

For convenience the expected number of nodes that Oscar needs to tamper with under different assurances and different security policies for renewal, are tabulated in Table 3.

A practical deployment should take proactive steps to ensure that it is extremely difficult for an attacker to compromise secrets from nodes, even for purposes of accomplishing the eavesdropping attack. Note the striking disparity between the effort involved for an attacker to eavesdrop on nodes (Table 1) and the effort needed for participating in subsequent key renewals (Table 3). As long as the effort needed for the latter is prohibitively expensive the motivation for an attacker to engineer eavesdropping attack is substantially reduced - all his efforts are wasted after the next round of key updates.

IV. CONCLUSIONS

In this paper, we have investigated some resonable assurances from technology for tamper resistance, and how, using those assurances along with some security policies or practices could substantially improve the security of random KPD schemes. While reliance on tamper resistance has been and perhaps will continue to be a controversial issue in the cryptographic community [29] - [31], there is no denying the fact that it is indeed mandatory in evolving application scenarios where trusted devices are called for. Fueled by this necessity, we can expect technology to rise up to the challenges.

While it is primarily the property of high resistance to node synthesis that makes HARPS a particularly attractive option for securing large scale deployments (especially when resources are limited), HARPS has many other advantages [1]. For instance, it offers simple extensions to a tree-hierarchical deployment, which could keep the size of each branch small enough so that key leaks can be traced more easily, and the resulting damage, contained. In addition, the *same* set of preloaded keys used for establishing shared secrets between nodes can also be used for discovery of instantaneous group secrets (or conference secrets), and non-instantaneous group secrets through the use of broad-

cast encryption [32].

Unlike most tree-based broadcast encryption schemes where the source of the broadcast is assumed to be at the root of the tree, broadcast encryption with HARPS also permits *any* node to be the source[6]. Broadcast encryption by the TA (or parent node in hierarchical deployments) can also be very useful for revocation of nodes from the deployment. Unlike revocation mechanism in PKI where nodes may need to store revocation lists, if broadcast encryption is used for revocation, only the latest revocation secret (which will not be available to revoked nodes) need to be stored.

In addition, HARPS also caters for a more efficient mechanism for broadcast authentication [33] (by appending key based MACs). The flexibility offered by the ability to choose the hash depths used for MACs renders HARPS significantly more efficient than schemes that do not employ hashing of distributed keys. Further, this flexibility also caters for a novel cryptographic paradigm of broadcast authentication with "preferred" verifiers.

The versatility of HARPS, combined with its tremendous resistance to node synthesis could be very useful as an enabler for a low complexity[7] security infrastructure or a Key Predistribution Infrastructure (KPI) [34], that provides all the basic functionality of PKI in addition to primitives for efficient realization of multicast security.

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.

[2] M. Kwiatkowska, V. Sassone, "Science for Global Ubiquitous Computing," Grand Challenges in Computing (Research), edited by T. Hoare and R. Milner, 2004.

[3] J. Crowcroft, "Scalable and Ubiquitous Computing Systems," Grand Challenges in Computing (Research), edited by T. Hoare and R. Milner, 2004.

[4] J. Lotspiech, S. Nusser, F. Pestonoi, "Anonymous Trust: Digital Rights Management using Broadcast Encryption," Proceedings of the IEEE, **92** (6), pp 898–909, 2004.

[5] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, T. Rabin, "Tamper Proof Security: Theoretical Foundations for Security Against Hardware Tampering," Theory of Cryptography Conference, Cambridge, MA, February 2004.

[6] B. C. Neuman, T. Ts'o, "Kerberos: An Authentication Service for Computer Networks", IEEE Communications, **32**(9), pp 33–38. September 1994.

[7] R. Needham and M. Schroeder, "Using encryption for authentication in large networks of computers," Communications of the ACM, 21(12), December 1978.

[8] S.Kiran, P. Lareau, S.Lloyd, "PKI Basics - A Technical Perspective," PKI-Forum (http://www.pkiforum.org), November 2002.

[9] M. Ramkumar, N. Memon, "On the Security of Random Key Predistribution Schemes," 5th Annual IEEE Information Assurance Workshop, West Point, NY, June 2004.

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

[11] B. Gassend, D. Clarke, M. van Dijk, S. Devadas, "Delay-based Circuit Authentication and Applications," Proceedings of the 2003 ACM symposium on Applied computing, Melbourne, Florida, pp 294 – 301, 2003.

[6]The capability of broadcast encryption by peers could serve as a foundation for the security of publish-subscribe systems

[7]HARPS needs only a single symmetric cryptographic primitive - a hash function or a block cipher - for its realization.

[12] Application Security Inc., "Encryption of Data at Rest: White Paper," available at http://www.appsecinc.com.

[13] 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.

[14] 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.

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

[16] 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.

[17] L. Gong, D.J. Wheeler, "A Matrix Key Distribution Scheme," *Journal of Cryptology*, **2**(2), pp 51-59, 1990.

[18] C.J. Mitchell, F.C. Piper, "Key Storage in Secure Networks," *Discrete Applied Mathematics*, **21** pp 215–228, 1995.

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

[20] P. Erdos, P. Frankl, Z. Furedi, "Families of Finite Sets in which no Set is Covered by the union of 2 Others," *Journal of Combinatorial Theory, Series A*, **33**, pp 158–166, 1982.

[21] P. Erdos, P. Frankl, Z. Furedi, "Families of Finite Sets in which no Set is Covered by the Union of r Others," *Isreal Journal of Mathematics*, **51**, pp 79–89, 1985.

[22] L. Eschenauer, V.D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," Proceedings of the Ninth ACM Conference on Computer and Communications Security, Washington DC, pp 41-47, Nov 2002.

[23] H. Chan, A. Perrig, D. Song, "Random Key Pre-distribution Schemes for Sensor Networks," IEEE Symposium on Security and Privacy, Berkeley, California, May 2003.

[24] D. Liu, P.Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," Proceedings of the 10th ACM Conference on Computer and Communication Security, Washington DC, 2003.

[25] R. Di Pietro, L. V. Mancini, A. Mei, "Random Key Assignment for Secure Wireless Sensor Networks," 2003 ACM Workshop on Security of Ad Hoc and Sensor Networks, October 2003.

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

[27] F. Stajano, R. Anderson. "The Resurrecting Duckling: Security Issues in Ad-Hoc Wireless Networks." In "Security Protocols, 7th International Workshop Proceedings", Lecture Notes in Computer Science. Springer-Verlag, 1999.

[28] M. Ramkumar, "DOWN with Trusted Devices," submitted to New Security Paradigms Workshop (NSPW-2005), Lake Arrowhead, CA, Sep 2005.

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

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

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

[32] M. Ramkumar, N. Memon, "A DRM Based on Renewable Broadcast Encryption," to appear, Visual Communications and Image Processing (VCIP) 2005, July 2005, Beijing, China.

[33] M. Ramkumar, "Broadcast Authentication with Preferred Verifiers," submitted to Crypto 2005.

[34] M. Ramkumar, N. Memon, "KPI: A Security Infrastructure for Trusted Devices," Pre-Conference Wireless and Mobile Security Workshop, The 12th Annual Network and Distributed System Security Symposium, San Diego, CA, Feb 2005.